IBM® WebSphere®

WebSphere Application Server V8.5 Concepts, Planning, and Design Guide

- Highlights end-to-end planning for WebSphere implementations
- Defines WebSphere concepts and best practices
- Addresses distributed and z/OS platforms

Jan Bajerski
Davide Barillari
Libor Cada
Susan Hanson
Guo Liang Huang
Rispna Jain
Shishir Narain
Jennifer Ricciuti
Christian Steege

Redbooks

ibm.com/redbooks

International Technical Support Organization

WebSphere Application Server V8.5 Concepts, Planning, and Design Guide

June 2012

SG24-8022-00

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

First Edition (June 2012)

This edition applies to Version 8.5 of IBM WebSphere Application Server.

Contact an IBM Software Services Sales Specialist

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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX®
alphaWorks®
BladeCenter®
Build Forge®
CICS®
ClearCase MultiSite®
ClearCase®
ClearQuest®
CloudBurst®
Cognos®
DataPower®
DB2®
developerWorks®
Domino®
Global Technology Services®
GPFS™
IBM SmartCloud™
IBM®
IMS™
InfoSphere®
Jazz™
Language Environment®
Lotus®
MVS™
Parallel Sysplex®
Passport Advantage®
PowerHA®
PR/SM™
Processor Resource/Systems Manager™
pureQuery™
pureScale®
pureXML®
RACF®
Rational Rose®
Rational Team Concert™
Rational®
Redbooks®
Redbooks (logo) ®
RequisitePro®
Resource Measurement Facility™
RMF™
System i®
System z®
Tivoli®
VTAM®
WebSphere®
z/OS®
z/VM®
zEnterprise®
zSeries®

The following terms are trademarks of other companies:

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

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

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

Preface

This IBM® Redbooks® publication provides information about the concepts, planning, and design of IBM WebSphere® Application Server V8.5 environments. The target audience of this book is IT architects and consultants who want more information about the planning and design of application-serving environments, from small to large, and complex implementations.

This book addresses the packaging and features in WebSphere Application Server V8.5, and highlights the most common implementation topologies. It provides information about planning for specific tasks and components that conform to the WebSphere Application Server environment.

Also in this book are planning guidelines for Websphere Application Server V8.5 and Websphere Application Server Network Deployment V8.5 on distributed platforms. It also includes guidelines for WebSphere Application Server for IBM z/OS® V8.5. This book contains information about migration considerations when moving from previous releases.

The team who wrote this book

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

Figure 1 Left to right: Jan, Libor, Jennifer, Shishir, Christian, Susan, Margaret, Leo, Rispna, and Davide

Jan Bajerski is WebSphere Connectivity IT Specialist in Software Group Community of Practice in CEE region, and has been working in the IT industry for 11 years. Previously he worked in IBM Software Services for WebSphere in Poland, supporting customers in implementing solution by using WebSphere Application Server, WebSphere MQ, and WebSphere Message Broker. He has a BSc degree from Warsaw University of Technology (Poland).

Davide Barillari is a Certified IT Specialist working for IBM Global Technology Services® in Italy. He joined IBM in 1996 and worked for three years with IBM Education as a z/OS instructor. He has 12 years of experience in the IBM Technical Support, with a deep knowledge of IBM zSeries® architecture and distributed environments. His main areas of expertise are infrastructure design, implementation, maintenance, and debugging of the WebSphere environment. Davide is an IBM Certified Solution Developer as well an IBM Certified System Administrator for WebSphere Application Server, WebSphere Process Server, and SOA Solutions. Since 2011, he has been a Certified Solution Architect for Cloud Computing, is accredited at the Senior level in the Product Services Profession. He is also Certified as Level 1 experienced IT Specialist by IBM Profession Office AITS. Davide is currently providing consulting services at customer sites in the banking sector on WebSphere Application Server for z/OS.

Libor Cada is an IT Specialist working in Integrated Delivery Center SSO, in Brno, Czech Republic. He has eight years of experience in the IT and banking industries on mainframe IBM System z® and Linux on System z environments. He previously held the position of z/OS database and data communication (DB/DC) systems programmer for IBM CICS®, IBM DB2®, WebSphere MQ, and IBM IMS™ products. He currently supports clients from multiple geographies in his role of WebSphere Application Server and z/OS certified System Programmer.

Susan Hanson is a member of the WebSphere Application Server foundation development team. She has 22 years of experience in developing and delivering IBM software products across the WebSphere and IBM Tivoli® brands. Her current focus products are WebSphere Application Server, WebSphere Virtual Enterprise, and WebSphere eXtreme Scale. Her focus areas are in release management, project management, and development process transformation. She also works part-time in the ITSO Redbooks organization as a project leader focused on the Growth Market Unit (GMU) areas. She is also part of the ITSO strategy team focused on enabling Industries and GMU. She holds a Bachelor's degree in Computer Science from East Carolina University and a Master's degree in Computer Information Systems from The University of Phoenix. She is based in Research Triangle Park, North Carolina and is temporarily working and residing in Shanghai, China.

Guo Liang Huang is an experienced technical support engineer working for IBM WebSphere AIM group in the China Lab. He has five years of expertise in supporting IBM WebSphere Process Server. He has over 11 years of experience in developing, testing, and supporting software products. Guoliang holds Bachelor's degrees in Computer Science from the Central South University of China. Guoliang also has expertise in SOA.

Rispna Jain is a Technical Software Deployment Manager for the WebSphere suite of products in IBM Global Technology Services, and works with clients in North America. She has seven years of experience on WebSphere Application Server product development at IBM Software Group in various roles such as development, Level 3 support, and test. Rispna has also been a technical speaker for WebSphere Application Server related topics at various WebSphere conferences. She is an IBM Certified SOA associate and holds a Master of Technology degree in Computer Science.

Shishir Narain is an Open Group certified Master IT Specialist with deep skills in IBM middleware products. He works in IBM Software Services for WebSphere at India Software Lab, Gurgaon. He has 13 years of experience in developing solutions for multiple clients. He has led several end-to-end IT implementations based on SOA. He holds a Master of Technology degree from Indian Institute of Technology, Kanpur.

Jennifer Ricciuti is a Course Developer and Instructor in WebSphere Education. She has 15 years of experience in developing and delivering education courses on various WebSphere products, including WebSphere Application Server, WebSphere Process Server, WebSphere

eXtreme Scale, and IBM Business Process Manager Advanced. Her areas of expertise include course design, development, and delivery. She holds a Bachelor's degree in Computer Science from Point Park University. She works and resides in Pittsburgh, Pennsylvania.

Christian Steege is an IT Architect within IBM Software Group Services for WebSphere in Zurich, Switzerland. He has more than 10 years experience in designing infrastructures and applications for IBM WebSphere Application Server, IBM WebSphere Business Process Management, IBM WebSphere Portal, and IBM WebSphere MQ. He implemented many of these infrastructures and applications at variety of Swiss IBM Customers. Christian holds a Master's degree in Information Management from the University of St. Gallen, Switzerland.

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

Margaret Ticknor
Carla Sadtler
Deana Coble
Tamikia Lee
Linda Robinson
Stephen Smith
Debbie Willmschen
International Technical Support Organization, Raleigh Center

Erik Altman
Donald C. Bagwell
Soloman J Barghouthi
Michael Cheng
Eric M Covener
Dana Duffield
David Follis
Jeremy Hughes
Chunlong Liang
Jeff Mierzejewski
Bill O'Donnell
Gary Picher
Brain Pulito
Sajan Sankaran
Keith B Smith
Christopher Vignola
IBM US

Alasdair Nottingham
IBM UK

Yee-Kang Chang
King Lam
Ilene Seelemann
Sam Wong
Felix Wong
IBM Canada

Lohitashwa Thyagaraj
IBM India

Thanks as well to the teams who wrote these books:

- *WebSphere Application Server V6.1: Planning and Design*, SG24-7305
- *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708
- *IBM WebSphere Application Server V8 Concepts, Planning, and Design Guide*, SG24-7957

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our 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 Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

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

Stay connected to IBM Redbooks publications

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction to WebSphere Application Server V8.5

IBM WebSphere Application Server is the leading software foundation for service-oriented architecture (SOA) applications and services for your enterprise. With IBM WebSphere Application Server, you can build business-critical enterprise applications and solutions, and combine them with innovative new functions. The WebSphere Application Server family includes and supports a range of products that helps you develop and serve your business applications. You can use these products to build, deploy, and manage dynamic websites and other more complex solutions productively and effectively.

This chapter introduces WebSphere Application Server V8.5 for distributed platforms and z/OS, and highlights other IBM software products that are related to WebSphere Application Server.

This chapter includes the following sections:

- Application server infrastructure
- Evolving Java application development standards
- Comprehensive programming model support
- Enhanced management capabilities
- Operational efficiency and intelligent management
- Security management
- Simplified interoperability
- Advanced tools and extensions
- Related products
- New features and capabilities in WebSphere Application Server V8.5

1.1 Application server infrastructure

WebSphere Application Server provides the environment to run your solutions and to integrate them with every platform and system. The core component in WebSphere Application Server is the application server runtime environment. An application server provides the infrastructure for executing the applications that run your business. It insulates the infrastructure from the hardware, operating system, and network (Figure 1-1).

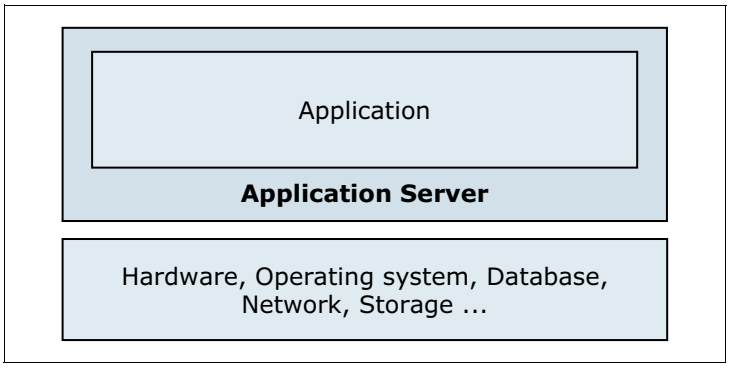


Figure 1-1 Basic presentation of an application server and its environment

An application server provides a set of services that business applications can use, and serves as a platform to develop and deploy these applications. The application server acts as middleware between back-end systems and clients. It provides a programming model, an infrastructure framework, and a set of standards for a consistent designed link between them. As business needs evolve, new technology standards become available. Since 1998, WebSphere Application Server has grown and adapted itself to new technologies and to new standards. It provides an innovative and cutting-edge environment so that you can design fully integrated solutions and run your business applications.

WebSphere Application Server is a key SOA building block, providing the role of the business application services (circled in Figure 1-2) in the SOA reference architecture.

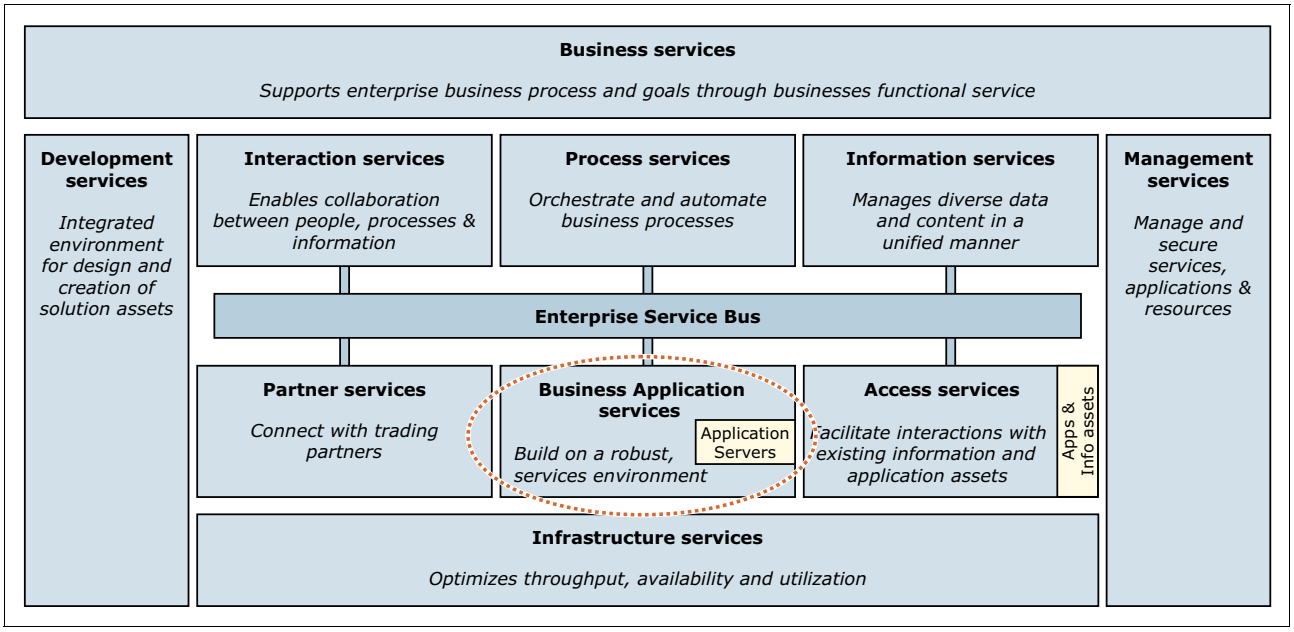


Figure 1-2 Position of business application services in an SOA reference architecture

From an SOA perspective, you can perform the following functions with WebSphere Application Server:

- Build and deploy reusable application services quickly and easily
- Run services in a secure, scalable, highly available environment
- Connect software assets and extend their reach
- Manage applications effortlessly
- Grow as your needs evolve, reusing core skills and assets

WebSphere Application Server is available on a range of platforms and in multiple packages to meet specific business needs. By providing an application server to run specific applications, it also serves as the base for other WebSphere products and many other IBM software products.

The packaging options available for WebSphere Application Server provide a level of application server capabilities to meet the requirements of various application scenarios. Although these options 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.

The following packages are available:

- WebSphere Application Server—Express V8.5
- WebSphere Application Server—Base V8.5
- WebSphere Application Server for Developers V8.5
- WebSphere Application Server Network Deployment V8.5
- WebSphere Application Server for z/OS V8.5

Figure 1-3 summarizes various WebSphere Application Server packaging options.

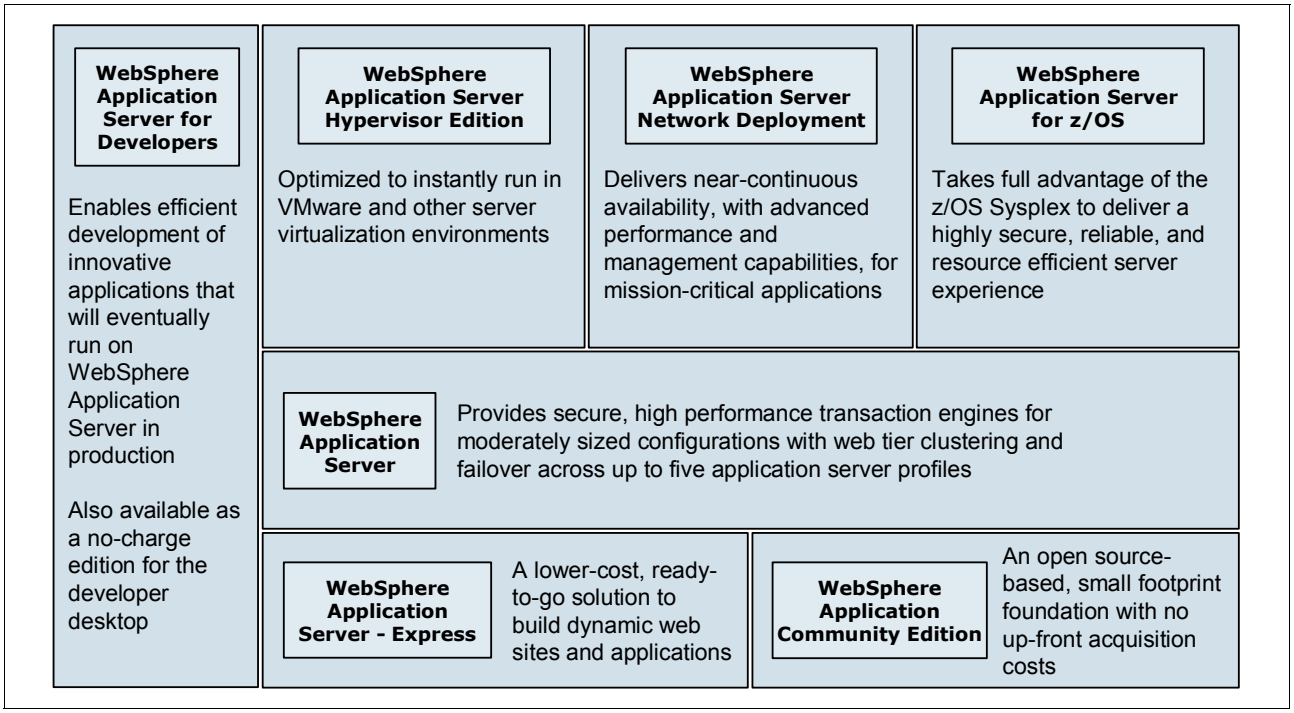


Figure 1-3 WebSphere Application Server editions

Figure 1-4 summarizes the main components that are included in each WebSphere Application Server package.

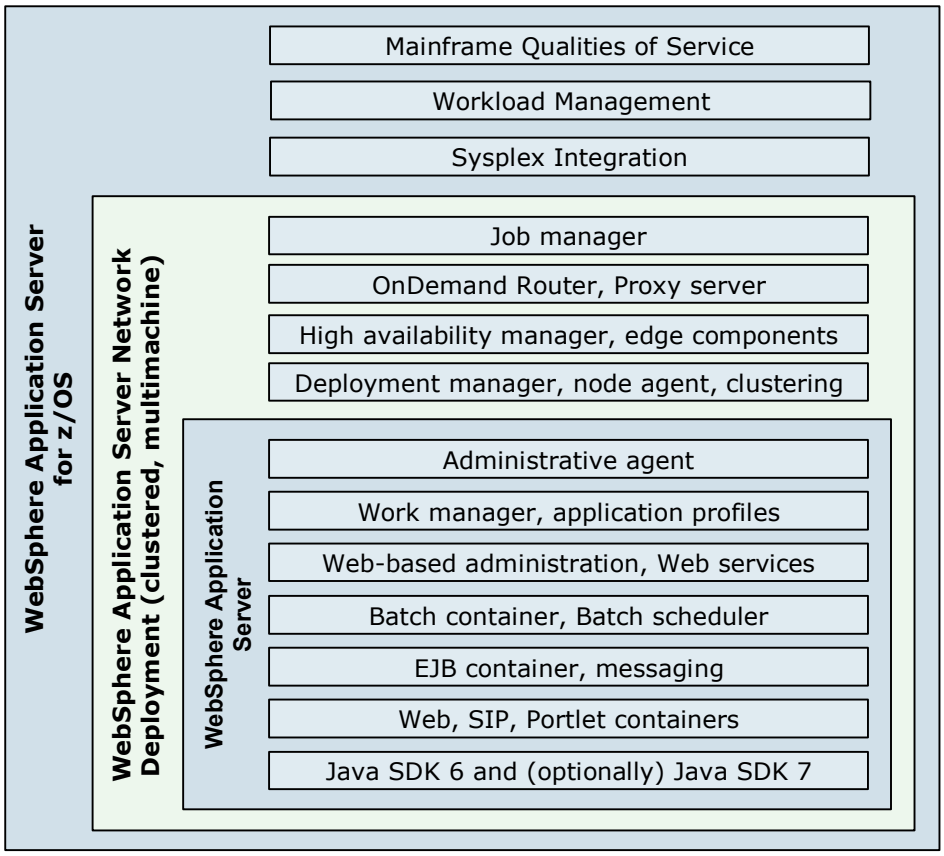


Figure 1-4 Packaging Structure WebSphere Application Server V8.5

1.1.1 WebSphere Application Server—Express V8.5

The WebSphere Application Server—Express V8.5 package is geared to those users who need to get started quickly with a strong and affordable application server based on standards. It is a ready-to-go application foundation for single server, small scale deployments of dynamic web applications. The package can be easily migrated to more advanced versions of the WebSphere Application Server family as your business needs change. It contains portions of key Java EE 6, EJB 3.1, and web services support. The package is limited to a single-server environment and a maximum of 480 Processor Value Units (PVUs) per server or virtualized partition, for licensing purposes.

WebSphere Application Server—Express V8.5 includes compliance with the Java EE 6 programming model and these other programming models:

- OSGi
- WebSphere Batch
- XML
- Service Component Architecture (SCA)
- Session Initiation Protocol (SIP)
- Communications Enabled Applications (CEA)

This package does not provide clustering and high availability features.

For more information about WebSphere Application Server—Express V8.5, see:

http://www.ibm.com/software/webservers/appserv/express/

1.1.2 WebSphere Application Server V8.5

The WebSphere Application Server V8.5 package is the next level of server infrastructure in the WebSphere Application Server family. Although the WebSphere Application Server package is functionally equivalent to WebSphere Application Server—Express (single-server environment), this package differs in packaging and licensing. This package is ideal for lightweight application solutions where cost and simplicity are key. This package is also called the *WebSphere Application Server V8.5 Base package*.

For more information about WebSphere Application Server V8.5, see:

http://www.ibm.com/software/webservers/appserv/was/

1.1.3 WebSphere Application Server for Developers V8.5

The WebSphere Application Server for Developers V8.5 package is functionally equivalent to the WebSphere Application Server V8.5 package, but it is licensed for development use only. WebSphere Application Server for Developers is an easy-to-use development environment to build and test applications for your SOA. It provides simplified and no-charge access to enable developers to build and test in the same environment that will ultimately support their applications.

For more information about WebSphere Application Server for Developers V8.5, see:

http://www.ibm.com/software/webservers/appserv/developer/index.html

1.1.4 WebSphere Application Server Network Deployment V8.5

WebSphere Application Server Network Deployment (ND) V8.5 provides enterprise-level advanced performance, management, and high-availability for mission critical applications. It extends the base package of WebSphere Application Server and includes the following features:

- Clustering capabilities
- Edge components
- Dynamic scalability
- High availability
- Intelligent management
- Advanced centralized management features for distributed configurations

These features become more important in larger enterprises. In large enterprises, applications tend to service a larger client base, and more elaborate performance and high availability requirements tend to be in place.

WebSphere Application Server Network Deployment V8.5 Edge Components provide high performance and high availability features. For example, Load Balancer (a software load balancer) provides horizontal scalability. It dispatches HTTP requests among several web server or application server nodes that support various dispatching options and algorithms to assure high availability in high volume environments. Using Edge Component Load Balancer can reduce web server congestion, increase content availability, and provide scaling ability for the web server.

WebSphere Application Server Network Deployment also includes a dynamic cache service, which improves performance by caching the output of servlets, commands, web services, and JSP files. This cache can be replicated to the other servers. The state of dynamic cache can be monitored with the cache monitor application.

For more information about WebSphere Application Server Network Deployment V8.5, see:

http://www.ibm.com/software/webservers/appserv/was/network/

1.1.5 WebSphere Application Server for z/OS V8.5

IBM WebSphere Application Server for z/OS V8.5 provides the capability to deliver on business objectives. It can contain or reduce costs for business critical applications that use the full capabilities of the z/OS platform. This full-function version of Websphere Application Server Network Deployment uses the z/OS qualities of service to achieve optimized performance and continuous availability for mission critical applications. Although it offers all the options and functions common to Websphere Application Server V8.5 on distributed platforms, it enhances the product in various ways:

- Defines service level agreements (SLAs) on a transaction base (response time per transaction)
- Protects your production applications with workload management in times of unpredictable peaks
- Uses z/OS functionality for billing based on used resources or transactions
- Enables a central security repository, including Java role-based security, by using the Security Access Facility interface
- Builds a cluster inside of a single application server (multiservant)
- Profits from near linear hardware and software scalability
- Profits from IBM System z cluster (Parallel Sysplex®) and up to 99.999% availability
- Provides optional z/OS specific Liberty profile features that take advantage of z/OS qualities of service

For more information about WebSphere Application Server for z/OS V8.5, see:

http://www.ibm.com/software/webservers/appserv/zos_os390/

1.1.6 Packaging summary

Table 1-1 lists details of the WebSphere Application Server features.

Table 1-1 WebSphere Application Server V8.5 packaging

Features	Express	Base	Network Deployment	z/OS
EJB 3.1	Yes	Yes	Yes	Yes
Java EE 6 support	Yes	Yes	Yes	Yes
Advanced security	Yes	Yes	Yes	Yes
Broad operating system support and database connectivity	Yes	Yes	Yes	Yes
Integration with IBM Rational® Application Developer Assembly and Deploy	Yes	Yes	Yes	Yes

Features	Express	Base	Network Deployment	z/OS
Rapid Java Development and Deployment Kit 6.0	Yes	Yes	Yes	Yes
Runtime provisioning	Yes	Yes	Yes	Yes
Dynamic caching	Yes	Yes	Yes	Yes
Administrative agent	Yes	Yes	Yes	Yes
Edge Components	No	No	Yes	Yes
Large-scale transaction support	No	No	Yes	Yes
Advanced clustering	No	No	Yes	Yes
Job manager and deployment manager	No	No	Yes	Yes
Workload management within a server integrated with z/OS Workload Manager (for SLAs on a transactional level and reporting for chargeback)	No	No	No	Yes
Reporting and charge back: Granular reporting on resource consumption	No	No	No	Yes

1.2 Evolving Java application development standards

Java is the technology that powers the WebSphere Application Server products. Over the years, many software vendors have collaborated on a set of server-side application programming technologies that help build web accessible, distributed, platform-neutral applications. These technologies are collectively branded as the Java Platform, Enterprise Edition (Java EE). They build on the foundation of the Java Platform, Standard Edition (Java SE).

The Java EE platform provides specifications for developing multitier enterprise applications with Java. It consists of application technologies for defining business logic and accessing enterprise resources. These resources include databases, enterprise resource planning (ERP) systems, messaging systems, internal and external business services, and email servers.

Java EE provides the following benefits:

- An architecture-driven application development approach that reduces maintenance costs and allows for construction of an IT infrastructure that can grow to accommodate new services.
- Application development standards, tools, and predefined rules improve productivity, and accelerate and shorten development cycles.
- Packaging, deployment, and management standards for enterprise applications facilitate systems and operations management.
- Industry-standard technologies allow clients to choose among platforms, development tools, and middleware to power applications.
- Platform independence gives flexibility to create a single application and run it on multiple platforms, providing true portability to enterprise applications.
- Embedded support for Internet and web technologies allows applications to bring services and content to a wider range of users. It does not require proprietary integration.

For more information about the Java EE specifications, see:

http://www.oracle.com/technetwork/java/javaee/overview/index.html

WebSphere Application Server V8.5 provides the runtime environment for applications that conform to the J2EE 1.2, 1.3, 1.4, Java EE 5, and Java EE 6 specifications. Java EE 6 support adds the ability to start the Java compiler from within the Java virtual machine (JVM). It includes scripts with the ability to access application programming interfaces (APIs) within the JVM. By continuing to support previous levels of the Java specifications in addition to adding support for the new standards, WebSphere Application Server provides stability and reduced costs. It also provides the infrastructure to add the latest technologies into business applications.

1.3 Comprehensive programming model support

WebSphere Application Server V8.5 supports a wide variety of programming models that provide flexibility and improve developer productivity. These programming models are included:

- Java EE 6
- OSGi applications
- Web 2.0 Mobile
- WebSphere Batch
- XML
- Service Component Architecture (SCA)
- Communications Enabled Applications (CEA)
- Session Initiation Protocol (SIP)

WebSphere Application Server V8.5 provides optional support for the IBM WebSphere SDK Java Technology Edition Version 7.0. This IBM software development kit (SDK) provides a full-function SDK for Java that is compliant with Java SE 7 application programming interfaces (APIs). With IBM WebSphere SDK Java Technology Edition V7.0, you can develop and deploy Java applications at the Java 7 API level. It continues the “write once, run anywhere” Java paradigm at the Java API level. The SDK contains the Java Runtime Environment and other tools that enable developers to create Java applications.

For more information about programming model support and application development features, see Chapter 11, “Application development and deployment” on page 341.

1.4 Enhanced management capabilities

WebSphere Application Server has several topology management options to meet your demands. You can create basic scenarios with single application server environments, or multiple application servers that are administered from a single point of control, the deployment manager. Furthermore, you can extend your environment as needs change. These application servers can be clustered to provide scalable and highly available environments.

WebSphere Application Server has several key management features that allow you to set up, deploy, and maintain your application environments. These management features help you to build advanced and large-scale topologies, and reduce management and maintenance complexity.

- IBM Installation Manager

  WebSphere Application Server V8.5 uses IBM Installation Manager, which is a single installation tool that loads and installs product components from a structured collection of files known as a repository. IBM Installation Manager uses remote or local software repositories to install, modify, or update IBM software products, including WebSphere Application Server V8.5. Using the live repository, you can get an up-to-date list of available maintenance for your installed features and select exactly what maintenance to install. The Packaging Utility also allows you to create a central repository that is used for maintenance within the enterprise. This repository allows for greater administrative control and greater consistency across the installed users' community.

- Administrative agent

  Using administrative agent, you can centralize node administration and manage multiple stand-alone servers from a central point. This configuration can reduce costs and provide greater control in a non-federated application server environment.

- Job manager

  The job manager allows you to remotely manage multiple administrative agents, deployment managers, stand-alone application servers, and Liberty profile runtime environments. Using the job manager, you can asynchronously submit and administer jobs to these servers and administrative agents. The jobs can manage applications, modify production configuration, start and stop applications, and distribute files.

- Centralized installation manager

  The centralized installation manager provides the capability to perform centralized installations and apply maintenance to remote endpoints. It can be used to consolidate and simplify the steps that are required to perform installations and to apply maintenance on systems. You can use the centralized installation manager to install Installation Manager instances, update Installation Manager with a repository, and manage Installation Manager offerings. These activities can be done with the administrative console or the **`wsadmin`** tool. It is available from the job manager and deployment manager in distributed and z/OS environments.

- High Performance Extensible Logging (HPEL) and cross-component tracing

  The HPEL component provides a convenient mechanism for storing and accessing log, trace, System.err, and System.out information produced by the application server and your applications. It provides greater flexibility and ease of use for administrators to manage logging resources and work with log and trace content.

  The cross-component trace facility enables correlation of log and trace entries with minimal cost by identifying the root cause of problems across components. The cross-component trace facility annotates log and trace entries. Log entries that are related to a request serviced by more than one thread, process, or even server are identified as belonging to the same unit of work. This enhancement enables administrators and support teams to follow the flow of a request from end-to-end as it traverses thread or process boundaries.

1.5 Operational efficiency and intelligent management

WebSphere Application Server V8.5 provides enhanced user availability and application server resiliency with the following integrated intelligent management capabilities:

- Use intelligent management for application edition management to manage interruption-free production application deployments. You can validate a new edition of an application in the production environment without affecting users. You can also upgrade applications without incurring outages to users. You can also run multiple editions of a single application concurrently, directing users to different editions.
- Use intelligent management for intelligent routing to improve business results by ensuring that priority is given to business critical applications. The on-demand router prioritizes and routes requests based upon administrator-defined rules. The on-demand router can queue less important requests momentarily so that more important requests are handled more quickly. For example, in an e-commerce application, a purchase (checkout) can be defined as a higher priority than browsing the catalog.
- Use intelligent management for application server health management to monitor the status of application servers and to respond to problem areas before an outage occurs. You can manage the health of the application that serves the environment with a policy-driven approach that enables specific actions to occur when monitored criteria is met. For example, when memory usage exceeds a percentage of the heap size for a specified time, health actions can run to correct the situation.
- Use intelligent management for improved performance with dynamic clusters to automatically scale the number of running cluster members as needed to meet response time goals. You can use overload protection to limit the rate at which the on-demand router forwards traffic to application servers. This process prevents heap and processor exhaustion from occurring.

For more information about intelligent management functions that are available in WebSphere Application Server V8.5, see Chapter 5, “Intelligent Management” on page 107.

1.6 Security management

WebSphere Application Server V8.5 adds value to installations by providing the following security management and auditing improvements:

- Single sign-on (SSO) provides an API so that developers can perform downstream SSO without storing and sending user credentials.
- You can create multiple security domains within a single WebSphere Application Server cell. Each security domain can have its own user population (and underlying repository). Additionally, the application domain can be separated from the administrative domain.
- Security auditing records the generation of WebSphere Application Server administrative actions. These actions can include security configuration changes, key and certificate management, access control policy changes, and system resources management. With this feature, you can hold administrative users accountable for configuration and runtime changes.
- Additional enhancements in WebSphere Application Server V 8.5 enable you to track changes made to the application server configuration by using checkpoints made through the extended repository service. A *full* checkpoint is a complete copy of the entire configuration repository. A *delta* checkpoint is a subset snapshot of the configuration repository that is made when you change a product configuration. Use checkpoint data to restore the configuration repository to a prior state. To determine changes in the

configuration, extract data from a delta checkpoint to obtain the before and after versions of the files that were saved.

- With the Auditing service provider setting, the WebSphere Application Server administrator can configure the behavior of the audit files when they reach maximum capacity.
- With the DMZ Secure Proxy, a proxy server hardened for DMZ topologies, you can have a more secure out-of-box proxy implementation outside the firewall.
- Fine-grained administration security can now be enforced through the administration console. You can restrict access based on the role of the administrator at the cell, node, cluster, or application level, offering fine-grained control over administrator scope. This capability is valuable in large-cell implementations where multiple administrators are responsible for subsets of the application portfolio that is running on the cell.

1.7 Simplified interoperability

The expanded integration support in WebSphere Application Server simplifies interoperability in mixed environments.

1.7.1 Web services

Web services allow for the definition of functions or services within an enterprise. These definitions can be accessed by using industry standard protocols (such as HTTP and XML) that are already in use today. These protocols allow for easy integration of both intra-business and inter-business applications that can lead to increased productivity, expense reduction, and quicker time to market. Web services are also the key elements of SOA, which provides reuse of existing service components and more flexibility to allow you to address changing opportunities.

WebSphere Application Server V8.5 includes support for the following web services and web services security standards:

- Web Services Interoperability Organization (WS-I) Basic Profile 1.2 and 2.0
- WS-I Reliable Secure Profile
- JAX-WS 2.2
- JAX-RS 1.1
- Java Architecture for XML binding (JAXB) 2.2
- SOAP 1.2
- SOAP Message Transmission Optimization Mechanism (MTOM) 1.0
- XML-binary Optimized Packaging (XOP)
- Web Services Reliable Messaging (WS-RM) 1.1
- Web Services Addressing (WS-Addressing) 1.0
- Web Services Secure Conversation (WS-SC) 1.0
- Web Services Policy 1.5

WebSphere Application Server supports the Web Services for Remote Portlets (WSRP) standard. By using this standard, portals can provide portlets, applications, and content as WSRP services. Other portals can integrate the WSRP services as remote portlets for their users. With WebSphere Application Server, you can provide WSRP services. A portlet container, such as WebSphere Portal, can use these services as remote portlets.

1.7.2 Messaging, connectivity, and transaction management

WebSphere Application Server supports asynchronous messaging through the use of a Java Message Service (JMS) provider and its related messaging system. WebSphere Application Server includes a fully integrated JMS 1.1 provider called the *default messaging provider*. The default messaging provider complements and extends WebSphere MQ and the 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 also supports your existing WebSphere MQ system as a JMS provider, and third-party messaging providers.

WebSphere Application Server also supports Java EE Connector Architecture (JCA) 1.5 resource adapters, which provide connectivity between application servers and Enterprise information systems (EIS). WebSphere Application Server V8.5 comes with Java Transaction API (JTA) 1.1 specification support, which provides standard Java interfaces for transaction management.

1.7.3 Authentication and authorization

WebSphere Application Server provides authentication and authorization capabilities to secure administrative functions and applications. The options for user registries include an operating system user registry, such as the IBM Resource Access Control Facility (IBM RACF®) on z/OS. They also include a Lightweight Directory Access Protocol (LDAP) registry (for example, IBM Tivoli Directory Server), custom registries, file-based registries, and federated repositories.

In addition to the default authentication and authorization capabilities, WebSphere Application Server has support for Java Authorization Contract for Containers (JACC) 1.1. This support gives you the option of using an external JACC-compliant authorization provider for application security. The IBM Tivoli Access Manager client that is embedded in WebSphere Application Server is JACC-compliant, and can be used to secure your WebSphere Application Server-managed resources.

1.7.4 Application client

With WebSphere Application Server, you can run client applications that communicate with a WebSphere Application Server by installing the application client component on the system on which the client applications run. This component provides a stand-alone client runtime environment for your client applications. It also enables your client to run applications in a Java EE environment that is compatible with EJB.

The Application Client for WebSphere Application Server V8.5 consists of the following client components:

- Java EE application client application

  This component uses services provided by the Java EE Client Container.

- Thin application client application

  This component does not use services provided by the Java EE Client Container, and includes a JVM API.

- Applet application client application

  With this component, users can access enterprise beans in the WebSphere Application Server through a Java applet in an HTML document.

- ActiveX to EJB Bridge application client application

  This component uses the Java Native Interface (JNI) architecture to programmatically access the JVM API (Microsoft Windows only).

1.8 Advanced tools and extensions

This section provides information about Websphere Application Server tool and extension enhancements.

1.8.1 Application development and deployment tools

WebSphere Application Server V8.5 includes a new application assembly and deployment tool, called IBM Assembly and Deploy Tools for WebSphere Administration. This tool replaces the previously available IBM Rational Application Developer Assembly and Deploy.

IBM Assembly and Deploy Tools for WebSphere Administration is targeted for the assembly and deployment of applications, providing the following capabilities:

- Import and validate applications.
- Edit deployment descriptors and binding files.
- Edit enterprise archive (EAR)-level configuration (enhanced EAR).
- Create and debug Jython and **`wsadmin`** scripts.
- Deploy EJB and web services.
- Deploy applications to local or remote WebSphere Application Server V8.5 runtime environments.
- Debug applications on WebSphere Application Server V8.5.

For more details about IBM Assembly and Deploy Tools for WebSphere Administration, see Chapter 11, “Application development and deployment” on page 341.

1.8.2 WebSphere Customization Toolbox

The WebSphere Customization Toolbox for WebSphere Application Server V8.5 includes the following tools for managing, configuring, and migrating various parts of the WebSphere Application Server environment:

- The Web Server Plug-ins Configuration Tool allows you to configure web server plug-ins.
- The Profile Management Tool for z/OS allows you to generate jobs and instructions for creating profiles for WebSphere Application Server for z/OS from a Windows or Linux system based on Intel.

- The z/OS Migration Management Tool allows you to generate definitions to migrate WebSphere Application Server for z/OS profiles from a Windows or Linux system based on Intel.

1.8.3 Web 2.0 and Mobile Toolkit

The WebSphere Application Server Web 2.0 and Mobile Toolkit simplifies the addition of Asynchronous JavaScript and XML (Ajax) rich desktop and mobile user interfaces and Representational State Transfer (REST) web services to Java web applications. Web 2.0 capabilities, such as Ajax and REST, help application developers create more connected, interactive applications that result in higher customer satisfaction, user productivity, and enhanced decision making. New mobile Ajax components enable developers to create mobile web applications that run on devices such as smart phones and tablets.

1.9 Related products

IBM offers complementary software products for WebSphere Application Server that provide a simplified development process, enhanced management features, and a high performance runtime environment. This section provides information about the following related products:

- WebSphere Application Server Community Edition
- WebSphere eXtreme Scale
- Rational Application Developer for WebSphere Software V8.5

1.9.1 WebSphere Application Server Community Edition

WebSphere Application Server Community Edition is a lightweight single-server Java EE application server built on Apache Geronimo, which is the open source application server project of the Apache Software Foundation. This edition of WebSphere Application Server is based on open source code and is available to download at no cost.

Product information: The code base of WebSphere Application Server Community Edition is different from the code base for WebSphere Application Server. WebSphere Application Server Community Edition is not a different packaging option for WebSphere Application Server. It is a separate product.

WebSphere Application Server Community Edition is a powerful alternative to open source application servers and has the following features:

- Brings together the best related technologies across the broader open source community to support Java EE specifications such as the following examples:
  - Apache Aries
  - Apache MyFaces
  - Apache OpenEJB
  - Apache Open JPA
  - Apache ActiveMQ
  - TranQL
- Includes support for Java EE 6 and Java SE 6
- Supports the JDK from IBM and Oracle
- Can be used as a run time for Eclipse with its plug-in

- Includes an open source Apache Derby database, which is a small-footprint database server with full transactional capability
- Contains an easy-to-use administrative console application
- Supports product binary files and source code as no-charge downloads from the IBM website
- Provides optional fee-based support for WebSphere Application Server Community Edition from IBM Technical support teams
- Can be included in advanced topologies and managed with the Intelligent Management functionality of WebSphere Application Server V8.5

For more information and the option to download WebSphere Application Server Community Edition, see:

http://www.ibm.com/software/webservers/appserv/community/

1.9.2 WebSphere eXtreme Scale

WebSphere eXtreme Scale provides the technology to enhance business by extending the data-caching concept with advanced features. With WebSphere eXtreme Scale, business applications can process large volumes of transactions with efficiency and linear scalability. WebSphere eXtreme Scale operates as an in-memory data grid that dynamically caches, partitions, replicates, and manages application data and business logic across multiple servers. It provides transactional integrity and not apparent failover to ensure high availability, high reliability, and consistent response times.

For more information about WebSphere eXtreme Scale, see:

http://www.ibm.com/software/webservers/appserv/extremescale/

1.9.3 Rational Application Developer for WebSphere Software V8.5

Rational Application Developer for WebSphere Software is a full-featured Eclipse-based IDE that includes a comprehensive set of tools to improve developer productivity. It is the only Java IDE tool that you need to design, develop, and deploy your applications for WebSphere Application Server.

Rational Application Developer for WebSphere Software adds functions to Rational Application Developer Standard Edition (Figure 1-5).

Figure 1-5 Rational development tools

Rational Application Developer for WebSphere software includes the following functions:

- Concurrent support for J2EE 1.2, 1.3, 1.4, Java EE 5, and Java EE 6 specifications and support for building applications with JDK 5 and JRE 1.6
- EJB 3.1 productivity features

- Visual editors such as:
  - Domain modeling
  - UML modeling
  - Web development
- Web services and XML productivity features
- Portlet development tools
- Relational data tools
- WebSphere Application Server V6.1, V7, V8, and V8.5 test servers
- Web 2.0 development features for visual development of responsive Rich Internet Applications with Ajax and Dojo
- Integration with the Rational Unified Process and the Rational tool set, which provides the end-to-end application development lifecycle
- Application analysis tools to check code for coding practices

  Examples are provided for best practices and issue resolution.
- Enhanced runtime analysis tools, such as memory leak detection, thread lock detection, user-defined probes, and code coverage
- Component test automation tools to automate test creation and manage test cases
- WebSphere adapters support, including CICS, IBM IMS, SAP, Siebel, JD Edwards, Oracle, and PeopleSoft
- Support for Linux and Microsoft Windows operating systems.

For more information about Rational Application Developer for WebSphere Software V8, see:

http://www.ibm.com/software/awdtools/developer/application/

1.10 New features and capabilities in WebSphere Application Server V8.5

This section introduces the features and capabilities in WebSphere Application Server V8.5. Later chapters provide details for these features. These key features are grouped into the following areas:

- Intelligent management and enhanced resiliency
- Light-weight, composable application server with the Liberty profile
- Improved operations, security, control, and integration
- Integrated tools
- Improved application development

1.10.1 Intelligent management and enhanced resiliency

WebSphere Application Server V8.5 contains the following features in the area of intelligent management:

- Application edition management

  Manage interruption-free production application deployment by validating a new edition of an application in production without affecting existing users or incurring an outage. You can also run multiple editions of a single application concurrently. This feature allows you

to direct users to one instance of the application or the other based on customer-defined routing rules.

- Application server health management

  Monitor the status of your application servers and proactively respond to problem areas before an outage occurs. Create policies that define what a problem area is and what actions to perform when that criteria is met.

- Intelligent routing

  Prioritize and route requests based on administrator-defined rules to ensure optimum business results. Ensure that requests for business critical applications get priority over less important application requests. Priority can also be defined within an application as needed.

- Dynamic clustering

  Automatically scale up or down the number of cluster members based on defined response time goals for users. Make better use of resources by not having to plan for all applications being at the highest peak usage at all times.

For more information about the intelligent management functions that are available in WebSphere Application Server V8.5, see Chapter 5, “Intelligent Management” on page 107.

WebSphere Application Server V8.5 has several resiliency features in the messaging engine. These features improve recovery and restarting of the messaging engine in the event of a failure.

1.10.2 Light-weight, composable application server with the Liberty profile

The Liberty profile is a simplified and lightweight run time for web applications. The small footprint and low resource usage, along with simplified configuration, makes the WebSphere Application Server V8.5 Liberty profile a good option for developers. It can be used to build web applications that do not require the full Java EE environment of traditional enterprise application server profiles.

The Liberty profile provides a lightweight development and application-serving environment that is configured with the level of capabilities needed for the individual applications. The Liberty profile allows you to specify only those features that are needed for the applications deployed, reducing the memory footprint and increasing performance.

The Liberty profile is optimized for developer and operational productivity. You can use it in both development and production environments. Enterprise qualities of service, such as security and transaction integrity, can be enabled as required. The Liberty profile provides the following key benefits:

- Installation by using an archive file or IBM Installation Manager
- A lightweight, composable runtime environment that starts only those services that are defined in the application server configuration
- A faster start time and small memory footprint, because only configured services are started
- Simplified configuration and dynamic configuration updates that increase developer productivity
- Built in configuration defaults with override capability in an easily editable XML file

- Shareable configurations across an application development team and an included capability to provide customization for teams without individual developers having to make manual updates
- Java EE and OSGi application deployment support for web applications
- Rapid application deployment by using a drop-in directory or adding applications to the server configuration
- Easy and quick distribution and deployment of a Liberty profile server and applications as a single package by using the job manager
- Provides broad tool support by using Eclipse plug-ins for WebSphere Application Server Developer Tools
- Enhanced development environment that supports distributed platforms, z/OS, and Mac OS

For more information about the Liberty profile, see Chapter 4, “An overview of the Liberty profile” on page 91.

1.10.3 Improved operations, security, control, and integration

WebSphere Application Server V8.5 improves operations, security, control, and integration through the following enhancements:

- Selectable Java Technology

  Better control over the Java level that business applications use. Also allows you to switch between using the default Java 6 level and an optionally installable Java 7 level.

- SCA programming model

  Partial support for several OASIS specifications.

- Derby 10.8

  Connect applications to the latest versions of databases, including the Derby 10.8 database, which is tested with WebSphere Application Server V8.5 and is included in the packaging.

- Problem determination

  Simplified problem determination with improvements in the area of cross component trace and the HPEL feature.

- Enhanced security for administrative configuration audit tracking

  Track changes made to the application server configuration through the extended repository service.

- WebSphere batch enhancements

  Enterprise level WebSphere batch enhancements that enable the entry, execution, and management of batch processing, while also integrating with existing enterprise components and online transaction processing.

1.10.4 Integrated tools

You can accelerate developer productivity by using integrated and optimized developer tools.

Rational Application Developer V8.5 provides a complete environment for enterprise development for Java, Java EE, web, web services, SOA, OSGi, and Portal designers and

developers. Develop, assemble, and deploy applications to WebSphere Application Server V8.5, and then test applications by using the included test environment.

IBM WebSphere Application Server Developer Tools for Eclipse V8.5 is a lightweight set of tools for developing, assembling, and deploying Java EE, OSGi, Web 2.0, and Mobile applications to WebSphere Application Server.

For more information about Application Deployment, see Chapter 11, “Application development and deployment” on page 341.

1.10.5 Improved application development

WebSphere Application Server V8.5 enables improved application development to enhance the developer experience. The following features enable an enhanced developer experience:

- Selectable Java 7 allows developers to take advantage of Java 7 enhancements where needed with the optionally installable and selectable Java 7 level. The developers can also choose to remain at the previous Java 6 level.
- Assemble OSGi applications from reusable bundles that contain EJB assets. For more information about application deployment, see Chapter 11, “Application development and deployment” on page 341.
- Extend the reach of business applications to mobile devices such as smart phones and tablets with the WebSphere Application Server Web 2.0 and Mobile Toolkit.

2

Concepts of WebSphere Application Server

Before you can plan a WebSphere Application Server installation and select a topology, you need to understand the basic structural concepts and elements that make up a WebSphere Application Server runtime environment.

This chapter includes the following sections:

- Core concepts of WebSphere Application Server
- Additional concepts for WebSphere Application Server
- Server configurations
- Security
- Service integration
- Clusters and high availability
- Run times

2.1 Core concepts of WebSphere Application Server

The following concepts are central to understanding the architecture of WebSphere Application Server V8.5:

- Applications
- Containers
- Application servers
- Profiles
- Nodes, node agents, and node groups
- Cells
- Deployment manager

A person in an administrative role must understand these concepts to manage WebSphere Application Server on a regular basis. Understanding these concepts and how they apply to your environment facilitates designing and troubleshooting.

This section provides information about these concepts. You can find additional concepts about WebSphere Application Server that build on these core concepts in 2.2, “Additional concepts for WebSphere Application Server” on page 39.

2.1.1 Applications

At the heart of WebSphere Application Server is the ability to run *applications*, including the following types:

- Enterprise
- Business-level
- Middleware

Websphere Application Server V8.5 can run the following types of applications, which are described in the following sections:

- Java Platform, Enterprise Edition applications
- Portlet applications
- Session Initiation Protocol applications
- Business-level applications
- OSGi applications
- Batch applications

Figure 2-1 illustrates the applications that run in the Java virtual machine (JVM) of WebSphere Application Server.

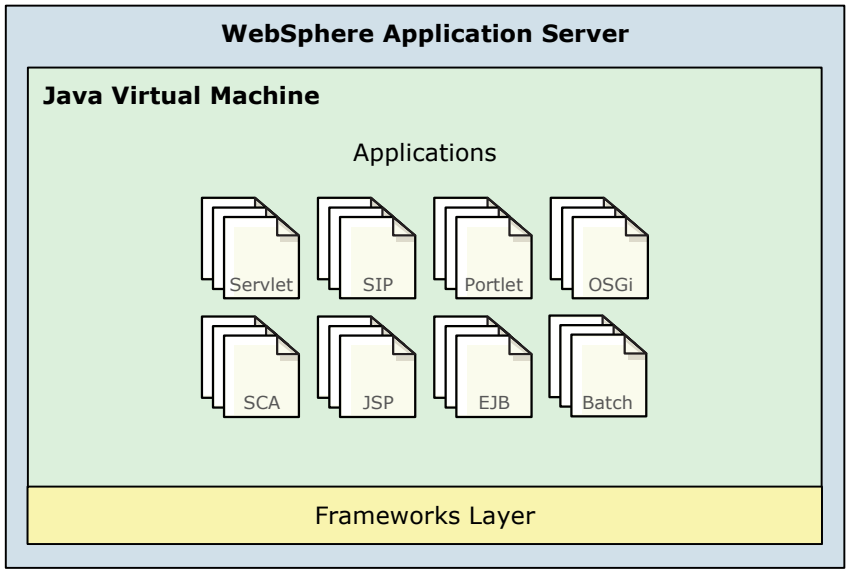


Figure 2-1 Applications running in WebSphere Application Server

Java Platform, Enterprise Edition applications

Java Platform, Enterprise Edition (Java EE) is the standard for developing, deploying, and running enterprise applications.

WebSphere Application Server V8.5 supports the Java EE 6 specification. New and existing enterprise applications can take advantage of the capabilities added by Java EE 6. If you decide not to use the Java EE 6 capabilities, portable applications continue to work with identical behavior on the current version of the platform.

Version note: IBM WebSphere SDK Java Technology Edition V6.0 is installed by default with WebSphere Application Server V8.5. Optionally, you can install IBM WebSphere SDK Java Technology Edition V7.0 in addition to the default Java version by using IBM Installation Manager. In WebSphere Application Server V8.5, you can select between Java SDK V6 and V7.

The Java EE programming model has the following types of application components:

- Enterprise JavaBeans (EJB)
- Servlets and JavaServer Pages (JSP) files
- Application clients (Java Web Start Architecture 1.4.2)

The primary development tool for WebSphere Application Server Java EE 6 applications is IBM Rational Application Developer for WebSphere V8.5. It contains tools to create, test, and deploy Java EE 6 applications. Java EE applications are packaged as enterprise archive (EAR) files.

For more information about Java EE 6 supported specifications, see the JSR page on the Java Community Process website at:

`http://jcp.org/en/jsr/detail?id=316`

For more information about web application specifications, see the following resources:

- JSR 154, 53 and 315 (Java Servlet 3.0 specification)

  http://jcp.org/en/jsr/detail?id=315
- JSR 252 and127 (Apache MyFaces JSF 2.0 specification)

  http://jcp.org/en/jsr/detail?id=314
- JSR 318 (EJB 3.1 specification)

  http://jcp.org/en/jsr/detail?id=318

Portlet applications

The *portlet container* in WebSphere Application Server V8.5 provides the runtime environment for Java Specification Requests (JSR) 286-compliant specification 2.0 portlets. Portlet applications are intended to be combined with other portlets collectively to create a single page of output. The primary development tool for portlets on WebSphere Application Server portlet applications is Rational Application Developer V8.5.

Portlets are packaged in web archive (WAR) files. The portlet run time does not provide the advanced capabilities of WebSphere Portal, such as portlet aggregation and page layout, personalization and member services, or collaboration features.

For more information about supported specifications for portlet application, see the JSR page for Portlet 2.0 on the Java Community Process website at:

http://jcp.org/en/jsr/detail?id=286

Session Initiation Protocol applications

Session Initiation Protocol (SIP) applications are Java programs that use at least one SIP 1.1 Servlet API specification JSR 289. SIP is used to establish, modify, and terminate multimedia IP sessions. SIP negotiates the medium, the transport, and the encoding for the call. After the SIP call is established, the communication takes place over the specified transport mechanism, independent of SIP. Examples of application types that use SIP include voice over IP (VOIP), click-to-call, and instant messaging.

Rational Application Developer V8.5 provides special tools for developing SIP applications. SIP applications are packaged as SIP archive (SAR) files and are deployed to the application server by using the standard WebSphere Application Server administrative tools. SAR files can also be bundled in a Java EE enterprise archive (EAR file), similar to other Java EE components.

For more information about SIP applications, see the following resources:

- JSR 289 SIP Servlet API 1.1 Specification

  http://jcp.org/en/jsr/detail?id=289
- RFC 3261 SIP Session Initiation Protocol

  http://www.ietf.org/rfc/rfc3261.txt

Business-level applications

A *business-level application* is an administrative concept that expands the options that are offered by the Java EE definition of an application. Business-level applications have a grouping notion. It includes WebSphere artifacts, such as Java EE artifacts and Service Component Architecture (SCA) packages, libraries, and proxy filters under a single application definition. Every artifact in the group is a composition unit.

A business-level application can be useful when an application has the following characteristics:

- Is composed of multiple composition units
- Applies to the post-deployment side of the application lifecycle
- Contains additional libraries or artifacts that are not based on Java EE
- Includes artifacts that run on heterogeneous environments that include WebSphere Application Server run times and run times that are not based on WebSphere Application Server
- Is defined in a recursive manner (for example, if an application includes other applications)

OSGi applications

The *OSGi application programming model* in WebSphere Application Server V8.5 enables you to develop, assemble, and deploy modular applications that use the Java EE 6 and OSGi R4 V4.2 Service Platform technologies. OSGi is a module system that is compatible with systems based on Java, and implements a dynamic component model.

OSGi applications are built on an architecture for developing and deploying modular applications and libraries. An OSGi logical container specifically supports developing Java applications that can be broken up into modules. From an administrator's and developer's perspective, OSGi provides these advantages:

- Different application modules (bundles) can be remotely installed, uninstalled, started, updated, and stopped without restarting the application server.
- More than one version of an application module can run at the same time.
- Applications are more portable, easier to re-engineer, and more adaptable to changing requirements. OSGi provides the infrastructure for the developing and deploying service-oriented, mobile, and embedded applications. It enforces service-oriented design at the module level.
- Application archive size, disk, and memory footprint can be reduced because of the augmentation that is related to the OSGi application deployment process.

Modular components and features that are created with OSGi technology are enabled in several ways. The OSGi specification determines how classes are loaded for OSGi bundles. An OSGi bundle is a JAR file, but has additional headers in the JAR file manifest. In a plain JVM, a bundle behaves similarly to a normal JAR file. In a JVM that includes an OSGi framework, the metadata in the bundle is processed by the framework, and additional modularity characteristics are applied. For example, because each bundle is placed in a sandbox, versions of logging libraries with one bundle do not conflict with other versions of the same product in different bundles.

The OSGi specification requires that the implementations of the modules include well-defined interfaces and a manifest that contains detailed information about the content. This use of interfaces and metadata in the manifest enforces loosely coupled, yet tightly cohesive, modules.

OSGi and WebSphere Application Server V8.5: WebSphere Application Server V8.5 uses Eclipse Equinox 3.6, which is the OSGi R4 v4.2 reference implementation of the core. It contains OSGi programming model enhancements, including EJB support. Other improvements are related to OSGi Blueprint security.

For more information, see the following resources:

- Reference information about developing enterprise OSGi applications for WebSphere Application Server:

  http://www.ibm.com/developerworks/websphere/techjournal/1007_robinson/1007_robinson.html

- IBM Education Assistance an online presentation about developing modular and dynamic OSGi applications:

  http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.was_v8/was/8.0/ProgramingModel/WASV8_OSGi_part1/player.html

- Preferred practices for working with OSGi applications:

  http://www.ibm.com/developerworks/websphere/techjournal/1007_charters/1007_charters.html

- Supported specifications for OSGi applications:

  http://www.osgi.org/Release4/HomePage

Batch applications

A *batch application* is an asynchronous, typically long-running application that is commonly used for bulk processing tasks. A frequent use case is processing large input files or Java Database Connectivity (JDBC) record sets. WebSphere batch applications are implemented as simple Java classes, and run according to job definitions described in xJCL job control language. For more information, see Chapter 6, “WebSphere Batch” on page 137 or IBM Education Assistance at:

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp

2.1.2 Containers

Containers are specialized to run specific types of applications and can interact with other containers by sharing session management, security, and other attributes. Figure 2-2 illustrates applications that run in different containers inside the JVM. *Containers* provide runtime support for applications.

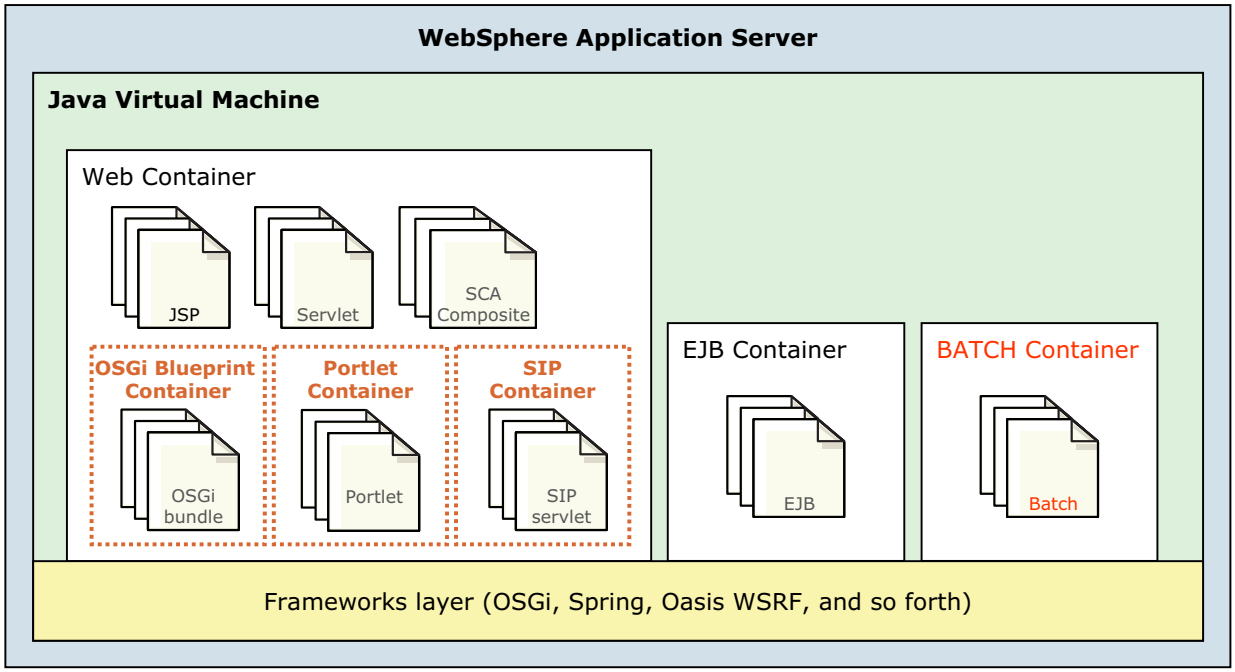


Figure 2-2 WebSphere Application Server V8.5 container services

WebSphere Application Server V8.5 includes the following logical containers:

- The *web container* processes servlets, JSPs, and other types of server-side objects.

  Each application server run time has one logical web container. Requests are received by the web container through the web container *inbound transport chain*. The chain consists of a Transmission Control Protocol (TCP) inbound channel that provides the connection to the network, an HTTP inbound channel that serves HTTP 1.0 and 1.1 requests. It also includes a web container channel over which requests for servlets and JSPs are sent to the web container for processing. Requests for HTML and other static content that are directed to the web container are served by the web container inbound chain.

- The *Enterprise JavaBeans (EJB) container* provides all of the runtime services that are needed to deploy and manage enterprise beans.

  This container is a server process that handles requests for both session and entity beans. The container provides many low-level services, including transaction support. From an administrative viewpoint, the container manages data storage and retrieval for the contained enterprise beans. A single container can host more than one JAR file.

- The *Batch container,* new in WebSphere Application Server V8.5*,* is where the job scheduler runs jobs written in XML job control language (xJCL).

  The batch container provides an execution environment for the execution of batch applications based on Java EE. Batch applications are deployed as EAR files and follow either the transactional batch or compute-intensive programming models.

The following containers are logical extensions of the web container main function:

- The *portlet container* provides the runtime environment to process JSR 286-compliant portlets. A simple portal framework is built on top of the web container to render a single portlet into a full browser page.
- The *SIP container* processes applications that use at least one SIP servlet written to the JSR 289 specification. It provides network services over which it receives requests and sends responses. It determines which applications to start and in what order. The container supports the UDP, TCP, and TLS/TCP protocols.
- The *OSGi Blueprint container* processes OSGi applications based on the OSGi framework. The OSGi Blueprint is separate from Java EE technology. However, they can be combined to deploy modular applications that use both Java EE 6/7 and OSGi R4 V4.2 technologies.

2.1.3 Application servers

At the core of each product in the WebSphere Application Server family is an *application server*. The *application server* is the platform on which Java language-based applications run (Figure 2-3). It provides services that can be used by business applications, such as database connectivity, threading, and workload management.

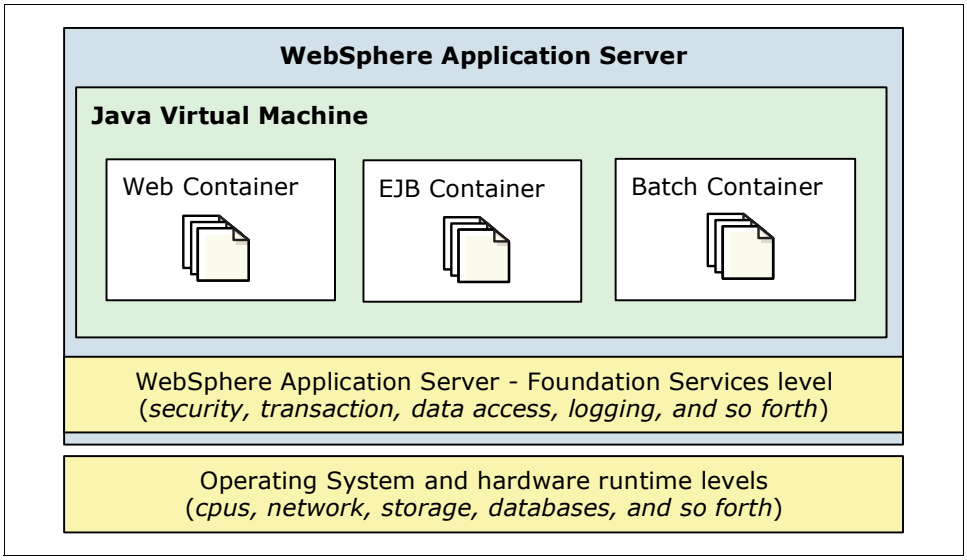


Figure 2-3 Relationship between applications and WebSphere Application Server

The following packaging options of the WebSphere Application Server family are presented in this book:

- IBM WebSphere Application Server Express V8.5, referred to as *Express*
- IBM WebSphere Application Server V8.5, referred to as *Base*
- IBM WebSphere Application Server Network Deployment V8.5, referred to as *Network Deployment* or *ND*
- IBM WebSphere Application Server Hypervisor Edition V7, referred to as *Hypervisor Edition*
- IBM WebSphere Application Server for z/OS V8.5, referred to as *WebSphere Application Server for z/OS*

Each member has essentially the same main architectural structure shown in Figure 2-4. They are built on a common code base. The difference between the options involves licensing terms and platform support.

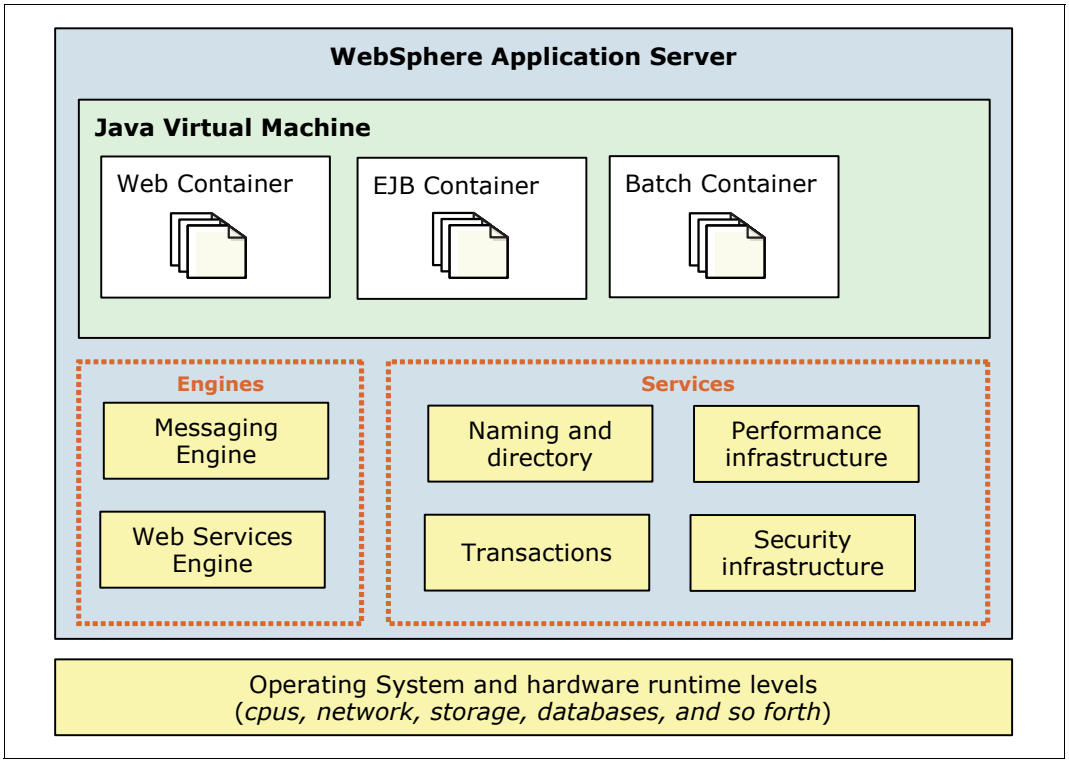


Figure 2-4 WebSphere Application Server architecture for Base and Express

The Base and Express platforms are limited to stand alone application servers. With the Network Deployment configuration (Figure 2-5 on page 30), more advanced topologies provide the following advantages:

- Workload management
- Scalability
- Near-continuous availability
- Central management of multiple application servers

These advantages are important for mission-critical applications. You can also manage multiple base profiles centrally, but you do not have workload management and the same capabilities for those base profiles.

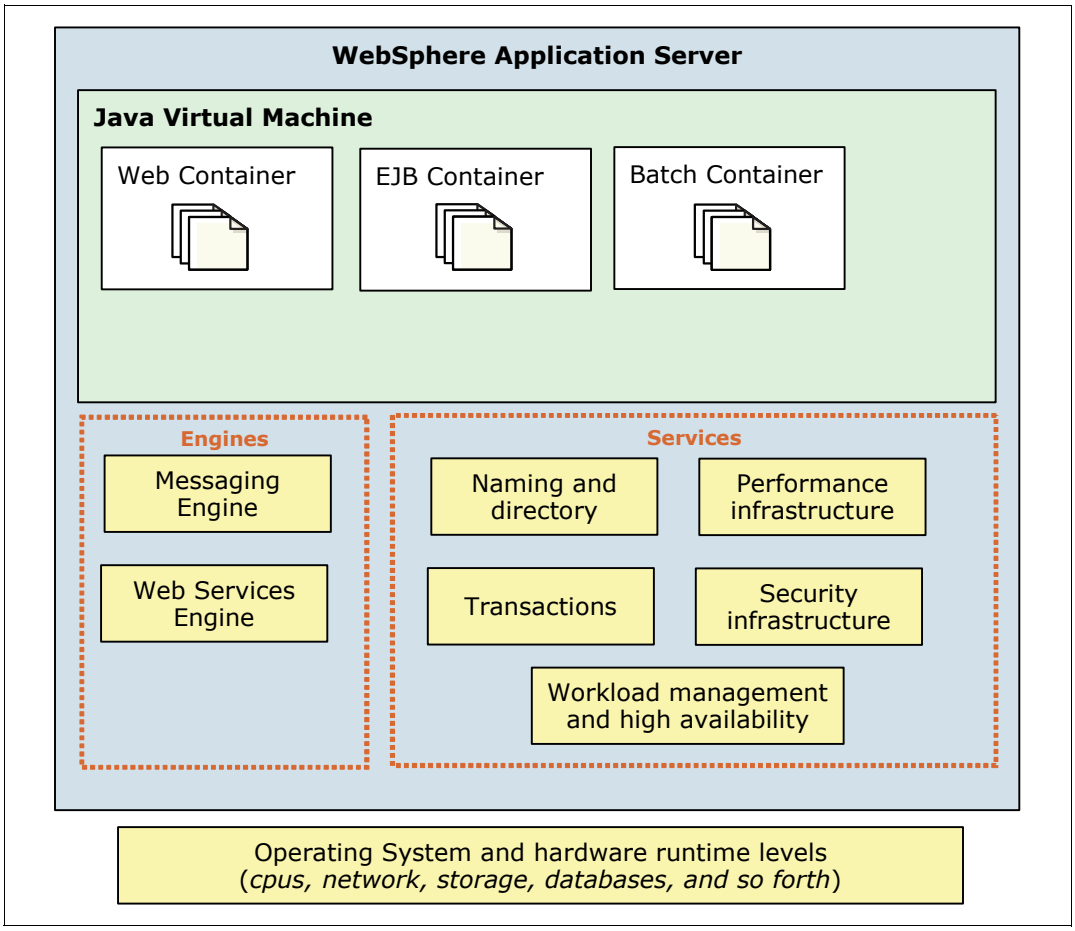


Figure 2-5 WebSphere Application Server architecture in a Network Deployment configuration

Stand-alone application servers

All WebSphere Application Server packages support a single stand-alone server environment. With a stand-alone configuration, each application server acts as a unique entity, functioning independently from other application servers. An application server runs one or more applications, and provides the services that are required to run these applications. Each stand-alone server is created by defining an application server profile (Figure 2-6).

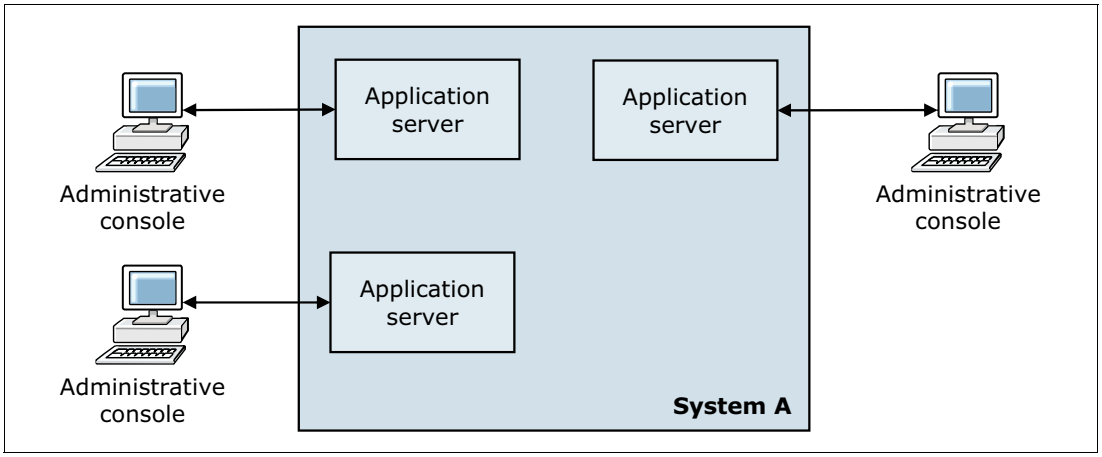


Figure 2-6 Stand-alone application server configuration

A stand-alone server can be managed from its own administrative console. You can also use the **wsadmin** scripting facility in WebSphere Application Server to perform every function that is available in the administrative console application.

Multiple stand-alone application servers can exist on a system. You can either use independent installations of the WebSphere Application Server product binary files, or create multiple application server profiles within one installation. However, stand-alone application servers do not provide workload management or failover capabilities. They are isolated from each other.

With WebSphere Application Server for z/OS, you can use workload load balancing and response time goals on a transactional base. You can also use balancing on a special clustering mechanism, the *multiple servant regions*, with a stand-alone application server. For more information, see 16.1.5, “Structure of an application server” on page 505.

Remember: With WebSphere Application Server V8.5, you can manage stand-alone servers from a central point by using administrative agents and a job manager. For more information, see 2.3.2, “Flexible management configurations” on page 52.

Distributed application servers

With Network Deployment, you can build a distributed server configuration to enable central administration, workload management, and failover. In this environment, you integrate one or more application servers into a cell that is managed by a central administration instance, a *deployment manager*. For more information, see 2.1.7, “Deployment manager” on page 38. The application servers can be on the same system as the deployment manager or on multiple separate systems. Administration and management are handled centrally from the administration interfaces of the deployment manager (GUI or scripting) as illustrated in Figure 2-7.

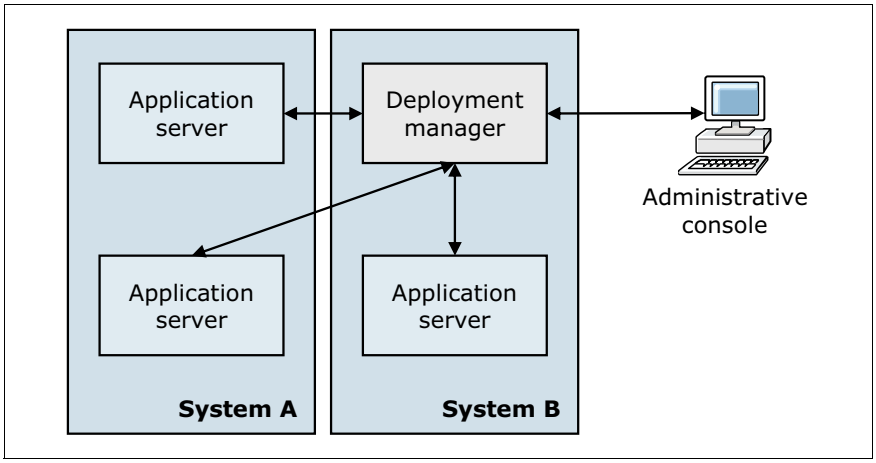


Figure 2-7 Distributed application servers with WebSphere Application Server V8.5

With a distributed server configuration, you can create multiple application servers to run unique sets of applications, and manage those applications from a central location. More importantly, you can cluster application servers to allow for workload management and failover capabilities. Applications installed in the cluster are replicated across the application servers. The cluster can be configured so when one server fails, another server in the cluster continues processing. Workload is distributed among containers in a cluster by using a weighted round-robin scheme.

Tip for z/OS: The weighted round-robin mechanism is replaced by the integration of WebSphere Application Server for z/OS in the Workload Manager (WLM). The WLM is a part of the operating system. Requests can be dispatched by using this configuration to a cluster member according to real-time load and regardless of whether the member reaches its defined response time goals.

Application servers types

WebSphere Application Server V8.5 provides the following server types, which can be defined and configured by using the administrative console:

- WebSphere Application Server
- Generic server
- On-demand router
- PHP server
- WebSphere proxy server
- WebSphere MQ server
- Community Edition server
- Web server

With the mixed server environment and mixed node definitions, other existing server types can be added and administered. These types include external WebSphere application servers, Apache Server, and Custom HTTP Server.

2.1.4 Profiles

WebSphere Application Server runtime environments are built by creating set of configuration files, named *profiles,* that represent a WebSphere Application Server configuration. The following categories of WebSphere Application Server files are available, as illustrated in Figure 2-8:

- *Product files* are a set of read-only static files or product binary files that are shared by any instances of WebSphere Application Server.
- *Configuration files (profiles)* are a set of user-customizable data files. This file set includes WebSphere configuration, installed applications, resource adapters, properties, and log files.

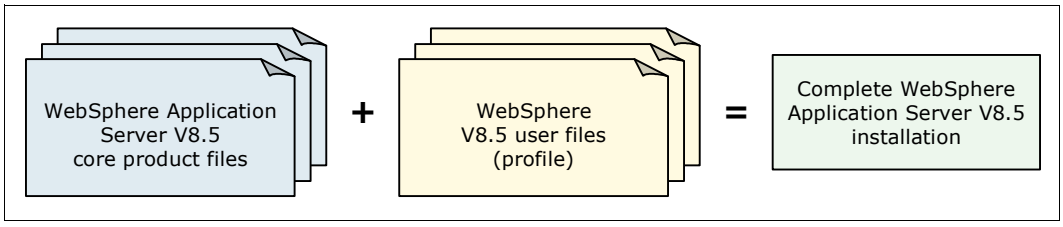


Figure 2-8 Anatomy of a profile

The Customization Toolbox allows you to create separate environments, such as for development or testing, without a separate product installation for each environment. Different profile templates are available in WebSphere Application Server V8.5 through the Customization Toolbox Profile Management Tool (PMT):

- Cell

  A cell template contains a federated application server node and a deployment manager.

- Deployment manager

  The Network Deployment profile provides the necessary configuration for starting and managing the deployment manager server.

- Default profile (for stand-alone servers)

  This server default profile provides the necessary configuration file for starting and managing an application server, and all the resources needed to run enterprise applications.

- Administrative agent

  This profile is used to create the administrative agent to administer multiple stand-alone application servers.

- Default secure proxy

  This profile is available when you install the DMZ secure proxy server feature.

- Job manager

  This profile coordinates administrative actions among multiple deployment managers, and administers multiple stand-alone application servers. It also asynchronously submits jobs to start servers, and completes various other tasks.

- Custom

  This profile, also known as *Empty Node* because it has no application server inside, can be federated to a deployment manager cell later. It is used to host application servers, clusters, an on-demand router, and other Java processes.

The Liberty profile: Do not confuse the Liberty profile with the concept of a profile created by the PMT in previous versions of WebSphere Application Server. The Liberty profile provides a composable and dynamic application server runtime environment on WebSphere Application Server V8.5. The Liberty profile is a subset of base functions of the WebSphere Application Server, which is installed separately.

You can create compressed files that contain all or subsets of the Liberty profile server installation. You can then extract these files on other target hosts as a substitute for the product installation.

With a simpler configuration model based on XML, you do not need to create a profile by using the PMT to create Liberty profile application servers.

Each profile contains files that are specific to that run time (such as logs and configuration files). You can create profiles during and after installation. After you create the profiles, you can perform further configuration and administration by using WebSphere administrative tools.

Each profile is stored in a unique directory path (Figure 2-9), which is selected by the user when the profile is created. Profiles are stored in a subdirectory of the installation directory by default, but can be located anywhere.

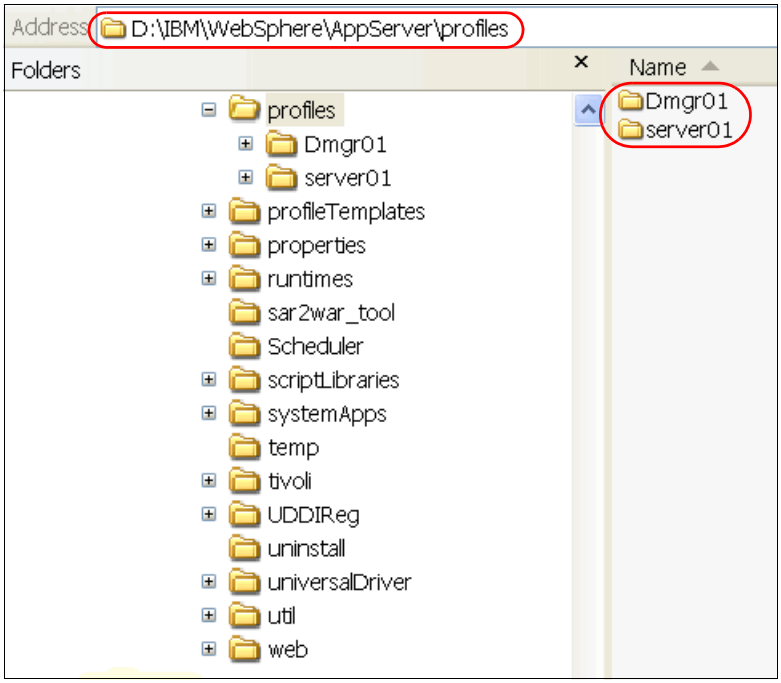


Figure 2-9 Profiles directory structure of WebSphere Application Server V8.5 on a Windows system

By creating various profiles, you can create a distributed server configuration by using one of the following methods:

- Create a deployment manager profile to define the deployment manager, and then create one or more custom node profiles. The nodes that are defined by each custom profile can be federated into the cell managed by the deployment manager. You can federate these nodes during profile creation, or manually later. The custom nodes can exist inside the same operating system image as the deployment manager or in another operating system instance. You can then create application servers by using the administrative console or **`wsadmin`** scripts.

  This method is useful when you want to create multiple nodes, multiple application servers on a node, or clusters.

- Create a deployment manager profile to define the deployment manager. Then create one or more application server profiles, and federate these profiles into the cell managed by the deployment manager. This process adds both nodes and application servers into the cell. The application server profiles can exist on the deployment manager system or on multiple separate systems or z/OS images.

  This method is useful in development or small configurations. Creating an application server profile gives you the option of having the sample applications installed on the server. When you federate the server and node to the cell, any installed applications can be carried into the cell with the server.

- Create a cell profile. This method creates both a deployment manager profile and an application server profile. The application server node is federated to the cell. Both profiles are on the same system.

  This method is useful in a development or test environment. Creating a single profile provides a simple distributed system on a single server or z/OS image.

2.1.5 Nodes, node agents, and node groups

This section provides details about the concepts of nodes, node agents, and node groups.

Nodes

A *node* is an administrative grouping of application servers for configuration and operational management within one operating system instance. You can create multiple nodes inside one operating system instance, but a node cannot leave the operating system boundaries. A stand-alone application server configuration has only one node. With Network Deployment, you can configure a distributed server environment that consists of multiple nodes that are managed from one central administration server.

From the administrative console, you can also configure middleware nodes (defined into a generic server cluster) to manage middleware servers by using a remote agent.

Figure 2-10 illustrates nodes that are managed from a single deployment manager.

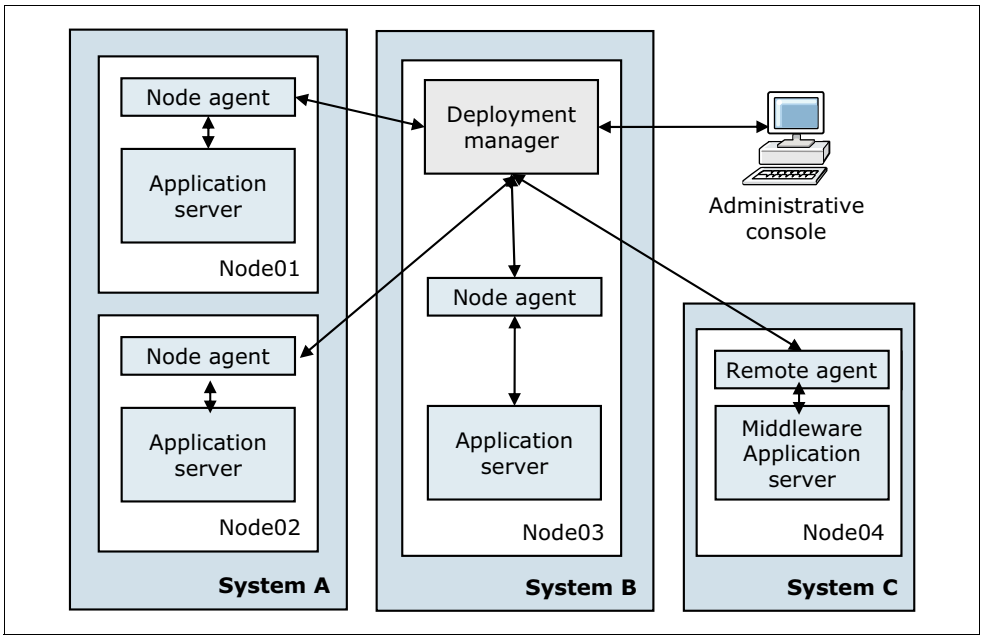


Figure 2-10 Node concept in a WebSphere Application Server Network Deployment configuration

Node agents

In distributed server configurations, each node has a *node agent* that works with the deployment manager to manage administration processes. A node agent is created automatically when you add (federate) a stand-alone application server node to a cell. Node agents are not included in the Base and Express configurations, because a deployment manager is not needed in these architectures. In Figure 2-10, each node has its own node agent that communicates directly or remotely with the deployment manager. The node agent is an administrative server that runs on the same system as the node. It monitors the application servers on that node, routing administrative requests from the deployment manager to those application servers.

Node groups

A *node group* is a collection of nodes within a cell that have similar capabilities in terms of installed software, available resources, and configuration. A node group is used to define a boundary for server cluster formation, so that the servers on the same node group host the same applications.

A node group validates that the node can run certain functions before allowing them. For example, a cluster cannot contain both z/OS nodes and non-z/OS nodes. In this case, you can define multiple node groups, one for the z/OS nodes and one for non-z/OS nodes. A DefaultNodeGroup is created automatically. The DefaultNodeGroup contains the deployment manager and any new nodes with the same platform type. A node can be a member of more than one node group.

Sysplex on z/OS: On the z/OS platform, a node must be a member of a system complex (*sysplex*) node group. Nodes in the same sysplex must be in the same sysplex node group. A node can be in one sysplex node group only. A sysplex is the z/OS implementation of a cluster. This technique uses distributed members and a central point in the cluster. It uses a *coupling facility* for caching, locking, and listing. The coupling facility runs special firmware, the Coupling Facility Control Code (CFCC). The members and the coupling facility communicate with each other by using a high-speed InfiniBand memory-to-memory connection of up to 120 Gbps.

Figure 2-11 shows a single cell that contains multiple nodes and node groups.

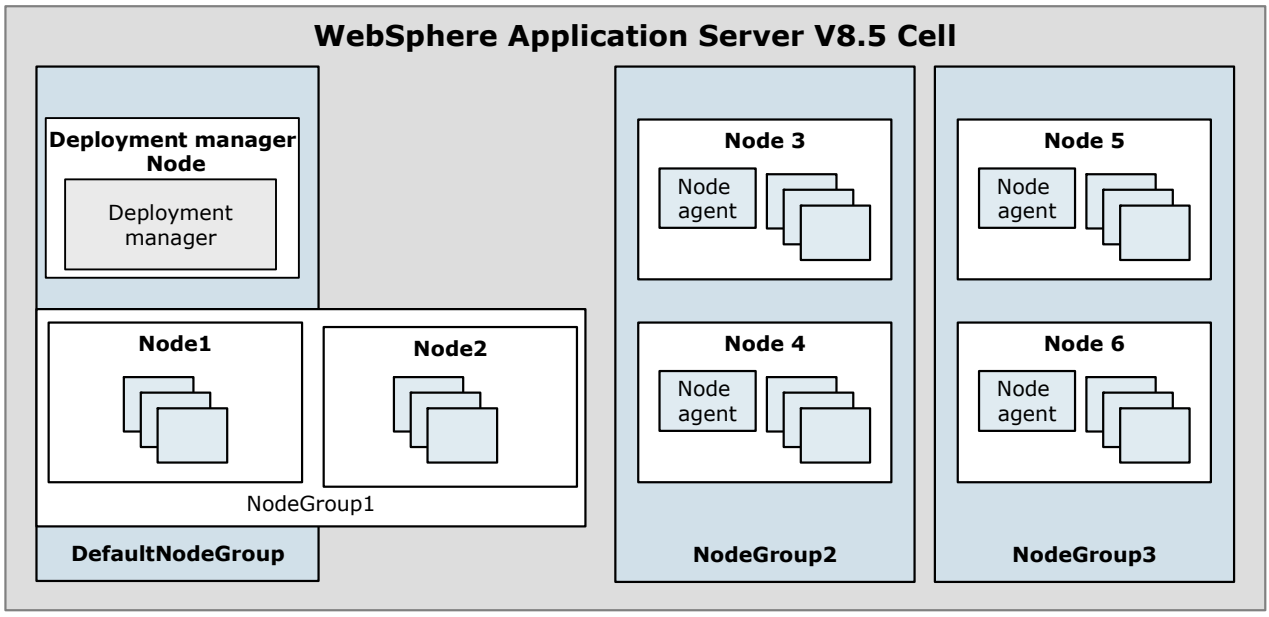


Figure 2-11 Examples of a node and node group

2.1.6 Cells

A *cell* is a grouping of nodes into a single administrative domain. A cell encompasses the entire management domain. In the Base and Express configurations, a cell contains one node, and that node contains one server. The left side of Figure 2-12 illustrates a system with two cells that are each accessed by their own administrative console. Each cell has a node and a stand-alone application server.

In a Network Deployment environment (the right side of Figure 2-12), a cell can consist of multiple nodes and node groups. These nodes and groups are all administered from a single point, the deployment manager. Figure 2-12 shows a single cell that spans two systems that are accessed by a single administrative console. The deployment manager is administering the nodes.

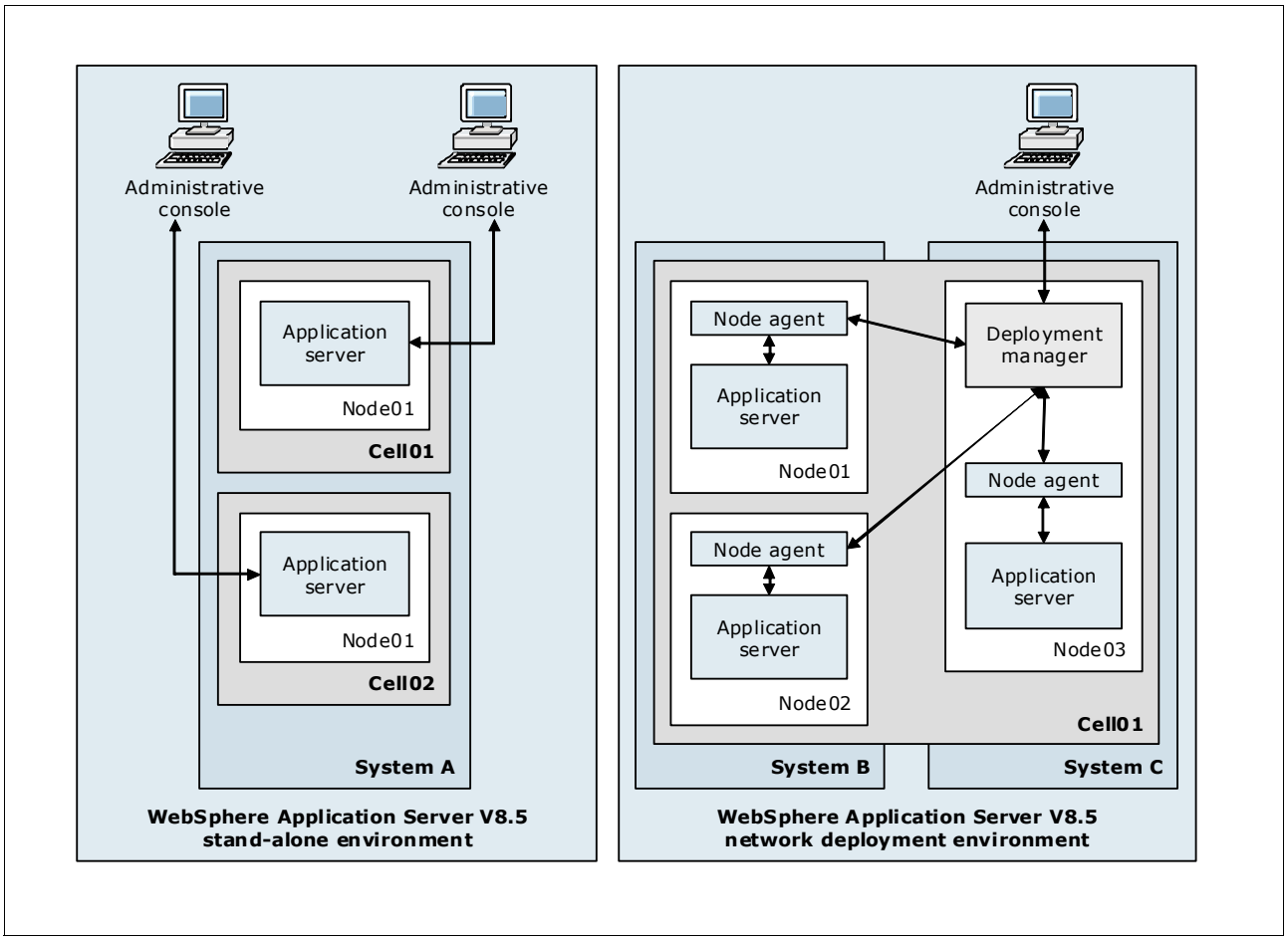


Figure 2-12 Cells representation in stand alone and network deployment environments

A cell configuration that contains nodes that are running on the same platform is called a *homogeneous cell*.

It is also possible to configure a cell that consists of nodes on mixed platforms. With this configuration, other operating systems can exist in the same WebSphere Application Server cell. Cells can span z/OS sysplex environments and other operating systems. For example, z/OS nodes, Linux nodes, UNIX nodes, and Windows system nodes can exist in the same WebSphere Application Server cell. This configuration is called a *heterogeneous* cell. A heterogeneous cell requires significant planning.

Figure 2-13 shows a heterogeneous cell, where node groups are defined for different operating systems.

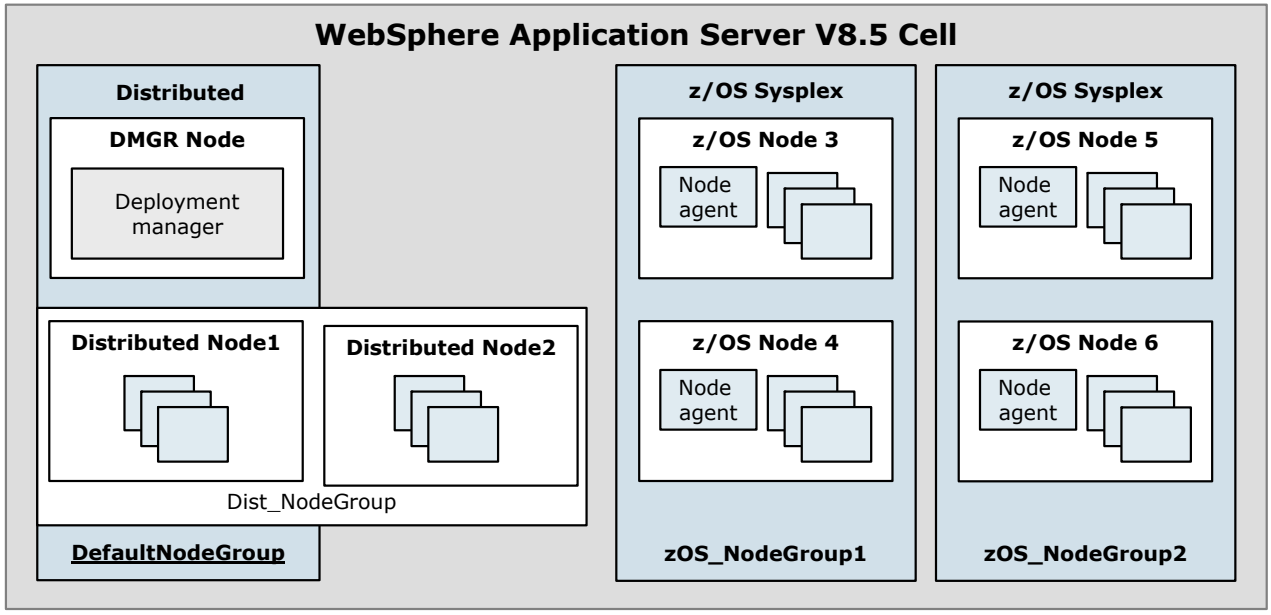


Figure 2-13 A heterogeneous cell with the coexistence of distributed and z/OS nodes

2.1.7 Deployment manager

The *deployment manager* is the central administration point of a cell that consists of multiple nodes and node groups in a distributed server configuration. It is similar to the configuration shown in Figure 2-10 on page 35. The deployment manager communicates with the node agents of the cell that it is administering to manage the applications servers within the node. The deployment manager provides management capability for multiple federated nodes, and can manage nodes that span multiple systems and platforms. A node can be managed by a single deployment manager, and the node must be federated to the cell of that deployment manager.

The configuration and application files for all nodes in the cell are centralized into the *master repository*. This centralized repository is managed by the deployment manager and regularly synchronized with local copies that are held on each of the nodes. If the deployment manager is not available in the cell, the node agents and the application servers cannot synchronize configuration changes with the master repository. This limitation continues until the connection with deployment manager is reestablished.

Version note: A high availability deployment manager is available in WebSphere Application Server V8.5. You can configure a hot-standby deployment manager to recover failures of the currently active deployment manager.

2.2 Additional concepts for WebSphere Application Server

This section provides information about the following additional concepts for WebSphere Application Server:

- Administrative agent in a stand-alone application server environment
- Job manager
- Web servers
- Web server plug-in
- Proxy servers
- Generic servers
- The centralized installation manager
- Intelligent runtime provisioning
- Intelligent Management
- Batch processing

2.2.1 Administrative agent in a stand-alone application server environment

An *administrative agent* (Figure 2-14) is a component that provides enhanced management capabilities for stand-alone application servers. All configurations that are related to the application server are connected directly to the administrative agent that provides services to administrative tools.

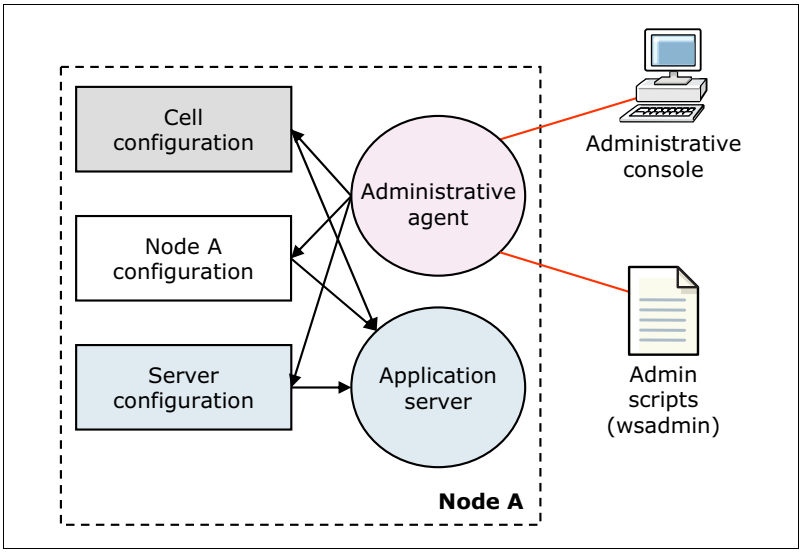


Figure 2-14 Administrative agent in a stand-alone configuration

An administrative agent can manage multiple stand-alone server instances on a single system or z/OS image. When using an administrative agent, because the number of application server instances increases, the redundancy of the administration footprint (for each application server) is eliminated. The administrative agent acts as the main component for the expanded multiple node remote management environment that is provided with the job manager.

When working with the administrative agent, consider the following circumstances:

- The administrative agent manages only application servers that are installed in the same operating system image as the administrative agent.
- The administrative agent provides only the management of these application servers and their applications. It does not provide clustering and failover capabilities. Clustering, failover, and centralized application management are available only in a WebSphere Application Server Network deployment.

2.2.2 Job manager

The *job manager* is a component that provides management capabilities for multiple stand-alone application servers, administrative agents, and deployment managers. With this component, you can submit administrative jobs asynchronously for application servers registered to administrative agents, for deployment managers, and for host systems. These jobs can be submitted to many servers over a geographically dispersed area. Host computers are registered with job manager to enable job manager to access applications, command files, and other resources on the host computer.

After you register stand-alone application servers, deployment managers, or host computers as *targets*, you can queue administrative jobs directed at the targets through the job manager. Many of the management tasks that you can perform with the job manager are tasks that you can already perform with the product. These tasks include application management, server management, and node management. You can complete job manager actions and easily run jobs from a deployment manager. The deployment manager administrative console has Jobs navigation tree choices where you can asynchronously administer job submissions.

A job manager can also submit jobs to the WebSphere Application Server Liberty profiles.

Figure 2-15 illustrates the job manager architecture.

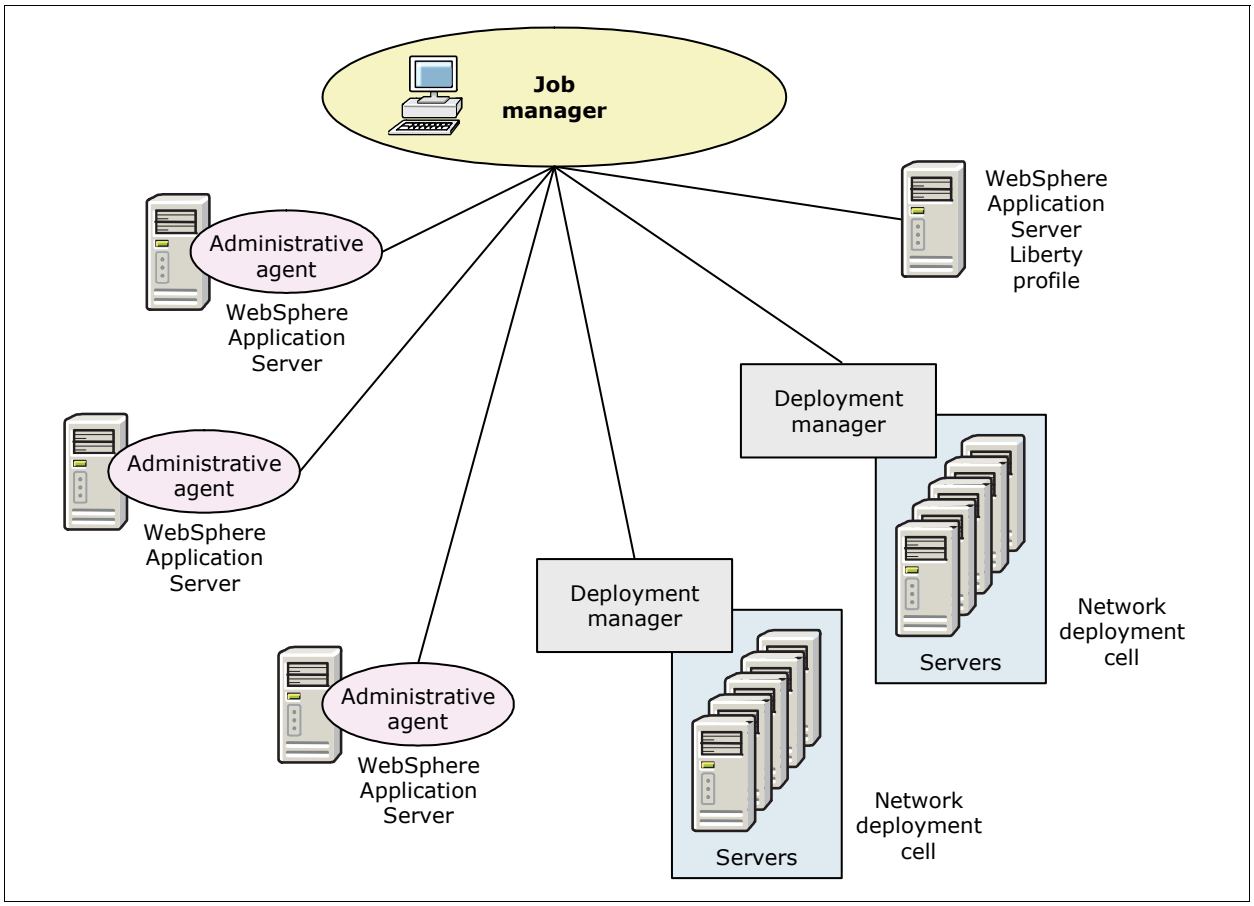


Figure 2-15 Job manager architecture

The job manager is available only with WebSphere Application Server Network Deployment and WebSphere Application Server for z/OS.

2.2.3 Web servers

Although web servers are independent products, they can be defined to and managed by the administration processes of WebSphere Application Server. This approach enables the administrator to associate applications with one or more defined web servers. Doing so generates the correct routing of information for web server plug-ins if multiple servers are used.

When you define a web server to Websphere Application Server, it is associated with a node. The node is considered either *managed* or *unmanaged*. A managed web server is a web server that is defined on a managed node. An unmanaged web server is on an unmanaged node.

Restriction: In a stand-alone server environment, you can define a single unmanaged web server. In a distributed environment, you define multiple managed or unmanaged web servers.

Managed nodes

Managed nodes have a node agent on the web server system that allows the deployment manager to administer the web server. You can start or stop the web server from the deployment manager, generate the web server plug-in for the node, and automatically push the plug-in to the web server. In most installations, managed web server nodes are behind the firewall with WebSphere Application Server installations. Figure 2-16 illustrates a managed server on a managed node.

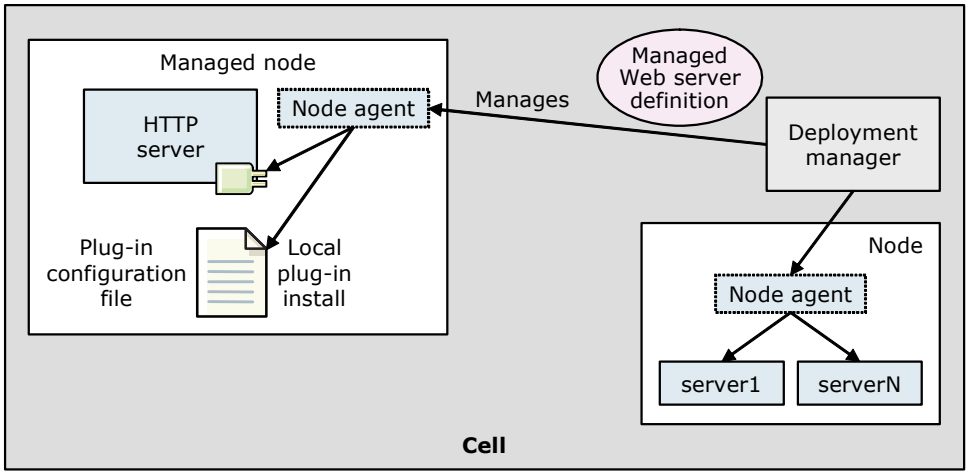


Figure 2-16 Managed web server on a managed node

Unmanaged nodes

Unmanaged nodes are not managed by WebSphere Application Server. You usually find these nodes outside the firewall or in the DMZ. You must manually transfer the web server plug-in configuration file to the web server on an unmanaged node. In a z/OS environment, you must use unmanaged nodes if the web server is not running on the z/OS platform. Figure 2-17 illustrates this configuration.

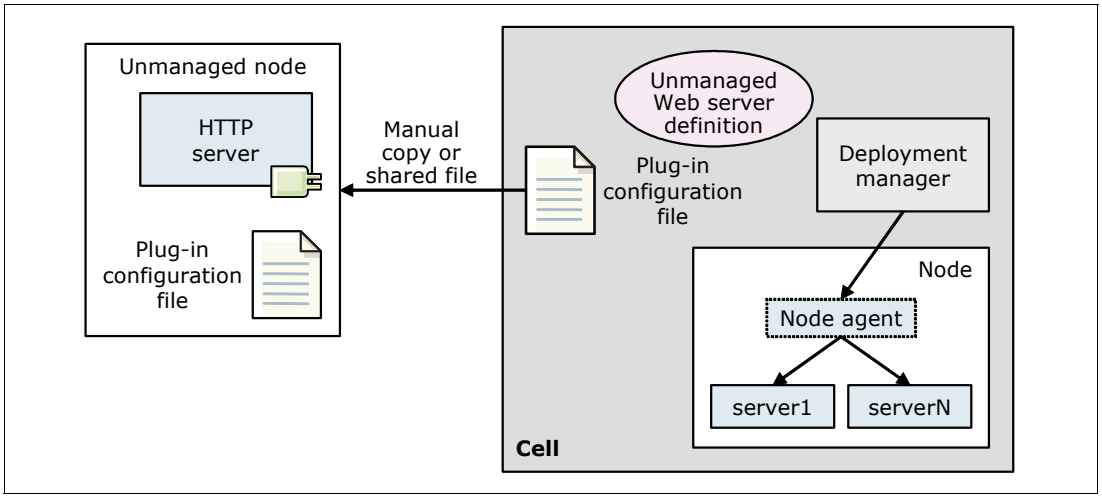


Figure 2-17 Unmanaged web server on a managed node

An IBM HTTP Server on an unmanaged node can be administered from the administrative console. However, this method is considered a special case. With this configuration, the administrator can automatically push the plug-in configuration file to the web server with the

deployment manager by using HTTP commands (Figure 2-18). This configuration does not require a node agent. IBM HTTP Server is shipped with all WebSphere Application Server packages.

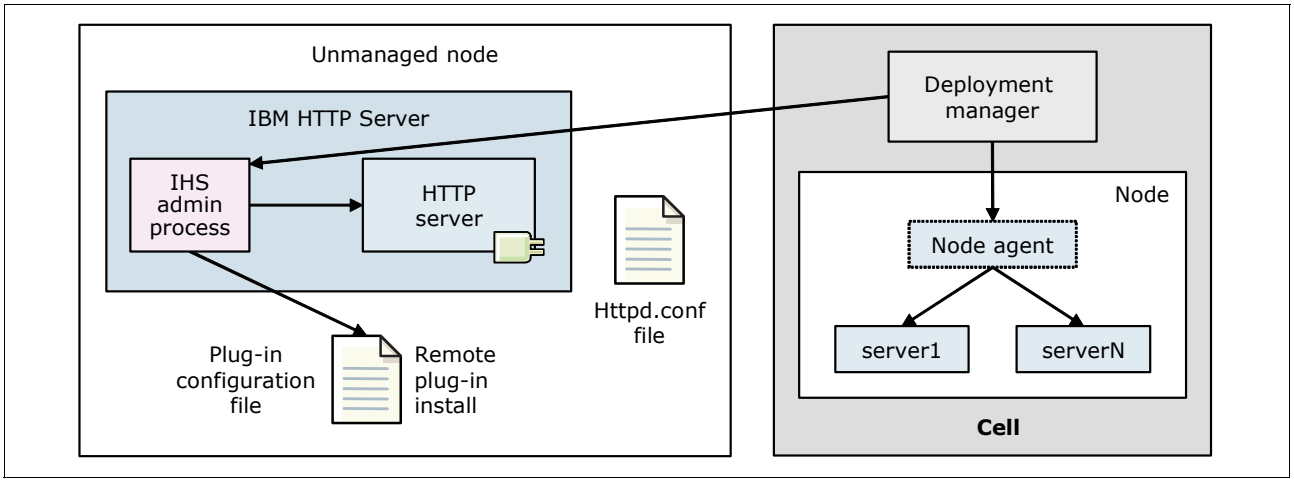


Figure 2-18 IBM HTTP Server on an unmanaged node

Remote web servers

You can create a web server definition in the administrative console. Do so when the web server and the web server plug-in for WebSphere Application Server are on the same system and the application server is on a different system. This configuration allows you to run an application server on one platform and a web server on another platform.

With a remote web server installation, WebSphere Application Server can facilitate plug-in administration functions, and generation and propagation of the `plugin-cfg.xml` file for IBM HTTP Server for WebSphere Application Serve. However, it cannot do this for other web servers.

You can choose a remote web server installation if you want the web server outside of a firewall and WebSphere Application Server on the inside. You can create a remote web server on an unmanaged node. Because there is no WebSphere Application Server or node agent on the system that the node represents, you cannot administer a web server on that unmanaged node unless the web server is IBM HTTP Server for WebSphere Application Server. With IBM HTTP Server, there is an administration server that facilitates administrative requests such as start and stop, view logs, and view and edit the `httpd.conf` file.

Important: The administration server is not provided with IBM HTTP Server for WebSphere Application Server that runs on z/OS platforms. Therefore, administration using the administrative console is not supported for IBM HTTP Server for z/OS on an unmanaged node.

For a web server that is not an IBM HTTP Server on an unmanaged node, you can generate a plug-in configuration based on WebSphere Application server repository changes. However, some functions are not supported on an unmanaged node for a web server that is not an IBM HTTP Server.

2.2.4 Web server plug-in

A web server can serve static contents and requests, such as HTML pages. However, when a request requires dynamic content, such as JSP or servlet processing, it must be forwarded to WebSphere Application Server for handling. The *web server plug-in* is used to route requests to one of multiple application servers, as illustrated in Figure 2-19.

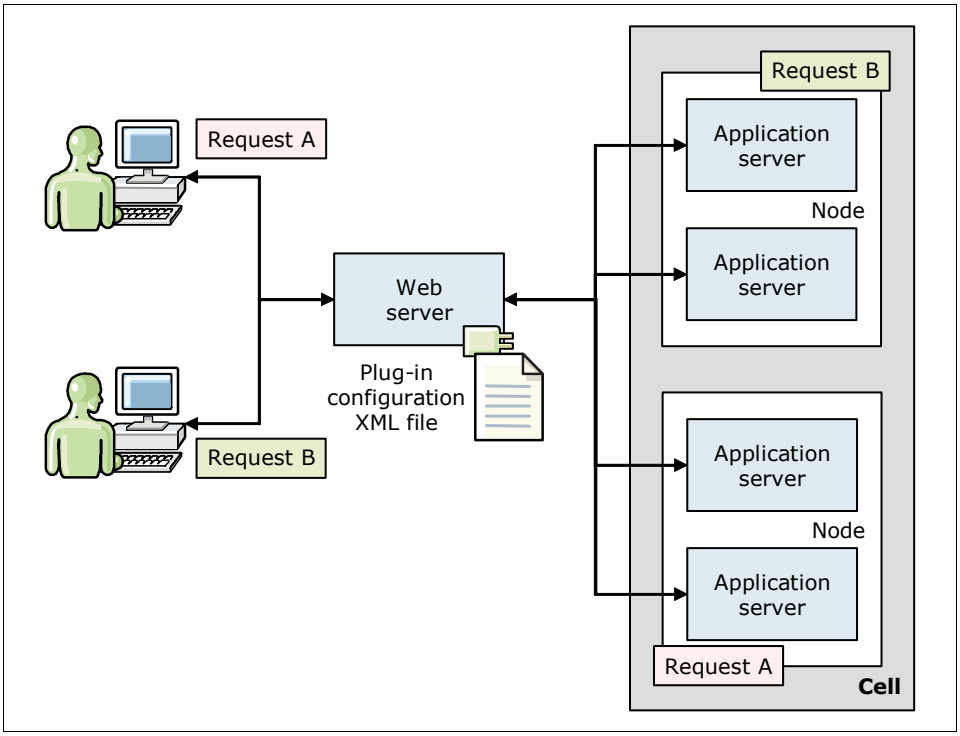


Figure 2-19 Web server plug-in and plug-in configuration file concept

The plug-in is included with all WebSphere Application Server packages for installation on a web server. The plug-in uses the plug-in configuration file to determine whether a request is handled by the web server or forwarded to an application server. The request can be transmitted by the plug-in by using either HTTP or HTTP Secured (HTTPS).

The plug-in configuration file is an XML file generated by WebSphere Application Server, propagated to the web server, and stored in the plug-in directory of the web server.

2.2.5 Proxy servers

A *proxy server* is a specific type of application server that routes HTTP requests to content servers that perform the work. The *reverse proxy server* is the initial point of contact, after the protocol firewall, for client requests that enters *into* the enterprise server. By contrast, a *forward proxy server* acts as the first point of contact for *outbound* traffic.

With WebSphere Application Server, you can create the following types of proxy servers:

- WebSphere Application Server Proxy
- DMZ Secure Proxy Server

WebSphere Application Server Proxy

A WebSphere Application Server Proxy provides many functions that a web server and the plug-in have. However, it is not a full replacement for the plug-in because it does not have web serving capabilities. Static content can be served directly from the proxy cache. If the web server is used only for load balancing and routing with session affinity, WebSphere Application Server Proxy can take the place of the web server.

WebSphere Application Server Proxy is not considered a secure proxy for DMZ deployments. For example, it cannot bind to protected ports without being a privileged user on most operating systems, and users cannot be switched after binding. WebSphere Application Server Proxy must stay in the intranet or secure zone. WebSphere Application Server V8.5 ships a DMZ-hardened version of WebSphere Application Server Proxy. For more information, see “DMZ Secure Proxy Server” on page 46.

WebSphere Application Server Proxy in V8.5 supports the HTTP and SIP protocols. You can configure WebSphere Application Server Proxy to use one or both of these protocols. This proxy server is used to classify, prioritize, and route HTTP and SIP requests to servers in the enterprise, and to cache content from servers.

HTTP proxy

The HTTP proxy server acts as a surrogate for content servers within the enterprise. As a surrogate, you can configure the proxy server with rules to route to and load balance the clusters of content servers. The proxy server is also capable of securing the transport by using Secure Sockets Layer (SSL). Content is secured by using various authentication and authorization methods. Another important feature is its capability to protect the identity of the content servers from the web clients by using response transformations (URL rewriting). The proxy server can also improve performance by caching content locally and protecting the content servers from surges in traffic.

You can modify an existing proxy server to run advanced routing options, such as routing requests to application servers that are not the WebSphere Application Server. You can also modify a proxy server to run caching.

Remember: When using WebSphere Application Server for z/OS V8.5, the proxy server uses the Workload Management component to run dynamic routing.

SIP proxy

The SIP proxy design is based on the HTTP proxy architecture. The SIP proxy extends the HTTP proxy features. It can be considered a peer to the HTTP proxy because both the SIP and the HTTP proxy run within the same proxy server. Both rely on a similar filter-based architecture for message processing and routing.

The SIP proxy server initiates communication and data sessions between users. It delivers a high performance SIP proxy capability. You can use this capacity at the edge of the network to route, load balance, and improve response times for SIP dialogs to back-end SIP resources. The SIP proxy provides a mechanism for other components to extend the base function and support additional deployment scenarios.

The SIP proxy is responsible for establishing outbound connections to remote domains on behalf of the back-end SIP containers and clients located within the domain hosted by the proxy. Another important feature of the SIP proxy is its capability to protect the identity of the back-end SIP containers from the SIP clients.

DMZ Secure Proxy Server

Because the WebSphere Application Server Proxy is not ready for a DMZ, WebSphere Application Server V8.5 ships a DMZ-hardened version of WebSphere Application Server Proxy. The *DMZ Secure Proxy Server* comes in a separate installation package that contains a subset of WebSphere Application Server Network Deployment. The package provides security enhancements that allow deployments inside a DMZ as illustrated in Figure 2-20.

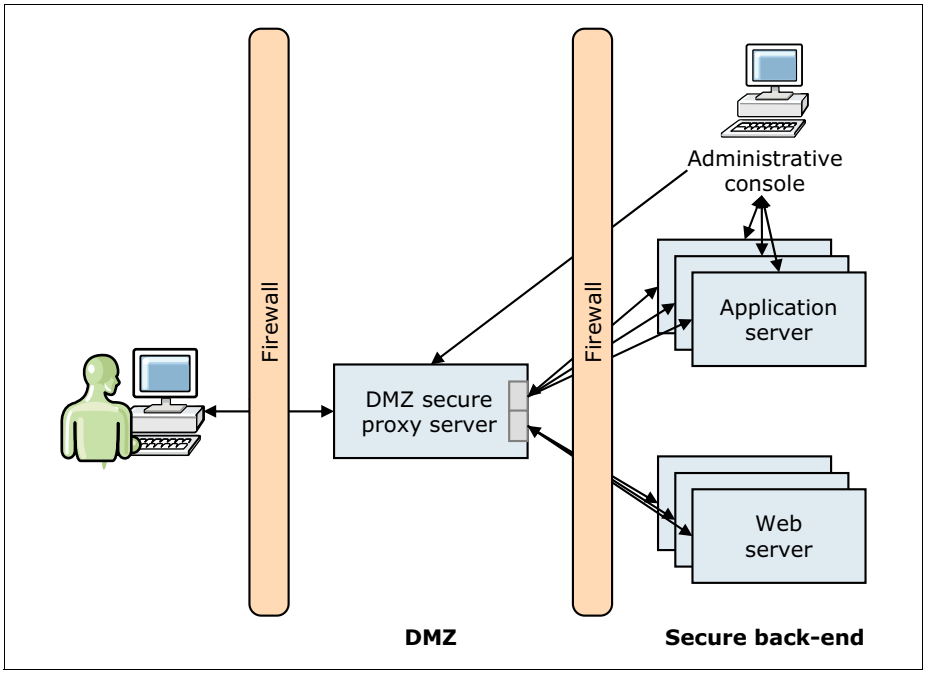


Figure 2-20 DMZ secure proxy simplified topology

The DMZ Secure Proxy Server can be managed locally or remotely by using the job manager console.

HTTP and SIP protocol support: The HTTP and SIP protocols that are supported by WebSphere Application Server Proxy are also supported by the DMZ Secure Proxy Server.

For a sample topology that uses the DMZ Secure Proxy Server as a reverse proxy, see 8.1.2, “Reverse proxies” on page 180.

2.2.6 Generic servers

A *generic server* is a server that is managed in the administrative domain of WebSphere Application Server, even though the server is not supplied by WebSphere Application Server. You can define a generic server as an application server instance within the WebSphere Application Server administration and associate it with a non-WebSphere Application Server or process.

The following basic types of generic application servers are available:

- Applications or processes that are not based on Java
- Java applications or processes

A generic server can be any server or process that is deemed necessary to support the application server environment:

- C or C++ server or process
- CORBA server
- Java server
- Remote Method Invocation (RMI) server

For more information about creating generic application servers and non-Java applications as a generic server, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=trun_genericsvr_create

2.2.7 The centralized installation manager

The centralized installation manager simplifies the installation and maintenance of application servers.

As an administrator, from the deployment manager you can remotely install or uninstall product packages and apply maintenance to specific nodes directly from the administrative console. This process allows you to avoid having to log in and repetitively perform these tasks. Using the centralized installation manager, you can reduce the number of steps required to create and manage the environment. This reduction can simplify installation and patch management.

Tip: There are two versions of the centralized installation manager in WebSphere Application Server V8.5. The previous centralized installation manager is used to install and maintain WebSphere Application Server V7 installations. It is available on the system administration section of the deployment manager administrative console. The centralized installation manager introduced in WebSphere Application Server V8 is available at the jobs section of the deployment manager or job manager administrative console. It is used to install or maintain WebSphere Application Server V8 and later.

The centralized installation manager does not replace the product installation wizard for WebSphere Software. As shown in Figure 4-21, the centralized installation manager remotely drives IBM Installation Manager to run installations and maintenance.

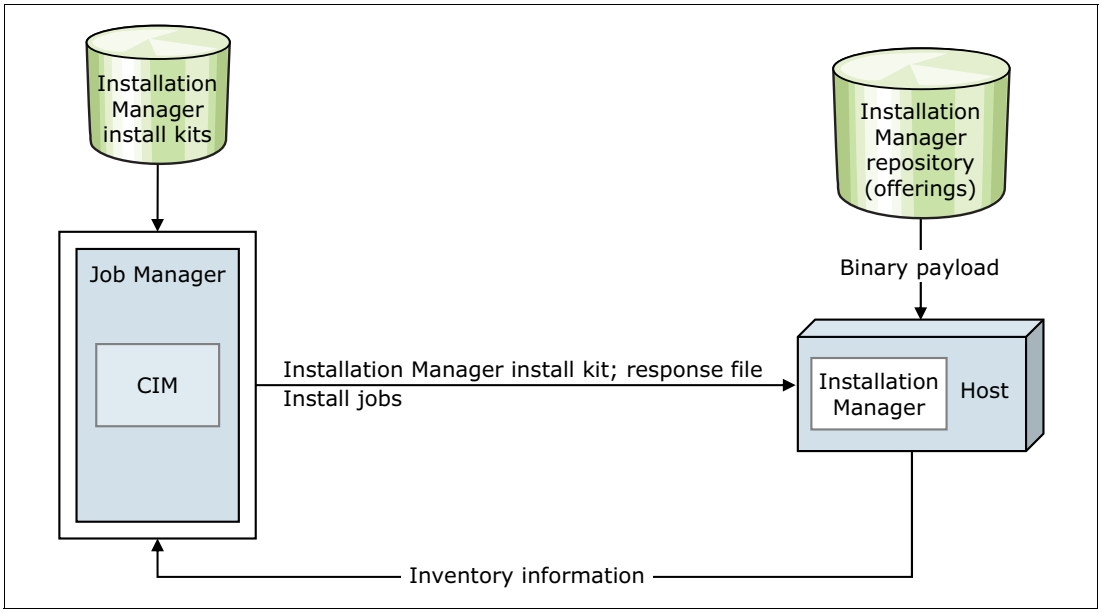


Figure 2-21 Centralized installation manager architecture in WebSphere Application Server V8.5

You can install or uninstall the following product packages and maintenance files with the centralized installation manager:

- IBM WebSphere Application Server
- IBM Installation Manager
- IBM HTTP Server
- Application clients
- DMZ secure proxy server
- Web server plug-in
- WebSphere Customization Toolbox

The centralized installation manager functions are integrated into the job manager through the administrative console. Thus, the centralized installation manager supports job scheduling, multiple cells, and better scalability. New centralized installation manager jobs can be submitted to run these and many other operations:

- Manage profiles
- Install SSH public keys
- Run commands on remote hosts
- Test connections
- Distribute files

2.2.8 Intelligent runtime provisioning

Intelligent runtime provisioning is a concept that was introduced with WebSphere Application Server V7. This mechanism selects only the runtime functions that are needed for an application. Each application is examined by WebSphere Application Server during deployment to generate an activation plan as shown in Figure 2-22. At run time, the application server uses the activation plan to start only those components that are required inside the application server. This process can reduce the runtime footprint, and can significantly reduce startup times.

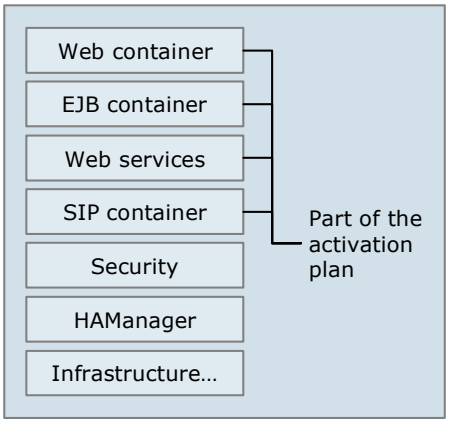


Figure 2-22 Intelligent runtime provisioning

2.2.9 Intelligent Management

The Intelligent Management functions allow advanced management and virtualization capabilities. It provides dynamic runtime capabilities by using an on-demand router server, which is an intelligent HTTP and SIP proxy server. The Intelligent Management capabilities bring autonomic computing, allowing the environment to be self-configuring, self-healing, self-protecting, and self-optimizing.

The Intelligent Management functionality is a collection of the following primary features:

- *Intelligent routing* improves business results by ensuring priority is given to business critical applications by using the on-demand router.
- *Health management* allows you to specify conditions to watch for and corrective actions to take when the conditions are observed.
- *Application edition management* allows you to roll out new versions of applications without experiencing downtime for a maintenance window.
- *Performance management* is self optimizing and automatically scales up and down the number of running cluster members as needed to meet response time goals for users.

Intelligent Management uses the following autonomic managers as part of the dynamic operation functionality:

- Autonomic Request Flow Manager
- Dynamic Workload Controller
- Application Placement Controller
- On Demand Configuration Manager

For more information, see Chapter 5, “Intelligent Management” on page 107.

2.2.10 Batch processing

The *job scheduler* is a system component responsible for delivering batch application job submissions to eligible application servers for execution.

The job scheduler provides API, command line (**`lrcmd`**/**`wsgrid`**), and web GUI (Job Management Console) interfaces that support all forms of interaction with batch application jobs.

Figure 2-23 shows a typical configuration for the job scheduler in a dedicated WebSphere application server or cluster in a Network Deployment cell configuration. It can also be used in a stand-alone application server to provide an independent, self-contained batch application job processing environment.

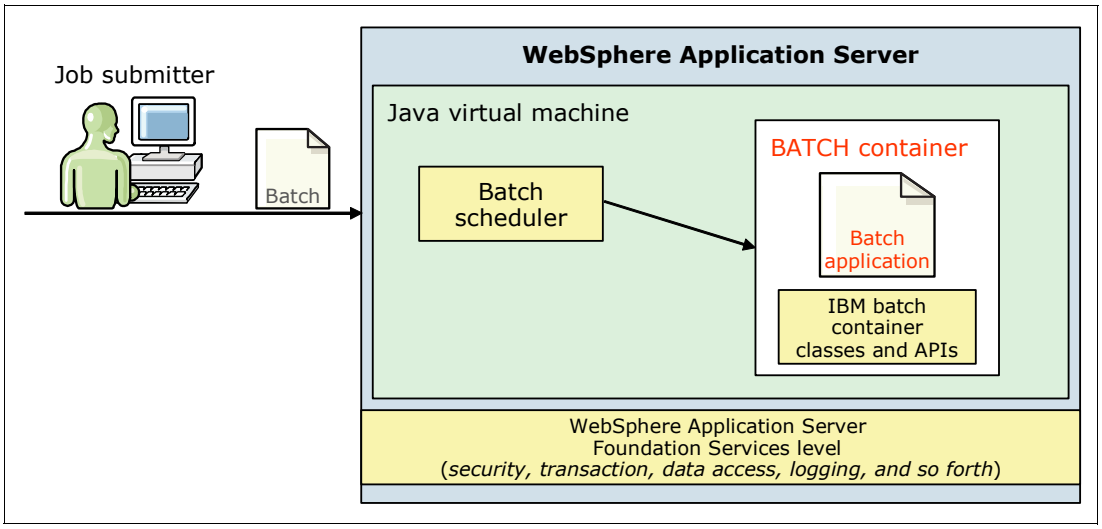


Figure 2-23 WebSphere Batch processing diagram flow

For more information on Batch, see Chapter 6, “WebSphere Batch” on page 137.

2.3 Server configurations

With WebSphere Application Server, you can build various server environments that consist of single and multiple application servers that are maintained from central administrative points. This section provides information about the following configurations that you can create by using WebSphere Application Server V8.5:

- Single cell configurations
- Flexible management configurations

2.3.1 Single cell configurations

A cell (see 2.1.6, “Cells” on page 37) groups nodes into a single administrative domain. With WebSphere Application Server, you can create two types of configurations in a single cell environment:

- Single system configurations
- Multiple systems configurations

A *system* is defined as one of the following types:

- A server system (a physical machine) that contains only one operating system
- An operating system virtual image where the host server system contains multiple operating system images
- A z/OS image

Single system configurations

With the Base, Express, and Network Deployment packages, you can create a cell that contains only a single node with a single application server (Figure 2-24).

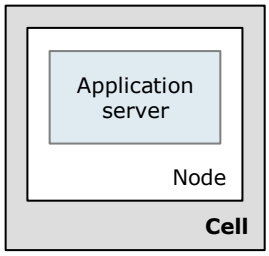


Figure 2-24 Single cell configuration in Base and Express packages

Single system is the only configuration option with Base and Express. The cell is created when you create the stand-alone application server profile.

A node agent at each node is the contact point for the deployment manager during cell administration. A single system configuration in a distributed environment includes all processes in one system as illustrated in Figure 2-25.

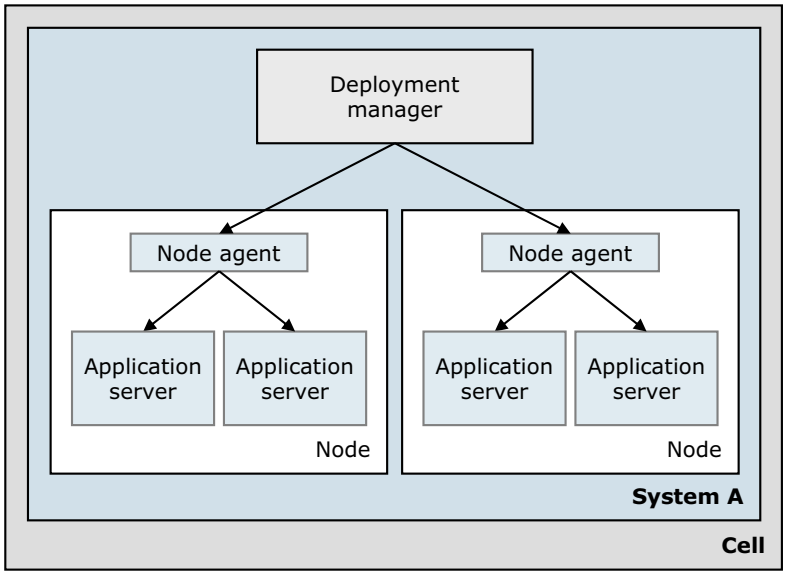


Figure 2-25 Cell configuration option in Network Deployment: Single system

Multiple system configurations

A Network Deployment environment allows you to install the WebSphere Application Server components on systems and locations that suit your requirements. With the Network Deployment package, you can create multiple systems configurations.

Figure 2-26 shows the deployment manager that is installed on one system (System A) and each node on a different system (System B and System C). The servers can be mixed platforms or the same platform. In this example, System A can be an IBM AIX® system, System B can be a Windows operating system, and System C can be a z/OS image.

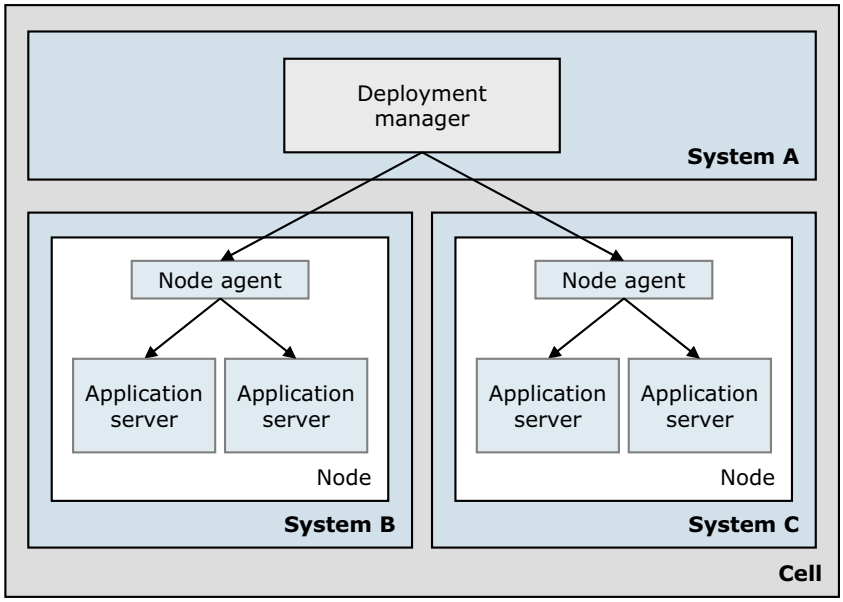


Figure 2-26 Cell configuration option in Network Deployment: Multiple systems

Using the same logic, other combinations can be installed. For example, you can install the deployment manager and a node on one system with additional nodes installed on separate systems.

2.3.2 Flexible management configurations

With *flexible management* components, such as the administrative agent and the job manager, you can build advanced and large-scale topologies. You can also manage single and multiple application server environments from a single point of control. This capability reduces management and maintenance complexity.

Multiple base profiles

The administrative agent component of WebSphere Application Server provides administration services and functions to manage multiple stand-alone application servers that are all installed in the same system. Figure 2-14 on page 39 shows the administrative agent management model.

It is possible to manage multiple administrative agents with a job manager. These administrative agents can be on one or multiple systems.

Job manager management model and advanced topologies

By using the job manager component of WebSphere Application Server Network Deployment, you can build advanced management models and topologies for your environment.

The job manager can manage multiple administrative agents in different systems, and can be the single point of control for these stand-alone server profiles (Figure 2-27).

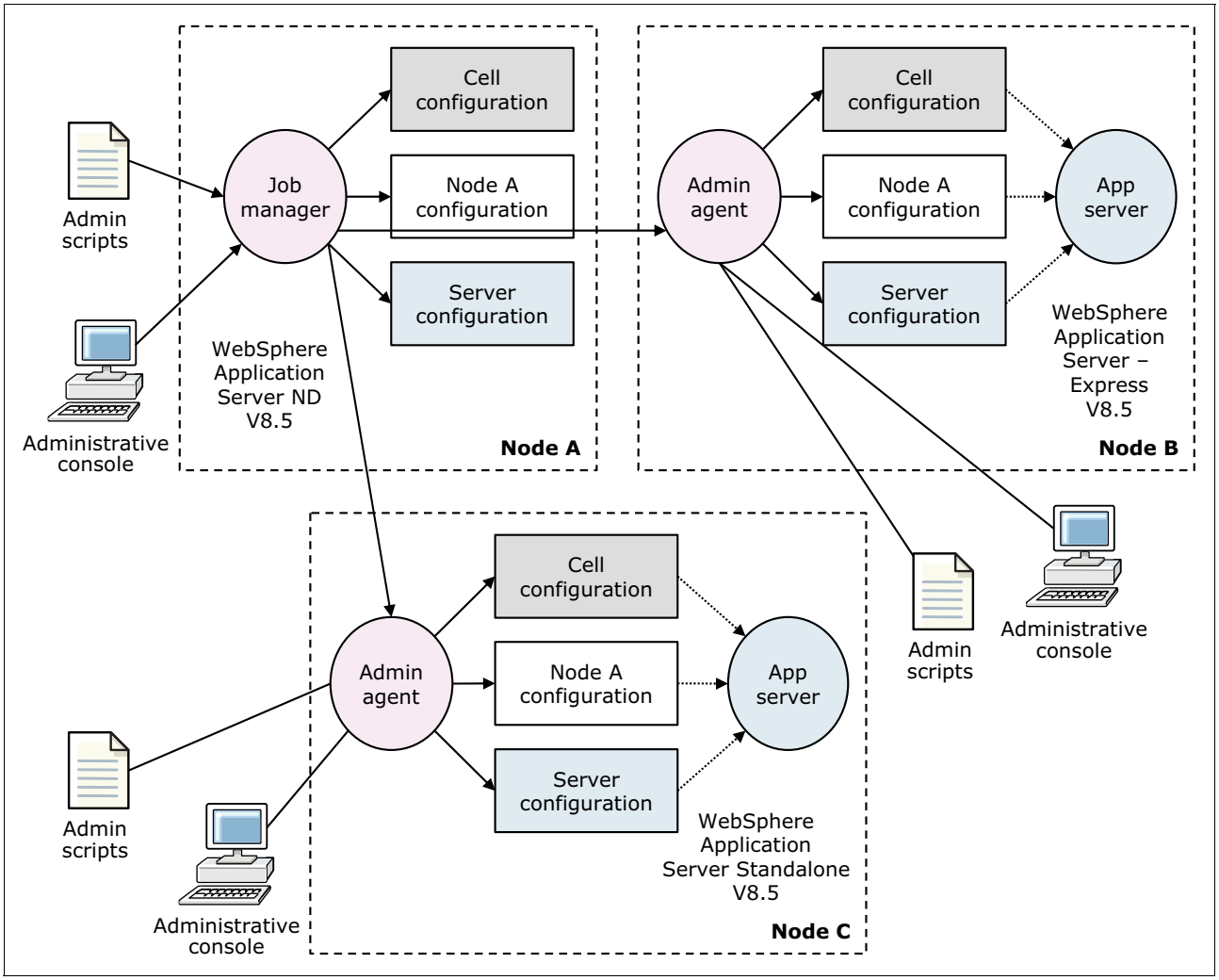


Figure 2-27 Job manager management model for multiple administrative agents

The job manager provides a loosely coupled management architecture. Rather than synchronously controlling a number of remote endpoints (node agents), the job manager coordinates management across a group of endpoints. It does so by providing an asynchronous job management capability across several nodes.

The advanced management model relies on the submission of management jobs to these remote endpoints, which can be either an administrative agent or deployment manager. In turn, the administrative agent or the deployment manager runs the jobs that update the configuration, starts or stops applications, and runs various other common administrative tasks.

A job manager is created with a job manager profile. The job manager can manage nodes that span multiple systems and platforms. A node managed by one job manager can also be managed by multiple job managers.

Important: The job manager is not a replacement for a deployment manager. It is an option for remotely managing a deployment manager or, more likely, multiple deployment managers, removing the cell boundaries for administration.

2.4 Security

WebSphere Application Server provides a set of features to help you to secure your systems and associated resources. Figure 2-28 illustrates the components that make up the operating environment for security in WebSphere Application Server.

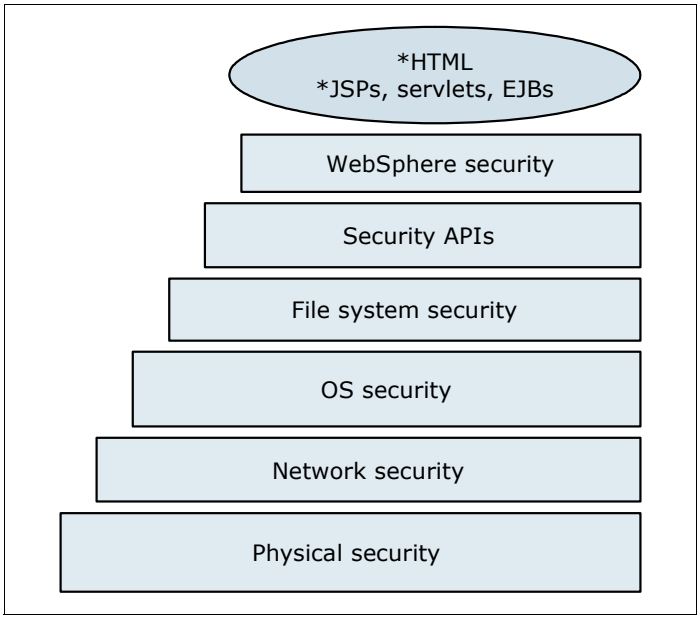


Figure 2-28 WebSphere Application Server security overview

From a broad perspective, a WebSphere security service runs locally in each process. This architecture, illustrated in Figure 2-29, distributes the security workload so that one process does not act as a bottleneck for the entire environment. If a security service failure occurs, only a single process is affected.

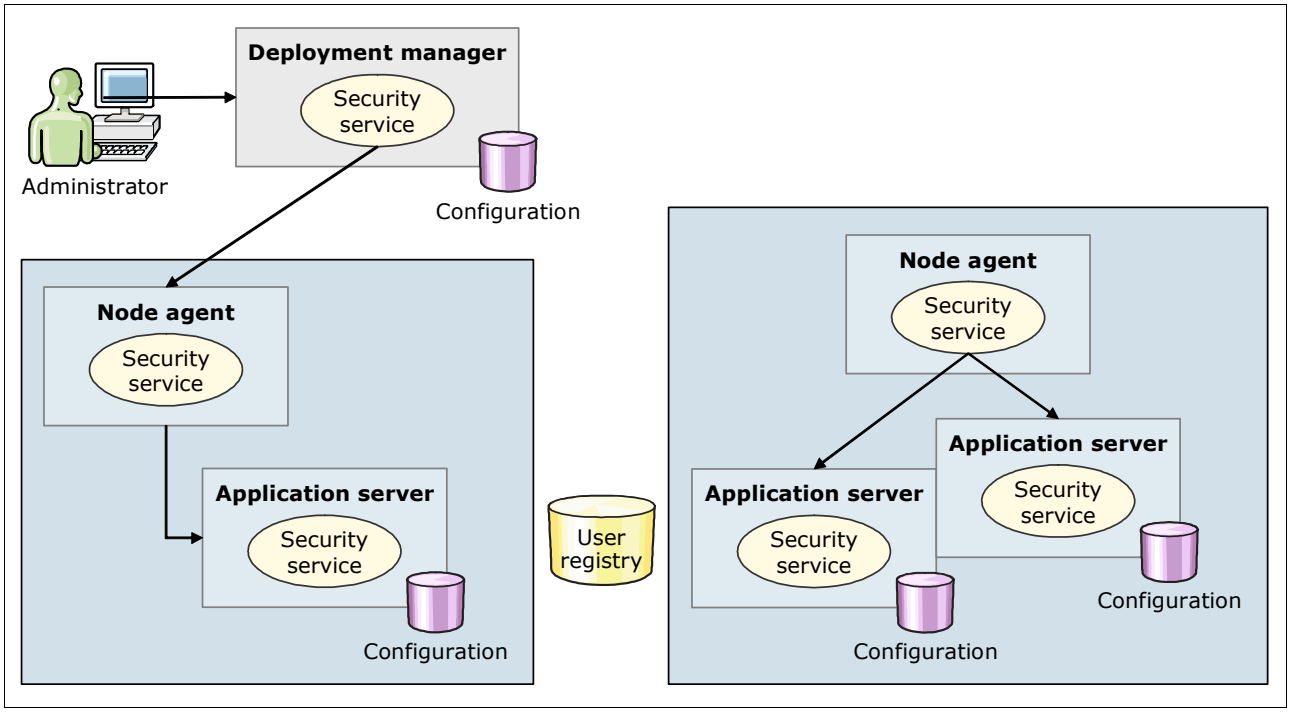


Figure 2-29 Overview of WebSphere security service

2.4.1 Security types

The security infrastructure of WebSphere Application Server is broadly divided into the following types of security (illustrated in Figure 2-30 on page 56) from the administrative console:

- *Administrative security* protects resources such as the administrative console, **wsadmin**, and scripts. When administrative security is enabled, naming security, authentication of HTTP clients, and use of SSL transports are also enabled.
- *Application security* protects access to applications. It provides application isolation and requirements for authenticating application users. These functions are done through security constraints to protect servlets and method permissions to protect EJB. Application security can be applied to resources within an EAR file through security roles defined in the deployment descriptor of the application. Security roles can then be mapped to actual users and groups during application deployment.
- *Java 2 security* protects the local system from applications that are deployed to WebSphere Application Server. When Java 2 security is enabled, it provides an access control mechanism to manage the access of an application to system-level resources.

Figure 2-30 Types of security

2.4.2 Authentication

Authentication is the process of identifying who is requesting access to a resource. For the authentication process, the server implements a challenge mechanism to gather unique information to identify the client. Secure authentication can be knowledge-based (user and password), key-based (physical and encryption keys), or biometric.

The authentication mechanism in WebSphere Application Server typically collaborates closely with a user registry, as illustrated in Figure 2-31. When running authentication, the user registry is consulted. A successful authentication results in the creation of a *credential*, which is the internal representation of a successfully authenticated client user. The abilities of the credential are determined by the configured authorization mechanism. You can configure a user registry to Lightweight Directory Access Protocol (LDAP), a file registry, a database, and so on.

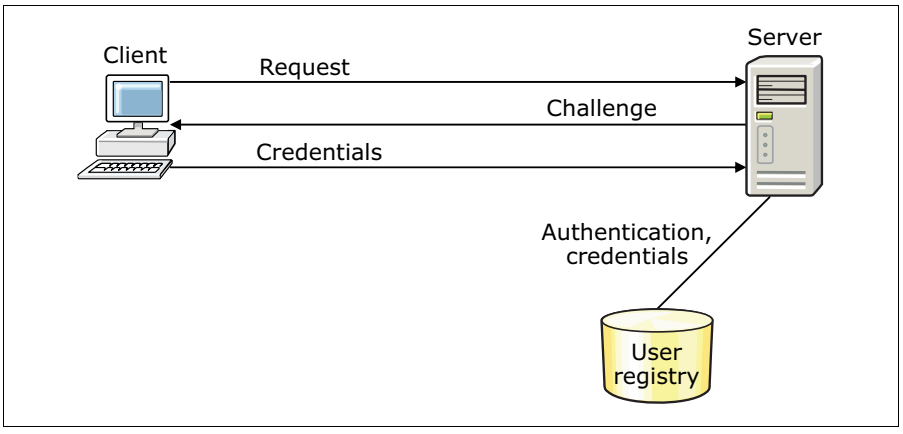


Figure 2-31 Simplified view of authentication

Websphere Application Server V8.5 provides support for multiple authentication mechanisms. You can configure the environment to have more than one active authentication mechanism at a time.

WebSphere Application Server supports the following authentication mechanisms:

- Lightweight Third Party Authentication (LTPA)

  LTPA provides the ability for single sign-on (SSO) and forwardable credentials that can be sent to your application or between multiple applications or products. LTPA can support security in a distributed environment through cryptography. With this support, LTPA can digitally sign, securely transmit authentication-related data (encrypted tokens), and later verify the signature.

- Kerberos

  Kerberos is a mature, standard authentication mechanism that enables interoperability with other applications that support Kerberos authentication. It provides SSO and end-to-end interoperable solutions, and preserves the original requester identity.

- Rivest Shamir Adleman (RSA) token authentication

  The RSA token authentication mechanism aids the flexible management objective to preserve the configurations of base profiles and isolate them from a security perspective. With this mechanism, the base profiles managed by an administrative agent can have different LTPA keys, different user registries, and different administrative users. The RSA token authentication mechanism can be used only for administrative requests.

2.4.3 Authorization

Authorization is the process of checking whether a user has the privileges necessary to get access to or perform actions on a requested resource (Figure 2-32). These resources include web pages, servlets, JSP, and EJB. Authorization controls access to resources through the following mechanisms:

- *Security lookup* determines the security privileges for a user. This information is stored in a user registry.
- *Rule enforcement* obtains rules from a registry. Given the privileges of a user and rules, access is determined.

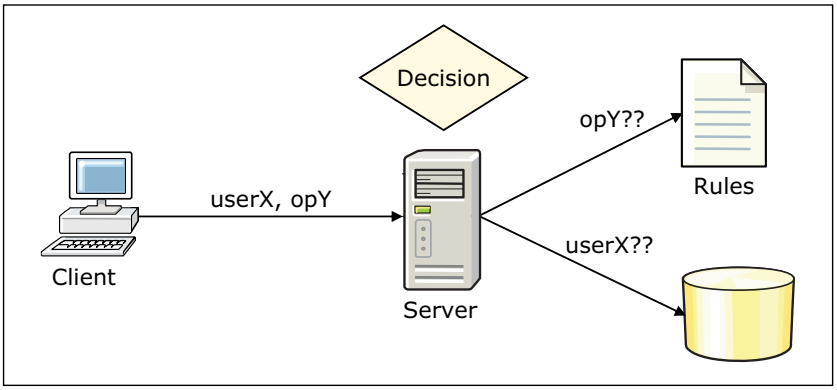


Figure 2-32 Simplified view of authorization

2.5 Service integration

Service integration technology provides the communication infrastructure for messaging and service-oriented applications, unifying this support into a common component. Service integration includes the following features:

- A JMS 1.1-compliant JMS provider

  This provider is called the *default messaging provider*.

- The service integration bus (called the *bus*)

  The bus provides the communication infrastructure for the default messaging provider and supports the attachment of web services requestors and providers.

- Support for the web services gateway

  This support provides a single point of control, access, and validation of web service requests. You can control which web services are available to different groups of web service users.

2.5.1 Default messaging provider

For messaging between application servers, you can use the default messaging provider. The default messaging provider supports JMS 1.1 common interfaces. With the default messaging provider, applications can use common interfaces for both point-to-point and publish/subscribe messaging. It also enables both point-to-point and publish/subscribe messaging within the same transaction.

2.5.2 Service integration bus

The *service integration bus* (bus) is the communication infrastructure that provides service integration through messaging. It is an administrative concept that is used to configure and host messaging resources. Service integration bus capabilities are fully integrated into WebSphere Application Server, enabling it to take advantage of WebSphere security, administration, performance monitoring, trace capabilities, and problem determination tools.

Figure 2-33 illustrates the service integration bus and how it fits into the larger picture of an enterprise service bus.

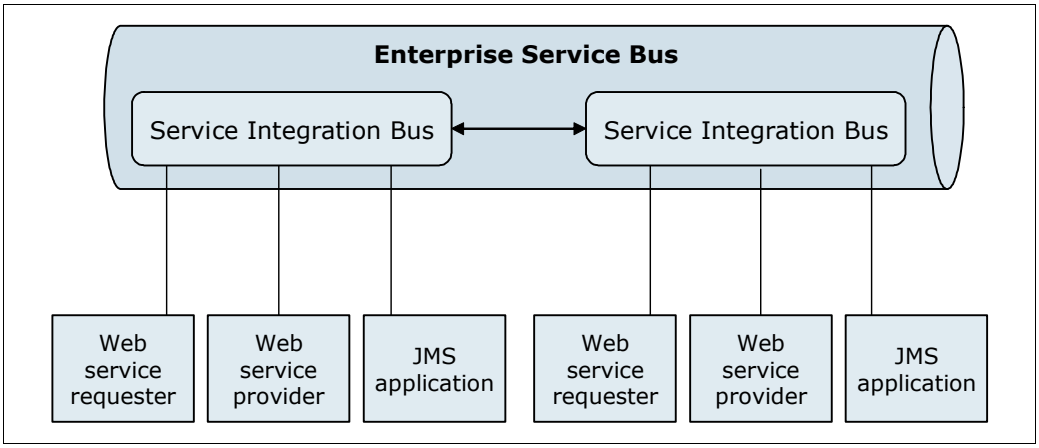


Figure 2-33 Service integration bus basics

The service integration bus supports the following application attachments:

- JMS messaging applications

  JMS applications that run in either WebSphere Application Server can connect to the bus by using the JMS programming model.

- Web services
  - Requestors using the JAX-RPC API
  - Providers running in WebSphere Application Server as stateless session beans and servlets (JSR-109)
  - Requestors or providers that attach through SOAP/HTTP or SOAP/JMS

Enhancements with WebSphere Application Server V8.5: The service integration bus is resilient to failures and can recover automatically from failures without administrator intervention. In a high availability environment, the service integration bus can determine where an active instance is hung. It can then transfer ownership to the standby messaging engine without depending on the active messaging engine to release the locks on the database.

For more information, see Chapter 13, “Messaging and service integration” on page 415.

2.5.3 Web services gateway

The *web services gateway* is a web services infrastructure component that is packaged with deployment manager. It is a SOAP processing engine that is focused on the operation of the intermediaries in the SOAP chain as illustrated in Figure 2-34. Typically, it does not act as an ultimate receiver, or as an initial sender of SOAP messages. Rather, it is a proxy for SOAP messages with the following capabilities:

- Alters the destination of a message (routing)
- Handles custom header tag processing
- Applies or removes message-level security (WS-Security)
- Runs protocol transformation, such as submitting incoming SOAP/HTTP messages to SOAP/JMS

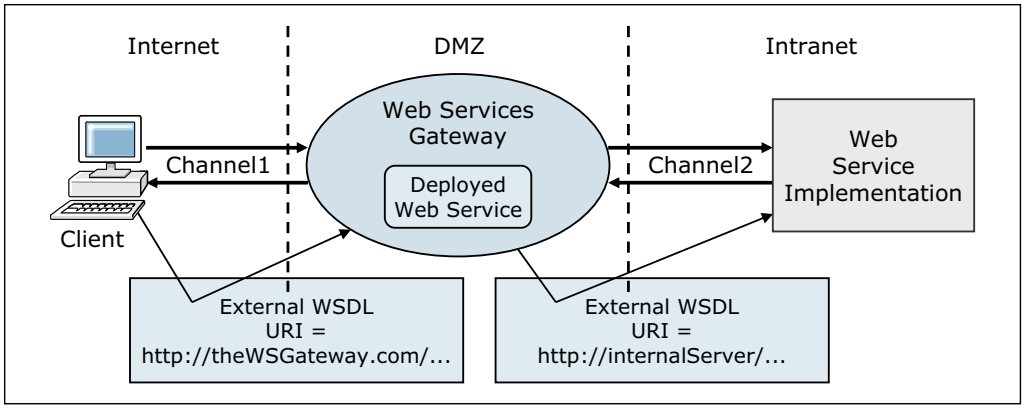


Figure 2-34 Simplified web services gateway architecture

The gateway acts as a proxy so that gateway service users do not need to know whether the underlying service is being provided internally or externally. The gateway provides a single point of control, access, and validation of web service requests. It can be used to control which web services are available to groups of web service users.

2.6 Clusters and high availability

A *cluster* is a collection of servers that are managed together. With clusters, enterprise applications can scale beyond the amount of throughput that can be achieved with a single application server. Also, enterprise applications are made highly available because requests are automatically routed to the running servers in the event of a failure. The servers that are members of a cluster can be on different host systems. A cell can include no clusters, one cluster, or multiple clusters.

WebSphere Application Server provides clustering support for the following types of servers:

- Application server clusters
- Proxy server clusters
- Generic server clusters
- Dynamic clusters

An *application server cluster* is a logical collection of application server processes that provides workload balancing and high availability. It is a grouping of application servers that run an identical set of applications managed so that they behave as a single application server (parallel processing). WebSphere Application Server Network Deployment or WebSphere Application Server for z/OS is required for clustering.

Application servers that are a part of a cluster are called *cluster members*. When you install, update, or delete an application, the updates (changes) are distributed automatically to all cluster members. By using the rollout update option, you can update and restart the application servers on each node. This process can be done one node at a time, providing continuous availability of the application to the user.

Application server clusters have the following important characteristics:

- A cluster member can belong to only a single cluster.
- Clusters can span server systems and nodes, but they cannot span cells.
- A cluster cannot span from distributed platforms to z/OS.
- A node group can be used to define groups of nodes that have enough in common to host members of a cluster. All cluster members in a cluster must be in the same node group.

2.6.1 Vertical cluster

When cluster members are on the same system, the topology is known as *vertical scaling* or *vertical clustering*. Figure 2-35 illustrates a simple example of a vertical cluster.

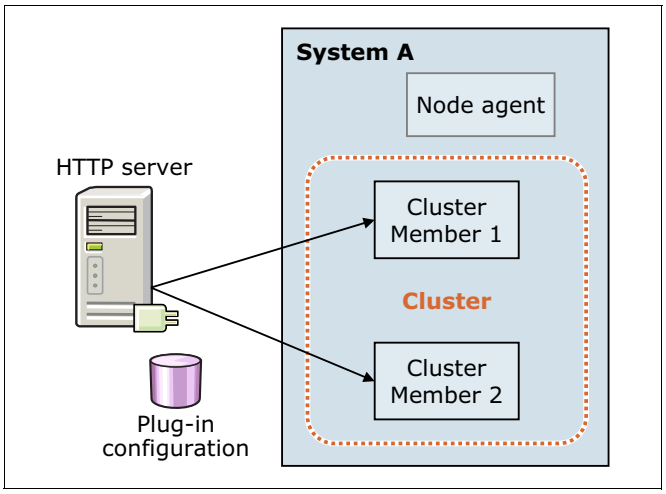


Figure 2-35 Vertical cluster

Vertical clusters offer failover support within one operating system image, provide processor level failover, and increase resource utilization. For more information, see 8.3.4, “Vertical scaling topology” on page 206.

2.6.2 Horizontal cluster

Horizontal scaling or *horizontal clustering* refers to cluster members that are spread across different server systems and operating system types (Figure 2-36). In this topology, each system has a node in the cell that is holding a cluster member. The combination of vertical and horizontal scaling is also possible.

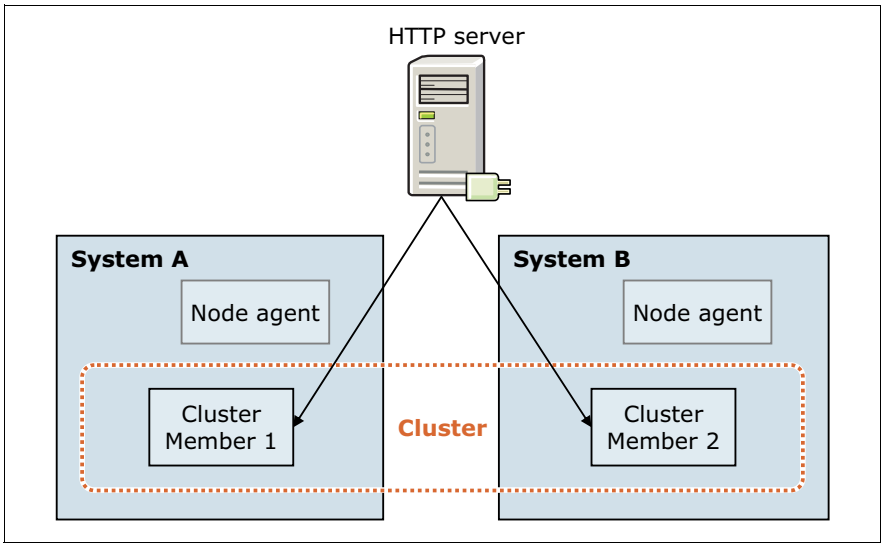


Figure 2-36 Horizontal cluster

Horizontal clusters increase availability by removing the bottleneck of using only one physical system and increasing the scalability of the environment. Horizontal clusters also support system failover. For more information, see 8.3.5, “Horizontal scaling topology” on page 209.

2.6.3 Mixed cluster

Figure 2-37 illustrates a cluster that has four cluster members and combines vertical and horizontal clustering. The cluster uses multiple members inside one operating system image (on one system) and that are spread over multiple physical systems. This configuration provides a mix of failover and performance.

Cluster members cannot span cells.

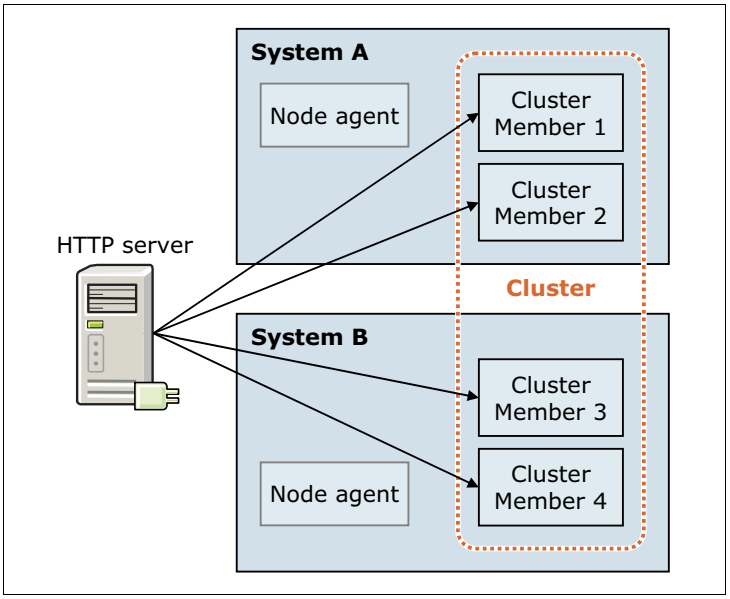


Figure 2-37 Vertical and horizontal clustering

2.6.4 Mixed-node versions in a cluster

A WebSphere Application Server Network Deployment V8.5 cluster can contain nodes and application servers from WebSphere Application Server V7 and V8. The topology illustrated in Figure 2-38 contains mixed version nodes within a cluster. You can upgrade any node in the cell and leave the other nodes at a previous release level. Consider using this feature only for migration scenarios.

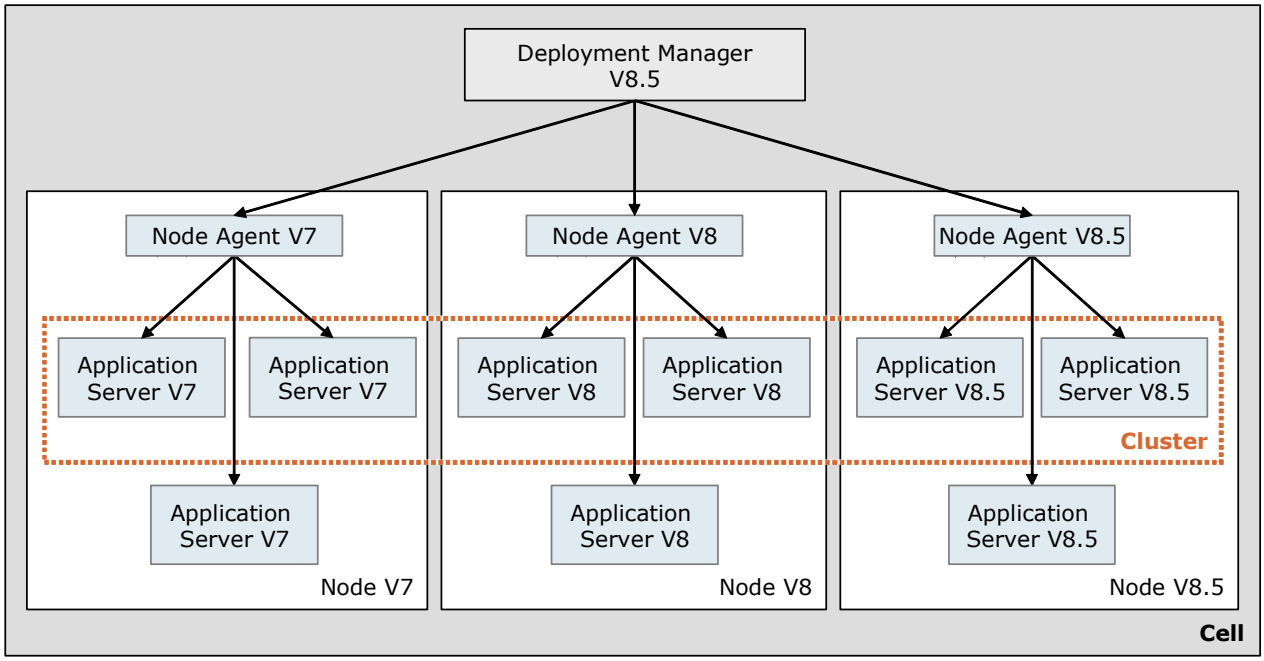


Figure 2-38 Mixed version nodes clustered in a cell

2.6.5 Dynamic cluster

Dynamic clusters are application deployment targets that operate at the application layer virtualization. Dynamic clusters provide capabilities to better manage dynamic workload by using the on-demand router server.

Keep in mind the following key points about dynamic clustering:

- Dynamic clusters grow and shrink depending on the workload demand.
- Dynamic clusters work closely with the on-demand router to ensure even distribution of workload among the cluster members.

2.6.6 Cluster workload management

This section highlights cluster workload management on distributed systems. It also addresses considerations for the z/OS platform.

Cluster workload management on distributed systems

Workload management, implemented by the use of application server clusters, optimizes the distribution of client processing requests. WebSphere Application Server can handle the workload management of servlet and EJB requests. HTTP requests can be workload-managed by using tools similar to a load balancer.

Using an HTTP traffic-handling device, such as IBM HTTP Server and the web server plug-in, is a simple and efficient way to front end the WebSphere HTTP transport.

WebSphere Application Server implements a server-weighted round-robin routing policy to ensure a balanced routing distribution. The policy is based on the set of server weights that is assigned to members of a cluster. In horizontal clustering, where each node is on a separate server system, the loss of one server system does not disrupt the flow of requests. Instead, requests are routed to cluster members on other nodes. In a horizontal cluster, the loss of the deployment manager has no impact on operations and primarily affects configuration activities. You can still use administration scripts to manage the WebSphere Application Server environment.

Cluster workload management consideration on z/OS

Workload management for EJB containers that run on z/OS can be performed by configuring the web container and EJB containers on separate application servers. Multiple application servers with the EJB containers can be clustered, enabling the distribution of enterprise bean requests between EJB containers on different application servers.

Instead of using a static round-robin procedure, workload management on the z/OS platform introduces a finer granularity and the use of real-time performance data. You can use these features to determine which member to process a transaction on.

Remember: Workload management is achieved by using the WLM subsystem in combination with the Sysplex Distributor (SD) component of z/OS. The Sysplex Distributor receives incoming requests through a Dynamic Virtual IP address and prompts WLM to indicate to which cluster member the request should be transmitted. WLM tracks how well each cluster member is achieving its performance goals in terms of response time. Therefore, it chooses the one that has the best response time to process the work.

You can classify incoming requests according to their importance. For example, requests that come from a platinum-ranked customer can be processed with higher importance (and therefore faster), than a silver-ranked customer.

When resource constraints exist, the WLM component can ensure that the member that processes a higher prioritized request gets additional resources. This system protects the response time of your most important work.

WLM changes: The WLM component can change the amount of processor, I/O, and memory resources that are assigned to the different operating system processes (the address spaces). To decide whether a process is eligible for receiving additional resources, the system checks whether the process meets its defined performance targets, and whether more important work is in the system. This technique is run dynamically so that there is no need for manual interaction after the definitions are made by the system administrator (the system programmer).

For more information about workload management on the z/OS platform in combination with WebSphere Application Server for z/OS, see 16.1.7, “Workload management for WebSphere Application Server for z/OS” on page 509.

2.6.7 High availability

WebSphere Application Server provides a high availability manager service to eliminate single points of failure in any application server. The high availability manager service provides failover when servers are not available, thus improving application availability. WebSphere Application Server also supports HTTP session memory-to-memory replication, and session database persistence that can replicate session data between cluster members.

XCF services on z/OS: On the z/OS platform, WebSphere Application Server V8.5 high availability manager uses native z/OS cluster technology, the cross-system coupling facility (XCF) services. This technology reduces the amount of processing used for the keep-alive check of clusters and improves the time it takes to detect a failed member. With XCF services, applications that are on multiple z/OS images can communicate with each other and monitor their status. For WebSphere Application Server for z/OS, the applications are the various cluster members.

For more information, see 16.1.9, “XCF support for WebSphere high availability manager” on page 514.

2.6.8 Core groups

In a high availability environment, a group of clusters can be defined as a *core group*. A core group is a high availability domain that consists of a set of processes in the same cell that can directly establish high availability relationships. Highly available components can fail over only to another process in the same core group. Replication can occur only between members of the same core group.

All of the application servers defined as members of a cluster included in a core group are automatically members of that core group. Every deployment manager, node agent, application server, and proxy server is a member of a core group. When a process is created, it is automatically added to a core group. A cell, by default, contains a single core group, called DefaultCoreGroup. All processes in the cell are initially members of this core group. A single core group is usually sufficient. However, you might want to define *multiple core groups* if there are many processes in the cell and the core group protocols consume correspondingly large amounts of resources. You might also define them if the core group protocols need tuning or configuring to use values that work best with smaller numbers of core group members.

A cell must contain at least one core group, although multiple core groups are supported. Each core group contains a core group coordinator to manage its high availability relationships. Each group also contains a set of high availability policies used to manage the highly available components within that core group. You can also have servers within the core group that are not part of any cluster. If members of different core groups need to share workload management or on-demand configuration routing information, use the core group bridge service to connect these core groups. The core groups can communicate within the same cell or across cells.

2.7 Run times

This section briefly explains how WebSphere Application Server processes run at run time. Executable processes include application servers, node agents, administrative agents,

deployment managers, and job managers. Because cells, nodes, and clusters are administrative concepts, they are not executable components.

2.7.1 Distributed platforms

On distributed platforms, WebSphere Application Server is built by using a single process model where the entire server runs in a single JVM process. Each process is displayed as a Java process. For example, when you start a deployment manager on Windows, a `java.exe` process is visible in the Windows Task Manager. Starting a node agent starts a second `java.exe` process, and each application server started is also seen as a `java.exe` process.

2.7.2 z/OS

WebSphere Application Server for z/OS uses multiple runtime components to form the different Websphere Application Server parts. These runtime components represent one or more logical application servers:

- Control region

  This address space handles in its JVM the incoming connections from the clients, and dispatches the request to the z/OS WLM queues.

- Control region adjunct

  This server address space is started by z/OS WLM when messaging functions are used. The JMS messaging engine runs completely inside the JVM of the control region adjunct.

- Servant region

  Applications run in the JVM of this address space. As configured, z/OS WLM can dynamically start multiple servant regions, even in a stand-alone application server environment. There must be at least one servant region for each control region.

- Location service daemon

  This unique cell component provides the location name service for external clients. It also provides access to modules in storage for all servers within the cell on the same sysplex. The daemon is started automatically when the first control region is started.

Figure 2-39 represented the runtime components for WebSphere Application Server for z/OS, in relationship with the other major subsystems and z/OS functional infrastructure.

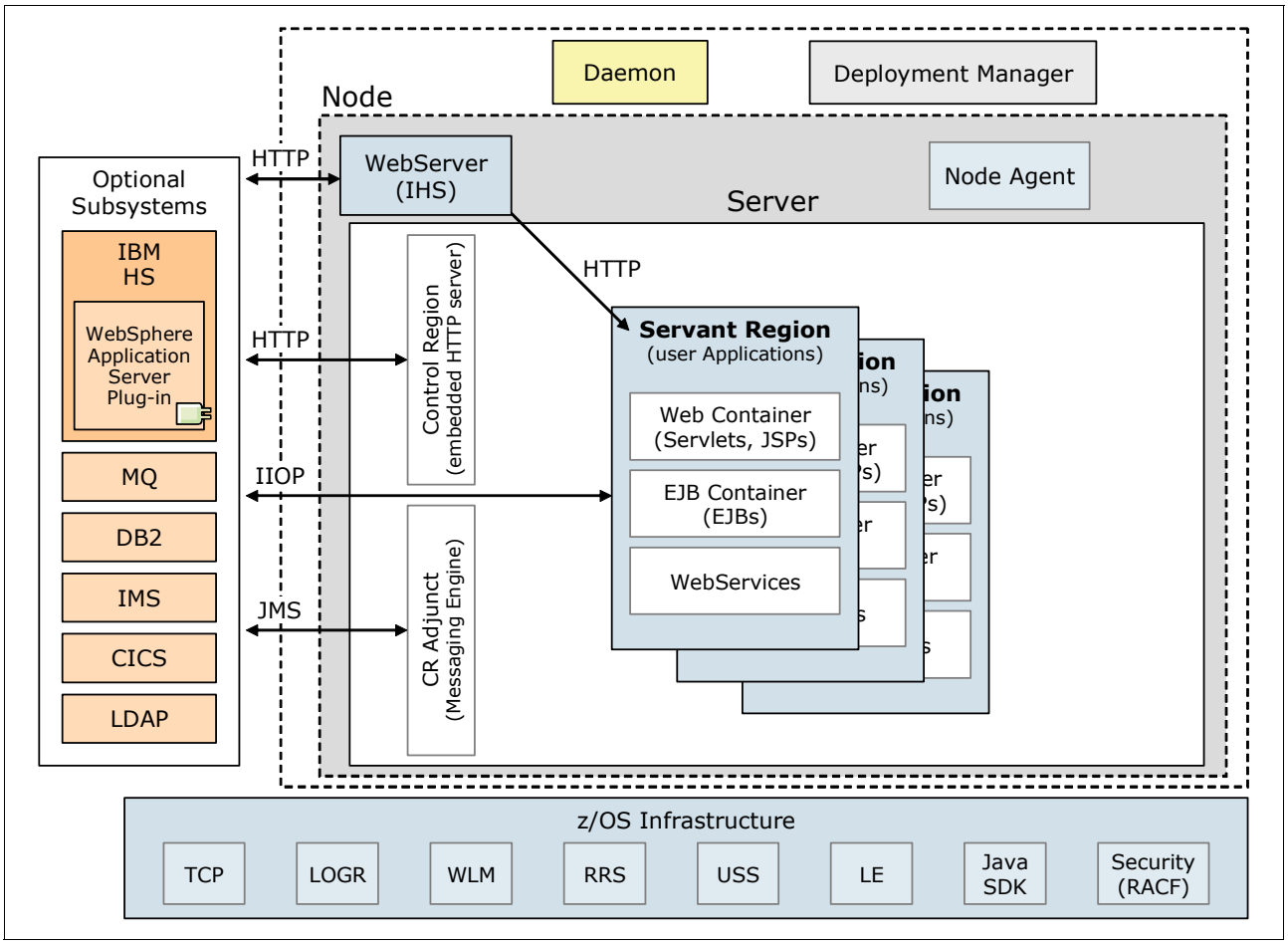


Figure 2-39 Architecture of WebSphere Application Server V8.5 for z/OS at run time

The WebSphere Application Server V8.5 Liberty profile provides the following additional z/OS runtime processes:

- Angel process (`bbgzangl`)
- Server process (`bbgzsrv`)

Cluster capabilities: Each logical application server on z/OS has cluster capabilities through the use of multiple servants. This capability enhances the performance and availability of an application because a failure in one of these servants does not harm the others. Each servant runs its own JVM and its own copy of the application.

For information about the WebSphere Application Server for z/OS run time, see 16.1.6, “Runtime processes” on page 507.

3

Integration with other products

WebSphere Application Server works closely with other IBM products to provide a fully integrated solution. This chapter introduces products that provide enhanced security and messaging options, and that provide broad integration features. This chapter includes the following sections:

- IBM Tivoli Access Manager for e-business
- IBM Tivoli Directory Server
- IBM WebSphere MQ
- IBM WebSphere Adapters
- IBM WebSphere DataPower Appliances
- IBM DB2
- IBM Tivoli Composite Application Manager for WebSphere
- IBM WebSphere Portal Server
- IBM Tivoli Workload Scheduler
- IBM WebSphere eXtreme Scale

3.1 IBM Tivoli Access Manager for e-business

IBM Tivoli Access Manager for e-business provides a holistic security solution at the enterprise level. This section provides information about the integration of Tivoli Access Manager for e-business with WebSphere Application Server.

3.1.1 Features of Tivoli Access Manager for e-business

Tivoli Access Manager provides the following features:

- Defines and manages centralized authentication, access, and audit policies for a broad range of business initiatives.
- Establishes a new audit and reporting service that collects audit data from multiple enforcement points, and from other platforms and security applications.
- Enables flexible single sign-on (SSO) to web-based applications that span multiple sites or domains with a range of SSO options. These options can eliminate help-desk calls and other security problems associated with multiple passwords
- Uses a common security policy model with the Tivoli Access Manager family of products to extend support to other resources.
- Manages and secures business environments from existing hardware (mainframe, PCs, servers) and operating system platforms, including Windows, Linux, AIX, Solaris, and HP-UX.
- Provides a modular authorization architecture that separates security code from application code.
- Automatically authenticates Microsoft Windows users with their Windows credentials in WebSphere if they are connected to a Microsoft Active Directory.
- Allows integration with Tivoli Identity Manager, which supports administering large numbers of user accounts in enterprise environments.

In summary, Tivoli Access Manager provides centralized authentication and authorization services to different products. Applications delegate authentication and authorization decisions to Tivoli Access Manager.

For more information about Tivoli Access Manager for e-business, see:

http://www.ibm.com/software/tivoli/products/access-mgr-e-bus/

3.1.2 Integration with WebSphere Application Server

WebSphere Application Server provides its own security infrastructure. This infrastructure consists of mechanisms that are specific to WebSphere Application Server, many that use open security technology standards. This security technology is widely proven, and the software can integrate with other enterprise technologies. For more information about WebSphere Application Server's security infrastructure, see Chapter 15, "Security" on page 469.

The WebSphere Application Server security infrastructure is adequate for many situations and circumstances. However, integrating WebSphere Application Server with Tivoli Access Manager allows for end-to-end integration of application security for the entire enterprise.

Using this approach at the enterprise level provides the following advantages:

- Reduced risk through a consistent services-based security architecture
- Lower administration costs through centralized administration and fewer security subsystems
- Reduced application development costs because developers do not have to develop customized security subsystems
- Built in, centralized, and configured handling of business concerns, such as privacy requirements

WebSEAL

The WebSEAL server is a resource manager in the Tivoli Access Manager architecture that you can use to manage and protect web content resources. WebSEAL works as a reverse HTTP/HTTPS proxy server in front of the web servers or application servers. It connects to the policy server for the access control list (ACL) as shown in Figure 3-1. Because it handles the HTTP/HTTPS protocol, WebSEAL is independent of the web server or application server implementation. With this feature, you can authenticate and authorize clients in a distributed, multivendor integrated environment.

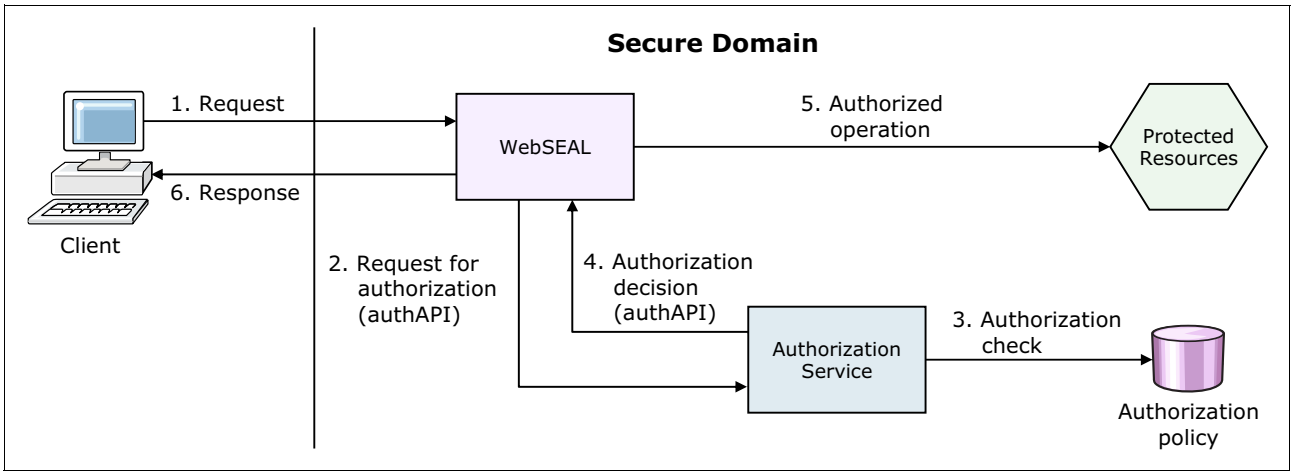


Figure 3-1 WebSEAL as a proxy in WebSphere integration

Repositories

In addition to WebSphere Application Server security, Tivoli Access Manager requires a user repository. It supports different repositories, such as IBM Tivoli Directory Server and Microsoft Active Directory. You can configure Tivoli Access Manager to use the same user repository as WebSphere Application Server, so that you can share user identities with both Tivoli Access Manager and WebSphere Application Server.

Tivoli Access Manager policy server

The Tivoli Access Manager policy server maintains the master authorization policy database. This database contains the security policy information for all resources and the credentials information of all participants in the secure domain, both users and servers. The authorization database is replicated across all local authorization servers.

Tivoli Access Manager for WebSphere component

The Tivoli Access Manager *server* is bundled with WebSphere Application Server Network Deployment.

The Tivoli Access Manager *client* is embedded in WebSphere Application Server. You can configure the client by using the scripting and GUI management facilities of WebSphere Application Server. For more information about configuring the embedded Tivoli Access Manager client, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=tsec_enable_TAM

All communication between the Tivoli Access Manager clients and the Tivoli Access Manager server is run through the Java Authorization Contract for Containers (JACC) application programming interface (API).

Tivoli Access Manager further integrates with WebSphere Application Server by supporting the special subjects `AllAuthenticated` and `Everyone`. `AllAuthenticated` and `Everyone` are subjects that are specific to WebSphere Application Server. The `AllAuthenticated` subject allows access to a resource for users who are authenticated, regardless of the repository user groups to which those users might belong. The `Everyone` subject allows access to a resource for all users regardless of whether they are authenticated.

Figure 3-2 shows the integration interfaces between WebSphere Application Server and Tivoli Access Manager.

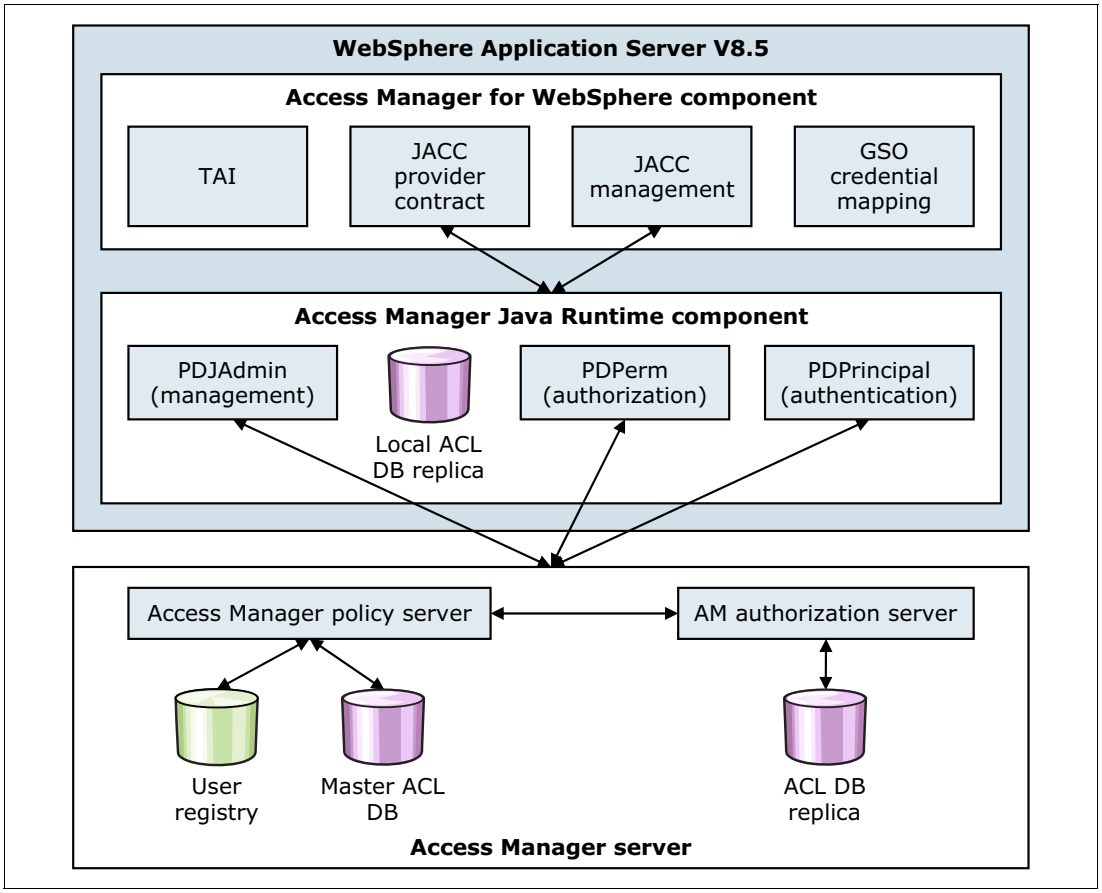


Figure 3-2 Integration of WebSphere Application Server with Tivoli Access Manager

Further advantages of using Tivoli Access Manager

In addition to the enterprise-level advantages, using Tivoli Access Manager at the application server level has the following advantages:

- Supports accounts and password policies
- Supports dynamic changes to the authorization table without having to restart applications

Security, networking, and topology considerations

A Lightweight Directory Access Protocol (LDAP) server contains sensitive data in terms of authentication, authorization, and privacy. The Tivoli Access Manager server manages this data. Therefore, the servers belong to the data layer of the network. Consider enabling Secure Sockets Layer (SSL) configuration options between the databases so that data is encrypted.

Legal considerations: Storage on IT systems of certain data types, such as personally identifiable data in the European Union, can be subject to legal or regulatory issues. Consult your legal department before deploying such information about your systems. These considerations vary by geographical area and industry.

3.2 IBM Tivoli Directory Server

In today's highly connected world, directory servers are the foundation of authentication systems for internal and external user populations in the corporate infrastructure. Tivoli Directory Server provides a high-performance LDAP identity infrastructure that can handle millions of entries. It is built to serve as the identity data foundation for web applications and identity management initiatives.

This section provides information about the integration of Tivoli Directory Server with WebSphere Application Server.

3.2.1 Features of Tivoli Directory Server

A *directory* is a data structure that enables the lookup of names and associated attributes arranged in a hierarchical tree structure. In the context of enterprise application servers, this structure enables applications to perform these functions:

- Look up a user principal
- Determine the attributes that the user has
- Determine the groups of which the user is a member

You can then make decisions regarding authentication and authorization by using this information.

Remember: *LDAP* is the name of the protocol that is used between a directory server and a client, which in this case is WebSphere Application Server. Often the directory server is called the *LDAP server*. LDAP is still a protocol, although the authentication happens by using the Lightweight Third Party Authentication (LTPA) mechanism. The LDAP carries data between WebSphere Application Server and the directory server as part of the authentication mechanism.

LDAP is a fast and simple way to query and maintain user entities in a hierarchical data structure. It has advantages over using databases as a user repository in terms of speed,

simplicity, and standardized models or schemas for defining data. Standard schemas have standard hierarchies of objects, such as objects that represent a person in an organization. These objects, in turn, have attributes such as a user ID and common name. The schema can have custom objects added to it, meaning that your directory is extensible and customizable.

Generally, LDAP is chosen over a custom database repository of users for these reasons. LDAP implementations (such as Tivoli Directory Server) use database engines in the background. However, these engines are optimized for passive lookup performance (through indexing techniques). LDAP implementation optimizations are based on the assumption that data changes relatively infrequently, and the directory is primarily for looking up data rather than updating data.

For more information about Tivoli Directory Server, see:

http://www.ibm.com/software/tivoli/products/directory-server/

3.2.2 Integration with WebSphere Application Server

You can enable security in WebSphere Application Server to manage users and to assign specific roles to them. When using the Tivoli Directory Server, you select either a stand-alone LDAP registry or a federated registry. With a stand-alone registry, WebSphere Application Server can connect to one directory server at a time. Thus, you can have only one LDAP server in your environment, or you can set up a failover cluster of the LDAP servers. The failover is managed by WebSphere Application Server.

If you use a federated repository, choose from the following repository solutions based on LDAP:

- Single LDAP (full LDAP tree)
- Subtree of an LDAP (used only when a group in LDAP needs access to WebSphere Application Server)
- Multiple LDAPs (uses a unique user ID through all the LDAP trees)

3.2.3 Security, networking, and topology considerations

Because the LDAP server contains sensitive data in terms of authentication, authorization, and privacy, the LDAP server belongs to the data layer of the network. Consider enabling SSL options in the WebSphere Application Server security configuration. Enabling these options ensures that the data is encrypted during transport between the application server layer and the data layer.

Remember: When LDAP is used with the SSL protocol, it is often called *LDAPS*.

Legal considerations: Storage on IT systems of certain data types, such as personally identifiable data in the European Union, can be subject to legal or regulatory issues. Consult your legal department before deploying such information about your systems. These considerations vary by geographical region and industry.

For a list of supported directory servers for WebSphere Application Server, see System Requirements for WebSphere Application Server Base and Network Deployment V8.5 at:

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

When you reach this web page, select the operating system and LDAP repository type (federated or stand-alone) that you want to use.

3.3 IBM WebSphere MQ

WebSphere MQ is an IBM middleware product that provides asynchronous messaging technology for application-to-application communication rather than application-to-user and user interface communication. This section provides information about the integration of WebSphere MQ with WebSphere Application Server.

3.3.1 Features of IBM WebSphere MQ

WebSphere MQ is available on many platforms and operating systems. It offers a fast, robust, and scalable messaging solution. WebSphere MQ assures one-time-only delivery of messages to queue destinations that are hosted by queue managers. This messaging solution has APIs in C, Java, COBOL, and other languages that allow applications to construct, send, and receive messages.

With the advent of Java Message Service (JMS), generic, portable client applications can be written to interface with proprietary messaging systems such as WebSphere MQ. The integration of WebSphere Application Server with WebSphere MQ over time is influenced by this dichotomy of generic JMS and proprietary WebSphere MQ access approaches.

For more information about WebSphere MQ, see:

http://www.ibm.com/software/integration/wmq/

3.3.2 Integration with WebSphere Application Server

WebSphere Application Server *messaging* is a general term for a group of components that provide the messaging function for applications. WebSphere MQ and WebSphere Application Server messaging are complementary technologies that are tightly integrated to provide for various messaging topologies.

WebSphere Application Server supports asynchronous messaging based on the JMS programming interface and the use of a JMS provider and its related messaging system. JMS providers must conform to the JMS specification version 1.1.

In WebSphere Application Server V8.5, you can use the following JMS providers:

- The default messaging provider
- WebSphere MQ
- Third-party JMS providers

The default messaging provider is the JMS API implementation for messaging (such as connection factories and JMS destinations). The concrete destinations (queues and topic spaces) behind the default messaging provider interface are implemented in a service integration bus. A *service integration bus* (bus) consists of one or more bus members, which can be application servers or clusters. Each bus member has one messaging engine (more, in the case of clusters) that manages connections to the bus and messages.

A bus can connect to other buses and to WebSphere MQ. Similarly, the WebSphere MQ JMS provider is the JMS API implementation with WebSphere MQ (with queue managers, for

example) implementing the real destinations for the JMS interface. WebSphere MQ can coexist on the same host as a WebSphere Application Server messaging engine.

Whether to use the default messaging provider, the direct WebSphere MQ messaging provider, or a combination depends on several factors. No set of questions can lead you directly to the decision. However, consider the following guidelines:

- In general, the default messaging provider is a good choice when you require messaging between WebSphere Application Server and an existing WebSphere MQ backbone and its applications.
- The WebSphere MQ messaging provider is a good choice in the following circumstances:
  - You are currently using a WebSphere MQ messaging provider and want to continue using it.
  - You require access to heterogeneous, non-JMS enterprise information systems (EIS).
  - You require access to WebSphere MQ clustering.

Using a topology that combines WebSphere MQ and the default messaging provider is beneficial. This combination provides tight integration between WebSphere and the default messaging provider (clustering), and the additional flexibility of WebSphere MQ.

For more information about messaging with WebSphere Application Server and new features for WebSphere MQ connectivity, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=cmj_jmsp_wmq

3.3.3 Connecting WebSphere Application Server to WebSphere MQ

If both WebSphere Application Server and WebSphere MQ exist in your environment, you can use the following options:

- Use the default messaging provider
- Use the WebSphere MQ provider
- Use a mixture of the default and the WebSphere MQ messaging provider

Both providers can transfer messages between application servers by using the WebSphere MQ infrastructure.

For a more information, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tmj_jmsp_mixed

If you decide to use a topology that includes both WebSphere MQ and the default messaging provider, the following methods can allow interaction between them:

- Extend the WebSphere MQ and bus networks by defining a WebSphere MQ link on a messaging engine in a WebSphere Application Server that connects the bus to a WebSphere MQ queue manager.

  WebSphere MQ perceives the connected bus as a queue manager. The bus perceives the WebSphere MQ network as another bus.

  WebSphere MQ applications can send messages to queue destinations on the bus. Default messaging applications can send messages to WebSphere MQ queues without being aware of the mixed topology. Similar to WebSphere MQ queue manager networks, this method can be used to send messages from one messaging network to the other.

Considerations: Keep in mind the following considerations:

- WebSphere MQ to bus connections are supported only over TCP/IP.
- A bus cannot be a member of a WebSphere MQ cluster.

Figure 3-3 shows a sample integration for WebSphere Application Server and WebSphere MQ.

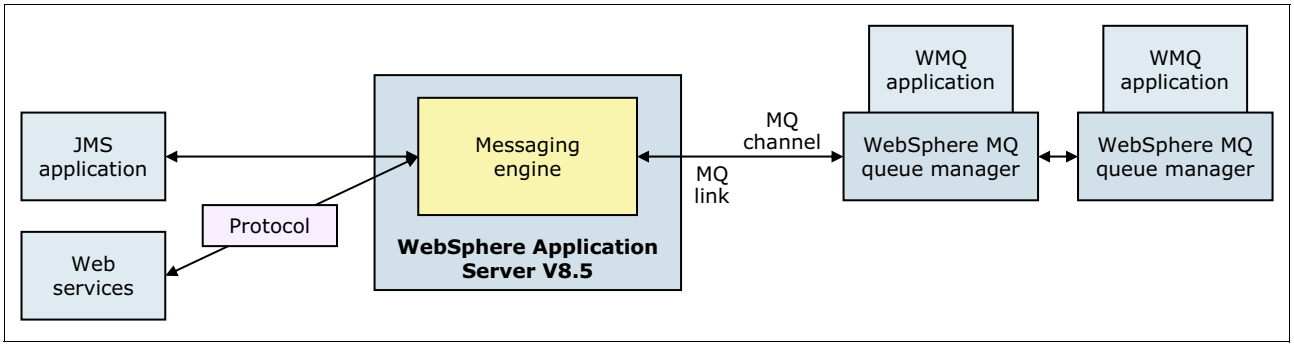


Figure 3-3 WebSphere Application Server integration with WebSphere MQ

- Integrate specific WebSphere MQ resources into a bus for direct, synchronous access from default messaging applications that are running in WebSphere Application Servers. Represent a queue manager or *queue sharing group* as a WebSphere MQ server in the WebSphere Application Server cell. Then add it to a bus as a bus member.

  Tip: A WebSphere MQ shared queue group is a collection of queues that can be accessed by one or more queue managers. Each queue manager that is a member of the shared queue group has access to any of the shared queues.

  WebSphere MQ queues on queue managers, and queue sharing groups that run on z/OS, can be accessed in this way from any WebSphere Application Server that is a member of the bus. Only the following configurations can be accessed from a bus in this way:

  – WebSphere MQ (distributed platforms) Version 7 or later queue managers
  – Queue sharing groups that run on z/OS Version 6 or later

  The WebSphere MQ server does not depend on any one designated messaging engine. Therefore, this type of connectivity to WebSphere MQ can tolerate the failure of any message engine if another is available in the bus. This configuration increases robustness and availability. This method can be used for both sending and consuming messages from WebSphere MQ queues.

  When a default messaging application sends a message to a WebSphere MQ queue, the message is immediately added to that queue. If the WebSphere MQ queue manager is not available, the message is not stored by the bus for later transmission to WebSphere MQ. When a WebSphere Application Server application receives a message from a WebSphere MQ queue, it receives the message directly from the queue.

For more information about the messaging features of the WebSphere Application Server V8.5, see Chapter 13, “Messaging and service integration” on page 415.

3.4 IBM WebSphere Adapters

A *resource adapter* is a system-level software driver that a Java application uses to connect to an EIS. A resource adapter plugs into an application client, and provides connectivity between the EIS and the enterprise application.

Exception: This topic is not relevant to the WebSphere MQ resource adapter. For more information about this adapter, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=tmj_wmqra_maint

This section provides information about the integration of WebSphere Adapters with WebSphere Application Server.

3.4.1 Features of IBM WebSphere Adapters

IBM WebSphere Adapters provide a set of generic technology and business application adapters with wizards that quickly and easily enable connections to enterprise information systems (EIS). These systems include existing enterprise applications, enterprise resource planning (ERP) systems, human resource (HR) systems, customer relationship management (CRM) systems, and supply chain systems. WebSphere Adapters can also be used to integrate those systems to the following products:

- IBM business process management (BPM) products
- IBM enterprise service bus (ESB) implementations
- Application server solutions in a service-oriented architecture (SOA)

WebSphere Adapters implement the Java EE Connector Architecture (JCA) and Enterprise Metadata Discovery specifications. This configuration provides a simple and quick integration experience with graphical discovery tools without needing to write code. WebSphere Application Server supports JCA versions 1.0, 1.5, and 1.6, including additional configuration features for JCA 1.5 and JCA 1.6.

WebSphere Adapters include the following types of adapters:

- Technology adapters

  The following adapters deliver file and database connectivity solutions:

  - Enterprise Content Management (ECM)
  - Email
  - File Transfer Protocol (FTP)
  - Flat Files
  - IBM i
  - Java Database Connectivity (JDBC)
  - Lotus® Domino®

- IBM WebSphere Adapters for System z provide connectivity options for mainframe transactions:

  - CICS Transaction Gateway
  - IMS SOA Integration Suite
  - IMS Connect
  - IMS TM
  - IBM InfoSphere® Classic Federation Server for z/OS
  - CICS Transaction Server for z/OS V4.1

- The following adapters integrate enterprise business application suites:
  - JD Edwards EnterpriseOne
  - Oracle E-Business Suite
  - PeopleSoft Enterprise
  - SAP Software
  - Siebel Business Applications

IBM provides the WebSphere Adapter Toolkit at no additional cost so that customers and business partners can develop custom JCA adapters to meet their unique business needs. WebSphere Adapter Toolkit integrates with IBM Rational Application Developer environment, which provides an integrated development and WebSphere test environment. For more information about the WebSphere Adapter Toolkit, see:

http://www.ibm.com/software/integration/wbiadapters/toolkit/

For more information about IBM WebSphere Adapters, see:

http://www.ibm.com/software/integration/wbiadapters/

3.4.2 Integration with WebSphere Application Server

WebSphere Adapters plug into WebSphere Application Server and provide bidirectional connectivity between enterprise applications (or Java EE components), WebSphere Application Server, and EIS.

Figure 3-4 shows the relationship between WebSphere Application Server and a WebSphere Adapter.

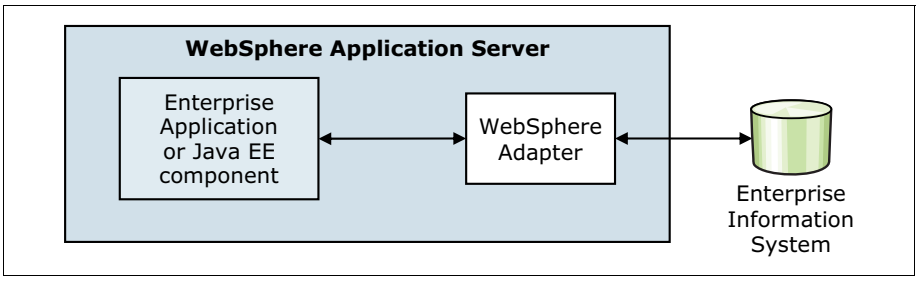


Figure 3-4 WebSphere Adapter integration with WebSphere Application Server

3.5 IBM WebSphere DataPower Appliances

IBM WebSphere DataPower® Appliances simplify, govern, and optimize the delivery of services and applications, and enhance the security of XML and IT services. They extend the capabilities of an infrastructure by providing a multitude of functions. The capabilities of the WebSphere DataPower Appliances span service-oriented architecture (SOA) connectivity, business-to-business (B2B) connectivity, advanced application caching, rapid integration with cloud-based systems, and more. DataPower Appliances are rack-mountable hardware devices or blade servers that mount in an IBM BladeCenter® chassis.

This section provides information about the integration of WebSphere DataPower Appliances with WebSphere Application Server.

3.5.1 DataPower appliance models

The WebSphere DataPower Appliance family contains the following models. Each appliance has its own characteristics and fits different business needs.

- WebSphere DataPower Service Gateway XG45 Appliance

  The WebSphere DataPower Service Gateway XG45 is a network appliance that is built for web services deployments, governance, light integrations, and hardened security. The XG45 provides protection against XML vulnerabilities by acting as an XML proxy. It runs XML well-formed checks, buffer overrun checks, XML schema validation, XML filtering, and XDoS protection. XG45 also includes many essential security functions beyond those of an XML firewall. These functions include web services access control authentication, authorization, and auditing (AAA), XML Encryption and Digital Signature, WS-Security, and content-based routing.

  For more information about the DataPower XG45, see:

  http://www.ibm.com/software/integration/datapower/XG45/

- WebSphere DataPower Integration Appliance XI52

  IBM WebSphere DataPower Integration Appliance XI52 is a hardware ESB, delivering common message transformation, integration, and routing functions in a network device. These functions cut operational costs and improve performance. By making on-demand data integration part of the shared SOA infrastructure, the XI52 is one of the few nondisruptive technologies for application integration.

  For more information about the DataPower XI52, see:

  http://www.ibm.com/software/integration/datapower/xi52/

- WebSphere DataPower Integration Appliance XI50B and XI50z

  IBM WebSphere DataPower Integration Blade XI50B, and the WebSphere DataPower Integration XI50z for IBM zEnterprise® are hardware ESBs. WebSphere DataPower Integration Blade XI50B provides all the same functions as the XI52, but is available in a blade form-factor. The WebSphere DataPower Integration XI50z for zEnterprise is also similar in function to the XI52, but is designed for System z environments.

  For more information about the DataPower XI50, see:

  http://www.ibm.com/software/integration/datapower/xi50/

- WebSphere DataPower B2B Appliance XB62

  IBM WebSphere DataPower B2B Appliance XB62 delivers secure trading partner data integration tracking, routing, and security functions in a network device. This cuts operational costs and improves performance. The XB62 is a nondisruptive technology that allows you to extend your existing B2B implementations and internal integration infrastructure. This functionality delivers rapid return on investment and reduces total cost of ownership.

  For more information about the DataPower XB62, see:

  http://www.ibm.com/software/integration/datapower/b2b_xb60/

- WebSphere DataPower XC10 Appliance

  IBM WebSphere DataPower XC10 V2 is a purpose-built, easy-to-use appliance designed for simplified deployment and hardened security. This deployment is run in the caching tier of your enterprise application infrastructure. XC10 V2 incorporates a large 240 GB cache into the DataPower line of appliances from IBM. It adds elastic caching functions that enable business critical applications to scale cost effectively with consistent performance. WebSphere DataPower XC10 V2 is designed for drop-in use in various application environments. These environments include WebSphere Application Server V6.1, V7.0,

V8.0, and V8.5, and other WebSphere family products. With these environments, it can deliver a cost-effective, distributed caching solution in support of data-oriented distributed caching scenarios.

For more information about the DataPower XC10, see:

http://www.ibm.com/software/webservers/appserv/xc10/

Figure 3-5 illustrates the flow of using the DataPower Integration Appliance in the various tiers.

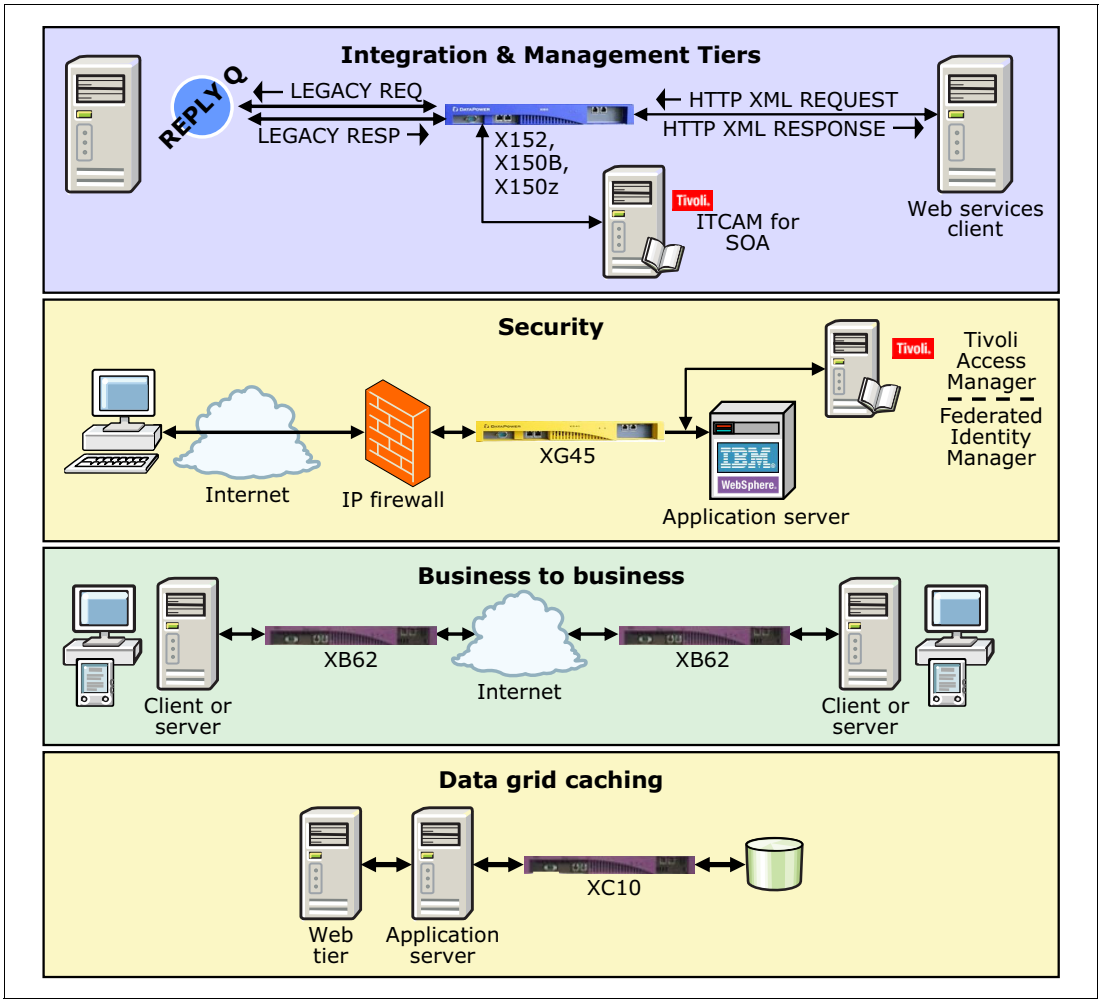


Figure 3-5 DataPower appliances

3.5.2 Integration with WebSphere Application Server

In WebSphere Application Server V8.5, with the consolidated administration feature for WebSphere DataPower, you can manage and integrate appliances into your environment. The administrative console contains an administration interface called the DataPower appliance manager. It is used to manage multiple WebSphere DataPower boxes, as shown in Figure 3-6.

Figure 3-6 DataPower appliance manager interface of the administrative console

The administrative console is the single point of administration to manage WebSphere Application Server, WebSphere DataPower, and the solutions that combine them.

From the DataPower appliance manager interface, you can perform the following tasks:

- Add, change, or remove a DataPower appliance, and monitor its operation and synchronization status.
- Add firmware versions, view existing firmware versions, or delete a firmware version.
- Add, view, or delete a managed set. A managed set is a group of appliances whose firmware, shareable appliance settings, and managed domains are all kept synchronized.
- View the status of a task. A DataPower task is a long-running request that you ask the DataPower appliance manager to process.

3.6 IBM DB2

DB2 is an open-standards, multiplatform, relational database system that is powerful enough to meet the demands of large corporations and flexible enough to serve medium-sized and small businesses.

This section provides information about the integration of DB2 with WebSphere Application Server.

3.6.1 Features of IBM DB2

DB2 has editions that work on Linux, UNIX, Windows, IBM System i®, and System z. These editions range from the single-user Personal edition, through the no-cost Express Community edition, to the partitioned and massively parallel Enterprise Server edition.

Starting with DB2 Version 9, the database supports two different data structures. The structures are the relational data structure for structured data, and a hierarchical data structure for XML content. The hierarchical data structure is implemented by the IBM pureXML® extender.

DB2 uses the following extenders to enhance the base functionality of the database:

- Geodetic extender
- Net Search Extender
- pureXML extender
- Spatial extender
- XML extender

For more information about DB2 and its editions, see:

http://www.ibm.com/db2/

IBM DB2 pureScale® is a high performance technology database solution. It implements in-memory transactions in a distributed server environment. For more information, see the following websites:

- DB2 pureScale product page:

  http://www.ibm.com/software/data/db2/linux-unix-windows/editions-features-purescale.html

- *What is DB2 pureScale? Going to extremes on scale and availability for DB2* on IBM developerWorks:

  http://www.ibm.com/developerworks/data/library/dmmag/DBMag_2010_Issue1/DBMag_Issue109_pureScale/

3.6.2 Integration with WebSphere Application Server

DB2 delivers enhanced integration capabilities and features with WebSphere Application Server. You can speed up your application development and web deployment cycles with this powerful combination.

You can integrate DB2 with WebSphere Application Server in many scenarios:

- DB2 can be the hybrid data store for your applications. It can enhance your data processing with its powerful XML capabilities. You can configure data sources to use DB2 by using JDBC drivers.
- With its IBM pureQuery™ runtime environment, DB2 provides an alternative set of APIs that can be used instead of JDBC to access the DB2 database. This environment is a high performance Java data access platform that helps manage applications that access data. PureQuery support is based on the Java Persistence API of Java EE and Java SE environments.
- You can configure a bus (messaging) member to use DB2 as a data store.
- You can configure the session management facility of WebSphere Application Server for database session persistence by using DB2 as the data store. You can collect and store session data in a DB2 database.

- You can use DB2 as the data store for your UDDI registry data.
- The scheduler database for storing and running tasks of the scheduler service of WebSphere Application Server can be a DB2 database. The scheduler service is a WebSphere programming extension that is responsible for starting actions at specific times or intervals.

For more information about data access resources for WebSphere Application Server, see:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=welcdataaccess

3.7 IBM Tivoli Composite Application Manager for WebSphere

This section provides information about the integration of Tivoli Composite Application Manager (ITCAM) for WebSphere with WebSphere Application Server.

3.7.1 Features of ITCAM for WebSphere

Typical testing, staging, and production environments consist of several components, which can be web servers, application servers, and databases. The application servers can be organized into one or more cells. Often stand-alone servers are used for specific functions. DataPower devices, Portal, and Process Servers can extend the system. In real production sites, all these components are organized into clusters.

Administrators need to monitor the performance, availability, and reliability of the system. Further, they must note any error quickly enough so they can fix it without downtime.

ITCAM for WebSphere is an application management tool that helps maintain the availability and performance of on-demand applications. It helps pinpoint, in real time, the source of bottlenecks in application code, server resources, and external system dependencies. ITCAM for WebSphere provides in-depth application performance analysis and tracing facilities based on WebSphere. It provides detailed reports that you can use to enhance the performance of your applications.

For more information about ITCAM for WebSphere, see:

http://www.ibm.com/software/tivoli/products/composite-application-mgr-websphere/

3.7.2 Integration with WebSphere Application Server

With ITCAM for WebSphere, you can analyze the health of the WebSphere Application Server and the transactions that are started in it. It can trace the transaction execution to the detailed method-level information. It connects transactions that are created from one application server to another. It also starts services from other application servers, including mainframe applications in Information Management System (IMS) or CICS.

ITCAM for WebSphere provides a flexible level of monitoring. The flexibility ranges from a non-intrusive production ready monitor to a detailed tracing for problems with locking or memory leaks. ITCAM for WebSphere provides a separate interactive web console for monitoring data that is displayed on the Tivoli Enterprise Portal.

ITCAM for WebSphere provides the following additional functions:

- Integration with IBM Tivoli Service Manager by providing a web services interface to obtain health status
- Improved memory leak and locking analysis pages
- Problem determination enhancements
- Advanced visualization, aggregation, persistence, and correlation of performance metrics in Tivoli Enterprise Portal
- Additional WebSphere server platform support, including WebSphere Portal Server and WebSphere Process Server
- Enhanced composite transaction tracing and decomposition
- Web session browser to help diagnose session-related problems

3.7.3 Architecture of ITCAM for WebSphere

ITCAM for WebSphere is a distributed performance monitoring application for application servers. Its components are connected through TCP/IP communication. The central component of ITCAM for WebSphere is the managing server. It collects and displays various performance information from application servers.

The application servers run a component of ITCAM for WebSphere, called the *data collector*, which is a collecting agent. The data collector helps you pinpoint, in real time, the source of bottlenecks in application code, server resources, and external system dependencies. The Tivoli Enterprise Monitoring Agent component collects information that shows the status of the WebSphere server, and sends this information to the Tivoli Enterprise Monitoring Agent. This agent is installed on the individual systems where the data collector is located.

Figure 3-7 shows the overall architecture of ITCAM for WebSphere.

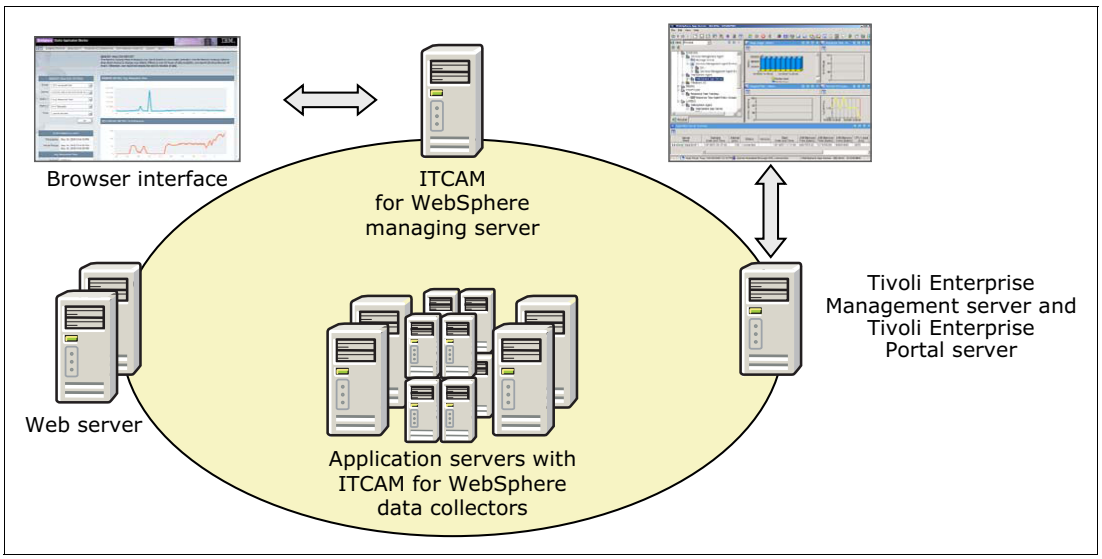


Figure 3-7 ITCAM for WebSphere architecture

For more information about ITCAM for WebSphere usage scenarios, see the following IBM Redbooks publications:

- *IBM Tivoli Composite Application Manager Family Installation, Configuration, and Basic Usage*, SG24-7151
- *Solution Deployment Guide for IBM Tivoli Composite Application Manager for WebSphere*, SG24-7293

3.8 IBM WebSphere Portal Server

WebSphere Portal is a web portal solution that is an integration framework for EIS. For integration, the portal renders the different information sources into one browser window. One or more portlets can be displayed on each page in a hierarchical page structure. Each portlet is similar to a small browser window, without the control buttons. It displays only one piece of content in HTML, but the WebSphere Portal Server does the integration job in one website structure.

This section provides information about the integration of WebSphere Portal Server with WebSphere Application Server.

3.8.1 Features of WebSphere Portal Server

WebSphere Portal Server has the following main features:

- The portal application is modular, which enables developers to create their application in smaller, more compact units, independent of other parts of the application.
- WebSphere Portal Server supports portlets that are developed according to the Java Portlet Specification defined by Java Specification Requests (JSRs) 168 and JSR 286.
- It has a more sophisticated authorization infrastructure than the WebSphere Application Server.
- Users and user groups can have their own page structure.
- Users can customize the pages.
- It has its own administration interface to manage users, page structures, and specific settings.
- It has a theme and skin definition that define the overall appearance and functionality of the WebSphere Portal page. Both the theme and the skin can be customized for customer needs.
- It can be built in a cluster to provide a highly available environment, similar to WebSphere Application Server.
- WebSphere Portal Server has a portlet interface such as Lotus Domino (email and database portlets), IBM Cognos® BI, and others.
- WebSphere Portal Server has many built-in portlets. Some of them support third-party application integration, such as Microsoft Outlook portlet, which can display the Outlook mailbox in a WebSphere Portal page.

For more information about the WebSphere Portal Server solution, see:

http://www.ibm.com/software/genservers/portal/server/index.html

3.8.2 Integration with WebSphere Application Server

The WebSphere Portal Server is a web application that is deployed to WebSphere Application Server. In addition to the tight integration, another option is available to connect WebSphere Portal Server with WebSphere Application Server.

WebSphere Application Server also supports the JSR 168 and JSR 286 specifications. WebSphere Application Server can receive and handle portlet rendering requests. By using the Web Services for Remote Portlets (WSRP) protocol, the portal can generate the portlet content in a remote WebSphere Application Server. The portal can then render this content by using the Portal Server aggregation engine.

3.9 IBM Tivoli Workload Scheduler

This section provides information about the integration of Tivoli Workload Scheduler with WebSphere Application Server.

3.9.1 Features of Tivoli Workload Scheduler

Tivoli Workload Scheduler helps you establish an enterprise workload automation backbone by driving composite workloads according to business policies. It provides automation capabilities to control the processing of an enterprise's production workload, including batch and online services. Tivoli Workload Scheduler functions as an automatic driver for composite workloads by maximizing the velocity of workloads, optimizing IT resource usage, and resolving workload dependencies. It extends the scope for integrated application and systems management by driving workloads on multiple, heterogeneous platforms and ERP systems.

Distributed and z/OS components can be used in a mix of configurations according to your business needs or organizational structure. Configurations can be a distributed workload automation environment, a z/OS environment, or a combination of a z/OS and distributed environment. The Tivoli Workload Scheduler for Applications component extends Tivoli Workload Scheduler to automate workloads on both ERP systems and non-native platforms.

Tivoli Workload Scheduler provides the following features:

- Allows enterprises to scale from small to large environments, and to run critical services day after day.
- Provides both calendar-based and event-based workload automation, which provides flexibility when moving from a static, platform-based view of production workloads to a dynamic, service-driven environment.
- Provides a central point to view and manage composite workloads to fine-tune performance and to handle exceptions. This central view allows you to create production reports and generate alerts based on workload, application, or system events.
- Provides open, standards-based APIs that allow you to extend workload automation control to custom and heritage applications. You can also build composite batch services, integrate batch services with online services, and fully automate all composite workloads, including both batch and online services.
- Uses an SOA based on IBM WebSphere components that allows you to control Java EE workloads and web services invocations. You can also manage dependencies between online and batch services.
- Uses IBM DB2 or makes capable the use of Oracle Database.

- Provides standards-based integration with grid computing technologies that allows you to use existing investments. You can dispatch and manage batch workloads across high performance computing grids.
- Provides integration with other Tivoli products for monitoring workload automation events in a business context or for additional autonomic capabilities.

For more information about IBM Tivoli Workload Scheduler, see:

http://www.ibm.com/software/tivoli/products/scheduler/

3.9.2 Integration with WebSphere Application Server

WebSphere Application Server V8.5 includes support for Java batch functions by using WebSphere Batch. See Chapter 6, “WebSphere Batch” on page 137 to learn more about this support.

Tivoli Workload Scheduler is an enterprise scheduler and it serves as an integration point from where the entire enterprise batch infrastructure is managed centrally. The WebSphere Application Server batch infrastructure works alongside enterprise schedulers, and provides a single destination where enterprise schedulers, like Tivoli Workload Scheduler, can dispatch.

To enable management of WebSphere batch jobs from external schedulers, such as Tivoli Workload Scheduler, use a WSGrid workload connector. *WSGrid* is a JMS client application that supports a synchronous job execution over a bidirectional JMS communication with either bus (all platforms) or WebSphere MQ (z/OS only).

When both WebSphere Application Server and Tivoli Workload Scheduler are on z/OS, WebSphere batch job integrates with Job Entry Subsystem (JES). This configuration allows jobs to be submitted by job control language (JCL). The JCL job step starts WSGrid to submit and monitor batch job. WSGrid writes intermediary results of the job into the log of the JCL job. WSGrid does not return the result until the underlying job is complete, providing a synchronous execution model. Tivoli Workload Scheduler is familiar with how to manage JES batch jobs and, by proxy, is also able to manage WebSphere batch jobs.

Figure 3-8 illustrates this integration with Tivoli Workload Scheduler on z/OS.

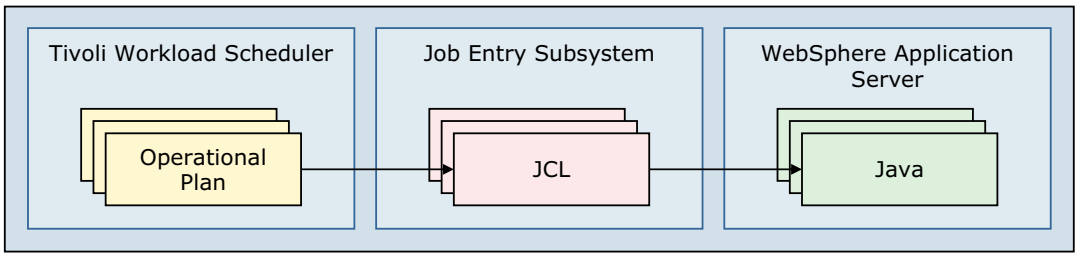


Figure 3-8 Integration with Tivoli Workload Scheduler on z/OS

For distributed systems, the Java-based adapter of WSGrid bridges the gap, by proxy, between the enterprise scheduler and Compute Grid. Figure 3-9 shows an example of this integration.

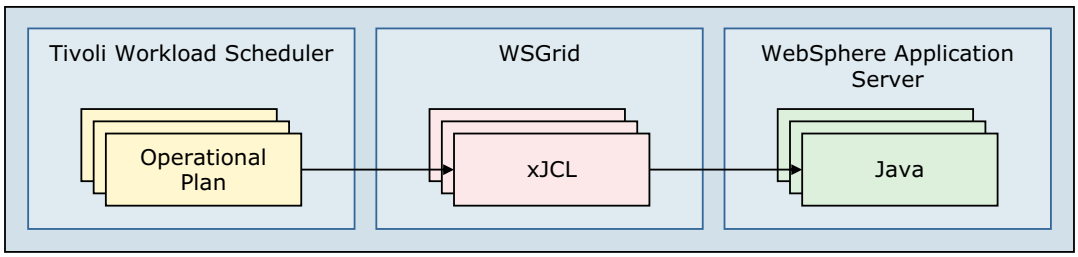


Figure 3-9 Integration with Tivoli Workload Scheduler on distributed systems

For more information about external scheduler integration for WebSphere Application Server, see:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=RconCommandAssist

3.10 IBM WebSphere eXtreme Scale

WebSphere eXtreme Scale provides an extensible framework to simplify the caching of data used by an application. It can be used to build a highly scalable, fault tolerant data grid with nearly unlimited horizontal scaling capabilities. WebSphere eXtreme Scale creates an infrastructure with the ability to deal with extreme levels of data processing and performance. When the data and resulting transactions experience incremental or exponential growth, the business performance does not suffer. There is no impact because the grid is easily extended by adding additional capacity in the form of Java virtual machines and hardware.

This section provides information about the integration of WebSphere eXtreme Scale with WebSphere Application Server.

3.10.1 Features of WebSphere eXtreme Scale

WebSphere eXtreme Scale includes the following key features:

- A highly available elastic and scalable grid
- Extreme transaction support
- Security
- Easy integration into existing solutions
- Support for JSE, Java EE, ADO.NET data services, and REST capable client applications
- Monitoring

A WebSphere eXtreme Scale grid can be effectively used as a data cache for a database or other data sources. These sources are generally slower to respond due to the need to access data on a hard disk. They are also generally more complicated and expensive to scale beyond a single instance, and the total throughput is limited. A WebSphere eXtreme Scale grid has no such limitation, and, when used properly, can scale with a linear response time without any reasonable upper bound.

3.10.2 Integration with WebSphere Application Server

You can integrate WebSphere eXtreme Scale with WebSphere Application Server in the following scenarios:

- Caching HTTP Sessions

  WebSphere eXtreme Scale can replace the existing HTTP session state management facilities of the application server by placing the session data in a data grid. It fetches user session information from the grid and writes changes back to the grid as necessary. Because the HTTP session data is transient in nature, it does not need to be backed up to disk. Therefore, it can be contained completely in a highly available replicated grid. The grid is not constrained to any one application server product or to any particular management unit, such as WebSphere Application Server cells. User sessions can be shared between any set of application servers, and even across data centers in the case of a failover scenario. This sharing allows for a more reliable and fault tolerant user session state.

  For more information, see the following IBM developerWorks® article:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1112_shenoy/1112_shenoy.html?ca=drs-

- Dynamic caching

  Many web-based applications use dynamic page generation techniques, such as JavaServer Pages (JSP). JSP are used for data that rarely changes, such as product details or information about corporate policies. WebSphere Application Server provides an in-memory dynamic cache. This cache is used to store the generated output the first time the page is rendered. It saves both the processing work and back-end system load for subsequent requests. You can configure dynamic cache to use WebSphere eXtreme Scale as your cache provider instead of the default dynamic cache engine.

  For more information, see the WebSphere Application Server V8.5 Information Center at:

  http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tdyn_extremescale

4

An overview of the Liberty profile

This chapter introduces the WebSphere Application Server V8.5 Liberty profile. This chapter includes the following sections:

- Introduction to the Liberty profile
- Installing the Liberty profile
- Configuring the Liberty profile
- Administering the Liberty profile
- Developing and deploying a Liberty profile application
- The Liberty profile application security
- The Liberty profile deployment topologies
- Troubleshooting

For more information about the Liberty profile, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=cwlp_about

4.1 Introduction to the Liberty profile

The Liberty profile is a dynamic and composable profile of WebSphere Application Server V8.5. It enables WebSphere Application Server to provision only the features that are required by the application (or set of applications) that are deployed to the server. For example, if an application requires just a servlet engine, Liberty profile can be configured to start the WebSphere Application Server kernel, the HTTP transport, and the web container. The Liberty profile therefore starts quickly and has a small footprint.

The Liberty profile provides a simplified and lightweight development and application-serving environment that is optimized for developer and operational productivity. This profile is intended for use as a development or production environment for running web applications that do not require a full Java Platform, Enterprise Edition (Java EE) stack. The Liberty profile provides enterprise qualities of service, including security and transaction integrity.

The Liberty profile includes the following key features:

- A dynamic and flexible run time to load only what the application needs
- A quick startup time (under 5 seconds with simple web applications)
- A simplified configuration that uses a single configuration file or modular configuration
- Support for deploying applications developed in the Liberty profile to run in the full profile
- Support of web applications, OSGi applications, and Java Persistence API
- Support for LDAP registry
- Ability to deploy an application and configured server as a package
- Managed, centralized deployment for many nodes of a packaged application and server
- Availability of WebSphere Application Server Developer Tools as Eclipse plug-ins for broad tool support
- Support for z/OS platform native features like System Authorization Facility (SAF), Resource Recovery Services (RRS), and z/OS Workload Manager (WLM)

The Liberty profile provides a development and a test environment as well as a production environment on all WebSphere Application Server V8.5 supported platforms. Additionally, the Liberty profile provides a development environment on the Macintosh operating system.

This section provides an overview of the Liberty profile and illustrates the high-level architecture. It also explains the features that are supported and the concept of dynamic feature management.

4.1.1 The Liberty profile architecture

As depicted in Figure 4-1, the Liberty profile is built on OSGi technologies. The server process runs as OSGi bundles and comprises a single Java virtual machine (JVM), the Liberty profile kernel, and any number of optional features.

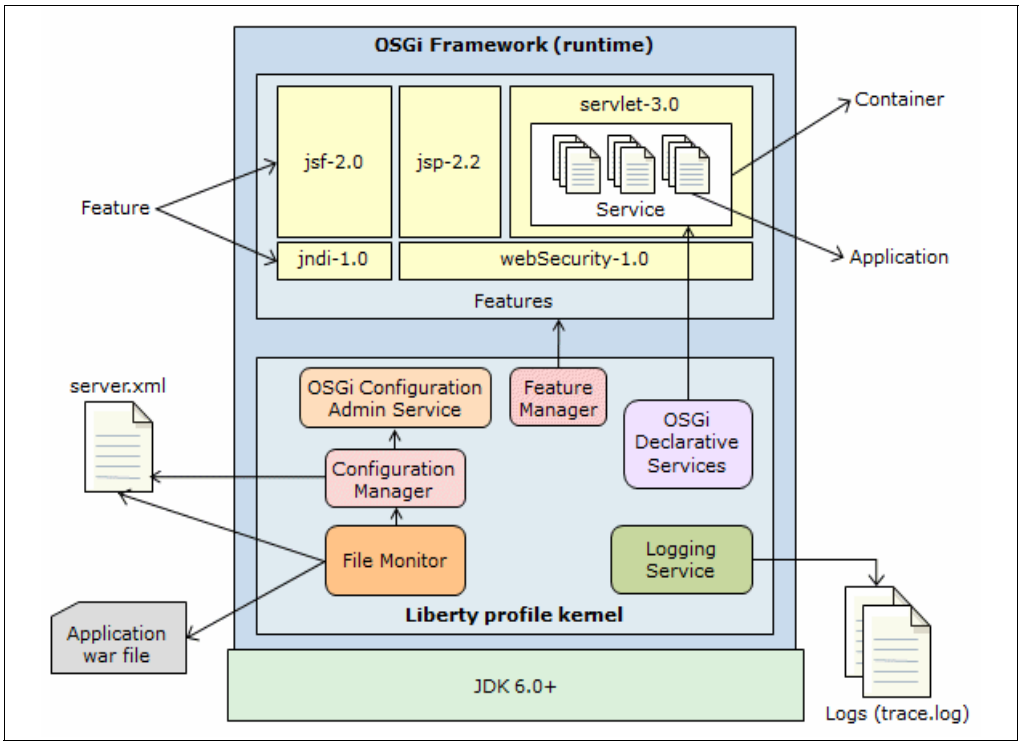


Figure 4-1 The Liberty profile architecture

A functional server is produced by starting the runtime environment with a configuration that includes a list of features that are to be used. *Features* are the units of capability by which the runtime environment is defined and controlled. They are the primary mechanism that makes the server composable. For example, if the servlet feature is specified, the runtime environment operates as a servlet engine.

By default, a server runs with no features. You can use the feature manager to add the features that are needed. The feature manager is one of the kernel bundles that runs these functions:

- Receives the configuration
- Resolves each feature to a list of bundles
- Installs the feature into the framework
- Starts the feature

When the features are specified, the default configuration provides a rich environment that is designed to cover most common requirements.

The configuration manager reads the server configuration from persistent files. It parses the configuration into sets of properties, then uses those sets of properties to populate the OSGi Configuration Admin service. This service maintains the runtime view of the configuration. When configuration updates are made, this service injects each set of properties into the service that “owns” them.

4.1.2 The Liberty profile feature management

The Liberty profile includes the following main features:

- Basic web application security
- Bean validation
- Blueprint
- Java Database Connectivity (JDBC)
- Java Management Extensions (JMX)
- Java Persistence API
- JavaServer Faces (JSF)
- JavaServer Pages (JSP)
- Secure Sockets Layer (SSL)
- Security
- Servlet
- Web application bundle (WAB)
- Web security
- z/OS security
- z/OS transaction management
- z/OS workload management

Each feature has a *version identifier*. This identifier is provided so that multiple versions of the same feature can be used in subsequent releases. Components are written so that multiple versions can run in the same process.

The feature manager maps each feature name to a list of bundles that provide the feature. When a feature configuration is changed, the feature manager recalculates the list of required bundles. It stops and uninstalls those bundles that are no longer needed, and then installs and starts any additions. All features are designed to cope with other features that are added or removed dynamically.

Figure 4-2 depicts dynamic feature management in the Liberty profile.

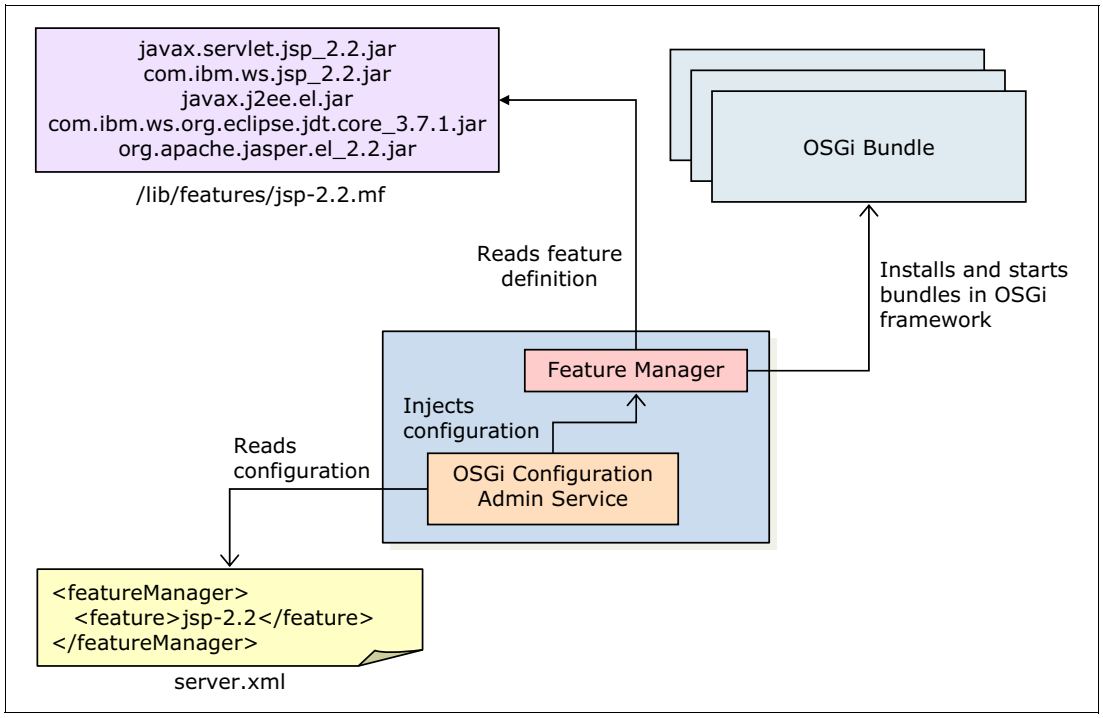


Figure 4-2 Dynamic feature management in the Liberty profile

For more information, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=cwlp_feat_mgmt

4.2 Installing the Liberty profile

You can use any of the following methods to install the Liberty profile:

- Extracting an archive file that contains the distribution image to a local folder
- Using Installation Manager
- Installing the Liberty profile developer tools

For more information about installing the Liberty profile, see 9.8, “Planning for the Liberty profile” on page 268.

4.3 Configuring the Liberty profile

The Liberty profile configuration operates from a set of built-in configuration defaults. You can specify only the required changes for your environment by using a simple XML format. This section provides details about how to configure the Liberty profile.

4.3.1 Liberty profile configuration characteristics

The Liberty profile configuration has the following characteristics:

- The persistent configuration has these characteristics:
  - Described in XML files
  - Small, easy to back up, and easy to copy to another system
  - Human readable and editable in a text editor
  - Shareable with the entire application development team
  - Composed such that features can add configurations to the system easily
- The runtime configuration has these characteristics:
  - Injected into the owning components on an update
  - Dynamically composable so that configuration for features can be added to or removed from the system easily
  - Supports zero cost migration between releases
- Configuration used by components is dynamically responsive to updates and forgiving. Missing values are assumed and unrecognized properties are ignored.

4.3.2 Simplified configuration

A Liberty profile server configuration consists of a `bootstrap.properties` file, a `server.xml` file and any (optional) files that are included by these files. The `bootstrap.properties` file specifies properties that need to be available before the main configuration is processed. These properties are kept to a minimum. The `server.xml` file is the primary configuration file for the server, and the file that users work with the most.

The `server.xml` file (and any files included with it) has a simple XML format that is suitable for text editors. The only required entry is to indicate that the file contains a server definition, as shown in Example 4-1.

Example 4-1 Server definition entry in the server.xml file

```
<server/>
```

You need to specify only overrides and additions to the default configuration values. Example 4-2 shows values that change the transaction timeout value.

Example 4-2 Transaction configuration entry in the server.xml file

```
<transaction timeout="30"/>
```

Example 4-3 shows an example of using a list of values. This example lists the features that are provided by the server.

Example 4-3 List of features in the server.xml file

```
<featureManager>
     <feature>jsp-2.2</feature>
     <feature>derby-10.8</feature>
</featureManager>
```

When a resource, such as an application, is configured, provide only the attributes that are unique for the resource. The other attributes can remain with their default values, as illustrated in Example 4-4.

Example 4-4 An application entry in the server.xml file

```
<application location="tradelite.war" />
```

Example 4-5 shows an example of a complete server configuration to run a web application.

Example 4-5 A complete server configuration defined in the server.xml file

```
<server>
   <featureManager>
      <feature>jsp-2.2</feature>
      <feature>derby-10.8</feature>
   </featureManager>
   <transaction timeout="30"/>
   <logging traceSpecification="webcontainer=all=enabled:*=info=enabled" />
   <application type="war" id="tradelite" name="tradelite"
location="tradelite.war"/>
   <jdbcDriver id="DerbyEmbedded" libraryRef="DerbyLib"/>
      <library id="DerbyLib">
         <fileset dir="${shared.resource.dir}/derby" includes="derby.jar"/>
      </library>
   < datasource id="DefaultDatasource" jdbcDriverRef="DerbyEmbedded"
jndiName="DefaultDatasource">
      <properties createDatabase="create"
databaseName="${shared.resource.dir}/data/product"/>
   </datasource>
</server>
```

For a list of configuration elements, their subelements, and the attributes that are supported in the `server.xml` file, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=rw4d_metatype_4ic

4.3.3 Flexible configuration

You can use the Liberty profile configuration at any level of granularity, from a single file to several files. Several servers can point to a remote XML file for common shareable configuration. This flexible configuration can be achieved by using shareable configuration snippets in the `server.xml` file, as illustrated in Example 4-6.

Example 4-6 Example of using shareable configuration snippets

```
<server>
    ...
    <include location="http://cfgserver/global.xml" />
    <include location="${shared.config.dir}\global.xml" />
<server>
```

You can use WebSphere Developer Tools for Eclipse to associate configuration snippets with a server configuration. Figure 4-3 on page 98 depicts this flexible configuration.

4.3.4 Dynamic configuration

In a Liberty profile configuration, the features of the profile provide the configuration default values. Thus, user-specified configuration is kept to a minimum. Any property can be overridden in a user-specific server configuration, and any changes made to the configuration are dynamically injected into the contributing feature immediately. There is no need to restart the server.

This dynamic configuration provides greater operational productivity to developers as they build the capabilities of an application, modify classes, add resources, and fix problems. The code and configuration changes that developers make can be reflected immediately in the test environment.

Figure 4-3 depicts this dynamic configuration.

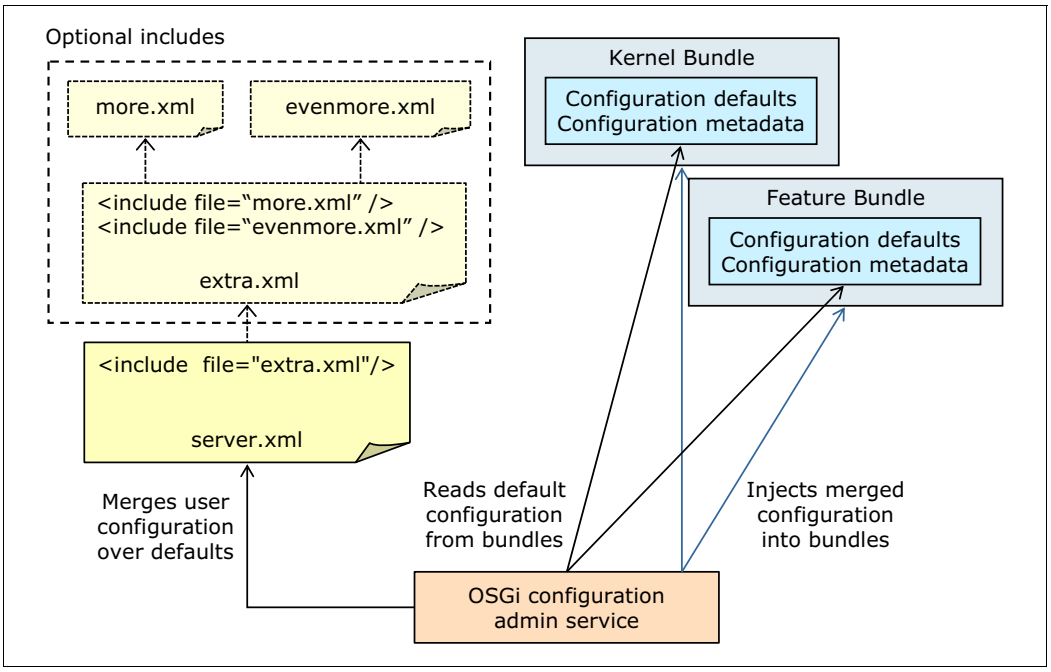


Figure 4-3 Flexible and dynamic configuration

4.4 Administering the Liberty profile

In WebSphere Application Server V8.5, you can easily administer the Liberty profile configuration files, configure the Liberty profile with web server plug-ins, and capture the Liberty profile server status. You can also package a Liberty profile server configuration with the applications that it runs for distribution to colleagues or for installation on other systems.

This section provides information about administering the Liberty profile.

4.4.1 Administering the Liberty profile configuration files

As mentioned previously, a Liberty profile server configuration consists of the following files:

- A `server.xml` file
- A `bootstrap.properties` file
- Any optional files that are included by these two main configuration files

There is no administrative console for the Liberty profile. However, administrators and developers can use the Liberty profile developer tools or a text editor to edit the configuration files.

The `server.xml` file is the primary configuration file for the server. You can edit this file in a text editor. Alternatively, you can use the editor that is part of the Liberty profile developer tools to edit the file. The `bootstrap.properties` file specifies properties that need to be available before the main configuration is processed.

The `server.env` file can be used for specifying environment variables and `jvm.options` file can be used to customize JVM options.

For more information about how to configure the Liberty profile runtime environment, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=twlp_setup_env

4.4.2 Configuring the Liberty profile with a web server plug-in

With WebSphere Application Server V8.5, you can configure a web server plug-in for the Liberty profile. When the web server receives an HTTP request for dynamic resources, the request is forwarded to the Liberty profile.

For more information about how to configure the Liberty profile with a web server plug-in, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=twlp_admin_webserver_plugin

4.4.3 Capturing the debug information for a Liberty profile server

WebSphere Application Server V8.5 provides the **`server dump`** command for problem diagnosis for a Liberty profile server. The file generated by this command contains server configuration, log information, and details of the deployed applications in the work area directory. A running server usually includes the following information:

- State of each OSGi bundle in the server
- Wiring information for each OSGi bundle in the server
- A component list that is managed by the Service Component Runtime (SCR)
- Detailed information about each component from SCR

For more information, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=twlp_setup_dump_server

4.4.4 Packaging a Liberty profile

Because a Liberty profile server is lightweight, it can be packaged easily with applications in a compressed file. This package can be stored, distributed to colleagues, and used to deploy the application to a different location or to another system. It can even be embedded in the product distribution.

For more details about how to package the Liberty profile, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=twlp_setup_package_server

4.4.5 Administering a Liberty profile on z/OS

WebSphere Application Server V8.5 provides features for administering a Liberty profile on a z/OS platform. You can use IBM MVS™ operator commands to start, stop, or modify the Liberty profile.

For more information, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=twlp_admin_zos

4.5 Developing and deploying a Liberty profile application

A Liberty profile supports web applications, OSGi applications, and Java Persistence API. Associated services, such as transaction and security, are supported for these application types and for Java Persistence API. With its lightweight and easy installation, and quick to use features, the Liberty profile provides a convenient and capable platform for developing and testing web and OSGi applications. This platform is beneficial when you are developing an application to run on the WebSphere Application Server full profile. Any application that runs on the Liberty profile will also run on the full profile.

Liberty profile provides the following options to deploy an application:

- Dropping the application into a previously defined “dropins” directory

  You can use the “dropins” directory for applications that do not require additional configuration, such as security role mapping.

- Adding an application entry to the server configuration file

  For applications that are not in a “dropins” directory, you can specify the location of the application as an entry in the server configuration file. The location of the server configuration file can be either a file system path or a URL.

You can use the developer tools that are supported by WebSphere Application Server V8.5 to develop and deploy applications to a Liberty profile. For more information, see Chapter 11, “Application development and deployment” on page 341.

4.6 The Liberty profile application security

The *appSecurity-1.0* feature of the Liberty profile provides support for securing the server runtime environment and web applications. The appSecurity-1.0 feature provides support for user registries, authentication, and authorization. The supported user registry types are basic user registry and LDAP user registry.

For secure communication between the client and the server, you can enable SSL for the Liberty profile. A minimal or detailed configuration can be done by adding the ssl-1.0 server feature to the server configuration file.

For authenticating users, the Liberty profile supports the following configurations:

- A basic user registry that defines user and group information for authentication to the Liberty profile
- A Lightweight Directory Access Protocol (LDAP) server for authentication
- A third-party security service using a trust association interceptor (TAI)

- Single sign-on (SSO) so that web users can authenticate once when accessing the Liberty profile resources such as HTML, JSP files, and servlets. Users can also authenticate once when accessing resources in multiple Liberty profile servers that share Lightweight Third Party Authentication (LTPA) keys
- A custom Java Authentication and Authorization Service (JAAS) login module to make additional authentication decisions or to make finer-grained authorization decisions inside an application

To configure authorizations for an application, you can add authorization tables to the application. The server then reads the deployment descriptor of the application to determine whether the user or group has the privilege to access the resource.

The Liberty profile server also provides various plug points that extend the security infrastructure.

For more information, see 15.10, “Securing the Liberty profile” on page 497.

4.7 The Liberty profile deployment topologies

The Liberty profile is designed to support different ways of preparing a compressed file for deployment. The simplest method is to store all resources in a compressed file. However, resources can be stored as read-only for sharing in some environments. If deployed on a single host, multiple servers can use the shared resources on that host. If deployed to a shared disk, servers on multiple hosts can share the resources.

A Liberty deployment can include the following types of resources:

- Project

  The project is used optionally as a container for resources. Related resources can be grouped under the same project for ease of management and to avoid name conflicts with resources from other projects.
- Run time (WebSphere Liberty profile)

  The run time includes the **bin**, **lib**, and **lafiles** binary files.
- Liberty_server

  The Liberty_server directory contains the following server definitions:

  – A self-contained directory that includes the `server.env` file, the `jvm.options` file, the `server.xml` file, and other configuration files and working directories.

  – A template directory that contains just the `server.xml` file and other configuration files. This directory allows one set of configuration files to be standardized and referred to from multiple server instances.

  – A localized directory that contains only the `server.env` file, the `jvm.options` file, the working directory, and a pointer to a template directory. The localization directory contains only host-specific information:

    - Host name
    - The location of the software development kit (SDK)
    - A pointer to the server template directory
    - The location of the application or applications

- Application_binary

  The Application_binary is an archive or a directory that contains the application. This archive or directory might or might not be deployed to a Liberty profile server.

- SDK

  The Java software development kit is used to run the Liberty profile servers.

A Liberty profile image is an archive file that contains one or more types of resources of the Liberty profile environment, depending on the topology that is deployed. You can extract them manually or can use an extraction tool to deploy the file to one or more systems. Alternatively, you can use the job manager to deploy these images.

In WebSphere Application Server V8.5, use the job manager to perform these functions:

- Package the Liberty profile runtime environments, configurations, and applications
- Distribute and deploy a Liberty profile server and applications
- Start embedded profile packages

For more information about managing the Liberty profile with a job manager, see 8.3.3, “Liberty profiles managed by a job manager” on page 202.

4.7.1 Example topology 1

Figure 4-4 illustrates a self-contained topology in which the compressed file contains the following resources:

- A Liberty profile run time, shown as WebSphere Liberty profile in Figure 4-4
- An SDK
- A Liberty profile server
- An application

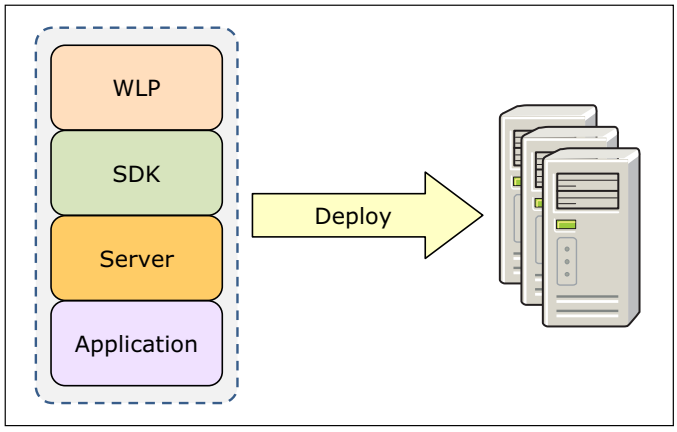


Figure 4-4 Self-contained topology 1

4.7.2 Example topology 2

Figure 4-5 illustrates a self-contained topology in which the compressed file contains the following components:

- A Liberty profile run time
- A Liberty profile server
- An application

The SDK is preinstalled on each host.

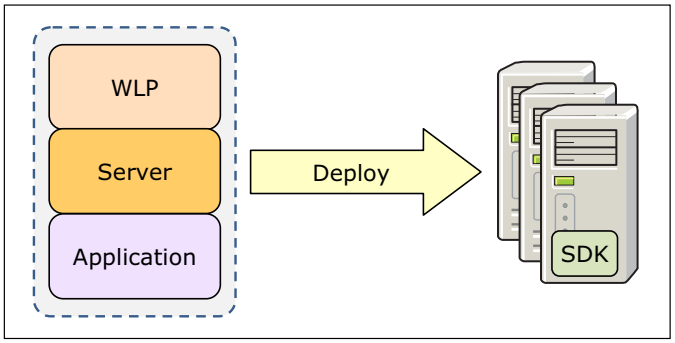


Figure 4-5 Self-contained topology 2

4.7.3 Example topology 3

Figure 4-6 illustrates a shared topology that involves installation of multiple compressed files. The Liberty profile server and the application are contained within each compressed file. The SDK and WebSphere Liberty profile, however, are preinstalled and shared by different servers.

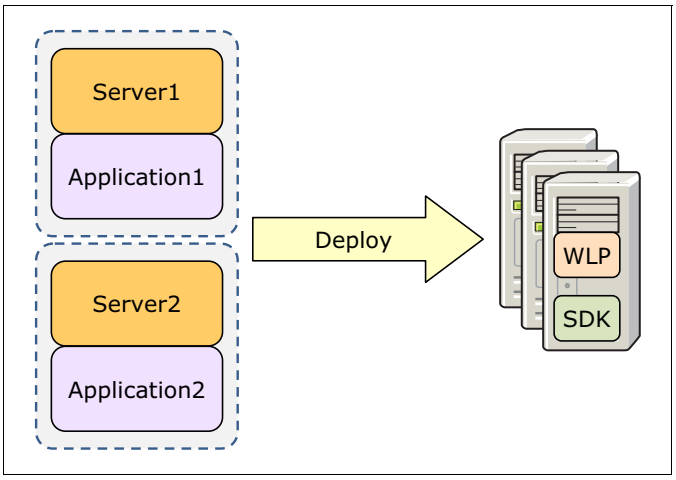


Figure 4-6 Shared topology 1

4.7.4 Example topology 4

Figure 4-7 illustrates a shared topology where each compressed file contains only the Liberty profile server definition. Applications are predeployed as read-only and shared across different servers. The Liberty profile and SDK are preinstalled and shared by different servers.

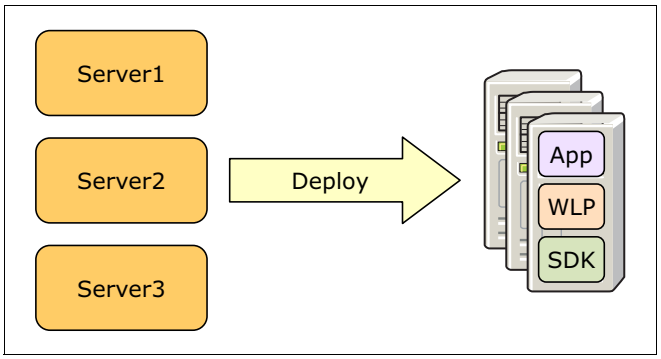


Figure 4-7 Shared topology 2

4.7.5 Example topology 5

Figure 4-8 illustrates a shared topology where each compressed file contains only the Liberty profile server definition. Shared artifacts are placed on shared disks and accessed by multiple servers. This topology has a single point of failure, and therefore is not recommended for a production environment.

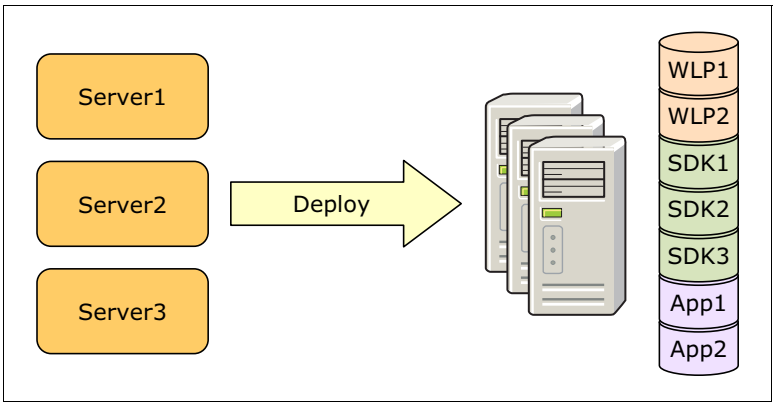


Figure 4-8 Shared topology 3

For more information, see Appendix B, “Sample topology using the job manager and a Liberty profile” on page 589.

4.8 Troubleshooting

Similar to the WebSphere Application Server V8.5 full profile, the Liberty profile provides the base implementations for logs, traces, and first-failure data capture (FFDC). You can control the logging service through either the server configuration file or through the `bootstrap.properties` file.

Logging properties in the `bootstrap.properties` file take effect before the server configuration files are processed. This process is useful for problem determination during

early server start or configuration processing. If you set different logging properties in the `bootstrap.properties` file and the server configuration file, the server configuration file takes precedence by default. This behavior can be overridden by specifying an `override.bootstrap.properties` property with a false value in the server configuration file.

Remember: The server configuration can be changed dynamically (that is, configuration changes can take effect while the server is running). However, changes to the `bootstrap.properties` file take effect only on a server restart.

You can set logging properties in the server configuration file by using developer tools or by adding a logging component to the server configuration file (Example 4-7).

Example 4-7 Trace specification in server configuration file

```
<logging traceSpecification="*=audit=enabled:com.myco.mypackage.*=debug=enabled"/>
```

For further details about the trace and logging feature in the Liberty profile, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=rwlp_logging

The Liberty profile uses the Eclipse Equinox implementation of the OSGi core specification. Eclipse Equinox currently provides an OSGi console that can be used to aid with debugging. This console is not available by default, but can be enabled by configuring a port for it. For more information about the Eclipse OSGi console, see the following developerWorks topic:

http://www.ibm.com/developerworks/library/os-ecl-osgiconsole/

5

Intelligent Management

This chapter addresses the concepts of the Intelligent Management capabilities that are introduced in IBM WebSphere Application Server V8.5.

Intelligent Management provides a virtualized infrastructure that redefines the traditional concepts of Java Platform, Enterprise Edition (Java EE) resources and applications, and their relationships. This application infrastructure virtualization allows the product to automate operations in an optimal manner, increasing the quality of service. By introducing an automated operating environment with workload management, you can reduce total cost of ownership by performing more work using less hardware.

This chapter includes the following sections:

- Introduction to Intelligent Management
- Virtualization, autonomic, and cloud computing
- Intelligent routing and dynamic operations
- Dynamic workload management
- Health management
- Application edition management
- Performance management
- Planning for hosting dynamic operations

5.1 Introduction to Intelligent Management

Intelligent Management features extend the quality of service provided by your middleware environment. Configurable operational policies govern the performance and health of your applications. Total cost of ownership is decreased through server consolidation and less administrative effort, and you experience lower response times and increased availability. In short, you experience the benefits of an autonomic middleware environment, which is self-configuring, self-protecting, self-healing, and self optimizing.

A key component of the Intelligent Management is the *on-demand router*. The on-demand router is a proxy server based on Java that proxies both the HTTP and SIP protocols. The on-demand router supports health, application edition, and performance management features. It can manage both WebSphere and non-WebSphere environments. The on-demand router can queue requests for less important applications so that requests from more important applications are handled quickly.

Intelligent Management includes the following primary features:

- *Intelligent routing* improves business results by ensuring priority is given to business critical applications. Requests to applications are prioritized and routed based on administrator-defined rules.
- *Health management* allows you to specify conditions to automatically watch for and corrective actions to take when the conditions are observed. You can monitor the status of your application servers, sense problem areas, and then respond to these problem areas before an outage occurs. The health monitoring and management subsystem continuously monitors the operation of servers against user-defined health policies. It detects functional degradation that is related to user application malfunctions.
- *Application edition management* allows you to roll out new versions of applications without experiencing downtime for a maintenance window. You can manage interruption-free production application deployments by using this feature. You can also validate a new edition of an application in your production environment without affecting users, and upgrade your applications without incurring outages to your users. You can also run multiple editions of a single application concurrently, directing different users to different editions.
- *Performance management* provides a self-optimizing middleware infrastructure. Dynamic clusters automatically scales up and down the number of running cluster members as needed to meet response time goals for users. You can take advantage of overload protection to limit the rate at which the on-demand router forwards traffic to application servers. Doing so helps prevent heap exhaustion, processor exhaustion, or both from occurring.

All of these capabilities together allow you to extend qualities of service through autonomic computing. These capabilities are called *dynamic operations*, which are the core functions that provide application infrastructure virtualization.

Intelligent Management is the integration of WebSphere Virtual Enterprise into WebSphere Application Server Network Deployment V8.5. The Intelligent Management functionality includes the following key features:

- Improved application performance and response times to meet service level agreements
- Increased application availability and minimized administration costs
- Interruption-free maintenance upgrades

The Intelligent Management functionality also provides support for a range of middleware servers. Middleware servers are all servers in the middleware tier that provide the infrastructure for applications or their data.

Middleware server support includes the following servers:

- Apache HTTP Server
- Apache Geronimo Server
- External WebSphere application servers
- WebSphere Application Server Community Edition

The term *Complete lifecycle server* includes any server that the environment can instantiate, or create. These server types include WebSphere Application Server types such as application servers, generic servers, web servers, and proxy servers.

The term *assisted lifecycle server* refers to servers that you define to WebSphere Application Server by using templates to create representations of the servers in the administrative console. However, these servers still exist within the administrative domain of their respective middleware platform. You add them as generic servers to the deployment manager capabilities. You can control the servers operationally, monitor and view server health and performance, and configure the administrative console to display log files and configuration files for these servers.

5.2 Virtualization, autonomic, and cloud computing

With Intelligent Management, virtualization, autonomic computing, and cloud computing are integrated into the single architectural model of WebSphere Application Server V8.5. The following sections describe these concepts.

5.2.1 Virtualization

This section describes the following concepts of virtualization:

- Application infrastructure virtualization
- Server virtualization

Application infrastructure virtualization

Typically, Java applications and resources are statically bound to a specific server. They often experience increases in load that last a short time. The most costly time for an application to become unavailable is during a period of high demand. Therefore, you need to build IT infrastructures to accommodate these peaks. Normally, when systems experience normal load, a large percentage of computing capacity goes unused, making inefficient use of IT investments.

By configuring application infrastructure virtualization in WebSphere Application Server with the Intelligent Management functionality, resources are pooled. This pool accommodates the fluctuations of workload in the environment, increasing the quality of service. You effectively break the bond between applications and the physical infrastructure on which they are hosted. Workloads are then dynamically placed and spread across a pool of application server resources, which allows the infrastructure to adapt and respond to business needs. Requests are prioritized and intelligently routed to respond to the most critical applications and users.

The static relationships of an application with the server to which it is deployed is replaced with a dynamic relationship. This relationship has looser coupling of applications or resources and server instances. Instead of statically binding applications to servers or clusters, you deploy applications to dynamic clusters. These clusters are application deployment targets that can expand and contract, depending on the workload in the environment.

After you deploy applications to dynamic clusters, the placement of the applications is determined by the operational policies that you define. Autonomic managers control the placement of the server instances and how workload is routed to each application. If workload increases for a specific application, the number of server instances for the dynamic cluster that is hosting the application can increase. The application can also use available resources from other applications that are not experiencing increased workload.

Virtualization provides the following benefits:

- Improved management of software and applications

  Management processes become more repeatable and less error-prone by using automated services and operational policies.

- Allocation of software resources

  Dynamic reallocation of resources can occur based on shifting distributions of load among applications.

- Increased number of applications

  More applications can run in a virtualized application environment than in a static configuration.

- Reduced configuration complexity

  Loosened coupling between applications and the application server instances reduces the overall complexity and provides for a better, more usable environment.

You deploy an application to a dynamic cluster that has a *node group* or a specified *membership policy.* A membership policy determines which nodes belong to the cluster. You do not deploy your applications to specific application servers. Instead, the application placement controller starts application server instances for the dynamic cluster based on the settings that you chose for the dynamic cluster.

Tip: The Intelligent Management function reacts to an increase in workload by starting an additional application server. Additional application servers can start on the nodes that are selected by the dynamic cluster membership policy to handle additional requests for the application.

Figure 5-1 illustrates how the workload increases for a specific application. The number of server instances for the dynamic cluster that is hosting the application can increase by using available resources from other applications. In this example, *New Application Server 3* contributes bandwidth to satisfy higher workload requests. The on-demand router manages this dynamic cluster growth and sends requests to the new server.

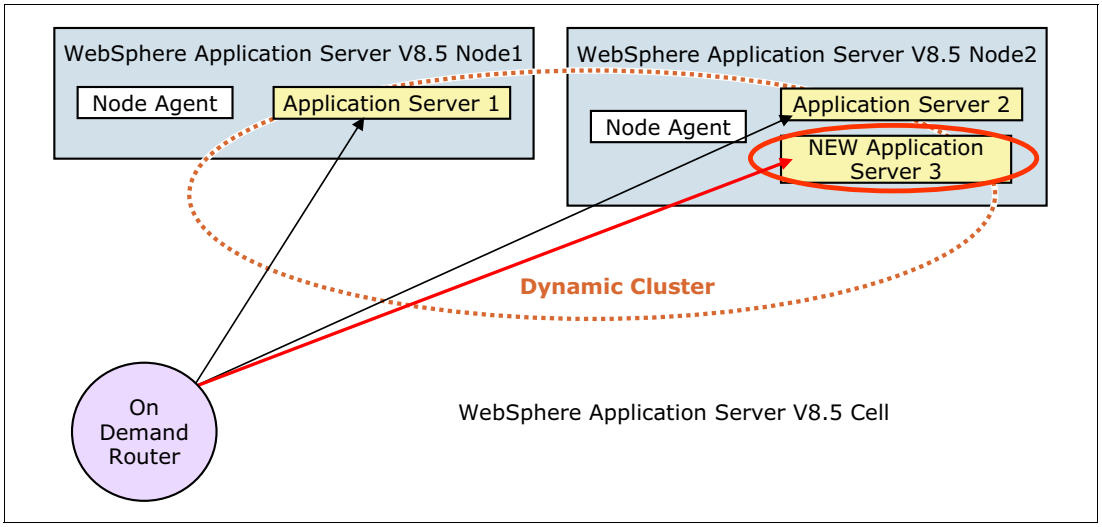


Figure 5-1 Reaction to workload increase

Server virtualization

You can combine the advantages that infrastructure virtualization provides with the advantages of hardware virtualization by using both in the same infrastructure.The hardware virtualization capabilities are provided by the physical hardware on which WebSphere Application Server V8.5 is hosted. Using server virtualization, you can share server resources across the virtual servers or logical partitions, as illustrated in Figure 5-2 on page 112. Server virtualization environments can run in a shared processor mode. When you use shared processor mode, the physical processors are pooled and shared between the servers or logical partitions that are running on the physical computer.

Hardware virtualization is not dependent on Intelligent Management, but Intelligent Management can take advantage of hardware virtualization.

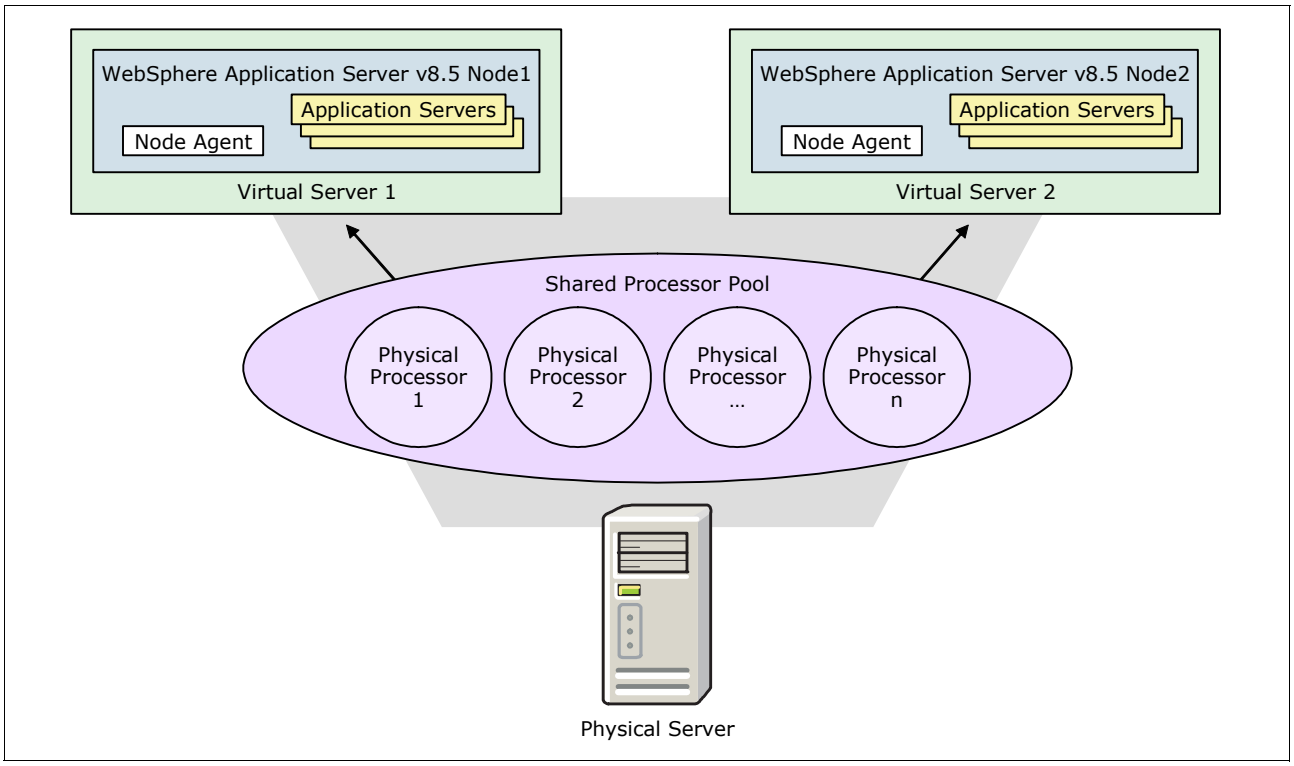


Figure 5-2 Pool of shared processor that is used by virtual servers

Server virtualization provides the following benefits:

- Reduced amount of hardware in the environment

  You can run multiple nodes on the same physical hardware.

- Improved hardware management

  You can more easily manage the environment because you have fewer physical computers and can use the server Virtualization software to manage images.

- High availability of hardware

  By configuring server failover, the physical hardware can be highly available. When one server fails, it can be replaced by another server.

- Dynamic allocation of hardware

  The physical resources, such as processors and memory, on hosting computers can be shared among the virtual servers in the environment and dynamically allocated as needed. Because the resources are allocated dynamically, restarting the servers is not necessary.

- Shared storage

  Multiple virtual servers or logical partitions can share physical storage. You do not need a physical hard disk drive for each virtual machine or LPAR.

For more information about virtualization, see the following YouTube video:

http://www.youtube.com/watch?v=IJM4GIfemT8

5.2.2 Autonomic computing

Autonomic computing refers to the self-managing characteristics of distributed computing resources that adapt to unpredictable changes while hiding intrinsic complexity from operators and users. Autonomic computing has evolved into a set of capabilities that are built into many IBM products.

WebSphere Application Server V8.5 now includes autonomic computing functions. Using Intelligent Management in WebSphere Application Server V8.5, you can create a self-managing environment that serves applications. It can help you overcome the complexity of systems management and reduce the barrier that complexity poses to further growth. An autonomic system makes decisions on its own, using high-level policies. It constantly checks and optimizes the status of the system, and automatically adapts it to changing conditions.

An autonomic computing framework can be composed of *autonomic components* that interact with each other. An autonomic component can be modeled in terms of two main control loops: Local and global. It can include sensors (for self-monitoring), effectors (for self-adjustment), knowledge, and a planner or adapter for using policies based on self-awareness and environmental awareness. In a self-managing autonomic system, the human operator takes on a new role. Instead of controlling the system directly, the human operator defines general policies and rules that guide the self-management process.

For this self-management process, IBM defined the following functional areas:

- Self-configuration: Automatic configuration of components
- Self-healing: Automatic discovery and correction of faults
- Self-optimization: Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements
- Self-protection: Proactive identification and protection from overload conditions

Figure 5-4 illustrates how WebSphere Application Server provides an autonomic system.

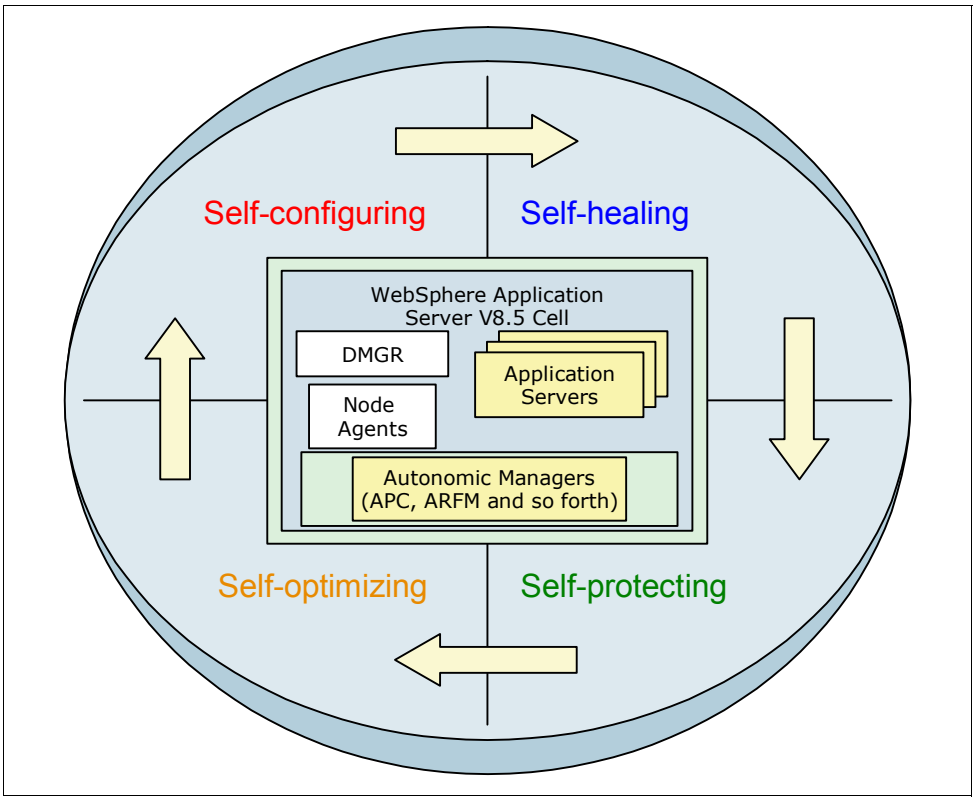


Figure 5-3 WebSphere Application Server V8.5 view as an autonomic system

As a comparative example of an autonomic function, consider the human central nervous system. In the human central nervous system, autonomic controls use motor neurons to send indirect messages to organs at a subconscious level. These messages regulate temperature, breathing, and heart rate without conscious thought.

In a computing environment, a network of organized computing components provide what we need, when we need it, without a conscious mental or physical effort. Autonomic computing is a comprehensive approach that you can use to build automated IT infrastructures that require minimal intervention.

This evolutionary path to autonomic computing is represented by the following levels:

- The *basic level* represents the starting point where a significant number of IT systems are today. Each element of the system is managed independently by systems administrators who set up the element, monitor it, and enhance it as needed.
- At the *managed level*, systems management technologies are used to collect information from disparate systems into one consolidated view. This process reduces the time that it takes for the administrator to collect and synthesize information.
- At the *predictive level*, new technologies are introduced that provide correlation among several elements of the system. The system itself can begin to recognize patterns, predict the optimal configuration, and provide advice on what course of action the administrator needs to take. As these technologies improve, people become more comfortable with the advice and predictive power of the system.

- The *adaptive level* is reached when systems go beyond providing advice on actions and automatically take corrective actions based on what is happening in the system.
- Finally, the *full autonomic level* is attained when the system operation is governed by business policies and objectives. Users interact with the system only to monitor the business processes or alter the objectives.

WebSphere, if configured with all its core components and its core functions as a full autonomic system, can be considered an autonomic element in a context called *computerized ecosystem*. It is also considered an *artificial neural network (ANN)*. In this adaptive system, autonomic elements are groups of interconnected nodes that interact with all other autonomic elements without any human intervention.

Figure 5-4 shows a system of fully autonomic level elements that are interacting with each other without human intervention by using their autonomic managers.

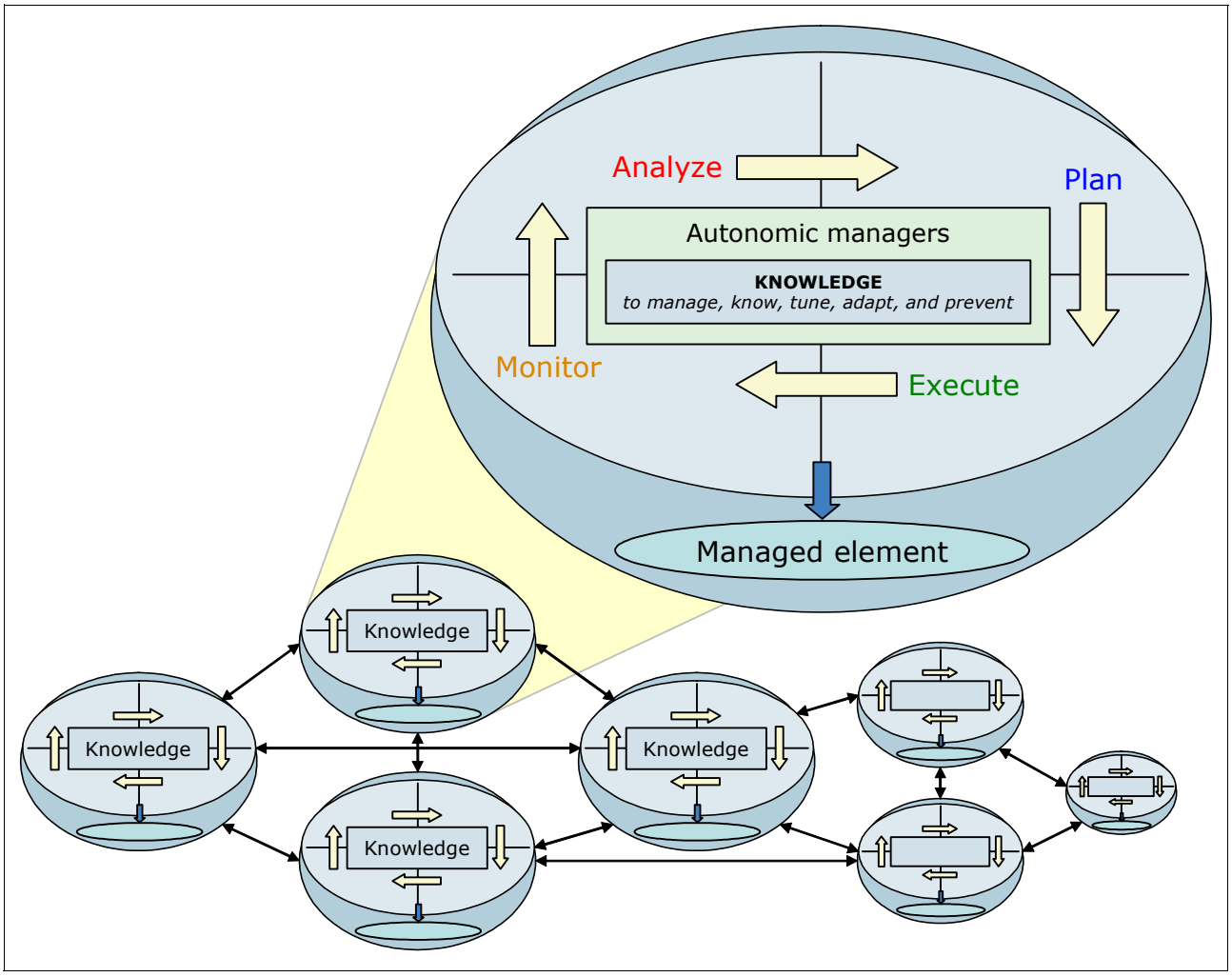


Figure 5-4 Representation of a computerized ecosystem with fully autonomic elements

5.2.3 Cloud computing

Virtualization and autonomic computing are steps toward *cloud computing*. By building a virtualization foundation, you put the secure, scalable, and efficient system in place on which to build a cloud. From an entry-private cloud, you can deploy advanced cloud functions, including full lifecycle management, automated provisioning, metering, and management capabilities.

The cloud service model defines the following services:

- Infrastructure as a service (IaaS)

  IaaS integrates basic services such as virtual servers, data storage, and databases into one platform to deploy applications. IaaS is a web-based service that provisions standard server, storage, network equipment, and software. It uses an automated self-service model. The IaaS model frees resources that would otherwise house, run, and maintain equipment and software. An IaaS approach is ideal for resource-intensive activities such as development, testing, and other dynamic workloads.

- Platform as a service (PaaS)

  PaaS enables developers to build and deploy web applications on a hosted infrastructure. It also allows them to take advantage of the seemingly infinite compute resources of a cloud infrastructure.

- Software as a service (SaaS)

  SaaS provides network-based access to commercially available software. It can lead to increased speed of software development, faster adoption of software, less support requirements, and ease in implementation and upgrades.

Further information: For more information about cloud computing, see:

`http://www.ibm.com/cloud-computing/us/en`

You can also subscribe to the IBM Cloud YouTube channel for latest videos:

`http://www.youtube.com/user/IBMCloud`

5.3 Intelligent routing and dynamic operations

Loss of availability translates into lost business, which means lost opportunity and lost revenue. To avoid this problem, the dynamic operations environment is a fluid environment that enables applications to be available continuously through these processes:

- Application virtualization
- Virtualization of WebSphere resources
- Provisioning of WebSphere applications
- Prioritization and scheduling of applications
- Integration with overall dynamic operations environment infrastructure management

The dynamic operations environment consists of autonomic managers whose purpose is to maximize utilization by using defined business goals. You can monitor performance metrics, analyze the monitored data, offer a plan for running actions, and run these actions in response to the flow of work. Dynamic operations allow an application environment to scale as needed by virtualizing WebSphere resources and by using a goals-directed infrastructure. Thus, you can increase the speed at which your company can adapt to business demands.

In traditional WebSphere Application Server environments, applications are deployed directly to servers or a static cluster of servers that are running on specific hardware systems (nodes). When the server starts, the application starts. A peak load on one system cannot take advantage of resources that are sitting idle on another system.

With Intelligent Management, applications are mapped to dynamic clusters that are spread throughout hardware pools. Each node in the dynamic cluster can run on one or more instances of an application server. The server can be started to accommodate the demand for that application, which is called *dynamic application placement*.

5.3.1 Key components of dynamic operations

This section provides information about the key components of dynamic operations.

Operational policies

An *operational policy* is a business or performance objective that supports specific goals for specific requests. Operational policies include service and health policies. Service policies are addressed in the next section. For more information about health policies, see 5.5, “Health management” on page 122.

Service policies

Imagine an environment with several applications where all client requests are given the same priority. This configuration makes it difficult to manage the system and provide the resources where they are most needed. One solution is to install critical applications in a separate system to enhance their performance. However, this configuration might not make the most efficient use of resources. A better solution is to use Intelligent Management capabilities to define service policies, and to categorize and prioritize work requests.

You can use *service policies* to designate performance goals and the business importance of applications. With service policies, you can classify, prioritize, and intelligently route workload. You can also adjust resources if needed to consistently achieve service policies. Service policies are a technical implementation of service level agreements (SLAs) in place between the business area and the IT area that is running their applications.

Service policy definitions include the following key items:

- The *importance* portion is used in times of resource contention to identify the most important work in the system and to give it higher priority. The options for importance vary from lowest to highest. Administrators who know the relative importance of applications can create realistic performance goals.
- The *goal* portion of the service policy defines how incoming work is evaluated and managed. It detects whether the work is meeting its assigned service policy levels. Service policies can have the following goals:
  - Discretionary
  - Average response time
  - Response time percentile

Specifying the goal portion of the service policy is optional. If you do not specify any goals, only the importance portion is used.

Node groups

A *node group* is a set of systems (nodes) that can host one or more applications. There can be more than one node group within an Intelligent Management cell. An application is placed into a node group, and is optimized based on service policies. Before defining node groups,

you need to know the systems that you want to include in the environment. That is, are all systems identical in terms of resources? Does an application need to be deployed on a specific set of systems because of its prerequisites?

Dynamic clusters

To take advantage of dynamic operations, use a *dynamic cluster*. A dynamic cluster is a server cluster that uses weights and workload management to balance the workloads of its cluster members dynamically. It balances based on performance information that is collected from the cluster members. Dynamic clusters expand to respond to workload demand and user-defined service goals and policies. Dynamic clusters consist of a number of servers that can stop or start in response to changing workload.

When you define a dynamic cluster, you define nodes that host application servers within that dynamic cluster. The member nodes can be designated, or be defined by rules. The latter is only possible with application servers with full lifecycle support. When membership is rules-based, any new nodes added to the cell that meet the rule criteria are automatically added to the dynamic cluster. Application servers are defined automatically on the membership nodes according to properties set in the dynamic cluster.

Dynamic clusters are similar to the server clusters that you can create with WebSphere Application Server Network Deployment, but key differences exist that make dynamic clusters much more robust. For complete lifecycle management servers, the product controls the creation and deletion of server instances, and can start and stop servers. For assisted lifecycle management servers, the product can control the state of servers by stopping and starting servers from a pool of predefined server instances.

Tip: You can use dynamic cluster isolation to isolate applications from other applications that are deployed in the cell. For example, you might create a dynamic cluster isolation configuration to isolate the critical applications that an external customer uses from internal applications that can tolerate some instability.

The on-demand router

The *on-demand router* is an intelligent Java-based HTTP proxy server and Session Initiation Protocol (SIP) proxy server built on the WebSphere run time. The on-demand router is a component that sits in front of your application servers. It is responsible for managing the flow of requests into the WebSphere environment and non-WebSphere environment. The on-demand router is asynchronous, high performance, and scalable. It can be clustered for high availability.

The on-demand router handles the queuing and dispatching of requests according to operational policy. An on-demand router can be defined and started before any service policies are defined. Operational policies can be defined before the appearance of the work to which they apply. However, if policies are not defined, the early work is handled by the default policies.

The on-demand router, similar to the web server plug-in for WebSphere Application Server, uses session affinity to route work requests. After a session is established on a server, later work requests for the same session go to the original server. This configuration maximizes cache usage and reduces queries to resources.

The on-demand router accepts incoming requests and distributes these requests to the system in an intelligent manner, reflecting configured business goals. This process is dependent on the characterization of requests so that the relative business importance of each request can be compared.

Figure 5-5 illustrates how the on-demand router dynamically distributes traffic between application servers in two different dynamic clusters. An equal amount of work can flow into the on-demand router. However, after the work is categorized, prioritized, and queued, a larger volume of work can be given a higher priority to be processed. A smaller volume of less important work might be sent to application servers or even wait in the queue until the application servers are able to serve the requests.

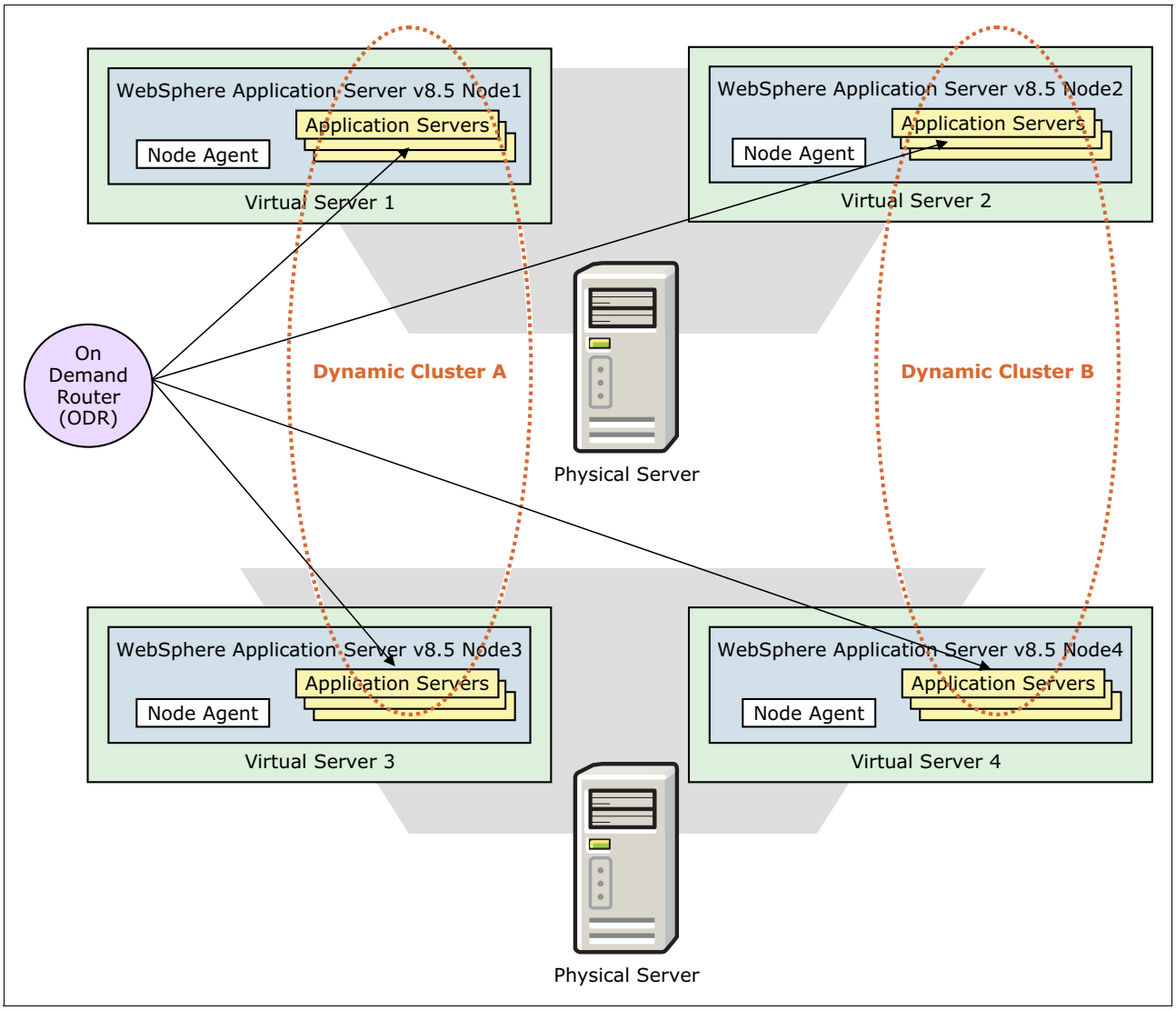


Figure 5-5 On-demand router routing concepts

The on-demand router can be used to set up a highly available deployment manager. It can then route to an active deployment manager and possibly a hot-standby deployment manager.

The high availability deployment manager function provides a hot-standby model for availability. With this support, you can define two or more deployment managers and start them in the same cell. One deployment manager is active, called the *primary* deployment manager. This deployment manager hosts the administrative function of the cell. The other deployment manager or managers are backup managers in standby mode. When in standby mode, you cannot use the deployment manager to perform administrative functions. If the

active manager is stopped or fails, a standby manager takes over and is designated the new active deployment manager.

The benefit of the highly available deployment manager function is that it eliminates the deployment manager as a single point of failure (SPOF) for cell administration. This SPOF is important in environments that have significant reliance on automated operations, including application deployment and server monitoring.

For more information, see 10.10, “Highly available deployment manager” on page 334.

5.3.2 Autonomic managers

With Intelligent Management, you can introduce autonomic capabilities into your infrastructure at your own pace. Autonomic capabilities are delivered in a set of components known as *autonomic managers*. Autonomic managers monitor performance and health statistics through a series of sensors, and optimize system performance and run traffic shaping.

The Intelligent Management includes the following autonomic managers as part of the dynamic operation functionality:

- Autonomic request flow manager
- Dynamic workload controller
- The application placement controller
- The on-demand configuration manager

Autonomic request flow manager

Traffic shaping is managed by the *autonomic request flow manager* (ARFM). The ARFM classifies incoming requests and monitors the performance of service classes on a continual basis. It contains the following components that prioritize incoming requests:

- A *controller* per target cell, which is the cell to which an ARFM gateway directly sends work. This controller is a process that runs in any node agent, on-demand router, or deployment manager.
- A *gateway* per combination of protocol family, proxy process, and deployment target. A gateway runs in its proxy process. For HTTP and SIP, the proxy processes are the on-demand routers. For Java Message Service (JMS) and Internet Inter-ORB Protocol (IIOP), the proxy processes are the WebSphere Application Server application servers.
- A *work factor estimator* per target cell, which is a high availability (HA) managed process that can run in any node agent, on-demand router, or deployment manager.

ARFM controls the order of requests into the application server tier and the rate of request flows. Using classification and the defined service goals, the ARFM decides how and when to dispatch HTTP requests to the next tier. The ARFM also decides when IIOP and JMS requests are run at the application server tier, even though these requests are not routed through the on-demand router. For IIOP requests, only stand-alone Enterprise JavaBeans (EJB) clients are supported. JMS support is only for message-driven beans.

An on-demand router contains the ARFM. The ARFM prioritizes inbound traffic according to service policy configuration and protects downstream servers from being overloaded. Traffic is managed to achieve the best balanced performance results, considering the configured service policies and the offered load. For an inbound User Datagram Protocol (UDP) or SIP message, the on-demand router can route the message to another on-demand router. That router can then check for and handle UDP retransmissions.

Dynamic workload controller

The dynamic workload controller dynamically adjusts server weights to even out and minimize response times across the cluster. There is one dynamic workload controller per cluster.

The dynamic workload controller maintains a list of active server instances for each dynamic cluster, and assigns each a routing weight according to observed performance trends. Requests are then routed to candidate server instances to balance workloads on the nodes within a dynamic cluster based on a weighted least outstanding requests algorithm.

The application placement controller

The application placement controller is responsible for the management of an application's location within a node group. A single application placement controller exists in the cell and is hosted in the deployment manager or in a node agent process.

The application placement controller starts and stops application server instances to manage HTTP, SIP, JMS, and IIOP traffic. The application placement controller can dynamically address periods of intense workflow that would otherwise require the manual intervention of a system administrator.

The on-demand configuration manager

The on-demand configuration manager maintains cell topology information and keeps the ARFM and other controllers aware of its environment. It tracks updates in cell topology and state, including the following changes:

- Applications installed and removed
- Servers started and stopped
- Nodes added and removed
- Classification updates

The on-demand configuration component allows the on-demand router to sense its environment. The on-demand router dynamically configures the routing rules at run time to allow the on-demand router to accurately route traffic to those application servers.

5.4 Dynamic workload management

Dynamic workload management is a feature of the on-demand router. It applies the same principles as Workload Manager (WLM), such as routing based on a weight system, which establishes a prioritized routing system. The dynamic workload controller autonomically sets the routing weights in WLM. With workload management, you manually set static weights in the administrative console. The system can dynamically modify these weights to stay current with the business goals.

The dynamic workload controller can be disabled. If you intend to use the automatic operating modes for the components of dynamic operations, do not set static WLM weights. Doing so prevents the on-demand function of the product from working properly.

The dynamic workload controller also applies to IIOP traffic if the following conditions apply:

- The client is using the WebSphere Application Server Java Development Kit (JDK) and Object Request Broker (ORB)
- The "prefer local" flag is not set for the application

5.4.1 Request flow prioritization by using service policies

ARFM controls the flow of requests for HTTP and SIP traffic through the on-demand router, and for IIOP and message-driven bean traffic from within an application server. It uses a concurrency -based or a rate-based algorithm that results in a more consistent loading and protecting of application server resources by ARFM.

Remember: A *service policy* is a user-defined categorization that is assigned to potential work as an attribute that is read by the ARFM. You can use a service policy to classify requests based on request attributes. These attributes included the Uniform Resource Identifier (URI), the client name and address, HTTP headers, query parameters, cookies, time of day, and so on. By configuring service policies, you apply varying levels of importance to the actual work. You can use multiple service policies to deliver differentiated services to different categories of requests.

Service policy work classes are used to group requests or messages into a group or class of work. Each request belongs to exactly one work class. Each work class contains zero or more rules that are evaluated for each request associated with the work class. Each rule contains an associated service policy that is used if the rule matches. If no rule is matched, the default service policy associated with the work class is used.

The service policy that is associated with the request or message is then used to govern if and for how long a request is queued. This determination is based on the current demand and resource utilization of the target application servers. After you define service policies and associated different service policies through configuration of work classes, you can categorize and prioritize work.

5.4.2 Enabling dynamic clusters

Dynamic clusters work with autonomic managers, including the application placement controller and the dynamic workload controller, to maximize the use of computing resources. Dynamic clusters are required to achieve the server consolidation benefits that are offered by the Intelligent Management features.

With Intelligent Management functionality, you can define performance goals and bind them to specific subsets of the incoming traffic. The on-demand router and associated autonomic managers support business goals in times of high load. They do so by making workload management decisions about the work that is being sent through the on-demand router. Not all the work in a configuration is equally important. The on-demand router can support this concept by forwarding different flows of requests more or less quickly to achieve the best balanced result and maintain the quality of service.

5.5 Health management

You can use the *health management* feature to monitor the status of application servers. This monitoring allows you to sense and respond to problem areas before an outage occurs. You can manage the health of an environment with a policy-driven approach that enables specific actions to occur when monitored criteria is met. For example, when memory usage exceeds a percentage of the heap size for a specified time, health actions can run to correct the situation.

Health monitoring can help you with both unexpected issues and unanticipated problems in your environment. It can help you bypass problems that would otherwise disrupt operations and affect performance.

Consider the Intelligent Management health management feature if you want the following capabilities:

- Automatically detect and handle application health problems without requiring administrator time and intervention
- Intelligently handle heath issues in a way that maintains continuous availability
- Requires administrator approval before an autonomic action is run
- Treats different applications in different ways, because not all of applications have the same health policies

The health management feature consists of the following components:

- Health policies
- Health controller

5.5.1 Health policies

With health management, you define *health policies*. A health policy works like a service policy, except that the health policy provides a health goal for the environment. Each health policy consists of a condition, one or more actions, and a target set of processes.

Health policies are designed to identify potential problems, and take corrective action when an event occurs. You can define health policies for common or custom server health conditions. These policies can monitor the system at the cell, dynamic cluster, static cluster, or application server or node level.

Intelligent Management comes with predefined health conditions, such as excessive memory usage and excessive request or response times, for use in building a health policy. Health management includes the following standard policies:

- Monitor when the heap utilization goes above a threshold, or a memory leak is detected, or when the percentage of time spent in garbage collections goes above a threshold
- Monitor when a server reaches a certain age or services a certain number of requests
- Monitor the percentage of time-out requests or the average response time

Tip: You can build a *custom health policy* by using a custom expression to define the condition. Custom conditions are built based on metrics that are gathered at the on-demand router or server, Performance Monitoring Infrastructure (PMI) metrics, MBean operations, and attributes. A few examples include hung thread detection, and database connection pool exhaustion or slow down.

When a health policy violation is detected, an action plan can be put into effect automatically. Actions to be taken when a monitored condition is detected are designed to bypass the problem and help in diagnosis. You can select the following predefined actions:

- Notifying an administrator
- Sending a Simple Network Management Protocol (SNMP) trap
- Restarting a server
- Putting a server into maintenance mode
- Generating Java cores or heap memory dumps for use in diagnosing the problem

You can also define a custom action to be taken. Actions can be taken automatically, or you can have them occur in supervised mode. Supervised mode requires an operator to confirm the action.

Heath conditions

Health conditions define the variables that you want to monitor in your environment. Several categories of health conditions are predefined. You can also create a health policy that defines a custom condition when the predefined health conditions do not fit your needs. The following predefined health conditions are available:

- Age-based

  Triggers when members associated with this policy reach a certain age value. You can use the age-based condition on all server types.

- Excessive request timeout

  Triggers when requests that are directed to an associated member timeout, and the percentage of timeouts exceed the specified value. You can use the excessive request timeout condition on all server types.

- Excessive response time

  Triggers when the members that are associated with this detection-based policy have an average response time for requests that exceed a certain amount of time. You can use the excessive response time condition on all server types.

- Excessive memory usage

  Triggers when the members associated with this detection-based policy use more memory than a percentage of the maximum heap size for a certain amount of time.

- Excessive garbage collection

  Triggers when the Java virtual machine (JVM) spends more than a configured percentage of time when running garbage collections.

- Memory leak

  Looks for consistent downward trends in free memory that are available to a server in the Java heap. The detection level setting determines when these trends are detected. The slower detection level setting requires the most historical data.

  The normal and faster detection level settings require the same amount of historical data. However, the faster setting allows analysis before the Java heap expands to its maximum configured size. This setting provides earlier detection capability, but it is also more prone to false positives. This condition supports heap memory dumps in addition to server restarts as reactions.

- Storm drain

  Detects situations where requests are shifted toward a faulty cluster member that advertises low response times. This condition is triggered when there is a significant drop in the average response time. This drop must be measured at the on-demand router, for a member of the cluster coupled with an increase in the dynamic weights for the cluster member.

- Workload

  Triggers when the members that are associated with this policy serve a user-defined number of requests. You can use the workload condition on all server types.

Heath actions

Depending on the configuration of the health policy, different *health actions* are run if a policy breach is detected:

- Restarting the application server

  When a server is a member of a dynamic cluster, another instance of the dynamic cluster is started. This instance serves user requests before the server that triggered the policy breach is shut down. This process allows WebSphere to handle potential issues with the least amount of impact to its consumers.

- Taking a thread memory dump (JavaCore)

  Three JavaCores are generated for this action. The option to take thread memory dumps is only supported for application servers that run in IBM JVMs.

- Putting a server into maintenance mode

  Maintenance mode is used to run diagnostic actions, maintenance, or tuning on a node or server without disrupting incoming traffic. Putting a server into maintenance mode allows the remaining requests on the server to be processed. Any requests that have an open session on the server are routed to the server until the session ends or times out. After all requests are completed, the server is moved to maintenance mode. Any new requests are routed to servers that are not in maintenance mode.

- Putting a server into maintenance mode and breaking HTTP and SIP affinity

  The same process as the previous action occurs, but the HTTP and SIP session affinity to the server is broken.

- Taking a server out of maintenance mode

  After the server reaches a healthy state, it can be reinstated to serve requests. For example, if a server exceeds a memory threshold, putting it in maintenance mode gives it a chance to recover. It can free up memory through garbage collection while no new requests are being sent to it. After heap utilization is back below the threshold, the server can be taken out of maintenance mode.

- Creating a custom action

  With a custom action, you define an executable file or Java code to run when the health condition occurs. A custom action must be created before you can use it in a health policy.

Remember: All actions are available for all health policies.

Reaction mode

The health management feature functions in a *reaction mode* that defines the level of user-interaction when the health condition determines corrective action is needed:

- Automatic mode

  When the reaction mode on the policy is set to automatic, the health management system takes action when a health policy violation is detected. The data is logged, and the defined reaction is run automatically.

- Supervised mode

  The health management system creates a runtime task that proposes one or more reactions. The recommendations on actions are sent to the administrator who can then approve or deny them. If the administrator follows the recommendations, the only action that is required is clicking a button to run the actions.

5.5.2 Health controller

The *health controller* is an autonomic manager that constantly monitors the defined health policies. When a condition specified by a health policy is not met in the environment, the health controller assures that the configured actions are taken to correct the problem.

To use health monitoring, you must make sure that the health controller is enabled. After you configure and enable the health controller, it runs as part of the cell. There is one controller per cell. The health controller is a highly available controller which runs in the deployment manager or a node agent process. If the active process fails, the health controller can become active on another node agent or deployment manager process. You can use the runtime topology in the administrative console to learn which process hosts the health controller.

You can disable or enable health management by using the health controller. If the health controller is disabled, no health policy monitoring occurs. You can also apply limits to the frequency that the server restarts or prohibit restarts during certain periods.

5.5.3 Planning for health monitoring

Health management is not meant to replace the testing and benchmarking phases of the application development lifecycle. However, if the system has had stability problems or you are unsure about the stability of an application, consider applying policies to the application. These policies are especially useful in the following health conditions:

- Excessive request timeout
- Excessive response time
- Excessive memory usage
- Excessive garbage collection

In addition, give particular consideration to the custom PMI health conditions and actions listed in Table 5-1.

Table 5-1 Suggested actions for PMI custom health conditions

PMI module	PMI metric	Sample expression	Suggested actions
Thread pool module	Concurrently hung threads	`PMIMetric_FromLastInterval$threadPoolModule$concurrentlyHungThreads > 3L`	Take thread dump files and then restart server
Process module	Process total memory (KB)	`PMIMetric_FromLastInterval$xdProcessModule$processTotalMemory > 2048L`	Restart server
Connection pool module	Average wait time (milliseconds)	`PMIMetric_FromLastInterval$connectionPoolModule$avgWaitTime > 5000L`	Start custom action or notify administrator of database issues

5.6 Application edition management

Companies commonly have a build and deployment process that is used while an application moves from development to production. A source control system is normally used to store the application source code and related artifacts. These library systems are typically designed to store multiple versions of these parts. The concept of an application version is established in the context of software libraries and build processes.

With the Intelligent Management functionality, you can store these application versions in the *system management repository* and deploy them as needed. While source control systems typically store the source code, the system management repository stores the compiled code. You need both.

Using application edition management, you can validate a new edition of an application in your production environment. This process does not affect users, so you can upgrade your applications without causing outages for your users. You can also run multiple editions of a single application concurrently, directing different users to different editions.

The application edition management feature also provides an application versioning model that supports multiple deployments of the same application in a cell. You can choose which edition to activate on a cluster, so you can roll out an application update or revert to a previous level.

Consider the Intelligent Management application edition management feature if you want the following capabilities:

- Incur no downtime when updating applications or the environment.
- Run multiple versions of applications concurrently.
- Verify that a new version of an application runs in production before directing user traffic to the application.
- Reduce infrastructure costs and decrease outages in the production environment.
- Update an operating system or WebSphere environment easily without incurring downtime to the environment.
- You can use the application edition manager feature if you are using WebSphere Batch and want to perform a rollout to batch applications.

5.6.1 Key features

The application edition manager provides an application versioning model that supports multiple deployments of the same application in an Intelligent Management cell. The application edition manager interacts with the on-demand router, dynamic workload manager, and application placement manager. This integration ensures predictable application behavior when you apply application updates. You get a smooth transition from one application edition to another while the system continues to manage your application performance goals.

The application edition manager's edition control center in the administrative console provides control over the application update and rollout process. This process includes edition activation across the application servers to which your application is deployed. Scripting APIs enable the integration of edition management functions with automated application deployment.

The application edition manager provides support for interruption-free application upgrades only for applications accessed through the on-demand router by way of HTTP or HTTPS.

Service continuity during application upgrade is not assured for inter-application access unless the inter-application access is accomplished by way of HTTP or HTTPS through another on-demand router layer.

5.6.2 Terminology

The application edition manager feature includes the terminology described in the following sections.

Application editions

An *application edition* represents a unique instance of an application in the environment. An application edition encompasses both application versions and deployment bindings. An application edition is an application that is uniquely identified by the combination of an application name and an edition name.

Edition names and descriptions

With application edition manager, you can install multiple editions of the same application. Each edition is identified with an application *edition name* and *description*. The edition name is a field in which you can specify a value to uniquely identify one application edition from other editions of the same application. Create a version number scheme for naming editions that is meaningful in your environment. Multiple editions of the same application have the same application name but different edition names.

When deploying an application, you can also specify an edition description next to the edition name, which gives you the ability to store additional information.

Non-destructive update

The existing application installation and update functions in Network Deployment are destructive. That is, they replace the old instance of the application with a new instance. Installing an application edition is *non-destructive*. You can install any number of application editions and keep them in the system management repository.

State

Each application edition deployed has a *state* that identifies the status of the application edition. Each application edition must be in one of the following states or modes:

- Active
- Inactive
- Validation

Because the application edition transitions from one state to another, various actions occur, such as installing, validating, activating, running a rollout, deactivating, and uninstalling. After installing a new edition of an application, the new edition is only activated if there is not already an active edition deployed to the same cluster. For each application and deployment target combination, there can be at most one edition in active mode and one edition in validation mode. An edition that is in the inactive state is not started when an application server starts.

5.6.3 Concepts

The application edition management feature provides the following capabilities:

- *Rollout* indicates policies that allow you to switch from one edition to another edition with no loss of service.
- *Concurrent activation* where multiple editions can be concurrently active for an extended period.
- A *validation mode* to send selective traffic to verify the correct operation.

Rollout

Rollout activation activates one edition in place of another, ensuring an interruption-free update in the process. Thus, all application requests are serviced during the rollout and none are lost. This process ensures continuous application operation from the perspective of the customers of that application. To do this, the application edition manager carefully coordinates the activation of the edition and the routing of requests to the application.

During rollout, you make the following choices:

- Soft or hard rollout

  A soft rollout stops and starts only the application, whereas a hard rollout stops and starts the application server. You might consider a hard rollout if an application must reload native code.

- Atomic or group rollout

  An atomic rollout guarantees that two editions do not service requests at the same time, whereas a group rollout does not make this guarantee. The atomic rollout can queue requests briefly in the on-demand router to guarantee atomicity. A group rollout does not queue requests.

- Drainage interval

  The drainage interval is the *maximum* amount of time that the application edition manager waits for sessions to expire before stopping an application server. During this interval, no new sessions are established on the application server, but requests with affinity continue to be routed to the application server. If all sessions expire before the completion of the drainage interval, the application server is stopped and the rollout continues. Therefore, this interval is the maximum time to wait, but the actual time might be much shorter, depending on the active session count.

Replacement of one edition with another in a production environment requires certain discipline in the evolution of the application. Because edition replacement happens while application users are potentially accessing the previous application edition, the new edition needs to be compatible with earlier versions. Thus, the new edition cannot add or change any existing application interfaces, including essential behavior. New interfaces can be added. In addition, existing interfaces can be algorithmically corrected and, in some cases, even extended and remain compatible with existing application users.

Figure 5-6 shows an example of a group rollout scenario. In the diagram, a dynamic cluster is created that consists of three servers. You first need to divide the cluster into groups, which tells the application edition manager how many servers to update at the same time. Performing a rollout to a group results in the servers in each group being upgraded to the new edition at the same time. Each server in the group is quiesced, stopped, and reset.

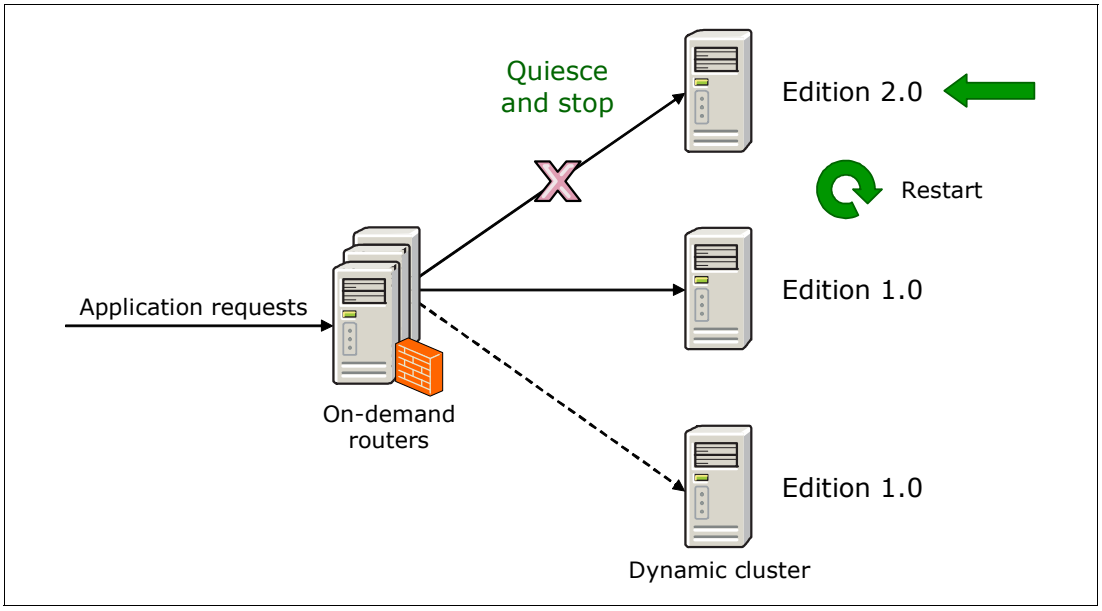


Figure 5-6 Rollout policies

As the rollout is run in Figure 5-6, one server in the cluster is moved from Edition 1.0 to Edition 2.0. During this time, the server does not receive user requests that are directed from the on-demand router, and the server is stopped. All application requests are sent to the servers that are running Edition 1.0. After the server that is running Edition 2.0 is available, application requests are directed by the on-demand router to that server. Any servers that are still running Edition 1.0 do not serve requests until the edition is updated to Edition 2.0.

Concurrent activation

Concurrent activation enables you to activate the same edition on different servers or clusters. To use multiple editions concurrently, you must distinguish user requests from one another so that the requests are sent to the application server that hosts the appropriate edition. For example, if you introduce a new edition of an application, you might want only a select group of users to test the edition.

When multiple editions of the same application are concurrently available to users, the on-demand router needs information to differentiate between the active editions. Based on that information, it then intelligently routes the request to the intended edition. You must configure a routing policy that tells the on-demand router to which edition to route a request. The routing policy is stored as part of the application metadata.

Figure 5-7 shows an example of concurrently active editions. There are two clusters that are hosting different application editions. The on-demand router uses the routing policies to determine where to deliver user requests when multiple editions of an application are activated.

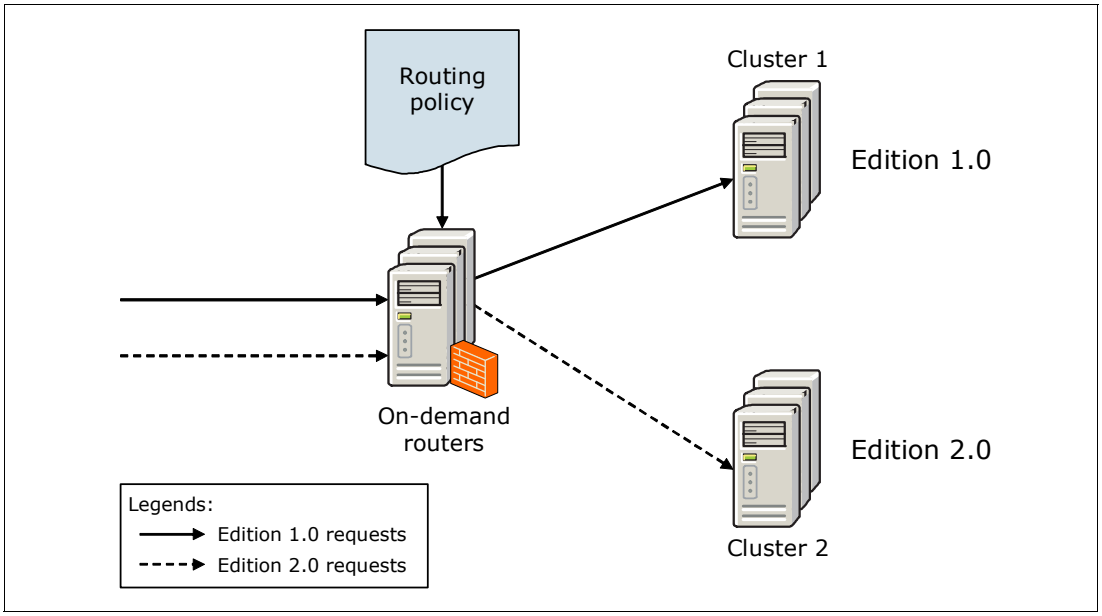


Figure 5-7 Concurrent activation

Validation mode

Validation activation is a special form of concurrent activation. It activates an edition on a clone of its original deployment target. The clone is created on activation of the edition. After the validation rollout to the original deployment target, the clone is removed automatically. This action allows you to perform final pre-production testing of an application edition in the actual production environment with a selected set of users.

To perform a validation mode scenario, the actual deployment target is cloned. The target edition is then activated on the cloned environment. Routing policies are used to tell the on-demand router how to divert selected user requests to the new edition.

Figure 5-8 shows an example of the validation mode.

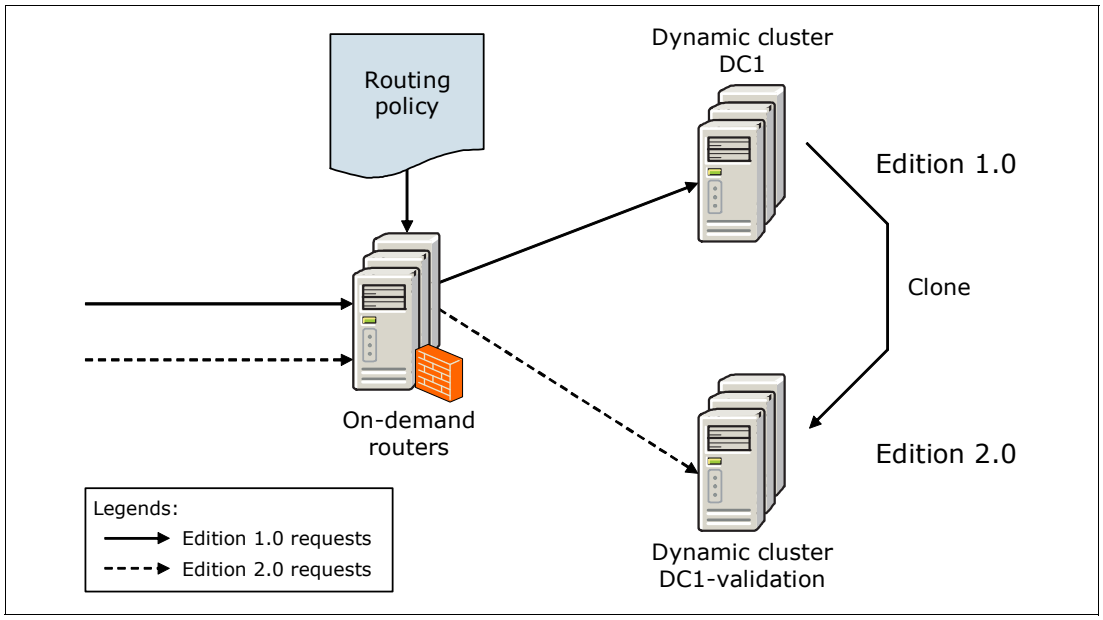


Figure 5-8 Validation mode

5.6.4 Maintenance modes

Periodic product maintenance is important to keep your system environment working correctly, and to avoid trouble caused by known issues. At some point in time, you might have a problem with a server and need to perform diagnostic tests to troubleshoot a specific application server. These situations can lead to the disruption of client requests to servers in your environment.

Using the Intelligent Management feature, you can maintain the environment without disrupting traffic to the production environment. You can use it to administratively put a server or node in the cell into *maintenance mode*. In a normal mode, the on-demand router sends requests to application servers. Using maintenance mode, you can stop routing from the on-demand router to the nodes or servers that are placed into maintenance mode. This action maintains these nodes or servers with minimum disruption to your environment. The Application Placement Controller also excludes the node or server from automatic application placement. Maintenance mode is only recognized by the on-demand router. However, the heath controller also uses the server maintenance mode as an action that is taken when a health policy is breached.

Node maintenance mode

You can put a node into maintenance mode when you need to apply operating system fixes or perform WebSphere maintenance. When a node is in maintenance mode, only traffic with affinity to servers on the node is routed to the server by the on-demand router. A maintenance immediate stop mode can be set that immediately stops the servers on the node.

Server maintenance mode

You can put a server into maintenance mode when you need to perform server level problem determination. When an application server is placed into maintenance mode, you can indicate one of these modes:

- Allow all traffic to the server
- Allow only traffic with affinity
- Allow no traffic during the maintenance period

There is also the maintenance immediate stop mode that immediately stops the application server. Each of the maintenance modes for nodes and servers can be enabled by using the administrative console or through **wsadmin** scripting.

5.7 Performance management

The performance management feature provides dynamic cluster capabilities and overload control. With dynamic clusters, you can automatically scale up and down the number of running cluster members as needed to meet response time goals for your users. You can use overload protection to limit the rate at which the on-demand router forwards traffic to application servers. Doing so helps prevent heap exhaustion, processor exhaustion, or both from occurring.

Consider the Intelligent Management performance management feature if you want the following capabilities:

- Associate service policies with your applications and have WebSphere efficiently manage these goals
- Decrease administrative effort required to monitor and diagnose performance issues
- Minimize the number of JVMs and virtual machines that run to reduce processor usage incurred by idle or lightly used JVMs or virtual machines
- Protect your middleware infrastructure against overload

5.7.1 Workload management with dynamic clusters

A dynamic cluster is a virtual cluster of application servers that hosts an application. These application servers are on groups of nodes that are indicated by using the node group function. The membership policy is compared against the nodes in your cell and servers are created for the dynamic cluster by using nodes that match the policy. When new nodes are added to your environment, they are added automatically to the dynamic cluster if they match the defined membership policy.

When configuring a dynamic cluster, you can use the following settings:

- *Minimum number of cluster instances* where you can select to have one or more servers started at all times. You can also stop all servers in times of inactivity.
- *Maximum number of cluster instances* where you can limit the number of servers that can start.
- *Vertical stacking of instances on a node* where you can indicate whether you want to allow more than one server instance to be started on the same node.
- *Isolation requirements* where you can indicate whether a cluster member can run on the same node as cluster members from a different dynamic cluster.

Lazy application start

The *lazy application start* feature optimizes server resource allocation during times of inactivity. The smallest size for a dynamic cluster is zero, implying that an application can be configured for execution but not running in any application server instances. When a request for that application is received by the on-demand router, an application server for that application is automatically started on any node in the dynamic cluster. A custom error page can be returned with a meta refresh tag to provide feedback to the user while waiting for the application to start.

A typical environment where the lazy application start feature is beneficial has these characteristics:

- The ratio of the number of dynamic clusters to the number of servers is high.
- Some dynamic clusters are not accessed for long periods of time.

In this environment, hibernating idle dynamic clusters temporarily (stopping all server instances) releases valuable resources to be used by active dynamic clusters.

Vertical stacking

Using vertical stacking, you can have more than one application server instance in a dynamic cluster on the same node. The benefit of this capability is better hardware utilization if a processor and memory are not used fully with a single application server on a node. Use vertical stacking only when a single application server instance cannot consume the full processor resources of a node.

5.7.2 Overload protection monitor

Overload protection is a feature that monitors the memory and processor usage of a server. It then regulates the rate at which traffic is sent to an application server to prevent memory and processor overload. Memory overload protection is disabled by default. To enable it requires the configuration of the autonomic request flow manager.

For a dynamic cluster, you can indicate a maximum heap utilization percentage that protects against out-of-memory errors. For processor overload protection, you can indicate a maximum processor percentage that protects against various failures that might occur when a processor is consumed. A rejection policy can be set that prevents a processor from being overloaded. The policy works by rejecting incoming HTTP or SIP messages that are not part of existing sessions for HTTP or SIP traffic.

5.8 Planning for hosting dynamic operations

Planning a production environment for dynamic operations is different from planning a static environment. In a static environment, you use dedicated servers for each application. To size the servers, look at your applications, the requirements, and the expected load during peak time. Your production environment must be prepared for the load during this possibly short period, meaning that during non-peak hours servers can be underused.

In general, most companies have more than one critical application, and the second application can have its peak load at a different time of the day. In a static environment, the servers that host the first application cannot be used for the peak load of the second application. Therefore, the quality of service suffers or you must purchase more or larger systems to ensure the quality of service.

Use the following questions to gather the information that is required to set up an environment for dynamic operations:

- What applications do you want to include? This list of applications affects the number of dynamic clusters.
- What are your critical applications? The critical applications affect the allocation of applications to dynamic clusters and the assignment of service policies.
- Are there applications that should not run on the same system? If so, these applications must be in different isolation groups.
- Which applications have the same load requirement and can share server configuration? These applications can be in the same dynamic cluster.
- Which servers or hardware do you want to include? In a heterogeneous environment, you might consider using multiple node groups, depending on the hardware type.
- Do all servers have the same resources available, such as network drivers, database connections, external drives, and additional software? Because every application can be started on every node in a node group, you need to have all resources available on each node. Take this list of available resources into consideration because it might increase license costs and require additional hardware resources on each system.

Based on this information, plan your node groups, dynamic clusters, and service policies.

5.8.1 Topology considerations for the on-demand router

When planning an Intelligent Management configuration, many of the planning considerations are focused on the on-demand router. It has the following primary functions:

- Request routing
- Intelligent routing based on a sense and response mechanism from back-end servers
- Classification of incoming requests based on rules defined by the business owner

Thus, it is important to ensure that the on-demand router is scalable and highly available.

Important: The on-demand router introduces an additional and critical processing layer to the server network topology. Because it is central to the functioning of the Intelligent Management environment, the on-demand router tier must not cause a processor bottleneck. Any performance issues with the on-demand router can affect the entire WebSphere Application Server environment. Therefore, the administrators and architects who plan the topology must ensure the high availability of on-demand routers. Factors that affect on-demand router performance include the number of supported clients, message size, secure sockets layer (SSL) implementation, and type of hardware.

The decision as to the number of on-demand routers to place in an environment depends on the enterprise and infrastructure. Generally, you need at least two on-demand routers to provide high availability. An on-demand router needs to balance workload between the servers within same cell and core group as the on-demand router.

Various factors come into play when determining whether to use additional on-demand routers. Consider the number of clients served, the number of applications, the types and size of sessions, and security factors. As the number of clients increases, more processor usage is required to tracking all the clients. Therefore, have a close estimation of clients that access the environment, and evaluate the performance levels of the current set of on-demand routers. Consider adding additional on-demand routers if the client base will be increasing due to a business activity such as a promotional offer.

You can create a dynamic cluster of on-demand routers. A cluster allows the application placement controller to select the best node on which to start the minimum number of on-demand routers. If an on-demand router stops for any reason, the application placement controller starts a new instance.

Remember: A dynamic cluster of on-demand routers in WebSphere Application Server V8.5 allows you to scale higher than the minimum number of clusters when needed. The prerequisite is that you must set the following cell custom property:

```
Name: APC.predictor
Value: CPU
```

This setting causes the application placement controller to operate based on processor usage alone rather than input that it receives from the on-demand router.

For information about sizing the number of on-demand routers, see:

https://www.ibm.com/developerworks/wikis/display/xdoo/Best+practices+for+managing+the+on+demand+router?showComments=false>

5.8.2 Monitoring dynamic operations

The Intelligent Management function allows you to monitor runtime operations from the administrative console. Real-time reporting shows alerts that indicate anomalies with the runtime environment. You can view the status of the cell based in on-demand routers, core groups, core components (autonomic managers), and nodes. Through this visualization feature, you can log historical performance metrics.

The enhanced charting in the administrative console provides advanced charting and graphics, and customizable reports. You can build and save customized reports for dynamic viewing. To build these reports, select the type of component you want to monitor. You can select on-demand routers, application servers, nodes, dynamic clusters, and service policies. Next, select a specific instance of the component type to monitor. Then select the data metric to use from a wide selection, including metrics such as average response time, throughput, and processor utilization. These reports can help you ensure that you are meeting service level goals and can help you identify potential problems in the early stages.

You can also use IBM Tivoli Composite Application Manager for WebSphere to retrieve and view application-specific metric information from a WebSphere environment. For more information, see 3.7, “IBM Tivoli Composite Application Manager for WebSphere” on page 84.

6

WebSphere Batch

This chapter addresses the concepts of WebSphere Batch introduced with IBM WebSphere Application Server V8.5.

WebSphere Batch is a mature component that delivers batch processing capabilities and provides a comprehensive execution environment for Java batch processing and unified batch architecture. It answers the need to provide efficient batch processing that can run in parallel by running Java batch inside WebSphere Application Server.

This chapter includes the following sections:

- Overview of WebSphere Batch
- WebSphere Batch programming models
- WebSphere Batch components
- Batch workflow
- New features in WebSphere Application Server V8.5 for WebSphere Batch

6.1 Overview of WebSphere Batch

Batch processing is a mission critical workload for the enterprise that can increase workload efficiency. The following are examples of tasks that are carried out with batch processing:

- Reporting on accounts for end of day, month, or year
- Bulk account processing for credit scores and for assessing interest
- Reconciling banking activities

Many enterprises depend on batch processing.

Online transactional processing (OLTP) systems have evolved over time, and application servers, such as WebSphere Application Server, serve as the foundation for this evolution. Standards for web services and other OLTP technologies have emerged. Programming models such as Java Platform, Enterprise Edition (Java EE) have been standardized. And service-oriented architecture (SOA) has been pursued. Throughout this evolution, however, batch systems are often overlooked.

WebSphere Batch provides a batch technology optimized for Java that ensures enterprises remain agile, scalable, and cost efficient. The WebSphere Batch function was delivered in WebSphere Application Server V7 as the *Modern Batch Feature Pack*. With WebSphere Application Server V8.5, it is now enhanced and fully integrated. The integration of WebSphere Batch provides advanced batch management functions without having to purchase an add-on product.

6.1.1 WebSphere Batch key features

WebSphere Application Server V8.5 adds efficiency and operational features through WebSphere Batch. WebSphere Batch includes the following key features:

- A *unified batch architecture* that provides a consistent programming model and consistent operational model for multiple platforms
- A *comprehensive batch solution* that allows for end-to-end development tools and execution infrastructure and enterprise integration to compliment the overall system

The integration of WebSphere Batch provides implicit components to manage the following Compute Grid and Virtual Enterprise based functions:

- Comprehensive development and management tools for building and deploying batch applications that are based on Java
- A resilient, highly available, secure, and scalable run time with container-managed services for batch applications
- A platform that supports 24x7 batch and OLTP processing and parallel computing on highly virtualized and cloud-based run times
- Integration capability with existing infrastructure processes, such as enterprise schedulers, and archiving and auditing technologies that are typically deployed in an enterprise batch solution
- Integration capability with the overall SOA strategy of reuse by enabling services to be shared across multiple domains, such as batch, OLTP, and real-time

- High-performance batch on the mainframe by integrating with z/OS. This function uses workload management and performance optimizations gained by running close to the application data
- Platform and batch applications that allow the location of the application data to dictate its deployment platform

6.1.2 Main concepts of batch processing

Long-running workloads or applications typically require more resources and different types of support than the standard lightweight, transactional work typical of Java EE applications. Long-running work can take hours or even days to complete, and can consume large amounts of memory or processing power while it runs. WebSphere Application Server V8.5 with WebSphere Batch provides an environment that supports long-running applications. This environment provides the capability to deploy different types of applications to different nodes within a WebSphere cell. It can also balance the work based on policy information.

The submission of a long-running workload, also known as a *job*, is asynchronous from the workload that is run. When long-running work begins, state information needs to be persisted to a highly available data store. Administrators need the ability to monitor and manage long-running work. Also, the environment needs to be able to schedule and prioritize the work based on service policy information that is set by the user.

WebSphere Batch components support the following types of long-running workloads or jobs:

- Batch applications

  A typical *batch application* does large amounts of work based on repetitive tasks. A batch application needs to provide the logic for a single unit of work. The container provides the support to run the job with transactions. It also provides the ability to checkpoint and restart the application as required. For example, a typical batch application processes many records. Each record can represent a unit of work. The application provides the logic to process a single record. The environment in turn manages the process of repeatedly starting the application task for processing each record until processing is complete.

- Compute-intensive applications

  Compute-intensive applications run work that requires large amounts of system resources, in particular processor capacity and memory. In this case, the application provides all the logic for completing the work, including acquiring the resources. The WebSphere Batch environment makes sure that the application is appropriately situated within the environment.

OLTP provides a request/response model where the duration of the processing is relatively short and the tasks are typically transactional in nature. In this model, the application server run time enforces timeouts for the workload. In contrast, batch processing is a submit/work/result set model where the duration of the processing is a function of the tasks to be completed. In some cases with batch processing, the tasks can require hours or even days to complete. In this model, the work tasks are typically transactional in nature and involve multi-step processes.

Figure 6-1 shows the differences between these two processing models.

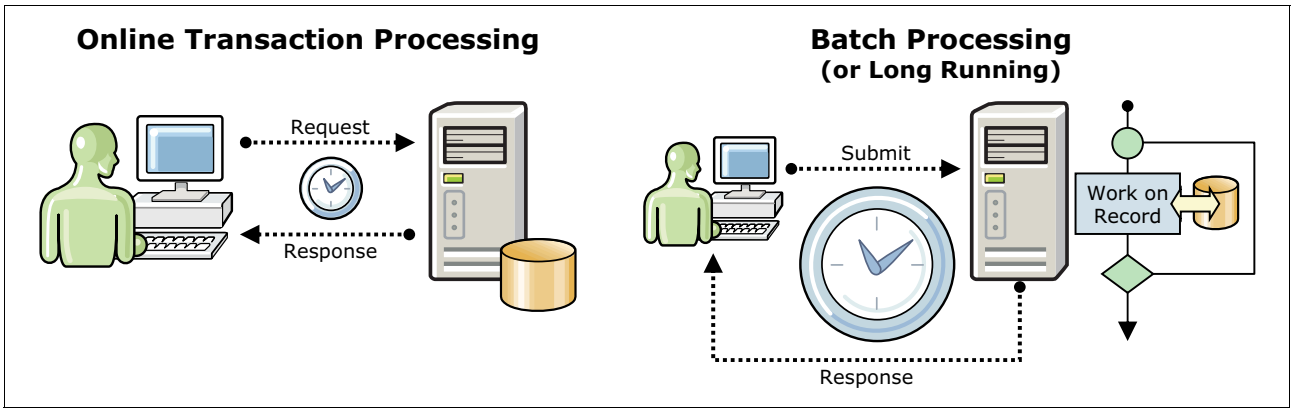


Figure 6-1 Processing models

Figure 6-2 illustrates how batch job submitters are not burdened with the details of the batch platform. Instead, they interact with a job management tier and view the remainder of the platform as a cloud. Job submitters submit batch job definitions to the job management tier. These definitions reference the business logic to be run, the parameters that describe the input and output data locations, and any job-specific qualities of service. Job submitters can also submit operational commands such as **`stop`**, **`start`**, **`cancel`**, or **`restart`** on their job instances.

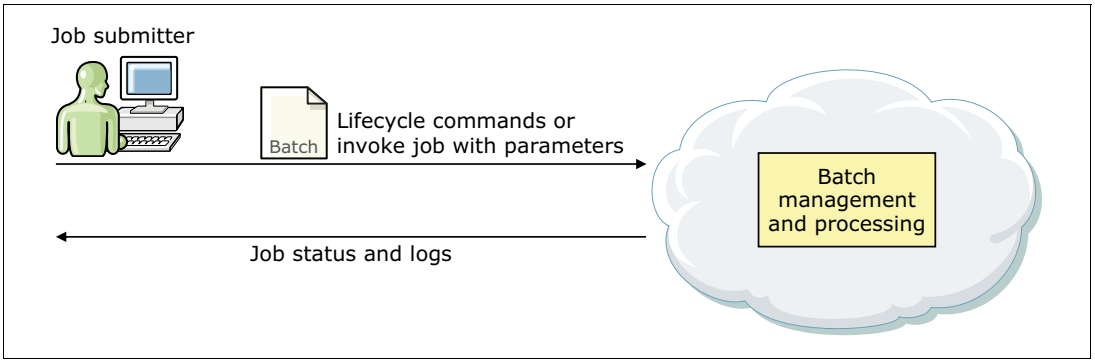


Figure 6-2 Batch management from a job submitter perspective

Batch job definitions can be stored in a *job repository*, which enables you to manage the lifecycle of the job definitions. Starting batch jobs stored in the repository is synonymous to making a remote-procedure call in other distributed computing paradigms. The name of the job and instance-specific parameters are passed to the system for execution. The output of the job execution, which is typically archived for auditing purposes, can be streamed back to the job submitter.

Figure 6-3 shows the process that happens inside this cloud and WebSphere Application Server to manage batch operations.

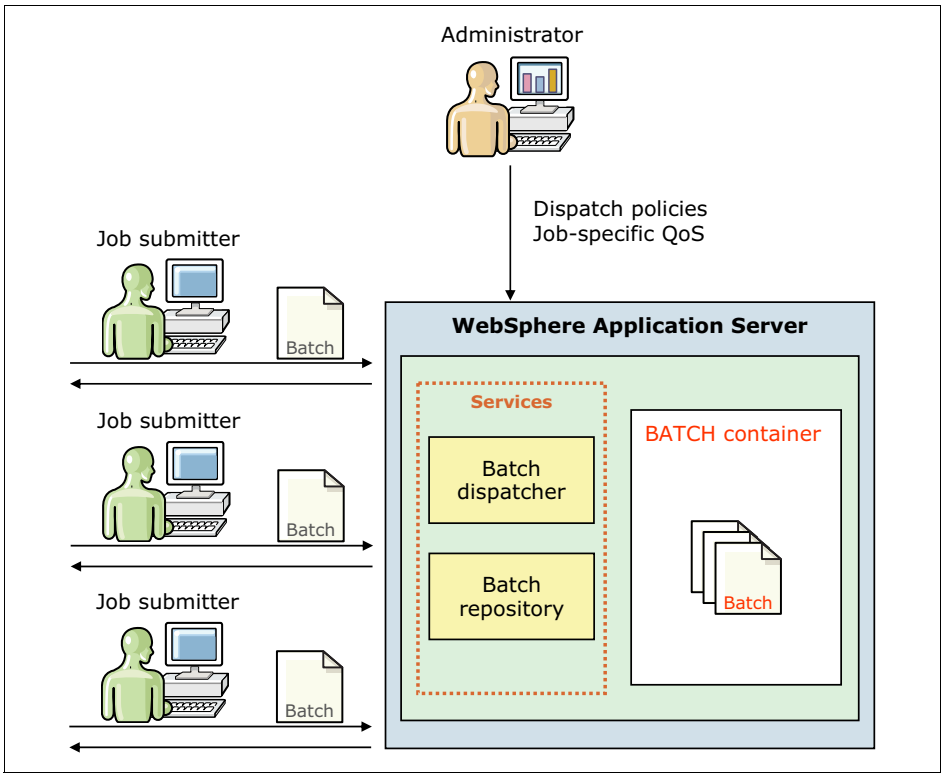


Figure 6-3 Batch infrastructure in WebSphere Application Server V8.5

WebSphere Batch is composed of the following primary components:

- The *job dispatcher tier* manages the execution of batch jobs for a collection of resources.
- The *batch container tier* runs the jobs themselves.

Administrators define policies that influence how batch jobs are run. These policies, coupled with autonomics that are built into the infrastructure, serve as the foundation for the cloud-enabled batch. Administrators can perform the following tasks (as depicted in Figure 6-3):

- Define policies that govern how jobs are run. The lifecycles of these policies can be managed through standard lifecycle management technologies. You can have a lifecycle that is independent of the applications.
- Configure dispatch policies that influence where batch jobs are run. For example, a dispatch policy can be defined where all jobs of a certain type must run in a 64-bit Java virtual machine (JVM).
- Configure partitioning policies that define how batch jobs are broken into parallel processing elements. These policies typically match how the data that is used by the batch applications is partitioned. Dispatch policies can be defined in conjunction with the parallel portioning policies to create a highly parallel solution with data-aware routing.
- Configure job-specific qualities of service (QoS) that influence how jobs are run within the batch container.

Both the batch container tier and the underlying infrastructure cloud convey capacity and execution metrics to the job-dispatching tier. The job-dispatching tier uses the metrics to

determine the best endpoint on which to run the jobs. This system uses autonomic algorithms to ensure that jobs are load-balanced throughout the system.

Consideration: As spikes in batch jobs occur, *elasticity services*, where the batch container and infrastructure cloud must scale up or down to meet demand, might be necessary. The z/OS platform has these services built in, and WebSphere Application Server V8.5 takes advantage of them. On distributed platforms, these services are integrated to create an *elastic infrastructure*.

As the use of cloud computing evolves, WebSphere Application Server and other cloud-enabling technologies can serve as the foundation for batch processing. Mixed application workloads (such as OLTP, batch, message-driven, and other types of workloads) run within highly virtualized infrastructures. Physical resources, such as processor capacity and memory, and logical resources, such as database locks, are manipulated to ensure that service level agreements (SLAs) for workloads are met by the system. Dynamic provisioning coupled with elastic applications ensure that the system can tolerate spikes in application demand.

6.1.3 Application server run time

You can manage Java batch tasks by using the following methods:

- Simple Java virtual machine launchers can manage single-step batch jobs that require basic data access.
- On the z/OS platform, the Java batch toolkit for z/OS enhances the BPXBATCH model. It does so by supporting conditional multi-step batch jobs with access to Multiple Virtual Storage (MVS) data sets and use of data definition (DD) cards.
- WebSphere Application Server provides multi-step job support, a managed container for execution of batch jobs, a job control interface, job checkpoint and restart capability, and a batch application development framework.

Enhancements: WebSphere Application Server V8.5 introduces support for these functions:

- Job repository and schedules
- Workload management
- Job usage reporting
- Batch application quiesce and update
- Parallel job support
- Pacing and throttling of jobs

It also adds integration with external schedulers and support for starting COBOL routines from Java batch applications.

6.2 WebSphere Batch programming models

WebSphere Batch provides a transactional batch programming model and a compute-intensive programming model. Both the transactional batch and compute-intensive programming models are implemented as Java objects. They are packaged in an enterprise archive (EAR) file for deployment into the application server environment. The individual programming models provide details about how the lifecycle of the application and jobs that are submitted to it are managed by WebSphere Application Server V8.5.

Central to all WebSphere Batch applications is the concept of a *job* to represent an individual unit of work to be run. You can mix job steps from transactional batch and compute-intensive applications. The run time uses a controller that is the same for every job, regardless of the type of steps that the job contains. The controller runs appropriate logic for the step, whether the step is for a batch or compute-intensive application. These different job step types can also be run in parallel.

6.2.1 Transactional batch programming model

Batch applications are Enterprise JavaBeans (EJB) based Java EE applications. These applications conform to a few well-defined interfaces that allow the batch runtime environment to manage the start of batch jobs destined for the application.

Figure 6-4 illustrates the batch programming model principal interfaces.

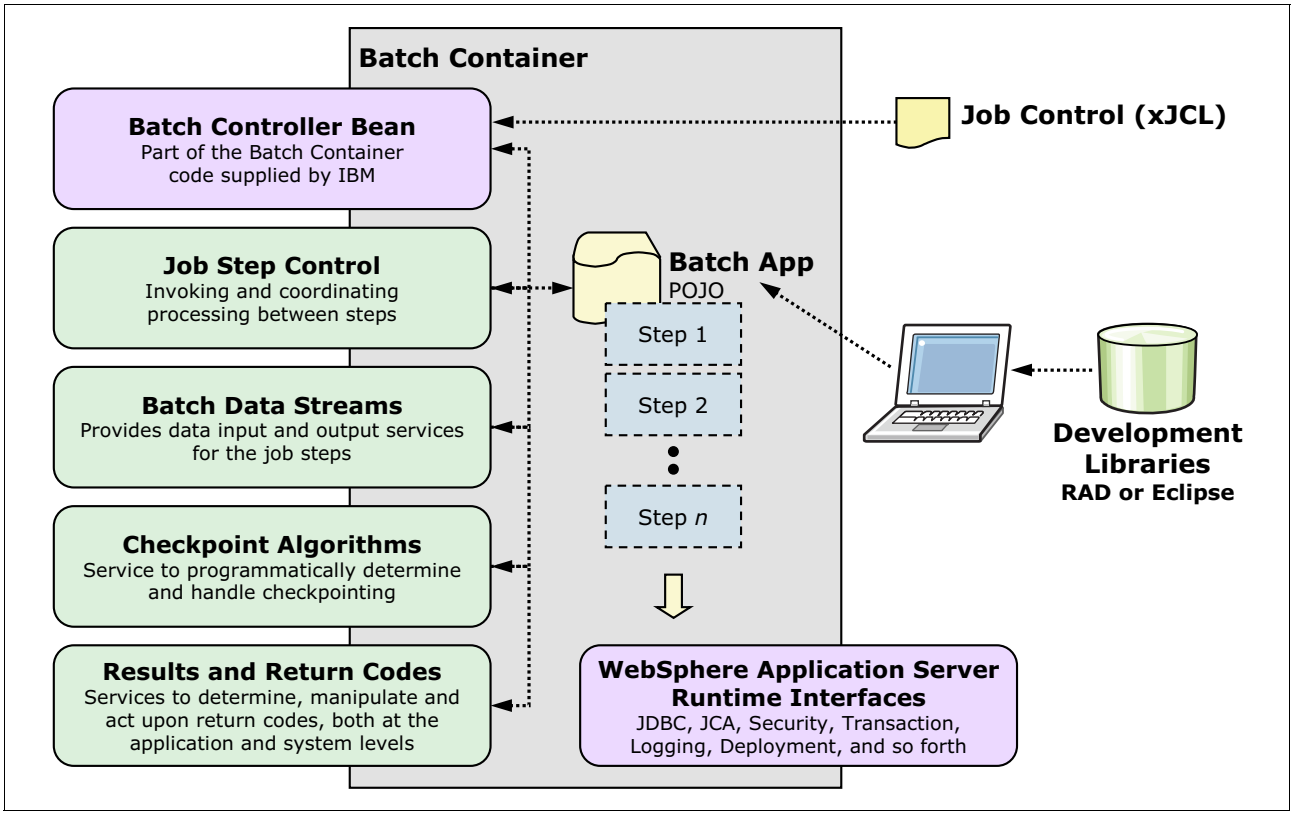


Figure 6-4 Batch programming model

The diagram illustrates the following principle interfaces:

- A batch application includes a stateless session *batch controller bean* that the product run time provides. This stateless session bean acts as a job step controller. The controller stateless session bean is declared in the application deployment descriptor once per batch application.
- The *job step control* defines the interaction between the batch container and the batch application. A batch job can be composed of one or more *batch steps*. All steps in a job are processed sequentially. Dividing a batch application into steps allows for separation of distinct tasks in a batch application. You can create batch steps by implementing the `com.ibm.websphere.batch.BatchJobStepInterface` interface. This interface provides the business logic of the batch step that the batch run time starts to run the batch application.

- The *batch data stream* abstracts a particular input source or output destination for a batch application. It defines the interaction between the batch container and a concrete BatchDataStream implementation. Methods on the BatchDataStream interface allow the batch runtime environment to manage the *data stream* that is used by a batch step. For example, one of the methods retrieves current cursor information from the stream to track how much data is processed by the batch step.
- A *checkpoint algorithm* defines the interaction between the batch container and a custom checkpoint policy implementation. The batch runtime environment uses *checkpoint algorithms* to decide how often to commit global transactions and under which batch the steps are started. A checkpoint policy is used to determine when the batch container checkpoints a running batch job. Checkpointing enables a restart to occur after a planned or unplanned interruption.

  The XML job control language (xJCL) definition of a batch job defines the checkpoint algorithms to be used. Properties that are specified for checkpoint algorithms in xJCL allow you to customize checkpoint behavior, such as transaction timeouts and checkpoint intervals, for batch steps.

  WebSphere Application Server V8.5 provides time-based and record-based checkpoint algorithms. A checkpoint algorithm system programming interface (SPI) is also provided for building additional custom checkpoint algorithms.
- A *results and return code algorithm* defines the interaction between the batch container and a custom results algorithm. The results algorithm provides the overall return code for a job. The algorithm has visibility to the return codes from each of the job steps.

  Results algorithms are an optional feature of the batch programming model. Results algorithms are applied to batch steps through xJCL. The algorithms are used to manipulate the return codes of batch jobs. Additionally, these algorithms are place holders for triggers based on step return codes. WebSphere Application Server V8.5 includes one ready-to use results algorithm.

  Batch job return codes fall into two groups named *system* and *user application*. System return codes are defined as *negative integers*, and user application return codes are defined as *positive integers*. Both system and user ranges include the return code of zero (0). If a user application return code is specified in the system return code range, a warning message is posted in the job and system logs.

6.2.2 Compute-intensive programming model

Compute-intensive applications are applications that run compute-intensive work that does not fit comfortably into the traditional Java EE request and response paradigm. A number of characteristics can make these applications unsuitable for traditional Java EE programming models:

- The need for asynchronous submission and start of work
- The need for work to run for extended periods of time
- The need for individual units of work to be visible to and manageable by operators and administrators

The compute-intensive programming model provides an environment that addresses these needs.

A compute-intensive application is started by the application server in the same way as other Java EE applications. If the application defines any start-up beans, those beans are run when the application server starts. When a job arrives for the application to run, the compute-intensive execution environment starts the `CIControllerBean` stateless session

bean. This bean is defined in the application EJB module deployment descriptor. The Java Naming and Directory Interface (JNDI) name of this stateless session bean is specified in the xJCL for the job.

For each job step, the `CIControllerBean` stateless session bean completes the following actions:

1. Instantiates the application `CIWork` object specified by the class name element in the xJCL for the job step. It uses the no-argument constructor of the CIWork class.
2. Starts the `setProperties()` method of the `CIWork` object to pass any properties that are defined in the xJCL for the job step.
3. Looks up the work manager that is defined in the deployment descriptor of the enterprise bean module. The step uses the work manager to asynchronously call the `run()` method of the `CIWork` object.

If the job is canceled before the `run()` method returns, the `CIControllerBean` starts the `CIWork` object `release()` method on a separate thread. The developer of the long-running application needs to arrange for logic in the `release()` method to cause the `run()` method to return promptly. The job remains in a cancel pending state until the `run()` method returns.

If the job is not canceled and the `run()` method returns without returning an exception, the job completed successfully. If the `run()` method returns an exception, the job status is execution failed. After the `run()` method returns either successfully or with an exception, no further calls are made to the CIWork object. All references to the `run()` method are dropped.

Unlike other batch jobs, compute-intensive jobs consist of a single job step. This job step is represented by an instance of a class that implements the `com.ibm.websphere.ci.CIWork` interface. The CIWork interface extends the `commonj.work` interface from the application server asynchronous beans programming model and Java Specification Request (JSR) 237. These extensions consist of two methods that provide a way to pass the job-step-specific properties that are specified in the job to the CIWork object.

6.3 WebSphere Batch components

Batch applications are hosted in *endpoints*. Configuring the batch environment includes configuring the job scheduler and endpoints. The job scheduler accepts job submissions and determines where to run them. Configuration for the job scheduler includes the selection of the deployment target, data source JNDI name, database schema name, and endpoint job log location.

WebSphere Batch includes the following main components, which are described in detail in the sections that follow:

- Job scheduler
- Batch container
- xJCL
- Interfaces
- Endpoints
- Batch database
- Batch toolkit

6.3.1 Job scheduler

The job scheduler provides job management functions, such as submit, cancel, and restart. It accepts job submissions and determines where to run them. It maintains a history of all job activity, including waiting jobs, running jobs, and completed jobs. Stand-alone application servers, dynamic clusters, and static clusters can host the job scheduler.

As part of managing jobs, the job scheduler uses a relational database to store job information. This relational database can be any relational database that is supported by WebSphere Application Server. If the scheduler is clustered, the database must be a network database, such as DB2.

Optionally, jobs can be controlled through an external workload scheduler, such as Tivoli Workload Scheduler.

6.3.2 Batch container

The batch container is the heart of the batch application support that is provided in WebSphere Application Server V8.5. The batch container provides the execution environment for batch jobs. It provides application services such as checkpoint or restart and job-logging. A WebSphere cell can include multiple batch containers.

The batch container runs a batch job under the control of an asynchronous bean, which can be thought of as a *container-managed thread*. The batch container ultimately processes a job definition and carries out the lifecycle of a job. It uses a relational database to store checkpoint information for transactional batch applications.

The batch container provides the following services:

- Checkpointing, which involves resuming batch work from a selected position
- Result processing, which involves intercepting and processing step and job return codes
- Batch data stream management, which involves reading, positioning, and repositioning data streams to the following destinations:
  - Files
  - Relational databases
  - Native z/OS data sets
  - Other types of input and output resources

Figure 6-5 illustrates the main architecture of the batch container. The function that dispatches the job is the batch container. The batch controller bean controls the batch application and processes the job definition from start to finish.

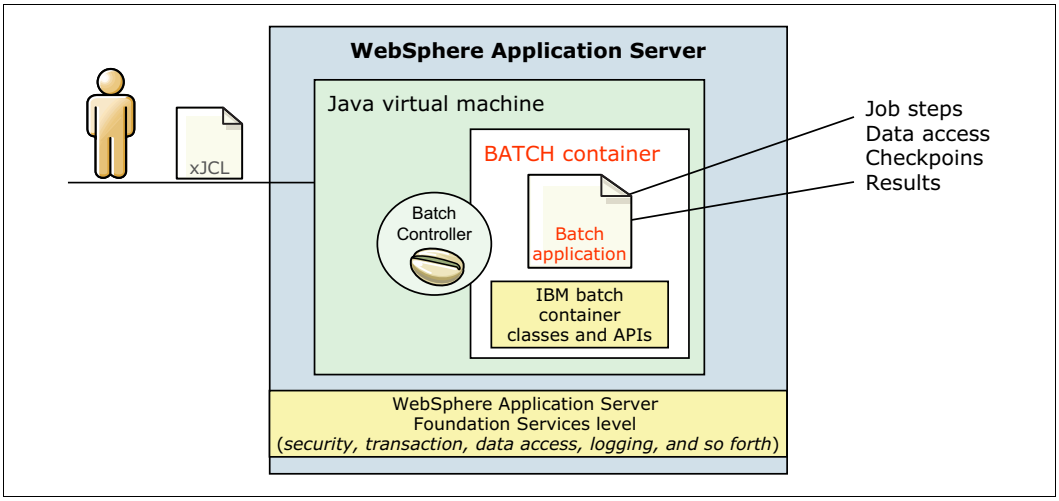


Figure 6-5 Batch container view

Batch applications run under the control of an asynchronous bean, which is similar to a container-managed thread.

6.3.3 xJCL

Jobs are described by using an *XML job control language* (xJCL) that identifies the batch application to run. The xJCL definition is similar to a traditional JCL. The xJCL definition of a job is not part of the batch application. Rather, it is constructed separately and submitted to the job scheduler to run. The job definition identifies which batch application to run, and its inputs and outputs. It also identifies which checkpoint algorithms and results algorithms to use. The job scheduler uses information in the xJCL to determine where and when to run the job.

Figure 6-6 shows an xJCL example.

Figure 6-6 xJCL example

xJCL sample for a compute-intensive job

Example 6-1 shows a generic compute-intensive sample.

Example 6-1 xJCL sample for a compute intensive job

```
<?xml version="1.0" encoding="UTF-8" ?>
<job name="OpenGrid" class="xyz" accounting="accounting info"
default-application-name="tryit"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<job-scheduling-criteria>
<required-capability expression="someExpression" />
<required-capability expression="anotherExpression" />
</job-scheduling-criteria>

<substitution-props>
<prop name="PATH" value="C:\\windows;C:\\java\\jre\\bin" />
</substitution-props>

<job-step name="Step1" application-name="tryit">

<env-entries>
<env-var name="PATH" value="${PATH}" />
<env-var name="CLASSPATH" value="C:\\windows" />
</env-entries>

<exec executable="java">
<arg line="command line args here" />
<arg line=" and more command line args here" />
</exec>
```

```
</job-step>

</job>
```

xJCL sample for a batch job

Example 6-2 shows a sample batch job that demonstrates how to start existing session beans from within job steps.

Example 6-2 xJCL sample for a batch job

```
<job name="PostingsSampleEar"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <jndi-name>ejb/com/ibm/websphere/samples/PostingsJob</jndi-name>
<step-scheduling-criteria>
<scheduling-mode>sequential</scheduling-mode>
    </step-scheduling-criteria>

    <checkpoint-algorithm name="${checkpoint}">
   <classname>com.ibm.wsspi.batch.checkpointalgorithms.${checkpoint}</classname>
   <props>
      <prop name="interval" value="${checkpointInterval}" />
   </props>
    </checkpoint-algorithm>

    <results-algorithms>
   <results-algorithm name="jobsum">
      <classname>com.ibm.wsspi.batch.resultsalgorithms.jobsum</classname>
   </results-algorithm>
    </results-algorithms>

    <substitution-props>
        <prop name="wsbatch.count" value="5" />
        <prop name="checkpoint" value="timebased" />
        <prop name="checkpointInterval" value="15" />
        <prop name="postingsDataStream"
value="${was.install.root}${file.separator}temp${file.separator}postings" />
    </substitution-props>
<job-step name="Step1">
        <jndi-name>ejb/DataCreationBean</jndi-name>
        <!-- apply checkpoint policy to step1 -->
        <checkpoint-algorithm-ref name="${checkpoint}" />
        <results-ref name="jobsum"/>
      <batch-data-streams>
         <bds>
                      <logical-name>myoutput</logical-name>

<impl-class>com.ibm.websphere.samples.PostingOutputStream</impl-class>
                      <props>
                      <prop name="FILENAME" value="${postingsDataStream}" />
             </props>
         </bds>
      </batch-data-streams>
           <props>
              <prop name="wsbatch.count" value="${wsbatch.count}" />
      </props>
```

```
    </job-step>

    <job-step name="Step2">
       <step-scheduling condition="OR">
     <returncode-expression step="Step1" operator="eq" value="0" />
          <returncode-expression step="Step1" operator="eq" value="4" />
  </step-scheduling>
       <jndi-name>ejb/PostingAccountData</jndi-name>
     <checkpoint-algorithm-ref name="${checkpoint}" />
     <results-ref name="jobsum"/>
     <batch-data-streams>
        <bds>
                    <logical-name>myinput</logical-name>
           <impl-class>com.ibm.websphere.samples.PostingStream</impl-class>
                    <props>
              <prop name="FILENAME" value="${postingsDataStream}" />
           </props>
        </bds>
     </batch-data-streams>
    </job-step>

    <job-step name="Step3">
      <step-scheduling>
          <returncode-expression step="Step2" operator="eq" value="4" />
      </step-scheduling>
      <jndi-name>ejb/OverdraftAccountPosting</jndi-name>
      <checkpoint-algorithm-ref name="${checkpoint}" />
          <results-ref name="jobsum" />
      <batch-data-streams>
      <bds>
              <logical-name>dbread</logical-name>

<impl-class>com.ibm.websphere.samples.OverdraftInputStream</impl-class>
       </bds>
      </batch-data-streams>
     </job-step>
</job>
```

6.3.4 Interfaces

WebSphere Batch offers the following interfaces for interacting with the job scheduler:

- Job management console
- Command-line interface
- Programmatic

Job management console

The *job management console* is a graphical user interface (GUI) that allows you to perform job management functions and interact with the job scheduler. It is hosted in the same server or cluster that hosts the job scheduler function.

The job management console provides the following essential job management functions:

- Job submission
- Job operations (cancel, stop, suspend, resume, restart, and purge)

- Job repository management (save and delete job definitions)
- Job schedule management (create and delete job schedules)

When role-based security is enabled, you must be granted the `lrsubmitter` role, the `lradmin` role, or the `lrmonitor` role through the administrative console to access the job management console. When group-based security is enabled, you must be in the user group of the job or the administrative group to access the job management console.

Command-line interface

The command-line interface provides a text-based interface that interacts with the job scheduler to submit and control jobs in the system. Using the command-line interface, you can perform the following functions:

- Display usage information for the command-line interface
- Submit a job to the job scheduler
- Cancel a previously submitted job
- Restart a job
- Purge job information
- Save an xJCL to the job repository
- Remove a job from the job repository
- Show an xJCL that is stored in the job repository
- Show the status of a Compute Grid job
- Suspend a job
- Resume a previously suspended job
- Display the output for a job
- Display the return code of a batch job
- Submit a recurring job request to the job scheduler
- Modify an existing recurring job request
- Cancel an existing recurring job request
- List all existing recurring job requests
- Show all recurring jobs of a request

Programmatic

With a programmatic interface, the job scheduler exposes application programming interfaces (APIs) that are available as either web services or EJB for the administration of jobs.

6.3.5 Endpoints

Batch applications are packaged as Java EE EAR files and are deployed similar to a transactional Java EE application. Batch applications are hosted in an *endpoint*. Endpoints run the work, and can be started and stopped based on job execution agreements. WebSphere Batch has Java EE type endpoints that can be application servers or dynamic/non-dynamic clusters. The information in xJCL determines when and in which endpoint to run the job.

The runtime environment is provided by a WebSphere Application Server supplied Java EE system application. The system application serves as an interface between the job scheduler and batch applications. It provides the runtime environment for both compute-intensive and transactional batch applications.

The application is installed automatically to an application server or cluster when you deploy a compute-intensive or transactional batch application. Therefore, the application server or cluster becomes an endpoint during the first deployment of a batch application.

From the deployment descriptor of a compute-intensive or transactional batch application, the WebSphere run time determines whether the deployment target needs to have special runtime support. The system application is installed as a part of the user application deployment process to convert the deployment target to be an endpoint.

Similar to the system application that provides the job scheduler function, this system application is not visible as an installed application in the administrative console. Both the compute-intensive and transactional batch applications are deployed to endpoints similar to a Java EE application. The WebSphere Batch administrator can then define service policies for the application in preparation for submitting a job.

6.3.6 Batch database

The job scheduler and batch container both require access to a relational database. The relational database used is JDBC connected. WebSphere Batch automatically configures a simple file-based Derby database by default to help get a functioning environment up quickly.

Important: The default Derby database does not support a clustered job scheduler or a clustered batch container. Therefore, it is suggested that you do not use the Derby database for production systems.

A highly available environment includes both a clustered job scheduler and one or more clustered batch containers, requiring a network database. Production grade databases, such as DB2 and other databases, are suggested for this purpose.

All batch containers in the same cell must use the same relational database type.

6.3.7 Batch toolkit

The batch toolkit supplied with WebSphere Application Server includes tools to facilitate batch application development. It combines batch development tools into a ready-to-use environment, and includes simple command-line utilities that deal with packaging applications and other tasks.

Rational tools also now support WebSphere batch applications. The batch toolkit does not include the full utilities that come with Rational Application Developer. However, the batch toolkit is available if you do not have Rational Application Developer and need to create batch applications to run in WebSphere Application Server.

The batch toolkit contains the following components:

- Batch framework
- Lightweight batch container
- Packaging tool
- xJCL generator
- Unit test server

Batch framework

The *batch framework* includes wizards, project facets, samples, scripts, and a run time for developing batch applications. The framework includes the following sample batch applications that provide code examples and resources:

- The Echo application demonstrates a trivial batch application.
- The TestBatchJobStep application demonstrates use of the basic batch programming interfaces.

Lightweight batch container

The *lightweight batch container* is a non-Java EE batch run time that uses the batch programming model. It is designed to help developers create logic and complete flow testing of batch applications. It runs in a Java development environment such as Eclipse. The batch container can generate packaging properties for batch applications to be used by the packaging tool.

Packaging tool

Batch applications run in WebSphere Application Server and are installed as Java EE EAR files. The *packaging tool* handles the details of the EJB deployment descriptor and other details about EAR file creation. It uses a properties file from the lightweight batch container for details about packaging an EAR file. The tool is the **`WSBatchPackager`** command-line tool.

xJCL generator

The *xJCL generator* generates job definitions.

Unit test server

The batch framework includes a *unit test server* environment that runs inside a stand-alone application server. This environment allows you to test batch applications before deploying to a production environment. The unit test environment includes a batch container, a job scheduler, and a Derby batch database for testing purposes.

Rational Application Developer includes the Java batch programming model, which allows you to build robust batch applications to perform long-running bulk transaction processing and compute-intensive work.

6.4 Batch workflow

The following steps describe the workflow of a submitted batch job:

1. Batch jobs are submitted to the system by using the job management console. They can also be submitted programmatically by using EJB, Java Message Service (JMS), or web services.
2. Each job is submitted in the form of an xJCL document.
3. The job scheduler selects the best endpoint for job execution based on metrics.
4. The endpoint sets up the jobs in the batch container and runs the batch steps based on the definitions in the xJCL.

Figure 6-7 shows that a job is submitted with its job control definition. The job scheduler then analyzes the request, and the job is dispatched to the batch endpoint. The batch endpoint begins execution, and the batch application starts.

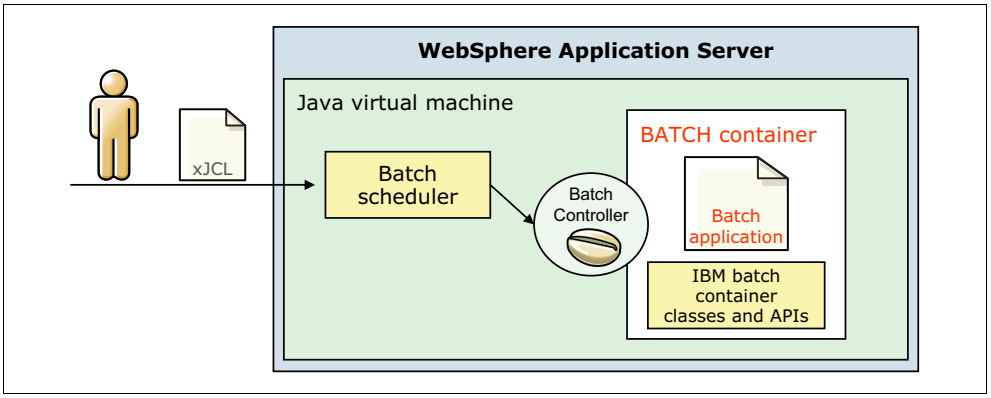


Figure 6-7 Job flow when submitted from the job scheduler to the batch endpoint

The job scheduler aggregates job logs and provides lifecycle management functions such as start, stop, cancel, and other functions.

Figure 6-8 shows a complete picture of the batch environment.

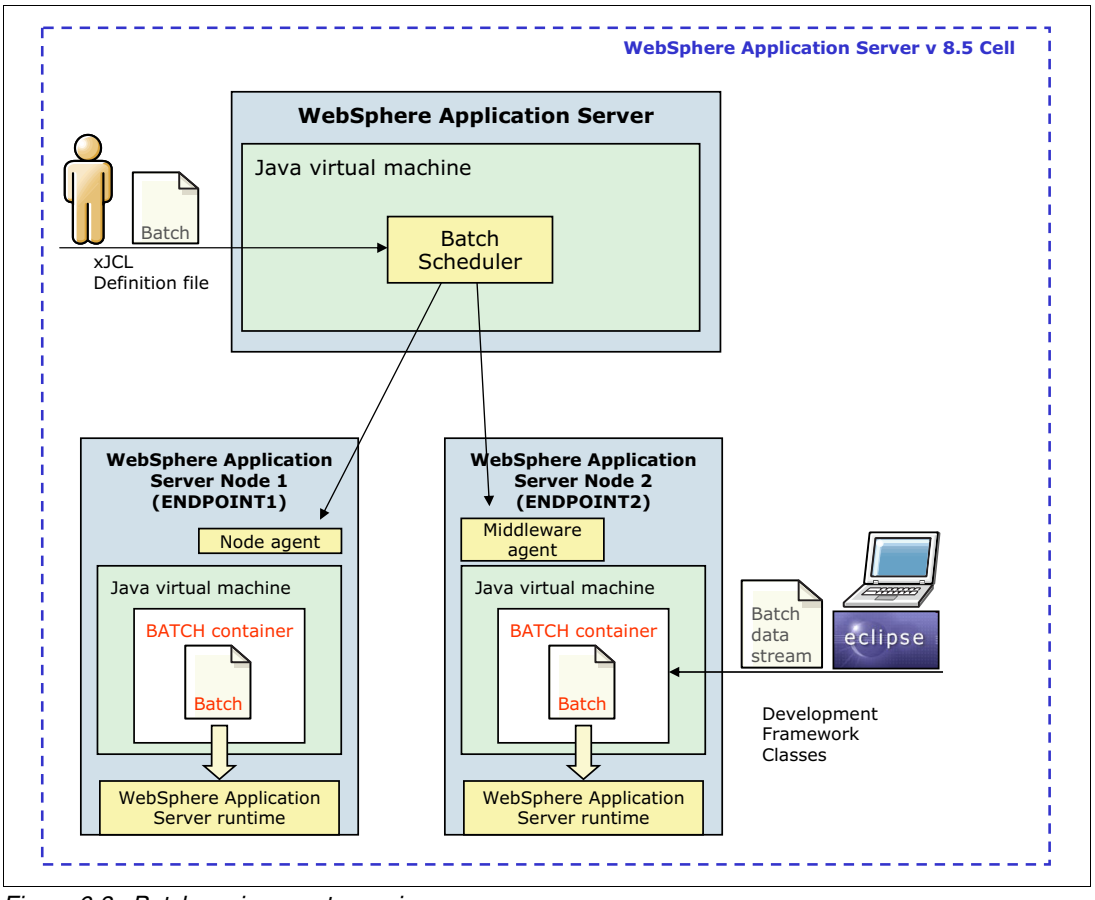


Figure 6-8 Batch environment overview

The job scheduler scope is a WebSphere Application Server cell, and can dispatch the batch job to any appropriate server endpoint in that cell.

6.5 New features in WebSphere Application Server V8.5 for WebSphere Batch

In addition to fully incorporating WebSphere batch functions into WebSphere Application Server V8.5, the following new enhancements are also included:

- Parallel batch
- Enterprise integration
- Cobol support
- CommandRunner utility job step

Other enhancements are included that are related to the following areas:

- Programming model enhancements for these functions:
  - OSGi batch applications
  - Record processing policy
  - Record metrics
  - Job and step listener
  - Persistent job context
  - Configurable transaction model
  - Batch data stream timeout
- Job definition enhancements for multi-threading, parallel steps, and heterogeneous steps
- Operational enhancements for features such as group security, memory overload protection, job log SPI, and SMF Type 120 Subtype 9

6.5.1 Parallel batch

Parallel batch is the ability to split the work and process jobs as multiple subordinate jobs concurrently. For more information, see the information center at:

`http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp`

The parallel batch function includes the following key features:

- Container-managed parallel processing
- Multiple cores for efficiency
- Simple “one job” operational control
- Divide and conquer approach to improve elapsed time
- Near linear runtime performance

The batch container provides a facility and framework for submitting and managing batch jobs that run as a coordinated collection of independent parallel subordinate jobs. Thus, any job can be processed in parallel.

Only a single xJCL file is required. The xJCL file combines the contents of the top-level job xJCL with the contents of the subordinate job xJCL files.

Tip: Because parallel batch is part of the batch container, you do not need to install and configure it. Package the parallel batch APIs in the batch application as a utility Java archive (JAR). No shared library is required. The contents of the `xd.spi.properties` file are part of the xJCL, so no separate `xd.spi.properties` file is required.

Using a parallel batch container operation to start an API involves the following process:

1. The xJCL is submitted to the job scheduler.
2. The job scheduler dispatches the xJCL to an endpoint that runs the application that the xJCL references. The batch container determines whether the job will have subordinate jobs that run in parallel by inspecting the `run` property in the xJCL file.
3. The batch container delegates the running of the job to the parallel batch subcomponent. The parallel batch container starts the parameters of the API and uses the information in the xJCL file to divide the job into subordinate jobs.
4. The batch container starts the LogicalTX synchronization API to indicate the beginning of the logical transaction. The container generates the subordinate job xJCL and submits the subordinate jobs to the job scheduler.
5. The job scheduler dispatches the subordinate jobs to the batch container endpoints so that they can run.
6. The batch container runs the subordinate job. When a checkpoint is taken, the subordinate job collector API is starting. This API collects relevant state information about the subordinate job. This data is sent to the subordinate job analyzer API for interpretation.
7. After all subordinate jobs reach a final state, the beforeCompletion and afterCompletion synchronization APIs are starting. The analyzer API is also started to calculate the return code of the job.

A logical transaction is a unit of work demarcation that spans the running of a parallel job. Its lifecycle corresponds to the combined lifecycle of the subordinate jobs of the parallel job. Because of an extension mechanism, you can customize application-managed resources so that they can be controlled in this unit of work scope for commit and rollback purposes.

Figure 6-9 summarizes the parallel batch architecture and shows where the APIs are called in this process.

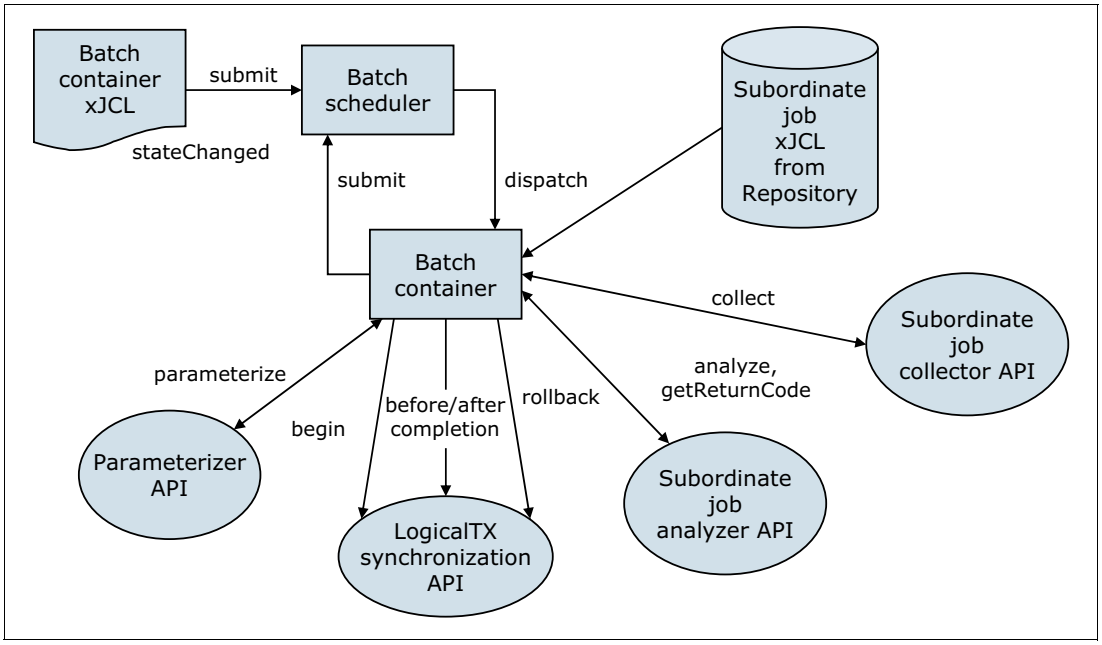


Figure 6-9 Parallel batch architecture

Figure 6-10 describes the parallel batch functions and shows the batch flow from the job scheduler to the batch container endpoints.

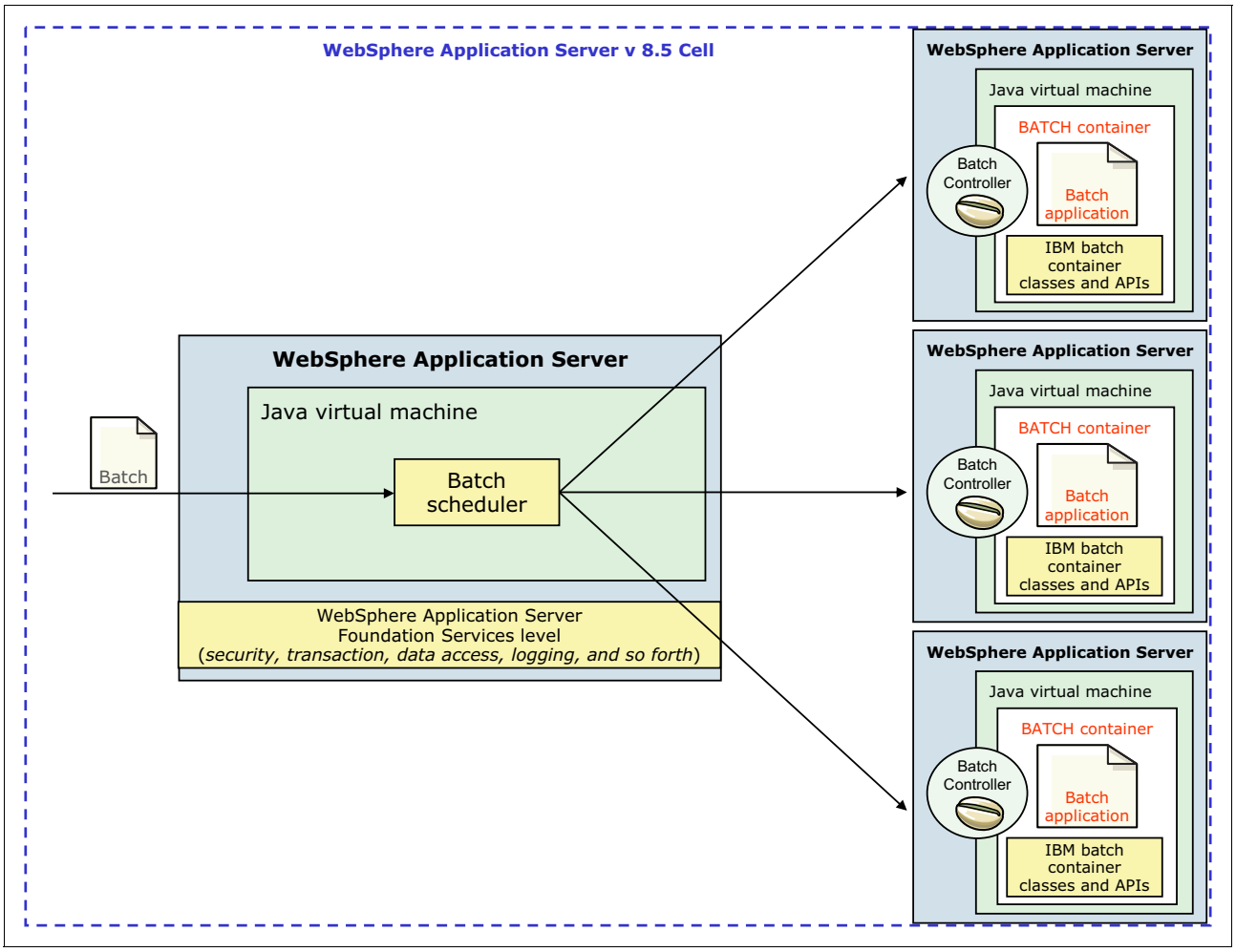


Figure 6-10 Parallel batch functional diagram

6.5.2 Enterprise integration

Many customers already use an external workload scheduler to manage batch workloads. Although a Java batch that runs inside a WebSphere Application Server environment is attractive, controlling batch jobs through an external workload scheduler is important.

You can integrate the job scheduler with an external workload scheduler by configuring and securing the job scheduler, enabling the interface, and running batch jobs.

You can set up the external scheduler interface by using the default messaging provider as a JMS provider. In z/OS, you also have the option of setting up the external scheduler interface by using WebSphere MQ as a messaging provider.

Because an external scheduler does not know how to directly manage batch jobs, a proxy model is used. The proxy model uses a regular job control language (JCL) job to submit, monitor, or submit and monitor the batch job. The JCL job step starts a special program provided by batch called WSGRID. The WSGRID application submits and monitors a specified batch job. WSGRID writes intermediary results of the job into the JCL job log.

WSGRID does not return until the underlying job is complete, providing a synchronous execution model.

Because the external scheduler can manage JCL jobs, it can manage a JCL job that started WSGRID. Using this pattern, the external scheduler can indirectly manage a job. An optional plug-in interface in the job scheduler enables a user to add code that updates the external scheduler operation plan. This update reflects the unique state of the underlying job, such as job started, step started, step ended, or job ended. The WSGRID program is written with special recovery processing. If the JCL job is canceled, the underlying job is canceled also, ensuring that the lifecycles of the two jobs are synchronized.

Enterprise integration includes the following key features:

- WebSphere Application Server V8.5 includes a special “connector” for Tivoli Workload Scheduler and competing workload schedulers.
- Tivoli Workload Scheduler and WebSphere Batch provide a common deployment pattern.
- WebSphere Batch allows full control of the enterprise workload scheduler.

Figure 6-11 shows the workload coming from Tivoli Workload Scheduler directly to the job scheduler, using a workload connector.

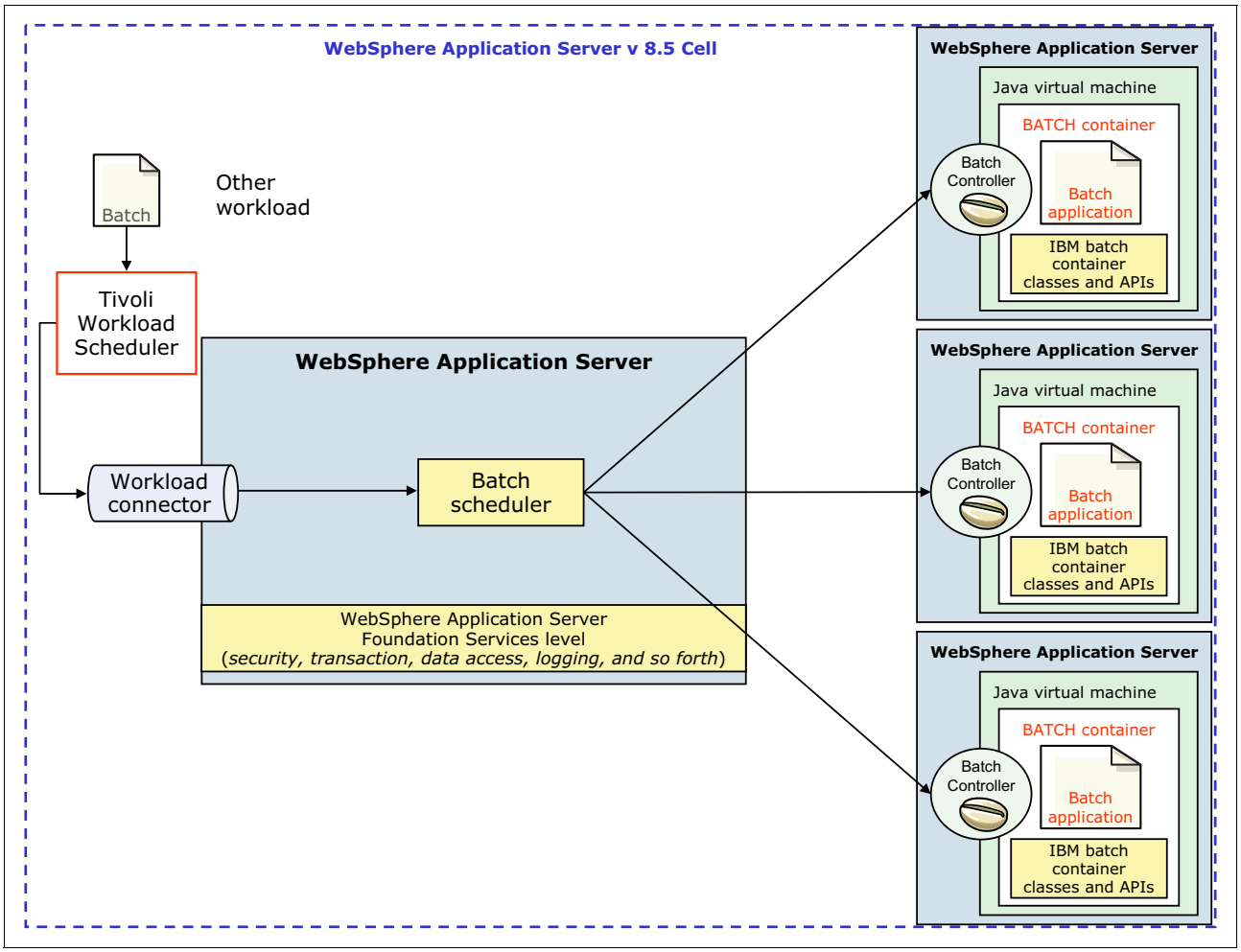


Figure 6-11 Enterprise integration functional diagram

6.5.3 Cobol support

With Cobol support, you can reuse Cobol modules in WebSphere applications. The Cobol container enables Cobol modules to be loaded into the Batch container. They can then be started directly. Java programs can pass parameters into Cobol and retrieve the results. The Cobol call stub generator tool is provided to create the Java call stubs and data bindings. It creates these based on the data and linkage definitions in the Cobol source.

You can use the Cobol container to start Cobol modules from a batch application, creating a direct integration of Cobol into Java batch processing. You can also dynamically update a Cobol module without having to restart the application server.

With WebSphere Application Server V8.5, Cobol support includes the following key features:

- In z/OS, you can call standard Cobol modules from Java on the same thread in same process.
- Java and Cobol run in same transaction scope.
- DB2 connections managed by WebSphere are shareable with Cobol.
- You can use Cobol working storage isolation per job step or per remote call.
- IBM Rational Application Developer tooling is available for Java call stub generation.

For more information about Cobol features on z/OS, see 16.6.2, “WebSphere Batch on z/OS” on page 540.

6.5.4 CommandRunner utility job step

Use the `CommandRunner` utility job step to run shell command lines as job steps. The command lines can include shell commands, shell scripts, and compiled programs. The `CommandRunner` utility runs the specified shell command line in an operating system process. Standard output and standard error streams are captured and written to the job log. The command-line return code is captured and set as the step return code. If the job step is canceled, the return code is -8.

Use the syntax in Example 6-3 for the `CommandRunner` utility.

Example 6-3 CommandRunner syntax

```
<job-step name={step_name}>
<classname>com.ibm.websphere.batch.utility.CommandRunner</classname>
{job_step_properties}
</job-step>
```

The job step code shown in Example 6-4 runs a command-line Java program.

Example 6-4 Job step that run a program by using the CommandRunner utility

```
<job-step name="RunJava">
<classname>com.ibm.websphere.batch.utility.CommandRunner</classname>
<props>
<prop name="com.ibm.websphere.batch.cmdLine"
                  value="java.exe com.ibm.websphere.batch.samples.TestCase" />
<prop name="CLASSPATH" value="${user.dir}\testcases;${user.dir}\bin" />
<prop name="Path" value="${java.home}\bin;${env:Path}" />
</props>
</job-step>
```

7

Infrastructure

You must consider many factors when planning and designing an infrastructure for a WebSphere Application Server environment. The most important aspects to create a WebSphere Application Server infrastructure to run a successful WebSphere project are addressed. This chapter includes the following sections:

- Infrastructure planning
- Environment planning
- Design considerations
- Sizing the infrastructure
- Monitoring
- Backup and recovery
- Cloud infrastructure

For more information about topics relevant for infrastructure decisions, see:

- Monitoring is addressed in Chapter 10, “Performance, scalability, and high availability” on page 285.
- Security considerations are addressed in Chapter 15, “Security” on page 469.

Terminology: In this chapter, the term *system* is a synonym for physical machines, logical partitions (LPARs), and operating system image.

7.1 Infrastructure planning

This section gives a general overview of the typical phases for a project. It explains how to gather requirements and apply those requirements to a WebSphere Application Server project.

Typically, a new project starts with only a concept. Little is known about specific implementation details, especially how they relate to the infrastructure. Your development team and infrastructure team must work closely together to meet the needs of the overall application environment.

Gather information that falls into the following categories:

- Functional requirements

  Functional requirements are usually determined by the use of the application and related functions.

- Nonfunctional requirements

  Nonfunctional requirements describe the properties of the underlying architecture and infrastructure such as reliability, scalability, availability, and security.

- Capacity requirements

  Capacity requirements include traffic estimates, traffic patterns, and expected audience size.

- Performance requirements

  Performance requirements are the response time of HTTP page requests or the processing time for batches.

Requirements gathering is an iterative process. Make sure that your plans are flexible enough to deal with future changes in requirements. Always keep in mind that the plans can impact other parts of the project. To support this effort, make sure that dependencies and data flows, whether application or infrastructure related, are clearly documented.

With this list of requirements, you can start to create the first draft of your design. Develop the following designs:

- Application design

  To create your application design, use functional and non-functional requirements to create guidelines for application developers about how the application is built.

- Implementation design

  This design defines the target deployment infrastructure on which the application is deployed.

  The final version of this implementation design contains details about the hardware, processors, software, and versions that are installed. However, you do not begin with all of these details. Initially, your implementation design lists component requirements. These components can include a database, a set of application servers, a set of web servers, and other components that are defined in the requirements phase. For more information, see 7.3, “Design considerations” on page 164.

With these two draft designs, you can begin formulating counts of servers, network requirements, and the other items related to the infrastructure. For more information, see 7.4, “Sizing the infrastructure” on page 170.

The last step in every deployment is to tune the system and make sure it can handle the projected load that the non-functional requirements specified. For more information, see 7.5, “Monitoring” on page 171.

7.2 Environment planning

Your infrastructure is made up of several different and distinct environments. Each environment has a specific function to answer a functional or nonfunctional requirement. An infrastructure can include the following possible environments:

- Development

  This environment is reserved for application developers to develop future applications and is often a simple stand-alone system without high availability. The Liberty profile provides the opportunity to run a light weight, flexible, and dynamic development environment that mirrors a WebSphere Application Server environment.

  You can download WebSphere Application Server for Developers for no additional cost at:

  `http://www.ibm.com/developerworks/downloads/ws/wasdevelopers/`

  This version cannot be used by multiple users at the same time.

- System Integration

  This environment is dedicated to application developers and integration teams to test the application deployment procedure and applications in a simple, highly available environment. This environment must contain a minimum of high availability components to ensure that the applications are compatible with these components. Use the deployment environment with a Liberty profile only if the intended production environment is also a Liberty profile.

- Technical qualification

  This environment is reserved for the infrastructure team to develop and test the technical procedures, such as backup and recovery and daily maintenance operations. It is also used to test new hardware and middleware patches. Usually, the infrastructure team builds this environment to ensure all of the components are compatible and create installation procedures. The technical qualification is often a light version of the production environment.

- Functional qualification

  This environment is dedicated to functional testers. Enough power must be available to support a few people who test concurrently.

- Performance

  The performance environment must mirror the production environment as closely as possible. This environment runs tuning and tests new scalability and high availability techniques before applying the changes to the preproduction and production environments.

  Important: To avoid direct tuning on the production environment, create a performance environment identical to the production.

- Preproduction

  The preproduction environment must be an exact copy of the production environment. All of the changes must be successfully tested in this environment. If a disaster occurs in your production, the preproduction environment can be used as a temporary substitute.

- Production

  The production environment is the final stage, and is dedicated to run the business applications and serve user requests. This environment is the most important environment of your infrastructure. You must clearly and strictly define specific rules and procedures to manage it.

Tip: To maintain these environments, use a list of roles, rules, and procedures. Try to keep the same configuration (such as versions and tuning) on all the systems. If you cannot, document the state and the difference between all the environments. The challenge of building and working daily with your infrastructure is to keep it as clean as possible.

7.3 Design considerations

This section provides information about key infrastructure concepts to consider when designing a WebSphere Application Server environment. These concepts significantly affect your design. This section includes the following topics:

- Scalability
- High availability
- Load balancing and failover
- Caching
- Infrastructures using a Liberty profile

7.3.1 Scalability

Scalability is the ability of the infrastructure to properly handle an increase in load volume. Most of the time, it means increasing throughput by adding more resources.

Understanding the scalability of the components in your WebSphere Application Server infrastructure and applying appropriate scaling techniques can greatly improve availability and performance. Scalability is required for high availability and performance.

To determine your key infrastructure components and identify scaling techniques that are applicable to your environment, perform these steps:

1. Understand the application environment.

   Applications are key to the scalability of the infrastructure. Ensure that the applications are designed for scaling. Understand the component flow and traffic volumes that are associated with existing applications, and evaluate the nature of new applications. You must understand each component and system that is used in the transaction flow.

2. Categorize your workload.

   Knowing the workload pattern for a site determines where you focus scalability efforts and which scaling techniques you need to apply. For example, a customer self-service site, such as an online bank, must focus on transaction performance and the scalability of customer information databases. These considerations are not as significant for a publish/subscribe site, where a user signs up to have data sent, usually through a mail message.

   Websites with similar workload patterns can be classified into the following site types:

   – Publish/subscribe
   – Online shopping
   – Customer self-service

- Online trading
- Business-to-business

3. Determine the components most affected.

   Knowing the workload pattern for an application determines where you focus scalability efforts and which scaling techniques you need to apply. From a scalability viewpoint, the infrastructure has these key components:

   - Load balancers
   - The application servers
   - Security services
   - Transaction and data servers
   - The network

   First focus on those components that are most heavily used by the key transactions of your applications. When the load increases, these components can become bottlenecks for your infrastructure.

4. Select the scaling techniques to apply.

   For the important components you identified, develop scaling approaches, which might include the following approaches:

   - Scaling up (or vertical scalability)

     Scaling up is done inside a component. You must add more resources (such as memory and processor power) to this component to handle the load.

   - Scaling out (or horizontal scalability)

     Scaling out is an alternative to scaling up, and involves increasing the number of instances of the component. For example, instead of doubling the number of processors and memory in a system, keep your first system and add a second identical system.

     Figure 7-1 illustrates the difference between scaling up and scaling out.

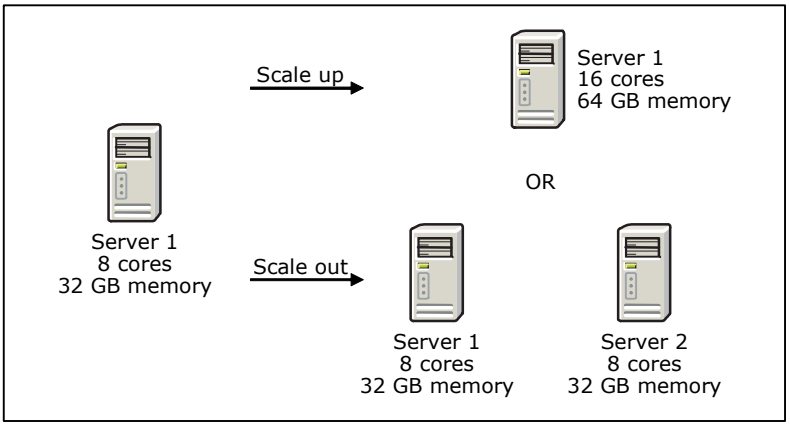


Figure 7-1 Scale up and scale out approaches

   - Using appliance servers

     You can use a dedicated appliance to perform a specific action for your workload. These systems are fast and optimized for specific functions. Be careful to not introduce single point of failure (SPOF).

– Segmenting the workload

  Another approach is to segment the workload into different chunks to obtain more consistent and predictable response times. Each chunk can be dedicated to a specific business area. This sharing improves the caching and the management. For example, you can segment the workload by global regions.

  The new Intelligent Management function of WebSphere Application Server 8.5 allows you to automatically detect and handle health problems and SLA violations. Using dynamic clusters also allows you to provide resources based on current demand and application policies.

– Using batch requests

  Reduce the total number of requests by using batch requests. The goal is to limit the additional cost of multiple requests. Batch requests are usually run during low peak hours. WebSphere Application Server V8.5 supports the development and deployment of Java batch applications.

– Aggregating user data

  To keep applications from accessing customer data from multiple existing applications, aggregate this data in a single back-end system. This action limits the number of connections to multiple systems.

– Managing connections

  Minimize the number of connections between your infrastructure layers to reduce the number of connections. You can use pools to share and maintain the connections.

– Using caching techniques

  You can improve performance and scalability by using caching techniques at different layers of the infrastructure. Caching limits the number of requests and reduces the consumption of component resources. Products based on caching techniques, such as WebSphere eXtreme Scale, are also used to scale a solution.

Each additional component (processors, memory, or Java virtual machine (JVM) in your infrastructure requires additional management. Therefore, the throughput cannot be linear.

5. Apply the techniques.

   Scalability testing needs to be a part of the performance testing. It is crucial that you determine whether the scaling techniques are effective and that they do not adversely affect other areas.

   Remember: Manageability, security, and availability are critical factors in all design decisions. Do not use techniques that provide scalability and compromise any of the previously mentioned critical factors.

6. Re-evaluate.

   Recognize that any system is dynamic. At some point, the initial infrastructure needs to be reviewed and possibly expanded. Changes in the nature of the workload can create a need to re-evaluate the current environment. Large increases in traffic require examination of the system configurations. Scalability is not a one-time design consideration. It is part of the growth of the environment.

For more information about this procedure, see:

http://www.ibm.com/developerworks/websphere/library/techarticles/hipods/scalability.html

For more information about scaling techniques, see 10.2, “Scalability” on page 289.

7.3.2 High availability

Designing an infrastructure for high availability means that your environment can survive the failure of one or multiple components. High availability creates redundancy by avoiding any SPOF on any layer (network, hardware, processes, and so on). The number of failing components your environment must survive without losing service depends on the requirements for your specific environment.

Consider the methods to identify high availability needs in an infrastructure:

- Talk to the sponsor of your project to identify the high availability needs for each of the services used. Because high availability in most cases means redundancy, it increases the cost of the implementation.

  Not every service has the same high availability requirements. Therefore, it might be a waste of effort to plan for full high availability for these types of services. Be careful when evaluating because the high availability of the whole system depends on the least available component. Determine where and what type of high availability are required to meet the service level agreements (SLAs) and non-functional requirements.
- After you gather the high availability requirements, review every component of the implementation design you developed in 7.1, “Infrastructure planning” on page 162. Determine how significant each component is for the availability of the service and how a failure might affect the availability of your service.
- Evaluate every component that you identified in the previous step against the following checklist:
  - How critical is the component for the service?

    The criticality of the component affects how much you are willing to invest to make this component highly available.
  - Consider regular maintenance.

    In addition to failure of components, consider maintenance and hang situations.
  - Is the service under your control?

    Sometimes components in the architecture are out of your control, such as external services provided by someone else. If the component is out of your control, document this component as an additional risk and inform the project sponsor.
  - What do you need to do to make the component highly available?

    Sometimes you have more than one option to make a component highly available. Select which option best fits your requirements.
  - Does the application handle outages in a defined way?

    Check with the application developers on how the application handles an outage of a component. Depending on the component and the error situation, the application might need a specific design or error recovery coded before it can use the high availability features of the infrastructure.
  - Prioritize your high availability investments.

    Decide the high availability implementation based on the criticality of the component and the expected outage rate. Document any deviations from the requirements gathering.
  - Size every component in a way that it can provide sufficient capacity even in cases of a failure of a redundant part.

After you complete the high availability design, update the implementation design to include the high availability features.

7.3.3 Load balancing and failover

As a result of the design considerations for high availability, you might identify several components that need redundancy. Consider how to implement redundancy to ensure that you get the most benefit from the systems during normal operations. You also need to consider how to manage a seamless failover if a component fails. These design considerations introduce the following techniques:

- Load balancing

  Load balancing refers to spreading the load across multiple, available copies of a component for optimum usage of the available resources.

- Failover

  Failover is the capability to automatically detect the outage of a component and route requests around the failed component. When the failed resource becomes available, the system detects it automatically and transparently rejoins it to workload processing.

To design load balancing and failover, you need to know the load balancing and failover capabilities of each component and how these capabilities can be used. Depending on the features that you use, additional hardware and software might be required to gain high availability.

In a typical WebSphere Application Server environment, you must consider various components, including the following types, when implementing load balancing and failover capabilities:

- Caching proxy servers
- HTTP servers
- Containers, such as the web, Enterprise JavaBeans (EJB), Session Initiation Protocol (SIP), and portlet containers
- Resources (data source or connection factory)
- Messaging engines
- Back-end servers (database, enterprise information systems, and so on)
- User registries

Although load balancing and failover capabilities for some of these components are incorporated into WebSphere Application Server, other components require additional hardware and software.

The on-demand router, introduced with WebSphere Application Server V8.5, is a Java-based HTTP and SIP Proxy Server. It provides health management, application edition management, and performance management features. All these features are integrated with the WebSphere environment, and are aware of the applications that run in this environment. This configuration provides additional options for infrastructure planning.

7.3.4 Caching

Caching is a widely used technique to improve performance of application server environments. WebSphere Application Server provides many caching features at different locations in the architecture.

WebSphere Application Network Deployment provides the following caching features for dynamic or static content:

- Infrastructure edge
  - Caching Proxy provided by Edge Components
  - WebSphere Proxy Server
- HTTP server layer

  Edge Side Include (ESI), which is provided by the WebSphere plug-in, allows in-memory caching of complete pages or fragment of pages.
- Application server layer
  - The dynamic cache service, inside the JVM of the application server, allows cache output of servlets, web services, commands, and JavaServer Pages (JSP)
  - At the data sources level, statements cache for prepared statements and callable statements
  - WebSphere Proxy Server

To use the caching mechanisms provided by WebSphere Application Server and other components of your environment, the application must also be designed for caching. Work in close cooperation with the application architect to design your caching components.

In addition to these caching features provided by WebSphere Application Server Network Deployment, consider using caching devices or external caching infrastructures provided by IBM and third parties. IBM offers the following caching software and appliance solutions:

- WebSphere eXtreme Scale is installed on top of WebSphere Application Server. It provides a powerful distributed object cache to replace disk operations with memory operations. WebSphere eXtreme Scale allows performance, scalability, and high availability.
- IBM WebSphere DataPower XC10 is an appliance with a large amount of memory included that provides a powerful object cache for the applications.

After you complete the design of your cache locations, complete the implementation design to include the caching components.

7.3.5 Infrastructures using a Liberty profile

The Liberty profile enables you to perform unit testing on a simple and flexible environment that can be defined easily by the developer. Only the parts of WebSphere Application Server that are necessary to run this particular application are configured and started.

The Liberty profile can be stored in a compressed file. The application server configuration becomes a development artifact that can be shared with other developers, checked into source repositories, or deployed to other environments.

The applications that are tested in a Liberty profile can be deployed to any WebSphere Application Server V8.5 without change. However, use a similar topology in the deployment environment as in the production environment to identify errors that occur only in that type of topology.

You can also run each application in its own Liberty profile in all environments, including production. The different application servers can be managed with a WebSphere Application Server Network Deployment job manager. This topology can make the deployment of

applications easier. It also separates the different applications, which prevents an error in one application from affecting other applications.

Load balancing and failover for Liberty profile servers can be achieved by using an HTTP Server in front of the application servers. Generate the appropriate web server plug-in configuration by using the job manager. Although this method is sufficient for many applications, the more sophisticated load balancing features of WebSphere Application Server Network Deployment currently cannot be used with Liberty profiles.

7.4 Sizing the infrastructure

After determining the initial application and infrastructure design, determine the system resources that are required for the project to keep the SLAs. Consider which hardware platforms you want to use based on these factors:

- The scaling capabilities of the platform
- The platforms that WebSphere Application Server supports
- The performance, security, and high availability requirements of the environment
- Integration with the current infrastructure
- Scaling techniques, such as horizontal scalability, vertical scalability, and other types of scalability

7.4.1 Sizing static infrastructures

Sizing estimates are based on your input, which means the more accurate the input is, the better the results are. Sizing work assumes an average standard of application performance behavior and an average response time for each transaction. Sizing an infrastructure requires accurate acknowledge of the workload.

To help size your environment, consider the following questions:

- What load does your new infrastructure have to support? Try to determine this answer for each component.
- What performance requirements must be met in terms of response time, throughput, and others?
- Is your workload steady, or does it peak? If it peaks, for which particular components?

Perform calculations based on the answers to determine the amount of hardware your infrastructure needs.

To size your infrastructure and choose the hardware you need, you can use the rPerf, Transaction Processing Performance Council (TPC), or Standard Performance Evaluation Corporation (SPEC) benchmark results. Run a simple, common workload on several platforms to give you an idea of the performance of the different systems. These reports, your experience, and your application inputs can help you decide.

The IBM Workload Estimator tool helps you to size your infrastructure. For more information, see the IBM Systems Workload Estimator page at:

http://www.ibm.com/systems/support/tools/estimator/index.html

If you need a more accurate estimation of your hardware requirements and you already have your application, you can run a benchmark. Before you start the production, validate the sizing with a performance test campaign.

For more information about benchmarks, see the following websites:

- rPerf

  http://www.ibm.com/systems/power/hardware/notices/rperf.html
- SPEC

  http://www.spec.org/benchmarks.html
- TPC

  http://www.tpc.org/information/benchmarks.asp

7.4.2 Sizing dynamic infrastructures

In a dynamic cluster, the application placement controller needs sizing and application placement decisions based on service policies. A service policy allows workloads to be classified, prioritized, and intelligently routed, which makes a manual sizing for individual resources unnecessary.

In a dynamic cluster, all resources that run in the applications are virtualized. They are then created, started, and stopped as needed for the current workload. To estimate the resources needed for a new application, use the techniques described in 7.4.1, “Sizing static infrastructures” on page 170. After the application is running on the performance test or production environment, the health controller can help you to decide whether the estimation was correct.

The health controller can run predefined and custom corrective actions if the defined health conditions for an application are not met. You can also define email notifications that are sent out when health conditions warrant.

7.5 Monitoring

Because most WebSphere technology-based applications are web-based applications, 24×7 availability is essential. The tolerance of Internet users for unavailable sites is low. They usually navigate to the next site if your site is not operable, meaning you lose potential customers. Therefore, track and monitor the availability of your site so that you recognize when things are going wrong and can react in a timely manner.

Efficient monitoring combined with a sophisticated alerting and problem handling procedure can increase the availability of your service significantly. Therefore, you must plan for monitoring and problem handling. Do not wait until your environment becomes unproductive.

7.5.1 Environment analysis for monitoring

Careful planning for monitoring is essential, and must start with a detailed analysis of the environment to be monitored. Ensure that the full environment is monitored and that no component is overlooked.

To analyze the monitoring requirements for your environment, consider the factors in the following sections to give you an overview of what needs to be done.

Components to be monitored

Each component that is required to run your service must be monitored. For each component that you identify, answer the following questions:

- What are the possible states of the component, and how can you retrieve them?
- What is the impact of each of the possible states that the component can have?
- What specific attributes of the component can be monitored?
- For each attribute that you can monitor, define the following values:
  - Which attribute values (or range of attribute values) show a normal status of the component?
  - Which attribute values (or range of attribute values) show a situation that requires the administrator's attention (warning level)?
  - Which attribute values (or range of attribute values) show a critical condition for the component and require immediate administrator action (alert)?

Prioritize the monitoring results of each component, and define the actions to be taken.

Monitoring software

Providing efficient 24×7 monitoring requires monitoring software. Many organizations have some monitoring infrastructure already set. Determine whether you can integrate a new WebSphere Application Server infrastructure with the existing monitoring infrastructure.

Monitoring agents

Depending on the monitoring software in use, monitoring agents for certain components might be available. Otherwise, most monitoring software provides some scripting interfaces that allow you to write your own scripts. The scripts check and produce output of the results that the monitoring software can analyze.

Infrastructure requirements

When running monitoring in your environment, you need to plan for additional resources. Monitoring affects almost all aspects of your environment. Monitoring requires memory, processor cycles, and network communications. It might even require separate, additional systems for gateways (or as server systems) for the monitoring solution. Ensure that all of the nonfunctional requirements for your infrastructure are also applied to these systems.

Monitoring levels

Monitoring must be in place in all layers of the infrastructure. You must ensure a comprehensive monitoring of the environment. You will likely end up with multiple monitoring tasks and solutions for different purposes.

Network monitoring

Network monitoring covers all networking infrastructure such as switches, firewalls, and routers. It must also monitor the availability of all the communication paths, including redundant communication paths.

Operating system monitoring

Most monitoring solutions provide monitoring capabilities for supported operating systems. By using these features, you can track the health of your environment from the operating system perspective. You can also monitor components such as processor use, memory use, file systems, and processes.

Middleware components monitoring

When using middleware components such as application servers and databases, monitoring on the operating system level is not sufficient. This is because the operating system has no knowledge of the middleware state. You need specific monitoring to the middleware that provides the runtime environment for your application. WebSphere Application Server provides various interfaces that allow the monitoring of your application server infrastructure. Many monitoring products, such as the IBM Tivoli Composite Application Monitoring suite, support these interfaces. They provide ready-to-use agents to monitor your WebSphere Application Server environment.

Transaction monitoring

The purpose of transaction monitoring is to monitor the environment from the user perspective. Transaction monitoring uses prerecorded transactions or click sequences, and replays them whereby the response for each replayed user interaction is verified against expected results.

7.5.2 Performance and fault tolerance

Keep in mind that monitoring your environment (no matter at which level) consumes additional resources. Ensure that your monitoring setup does not cause an unacceptable effect on your environment.

The more you monitor, and the shorter the intervals between your monitoring cycles, the quicker you can determine when something is out of the ordinary. However, this setup also consumes more processor resources. The key to success is to find a good balance between monitoring in sufficient short intervals to determine failures without consuming an unacceptable amount of resources.

In addition to the performance impact, make sure that any problems in your monitoring infrastructure do not affect your environment. Even if something is wrong in the monitoring infrastructure, monitoring must never be the cause for a service outage.

7.5.3 Alerting and problem resolution

Monitoring alone is not enough to monitor the health of your environment, because monitoring does not solve issues. You improve availability if you combine monitoring with appropriate alerting to the responsible problem resolvers. What is the use of monitoring if nobody knows that there is a problem? Consider the following questions when planning for alerting:

- Who is alerted for which event?
- What are the required response times?
- How will the responsible persons be alerted?
- How will you avoid repeated alerts for the same events?
- How will alerts and the resolution of the alerts be documented?
- Who will track the alerts and problem resolution?
- Who is in charge of the alert until it is finally resolved?
- Who will perform the root cause analysis to avoid reoccurrences of the alert?

Alerting is just a first part of your incident and problem management.

7.5.4 Testing

As with each component in your environment, do not forget to test your monitoring infrastructure regularly. If the implementation is new, test every monitoring alert, and ensure that your monitoring detects each condition of your system properly.

Do not stop your testing when you see a monitoring situation raised. Test the whole process, including alerting and incident management, and ensure that conditions are reset automatically as soon as the situation is back to normal.

7.6 Backup and recovery

In general, IT hardware and software are reliable. However, failures can occur that can damage a system, network device, software product, configuration, or more importantly, business data. Do not underestimate the risk of a human error that might lead to damage. It is important to plan for such occurrences.

Planning for recovery is a complex task, and requires end-to-end planning for all components of your infrastructure. For each component, you can have several solutions. Creating a backup and recovery plan entails several stages as explained in the following sections.

7.6.1 Risk analysis

The first step to creating a backup and recovery plan is to complete a comprehensive risk analysis. The goal is to discover which areas are the most critical and which hold the greatest risk. Identify which business processes are the most important and prioritize them accordingly.

For each infrastructure component, consider the following important points when planning for disaster recovery:

- Recovery time objective (RTO)

  How much time can pass before the failed component must be up and running?

- Recovery point objective (RPO)

  How much data loss is affordable? The RPO sets the interval for which no data backup can be provided. If no data loss can be afforded, the RPO is zero.

The best approach is to classify components by risk level, and then decide which backup or recovery techniques to use.

7.6.2 Recovery strategy

After you identify critical areas, develop a strategy for recovering those areas. Numerous backup and recovery strategies are available that vary in recovery time and cost. In most cases, the cost increases as the recovery time decreases.

The key to the appropriate strategy is to find the correct balance between recovery time and cost. The business impact is the determining factor in finding the correct balance. Business-critical processes need quick recovery time to minimize business losses. Therefore, the recovery costs are greater.

7.6.3 Backup plan

Based on your recovery strategy, a backup plan needs to be created to handle the daily backup operations. The backup plan is your strategy to save important data from your infrastructure so that you can restore it if a problem occurs.

Numerous backup methods are available that vary in cost and effectiveness:

- At the global infrastructure level:
  - For vital applications, a *hot backup site* provides real-time recovery by automatically switching to a whole new environment quickly and efficiently.
  - For less critical applications, *warm* and *cold backup sites* can be used. These sites are similar to hot backup sites, but are less costly and effective.
- At the component layer:
  - For vital components, a *hot backup* (also known as an *active* or *dynamic backup*) provides real-time backup without stopping the processes. It is useful for databases when you are unable to stop. Try to plan this backup during low working hours to avoid disturbing users.
  - For less critical components, a *cold backup* can be used. To take the backup, you need to stop the applications and processes.

More commonly, sites use a combination of backup, load balancing, and high availability to maintain the service available.

Other common backup strategies combine replication, shadowing, incremental, and remote backup, with more mundane methods such as tape backup or Redundant Array of Independent Disks (RAID) technologies. All methods are as viable as a hot backup site but require longer restore times.

Any physical backup must be stored at a remote location to recover from a disaster. New technologies make remote electronic vaulting a viable alternative to physical backups. Many third-party vendors offer this service.

A simple backup plan, for example, for a WebSphere Application Server high available infrastructure, can be composed of two HTTP servers, two WebSphere Application Server nodes, and one database instance. The two web servers load balance the load on a WebSphere cluster composed of several JVM spreads on the two nodes.

In this backup plan, the following components are backed up:

- The HTTP server configuration is backed up one time each week, and one month of backup is kept on remote storage. After one month, the backups are archived on tape.

  In this scenario, one of the IBM HTTP servers is stopped, the configuration repository is saved in the remote storage, and the server is restarted. Then the same operation is run for the second server.
- The WebSphere configuration and applications are backed up one time each week, and one month of backup is kept on remote storage. After one month, the backups are archived on tape.

  WebSphere Application Server provides the **`backupConfig`** command to back up the configuration online. Remember to also copy the applications from the applications directory.
- The database is backed up every night with a backup online, and three days of backup are kept on remote storage. After two weeks, the backups are archived on another storage box that consists of low performance disks.

This solution is not apparent to the users due to the load balancing and the backup online.

7.6.4 Recovery plan

The recovery plan must be coordinated with the backup plan to ensure that the recovery happens smoothly. The recovery plan consists of a group of procedures. These procedures allow for recovery to an operational state in a minimum amount of time, regardless of the situation.

7.6.5 Update and test process

You must revise the backup and recovery plan on a regular basis to ensure that the recovery plan meets your current needs. You also must test the plan several times a year to ensure that the technologies are functional and that the team involved knows their responsibilities.

In addition to these regular scheduled reviews, review the backup and recovery plan whenever you change your infrastructure.

7.7 Cloud infrastructure

WebSphere Application Server is also available in both public and private clouds.

7.7.1 Public cloud

A public cloud is offered as a service for companies. Customers do not need to manage the whole infrastructure stack, but they do have access to virtual environments. The environments provided by a public cloud are mainly used for development and testing phases by customers.

Using the public cloud in your infrastructure has the following advantages:

- You pay for only what you need. If you need to test new application features for a few weeks that you would need more hardware for, you do not need to buy that hardware.
- You can take advantage of easy and rapid self-provisioning of your WebSphere environment. It takes only a few minutes to get a standard WebSphere image.
- You can reduce the cost of your infrastructure.

Today, WebSphere Application Server is available with the following cloud offerings:

- IBM offers IBM SmartCloud™ Enterprise (see the following web page) as a way to access secure WebSphere environments:

  `http://www.ibm.com/services/us/igs/cloud-development/`
- Amazon offers Amazon Elastic Compute Cloud (EC2), which provides WebSphere Application Server images:

  `http://aws.amazon.com/ec2/`

7.7.2 Private cloud

Contrary to the public cloud, the private cloud is deployed inside the company infrastructure and is managed by the company. IBM provides two products to integrate WebSphere Application Server to a private cloud:

- An appliance called IBM Workload Deployer (previously known as *IBM WebSphere CloudBurst® Appliance*). This hardware appliance provides access to IBM middleware virtual images and patterns. This access allows you to easily, quickly, and repeatedly create application environments that can be securely deployed and managed in a private cloud. The virtual images do not run on the appliance. Instead, they run on hypervisors. You can deploy the images by using the existing topologies or create your own topology. The images are customizable.
- A virtual edition of WebSphere Application Server named *WebSphere Application Server Hypervisor Edition* that runs on top of different hypervisors. It is a virtual image in Open Virtual Machine Format (OVF). The image contains an operating system, WebSphere Application Server binary files, IBM HTTP server binary files, and WebSphere profiles. All the components are preinstalled, configured, and tuned.

With these technologies, you can build and manage a customized WebSphere Application Server infrastructure. For more information, see:

http://www.ibm.com/software/webservers/workload-deployer

8

Topologies

A *topology* describes how the different elements involved in a WebSphere Application Server solution are deployed and interconnected. For a better understanding of the solution, both hardware and software can be depicted in a *topology diagram*.

When choosing the correct topology for your environment, you need to consider several aspects of your business. Keep in mind that choosing the most flexible and scalable topology from the beginning can help make your WebSphere Application Server implementation successful.

This chapter addresses the most widely used topologies according to business size and needs. It can help you understand the different components involved in a topology, and the best way to implement them according to your business needs.

This chapter includes the following sections:

- Terminology
- Topology selection criteria
- Topologies in detail

8.1 Terminology

Before you examine the topologies, become familiar with the terminology that is highlighted in this section. These elements are used in the diagrams that describe each topology later in this chapter.

8.1.1 Load balancers

A *load balancer*, also called an *IP sprayer*, enables horizontal scalability by dispatching TCP/IP traffic among several identically configured servers. Depending on the product that is used for load balancing, different protocols are supported.

In the topologies in this book, the load balancer is implemented by using the Edge Component Load Balancer that is provided with WebSphere Application Server Network Deployment. This component provides load balancing capabilities for the following protocols and any other TCP-based applications:

- FTP
- HTTP
- Internet Message Access Protocol (IMAP)
- Network News Transfer Protocol (NNTP)
- Post Office Protocol Version 3 (POP3)
- Secure Sockets Layer (SSL)
- Session Initiation Protocol (SIP)
- Simple Mail Transfer Protocol (SMTP)
- Telnet

The load balancer that is included in the WebSphere Edge Components provides the following capabilities:

- Client-to-server affinity
- Easy integration
- Efficient use of equipment
- High availability
- Low processor usage
- Load balancing of a private network
- Scalability

For more information about these features and the functions of Load Balancer, see:

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=%2Fcom.ibm.websphere.edge.doc%2Flb%2Fcovr_features.html

Remember: On the z/OS platform, the Sysplex Distributor provides intelligent load balancing. It balances incoming requests based on real-time information about whether the possible members achieve their performance goals. The member with the best performance rating processes the incoming request.

8.1.2 Reverse proxies

The purpose of a *reverse proxy* is to intercept client requests, retrieve the requested information from the content servers, and deliver the content back to the client. Caching proxies provide an additional layer of security that hides the servers from the clients. The caching proxy products provided by WebSphere Application Server V8.5 are the *stabilized* Edge Component caching proxy, the DMZ secure proxy, and the WebSphere proxy server.

These products provide the capability to store cacheable content in a local cache. Subsequent requests for the same content can be served out of this cache. This configuration allows faster response and decreases the load on the servers and the internal network.

Stabilized means that no new features will be delivered, but new platforms will be supported.

Edge Component caching proxy

The caching proxy provided with WebSphere Application Server V8.5 with the Edge Components can be configured as a reverse and a forwarding proxy. This proxy server supports the HTTP, HTTPS, FTP, and Gopher protocols.

DMZ secure proxy

With *DMZ secure proxy*, you can install proxy servers in the DMZ with reduced security risk when compared to installing an application server to host a proxy server. This approach is achieved by removing all the features from the application server that are not required to provide the proxy function. For example, a DMZ secure proxy does not have a web container or EJB container.

The DMZ Secure Proxy Server supports the HTTP and SIP protocols with and without encryption. To implement a DMZ secure proxy, you must install the DMZ secure proxy product and create a profile by using the *secureproxy* profile template.

Important: Do not confuse the DMZ secure proxy with the WebSphere Application Server Proxy that you can configure in a WebSphere Application Server Network Deployment manager cell.

For the following features, you can select levels of security (low, medium, or high). You can also customize these features based on your requirements:

- Startup user permissions

  A privileged or unprivileged user can run this feature.

- Routing considerations

  Requests can be routed based on static or dynamic information. A high level cannot be used with SIP proxy servers because static routing is not supported for that server type.

- Administration options

  Remote or local administration is possible.

- Error handling

  Custom error pages can be used for specific error codes or groups of error codes.

Remember: You can also switch from a Java Development Kit (JDK) to a Java runtime environment (JRE) when using the DMZ secure proxy. Using a JRE prevents a compiler from being included with the installation, which could be used for malicious purposes if a security breach occurs.

WebSphere Application Server Proxy

WebSphere Application Server Proxy is a proxy server that you configure in a WebSphere Application Server Network Deployment cell. This proxy runs inside the secure zone of the network as an application server. It has access to cell information, and the current state of all servers and applications inside the cell.

8.1.3 Domain and protocol firewall

A *firewall* is a system that manages the flow of information between networking zones such as the Internet and the private network of an organization. Firewalls can prevent unauthorized Internet users from accessing services on private networks that are connected to the Internet, especially intranets. In addition, firewalls can block some virus attacks that must cross the network boundaries protected by the firewall. Another typical usage of firewalls is to prevent denial-of-service (DoS) attacks.

A firewall can separate two or more parts of a local network to control data exchange between departments, network zones, and security domains. Components of firewalls include filters or screens, each of which controls the transmission of certain classes of traffic. Firewalls provide the first line of defense for protecting private information. Comprehensive security systems combine firewalls with encryption and other complementary services, such as content filtering and intrusion detection.

Firewalls control access from a less trusted network to a more trusted network. Traditional firewall services include the following implementations:

- Screening routers (the protocol firewall)

  These routers prevent unauthorized access from the Internet to the DMZ. The role of this node is to provide Internet traffic access only on certain ports, and to block other IP ports.

- Application gateways (the domain firewall)

  Application gateways prevent unauthorized access from the DMZ to an internal network. The firewall allows the network traffic that originates from the DMZ and not from the Internet. It also provides some filtering from the intranet to the DMZ. A pair of firewall nodes provides increased levels of protection at the expense of increased computing resource requirements. The protocol firewall is typically implemented as an IP router.

8.1.4 Web servers and WebSphere Application Server plug-in

Most WebSphere Application Server topologies have a web server that receives HTTP requests from clients. For security reasons, place the web server in a separate network zone secured by firewalls (a DMZ).

Usually the web server, along with the WebSphere Application Server plug-in, provides the following functions in the topology:

- It serves requests for static HTTP content such as HTML files and images.
- Requests for dynamic content are forwarded to the appropriate WebSphere Application Server through the WebSphere Application Server plug-in. This dynamic content includes JavaServer Pages (JSPs), servlets, and portlets.
- The WebSphere plug-in offers load balancing and failover functions between the application servers in a static cluster. It uses a round-robin or random policy to load balance the requests. It can also detect failed application servers and stop sending requests to them until they can handle requests again.
- It allows caching of response fragments by using the Edge Side Include (ESI) cache.
- It is the endpoint of the SSL connection from the client (unless the break is done by another device in the architecture). Optionally it opens a separate secured connection from the web server to the web container on the application server system.

WebSphere Application Server comes with web server plug-ins for all supported web servers. For more information about the web servers that are supported by Websphere Application Server V8.5, see:

http://www.ibm.com/support/docview.wss?uid=swg27021246

The plug-in uses a configuration file (the `plugin-cfg.xml` file) that contains settings that describe how to pass requests to the application server. The configuration file is generated on the application server. Each time a change on the application server affects the request routing of requests, the plug-in must be regenerated and propagated to the web server again.

If the plug-in configuration service is enabled (which is the default), a plug-in configuration file is automatically generated for a web server when any of the following events occur:

- The WebSphere Application Server administrator defines a new web server.
- An application is deployed to an application server.
- An application is uninstalled.
- A virtual host definition is updated and saved.

Restriction: In a stand-alone topology, only unmanaged web servers are possible, meaning that the plug-in must be pushed out manually to the web server system. However, if you are using IBM HTTP Server, the application server can propagate the plug-in configuration file automatically to IBM HTTP Server by using the administrative instance of IBM HTTP Server. This process occurs even if it is an unmanaged node,

WebSphere Application Server V8.5 ships with IBM HTTP Server V8.5 on distributed platforms and on z/OS, which is based on Apache 2.2.8 plus its additional fixes. New features have been added for Global Security Kit (GSKit), IKEYMAN utility, and web server hardening. This version also includes the ability to use 64-bit addressing mode for new platforms.

For more information about what is new in IBM HTTP Server V8.5, see:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=tins_webserver

8.1.5 On-demand routers

An *on-demand router* is a Java-based HTTP Proxy and stateless SIP Proxy built on the WebSphere run time. It is asynchronous, highly available, and scalable. The on-demand router is aware of the state of the application servers and applications. It can route requests according to this state by using routing policies.

For more information about the on-demand router, see 5.3, “Intelligent routing and dynamic operations” on page 116.

Important: Do not place the on-demand router in the DMZ.

8.1.6 Application servers

Application servers are the heart of your topology. This layer in the architecture provides the runtime environment for your Java Platform, Enterprise Edition (Java EE) applications.

To provide all the flexibility and functions offered by WebSphere Application Server, various profile types are available. Some of the profile types are for management purposes only. Others are required to process user requests at run time. The management-related

components of the runtime environment are implemented through specific application servers with predefined names. These application servers are created for you when creating certain profiles. For your topology, you must consider which of these management servers are needed and where to place them.

Application servers that run your applications can be grouped into static or dynamic clusters for workload balancing, failover, and scalability purposes.

8.1.7 Directory and security services

Directory and security services supply information about the location, capabilities, and attributes of resources and users known to this WebSphere Application Server environment. This information includes user ID and password pairs, and certificates. This node can supply information for various security services (authentication and authorization), and can run the actual security processing such as verifying certificates.

An example of a product that provides directory services is IBM Tivoli Directory Server, which is included in WebSphere Application Server Network Deployment.

8.1.8 Messaging infrastructure

WebSphere Application Server can connect to and use an existing *messaging infrastructure*, or it can provide its own infrastructure for messaging through embedded messaging. The messaging service of the default messaging provider in WebSphere can run in any user-created application server.

For more information, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=welc6tech_msg_intro

8.1.9 Data layer

The *data layer* in the topology refers to various back-end resources that hold real business data and logic for the enterprise. The enterprise applications that run on WebSphere Application Server access these resources to build responses for the users and to update data based on user input. The data layer can be a database, an enterprise information system (EIS), or a transaction monitor such as CICS and a web service.

8.2 Topology selection criteria

Regardless of the size of your business, choose the topology selection and include the correct people during planning. Remember to include WebSphere Application Server specialists and, depending on your environment, security, networking, or hardware specialists, as well as developers. These specialists can provide valuable feedback about the correct topology.

This section provides an overview of the important criteria for selecting a topology, and includes the following sections:

- Simplicity
- High availability
- Disaster recovery

- Security
- Performance
- Scalability
- Manageability
- Application deployment
- Summary of topology selection criteria

8.2.1 Simplicity

You can use the WebSphere Application Server Liberty profile to create simple servers. The Liberty profile run time is distributed along with the applications and server configuration, making the distribution and running of these servers simpler. In addition, no cell and cluster levels are required.

Using the WebSphere job manager, you can manage multiple Liberty profiles from a single system. If you create more than one server with the same application and have an HTTP server in front of them, you can achieve high availability and failover ability. In this scenario, the administrator creates procedures that ensure the same application version is installed on all Liberty profile servers. For more information, see 8.3.3, “Liberty profiles managed by a job manager” on page 202. In addition, creating job manager target groups can help with this task.

8.2.2 High availability

High availability means that a system can continue to process work within one location after routine failures. High Availability planning assumes a single failure, with a goal of brief disruptions for only some users during unplanned outages. Depending on which component of the topology failed, the system might continue to run with degraded performance. A simple scenario is an application that is running on a cluster of two application servers and that is deployed on two physical nodes. If the hardware that hosts one of the nodes fails, the application continues to run, but the cluster can process only half the workload it normally does. Remember this scenario to ensure that your high availability design fulfills your business service level agreement (SLA). High availability applies to WebSphere Application Server and to all the components that are required to serve application requests.

High availability is achieved by introducing redundancy in your architecture to be fault tolerant. You need redundancy at different levels, depending on your availability requirements. For example, you might need additional power supplies, network cables, switches, processes, and systems. WebSphere Application Server Network Deployment has various options to provide a highly available runtime environment for your applications. Often part of planning for high availability is a goal of continuous operations, in which the system is never unavailable during planned activities. For example, when upgrading the application to a new version, you might want to perform the upgrade to avoid all downtime. The following sections explain many of the high availability features of Websphere Application Server and how you can benefit from them.

Avoidance of single points of failure

To avoid a single point of failure (SPOF) and to maximize system availability, the topology must have a degree of redundancy. The common way to achieve this redundancy with WebSphere Application Server is through horizontal and vertical scaling. For more information, see8.2.6, “Scalability” on page 190. However, systems often depend on other external systems that are beyond your control. In this situation, consider alternatives for getting the data that these systems provide, if possible, in case they are out of service. This approach can improve the overall availability of the application and serve client requests more consistently.

You can avoid a single point of failure by using either of the following approaches:

- Hardware redundancy
  - Use horizontal scaling to distribute application servers (and applications) across multiple physical machines or z/OS images. If a hardware or process failure occurs, clustered application servers can handle client requests. Additional web servers and IP sprayers can also be included in horizontal scaling to provide higher availability.
  - Use backup servers for databases, web servers, IP sprayers, and other important resources, ensuring that they remain available if a hardware or process failure occurs. Keep the servers (physical machines) within the cluster sprayed in different secured rooms to prevent site-related problems.
  - Use virtualization to get the systems back if a hardware failure occurs. The advantage of this approach is that snapshots of the operating system can be taken. If a hardware failure occurs, those snapshots can be restored on alternate hardware. Depending on your business needs and storage capabilities, snapshots can be taken daily to ensure current backups. Virtualization also helps to manage workloads more efficiently, improving application flexibility, availability, and performance.

    For more information about how virtualization can help speed up the deployment process, see “Using virtual image templates to deploy WebSphere Application Server” on IBM developerWorks at:

    http://www.ibm.com/developerworks/websphere/techjournal/0705_willenborg/0705_willenborg.html
- Process redundancy
  - Use horizontal scaling, placing application servers on different systems.
  - Use vertical scaling for process isolation as related to WebSphere processes only. In this case, a failing server does not affect the remaining healthy servers. Furthermore, with this method, you can take maximum advantage of the resources that are available on the server.
  - Deploy the web server on a different system than the application servers. This configuration ensures that problems with the application servers do not affect the web server and vice versa. Separate systems also increase the security level.

Load balancing

Use load balancing techniques to ensure that individual servers are not overwhelmed with client requests while other servers are idle. Load balancing can also help avoid bottlenecks in the topology. Load balancing includes the following techniques:

- Use an IP sprayer to distribute requests across web servers in the configuration.
- Direct requests for high-volume URLs to more powerful servers.

The Edge Components included with WebSphere Application Server Network Deployment provide these features.

Important: For Edge Component Load Balancer to stop distributing a load across nonresponsive web servers, you must configure the Advisor feature. This feature is not enabled by default after the product is installed.

WebSphere Application Server also provides the following load balancing mechanisms:

- The HTTP server plug-in in Websphere Application Server to spread requests across cluster members. Correct tuning of the plug-in can help detect failures or problems in the application servers more efficiently.
- The on-demand router to manage the workload across application servers in a dynamic cluster.
- The Enterprise JavaBeans (EJB) workload management mechanism, which is built into WebSphere Application Server to balance EJB workload across cluster members.
- The use of partitioned queues. If the application allows, you can configure partitioned queues to split message processing workload across multiple servers.

Failover support

The environment must be able to continue processing client requests, even if one or more components are offline. To take maximum advantage of this technique, the failover process must be automatic when possible.

The following methods can provide failover support:

- Use horizontal scaling with workload management to take advantage of failover support.
- Use an IP sprayer to distribute requests across web servers in a configuration.
- Use HTTP server plug-in support to distribute client requests among application servers.
- Use the on-demand router to distribute client requests across application servers in a dynamic cluster.
- Use EJB workload management to realign EJB requests if an application server goes down.
- Use the high availability manager in WebSphere to provide failover support of critical services, the singletons, such as messaging engines and transaction service.
- Use an optimized local adapter to specify an alternate connection factory Java Naming and Directory Interface (JNDI) name in case the primary connection factory fails.
- Use external high availability frameworks and service integration to specify alternate connection names for the link sender channel of WebSphere MQ. If a failure occurs in the active gateway queue, the bus reconnects to a standby gateway queue manager by using this information.
- Use resource workload routing to fail over resources, such as data sources and connection factories, and then fail back from previously defined backup resources. The backup resource must be compatible with all applications that use the primary resource. Resource workload routing is created the same way that the primary resource is, but applications can use it only when the primary resource is not active. You must test the suitability of this feature in your environment before enabling failover support.

Operating system-based clustering

The high availability framework of WebSphere Application Server provides integration into an environment that uses other high availability frameworks. This method provides high availability for resources for which WebSphere does not provide specific high availability functions. The other high availability frameworks include operating system-based clustering software, such as IBM PowerHA® on AIX, Parallel Sysplex on z/OS, and Windows Server Failover Clustering for Windows. Consider such a technique for WebSphere Application Server components such as a deployment manager and single server environments.

8.2.3 Disaster recovery

Disaster recovery is the reconstruction of the physical production site in an alternate physical site, occurring after the loss of a primary data center. Disaster recovery is the process of bringing up servers and applications, in priority order, to support the business from the alternate site. When planning for disaster recovery, keep in mind the following considerations:

- How to start a fully operative environment after a disaster strikes the system
- How much data loss you can afford
- How you ensure that data remains consistent

Split-brain isolation is a potential threat to data consistency to avoid in all circumstances. After data consistency issues are resolved (problems with WebSphere data, such as the configuration repository and logs), you can resume planning for WebSphere disaster recovery.

Explanation: In this context, *split-brain isolation* refers to a condition where each member of a clustered system considers the other member to be gone. In this case, both take over the service. The result is that two systems are manipulating data, creating inconsistency.

One possible scenario is the simultaneous failure of network links between nodes in a cluster when all the nodes in the cluster are still running. If this scenario occurs, each node in the cluster might mistakenly decide that every other node is down. Each node then attempts to start or recover services that other nodes are still running. This process can result in data corruption on the shared storage because network connectivity is required to maintain data consistency.

No common solution exists in a disaster recovery scenario because it depends on the existing environment, requirements, applications, and budget. Avoid running a cell across different data centers, because this approach can cause split-brain isolation, compromise data integrity, add operational complexity, and degrade performance. Also, a deployment with a cell that spans data centers depends heavily on network reliability.

You can find a discussion of the issues that can arise when a cell spans data centers at:

http://www.ibm.com/developerworks/websphere/techjournal/0606_col_alcott/0606_col_alcott.html#sec1d

If both data centers do not depend directly on each other and the application can work on both sites without sharing data, an alternative approach is to load balance the workload between the data centers.

For more information about these approaches, see:

http://www.ibm.com/developerworks/websphere/techjournal/0707_col_alcott/0707_col_alcott.html

8.2.4 Security

Security is a critical consideration when designing a new system. Its objective is to protect the different components that can give access to the most valuable enterprise resource, the information. Place security controls in every layer of your topology, and plan how to protect every element in that layer.

You must also comply with certain regulations, which vary by industry. In this case, ensure that the people who know the regulations in-depth are also involved during the planning phase.

Security is a vast topic but can be thought of in two categories:

- Physical security

  Physical security refers to protection against physical threats, such as controlling physical access to systems and protecting the environment of the systems.

- Logical security

  Logical security is connected to a specific IT solution, architecture, and application design. It deals with all aspects of access to runtime resources and data.

Consider the three-tier architecture as an option for your topology design as illustrated in Figure 8-1. This architecture offers the benefit that, if a security breach occurs in one of the tiers, only that level is compromised. Only the necessary ports can be opened between layers that exchange information. This way, the information flows from one level to the other in a controlled manner.

Usually, web servers in the first tier are protected by one firewall that filters data from the outside network. They are usually also protected by another firewall that filters information that the web servers forward to application servers in the second tier. This concept is known as a *DMZ*. The primary objective of the DMZ is to protect sensitive business logic or information hosted in the application servers or databases. It shields this data from possible attacks from untrusted networks on the Internet.

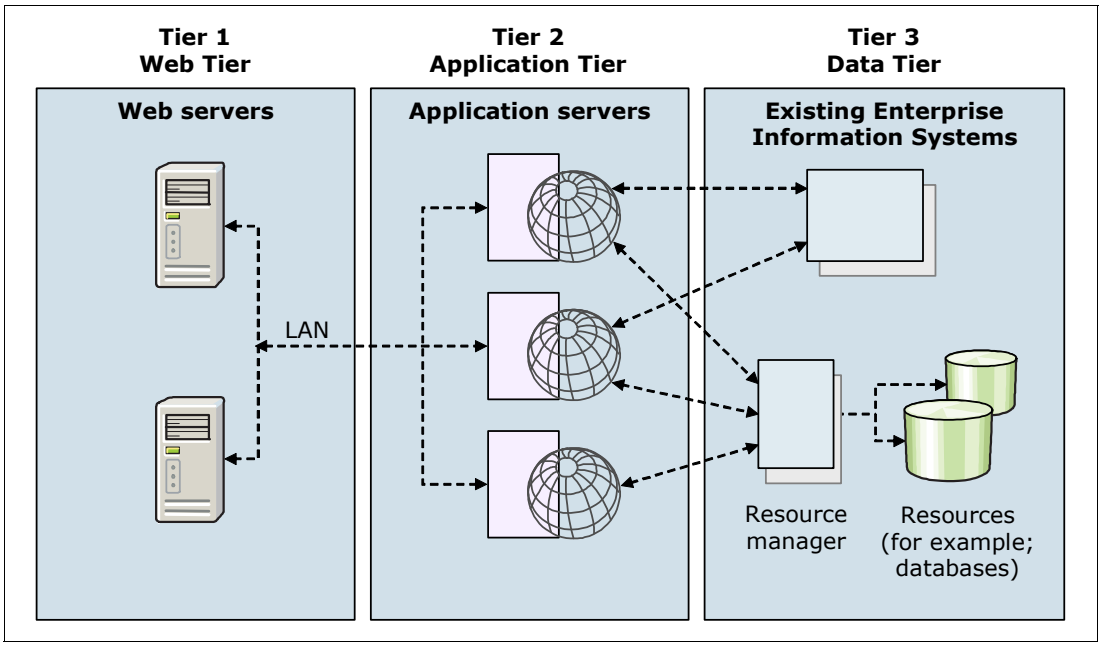


Figure 8-1 Three tier topology

For more information, see Chapter 15, “Security” on page 469.

8.2.5 Performance

Performance determines the ability of an environment to process work in an interval. The higher the performance is, the smaller the interval is needed to process a specified set of work. This smaller interval allows more work to be processed in the same amount of time.

Choosing the correct topology can help in getting good performance from your system. The most important aspect when planning for performance is to ensure that the applications that run on your topology are developed by following preferred practices for performance. If you have poorly designed applications with inefficient code or leaks, it is unlikely that your topology can counteract your performance problem.

When planning for performance, the following metrics are widely used:

- Response time

  The response time metric is a generic approach for a single type of request. It defines the maximum amount of time that a request is allowed to take until it is finished. This metric is most often used in online workloads, where a request must achieve a real-time goal.

  Tip: When using this metric, make sure that the response time is achieved in a single user transaction scenario. It also needs to be achieved when the projected production load is run against the system.

- Throughput

  The throughput metric measures the overall amount of work that is processed in a certain amount of time. It is usually used for batch-type workloads that need to be finished in a certain time window.

Always plan your topology to be as simple as possible. A multitier design can offer better throughput for heavy loads. However, keep in mind that a large topology with many layers can cause a performance penalty. This performance loss is due to the network traffic between the layers and the components within them. The same guidance applies for web containers and EJB containers. Having both of them on the same Java virtual machine (JVM) can result in better throughput.

For production environments, consider using an external web server rather than the embedded web server, formerly called *WebContainer Inbound Chain*, that is part of the WebSphere Application Server architecture. By using this approach, the web server can handle the static content, so that you can benefit from the performance benefits. This way, the application server can use its resources for serving dynamic content only. Furthermore, the benefits that are provided by the WebSphere plug-in, such as session affinity and load balancing, can also be used.

Additionally, with WebSphere Application Server Network Deployment, you can cluster application servers so that multiple server instances are running the same application. These instances are then available to handle incoming requests. Clustering generally provides improvements for performance, due to an optimized scaling.

8.2.6 Scalability

Scaling represents the ability of a system to grow as the load grows on a system. You can use scaling to avoid SPOF, to take better advantage of the hardware-free resources, or to improve performance. You can achieve scaling in multiple ways. For example, you can configure

multiple systems to add processing power, improve security, maximize availability, and balance workloads. Scaling can be vertical or horizontal, as illustrated in Figure 8-2.

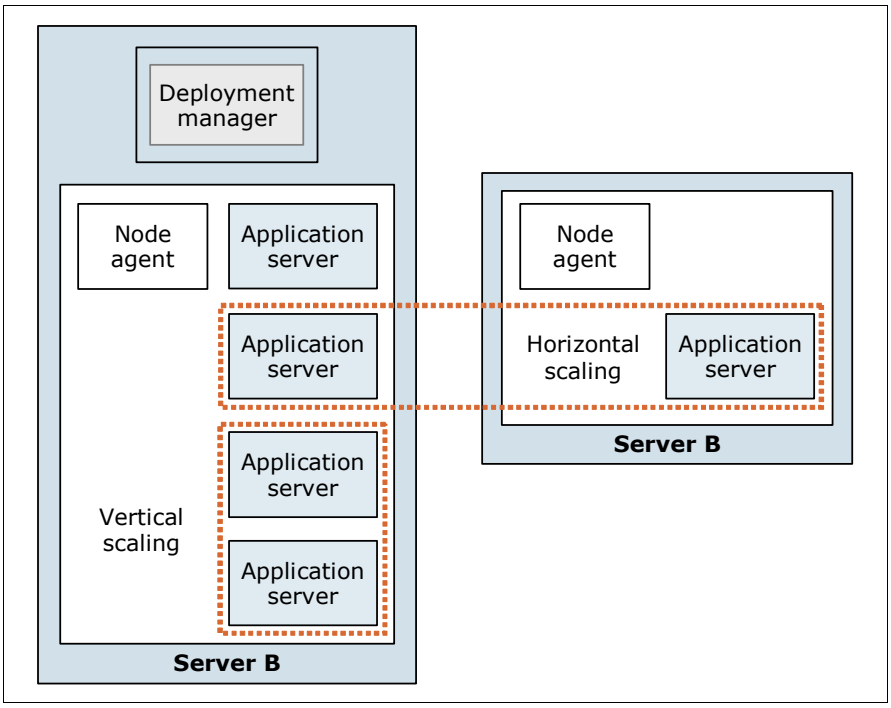


Figure 8-2 Vertical and horizontal scaling with WebSphere Application Server

The method that you use depends on where and how the scaling is taking place:

- Vertical scaling

  Vertical scaling involves creating additional application server processes on a single physical system or z/OS image. This configuration provides application server failover and load balancing across multiple application servers. This topology does not provide efficient fault tolerance. A failure of the operational system or the hardware on the physical system might cause problems for all servers in the cluster.

- Horizontal scaling

  Horizontal scaling involves creating application servers on multiple systems to take advantage of the additional processing capacity available on each system. Using horizontal scaling techniques also provides hardware failover support.

  Tip: When planning horizontal scaling, you must decide whether the response times and throughput must remain the same if one of the nodes fails. If these values must remain the same, consider this factor during hardware sizing for the nodes. Each node needs enough resources available to handle the additional workload that corresponds to the failing node.

When implementing horizontal scalability, using multiple smaller systems can be more cost effective. However, when planning to use vertical scalability, make sure that the systems have enough available resources to host multiple processes. Bigger systems can be more expensive, but vertical scalability takes advantage of the investment in resources.

Although spreading a cluster across two or more nodes improves availability and performance, the application still relies on the back-end systems where the information is obtained. If one of those systems fails, all the application servers in your cluster are affected.

Under such circumstances, it is possible for a “domino effect” to occur, with the back-end problem affecting all related layers. Therefore, consider combining the new features in WebSphere Application Server V8.5, as explained in “Failover support” on page 187, with your scalability schema for a more resilient environment.

Consider the following guidance when planning your scaling strategy:

- Scale in a maintainable manner.

  If your topology grows too complex, it is difficult to apply maintenance work, and you end up with an error-prone environment.

- Try to keep consistent versions of your operating systems and servers.

  Although it is possible to run cell nodes in different operating systems, this approach increases complexity when maintaining the servers. You must also train administrators on how to run different operating systems. For administrators, it is easier to manage log or configuration files on similar directory structures.

- Keep clocks synchronized in all servers in the topology.

  When troubleshooting a problem, synchronization helps to correlate events during the analysis of log files. Also, the synchronization between nodes is not effective if there is a difference of more than 5 minutes between the servers.

The following components also provide functions for configuring scalability:

- Cluster support in WebSphere Application Server

  The use of application server clusters (static and dynamic) can improve the performance of a server, simplify its administration, and enable the use of workload management.

- WebSphere workload management

  You can use the workload management capabilities of Websphere Application Server to distribute requests among converged and EJB containers in clustered application servers. These capabilities enable both load balancing and failover, improving the reliability and scalability of WebSphere applications. On the z/OS platform, the workload management function is tightly integrated with the operating system to take advantage of the superior workload management features of z/OS.

- IP sprayer

  The IP sprayer transparently redirects all incoming HTTP requests from web clients to a group of web servers. The clients behave as though they are communicating directly with one web server. However, the IP sprayer intercepts all those requests and distributes them among all the available web servers in the group. IP sprayers (such as Edge Component Load Balancer or Cisco Local Director) can provide scalability, load balancing, and failover for web servers. They can also provide these services for other TCP/IP-based servers whose protocol is understood by the IP sprayer.

8.2.7 Manageability

WebSphere Application Server V8.5 provides an Intelligent Management feature that you can use to create dynamic virtual clusters. These clusters are created, started, and removed as they are needed by the application server software. Also, the applications are placed on the clusters automatically. All requests to applications on dynamic clusters are distributed by an on-demand router.

Using dynamic clusters and on-demand routers, you can create topologies that adapt themselves to the current workload. Doing so makes these topologies easier to manage, and

provides scalability and high availability. For more information about dynamic clusters, see Chapter 5, “Intelligent Management” on page 107.

A topology that contains virtual clusters can consist of one or more cells. The cells and nodes are created manually, and each cell can create multiple physical nodes. You can define service policies that influence the behavior of the dynamic infrastructure.

To improve the scalability of the dynamic infrastructure, you can create a *multi-cell star topology*. In this topology, you create one cell that runs the on-demand routers, and all traffic is routed through this cell. Then, you create one or more cells that run the applications. With this topology, you can add additional cells and additional physical hardware easily when necessary.

8.2.8 Application deployment

Various application deployment decisions affect topology decisions. Clarify the application deployment-related considerations addressed in this section before you finalize the architecture.

EJB deployment

The way you deploy EJB can significantly impact your application topology and the performance you can expect. You can choose to deploy EJB to the same application servers and clusters as the client modules starting the EJB. You can also deploy EJB to separate dedicated application servers and clusters running the EJB only. Both of these options are valid approaches that depend on your environment and requirements. For the explanation in this section, the EJB provides local and remote interfaces to be started.

Consider deploying your EJB to a different application server or cluster than the EJB clients when you have the following requirements:

- Use the same version of EJB across all enterprise applications.
- Reuse EJB code.
- Require different tuning between enterprise applications and EJB.
- Need a faster start of the application servers that host the web applications. This improved speed is possible because the EJB container is loaded only where the EJB is deployed.
- Optimize the use of the JVM for your web application. Set its resources aside for web requests only, and delegate client application calls to the EJB servers.
- Deploy the web applications or EJB on different systems or platforms to take advantage of the infrastructure on your enterprise.
- Simplify the deployment process of applications and EJB.

However, if any of the following scenarios apply to your environment, consider deploying the EJB and EJB client on the same server:

- Performance issues. Deploying the EJB on a separate server from the EJB client slows performance. This slowdown occurs because much serialization and deserialization must be done when sending the data through the network.
- Inexistent or incipient development methodology occurs because all enterprise applications must be compatible with the existing version of the deployed EJB. This situation requires coordination during the development phase.
- Memory footprint issues exist on your environment. Additional servers for the EJB demand more memory to run.

- A complex cell requires more servers to manage and can make administration less practical.
- Visibility of the deployed applications and how the EJB upgrades might affect them is lacking.

Assignment of applications to clusters

When you are running multiple applications in your environment, decide which of the following actions you want to take:

- Deploy all your applications to the same application servers and clusters
- Set up separate application servers and clusters for each application

If you have the following requirements for your environment, an application per server or cluster can be a valid approach:

- You need to prevent critical applications from being affected by other faulty applications (unless the faulty application is a common component).
- You need easier administration for applications or cluster-related tasks.
- You require specific server tuning according to the application needs.
- You must reduce the time spent in the garbage collection cycles because the heap can be smaller when running just one application.

  Important: The heap size of a JVM is finite. Even when using a 64-bit implementation of WebSphere Application Server, be careful with heap sizing to avoid performance problems during garbage collection.

- You seek benefits from the runtime provisioning capabilities of WebSphere Application Server V8.5.

If the following scenarios apply to your environment, carefully decide whether one application per server can be beneficial to your topology:

- If you have large environments with complex cell administration, a cluster or server per application increases the complexity. It can also increase start times while the high availability managers establish connectivity across the cell.
- You are likely to experience slowed performance when calling EJB. This process is an out-of-process call if the EJB is on a separate application server from the EJB client.
- Memory footprint issues can occur because each JVM has a basic memory footprint, which increases the overall footprint.

Location of the embedded messaging infrastructure

When using the embedded messaging infrastructure of WebSphere Application Server, you must decide in which application servers the messaging service runs. You can run the embedded messaging service on a separate application server and cluster, or co-located on an application server that runs your applications. Your choice depends on your needs.

To determine what is best for your environment, consider carefully whether to run the embedded messaging infrastructure on a separate set of application servers and clusters. Consider locating your embedded messaging infrastructure on different application servers or clusters when you have the following requirements:

- You need to optimize the utilization of your application server JVM running critical applications. You must focus its resources for application requests only, especially if the messaging infrastructure is used by many applications.

- You have different tuning needs between applications and the messaging infrastructure.
- If the messaging infrastructure is heavily used, you want to avoid failover caused by restarting the application servers when configuration changes are made.
- You seek benefits from the runtime provisioning capabilities of WebSphere Application Server V8.5.

If the following scenarios apply to your environment, carefully choose whether locating the message infrastructure on separate servers is beneficial to your topology:

- In large environments with complex cell administration, locating the message infrastructure on separate servers increases its complexity. It can also increase startup times while the high availability managers establish connectivity across the cell.
- Memory footprint issues can occur because each JVM has a basic memory footprint. More application servers increase the overall footprint.
- The deployment can become more complex when using mediation modules.

8.2.9 Summary of topology selection criteria

The following tables list the requirements for topology selection and possible solutions.

Table 8-1 summarizes topology selection based on availability requirements.

Table 8-1 Topology selection based on availability requirements

Requirement = availability	Solution or topology
Web server	Load Balancer (with hot backup) or a comparable high availability solution, based on other products
Application server	► Horizontal scaling (process and hardware redundancy), vertical scaling (process redundancy), or a combination of both ► Multiservant regions on z/OS ► Virtualization ► Hardware clustering for single server environments
Database server	► Database or operating system-based high availability solution ► Data mirroring
User registry	Depends on the user registry in use. WebSphere provides backup support for some user registries, such as Lightweight Directory Access Protocol (LDAP) servers

Table 8-2 summarizes topology selection based on performance requirements.

Table 8-2 Topology selection based on performance requirements

Requirement = performance/throughput	Solution or topology
Web server	► Multiple web servers with Load Balancer ► Caching Proxy Servers with Load Balancer ► Dynamic caching with Edge Side Includes (ESI) ► WebSphere plug-in tuning

Requirement = performance/throughput	Solution or topology
Application server	► Clustering ► Deploy EJB to the same JVM as the starting client ► Dynamic caching at the application server ► Offload of static content to be served from the web server and, therefore, offload of the application servers ► Avoidance of heap sizes that are too large ► Workload management and transaction classes on z/OS to keep response times
Database server	► Separate database server ► Partitioned database servers

Table 8-3 summarizes topology selection based on security requirements.

Table 8-3 Topology selection based on security requirements

Requirement = security	Solution or topology
Web server	► Separate the web server into a DMZ, either on a logical partition (LPAR) or a separate system. ► Use a DMZ secure proxy instead of a web server with WebSphere plug-in. ► Separate administrative traffic from productive traffic.
Application server	► Implement WebSphere Application Server security. Consider Java 2 security to restrict application access to local resources if needed. ► Create a separate network tier for the application server. ► Separate the application servers from the database and EIS layer. ► Separate administrative traffic from productive traffic.
Database server	► Use a separate server. ► Consider placing a firewall to improve network security.

Table 8-4 summarizes a topology that uses the Intelligent Management feature.

Table 8-4 Topology using the Intelligent Management feature

Requirement = security	Solution or topology
Web server	► Multiple web servers with Load Balancer ► Caching Proxy Servers with Load Balancer ► Dynamic caching with Edge Side Includes (ESI) ► WebSphere plug-in tuning
Application server	► Define the nodes on your available hardware ► Create one cell that is running the on-demand routers ► Create one or more cells that run applications ► Modify the service policies according to your needs ► Application servers are created as needed
Database server	► Use a separate server ► Partitioned database servers

8.3 Topologies in detail

Because of the vast amount of configuration possibilities, WebSphere Application Server provides many options to fit almost every requirement. This section provides information about basic configuration topologies (that can also be combined), depending on the requirements of your environment.

The topologies in this section can be implemented for production and non-production environments. However, when testing the solution before going into production, the environment where you do the testing must reflect the production environment as closely as possible. In some cases, testing applications on single server environments can have different results when tested on a distributed cell, These results depend on how the applications were developed. Testing in accurate preproduction environments can save you from unexpected results when going live in production.

Considerations:

- With WebSphere Application Server V8.5, you can create profiles graphically by using the Profile Management Tool or the **`manageprofiles`** command. For traceability reasons, the command-based creation is preferred. The samples in this chapter are based on the **`manageprofiles`** command.
- The Profile Management Tool Graphical Interface for 64-bit architectures is available on these platforms:
  - Linux for zSeries
  - x86-based Linux and Windows
  - Linux on Power PC
  - AIX Power PC

  However, you can use the Profile Management Tool Graphical Interface on other 64-bit architectures if you use a WebSphere Application Server 32-bit installation.
- On the z/OS platform, all topologies introduced in this section profit from the workload management capabilities that are offered from the Workload Manager component and WebSphere Application Server for z/OS. With this management, you can set and keep performance-focused SLAs on a transactional level.

  For more information about the workload management capabilities, see 16.1.7, “Workload management for WebSphere Application Server for z/OS” on page 509.

8.3.1 Stand-alone server topology

The topologies in this section all use a web server as a front-end device. The benefits of using a web server are that you do not have to deploy an application server in the DMZ. You can also use it for caching purposes.

Application server

A *stand-alone server topology* refers to the installation of WebSphere Application Server on one single (physical) system or LPAR with one application server only. When implementing such a topology, keep in mind that it does not provide any load balancing or high availability capability.

Tip: A stand-alone application server on the z/OS platform offers some degree of load balancing and high availability for the application itself. WebSphere Application Server for z/OS uses multi-servant regions, which are best thought of as an application cluster to build each application server. Multi-servant regions provide one application image to the user while running multiple, independent instances of the application.

The system administrator can determine whether multiple application images are used. For more information, see 16.1.5, “Structure of an application server” on page 505.

Web server

Although you can install the web server on the same system as WebSphere Application Server, employ a web server in a DMZ as a front-end system to receive requests. The web server in the DMZ provides a secure hardened presence, whereas the application server that contains business logic is securely in a separate network.

Figure 8-3 illustrates a stand-alone topology with a web server in a DMZ.

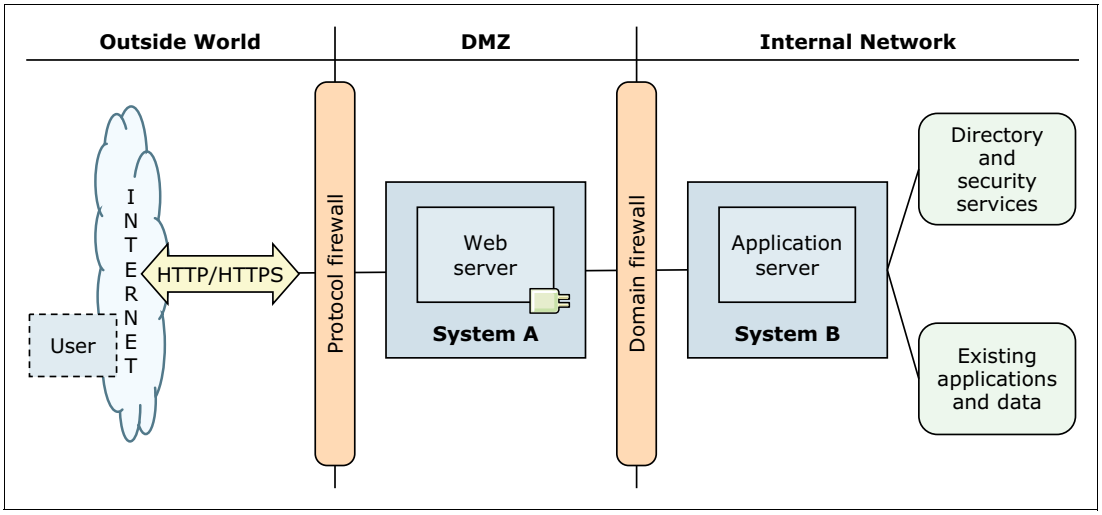


Figure 8-3 Stand-alone server topology with web server in a DMZ

Advantages

The stand-alone server topology has the following advantages:

- You can size and configure servers appropriately for each task.

  By installing components (web server and application server) on separate systems or z/OS images, you can size and configure each task to optimize performance.

- It removes resource contention.

  By installing the web server in a different and physically independent server from the application server, a high load of static requests will not affect the resources available to WebSphere. These resources include processor, memory, and disk. Nor does it affect the ability of WebSphere to service dynamic requests. The web server might have more resources available while serving dynamic content by using other technologies, such as Common Gateway Interface (CGI).

- It increases maintainability due to component independence.

  Server components can be reconfigured or replaced without affecting the installation of other components, because they are on separate systems or LPARs.

- It offers increased security by using a DMZ.

  Isolating the web server in a DMZ protects the business applications and data in the internal network. It does so by restricting access from the public website to the servers and databases where this information is stored. Consider avoiding topologies in which servers in the DMZ have direct access to the database that is storing business or other security-sensitive data.

Considerations

Keep in mind the following considerations when using the stand-alone server topology:

- It can require additional administration for the web server plug-in.

  The plug-in configuration file is generated on the WebSphere Application Server system. You must copy it to the web server system each time a configuration change occurs, which affects requests for applications. Although WebSphere Application Server V8.5 provides tools to automate this step, not every environment is suitable to use these tools.

- Increased communications over the network can cause a possible drop of performance.

  The network capacity and the distance of the web server can limit the network response time for communications between the application server and web server. To prevent having limited response time, ensure that you have adequate network bandwidth between the web server and the application server.

- It requires additional security processing due to SSL communication.

  When using SSL communication from the client to the web server, the communication from the plug-in to the application server must be encrypted. This process prevents sensitive data from being “sniffed” in the network. This additional encryption introduces a performance penalty and increased resource use. Consider configuring the connection from the plug-in to the web container so that the plug-in and web container mutually authenticate each other by using certificates. This approach prevents unauthorized access to the web container.

- It has additional systems to administer.

  Because the web server runs on a separate system, you have one more system to manage and operate, which increases the operation cost of the environment.

Setting up the topology

To set up an environment similar to the one in Figure 8-3 on page 198, install and configure the environment as in the following sections.

Setting up System A

To set up System A, complete these steps:

1. Install IBM Installation Manager.
2. Using IBM Installation Manager, install the following applications:
   - Web server plug-ins for WebSphere Application Server
   - WebSphere Customization Toolbox
   - IBM HTTP Server

     If you are not using IBM HTTP Server, install any other supported web server.
3. Open the WebSphere Customization Toolbox, and start the Web Server Plug-ins Configuration Tool.

4. Configure the web server plug-in and create the web server definition. For details about this task, see the Websphere Application Server V8.5 Information Center at:

   http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=tins_webplugins

Setting up System B

To set up System B, complete these steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server V8.5.
3. Create an application server profile by using the *app_server_root*/profileTemplates/default profile template.
4. Create a web server definition through the administrative console or by running the configureweb_server_name script locally from the profile_root/bin path. This script is on Server A in the plugins_root/bin directory. If just one system is running Windows, the script is in the plugins_root/bin/crossPlatformScripts directory.

> **Remember:** The web server definition is used by the application server to generate the plug-in configuration file. In a stand-alone topology, only one web server can be defined to the configuration, and it must be an unmanaged web server.

8.3.2 Multiple stand-alone servers topology

The multiple stand-alone servers topology is a variant of the stand-alone server topology described in 8.3.1, “Stand-alone server topology” on page 197. The difference is that you have more than one profile on the same system, and every profile has its corresponding web server. Figure 8-4 illustrates the multiple stand-alone servers topology. Notice the one-to-one relationship between the application servers and the web servers.

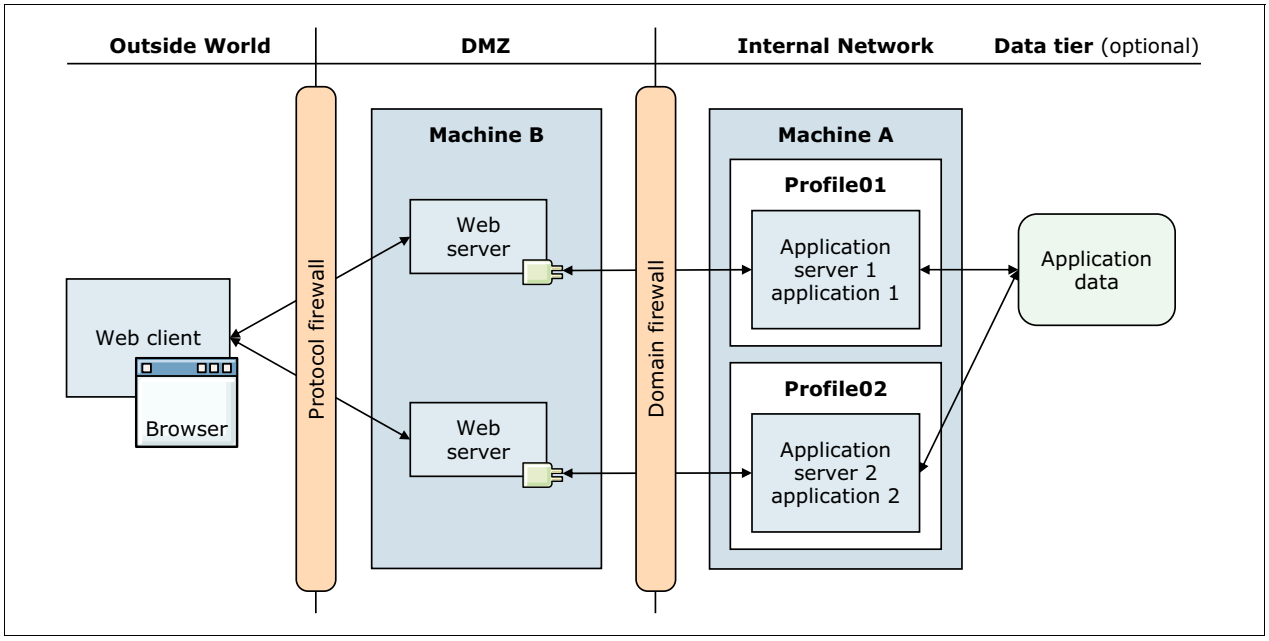


Figure 8-4 Multiple stand-alone servers topology with multiple web servers in a DMZ

Advantages

The multiple stand-alone servers topology offers the following advantages in addition to the advantages offered by a stand-alone server topology:

- Isolation for critical applications

  By having multiple application servers on the same system, critical applications can be deployed on their own server. This configuration prevents such applications from being affected by faulty applications on the same server as can happen in a stand-alone topology.

- Dedicated resources

  Each profile has unique applications, configuration settings, data, and log files, but share the set of core product files. Also, each application has its own JVM that can help customize tuning, depending on the application needs.

- Enhanced serviceability

  Profiles share a single set of product core files. During the update of the product, those files are updated, and, therefore, all of the profiles are updated. Creating profiles is more efficient and less error-prone than full installations on separate servers.

Considerations

Keep in mind the following considerations for a multiple stand-alone servers topology in addition to the considerations for a single stand-alone server topology:

- Additional administration

  If a common component, such as a database, is used by different profiles, the corresponding configurations must be done individually for each profile.

- SPOF

  All web servers and profiles rely on the same hardware or operating system. A failure on any of them makes the system unavailable.

- Any upgrade to the WebSphere binary files impacts all profiles. If one profile needs a certain version of WebSphere, you cannot upgrade the product only for that single profile.

Setting up the topology

To set up an environment similar to the one illustrated in Figure 8-4 on page 200, perform the steps in this section.

Setting up System A

To set up System A, complete these steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install the following applications:
   a. Web server plug-ins for WebSphere Application Server
   b. WebSphere Customization Toolbox
   c. IBM HTTP Server

   If you are not using IBM HTTP Server, install any other supported web server.
3. Open the WebSphere Customization Toolbox, and start the Web Server Plug-ins Configuration Tool.

4. Configure the web server plug-in and create the web server definition. For details about this task, see the Websphere Application Server V8.5 Information Center at:

   http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=tins_webplugins

5. Repeat steps 2c, 3, and 4 to install subsequent IBM HTTP Server instances or other supported web servers.

Setting up System B

To set up System B, complete these steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server V8.5.
3. Create an application server profile by using the *app_server_root*/profileTemplates/default profile template.
4. Create a web server definition through the administrative console or by running the configureweb_server_name script locally from the profile_root/bin path. This script is on Server A in the plugins_root/bin directory. If just one system is running Windows, the script is in the plugins_root/bin/crossPlatformScripts directory.
5. Repeat steps 3 and 4 to create additional profiles, and configure them to use the corresponding web server.

8.3.3 Liberty profiles managed by a job manager

Liberty profiles provide the option to run small footprint servers. You can install multiple applications in one Liberty profile server, and multiple Liberty profile servers can run on the same hardware. The same application can run on multiple Liberty profile servers to achieve a high availability.

To manage the Liberty profiles in your topology, use the WebSphere job manager. It provides a central asynchronous management of all local and remote Liberty profiles. To install a job manager, you need WebSphere Application Server Network Deployment.

The example shown in Figure 8-5 shows using two HTTP servers to avoid a SPOF. You can manage the plug-ins for the HTTP Servers by using the job manager. The job manager can map one application to multiple Liberty profiles.

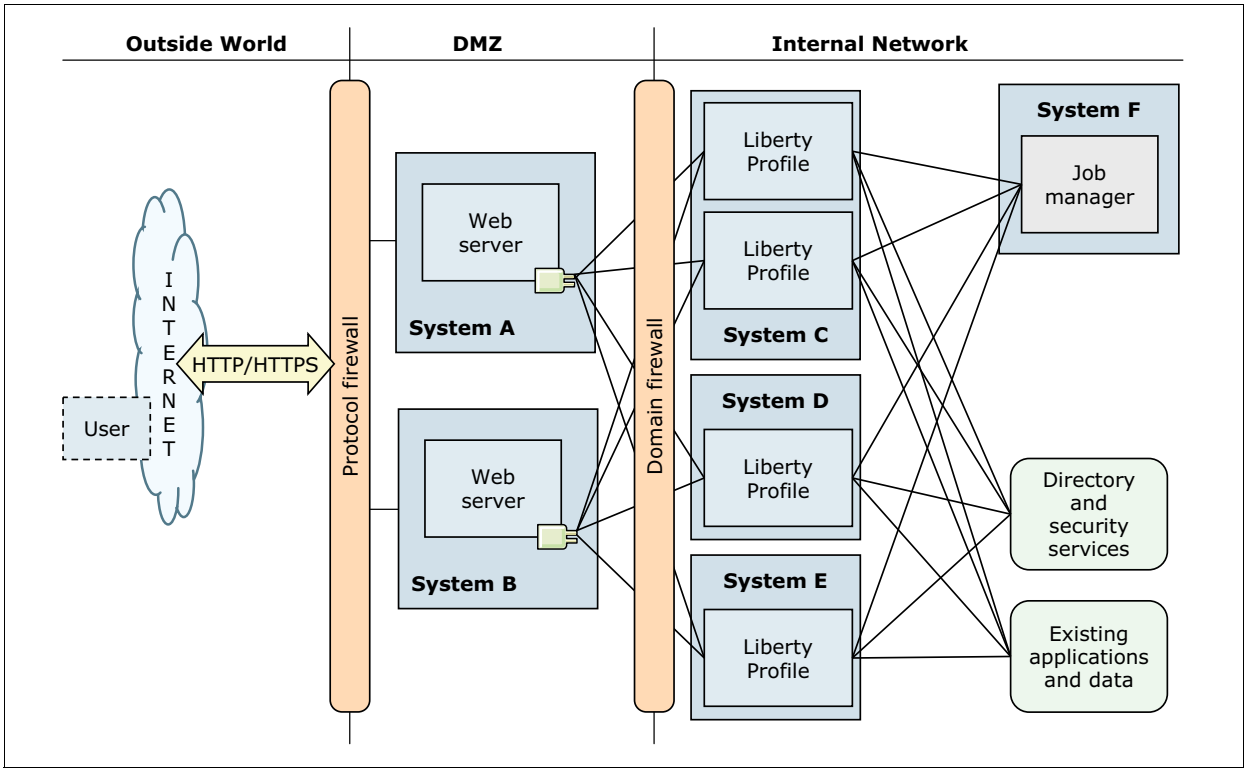


Figure 8-5 Liberty profiles that are controlled by a job manager

Advantages

Using a Liberty profile has the following advantages:

- Easy installation

  You can prepare each Liberty profile as one or more compressed files by using one of the following options:

  - Prepare a file that contains the Java software development kit (SDK), the Liberty profile installation run time, the server definitions, and the applications. This file is self-contained and does not require any preinstalled software on the target host.
  - Prepare a file that contains the Liberty profile installation run time, the server definitions, and the applications. You can install this file on a server that has a Java SDK installed.
  - Prepare a file that contains only the parts that you need, for example only applications, or only applications and server configurations. This method allows you to deploy each part separately as needed.
  - Prepare a compressed file that contains a new version of the Java SDK.

  While deploying the file, the job manager analyzes it to ensure that the new resources do not collide with other resources already deployed.

- Low maintenance effort

  Each server can be prepared by the developer, including all the server resources that the application needs. The administrator can modify the `server.xml` file before deploying it to include database names, user names, and passwords for the production environment.

After the server is distributed by using the job manager, no application server-specific maintenance on the systems that run the Liberty profiles is required.

- Low resource consumption

  In a Liberty profile, only the application server functions that are needed for the applications that are running on this server are started. Depending on the application, this method can dramatically reduce the server resources needed to run the profile.

- Isolation between applications

  Low maintenance effort and low resource consumption enable you to create a Liberty profile per application. This method provides an excellent isolation between the applications, and makes it unlikely that one faulty application can influence other applications.

- Administration tasks can be scheduled

  The job manager allows you to schedule administration tasks. No person must be present while the task is run.

- The HTTP servers provide load balancing and failover

  The Liberty profile servers do not provide high availability and load balancing features. However, if you access Liberty profile servers through HTTP servers, the HTTP server plug-ins provide load balancing and failover. The job manager can update the plug-in configurations.

- Scalability

  Because it is easy to add Liberty profiles to this topology, the application server layer has a high scalability. You can scale the topology horizontally and vertically.

Considerations

The job manager can update only Liberty profile servers that are currently not running. If you require application availability while an application is being updated, ensure that the application is deployed on more than one Liberty profile server. You can update Liberty profiles without outage by using one of these patterns:

- Create new servers that run concurrently with existing servers:
  a. Create a copy of the existing Liberty profile server configuration as the new server configuration. Name the new server with an edition name, for example `AppSrv01_V02`.

     If the new server is to run on the same host as the old server, reconfigure its ports as well.
  b. Deploy the new version to the target hosts.
  c. Start the servers, and perform initial testing.
  d. Add each new server to the HTTP server plug-in configuration.
  e. Change the HTTP server plug-in configuration to stop routing requests to the old servers.
  f. Stop the existing Liberty profile servers after the sessions end or time out.
- Create new servers that do not run concurrently with the existing servers:
  a. Create a copy of the existing Liberty profile server configurations as the new server configuration. Name the server with an edition name in mind, for example `AppSrv01_V02`.
  b. Deploy the new version to target hosts, which might be the same host as the existing servers.

c. Switch from the existing servers to the new servers, the first server, or server group:

   i. Change the weight of the existing server group in the HTTP server plug-in configuration to prevent new requests from going to these servers.

   ii. Wait for the session to time out.

   iii. Stop the existing server group.

   iv. Start the new server group.

   v. Update the HTTP server plug-in configuration with the new servers and remove the existing servers.

   vi. Repeat these steps for all subsequent servers or server groups.

- Update the servers that are in place by reusing existing servers:

  Switch over from existing servers to new servers for the first server or server group by performing these steps:

  a. Change the plug-in weight of the existing server group.
  b. Wait for the session to time out.
  c. Stop the existing group of servers.
  d. Uninstall the existing servers, and then reinstall the servers with new versions.
  e. Start the new group of servers.
  f. Update the HTTP server plug-in configuration with the new servers.

  Repeat these steps for all subsequent servers or server groups.

  Important: This approach makes it more difficult to undo the update.

Make sure that after the application update finishes, all Liberty profiles run the same application edition to ensure consistent behavior for users.

Setting up the topology

To set up a topology by using job manager, Liberty profiles, and HTTP servers (Figure 8-5 on page 203), complete the steps in this section.

Setting up the HTTP servers (Systems A and B)

To set up an HTTP server for Systems A and B, follow these steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install the following applications:
   - Web server plug-ins for WebSphere Application Server
   - WebSphere Customization Toolbox
   - IBM HTTP Server

Repeat these steps for every HTTP server system that you require. If you are not using IBM HTTP Server, install any other supported web server.

Setting up the job manager (System F)

To set up the job manager (System F), follow these steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server V8.5.
3. Create an application server profile by using the *app_server_root*`/profileTemplates/management` profile template and setting the server

type to the job manager. If you are using the **manageprofiles** command, add the following parameter:

```
-serverType JOB_MANAGER
```

4. Start the job manager.

Setting up the Liberty profiles (Systems C, D, and E)

To set up the Liberty profiles for Systems C, D, and E, follow these steps:

1. Download the WebSphere Liberty profile compressed file.
2. Add your server configuration and applications to the file. You can create multiple compressed files that contain different configurations and applications.
3. In job manager, define each system that will run a Liberty profile as a target.
4. Submit an “Install Liberty profile server resources” task, and specify the compressed file for the target. Job manager deploys the Liberty profiles.
5. Run the “Merge generated plug-ins of Liberty profile servers” task, and distribute the plug-in configurations to the HTTP servers.
6. Run the “Start Liberty profile server” task to start the Liberty profile servers.

8.3.4 Vertical scaling topology

A *vertical scaling topology* (illustrated in Figure 8-6) is a configuration with multiple application servers on a single system or LPAR, and a cluster of associated application servers. These servers all host the same applications. All members of the cluster are displayed as one logical unit that serves the applications that are deployed to the cluster.

Keep in mind that a WebSphere Application Server cluster can be implemented only with WebSphere Application Server Network Deployment or WebSphere Application Server for z/OS.

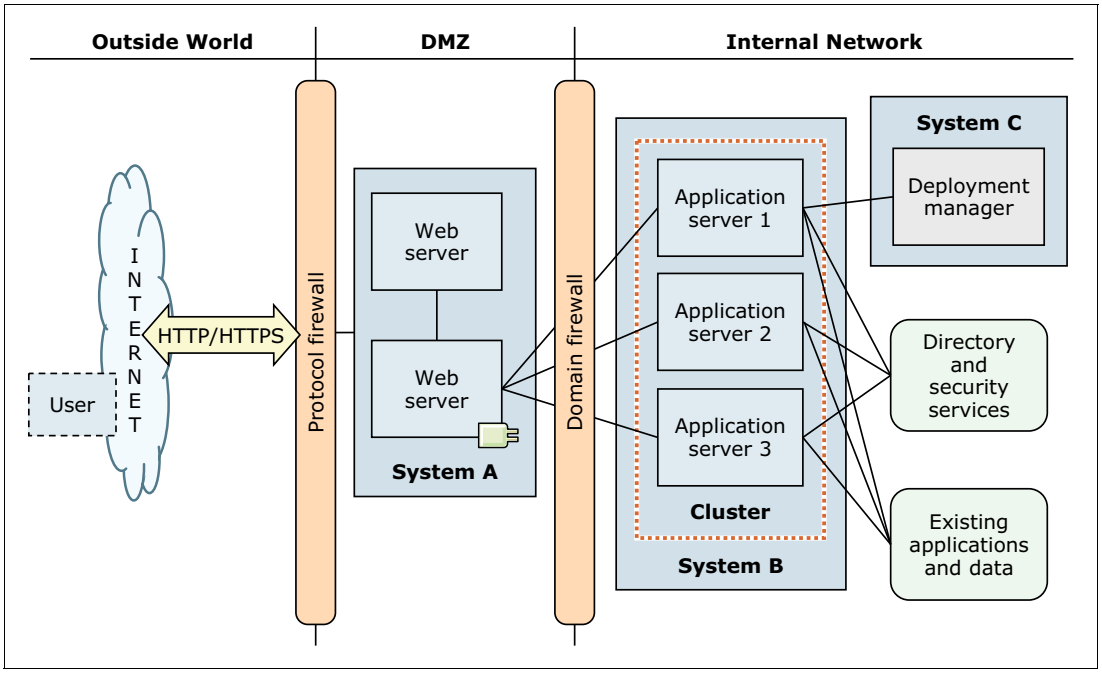


Figure 8-6 Vertical scaling topology with WebSphere Application Server

This vertical scaling example includes a cluster and three cluster members. The web server plug-in routes the requests according to the availability of the application server. Basic load balancing is run at the web server plug-in level based, by default, on a weighted round-robin algorithm.

Important: The illustration in Figure 8-6 is intended to show a vertical scaling topology of application servers, but still contains several SPOFs.

You can combine vertical scaling with other topologies to optimize performance, throughput, and availability.

Advantages

Implementing vertical scaling in your topology provides the following advantages:

- Improved throughput

  Because multiple application servers service client requests simultaneously, you can expect improved throughput from your installation.

- Optimized resource use

  With vertical scaling, each application server that runs its own JVM uses a portion of the processor and memory of the system. The number of application servers on a system can be increased or decreased to optimize the resource use of the system.

- Growth beyond the limits of a single JVM

  With a vertical scaling implementation, you can grow your environment with your implementation beyond the limits of a single JVM. You can run multiple JVMs in parallel.

- Benefits from the workload management capabilities of WebSphere Application Server

  Because vertical scaling is implemented through clusters, you benefit from WebSphere Application Server workload management.

- Failover support

  Because vertical scaling is implemented by using clusters, vertical scaling topologies can also take advantage of the failover support provided by WebSphere Application Server. If one of the application server processes is stopped, the remaining cluster members continue to process and realign the workload.

Considerations

Keep in mind the following considerations with vertical scaling:

- SPOF

  Unless you combine the vertical scaling architecture with horizontal scaling, you still have SPOFs (such as hardware and operating system processes) in your architecture.

- Additional investment and processes

  To implement vertical scaling, you need WebSphere Application Server Network Deployment. You need additional application server processes, such as the deployment manager and the node agent process, to manage such an environment.

- Additional planning and implementation work required

  To benefit from the load balancing and failover capabilities, you need to plan for these scenarios. To benefit from a failover mechanism, you must consider what is required for a successful failover (such as session data). You must also size carefully for all possible situations.

Setting up the topology

To set up an environment similar to the one illustrated in Figure 8-6 on page 206, complete the steps in this section. These steps include the minimum software configuration that you need for this topology.

Setting up System A

To set up System A, complete these steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install the following applications:
   - Web server plug-ins for WebSphere Application Server
   - WebSphere Customization Toolbox
   - IBM HTTP Server

   If you are not using IBM HTTP Server, install a supported web server.
3. Open the WebSphere Customization Toolbox, and start the Web Server Plug-ins Configuration Tool.
4. Configure the web server plug-in, and create the web server definition. For details about this task, see the Websphere Application Server V8.5 Information Center at:

   `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=tins_webplugins`

Setting up System B

To set up System B, complete these steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile, also known as a *custom profile*:
   - Use the *app_server_root*`/profileTemplates/managed` profile template. Then federate the application server profile to the deployment manager on System C during profile creation or after the profile creation by running the **`addNode`** command.
   - Use the *app_server_root*`/profileTemplates/default` profile template. Then federate the node to the deployment manager that is running on System C by using the **`addNode`** command.

Setting up System C

To set up System C, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile:
   - Use the *app_server_root*`/profileTemplates/dmgr` profile template.
   - Use the *app_server_root*`/profileTemplates/management` template, and specify `-serverType` for DEPLOYMENT_MANAGER.
4. Create a web server definition by using either the administrative console or the **`wsadmin`** scripting interface.
5. Create a WebSphere Application Server cluster with three cluster members on System B.

8.3.5 Horizontal scaling topology

In a horizontal scaling topology, you create one logical unit of servers across multiple systems or LPARs where each member of the unit serves each request. Horizontal scaling at the application server tier does not require an IP sprayer. If you also want to scale at the web server tier, you can use an IP sprayer.

This section introduces two topologies. One is without an IP sprayer, and the other has the IP sprayer component. For more information, see 8.3.6, “Horizontal scaling topology with an IP sprayer” on page 211.

Horizontal scaling topology without an IP sprayer

In the topology illustrated in Figure 8-7, a single application spans multiple systems, but presents itself as a single logical image. In this example, the WebSphere Application Server cluster spans Systems B and C, each with one application server. The deployment manager is installed on a separate server, System D.

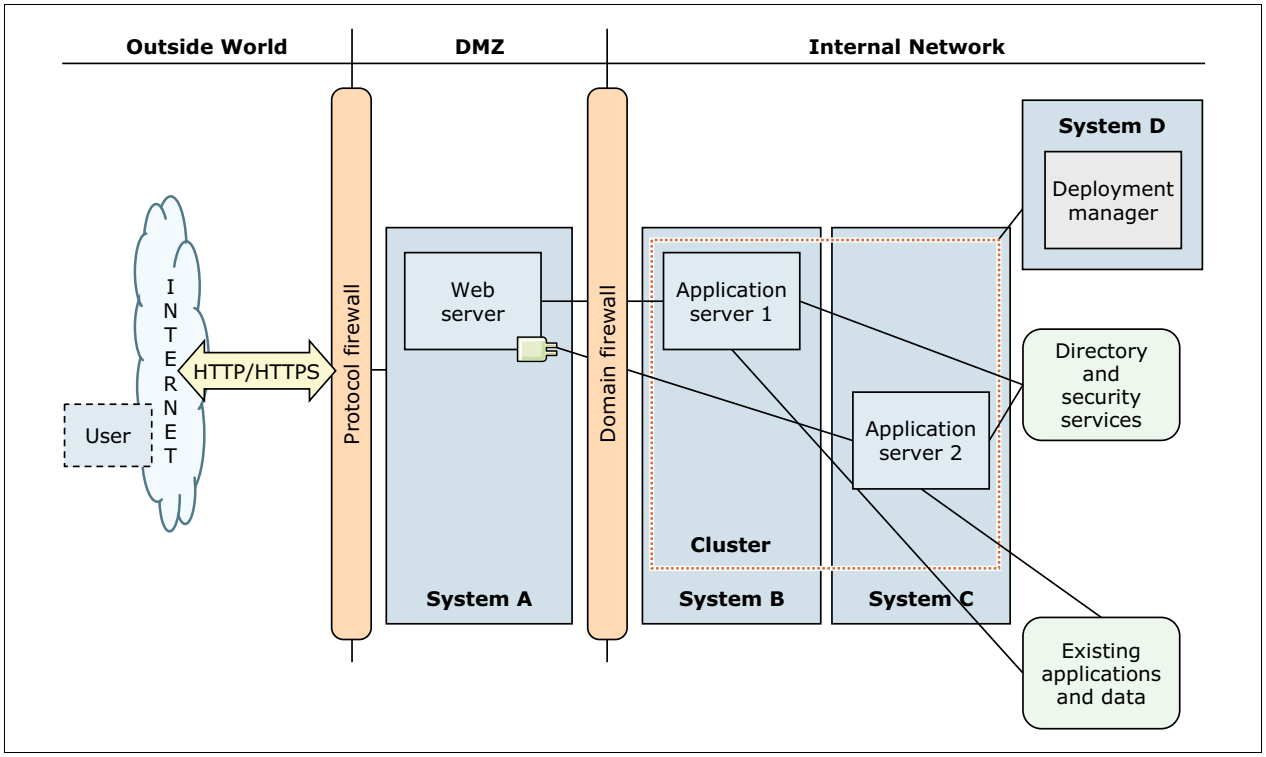


Figure 8-7 Horizontal scaling with cluster

The web server plug-in distributes requests to the cluster members on each server that runs load balancing. It offers an initial failover in a similar manner as it does in the vertical clustering topology. If any component (hardware or software) on System B fails, the application server on System C can continue to serve requests and vice versa.

Important: The illustration in Figure 8-7 is intended to show a horizontal scaling topology of application servers but still contains a SPOF (namely, the web server). To avoid this SPOF, you must enhance the topology as illustrated in 8.3.6, “Horizontal scaling topology with an IP sprayer” on page 211.

Advantages

Using horizontal scaling with clusters has the following advantages:

- Improved throughput

  Because multiple systems service client requests simultaneously without competing for resources, you can expect improved throughput from your installation.

- Improved response times

  By hosting cluster members on multiple systems, each cluster member can use the processing resources of the system, avoiding bottlenecks and resource contention. Therefore, response times improve in most scenarios.

- Benefits from the workload management capabilities of WebSphere Application Server

  Because horizontal scaling is implemented through clusters, it benefits from the workload management capabilities of WebSphere Application Server.

- Provides enhanced failover support

  Because the cluster members are spread over multiple systems, this topology provides hardware failover capabilities. Client requests can be redirected to cluster members on other systems if a system goes offline. The outage of a system or an operating system failure does not stop a service from working.

Considerations

Keep in mind the following considerations when using horizontal scaling with clusters:

- Increased resource usage

  Because multiple systems are required to implement this topology, hardware costs increase. To implement horizontal scaling, you need WebSphere Application Server Network Deployment. Therefore, you need additional application server processes, such as the deployment manager and the node agent process, to manage this type of environment. This method increases processing and the memory footprint of the installation.

- More complex administration

  The maintenance and administration of the environment are more complex because the number of systems increases.

Setting up the topology

To set up a topology environment similar to the one illustrated in Figure 8-7 on page 209, complete the steps in this section. These steps include the minimum software configuration that you need for this topology.

Setting up System A

To set up System A, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install the following applications:
   - Web server plug-ins for WebSphere Application Server
   - WebSphere Customization Toolbox
   - IBM HTTP Server

     If you are not using IBM HTTP Server, install a supported web server.
3. Open the WebSphere Customization Toolbox, and start the Web Server Plug-ins Configuration Tool.

4. Configure the web server plug-in, and create the web server definition. For more information, see the Websphere Application Server V8.5 Information Center at:

   http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=tins_webplugins

Setting up Systems B and C

To set up Systems B and C, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile, also known as a *custom profile*:
   - Use the *app_server_root*/profileTemplates/managed profile template. Then federate this profile to the deployment manager on System D during profile creation or after profile creation by running the **addNode** command.
   - Use the *app_server_root*/profileTemplates/default profile template. Then federate the node to the deployment manager that runs on System D by using the **addNode** command.

Setting up System D

To set up System D, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile:
   - Use the *app_server_root*/profileTemplates/dmgr profile template.
   - Use the *app_server_root*/profileTemplates/management template, and specify -serverType for DEPLOYMENT_MANAGER.
4. Create a web server definition through the administrative console or the **wsadmin** scripting interface.
5. Create a WebSphere Application Server cluster with one cluster member on System B and one cluster member on System C.

8.3.6 Horizontal scaling topology with an IP sprayer

You can use load balancing products to distribute HTTP requests among web servers that are running on multiple physical systems. The load balancer component of Network Dispatcher, for example, is an IP sprayer that runs intelligent load balancing among web servers based on server availability and workload.

The active load balancer hosts the highly available TCP/IP address, the cluster address of your service, and spray requests to the web servers. At the same time, the load balancer tracks web server health and routes requests around web servers that are not available. To avoid having the load balancer be a SPOF, set up the load balancer in a hot-standby cluster. The primary load balancer communicates its state and routing table to the secondary load balancer. The secondary load balancer monitors the primary load balancer through heartbeat and takes over when it detects a problem with the primary load balancer. Only one load balancer is active at a time.

Both web servers are active at the same time. They run load balancing and failover between the application servers in the cluster through the web server plug-in. The plug-in detects any component on System C or System D that fails, and the other server can continue to receive requests.

Figure 8-8 illustrates a horizontal scaling configuration that uses an IP sprayer to redistribute requests between web servers on multiple systems.

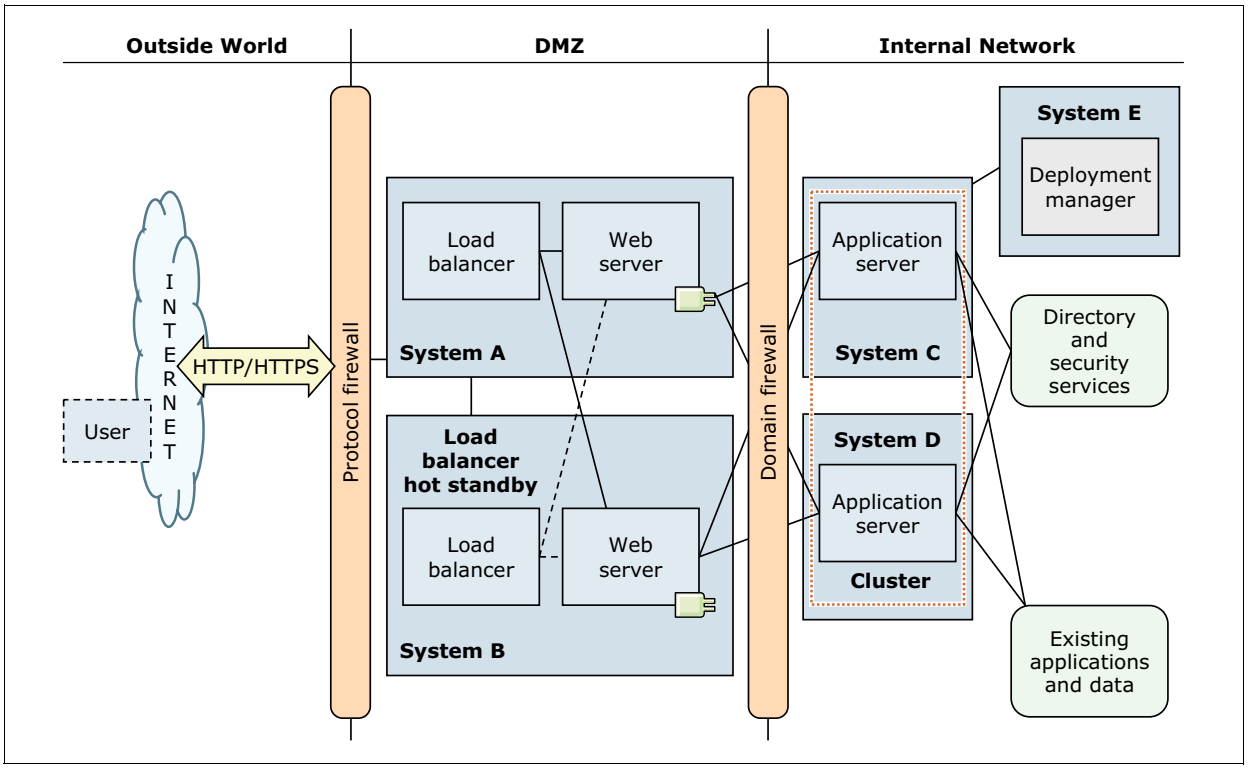


Figure 8-8 Simple horizontal scaling topology with an IP sprayer

Advantages

Using an IP sprayer to distribute HTTP requests has the following advantages:

- Improved throughput and performance

  By maximizing parallel processor and memory usage, you can expect increased throughput and performance. The distribution of incoming TCP/IP requests among a group of web servers spreads the workload among the available application servers. This sharing helps to improve performance.

- Increased capacity

  The usage of multiple web servers increases the number of connected users that can be served at the same time.

- Elimination of the web server as a SPOF

  Used in combination with load balancers, this topology eliminates the web server as a SPOF.

Considerations

Keep in mind cost considerations when using a load balancer. It also causes increased complexity. This configuration requires the load balancer component to be installed and maintained. Therefore, this component increases the complexity of the installation and

configuration. Because a load balancer runs on a separate system, there are more systems to manage and operate. Adding more systems in turn increases the operational cost of the environment.

Setting up the topology

To set up a topology environment similar to the one illustrated in Figure 8-8 on page 212, complete the setup steps in this section. These steps include the minimum software configuration that you need for this topology.

Setting up Systems A and B

To set up Systems A and B, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install IBM WebSphere Edge Components.
3. Configure the load balancer component according to your network topology. You need two valid IP addresses:
   - An IP address for your web servers cluster
   - An IP address for the system
4. Using Installation Manager, install the following applications:
   - Web server plug-ins for WebSphere Application Server
   - WebSphere Customization Toolbox
   - IBM HTTP Server

   If you are not using IBM HTTP Server, install a supported web server.
5. Open the WebSphere Customization Toolbox, and start the Web Server Plug-ins Configuration Tool.
6. Configure the web server plug-in, and create the web server definition.

Setting up Systems C and D

To set up Systems C and D, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile:
   - Use the *app_server_root*/profileTemplates/managed profile template. Then federate this profile to the deployment manager on System E during profile creation or after the profile creation by running the **addNode** command.
   - Use the *app_server_root*/profileTemplates/default profile template. Then federate the node to the deployment manager that runs on System E by using the **addNode** command.

Setting up System E

To set up System E, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.

3. Create an application server profile:
   - Use the *app_server_root*/profileTemplates/dmgr profile template.
   - Use the *app_server_root*/profileTemplates/management template, and specify -serverType for DEPLOYMENT_MANAGER.
4. Create the web server definitions by using either the administrative console or the **wsadmin** scripting interface.
5. Create a WebSphere Application Server cluster by using either the administrative console or **wsadmin** with one cluster member on System C and one cluster member on System D.

8.3.7 Reverse proxy topology

Reverse proxy servers, such as the one provided with the Edge Components or the DMZ secure proxy, are typically used in DMZ configurations for the following reasons:

- To provide additional security between the public Internet and web servers (and application servers)
- To increase performance and reduce the load on servers by content caching

The topology in Figure 8-9 shows the use of DMZ Secure Proxy Server as the reverse proxy server. It is used in this example because it offers a more secure option than the Proxy Server profile.

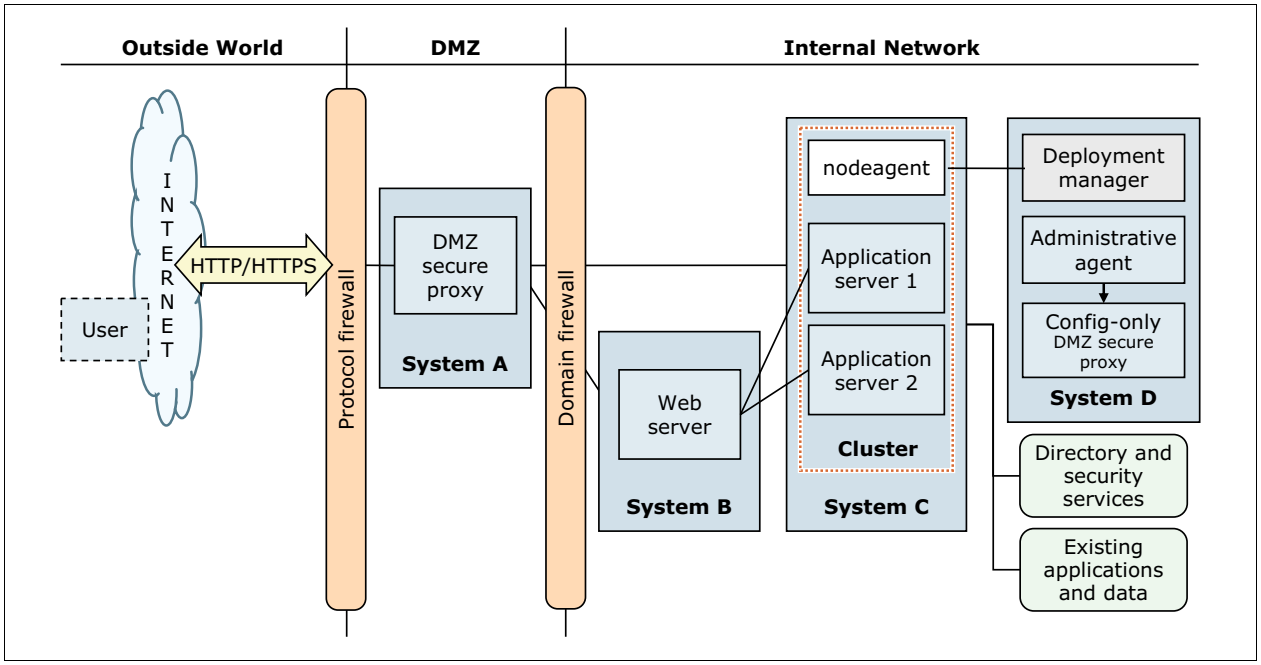


Figure 8-9 Topology using a DMZ Secure Proxy Server

Remember: The illustration in Figure 8-9 is intended to provide an overview of a topology that contains a reverse proxy. High availability is not incorporated here. To achieve high availability, at least another reverse proxy server and two sets of load balancer cluster addresses are required. One cluster address is for the proxy servers, and one cluster address is for the web servers.

The DMZ Secure Proxy Server intercepts the requests that are going to the application servers. It then looks for a valid copy of the requested object in its cache. If a valid, cached version is found, the cached copy is returned to the client. If no valid copy of the requested object is found, the proxy server forwards those requests to the web server in the internal network. Responses are returned through the reverse proxy to the web client. This process hides the web servers from the clients and allows the proxy server to store a copy of the object in the local cache, if the configuration permits.

Reverse proxy configurations support high-performance DMZ solutions that require as few open ports in the firewall as possible. The reverse proxy requires only one open port per protocol to access the web server behind the firewall.

The DMZ secure proxy on System D is used to create a configuration for the DMZ secure proxy on System A.

Advantages

Using a reverse proxy server in a DMZ configuration has the following advantages:

- Independent configuration

  The reverse proxy installation has no effect on the configuration and maintenance of a WebSphere application.

- Offloading the web servers

  The reverse proxy servers delivered with WebSphere Application Server V8.5 provide caching capabilities. These servers offload the web servers and the application servers if dynamic caching is also supported.

Considerations

Keep in mind the following considerations when using a reverse proxy server in a DMZ configuration:

- Increased complexity

  This configuration requires a reverse proxy server component to be installed and maintained, increasing the complexity of the installation and configuration.

- Increased latency for non-cacheable objects

  Requests for non-cacheable objects increase network latency and lower performance. To be effective, a sufficiently high cache hit rate is required.

Setting up the topology with the DMZ secure proxy

The DMZ secure proxy is *not* supported when using the base version of WebSphere Application Server. Therefore, you need the Network Deployment version.

To set up a topology environment similar to the one illustrated in Figure 8-9 on page 214, complete the steps in this section. These steps include the minimum software configuration that you need for this topology. In this example, the DMZ secure proxy is set up with a security level of HIGH and, therefore, it supports only static routing.

Setting up System A

To set up System A, complete the following steps:

1. Install IBM Installation Manager.
2. Install the DMZ Secure Proxy Server for IBM WebSphere Application Server.
3. Create an application server profile by using the *app_server_root*/profileTemplates/secureproxy template.

Setting up System B

To set up System B, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install the following applications:
   - Web server plug-ins for WebSphere Application Server
   - WebSphere Customization Toolbox
   - IBM HTTP Server

   If you are not using IBM HTTP Server, install a supported web server.
3. Open the WebSphere Customization Toolbox, and start the Web Server Plug-ins Configuration Tool.
4. Configure the web server plug-in, and create the web server definition. For details about this task, see the Websphere Application Server V8.5 Information Center at:

   `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=tins_webplugins`

Setting up System C

To set up System C, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile:
   - Use the *app_server_root*`/profileTemplates/managed` profile template. Then federate this profile to the deployment manager on System D during profile creation, or after the profile creation by running the **addNode** command.
   - Use the *app_server_root*`/profileTemplates/default` profile template. Then federate the node to the deployment manager that is running on System D by using the **addNode** command.

Setting up System D

To set up System D, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile:
   - Use the *app_server_root*`/profileTemplates/dmgr` profile template.
   - Use the *app_server_root*`/profileTemplates/management` template, and specify `-serverType` for DEPLOYMENT_MANAGER.
4. Create WebSphere Application Server clusters or unclustered servers on System C.
5. Deploy the applications to the application servers, and make sure that they are started.

6. Export the routing information. Because you are setting up a DMZ secure proxy with a security level of HIGH, only static routing is supported.
   a. Go to the *profile_root*/bin directory of the deployment manager profile.
   b. Start **wsadmin.bat(sh) -lang jython** by using the Jython scripting language, and run the commands shown in Example 8-1 to export the static routing information.

   Example 8-1 Jython code to export static routing information

   ```
   mbean=AdminControl.queryNames('*:*,type=TargetTreeMbean,process=dmgr')
   AdminControl.invoke(mbean, 'exportTargetTree', 'directory/targetTree.xml')
   ```

   c. Copy the *directory*/targetTree.xml file to System A.
7. Create an application server profile by using the *app_server_root*/profileTemplates/management profile template, and specify -serverType for ADMIN_AGENT to create the administrative agent profile.
8. Create an application server profile by using the *app_server_root*/profileTemplates/secureproxy template.

   Remember: The profiles for the administrative agent and DMZ secure proxy are for administration purposes only. The DMZ Secure Proxy Server is a configuration-only profile, meaning that the server cannot be started or used for any work. This server is an administrative place holder for the DMZ Secure Proxy Server on System A. If you try to start the configuration-only profile, it fails with the following error message in the SystemOut.log file:

   ```
   Caused by: com.ibm.ws.proxy.deployment.ProxyServerDisabledException: This
   secure proxy server is part of a configuration-only installation and cannot
   be started.
   ```

   Tip: Use the same proxy server and the node names in the configuration-only profile on System D as you did in the DMZ Secure Proxy Server on System A. Use the -serverName and -nodeName parameters when running **manageprofiles**.

9. Register the configuration-only profile of the DMZ secure proxy to the administrative agent.
10. Using the administrative console from the administrative agent, manage the configuration-only template of the DMZ secure proxy.
11. Export the configuration-only DMZ secure proxy to move the changes to the real DMZ secure proxy:
    a. Go to the *profile_root*/bin directory of the configuration-only DMZ secure proxy profile.
    b. Enter the following command:

       ```
       wsadmin -lang jython -conntype NONE
       ```

    c. Export the proxy profile:

       ```
       AdminTask.exportProxyProfile('[-archive directory/DMZProxy.car]'
       ```

12. Copy the *directory*/DMZProxy.car file to System A.

Setting up System A

To set up System A, complete the following steps:

1. Copy the `targetTree.xml` file to the *`profile_root`*`/staticRoutes` directory.

   Important: The static routing information is not updated automatically. Whenever a change occurs (for example when an application is installed or removed), the routing information must be manually refreshed. You must restart the DMZ Secure Proxy Servers after each refresh of the routing information to activate the change. If restarting is not feasible, switch dynamic routing to use a lower security level.

2. Go to the *`profile_root`*`/bin` directory, and enter the following command:

   ```
   wsadmin -lang jython -conntype NONE
   ```

3. Import the profile changes from the *`directory`*`/DMZProxy.car` file by running the **`wsadmin`** command shown in Example 8-2.

   Example 8-2 Importing the proxy profile

   ```
   AdminTask.importProxyProfile('-archive directory/DMZProxy.car
   -deleteExistingServers true')
   AdminConfig.save()
   ```

4. Start the DMZ Secure Proxy Server.

8.3.8 Topology with redundancy of multiple components

To remove SPOF in a topology, add redundant components. Most components in a WebSphere Application Server topology provide the options to implement redundancy. Such examples include a load balancer hot standby server with a primary load balancer server, clustered web servers, and clustered application servers.

In a topology with redundant components, those components can be in an active state, known as *active-active redundancy*, or passive state, known as *active-passive redundancy*. In *active-active redundancy*, both the primary and redundant components process requests and serve as failover components for each other. In *active-passive redundancy*, only one of the components processes requests while the other waits to take the work of the other component if it fails.

The topology in Figure 8-10 on page 219 shows the minimum WebSphere components that are used in an installation with the usual high availability requirements. The number of application servers might vary. This figure illustrates a topology with redundancy of several components. In this scenario, the load balancer clusters are in active-passive redundancy, and the proxy servers, web servers, and application servers are in active-active redundancy.

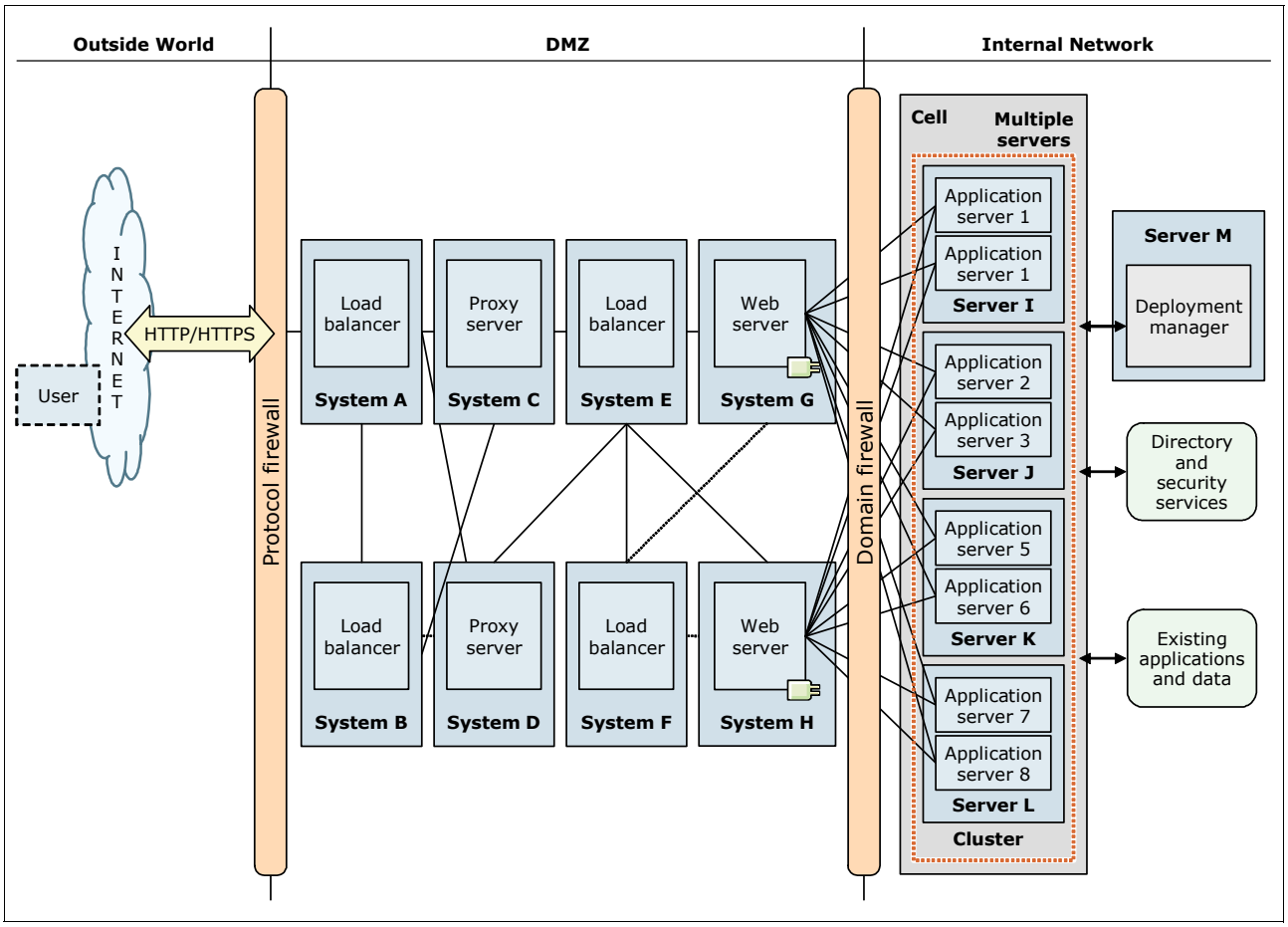


Figure 8-10 Topology with redundancy of several components

The following components are redundant in this example:

- Two clusters of load balancers

  The topology illustrated in Figure 8-10 shows two clusters of load balancers. Each cluster provides a highly available cluster address. The first cluster, which runs on System A with System B as a hot standby, provides the cluster address for the reverse proxies. This cluster address is used by the clients to access the service.

  The second cluster runs on System E, with System F as a hot standby. This cluster provides the cluster address for the web servers. It is used by the proxy servers to retrieve content if user requests cannot be served out of the cache. For a high-level overview of how load balancers work, see 8.3.6, “Horizontal scaling topology with an IP sprayer” on page 211.

  Tip: Each load balancer installation can host multiple cluster addresses. You do not need a separate installation for each cluster address.

- Two reverse proxy servers

  Both of the reverse proxy servers (running on System C and System D) receive requests from the load balancer. They share the requests that are coming from the clients. Each proxy server is installed on a different system to provide a maximum level of redundancy. Keep in mind that both reverse proxy servers must have an identical configuration.

- Two web servers

  Each web server, one running on System G and the other one running on System H, receives requests from the second load balancer cluster. They share the requests that come from the reverse proxies. Each web server is installed on a different system, but they must have an identical configuration.

- An application server cluster

  The cluster is spread across four server systems, and implements a combination of vertical and horizontal scaling. The cluster consists of eight cluster members, two on each server. Although the application servers are grouped together in one cluster, they might originate from different installations. The application servers on System I, for example, can be two separate installations of WebSphere Application Server. The application servers on System J can be profiles of a single installation.

- Two database servers

  The database servers need to be made highly available by using database system-specific tools or operating system-based clustering software.

- Two LDAP servers

  The LDAP servers can use a high availability software product (such as another load balancer) or backup LDAP server support. This backup support is provided through the user registry of the federated repositories in WebSphere Application Server. The LDAP servers must have an identical structure and user population.

This topology maximizes performance, throughput, and availability. It incorporates the benefits of the other topologies described earlier in this chapter.

No high availability is considered for the deployment manager because this component is not a SPOF. Therefore, cell-wide services, such as high availability (HA) manager and JNDI, are not highly available. Nevertheless, you can implement operating system or hardware high availability for this component to avoid losing the enhanced administration capabilities that it offers. However, consider the associated costs, such as hardware and operational costs, of having a highly available deployment manager. For more information about how to accomplish this task, see *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688.

Advantages

The topology with redundancy of multiple components has the following advantages:

- Elimination of most SPOF

  This topology does not have any SPOF. The load balancer node, reverse proxy server, web server, application server, database server, and LDAP server are set up in a redundant way.

- Horizontal scaling

  Horizontal scaling is done by using both the IP sprayer (for the reverse proxy and the web server nodes) and application server cluster technology to maximize availability. For more information about the benefits of horizontal scaling, see 8.3.5, “Horizontal scaling topology” on page 209.

- Improved application performance

  In most cases, application performance is improved by using the following techniques:

  - Hosting application servers on multiple physical systems, z/OS images, or both to optimize the usage of available processing power.
  - Using clusters to scale application servers vertically, which makes more efficient use of the resources of each system.

- Usage of workload management technologies

  Applications in this topology can benefit from workload management techniques. In this example, workload management is run as follows:

  - Load Balancer Network Dispatcher distributes client HTTP requests to each reverse proxy server.
  - Load Balancer Network Dispatcher distributes requests from the proxy servers to each web server.
  - The workload management feature of WebSphere Application Server Network Deployment distributes work among clustered application servers.

Considerations

When using this topology, keep in mind that redundancy that uses multiple components increases cost. For this combined topology, consider costs in hardware, complexity, configuration, and administration. Consider these costs in relation to advantages in performance, throughput, and reliability.

Remember: Because this topology is a combination of the topologies described earlier in this chapter, the considerations of other base topologies also apply here.

Setting up the topology

To set up a topology environment similar to the one illustrated in Figure 8-10 on page 219, complete the steps in this section. These steps include the minimum software configuration that you need for this topology.

Setting up Systems A, B, E, and F

To set up Systems A, B, E, and F, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Edge Component Load Balancer for IPv6.
3. Configure the Edge Component Load Balancer component according to your network topology.

   Consideration: Keep in mind that Systems A and B form one cluster and Systems E and F form another, different cluster.

Setting up Systems C and D

To set up Systems C and D, install and set up the Proxy Server.

Setting up Systems G and H

To set up Systems G and H, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install the following applications:
   - Web server plug-ins for WebSphere Application Server
   - WebSphere Customization Toolbox
   - IBM HTTP Server

   If you are not using IBM HTTP Server, install a supported web server.
3. Open the WebSphere Customization Toolbox, and start the Web Server Plug-ins Configuration Tool.
4. Configure the web server plug-in, and create the web server definition. For more information, see the Websphere Application Server V8.5 Information Center at:

   `http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.doc/ae/twsv_plugin.html`

Setting up Systems I, J, K, and L

To set up Systems I, J, K, and L, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile:
   - Use the *app_server_root*`/profileTemplates/managed` profile template. Then federate this profile to the deployment manager on System M during profile creation, or after the profile creation by running the **addNode** command.
   - Use the *app_server_root*`/profileTemplates/default` profile template. Then federate the node to the deployment manager that runs on System M by using the **addNode** command.

Setting up System M

To set up System M, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile:
   - Use the *app_server_root*`/profileTemplates/dmgr` profile template.
   - Use the *app_server_root*`/profileTemplates/management` template, and specify `-serverType` for DEPLOYMENT_MANAGER.
4. Create web server definitions through the administrative console or through the **wsadmin** scripting interface.
5. Create a WebSphere Application Server cluster with two cluster members on System I, J, K, and L.

8.3.9 Heterogeneous cell topology

Cells can span servers across multiple heterogeneous operating systems such as z/OS sysplex environments and distributed platforms. For example, z/OS nodes, Linux nodes, UNIX nodes, and Microsoft Windows nodes can exist in the same application server cell. This configuration type is called a *heterogeneous cell*. With WebSphere Application Server V8.5, many different topologies are possible to compose a heterogeneous cell, as illustrated in Figure 8-11.

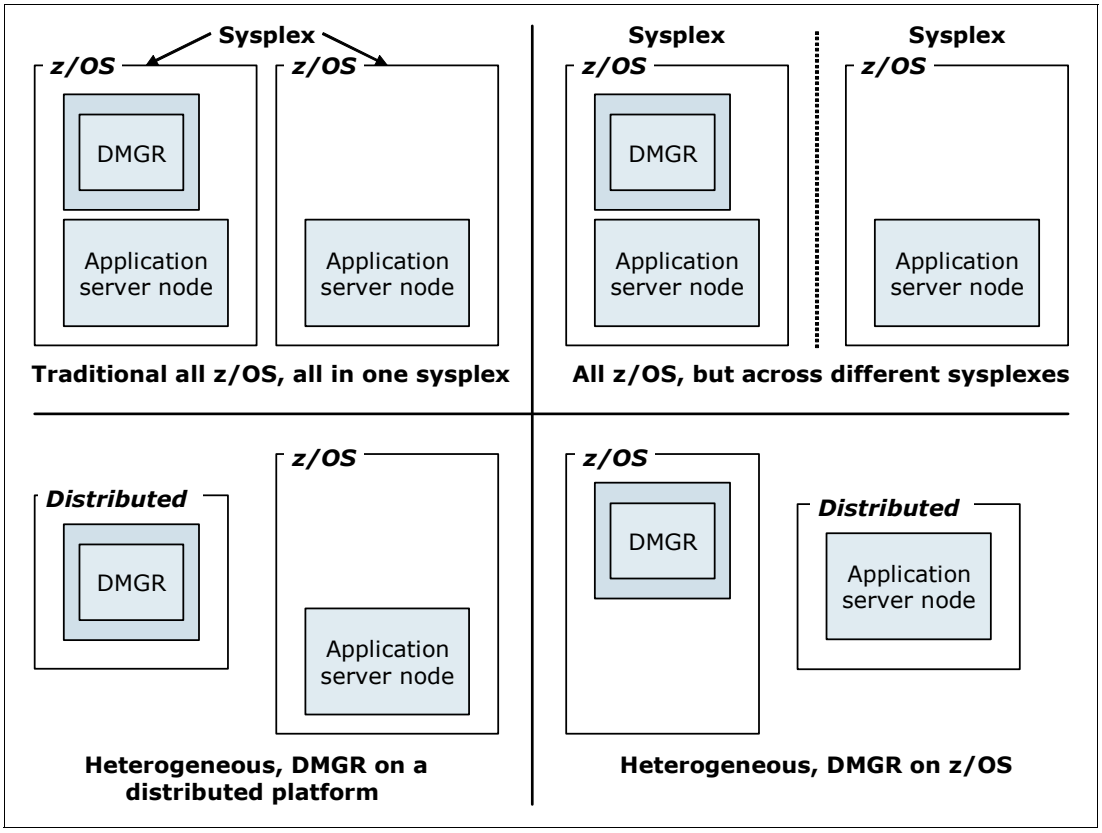


Figure 8-11 Different configurations available with a heterogeneous cell

WebSphere Application Server V8.5 products can coexist with the following supported versions:

- IBM WebSphere Application Server V6.1
- IBM WebSphere Application Server Network Deployment V6.1
- IBM WebSphere Application Server V6.1 for z/OS
- IBM WebSphere Application Server V7.0
- IBM WebSphere Application Server Network Deployment V7.0
- IBM WebSphere Application Server V7.0 for z/OS
- IBM WebSphere Application Server V8.0
- IBM WebSphere Application Server Network Deployment V8.0
- IBM WebSphere Application Server V8.0 for z/OS

WebSphere Application Server V6 and later, V7, and V8 clients can coexist with V8.5 clients.

A WebSphere Application Server V8.5 Network Deployment cell can contain mixed releases of V6.1, 7.0, and 8.0 nodes. Upgrade all V6.0 nodes to at least V6.1. This upgrade allows the nodes to be administered by a V8.5 deployment manager.

Considerations for z/OS:

- Multiple WebSphere Application Server for z/OS cells can coexist in the same sysplex.
- Multiple WebSphere Application Server for z/OS nodes can coexist on the same LPAR.
- No two cells can have the same cell short name.
- Separate cells need separate configuration file system mount points and job control language (JCL) procedures.

Advantages

This topology maximizes performance, throughput, and availability. It incorporates the benefits of the other distributed server topologies, and adds the possibility to mix different operating systems. This topology has the following advantages:

- Horizontal and vertical scaling (described in previous sections)
- Flexible deployment of components

  Components can be deployed to systems on which they provide the best value and effectiveness.
- Easier integration and reuse of existing software components

  Because multiple systems can be included in the cell, the integration of existing, platform-specific software components is much easier.
- Easier migration

  Running different versions and platforms of WebSphere Application Server in a cell is a possible approach for migrating WebSphere Application Server versions. Although this environment is supported, mixed version cells are not a permanent solution.

Considerations

Keep in mind the following considerations when using this topology:

- Complex administration

  Because of the heterogeneous environment, the administration is complex and requires administrator knowledge for all platforms.
- Increased administration and operational costs

  For this combined topology, consider hardware, configuration, and administration costs. Consider these costs in relation to gains in performance, throughput, and reliability.

For more information about planning and system considerations that are required to build a heterogeneous cell, see the IBM white paper *WebSphere for z/OS -- Heterogeneous Cells* at:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100644

8.3.10 Multi-cell topology

The topologies introduced in the following sections provide a high level of availability and redundancy for all types of WebSphere components:

- 8.3.8, “Topology with redundancy of multiple components” on page 218
- 8.3.9, “Heterogeneous cell topology” on page 223

Nevertheless, application software problems or malfunctioning of components that affect the entire cell, although rare, are potential threats to the availability of your service. These threats include a malfunctioning network router or a shared file system interruption

A possible approach is a two cell architecture as outlined in Figure 8-12. This topology is a duplication of the topology introduced in 8.3.8, “Topology with redundancy of multiple components” on page 218. However, here independent cells are implemented. Even cell-level problems can be handled quickly in this topology, because full cells and their related infrastructure can be activated and deactivated as needed. Both cells can have the same applications installed. This approach can also be used to provide independent cell resources usage for different applications.

Carefully consider which business requirement you are trying to fulfill when planning to implement this topology type. Although this topology is suited for high availability when combined with a requirement for continuous operations, disaster recovery must be addressed in a different manner.

If you are looking for disaster recovery, keep in mind that the topology illustrated in Figure 8-12 is not a complete solution. You must consider several factors when planning for disaster recovery. For more information, see 8.2.3, “Disaster recovery” on page 188.

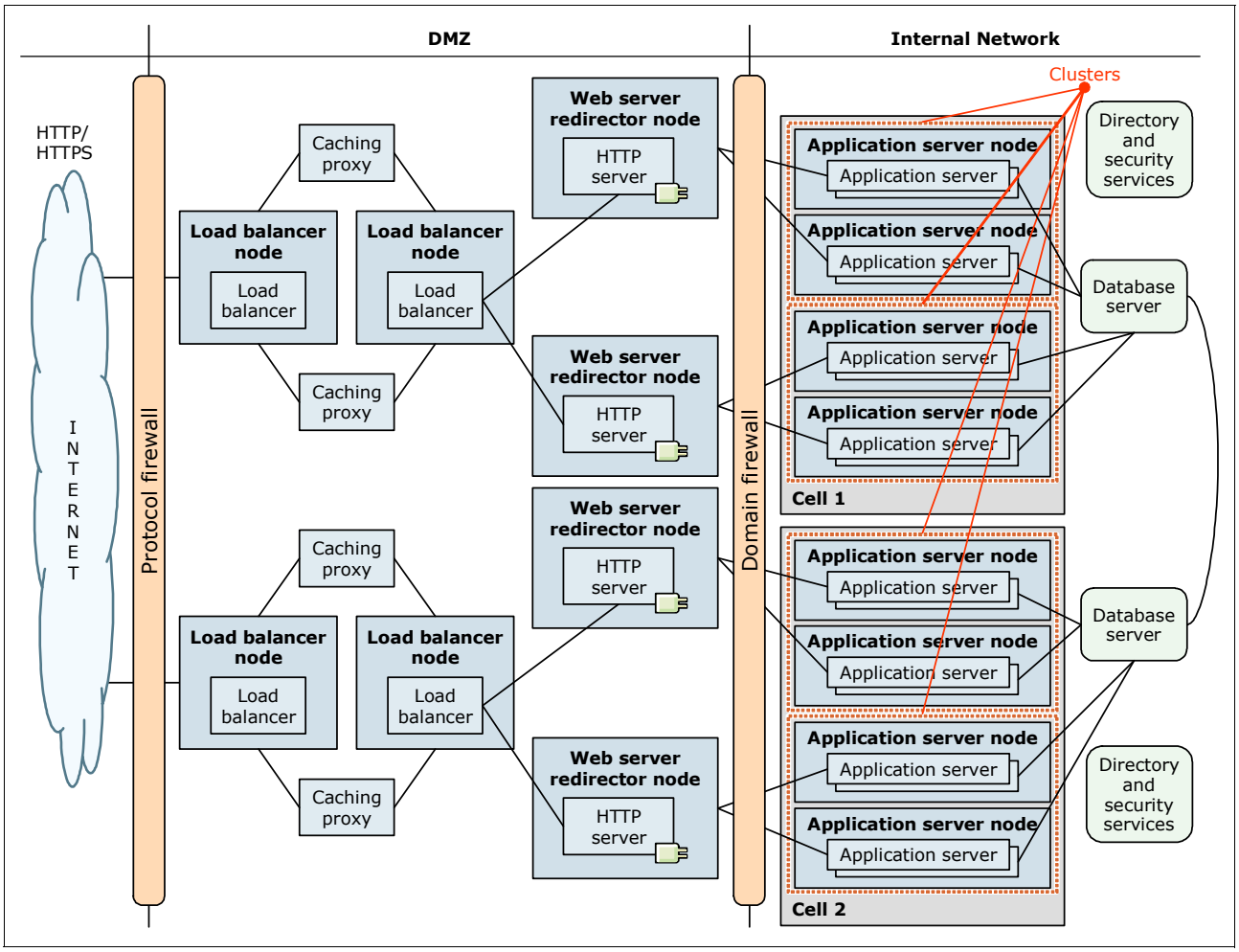


Figure 8-12 Multi-cell architecture

Advantages

The multi-cell topology has the following advantages:

- Provides all the advantages defined in 8.3.8, “Topology with redundancy of multiple components” on page 218.
- Allows you to react quickly to cell level problems

  If one of the cells is having issues, you can redirect the request that is going to the failing cell to another cell. One possible way to redirect the request is through Domain Name System (DNS) resolution.
- Allows stepwise WebSphere upgrades

  This topology allows independent releases of WebSphere Application Server software in each cell. Therefore, each cell can be upgraded on its own, which lowers risk during an upgrade and provides a fall-back scenario in case of upgrade problems.
- Allows stepwise application upgrades

  This topology allows independent application releases in each cell, and provides a quick fall-back scenario in case you encounter application problems in your production environment.

  Requirement: Compatibility between application software releases is required.
- Possible approach for disaster recovery

  Having the cells in different data centers is the preferred approach for a disaster recovery solution from a WebSphere perspective. Two cells in the same data center do not provide for redundancy during a catastrophic event that results in the outage of a data center.

  Considerations: For a real solution implementation for disaster recovery, you must address the following issues:

  - How do you route traffic to each of the cells?
  - How will you handle affinity of requests?
  - How will you handle session data?
  - How will you handle security data?
  - How will you address the data replication and consistency challenge?
  - How will you handle a cell failover for each type of requests in your application? These types include web requests, SIP requests, EJB requests, and web services.
- Possibility of collocation of cells

  You can collocate the two cells on the same system to achieve the software release independence described previously. However, the collocation limits the usability of disaster recovery.

Considerations

For the multi-cell topology, consider the increased costs in hardware, complexity, configuration, and administration. However, you need to consider these costs in relation to the gains in performance, throughput, and reliability. You probably have specific requirements you need to consider in such an architecture.

Remember: Because this topology is a combination of topologies described previously, the multi-cell topology also has the considerations of those base topologies.

Setting up this topology

The steps for this topology are the same as for a topology with redundancy of multiple components as shown in “Setting up the topology” on page 221.

8.3.11 Advanced topology using an administrative agent

Starting with WebSphere Application Server V7, an additional administration component is available called the *administrative agent*. The administrative agent is intended to reduce the administration costs of large WebSphere deployments. A single server installation is sufficient for several installation scenarios built according to the requirements. The issue of having multiple stand-alone application servers in different environments (for example, development, test, quality assurance, or production) is that they all lack a common administrative interface.

An administrative agent provides a single interface to administer multiple unfederated application server nodes in such environments. The administrative agent and application servers must be on the same system. However, you can connect to the system from a browser or the **`wsadmin`** tool on another system.

Restriction: You can register an application server node with the administrative agent or federate the node with a deployment manager, but you cannot do both.

In addition, a DMZ proxy does not work with the administrative agent when security is enabled. Keep security enabled, and do not use the administrative agent in a DMZ proxy environment.

Figure 8-13 shows a possible topology that uses an administrative agent to manage all of the single server installations on System B. Instead of running the Configuration service, the administrative console application, and so on, in each application server, these services are running in the administrative agent for all profiles. The administrative agent, therefore, reduces the administrative tasks for the installation and simplifies the administration, because all administrative access uses one central point. Therefore, you have one URL for the administration instead of several.

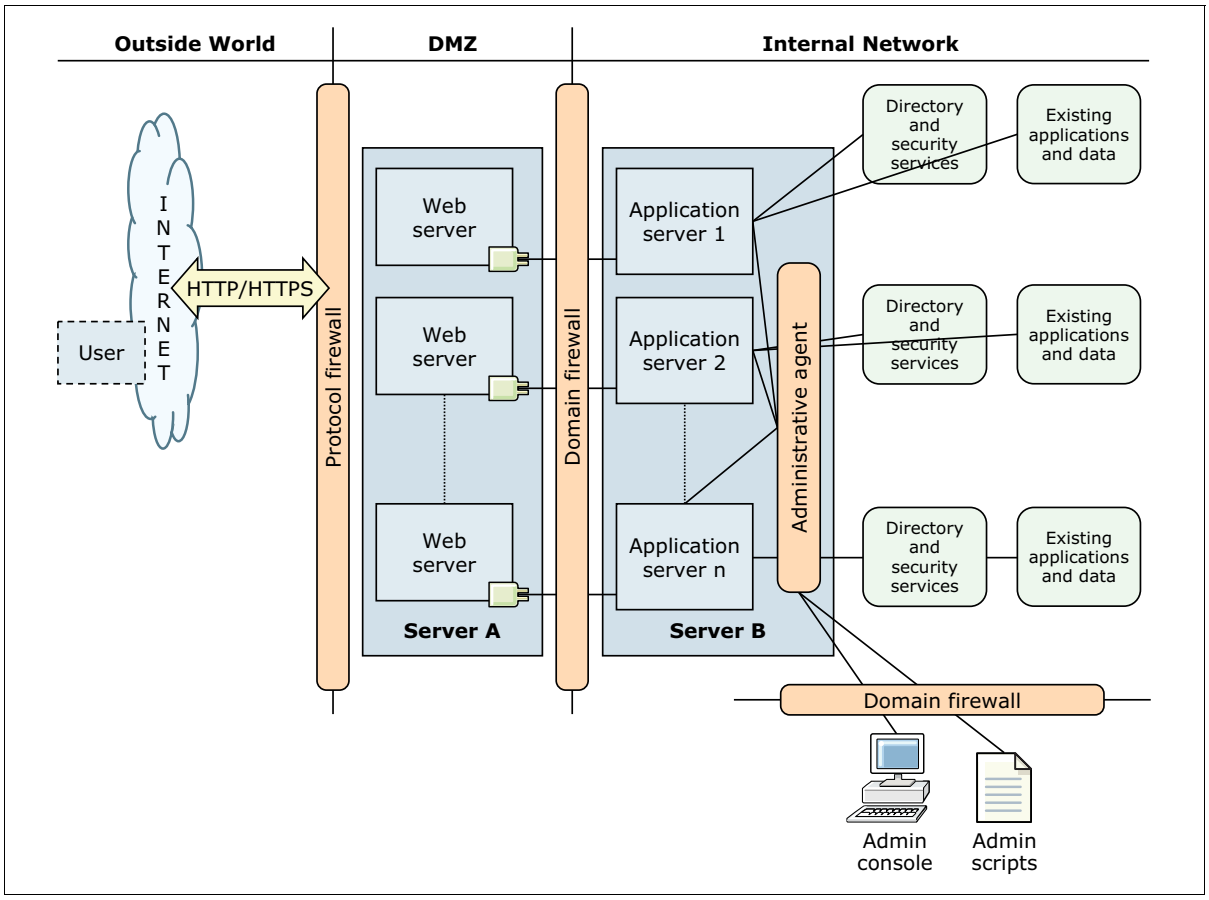


Figure 8-13 Topology containing an administrative agent

Advantages

The implementation of an administrative agent profile has the following benefits for the installation:

- Reduced administrative footprint

  The administrative footprint is reduced when using multiple, single server installations in the same system.

- Central access to administration tools

  Access to the administrative tools (the administrative console or **wsadmin**) is simplified, because all access is through the administrative console. Only one URL is used instead of multiple URLs.

- Less firewall ports required

  If a firewall is between the administrator's workstation and the servers, fewer ports need to be opened on the firewall. Instead, you need accessibility only to the administrative agent to each application server.

Considerations

The implementation of an administrative agent profile requires an additional JVM. The additional JVM runs on the system, requiring multiple single servers to manage the administrative agent to avoid an increased memory footprint for the overall solution.

Setting up the topology

To set up an environment as illustrated in Figure 8-13 on page 228, complete the steps in the following sections.

Setting up System A

To set up System A, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install the following applications:
   - Web server plug-ins for WebSphere Application Server
   - WebSphere Customization Toolbox
   - IBM HTTP Server

   If you are not using IBM HTTP Server, install a supported web server.
3. Open the WebSphere Customization Toolbox, and start the Web Server Plug-ins Configuration Tool.
4. Configure the web server plug-in, and create the web server definition. For more information, see the Websphere Application Server V8.5 Information Center at:

   `http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.doc/ae/twsv_plugin.html`

   Remember: This topology has multiple ways for setting up the web server. You can run one instance of a web server, multiple instances of the same web servers, or multiple installations if you are using different web servers. Make sure that each web server plug-in points to the correct application server on System B.

Setting up System B

To set up System B, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server V8.5.
3. Create as many application server profiles as needed by using the *app_server_root*`/profileTemplates/default` profile template. These profiles are the single server profiles that run your applications.
4. To create an administrative agent profile, create an application server profile by using the *app_server_root*`/profileTemplates/management` profile template, and specify `-serverType` for ADMIN_AGENT.
5. Go to the binary directory of your administrative profile, and register each single server profile to the administrative agent (**`registerNode`**) as follows:

   `registerNode -profilePath` *user_data_root*`/profiles/AppSrv01`
6. Open the administrative console or a **`wsadmin`** session to the administrative agent, and select the application server you want to manage.
7. For each single server installation, create a web server definition as needed for your environment.

8.3.12 Multi-cell star topology using Intelligent Management

The Intelligent Management features allow you to create dynamic clusters. In a dynamic cluster, the cluster members are created, started, and removed automatically by the Intelligent Management capabilities. To achieve better scalability, you can separate the cell that runs the on-demand router from the cells that run the applications.

If these cells share hardware, run the application placement controller only in the cell that is running the on-demand router. This configuration is achieved by setting a custom property for the cell. In the example illustrated in Figure 8-14, the on-demand router and the application cells do not share hardware.

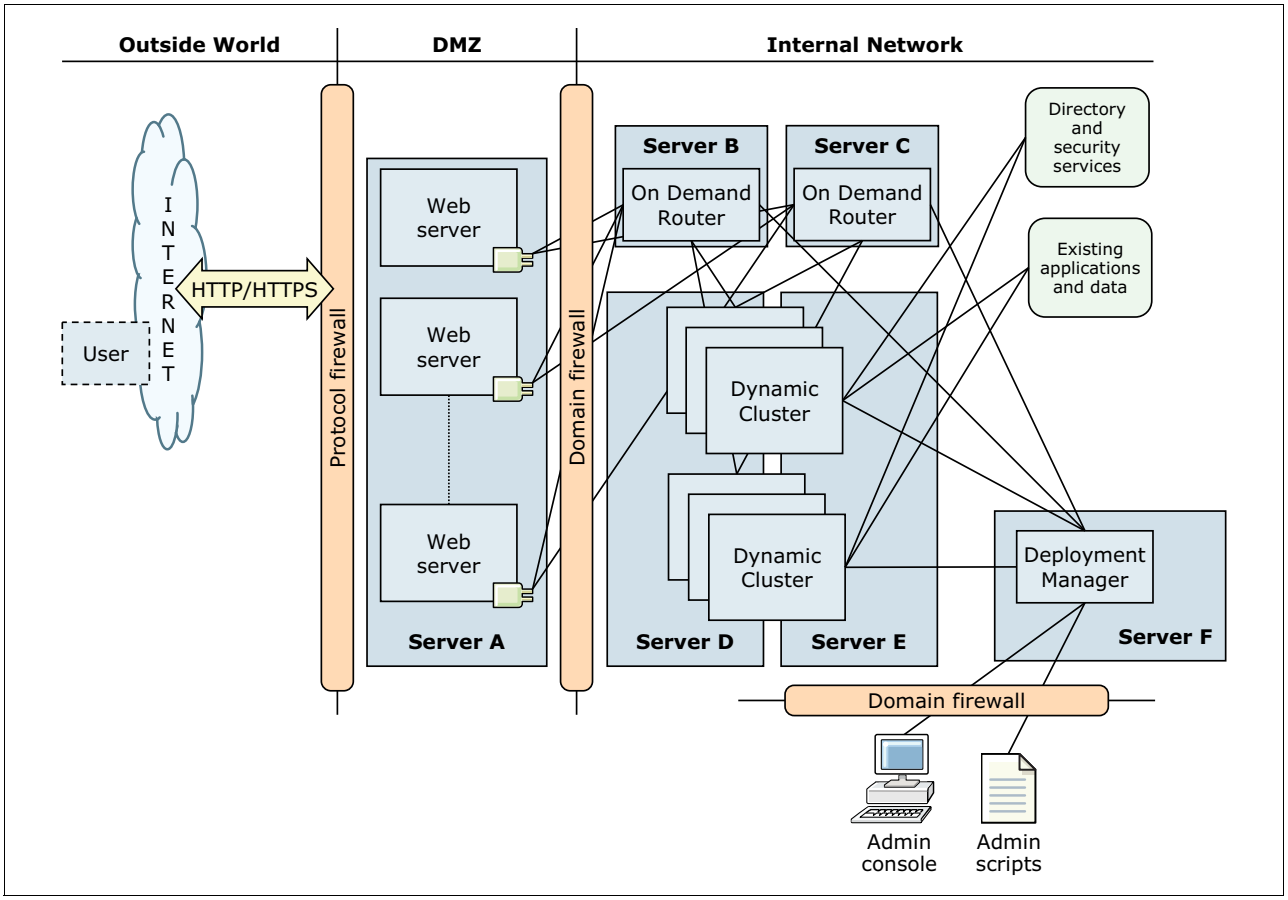


Figure 8-14 The Intelligent Management multi-cell star topology

Advantages

Using a multi-cell star topology with the Intelligent Management function has the following advantages:

- High availability

  All components in this topology are redundant. The dynamic clusters create multiple nodes, so there is no SPOF. The on-demand routers fail over to other cluster members if one of the application nodes fails. If necessary, even a second deployment manager can be created.

- High efficiency

  The Intelligent Management feature starts only the number of cluster members that are necessary to provide the service that is currently requested. Thus, processor time and memory is saved, and can be used by other services if required.

- Flexible performance, according to requirements

  Service policies for different applications can be defined independently. The Intelligent Management feature always tries to provide the service as defined. It warns you if the available hardware cannot provide the requested service level.

- Low maintenance effort

  Because the Intelligent Management feature starts and stops the resources as needed, you need to define only the service policies. After you define the policies, the maintenance effort for the infrastructure is low. The option to install and run multiple versions of the same application helps when you need to update an existing application.

Considerations

The on-demand routers might require additional hardware.

Setting up the topology

Use the following steps to set up the multi-cell star topology.

Setting up System A

To set up System A, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install the following applications:
   - Web server plug-ins for WebSphere Application Server
   - WebSphere Customization Toolbox
   - IBM HTTP Server

     If you are not using IBM HTTP Server, install a supported web server.
3. Open the WebSphere Customization Toolbox, and start the Web Server Plug-ins Configuration Tool.
4. Configure the web server plug-in, and create the web server definition. For more information, see the Websphere Application Server V8.5 Information Center at:

   `http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.doc/ae/twsv_plugin.html`

Setting up the deployment manager (System F)

To set up the deployment manager on System F, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile:
   - Use the *`app_server_root`*`/profileTemplates/dmgr` profile template.
   - Use the *`app_server_root`*`/profileTemplates/management` template, and specify `-serverType` for DEPLOYMENT_MANAGER.
4. Create web server definitions through the administrative console or through the **`wsadmin`** scripting interface.

Setting up the on-demand routers (Systems B and C)

To set up the on-demand routers for Systems B and C, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server V8.5.
3. Create one on-demand router profile per system by using the *app_server_root*/profileTemplates/default profile template.
4. Define the on-demand routers on the deployment manager by using the administrative console or through the **wsadmin** scripting interface.

Setting up the dynamic clusters (Systems D and E)

To set up the dynamic clusters on Systems D and E, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile:
   - Use the *app_server_root*/profileTemplates/managed profile template. Then federate this profile to the deployment manager on System F during profile creation or after the profile creation by running the **addNode** command.
   - Use the *app_server_root*/profileTemplates/default profile template. Then federate the node to the deployment manager that runs on System F by using the **addNode** command.
4. Define the dynamic clusters on the deployment manager by using the administrative console or through the **wsadmin** scripting interface.

8.3.13 Advanced topology using a job manager

The job manager is an administration feature that addresses scalability issues of the administrative run time when the components are spread over multiple remote locations. An example of such a deployment is a typical branch deployment where central management is wanted, but the nodes themselves are in branch locations.

The job manager uses a loosely coupled asynchronous administration model to manage several remote endpoints. The job manager introduces different administrative options and flexibility to set up a centralized administration model. In WebSphere Application Server V8.5, you can now complete job manager actions and run jobs from a deployment manager. The deployment manager administrative console has jobs navigation tree choices similar to those choices in the job manager administration console.

The topology illustrated in Figure 8-15 shows how to use the job manager to centrally administer multiple heterogeneous environments.

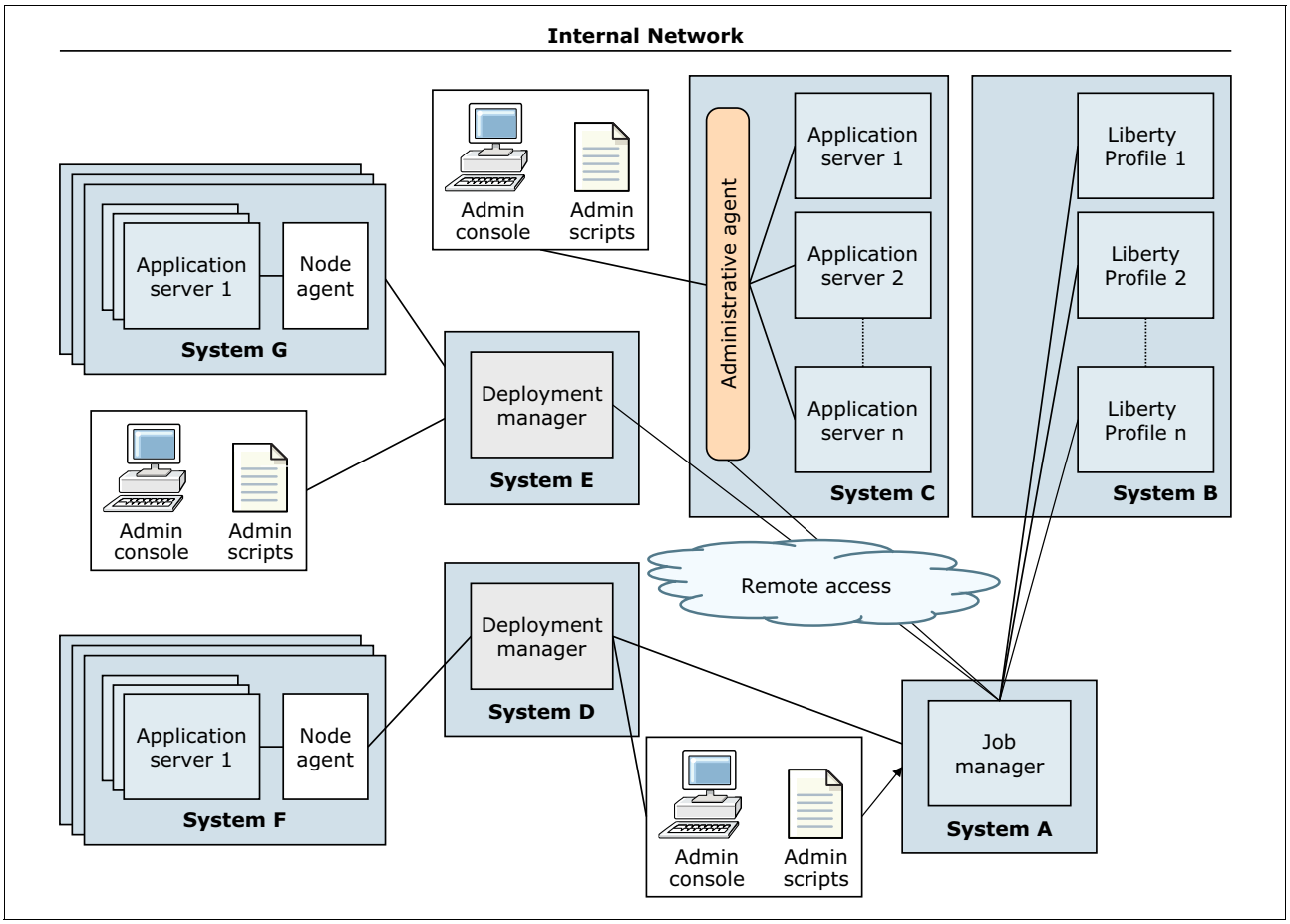


Figure 8-15 Topology using a job manager

The job manager node on System A acts as a coordinator across multiple deployment managers (System H and System D) and administrative agents (System B and System C). It does so through its asynchronous job management capabilities. The job manager is *not* a replacement for deployment managers or administrative agents. The job manager relies on the local management capabilities to run the management jobs.

Advantages

Running a job manager in your environment provides the following advantages for the administration of your deployments:

- Allows central, remote management of multiple different administrative entities through wide area networks (WANs)
- Allows local and remote management of each installation
- Enhances existing management models

Considerations

The job manager does not have any real disadvantages, except that you need an additional JVM (namely the *jobmgr* application server) running.

Setting up the topology

To set up the environment illustrated in Figure 8-15 on page 233, complete the steps in the following sections.

Setting up System A

To set up System A, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. To create the job manager profile, create an application server profile by using the *app_server_root*/profileTemplates/management profile template, and specify -serverType for JOB_MANAGER.

Setting up Systems B

To set up Systems B and C, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create as many application server profiles as needed by using the *app_server_root*/profileTemplates/default profile template as required. These profiles are the single server profiles that are running your applications.
4. To create the administrative agent profile, create an application server profile by using the *app_server_root*/profileTemplates/management profile template. Then specify -serverType for ADMIN_AGENT.
5. Go to the binary directory of your administrative profile, and register each single server profile to the administrative agent (**registerNode**).
6. Use **wsadmin** in the binary subdirectory of the administration agent profile directory, and register the administrative agent with the job manager by running the **AdminTask.registerWithJobManager** task.

Setting up System C

To set up System C, complete the following steps:

1. Download the WebSphere Liberty profile compressed file.
2. Add your server configuration and applications to the file. You can create multiple compressed files that contain different configurations and applications.
3. In job manager, define each system that runs a Liberty profile as a target.
4. Submit an **Install Liberty profile server resources** task, and specify the compressed file for the target. Job manager deploys the Liberty profiles.
5. Run the **Start Liberty profile server** task to start the Liberty profile servers.

Setting up Systems D and E

To set up Systems D and E, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.

3. Create an application server profile:
   - Use the *app_server_root*/profileTemplates/dmgr profile template.
   - Use the *app_server_root*/profileTemplates/management template, and then specify -serverType for DEPLOYMENT_MANAGER.
4. Register the deployment manager with the job manager:
   - Use **wsadmin** in the binary subdirectory of the deployment manager profile directory, and register the deployment manager with the job manager by running the **AdminTask.registerWithJobManager** task.
   - Register with the job manager by using the administrative console. In the deployment manager console, click **System Administration** → **Deployment manager** → **Job manager**, select a deployment manager node, and click **Register with Job Manager**. The deployment manager nodes that you register with the job manager become the managed nodes of the job manager.

Setting up Systems F and G

To set up Systems F and G, complete the following steps:

1. Install IBM Installation Manager.
2. Using Installation Manager, install WebSphere Application Server Network Deployment V8.5.
3. Create an application server profile:
   - Use the *app_server_root*/profileTemplates/managed profile template. Federate this profile to the correct deployment manager during profile creation, or after the profile creation by running the **addNode** command.
   - Use the *app_server_root*/profileTemplates/default profile template. Then federate the node to the correct deployment manager by using the **addNode** command.

9

Installation planning

This chapter provides general guidance for planning the installation of WebSphere Application Server V8.5 and many of its components. To effectively plan an installation, you need to select a topology, hardware, and operating system, and prepare your environment for WebSphere Application Server installation.

Prerequisites: For more information about prerequisite for WebSphere Application Server V8.5, see System Requirements for WebSphere Application Server Base and Network Deployment V8.5 at:

http://www-01.ibm.com/software/webservers/appserv/was/network/requirements/

This chapter includes the following sections:

- Installation features in WebSphere Application Server V8.5
- Selecting a topology
- Selecting hardware and operating systems
- Planning for disk space and directories
- Naming conventions
- IBM Installation Manager
- Planning for WebSphere Application Server
- Planning for the Liberty profile
- WebSphere Customization Toolbox
- Planning for Edge Components
- Planning for the DMZ secure proxy
- Planning for the HTTP server and plug-in
- IBM Support Assistant
- Installation checklist
- Resources

This chapter does not explain how to install WebSphere Application Server V8.5. For more information about installing Websphere Application Server V8.5, see the Websphere Application Server V8.5 Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp

9.1 Installation features in WebSphere Application Server V8.5

The following installation options are available for the WebSphere Application Server V8.5 editions for *distributed systems*:

- IBM Installation Manager

  IBM Installation Manager is an Eclipse-based installation management tool that runs product installations, updates, and uninstallations with integrated prerequisite and interdependency checking. The Installation Manager includes the following concepts:

  - A *package* is a separately installable product that can be installed by Installation Manager. It can operate independently, or it can be dependent on other packages. For example, IBM WebSphere SDK Java Technology Edition V7.0 cannot be installed unless WebSphere Application Server V8.5 is installed.
  - A *repository* is the place where the packages to be installed can be found. The repository includes metadata and product binary files.

  With Installation Manager, you can perform these tasks:

  - Identify product and maintenance packages
  - Install packages after prerequisite and interdependency checking
  - Add or remove optional features from an installation
  - Uninstall and roll back previously installed packages

  WebSphere Application Server V8.5 products and related components are installed, modified, or updated by using remote or local repositories:

  - Application Client for IBM WebSphere Application Server
  - Edge Components V8.5
  - IBM HTTP Server V8.5
  - Pluggable Application Client for IBM WebSphere Application Server
  - WebSphere Customization Toolbox
  - WebSphere DMZ Secure Proxy Server
  - Web server plug-ins
  - IBM WebSphere SDK Java Technology Edition V7.0
  - WebSphere Application Server Web 2.0 and Mobile Toolkit

- Centralized installation manager

  The centralized installation manager is used to install and apply maintenance of WebSphere Application Server and its supplementary products on remote computers. It has supported Installation Manager since WebSphere Application Server V8. You can use the centralized installation manager to perform these tasks by using either the administrative console or the **`wsadmin`** tool:

  - Install Installation Manager instances
  - Update Installation Manager with a repository
  - Manage Installation Manager offerings

  You access centralized installation manager through the deployment manager or job manager. Using the centralized installation manager, you can perform the following functions:

  - Install, update, and uninstall Installation Manager on remote systems.
  - Install, update, and uninstall WebSphere Application Server V8.5 and previous versions of WebSphere Application Server on remote systems.
  - Collect, distribute, and delete files on remote hosts.
  - Manage WebSphere Application Server V8.5 profiles on remote hosts.

- Run scripts on remote hosts.
- Add targets outside the cell.
- Schedule jobs.

The benefit of using centralized installation manager is you can manage a number of installations at the same time from the job manager or deployment manager.

Remember: These functions apply to installation of WebSphere Application Server V8.0 or later only. There is another version of centralized installation manager that can be used to install WebSphere Application Server V7. It is located under the System Administration area of the deployment manager in the administrative console.

The WebSphere Application Server Liberty profile is a lightweight profile of the application server, along with a simplified configuration approach for the development environment. When you install WebSphere Application Server V8.5 using the Installation Manager, you have the choice of installing the full profile or the Liberty profile. For more information about installing the Liberty profile, see 9.8, “Planning for the Liberty profile” on page 268.

The following installable features are available with the Websphere Application Server V8.5 packages:

- IBM WebSphere SDK Java Technology Edition

  WebSphere Application Server provides support for the IBM WebSphere SDK Java Technology Edition V 7.0 as an optional pluggable Java development kit (JDK). Java 6 is installed with the product and used by default. You can install IBM WebSphere SDK Java Technology Edition V7.0 by using the Installation Manager. Use the **managesdk** tool to optionally enable Java 7. You can also switch between using Java 6 or Java 7 to best meet your business needs.

- WebSphere Customization Toolbox V8.5

  The WebSphere Customization Toolbox for WebSphere Application Server V8.5 includes tools for managing, customizing, and migrating various parts of the WebSphere Application Server environment. The WebSphere Customization Toolbox is available as two different offerings, and each offering has various combinations of tools on different platforms.

9.2 Selecting a topology

Chapter 8, “Topologies” on page 179, provides information about common configurations. Each topology description contains information about the software products required and the information needed to create the runtime environment for WebSphere Application Server.

After you identify the topology that best fits your needs, map the components from that topology to a specific hardware and operating system. Plan for the installation of the required products.

9.3 Selecting hardware and operating systems

After you select a topology, decide which platforms you will use to map the topology to a specific hardware. These selections are driven by several factors:

- Existing conditions
- Future growth
- Cost
- Skills within your company

When choosing the platform or platforms, determine the configuration of each server by considering the following aspects:

- Processor features
- The amount of memory
- The number of direct access storage device (DASD) arms
- The amount of storage space that is required

Along with selecting the hardware, you must select the operating system. The operating system must be at a supported version with a correct maintenance level installed before WebSphere Application Server works properly and gets support. Keep in mind that not every product you receive with WebSphere Application Server V8.5 is supported on each operating system and platform.

For an updated list of the hardware and software requirements and supported platforms for WebSphere Application Server V8.5, see System Requirements for WebSphere Application Server Base and Network Deployment V8.5 at:

`http://www-01.ibm.com/software/webservers/appserv/was/network/requirements/`

9.4 Planning for disk space and directories

Before you install WebSphere Application Server components, you must provide sufficient disk space for successful installation and for operation of the environment.

Terminology: In this section, the term *file system* is a synonym for *manageable disk storage*. File system can refer to file systems on UNIX technology-based systems, disk partitions, and drive letters on Windows, hierarchical file system (HFS), or zSeries file system (zFS) for z/OS.

Although WebSphere Application Server products provide a default directory structure for their components, they might not be the best choice. The default structure might limit the flexibility or become inconsistent in terms of naming. Keep in mind that your directory names are bound to the naming rules.

You can use one file system, following the default directory structure, or create multiple file systems by using different mount points. Generally, disk space management is more flexible and efficient if you split the installation into different file systems. When planning your directory structure and file systems, consider other criteria such as performance, backup requirements and capabilities, availability, and security.

Different file systems can be designated for the following files:

- Application binary files

  This file system stores the product binary files as dumped by the installer. When designing this file system, keep in mind that installing maintenance causes this file system to grow.

- Profiles

  This file system stores the profile-specific data that defines your runtime environment. The minimum disk space required depends on the profile type that you create. The amount of user data needed depends on the applications that are deployed to the profile.

- Log files

  The purpose of this file system is to hold the log files of the application servers. If this file system is not mounted under the default mount point, you must change the server configuration for each server. You can change the server configuration by using scripting or a custom server template.

  The size of the log files depends on the application and on the log retention policy of the application servers.

- Dump files

  System core dumps and Java heap dumps can be large and quickly fill a file system. Therefore, redirect the system memory dumps, Java heap dump files, and the Java core dump files to a dedicated directory. This approach prevents a dumping process or Java virtual machine (JVM) from filling up file systems and impacting other running applications. It also allows you to locate the files easily.

  The size of the dump files depends on the following factors, among others:

  - The number of JVMs dumping to this directory
  - The individual sizes of the dump files
  - The number of dump files that you want to retain

- Maintenance packages or Installation Manager repositories

  Installable packages are maintained in Installation Manager repositories. The centralized installation manager uses Installation Manager to install WebSphere Application Server and WebSphere Application Server maintenance to remote hosts. Those hosts must have access to Installation Manager repositories with the required content.

  Remember: The centralized installation manager also keeps a directory of Installation Manager installation kits on the job manager. To install Installation Manager, the centralized installation manager pushes the Installation Manager installation kit to a target system and installs it.

- User data and content

  Use this file system to store other user data and content that is used in the applications.

9.5 Naming conventions

Naming conventions make the runtime environment more comprehensible. A consistent naming convention helps standardize the structure of the environment and allows for easy expansion of the environment and each component.

Develop, establish, and maintain naming conventions for the hardware and networking infrastructure as well as the WebSphere Application Server infrastructure, applications, and

resources. When it comes to naming, most companies have already developed a working naming convention for their existing infrastructure. It is best to adhere to the existing convention instead of trying to invent a new one specific to WebSphere.

Because naming conventions are also related to many different aspects of a company, these conventions vary depending on the characteristics of the environment. With a useful naming convention, you can understand the purpose of an artifact just by looking at its name.

When you develop a naming convention, consider which hardware and software components are affected and what naming restrictions apply. On many systems, naming restrictions exist in terms of specific characters and length of names. In a heterogeneous environment, such restrictions might become a pitfall. Generally, avoid any special or national language-specific characters in the names.

9.6 IBM Installation Manager

WebSphere Application Server V8.5 is installed by the IBM Installation Manager. The Installation Manager is based on the following architectural principles:

- Product independence

  Installation Manager is independent from the product it installs or maintains.

- Product repositories

  Product binary files are in repositories to which Installation Manager can connect. They acquire all required binary files that are relevant to user selections and the system environment.

9.6.1 Benefits of Installation Manager

Using Installation Manager provides the following benefits:

- Full product lifecycle management operations, including the following operations:
  - Install
  - Update
  - Modify
  - Rollback
  - Uninstall
- A single tool for all WebSphere Application Server platforms
- A consistent user experience across multiple IBM products
- The following methods for performing lifecycle management activities:
  - A graphical user interface
  - A command-line interface
  - An interactive text-based interface
  - A response file

  Restriction: The interactive text-based interface or console mode is currently available only with WebSphere Application Server V8.5 z/OS. Installation Manager supports console mode for all platforms. However, WebSphere Application Server supports it only on z/OS.

- Easily downloadable and installable code for a number of IBM software packages

- One-pass installation
  - Install the level of service wanted without multiple steps to install GA level and updates
  - Can install multiple products
- Validation and system checking completed before downloading binary files:
  - System parameters are checked based on user-defined metadata
  - Relationship checking among products and fixes is available for installation
- Downloads from the repository and installs only binary files relevant to user selections and the system environment
- Better handling of optional installable features:
  - Going back to the base version of the product to enable an optional feature is no longer necessary
  - Optional installable features can be selected during the fix pack installation or from the modify option
- Better management of files for rollback

9.6.2 Installation Manager repositories

Installation Manager enables flexible installation scenarios with an enterprise. Each software package that can be installed with Installation Manager is known as a *package*. An installed package has a product level and an installation location. A *package group* consists of all of the products that are installed at a single location. Packages are stored in repositories, which are flat files.

Installation Manager repositories can be exposed to enterprise users in one of the following ways:

- Local Installation Manager repository
- IBM hosted repository
- Enterprise hosted custom repository

The IBM Packaging Utility is a packaging tool that creates and manages software repository content in the correct format for IBM Installation Manager. With the IBM Packaging Utility, you can manage repository content. You can use CD images, PPA compressed files, IBM hosted repositories, or any other Installation Manager repository as sources for creating a custom repository by using IBM Packaging Utility.

With the IBM Packaging Utility, you can perform the following tasks:

- Create a repository
- Copy multiple packages into a repository
- Copy multiple versions of a product to a repository
- Delete packages
- Add fixes into a repository

The IBM Packaging Utility has both a GUI and command-line interface. WebSphere Application Server V8.5 depends on IBM Packaging Utility V1.5.2.

Remember: Installation objects, including product installations, updates, and fixes, that can be manipulated or used by Installation Manager are known as *software packages*.

The Installation Manager and Packaging Utility works with fix packs and interim fixes. Fix packs are cumulative updates to a software package. A fix pack implies that a new version of

the software package is available. Interim fixes apply to a specific version of a software package and typically fix a single or several small critical issues.

The IBM Packaging Utility V1.5.2 includes a **-platform** option that allows you to copy repository files for a specific platform or architecture. You can save disk space and network bandwidth by copying only the files that you require. To use the **-platform** option, install the IBM Packaging Utility and use the **pucl** command-line interface. On the command line, indicate the **os** attribute to specify the operating system. You use the **arch** attribute to specify the architecture. These settings identify only the files for the operating system and architecture that you need.

When using **pucl**, you can create a repository for installation on Linux PPC by using the following command:

```
PUCL.exe copy com.ibm.Websphere.ND.v80_8.5.0.20120222_0247 -repositories
http://ibm.com/repository/package -target "/IBM/IBMPackages/" -platform
os=linux,arch=ppc -acceptLicense
```

Tip: IBM Packaging Utility V1.5 and V1.5.1 include the **-platform** option as a technology preview.

For more information about the values that can be used for the *os* attribute and the *arch* attribute, see:

http://www-01.ibm.com/support/docview.wss?uid=swg27023080

For more information about creating custom installation repositories for WebSphere by using the IBM Packaging Utility, see:

http://www.ibm.com/developerworks/websphere/library/techarticles/1201_seelemann/1201_seelemann.html

The IBM Packaging Utility V1.5.2 has a technology preview that includes the **-updateFromVersion** option used to specify the version of the package that you are updating from. This option allows you to copy only the files that are required to update to a later version. For more information about the technical preview, see:

http://www-01.ibm.com/support/docview.wss?uid=swg27023960

Local repository

Figure 9-1 shows a local Installation Manager repository on the same server where Installation Manager is installed. This server is where installations can be performed.

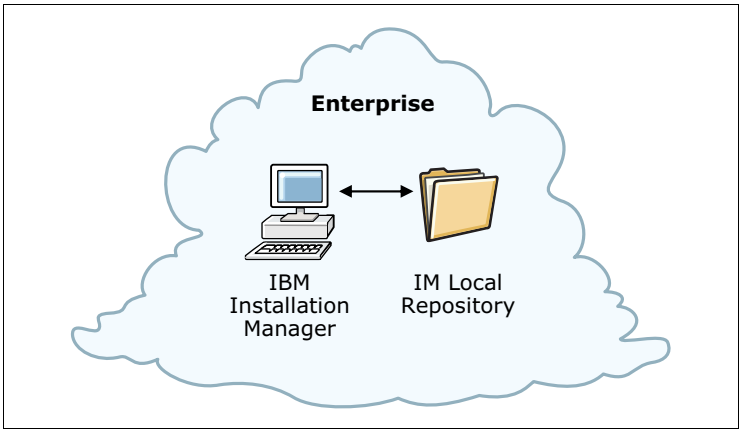


Figure 9-1 Local Installation Manager repository

IBM hosted repository

The Installation Manager can also download packages from IBM hosted repositories, such as IBM Passport Advantage®, as shown in Figure 9-2.

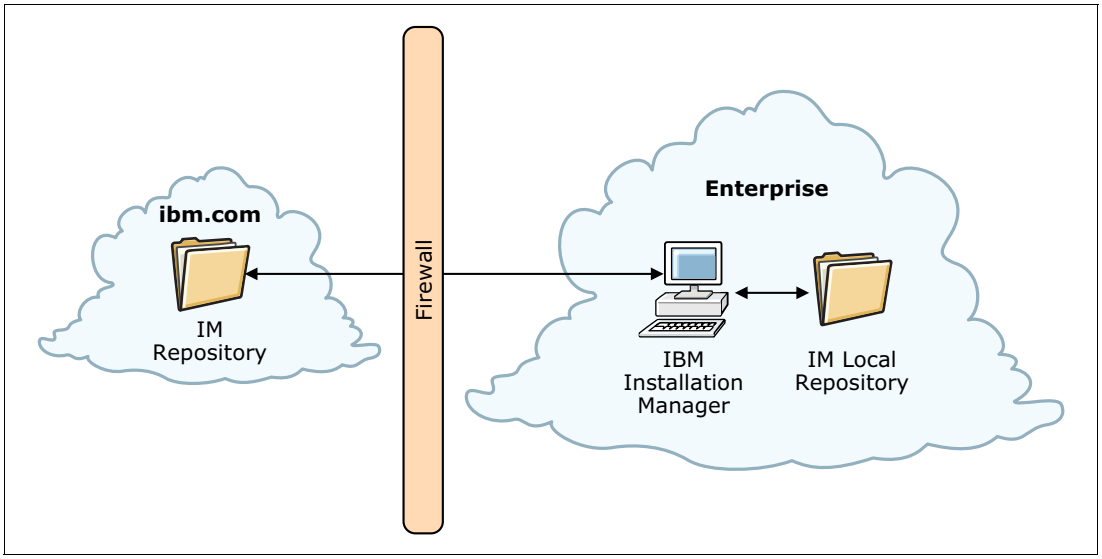


Figure 9-2 IBM hosted repository

Enterprise hosted custom repository

An enterprise hosted custom repository contains software packages and package groups that are shared with one or more computers within the enterprise. A typical workflow, as shown in Figure 9-3 on page 246, might involve the following tasks:

1. Download images from an IBM hosted repository or other repositories to a local computer with the Packaging Utility.

   Images can include compressed CD installation images that contain software package repositories, including software packages such as WebSphere Application Server.

2. Copy software packages to a local repository or an enterprise repository.

   After the compressed files are extracted, IBM Packaging Utility is used to copy the software package repositories to a local or enterprise repository.

3. Use Installation Manager to install the software package from the local or enterprise repository.

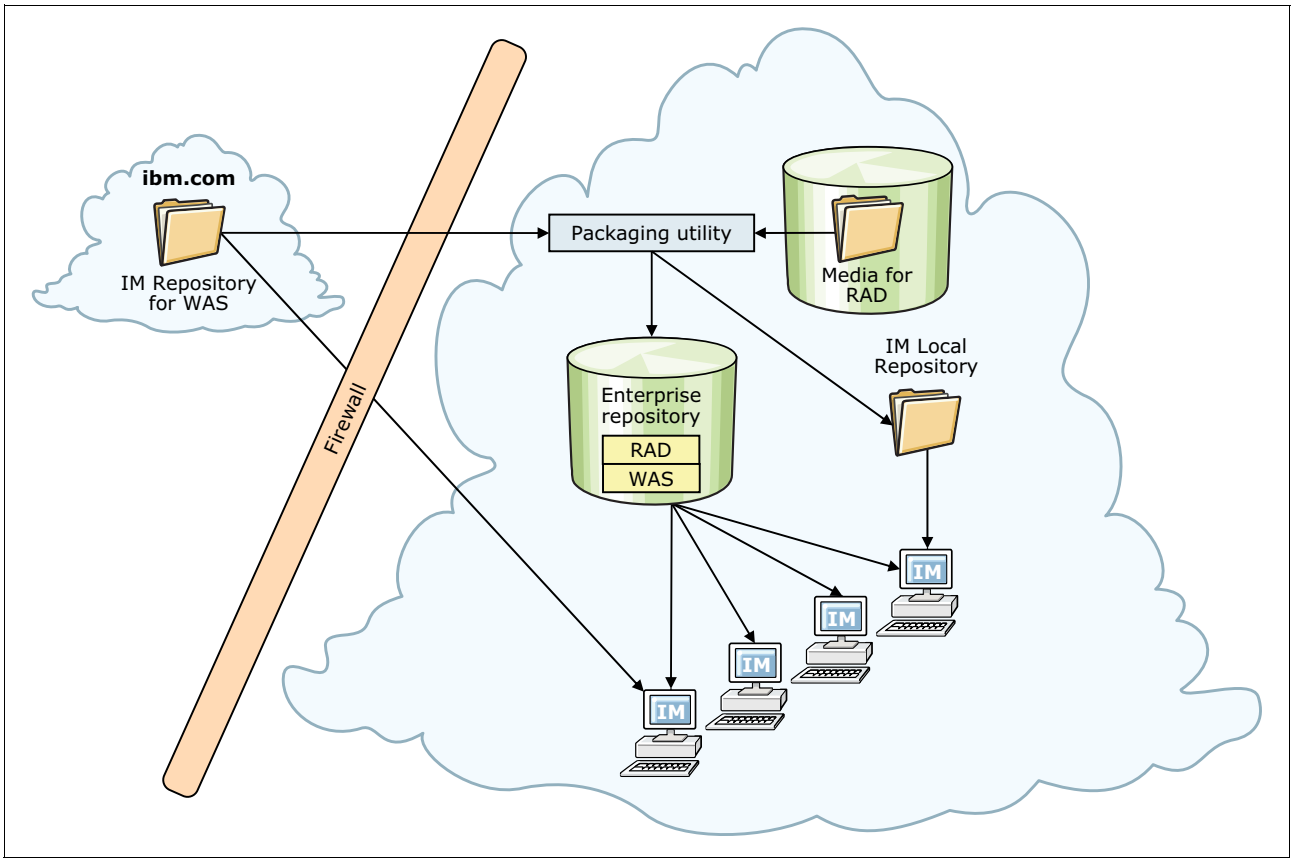


Figure 9-3 Enterprise hosted custom repository

Tip: For more information about IBM Installation Manager, see the IBM Installation Manager Information Center at:

`http://publib.boulder.ibm.com/infocenter/install/v1r5/index.jsp?top`

9.7 Planning for WebSphere Application Server

WebSphere Application Server V8.5 entails a full product installation, not an upgrade installation. Consider the best installation process to use based on the number of systems and the complexity of the installations.

The WebSphere Application Server Information Center contains planning topics for all WebSphere Application Server packages divided by platform supported. Be sure to review this documentation. This section provides a high-level view of the planning tasks that you need to perform.

Tip: For information about installation considerations for WebSphere Application Server V8.5 for z/OS, see 16.3, “Installing WebSphere Application Server for z/OS” on page 526.

Before you start installing WebSphere Application Server, you must address the following items, which are explained in more detail throughout this section:

- File systems and directories

  When installing Websphere Application Server, select the file systems on which you want to install the product. Also select where you want to store your runtime environment, logs, and so on.

- Single installation or multiple installations

  The standard installation is to install WebSphere Application Server once per system and to create multiple runtime environments by using profiles. Each profile has its own configuration data, but shares the product binary files. In some instances (such as for test environments), and depending on your chosen topology, you might want to install multiple instances.

- Installation method

  You have multiple options for the installation. Your choice is influenced by several factors:

  – The size of the installation (the number of systems)
  – The operating systems involved
  – The number of times you anticipate performing the same installation (by using a GUI or performing a silent installation)
  – If you are performing remote installations with unskilled personnel

  For some environments, a silent installation is the only method available because graphic libraries are not allowed for security reasons.

- Installing updates

  To apply maintenance to Websphere Application Server, you need IBM Installation Manager.

- Profile creation

  The environment is defined by creating profiles. You must determine the types of profiles that you need and on which systems you need to install them.

- Naming convention

  Naming conventions can be an important management tool in large environments. Naming not only makes it easier to understand the environment, but having a consistent naming convention in place is helpful when you write scripts.

- TCP/IP port assignments

  Each type of server (such as deployment manager, node agent, or application server) uses a series of TCP/IP ports. These ports must be unique on a system and must be managed properly. The port assignments are essential to avoid port conflicts if you are planning for multiple installations and profiles.

- Security considerations

  Security for WebSphere falls into two categories: *administrative security* and *application security*. During the profile creation, you can enable administrative security. Plan a scheme for identifying administrative users, their roles, and the user registry that you use to hold this information.

- IBM Support Assistant Agent

  The IBM Support Assistant Agent is an optional feature that allows remote troubleshooting (such as remote system file transfer, data collections, and inventory report generation).

9.7.1 File systems and directories

WebSphere Application Server uses a default file system structure to store the binary files and the runtime environment unless specified otherwise. Review the default directory structure, and decide whether this satisfies your needs. For more information, see 9.4, “Planning for disk space and directories” on page 240.

9.7.2 Single installation or multiple installations

You can install WebSphere Application Server V8.5 multiple times on the same system in different directories. You can also install WebSphere Application Server V8.5 in parallel to older versions of WebSphere Application Server on the same system. These installations are independent of each other. If a fix is applied on a particular installation, it affects only that specific WebSphere Application Server installation, leaving the remaining installations on that system unaffected. You do not have to stop the other installations while applying fixes to a specific installation.

When you have a single installation of WebSphere Application Server V8.5, you can create multiple application server profiles. In this case, all profiles share the product binary files. Therefore, you must stop all application server JVMs for all profiles before installing fixes. When fixes are installed, they affect all profiles. Each profile has its own user data.

Figure 9-4 shows the difference between multiple installations and multiple WebSphere profiles in a stand-alone server environment.

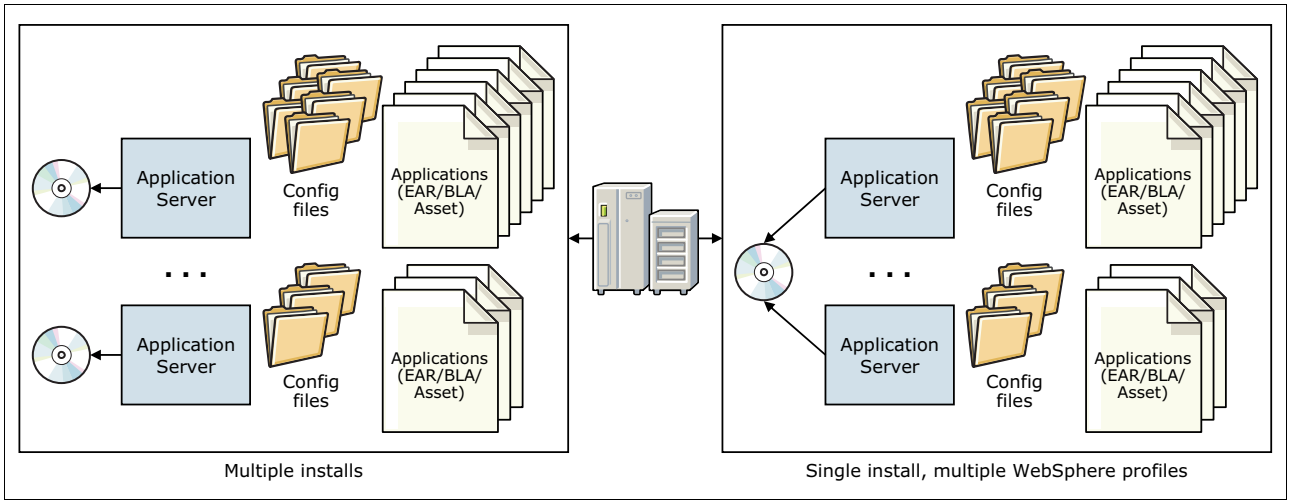


Figure 9-4 Stand-alone server installation options

Consideration: There is no architectural limit for multiple installations or multiple profiles. The real limitation depends on the hardware capacity and licensing.

The same logic holds true for Network Deployment installations. You can install the product several times on the same system (multiple installations), each one for administering different cells. Alternatively, install Network Deployment one time and create multiple profiles so that each profile is used to administer a different cell (Figure 9-5).

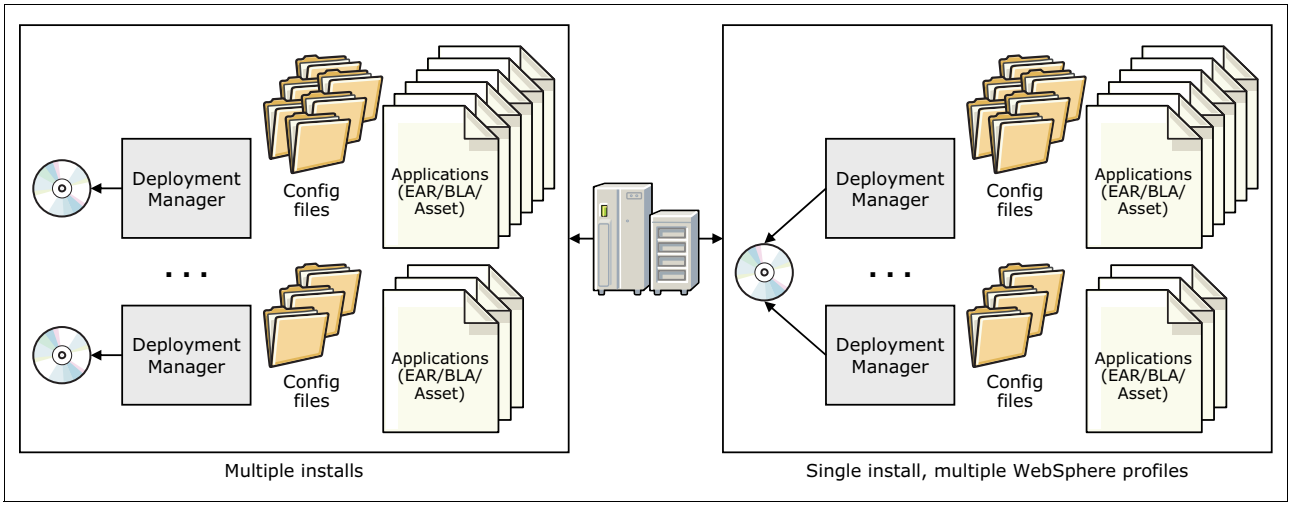


Figure 9-5 Deployment manager installation options

Another possibility is the combination of multiple installation instances and multiple profiles. Figure 9-6 illustrates a Network Deployment environment where multiple runtime environments were created by using profiles. In Figure 9-6, cell 1 contains a deployment manager and application server on separate systems that uses separate installation instances. Cell 2 contains a deployment manager and two application servers that span three installation instances.

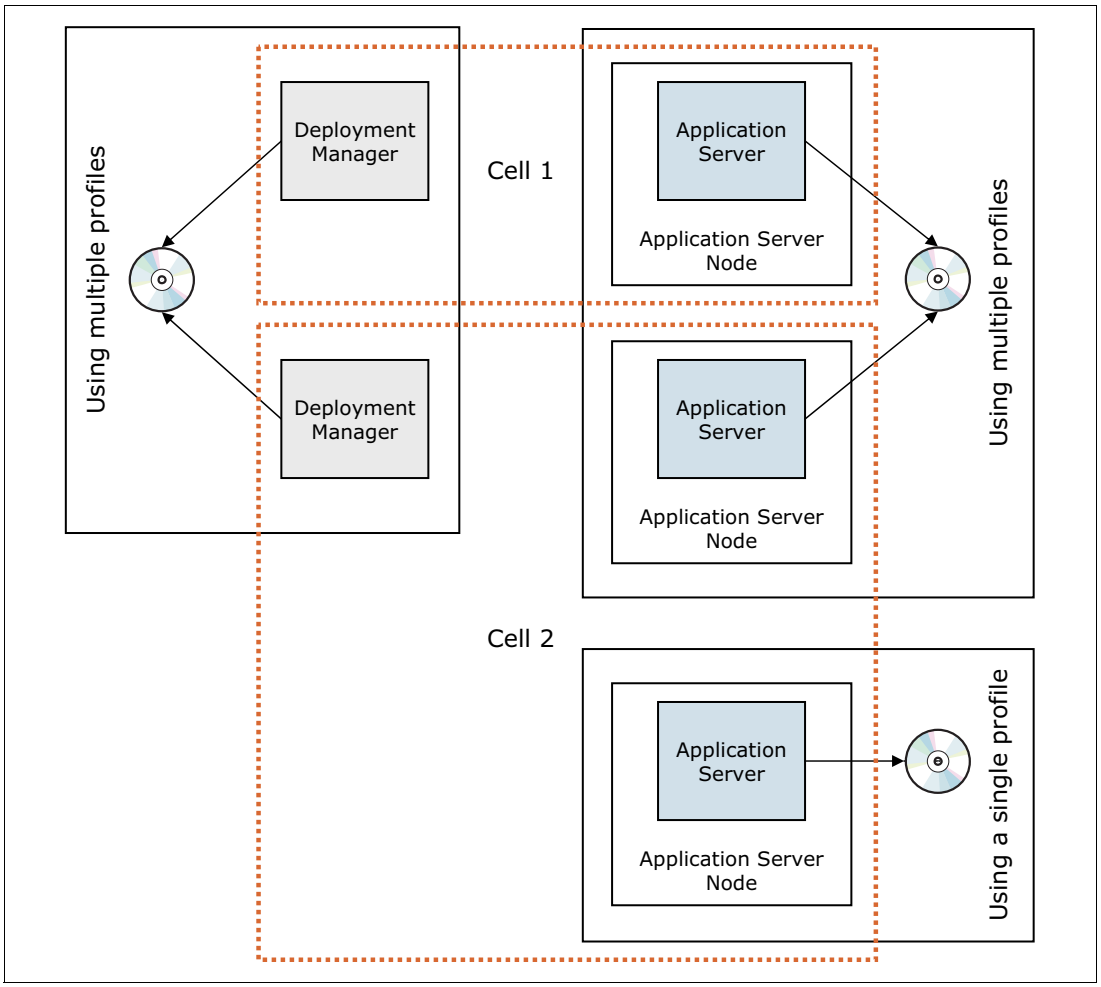


Figure 9-6 Cell configuration flexibility

9.7.3 Installation method

There are a number of ways you can install WebSphere Application Server by using the Installation Manager. Before you start the installation activities, review the options that you have for installing the code, and select the option that best fits your needs. On distributed systems, you have several choices for installation:

- Wizard installation
- Silent installation
- Centralized installation manager

Wizard installation

In wizard mode, you run Installation Manager from a GUI. It is typically used by passing a response file to the Installation Manager launcher. Wizard mode is suitable for installing WebSphere Application Server on a few systems. Running wizard mode installs one system.

You can start with the Launchpad web application that is the starting point for installing WebSphere Application Server. To use the Launchpad application, you must have already installed the Installation Manager.

Installation Manager verifies the required operating system level, sufficient disk space, and user permissions before downloading the required packages.

Silent installation

To install WebSphere Application Server V8.5 on multiple systems or remote systems, use the silent installation. With this option, you can store installation options in a single response file and then enter a command to perform the installation. The silent installation approach offers the same options as the graphical installer. Providing the options in a response file offers the following advantages over using the graphical installation wizard:

- The installation options can be planned and prepared in advance.
- The prepared response file can be tested.
- The installation is consistent and repeatable.
- The installation is less fault-prone.
- The installation is documented through the response file.

Response files can be created by using either the GUI or console mode, and then modified them as needed to suit your environment.

An alternative to using response files to run a silent installation is by using the Installation Manager command-line interface (**`imcl`**). When using the **`imcl`** command, you must identify the number installation attributes in the command line such as the package identifier, repository, and installation directory.

Important: Do not use the same response files that are used with WebSphere Application Server V7 or earlier versions to install or uninstall V8.5 silently. Use response files that are based on Installation Manager to install, update, or uninstall V8 or later.

Centralized installation manager

Another product feature that you can use to install and update WebSphere Application Server Network Deployment installations is the centralized installation manager. For more information, see 12.7.3, “Centralized installation manager” on page 402.

Verifying the installation

You can verify the installation of WebSphere Application Server V8.5 using the installation verification features provided by the Installation Manager.

Verify the installation by using one of the following methods:

- Use the **`listInstalledPackages`** command to display a list of the packages that are installed by Installation Manager. Use the **`-long`** command option to provide more details.
- Start the Installation Manager GUI, and verify the installation by selecting **File** → **View Installed Packages**.

If the installation is successful, you can verify the installation location by viewing the `installation_manager_root/properties/version/installed.xm` file. Look for the location element as shown in Example 9-1.

Example 9-1 The installed.xml file

```
<location id="IBM WebSphere Application Server V8.5" kind="product"
path="C\Program Files\IBM\WebSphere\AppServer">..... </location>
```

If you used the **-log** option of Installation Manager during the installation, verify that the log file does not contain any errors.

The `installed.xml` file can also be opened in a web browser, which provides a nicely formatted view of the installation. The `installed.xml` file is formatted automatically by using the `installed.xsl` file found in the same directory. It is a little easier to examine the contents of this file by using the browser as opposed to a text editor due to the formatting.

9.7.4 Installing updates

The Installation Manager or the centralized installation manager can be used to apply maintenance to WebSphere Application Server V8.5. In either case, the maintenance files are in an Installation Manager repository. If the maintenance is being installed on a single local system, you can start the Installation Manager directly on that system and apply the maintenance. To apply maintenance on multiple or remote systems, use the centralized installation manager from the administrative console or job manager. The centralized installation manager starts the Installation Manager on each system to run the updates. The centralized installation manager requires you to prepare repositories and response files to be used for the installation.

The centralized installation manager can also be used to update the Installation Manager on remote systems. It installs a newer version by using an Installation Manager installation kit or from a repository.

9.7.5 Profile creation

The installation process of WebSphere Application Server provides the product packages that are required to create a runtime environment. However, the actual run time is defined through the usage of profiles. The product binary files remain unchanged after installation until you install maintenance. All profiles of an installation share binary files. Therefore, all server processes of all profiles of an installation use the updated level of the binary files after installing the service. Profiles can be created any time after the installation of the product is finished.

Before you create the profiles, consider the following questions:

- What profile types will you need?

  See “Profile types” on page 253.

- How will you create the profiles?

  See “Creating profiles” on page 255.

- Where do you store the profile configuration files?

  See “Profile location” on page 263.

  You can store profiles under the installation root for Websphere Application Server. Alternatively, you can store them in any location you choose depending on your planning

for disk and directories. For more information, see 9.4, “Planning for disk space and directories” on page 240.

Profile types

The types of profiles that are available to you depend on the WebSphere Application Server package that you installed. The profile types that you need are determined by your topology. The profile types are as follows:

- Management profile with a deployment manager server

  Creating a deployment manager profile creates an application server named `dmgr` and deploys the administrative console. The deployment manager provides a centralized administration interface for multiple nodes with all attached servers in a single cell. The deployment manager profile is the basis for configuring a topology with clustering, high availability, failover, and so on.

  You can use the **`manageprofiles`** command with one of the following options to create a deployment manager profile:

  – Specify `-profileTemplate` *`app_server_root`*`/profileTemplates/management`, and then specify `-serverType DEPLOYMENT_MANAGER`.

  – Specify `-profileTemplate` *`app_server_root`*`/profileTemplates/dmgr`.

- Management profile with an administrative agent server

  Creating an administrative agent profile creates the application server named `adminagent` and deploys the administrative console.

  The administrative agent provides a centralized administration interface for multiple unfederated application server profiles on the same system.

  When using **`manageprofiles`**, specify `-profileTemplate` *`app_server_root`*`/profileTemplates/management`, and then specify `-serverType ADMIN_AGENT`.

- Management profile with a job manager server

  Creating a job manager profile creates the application server named `jobmgr` and deploys the administrative console.

  The job manager provides a centralized interface for the following tasks:

  – Administering multiple unfederated application server profiles through the administrative agent

  – Deployment manager profiles

  – Asynchronous job submissions

  When using **`manageprofiles`**, specify `-profileTemplate` *`app_server_root`*`/profileTemplates/management`, and then specify`-serverType JOB_MANAGER`.

- Application server profiles

  An application server profile creates an application server and deploys these applications:

  – Default applications (optional)
  – Sample applications (optional)
  – Administrative console

  The default name of the application server is server1. This name can be overridden through the `-serverName` parameter in the **`manageprofiles`** command or when using the advanced profile creation option in the Profile Management Tool. The application server can run as a stand-alone application server, or it can be federated to cell defined by a deployment manager profile.

When using **manageprofiles**, specify `-profileTemplate` *app_server_root*`/profileTemplates/default` to create an application server profile.

- Custom profiles

  The custom profile creates an empty node in a cell that does not contain an administrative console or servers. No applications are deployed. The typical use for a custom profile is to federate its node to a cell either during profile creation or at a later time.

  When using **manageprofiles** specify `-profileTemplate` *app_server_root*`/profileTemplates/managed`.

- Cell profiles

  The cell profile option creates a deployment manager profile and a federated application server profile on a single system. The profile uses default naming conventions and the administrative console is deployed. The result of this profile creation is a fully functional cell. The following applications can be deployed to the federated application server during the profile creation:

  – Default applications (optional)
  – Sample applications (optional)

  From the functional perspective, the cell profile approach is the same as performing these steps:

  – Creating a management profile with a deployment manager server and an application server profile
  – Federating the application server profile to the deployment manager

  When using **manageprofiles**, two profiles must be created. To create the deployment manager portion of the profile, specify `-profileTemplate` *app_server_root*`/profileTemplates/cell/dmgr`. For the cell node portion of the profile, specify `-profileTemplate` *app_server_root*`/profileTemplates/cell/default`.

- Secure proxy profile

  A secure proxy profile creates a proxy server that is supposed to run in the DMZ. This server supports HTTP, Session Initiation Protocol (SIP), and the corresponding secure version of the protocols.

  The default name of the application server is `proxy1`. However, this name can be overridden with the **-serverName** parameter in the **manageprofiles** command or when using the advanced profile creation option in the Profile Management Tool.

  When using **manageprofiles**, to create a secure proxy profile, specify `-profileTemplate` *app_server_root*`/profileTemplates/secureproxy`.

Table 9-1 shows a list of the available profile types per WebSphere Application Server edition.

Table 9-1 Available profile types for editions of WebSphere Application Server

Product	WebSphere profiles available
WebSphere Application Server Express V8.5	► Management profile with an administrative agent server ► Application server profile
WebSphere Application Server V8.5 (Base)	► Management profile with an administrative agent server ► Application server profile

Product	WebSphere profiles available
WebSphere Application Server Network Deployment V8.5	► Management profile with a deployment manager server ► Management profile with an administrative agent server ► Management profile with a job manager server ► Application server profile ► Cell profile ► Custom profile ► Secure proxy profile

Creating profiles

On distributed platforms, profiles are created after installing the product by using either the Profile Management Tool or the **manageprofiles** command.

Express and Base installation

The installation procedure for WebSphere Application Server V8.5 Express and Base installs the core product files. After the installation, you can create application server profiles by using the Profile Management Tool. Additional profiles that you create can be anywhere on the file system.

Network Deployment installations

The installation procedure for WebSphere Application Server Network Deployment V8.5 installs the core product files. After the installation, you can create profiles by using the Profile Management Tool or the **manageprofiles** command.

WebSphere for z/OS installations

IBM Installation Manager is used on z/OS to install, update, and provide maintenance to WebSphere Application Server environment. SMP/E is only used to receive and accept the base function modification identifiers (FMIDs) or service level fix pack APARs. These elements create the IBM Installation Manager repository in the DDDEF mounted file system.

After the installation, you create profiles by using the z/OS Profile Management Tool, which is available in the WebSphere Customization Toolkit. For more information about z/OS installation and the configuration steps, see 16.3, “Installing WebSphere Application Server for z/OS” on page 526.

Using the Profile Management Tool and the advanced path, or the **manageprofiles** command, to create the profiles provides more flexibility in the options you can select.

Tip:

- Use **manageprofiles.bat(sh)** to create your production profiles. The scripting approach allows reuse and easier documentation.
- To determine the parameters that **manageprofiles.bat(sh)** requires for a specific profile type, run the following command:

  ```
  manageprofiles.bat(sh) -create -templatePath templatePath -help
  ```

  For example, on a Windows system, run the following command:

  ```
  .manageprofiles.bat -create -templatePath
  \WebSphere\Appserver\profileTemplates\management -help
  ```

Options for the deployment manager profile

Table 9-2 summarizes the options that are available when creating a profile for a deployment manager. The options depend on whether you choose the typical path or advanced path through the Profile Management Tool.

Table 9-2 Options for the deployment manager profile

Typical settings	Advanced options
The administrative console is deployed by default.	You can deploy the administrative console (recommended and preselected).
The profile name is `Dmgr`*`xx`* by default, where *xx* is 01 for the first deployment manager profile and increments for each profile that is created. The profile is stored in the *app_server_root*`/profiles/Dmgrxx` directory.	You can specify the profile name and its location.
The profile is not marked as the default profile.	You can choose whether to make this profile the default profile. Commands that are run without specifying a profile are run against the default profile.
The cell name is *host*`Cellxx`. The node name is *host*`CellManagerxx`. The host name defaults to the DNS host name of the system.	You can specify the node, host, and cell names.
You can select whether to enable administrative security. By default, the **Enable administrative security** option is preselected. If you select **yes**, you must specify a user name and password that have administrative authority.	
Creates a default personal certificate for this profile by using the domain name (DN): `cn=`*`hostname`*`,ou=`*`cellname`*`,ou=`*`nodename`*`,o=IBM,c=US`	You can enter the DN for the new certificate that is being created, or import an existing default personal certificate from a keystore.
Creates a new root signer certificate for this profile by using the DN: `cn=`*`hostname`*`,ou=Root Certificate,ou=`*`cellname`*`,ou=`*`nodename`*`,o=IBM,c=US`	You can enter the DN for the new root signer certificate that is being created, or import an existing root signing certificate from a keystore.
The default expiration date for the personal certificate is one year.	You can enter the expiration period.
The default expiration date for the signer certificate is 15 years.	You can enter the expiration period.
The keystore password is `WebAS`.	You can enter a unique password for the keystore.
TCP/IP ports default to a set of ports that is 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.
For Windows, the deployment manager is run as a service by using a local system account and startup type of *Automatic*. For Linux, the deployment manager does not run as a Linux service.	For Windows, select whether the deployment manager runs as a service, under which account the service runs, and the startup type that is used. For Linux, you can create a Linux service and specify the user name from which the service runs.

Options for the administrative agent profile

Table 9-3 summarizes the options that are available when creating a profile for an administrative agent. The options depend on whether you choose the typical path or advanced path through the Profile Management Tool.

Table 9-3 Options for the administrative agent profile

Typical settings	Advanced options
The administrative console is deployed by default.	You can deploy the administrative console (recommended and preselected).
The profile name is `AdminAgent`*`xx`* by default, where *xx* is 01 for the first administrative agent profile and increments for each profile that is created. The profile is stored in the *`app_server_root`*`/profiles/AdminAgentxx` directory.	You can specify the profile name and its location.
The profile is not marked as the default profile.	You can choose whether to make this profile the default profile. Commands that are run without specifying a profile are run against the default profile.
The cell name is *`host`*`AACellxx`. The node name is *`host`*`AANodexx`. The host name defaults to the DNS host name of the system.	You can specify the node, host, and cell names.
You can select whether to enable administrative security. By default **Enable administrative security** is preselected. If you select **yes**, you must specify a user name and password that have administrative authority.	
Creates a default personal certificate for this profile by using the DN: `cn=`*`hostname`*`,ou=`*`cellname`*`,ou=`*`nodename`*`,o=IBM,c=US`	You can enter the DN for the new certificate that is being created, or import an existing default personal certificate from a keystore.
Creates a new root signer certificate for this profile by using the DN: `cn=`*`hostname`*`,ou=Root Certificate,ou=`*`cellname`*`,ou=`*`nodename`*`,o=IBM,c=US`	You can enter the DN for the new root signer certificate that is being created, or import an existing root signing certificate from a keystore.
Default expiration date for the personal certificate is one year.	You can enter the expiration period.
Default expiration date for the signer certificate is 15 years.	You can enter the expiration period.
The keystore password is `WebAS`.	You can enter a unique password for the keystore.
TCP/IP ports default to a set of ports that is 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.
For Windows, the deployment manager is run as a service by using a local system account and startup type of *Automatic*. For Linux, the deployment manager does not run as a Linux service.	For Windows, select whether the deployment manager runs as a service, under which account the service runs, and the startup type that is used. For Linux, you can create a Linux service and specify the user name from which the service runs.

Options for the job manager profile

Table 9-4 summarizes the options that are available when creating a profile for a job manager. The options depend on whether you choose the typical path or advanced path through the Profile Management Tool.

Table 9-4 Options for the job manager profile

Typical settings	Advanced options
The administrative console is deployed by default.	You can deploy the administrative console (recommended and preselected).
The profile name is `JobMgr`*`xx`* by default, where *xx* is 01 for the first administrative agent profile and increments for each profile that is created. The profile is stored in the *`app_server_root`*`/profiles/JobMgrxx` directory.	You can specify the profile name and its location.
The profile is not marked as the default profile.	You can choose whether to make this profile the default profile. Commands that are run without specifying a profile are run against the default profile.
The cell name is *`host`*`JobMgrCellxx`. The node name is *`host`*`JobMgrxx`. The host name defaults to the DNS host name of the system.	You can specify the node, host, and cell names.
You can select whether to enable administrative security. By default, **Enable administrative security** is preselected. If you select **yes**, you must specify a user name and password that have administrative authority.	
Creates a default personal certificate for this profile by using the DN: `cn=`*`hostname`*`,ou=`*`cellname`*`,ou=`*`nodename`*`,o=IBM,c=US`	You can enter the DN for the new certificate that is being created, or import an existing default personal certificate from a keystore.
Creates a root signer certificate for this profile by using the DN: `cn=`*`hostname`*`,ou=Root Certificate,ou=`*`cellname`*`,ou=`*`nodename`*`,o=IBM,c=US`	You can enter the DN for new root signer certificate that is being created, or import an existing root signing certificate from a keystore.
The default expiration date for the personal certificate is one year.	You can enter the expiration period.
The default expiration date for the signer certificate is 15 years.	You can enter the expiration period.
The keystore password is `WebAS`.	You can enter a unique password for the keystore.
TCP/IP ports default to a set of ports that is 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.
For Windows, the deployment manager is run as a service by using a local system account and startup type of *Automatic*. For Linux, the deployment manager does not run as a Linux service.	For Windows, select whether the deployment manager runs as a service, under which account the service runs, and the startup type that is used. For Linux, you can create a Linux service and specify the user name from which the service runs.

Options for the application server profile (non-Express V8.5)

Table 9-5 summarizes the options that are available when creating a profile for an application server. The options depend on whether you choose the typical path or advanced path through the Profile Management Tool.

Table 9-5 Options for the application server profile (non-Express V8.5)

Typical settings	Advanced options
The administrative console and default application are deployed by default. The sample applications are not deployed.	You can deploy the administrative console (recommended and preselected), the default application (preselected), 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 profile that is created. The profile is stored in the *app_server_root*`/profiles/AppSrvxx` directory.	You can specify the profile name and its location.
The profile is not marked as the default profile.	You can choose whether to make this profile the default profile. Commands that are run without specifying a profile are run against the default profile.
The application server is built by using the default application server template.	You can choose the default template or a development template that is optimized for development purposes.
The node name is *host*`Nodexx`. The server name is `server1`. The host name defaults to the DNS host name of the system.	You can specify the node name, server name, and host name.
You can select whether to enable administrative security. By default **Enable administrative security** is preselected. If you select **yes**, you must specify a user name and password that have administrative authority.	
Creates a default personal certificate for this profile by using the DN: `cn=`*hostname*`,ou=`*cellname*`,ou=`*nodename*`,o=IBM,c=US`	You can enter the DN for the new certificate that is being created, or import an existing default personal certificate from a keystore.
Creates a root signer certificate for this profile using the DN: `cn=`*hostname*`,ou=Root Certificate,ou=`*cellname*`,ou=`*nodename*`,o=IBM,c=US`	You can enter the DN for the new root signer certificate that is being created, or import an existing root signing certificate from a keystore.
The default expiration date for the personal certificate is one year.	You can enter the expiration period.
The default expiration date for the signer certificate is 15 years.	You can enter the expiration period.
The keystore password is `WebAS`.	You can enter a unique password for the keystore.
The TCP/IP ports default to a set of ports that is 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.
For Windows, the application server is run as a service by using a local system account and startup type of *Automatic*. For Linux, the application server does not run as a Linux service.	For Windows, select whether the application server runs as a service, under which account the service runs, and the startup type that is used. For Linux, you can create a Linux service and specify the user name from which the service runs.
Does not create a web server definition.	You can define an external web server to the configuration.

Options for the application server profile (Express V8.5)

Table 9-6 summarizes the options that are available when creating a profile for an application server in WebSphere Application Server Express V8.5. The options depend on whether you choose the typical path or advanced path through the Profile Management Tool.

Table 9-6 Options for the application server profile (Express V8.5)

Typical settings	Advanced options
The administrative console and default application are deployed by default. The sample applications are not deployed.	You can deploy the administrative console (recommended and preselected), the default application (preselected), 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 profile that is created. The profile is stored in the *app_server_root*`/profiles/AppSrvxx` directory.	You can specify the profile name and its location.
The application server is built by using the default application server template.	You can choose the default template or a development template that is optimized for development purposes.
The node name is *host*`Nodexx`. The server name is `server1`. The host name defaults to the DNS host name of the system.	You can specify the node name, server name, and host name.
You can select whether to enable administrative security. By default **Enable administrative security** is preselected. If you select **yes**, you must specify a user name and password that are given administrative authority.	
Creates a default personal certificate for this profile by using the DN: `cn=`*`hostname`*`,ou=`*`cellname`*`,ou=`*`nodename`*`,o=IBM,c=US`	You can enter the DN for the new certificate that is being created or import an existing default personal certificate from a keystore.
Creates a root signer certificate for this profile using the DN: `cn=`*`hostname`*`,ou=Root Certificate,ou=`*`cellname`*`,ou=`*`nodename`*`,o=IBM,c=US`	You can enter the DN for new root signer certificate that is being created, or import an existing root signing certificate from a keystore.
The default expiration date for the personal certificate is one year.	You can enter the expiration period.
The default expiration date for the signer certificate is 15 years.	You can enter the expiration period.
The keystore password is `WebAS`.	You can enter a unique password for the keystore.
The TCP/IP ports default to a set of ports that is 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.
For Windows, the application server is run as a service by using a local system account and startup type of *Automatic*. For Linux, the application server does not run as a Linux service.	For Windows, select whether the application server runs as a service, under which account the service runs, and the startup type that is used. For Linux, you can create a Linux service and specify the user name from which the service runs.
Does not create a web server definition.	You can define an external web server to the configuration.

Cell profile options

Table 9-7 summarizes the options that are available when creating a cell profile. Using these options creates two distinct profiles: A deployment manager profile and an application server profile. The application server profile is federated to the cell. The options that you see are a reflection of the options that you might see when creating the individual profiles versus a cell profile.

Table 9-7 Cell profile options

Typical settings	**Advanced options**
The administrative console and default application are deployed by default. The sample applications are not deployed.	You can deploy the administrative console (recommended and preselected), the default application (preselected), and the sample applications (if installed).
The profile name for the deployment manager is `Dmgr`*`xx`* by default. *xx* is 01 for the first deployment manager profile and increments for each profile that is created.	You can specify the profile name.
The profile name for the federated application server and node is `AppSrv`*`xx`* by default. *xx* is 01 for the first application server profile and increments for each profile that is created.	You can specify the profile name.
The *app_server_root*`/profiles` directory is used as *profile_root*. The profiles are created in the *profile_root*`/`*profilename* directory.	You can specify the *profile_root* directory. The profiles are created in the *profile_root*`/`*profilename* directory.
Neither profile is made the default profile.	You can 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 defaults to the DNS host name of the system.	You can specify the cell name, the host name, and the profile names for both profiles.
You can select whether to enable administrative security. By default **Enable administrative security** is preselected. If you select **yes**, you must specify a user name and password that have administrative authority.	
Creates a default personal certificate for this profile by using the DN: `cn=`*hostname*`,ou=`*cellname*`,ou=`*nodename*`,o=IBM,c=US`	You can enter the DN for the new certificate that is being created or import an existing default personal certificate from a keystore.
Creates a root signer certificate for this profile using the DN: `cn=`*hostname*`,ou=Root Certificate,ou=`*cellname*`,ou=`*nodename*`,o=IBM,c=US`	You can enter the DN for new root signer certificate that is being created, or import an existing root signing certificate from a keystore.
The default expiration date for the personal certificate is one year.	You can enter the expiration period.
The default expiration date for the signer certificate is 15 years.	You can enter the expiration period.
The keystore password is `WebAS`.	You can enter a unique password for the keystore.
The TCP/IP ports default to a set of ports that is 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.

Typical settings	Advanced options
For Windows, the application server is run as a service by using a local system account and startup type of *Automatic*. For Linux, the product is not selected to run as a Linux service.	For Windows, select whether the application server runs as a service, under which account the service runs, and the startup type that is used. For Linux, you can create a Linux service and specify the user name from which the service runs.
Does not create a web server definition.	You can define an external web server to the configuration.

Custom profile options

Table 9-8 summarizes the options that are available when creating a custom profile.

Table 9-8 Custom profile options

Typical settings	Advanced options
The profile name is `Customxx`. The profile is stored in the *app_server_root*`/profiles/Customxx` directory. By default, it is not considered the default profile.	You can specify the profile name and location. You can also specify whether you want this profile to be the default profile.
The profile is not selected to be the default profile.	You can select this profile to be the default profile.
The node name is *host*`Nodexx`. The host name defaults to the DNS host name of the system.	You can specify the node name and host name.
You can choose to federate the node later, or during the profile creation process. If you want to federate the node now, specify the deployment manager host and SOAP port (by default, `localhost:8879`). If security is enabled on the deployment manager, you must specify a user ID and password.	
Creates a default personal certificate for this profile by using the DN: `cn=`*hostname*`,ou=`*cellname*`,ou=`*nodename*`,o=IBM,c=US`	You can enter the DN for the new certificate that is being created or import an existing default personal certificate from a keystore.
Creates a root signer certificate for this profile by using the DN: `cn=`*hostname*`,ou=Root Certificate,ou=`*cellname*`,ou=`*nodename*`,o=IBM,c=US`	You can enter the DN for new root signer certificate that is being created, or import an existing root signing certificate from a keystore.
The default expiration date for the personal certificate is one year.	You can enter the expiration period.
The default expiration date for the signer certificate is 15 years.	You can enter the expiration period.
The keystore password is `WebAS`.	You can enter a unique password for the keystore.
The TCP/IP ports default to a set of ports that is 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.

Starting the Profile Management Tool

After you install Websphere Application Server V8.5 on distributed systems, you can start the Profile Management Tool in the following ways:

- From the First Steps window.
- On Windows systems, from the Start menu (**Start** → **Programs** → **IBM WebSphere** → **IBM WebSphere Application Server [Network Deployment V8.5.0] Tools** → **Profile Management Tool**).

- By running the **wct.bat(sh)** command:
  - For operating systems such as AIX or Linux, the command is in the *app_server_root*/bin/ProfileManagement directory.
  - For the Windows platform, the command is in the *app_server_root*\bin\ProfileManagement directory.

The Profile Management Tool provides a GUI for the ***app_server_root*/manageprofiles.bat(sh)** command. However, you can also use this command directly to manage profiles without the graphical interface.

Profile location

Profiles that are created by using the typical settings are automatically placed in the *app_server_root*/profiles directory. You can designate the location where the profiles are stored. For more information about considerations about disk space and directory planning, see 9.4, “Planning for disk space and directories” on page 240.

9.7.6 Naming convention

The purpose for developing systematic naming concepts and rules for a WebSphere site is two-fold:

- Provide guidance during setup and configuration
- Quickly narrow down the source of any issue that arises

Naming the WebSphere Application Server infrastructure artifacts, such as cells, nodes, and application servers, must follow the normal naming conventions of the company as closely as possible.

Keep in mind the following considerations, among others, when developing the key concepts for the site during the installation planning:

- Naming profiles

  The profile name can be any unique name, but have a standard for naming profiles. Having a standard helps administrators easily determine a logical name for a profile when creating it. It also helps them to find the correct profiles easily after creation. For example, a profile can include characters that indicate the profile type, server, and an incremental number to distinguish it from other, similar profiles.

  Do not use any of the following characters when naming your profile:

  - Spaces
  - Special characters that are not allowed within the name of a directory on your operating system (namely * & ? ‘ “ and ,)
  - Slashes (/ or \)

- Naming cells

  A *cell* represents an administrative domain.

  In a stand-alone environment, the cell name is not visible to administrators, and a naming convention is not required. The name is automatically generated during profile creation and is in the following format:

  <system_name><node_name><number>Cell

  The <number> increments, starting with 01, with every new node, for example, server1Node01Cell and server1Node02Cell.

In a distributed server environment, there are considerations for naming a cell. A cell name must be unique in the following situations:

- When the product is running on the same physical machine or cluster of machines, such as a sysplex
- When network connectivity between entities is required either between the cells or from a client that must communicate with each of the cells
- When the name spaces of the cell are going to be federated

Often a naming convention for cell names includes the name of the stage (such as integration test, acceptance test, or production). If appropriate, it also includes the name of the department or project that owns it.

- Naming nodes

  In a stand-alone environment, you have a single node with a single application server. A naming convention is not really a concern. However, you can specify a node name during profile creation. If you use the default, the node name is in the format *system_name*`NODE`*number*. The *number* value increments, starting with 01, with every new node, for example, `server1Node01` and `server1Node02`.

  In a distributed server environment, the node must be unique within a cell. Nodes generally represent a system and often include the host name of the system. You can have multiple nodes on a system, which is important to keep in mind when planning WebSphere names.

  Naming conventions for nodes often include the physical machine name where they are running, such as `NodexxAP010` if the server name is `ServerAP010`. They often add an incremented number to enable growth if additional nodes need to be created.

- Naming application servers

  In stand-alone environments, the default server name is `server1`. However, this name can be overridden through the **`manageprofiles.bat(sh)`** command or by using the advanced profile creation options in the Profile Management Tool.

  Consideration: When you federate multiple stand-alone application servers created by using the default naming schema to a cell, you have a unique combination of a node name and `server1`. The result is that you end up with multiple occurrences of `server1` in the cell.

  In a distributed server environment, new application servers are usually created on a federated node by using the administrative console or another administrative tool. In this case, you can assign the server a meaningful name. Name servers based on their location, function, membership in a cluster, or some other scheme depending on how your servers will be used and administered.

  Important: The server name must be unique within the node.

  If each application server hosts only a single application, the application server name can include the name of the application. If several applications make up a total system or project, that name can be used as a prefix to group the application servers. This notation makes it easier to find them in the administrative console.

  If an application server hosts multiple applications, develop another suitable naming convention. For example, use the name of a project or the group of applications deployed on the server.

- General naming rules

  Avoid using reserved folder names as field values. The use of reserved folder names can cause unpredictable results. The following words are reserved in WebSphere Application Server:

  – Cells
  – Nodes
  – Servers
  – Clusters
  – Applications
  – Deployments

  When you create an object by using the administrative console or a **wsadmin** command, you often must specify a string for a name attribute. Most characters are allowed in the name string (numbers, letters). However, the name string cannot contain special characters or signs. The dot is not valid as a first character. The name string also cannot contain leading and trailing spaces.

  Tip: Avoid any language-specific characters in names.

9.7.7 TCP/IP port assignments

Develop the port assignment scheme for a WebSphere site in close cooperation with the network administrators. From a security point-of-view, know the usage of each port ahead of time, including the process names and the owners who are using them.

Depending on the chosen WebSphere Application Server configuration and hardware topology, the setup for a single system can involve having multiple cells, nodes, and server profiles in a single process space. Each WebSphere process requires exclusive usage of several ports and knowledge of certain other ports for communication with other WebSphere processes.

To simplify the installation and provide transparency to the ports use, the following approach is reliable and considerably reduces the complexity of such a scenario:

- With the network administration, decide on a fixed range of continuous ports for exclusive use for the WebSphere Application Server installation.
- Draw the overall topology of WebSphere Application Server, and map your application lifecycle stages to WebSphere profiles.
- List the number of required ports per WebSphere profile, and come up with an enumeration scheme. Use appropriate increments at the cell, node, and application server level, starting with your first cell. Make sure to document the ports in an appropriate way.

  Tip: You can use the same spreadsheet for the server names, process names, user IDs, and so on.

When creating your profiles with the Profile Management Tool by using the advanced options path, you can set the ports for your profile as needed. The Profile Management Tool identifies the ports used in the same installation on that system and those currently in use, and suggests unique ports to use.

With the **manageprofiles.bat(sh)** command, you can control the port numbers through the **-portsFile** and **-startingPort** parameters.

To ensure that you do not have port conflicts between WebSphere Application Server profiles and products, use the port validator tool to verify your configuration. The port validator tool is one of the tools that is available with the **servicetools** script.

For a list of the ports used by WebSphere Application Server and their default settings, see the Websphere Application Server V8.5 Information Center at the following address:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=urun_rendpoint_inst

9.7.8 Security considerations

To plan a secure WebSphere Application Server environment, you must have highly skilled security specialists who can evaluate your business and network security needs. You need to have a clear idea of your plans for security before you install any production systems.

Installers must take into account the following security considerations during the installation planning phase:

- Certificates
  - If you use digital certificates, make sure that you request them with enough lead time so that they are available when you need them.
  - If default certificates or dummy key ring files are provided with any of the products you plan to install, replace them with your own certificates.
  - If you are using self-signed certificates, plan your signer structure carefully, and exchange signer certificates if necessary.

    Remember: In WebSphere Application Server V8.5, signer and personal certificates can be created or imported during profile creation. If you have new certificates created, you can choose the correct DN during profile creation.

- Network and physical security
  - Usually one or more firewalls are part of the topology. After determining the ports that need to be open, make a request to the firewall administrator to open them.
  - Plan the physical access to the data center where the machines are going to be installed. This planning helps prevent delays to the personnel involved in the installation and configuration tasks.
- User IDs
  - Request user IDs with enough authority for the installation purposes. For example, you need a root ID on a Linux or UNIX operating system and a member of the administrator group on a Windows operating system. For more information, see the Websphere Application Server V8.5 Information Center at:

    http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=tins_installation_dist_cl

    You can run Installation Manager in group mode. For more information about group mode, see “Installation modes” on page 267.
  - Ensure that any policies on password expiration are well known to avoid disruption on the service. These policies include password expiration of root, administrator, and the users who access a database.

Installation modes

You can install WebSphere Application Server V8.5 with Installation Manager by using one of the following modes:

- Administrator

  The Installation Manager is installed from an administrator or root ID, and is used to install software by any administrator or root user.

- Nonadministrator

  The Installation Manager is used to install software only by the user who installed Installation Manager, which is also known as *user mode*.

- Group

  The Installation Manager can be used to install software by any user who is connected to the default group of the user who installed Installation Manager. Group mode is not available on Windows and IBM i platforms.

> **More information:** For detailed information about installing in group mode, see Websphere Application Server V8.5 Information at:
>
> http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=tins_installation_dist_group

Secure administration tasks

WebSphere Application Server provides a mechanism to secure the administrative interfaces. With WebSphere Application Server V8.5, you can enable security for administrative tasks during profile creation for an application server or deployment manager. This includes those tasks that were created with cell profiles. This option does not enable application security.

The user ID and password specified during profile creation are created in the repository and assigned the Administrator role. This ID can be used to access the administration tools and to add additional user IDs for administration. When you enable security during profile creation, Lightweight Third Party Authentication (LTPA) is used as the authentication mechanism. The federated repository realm used is used as account repository.

On distributed systems, an XML file-based user repository is created and populated with the administrator ID. This XML file-based system can be federated with other repository types to form an overall repository system. If you do not want to use the file-based repository, do not enable administrative security during profile creation or change it afterward. In WebSphere for z/OS, you can use the file-based repository or the z/OS system System Authorization Facility (SAF)-compliant security database. Whether you choose to enable administration security during profile creation or after, you must do it before going into production.

9.7.9 IBM Support Assistant

The IBM Support Assistant is a tool provided by IBM at no charge to troubleshoot a WebSphere Application Server environment. IBM Support Assistant consists of the following components:

- IBM Support Assistant Workbench
- IBM Support Assistant Agent Manager
- IBM Support Assistant Agent

For installation instructions and more details, see IBM Support Assistant at:

http://www.ibm.com/software/support/isa/

9.8 Planning for the Liberty profile

The WebSphere Application Server V8.5 Liberty profile is a profile of the application server that is optimized for developer productivity and smaller, simpler production server deployments.

> **More information:** To learn more about the Liberty profile, see Chapter 4, “An overview of the Liberty profile” on page 91.

Before you install the Liberty profile, you must determine the installation method that you want to use. In WebSphere Application Server V8.5, you can install the Liberty profile, the Full profile, or both by using the Installation Manager.

You can install the Liberty profile application-serving environment by using one of the following methods:

- Install the Liberty profile by using the Installation Manager

  The Liberty profile is an optional feature that can be selected during installation.

- Install the Liberty profile developer tools and application-serving environment

  The Liberty profile developer tools are an optional installation feature in WebSphere Application Server V8.5 developer tools. Installing the Liberty profile developer tools requires the following components:

  – An Eclipse IDE for Java EE Developers
  – A Java runtime environment (JRE)

  To learn more about installing the Liberty profile developer tools and application-serving environment, see Chapter 11, “Application development and deployment” on page 341.

- Install the Liberty profile by extracting an archive file

  You can download the Liberty profile run time outside of the Liberty profile tools. The Liberty profile is packaged as an archive file. You can then install the Liberty profile by extracting the archive file. All of the files that are needed for the Liberty profile run time are placed in a `wlp` directory.

  You can download the Liberty profile archive file from the following URL:

  `http://www.ibm.com/developerworks/mydeveloperworks/blogs/wasdev/entry/download`

If you have a WebSphere Application Server Base, Express, or Network Deployment package, you can install the Liberty profile by using the Installation Manager. The Installation Manager gives you the option of installing the full profile, the Liberty profile, or both.

Figure 9-7 shows the Installation Manager Install Packages window where you select Full profile, Liberty profile, or both.

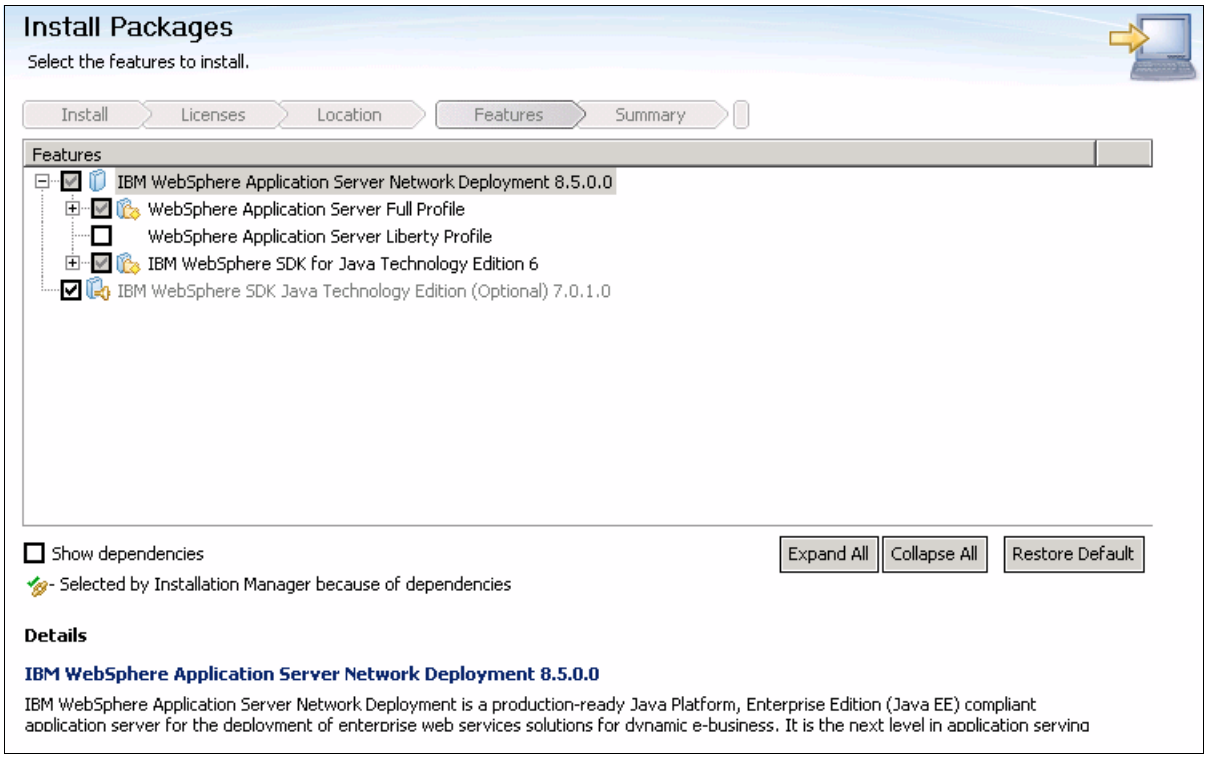


Figure 9-7 Installation Manager Install Packages window

If the Liberty profile is not installed when you install WebSphere Application Server, you can install it later using the Installation Manager. However, the Liberty profile must be installed as another package group and into a separate installation directory. You cannot modify an existing WebSphere Application Server package group to add the Liberty profile.

When you install either the Liberty profile or the Full profile by using the Installation Manager, IBM WebSphere Java SDK 6.2.6 is installed. You can then install the IBM WebSphere Java Technology Edition SDK V7.0 using the Installation Manager. The Full profile requires an IBM SDK. The Liberty profile runs on any supported JRE.

For more information about the minimum supported Java levels for the Liberty profile, see:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=rwlp_restrict

9.9 WebSphere Customization Toolbox

The WebSphere Customization Toolbox for WebSphere Application Server V8.5 includes tools to help you manage, customize, and migrate various parts of the WebSphere Application Server environment. The WebSphere Customization Toolbox is available as the following offerings, each with various combinations of tools on different platforms:

- The *embedded* offering is installed when WebSphere Application Server V8.5 is installed. It includes the Profile Management Tool and the Configuration Migration Tool.
- The *stand-alone* offering comes as its own product offering and is installed by using the Installation Manager. It includes the Web Server Plug-ins Configuration Tool, z/OS Profile

Management Tool, z/OS Migration Management Tool, and Remote Installation Tool for IBM i.

The WebSphere Customization Toolbox offerings include the following tools:

- The Profile Management Tool provides a user interface for profile creation and augmentation.
- The Configuration Migration Tool provides a graphical interface to the migration tools that are included in WebSphere Application Server.
- The Web Server Plug-ins Configuration Tool is a new tool that you can use to configure your web server plug-ins on distributed and Windows operating systems. It is used to communicate with the application server. If possible, this tool also creates a web server configuration definition in the application server.
- The z/OS Profile Management Tool can be used on an Intel-based or Linux operating system to generate jobs and instructions for creating profiles for WebSphere Application Server on z/OS systems. The jobs are then uploaded and run on a target z/OS system.
- The z/OS Migration Management Tool can be used on an Intel-based or Linux operating system to generate definitions for migrating WebSphere Application Server for z/OS nodes. Each migration definition is a set of jobs and instructions that can be uploaded and run on a target z/OS system.
- Remote Installation Tool for IBM i installs Installation Manager or a WebSphere Application Server component from a Windows system to a remote target IBM i system. It can be used only on an Intel-based operating system.

Restriction: These tools are not required or supported for use with the Liberty profile.

The embedded WebSphere Customization Toolbox comes as part of the WebSphere Application Server V8.5 package. It is installed on all platforms where WebSphere Application Server V8.5 is installed and supported. When installing the embedded offering, both the Profile Management Tool and Configuration Migration Tool are installed automatically. The tools are not listed for selection in the Installation Manager.

The stand-alone WebSphere Customization Toolbox comes as its own product offering. It can be found in the WebSphere Application Server V8.5 supplements package, and is installed by using the Installation Manager. When installing the stand-alone offering, select the tools that you want to install. However, the z/OS Profile Management Tool and z/OS Migration Management Tool have co-dependencies that are recognized by the Installation Manager, and therefore must be installed together.

9.10 Planning for Edge Components

Edge Components are a part of the WebSphere Application Server offering. You can use Edge Components in conjunction with WebSphere Application Server to control client access to web servers. With Edge Components, you can provide better service to users who access web-based content over the Internet or a corporate intranet. Using Edge Components can reduce web server congestion, increase content availability, and improve web server performance.

WebSphere Application Server V8.5 includes the following Edge Components:

- Load Balancer
- Caching Proxy

Edge Components usually run on systems that are close to the boundary between an enterprise's intranet and the Internet.

Load Balancer

The Edge Components Load Balancer creates edge-of-network systems that direct network traffic flow, reduce congestion, and balance the load on various other services and systems. Load Balancer provides site selection, workload management, session affinity, and failover.

Load Balancer consists of the following components that can be used separately or together:

- *Dispatcher* distributes the load it receives to servers contained in a cluster of servers that run the same applications. This mechanism is also known as *IP spraying*.
- *Content Based Routing* load balances based on the content of the request. With the Caching Proxy components, the Content Based Routing component can proxy HTTP and HTTPS requests to specific servers based on content requested.
- *Site Selector* runs load balancing by using a DNS round-robin approach or a more advanced user-specified approach.
- *Cisco CSS Controller* and *Nortel Alteon Controller* are controllers that can be used to generate server weighting metrics. The metrics are then sent to the Cisco and Alteon Switch for optimal server selection, load optimization, and fault tolerance.
- *Metric Server* is a component that is installed and runs in each back-end server. Metric Server can additionally provide values for the server where it is running

Caching Proxy

The Caching Proxy intercepts requests from the client, retrieves the requested information from the content-hosting systems, and delivers that information back to the client. You can configure Caching Proxy to handle protocols such as HTTP, FTP, and Gopher.

The Caching Proxy stores content that can be cached in a local cache before delivering it to the requestor. Examples of content that can be cached include static web pages and whole dynamic web pages. The Caching Proxy can then satisfy subsequent requests for the same content by delivering it directly from the local cache. This process can be quicker than retrieving it again from the content host.

You can configure the Caching Proxy as a reverse or forward proxy server. The cache can be stored on physical storage devices or in memory:

- Forward proxy

  When configured in *forward proxy* mode, the Caching Proxy handles requests from multiple client browsers, retrieves data from the Internet, and caches the retrieved data for future use. In this case, you need to configure the client browser to use the proxy server.

  When a client requests a page, the caching proxy connects to the content host that is located across the Internet. It then sends the request that it received from the client, caches the retrieved data, and delivers the retrieved data to the client. If another client sends the same request, that request is served from the cache. This process decreases network use and provides better response times.

- Reverse proxy

  IP-forwarding topologies use a *reverse proxy* server, such as the Caching Proxy, to receive incoming HTTP requests and forward them to a web server. The web server forwards the requests to the application servers for actual processing. The reverse proxy returns completed requests to the client, masquerading as the originating web server.

If a client then requests the same data the next time, the requests are not sent to the back-end server for processing. Instead, the requests are served from the cache. This method prevents unnecessary back-end processing for the same data requests, and therefore provides better response times.

9.10.1 Installation

Before you install Edge Components, consult the WebSphere Application Server V8.5 Information Center to ensure that all required hardware and software prerequisites are met.

You use Installation Manager to install Edge Components. For detailed installation documentation about Load Balancer and Caching Proxy, see the Websphere Application Server V8.5 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp`

Before starting the installation of Load Balancer, you must complete the planning tasks. Finish the detailed network planning for your environment and have an exact understanding of the data flow in your environment.

Edge Components for WebSphere Application Server V8.5 is shipped with the following versions of Load Balancer:

- Load Balancer for IPv4
- Load Balancer for IPv4 and IPv6

Unless you have a specific requirement to use Load Balancer for IPv4, use Load Balancer for IPv4 and IPv6. The Site Selector, Nortel Alteon Controller, and Cisco CSS Controller are not available with Edge Component installations of Load Balancer for IPv4 and IPv6.

9.10.2 Configuring the Load Balancer

After you install Load Balancer, you must configure it for your environment. Load Balancer provides various configuration options and options for forwarding packets. The following sections give an overview about some of these configuration tools and configuration options.

Configuration methods

The Load Balancer provides the following methods for configuration:

- Command line
- Configuration Wizard
- GUI
- Scripts

Methods for forwarding packages

Load Balancer provides different methods to forward packages to the servers to which they are dispatching:

- Media Access Control (MAC)-level routing
- Encapsulation forwarding

Configuring advisors

Advisors are used to track the health of the servers to which Load Balancer forwards the IP packets. The settings of the advisors are critical in terms of how quickly an outage of a server can be determined. The more frequent the advisor runs, the quicker an outage is determined.

However, because advisors are basically clients for the TCP/IP protocol used to access the server, frequent advisor runs increase server load and network use.

Load Balancer provides built-in advisors. You can also create custom advisors and configure Load Balancer to react based on the response of a custom advisor.

9.10.3 Configuring the Caching Proxy

After you install Caching Proxy, you must configure it for your environment. Caching Proxy provides the following methods for configuration:

- Configuration and Administration forms
- Configuration Wizard
- Manual editing of the Caching Proxy configuration file

9.11 Planning for the DMZ secure proxy

The DMZ secure proxy for WebSphere Application Server is available through a separate installation media and is installed by using Installation Manager. A `secureproxy` profile template is created upon installation of the DMZ Secure Proxy Server and WebSphere Application Server Network Deployment. These two profiles templates are different. The Network Deployment installation provides a `secureproxy` profile template that generates a configuration-only profile. This profile can be used for the administration of the DMZ secure proxy but is not runnable.

The `secureproxy` profile template that comes with DMZ Secure Proxy Server is the base for a proxy server that runs in the DMZ. The proxy server forwards requests to the content servers.

Installation: The DMZ follows the same base principles for the installation as WebSphere Application Server. The DMZ secure proxy differentiates between product binary files and runtime configuration files by using profiles.

Address the following items before you start installing the DMZ Secure Proxy Server:

- Plan your file systems and directories
- Determine whether to perform a single installation or multiple installation
- Select an installation method
- Install updates
- Plan for profiles
- Plan for names
- Plan for TCP/IP port assignments
- Consider security for the installation
- Install IBM Support Assistant Agent

For more information, see 9.7, “Planning for WebSphere Application Server” on page 246.

9.12 Planning for the HTTP server and plug-in

The options for defining and managing web servers depend on your chosen topology and your WebSphere Application Server package. You must decide whether to collocate the web server with other WebSphere Application Server processes and whether to make the web server managed or unmanaged.

The installation process includes the following steps:

1. Install the WebSphere Customization Toolbox.
2. Install a supported web server.
3. Install the web server plug-in by using Installation Manager.
4. Define the web server to WebSphere Application Server by using the Web Server Plug-ins Configuration Tool.
5. Configure a supported web server to an installed web server plug-in.

   WebSphere Customization Toolbox is in the supplements directory of WebSphere Application Server, along with IBM HTTP Server and the web server plug-in.

9.12.1 Web Server Plug-ins Configuration Tool

The Web Server Plug-ins Configuration Tool configures the web server for communicating with the application server. Depending on the topology, it also creates a web server definition in the application server. If the Web Server Plug-ins Configuration Tool cannot create the web server definition in the application server configuration directly, it creates a script. This script can then be copied to the application server system and run to create the web server configuration definition within the application server configuration.

The Web Server Plug-ins Configuration Tool is started from the WebSphere Customization Toolbox. Figure 9-8 shows the main window of the Web Server Plug-ins Configuration Tool.

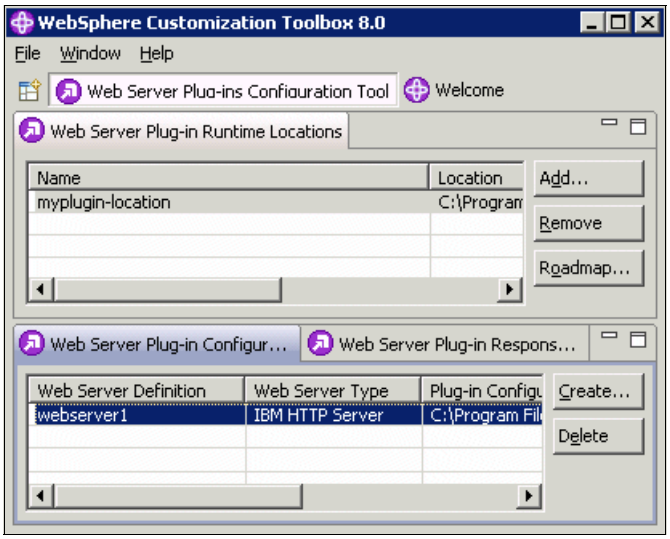


Figure 9-8 Main window of the Web Server Plug-ins Configuration Tool

When the Web Server Plug-ins Configuration Tool GUI is used for plug-in configuration, the selections are saved and are available in a response file (Figure 9-9). Instead of using the Web Server Plug-ins Configuration Tool, use the command-line tool for Web Server Plug-ins Configuration Tool with a response file to configure a web server.

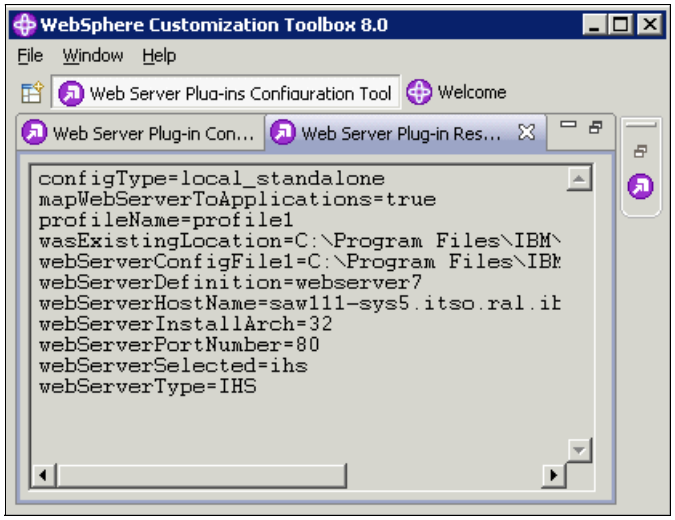


Figure 9-9 Response file for web server plug-in

9.12.2 Stand-alone server environment

In a stand-alone application server environment, a web server can be either remote or local to the application server. However, only one can be defined to the application server. Because there are no managed nodes in a stand-alone environment, the web server is always on an unmanaged node.

Remote web server

In this scenario (Figure 9-10), the application server and web server are on separate systems.

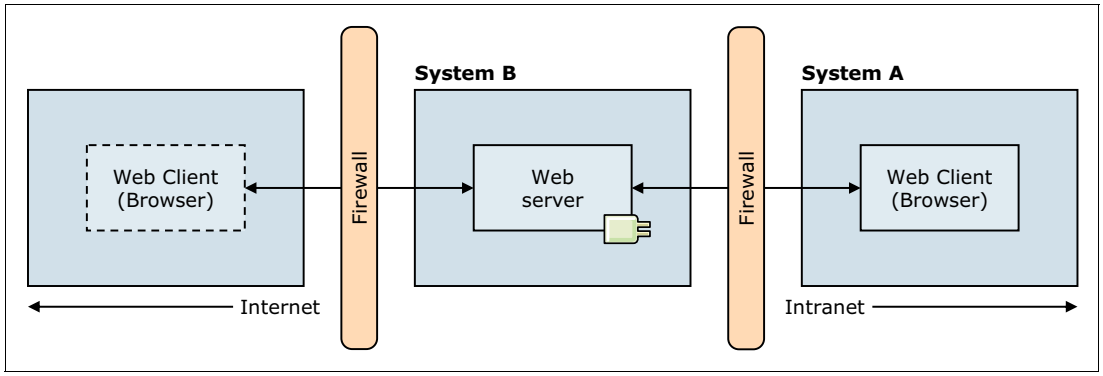


Figure 9-10 Remote web server in a stand-alone server environment

The application server is already installed and configured on system A. To create the environment shown in Figure 9-10, complete the following tasks:

1. Install the web server on System B.
2. Install the web server plug-in on System B.
3. Install the Web Server Plug-ins Customization Tool on System B.

4. Using the Web Server Plug-ins Customization Tool, configure the web server plug-in by performing the following steps:
   a. Select the type of web server.
   b. Identify the web server configuration file and the web server port.
   c. Enter a name for the web server definition to be created. The default is `webserver1`.
   d. Select **Remote** for the configuration scenario.

      Remember: During configuration, the following tasks are run automatically:
      - The web server configuration file is updated with the plug-in configuration, including the location of the plug-in configuration file.
      - A script is generated to define the web server to WebSphere Application Server. The script is in the `plugin_root/bin/configureweb_server_name` directory.

   e. After the configuration is complete, review the information in the Plug-in Configuration Result window (Figure 9-11 on page 277). This window shows the following information:
      - Configuration status
      - Information that describes the next required steps
      - Location of the configuration script
      - The web server type that was configured
      - The web server definition name
      - The name and location of the web server plug-in configuration file

      Optionally, select **Launch the plug-in configuration road map**.
   f. Click **Finish**.

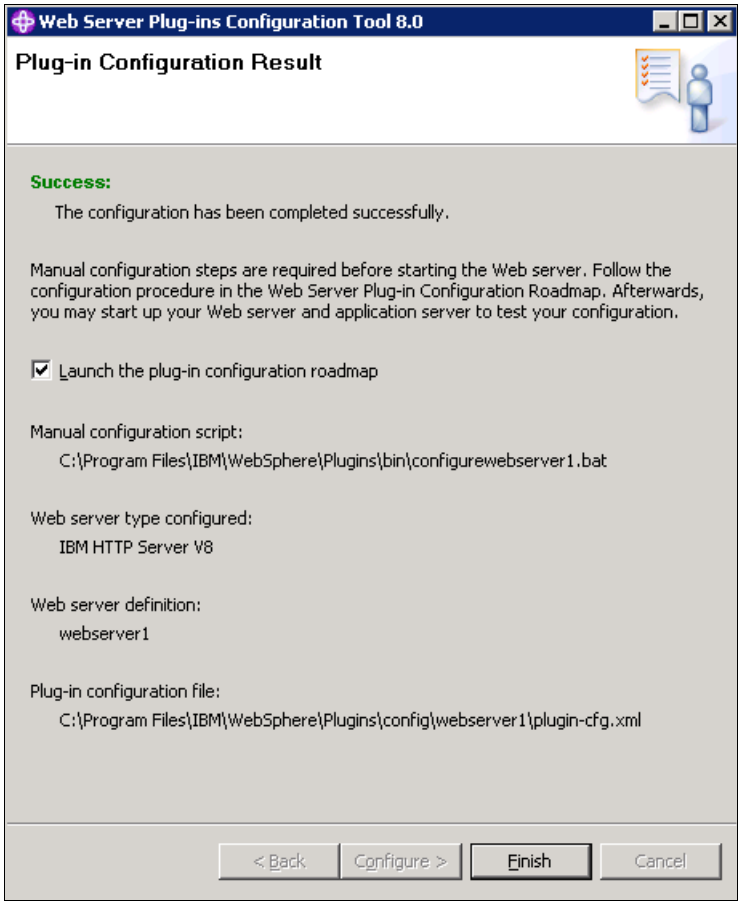


Figure 9-11 Configuration results for a remote configuration scenario

5. Follow the configuration procedure specified in the Web Server Plug-in Configuration road map.

> **Tip:** The **Launch the plug-in configuration road map** option is in the Plug-in Configuration Result window (Figure 9-11). If you do not select this option, click **Roadmap** in the main window of the Web Server Plug-ins Configuration Tool (Figure 9-8 on page 274).

   a. Copy the script to the *app_server_root*/bin directory of the application server system on System A.
   b. Start the application server.
   c. Start the script.

   When the web server is defined to the application server, the plug-in configuration file is generated automatically. For IBM HTTP Server, the new plug-in file is propagated to the web server automatically. For other web server types, you need to copy the new plug-in configuration file to the web server.

Local web server

In this scenario (Figure 9-12), a stand-alone application server exists on System A. The web server and web server plug-in are also installed on System A. This topology is suited to a development environment or internal applications.

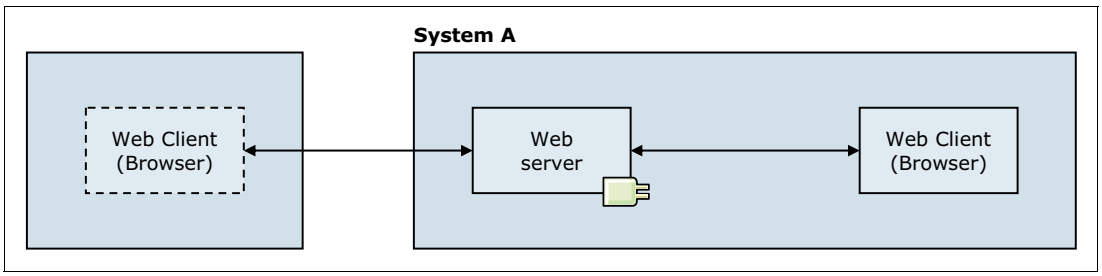


Figure 9-12 Local web server in a stand-alone server environment

In this scenario, the application server is already installed and configured. To create the environment shown in Figure 9-12, complete these steps:

1. Install the web server on System A.
2. Install the web server plug-in on System A.
3. Install the Web Server Plug-ins Customization Tool on System A.
4. Using the Web Server Plug-ins Customization Tool, configure the web server plug-in on System A:
   a. Select the type of supported web server.
   b. Identify the web server configuration file and the web server port.
   c. Enter a name for the web server definition to be created. The default is `webserver1`.
   d. Select **Local** for the configuration scenario and enter the path of the installed WebSphere Application Server, for example: `C:\Program Files\IBM\WebSphere\Appserver` or `/opt/IBM/WebSphere/Appserver`.
   e. Select the profile to be used.

   Remember: During configuration, the following tasks are run automatically:
   - The web server configuration file is updated with the plug-in configuration, including the location of the plug-in configuration file.
   - The WebSphere Application Server configuration is updated to define the new web server.

   f. After the configuration is complete, review the information in the Plug-in Configuration Result window (Figure 9-13 on page 279). This window shows the following information:
      - Configuration status
      - Information describing the next required steps
      - The web server type that was configured
      - The web server definition name
      - The name and location of the web server plug-in configuration file

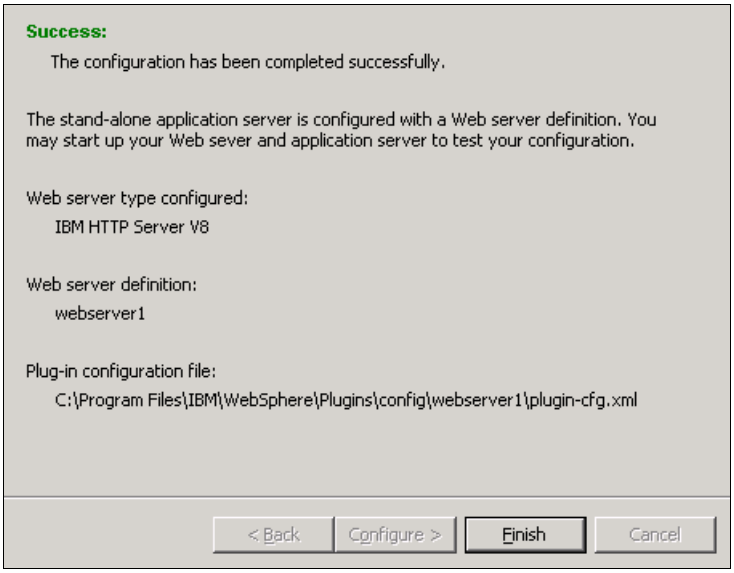


Figure 9-13 Configuration results for a local configuration scenario

In a local scenario, the plug-in configuration file is automatically generated directly in the location from which the web server reads it. Therefore, this file does not need to be propagated to the web server when it is regenerated. Then click **Finish**.

5. Follow the configuration procedure specified in the Web Server Plug-in Configuration road map. You can access the road maps by clicking **Roadmap** from the main window of the Web Server Plug-ins Configuration Tool (Figure 9-8 on page 274).

9.12.3 Distributed server environment

Web servers in a distributed server environment can be local to the application server or remote. The web server can also be on the deployment manager system. You can define multiple web servers. The web servers can be on managed or unmanaged nodes.

Remote web server on an unmanaged node

In this scenario, the deployment manager and the web server are on separate systems. The process for this scenario is almost identical to the process for a remote web server in a stand-alone server environment. The difference is that the script that defines the web server is run against the deployment manager. You see an unmanaged node created for the web server node.

In Figure 9-14, the node is unmanaged because no node agent is on the web server system.

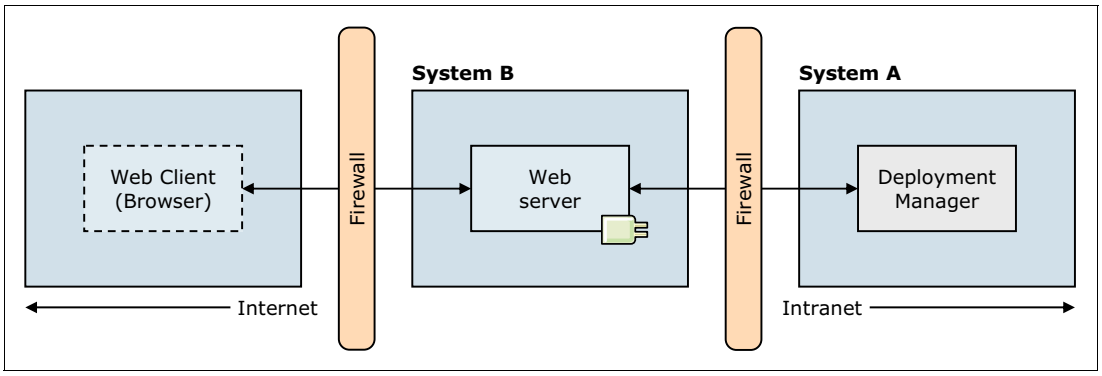


Figure 9-14 Remote web server in a stand-alone server environment

In this scenario, the deployment manager is already installed and configured on System A. To create the environment shown in Figure 9-14, complete these steps:

1. Install the web server on System B.
2. Install the web server plug-in on System B.
3. Install the Web Server Plug-ins Customization Tool on System B.
4. By using the Web Server Plug-ins Customization Tool, configure the web server plug-in on System B:
   a. Select the type of supported web server.
   b. Identify the web server configuration file and the web server port.
   c. Enter a name for the web server definition. The default is `webserver1`.
   d. Select **Remote** for the configuration scenario.

   Remember: During configuration, the following tasks are run automatically:
   - The web server configuration file is updated with the plug-in configuration, including the location of the plug-in configuration file.
   - A script is generated to define the web server to WebSphere Application Server. The script is in the `plugin_root/bin/configureweb_server_name` directory.

   e. After the configuration is complete, review the information in the Plug-in Configuration Result window (Figure 9-11 on page 277). This window shows the following information:
      - Configuration status
      - Information describing the next required steps
      - Location of the configuration script
      - The web server type that was configured
      - The web server definition name
      - The name and location of the web server plug-in configuration file

      Optionally select **Launch the plug-in configuration road map**.
   f. Click **Finish**.

5. Follow the configuration procedure specified in the Web Server Plug-ins Configuration road map.

   Tip: The **Launch the plug-in configuration road map** option is in the Plug-in Configuration Result window (Figure 9-11 on page 277). If you do not select this option, click **Roadmap** in the main window of the Web Server Plug-ins Configuration Tool (Figure 9-8 on page 274).

   a. Copy the script to the *app_server_root*/bin directory of the application server system, System A.

   b. Make sure that the deployment manager and the node agent are running.

   c. Run the script.

   When the web server is defined to WebSphere Application Server, the plug-in configuration file is generated automatically. For IBM HTTP Server, the new plug-in file is propagated to the web server automatically. For other web server types, you need to propagate the new plug-in configuration file to the web server.

   Restriction: Propagation of a plug-in configuration to remote web servers is supported only for IBM HTTP Servers that are defined on an unmanaged node.

Local to a federated application server (managed node)

In this scenario (Figure 9-15), the web server is installed on a system that also has a managed node.

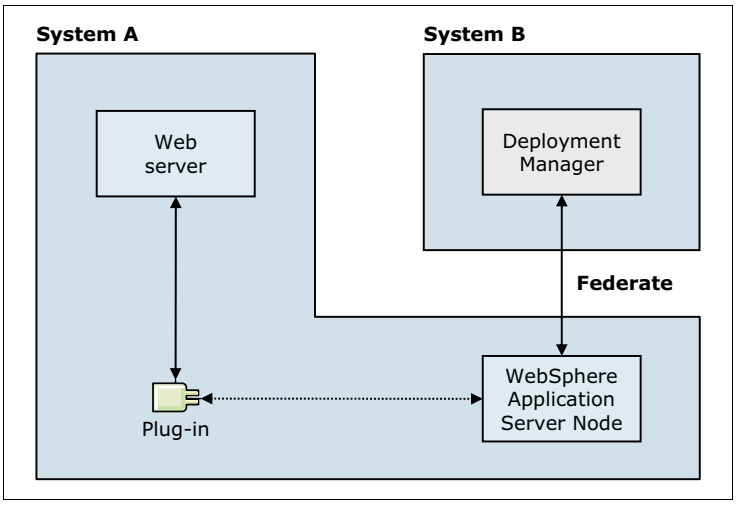


Figure 9-15 Web server installed locally on an application server system

In this scenario, the application server is already installed, configured, and federated to the deployment manager cell. To create the environment shown in Figure 9-15, complete these steps:

1. Install the web server on System A.
2. Install the web server plug-in on System A.
3. Install the Web Server Plug-ins Customization Tool on System A.

4. Using the Web Server Plug-ins Customization Tool, configure the web server plug-in on System A:
   a. Select the type of supported web server.
   b. Identify the web server configuration file and the web server port.
   c. Enter a name for the web server definition. The default is `webserver1`.
   d. Select **Local** for the configuration scenario, and enter the path of the installed WebSphere Application Server, for example: `C:\Program Files\IBM\WebSphere\Appserver` or `/opt/IBM\WebSphere/Appserver`.
   e. Select the profile to be used.

      Remember: During configuration, the following tasks are run automatically:
      - The web server configuration file is updated with the plug-in configuration, including the location of the plug-in configuration file.
      - A script is generated to define the web server and a managed node to WebSphere Application Server. The script is in the `plugin_root/Plugins/bin/configureweb_server_name` directory.

   f. After the configuration is complete, review the information in the Plug-in Configuration Result window (Figure 9-13 on page 279). This window shows the following information:
      - Configuration status
      - Information about the next required steps
      - The web server type that was configured
      - The web server definition name
      - The name and location of the web server plug-in configuration file

      Optionally select **Launch the plug-in configuration road map** option.
   g. Click **Finish**.
5. Follow the configuration procedure specified in the Web Server Plug-ins Configuration road map.

   Tip: The **Launch the plug-in configuration road map** option is in the Plug-in Configuration Result window (Figure 9-11 on page 277). If you do not select this option, click **Roadmap** in the main window of the Web Server Plug-ins Configuration Tool (Figure 9-8 on page 274).

   a. Copy the script to the *app_server_root*`/bin` directory on System A.
   b. Make sure that the deployment manager and the node agent are running.
   c. Run the script.

   The deployment manager configuration is updated and propagated back to System A at node synchronization. The plug-in configuration file is generated automatically and propagated at the next node synchronization.

   For security reasons, avoid installing managed web servers in the DMZ.

9.13 IBM Support Assistant

IBM Support Assistant is available at no additional cost. This tool helps you to research, analyze, and resolve problems by using various support features and problem determination tools. With IBM Support Assistant, you can determine the cause for most problems faster and find solutions in a shorter time, increasing the availability of your installation. IBM Support Assistant provides many different tools for problem determination and materials collections. With this tool, you can organize and transfer troubleshooting efforts between members of your team, or send data to IBM for further support.

The IBM Support Assistant includes the following features:

- IBM Support Assistant Workbench

  The IBM Support Assistant Workbench, or simply “the Workbench,” is the client-facing application that you can download and install on your workstation. By using the Workbench, you can use all the troubleshooting features of the Support Assistant. These features include Search, Product Information, Data Collection, Managing Service Requests, and Guided Troubleshooting. The Workbench can run these functions only on the system where it is installed (except for the Portable Collector). For more information about the tools that are available in IBM Support Assistant, see IBM Support Assistant Tool Add-Ons List at:

  http://www.ibm.com/support/docview.wss?rs=3455&uid=swg27013116

- IBM Support Assistant Agent

  The IBM Support Assistant Agent, or simply “the Agent,” is software that you must install on every system that you need to troubleshoot remotely. After an Agent is installed on a system, it registers with the Agent Manager. Then you can use the Workbench to communicate with the Agent. You can also use features such as remote system file transfer, data collections, and inventory report generation on the remote system.

- IBM Support Assistant Agent Manager

  You need to install the IBM Support Assistant Agent Manager, or simply “the Agent Manager,” only one time in your network. The Agent Manager provides a central location where information about all available agents is stored, and acts as the certificate authority. For remote troubleshooting to work, all Agent and Workbench instances register with this Agent Manager. Any time a Support Assistant Workbench needs to run remote functions, it authenticates with the Agent Manager and gets a list of the available Agents. Then, the Workbench can communicate directly with the Agents.

Tip: For more information about IBM Support Assistant and installation instructions, see the IBM Software Support page for IBM Support Assistant at:

http://www.ibm.com/software/support/isa/

9.14 Installation checklist

When planning for your installation, consider the following checklist:

- Examine your selected topology to determine hardware needs and software licenses. Create a list of the software to install on each system. In this software list, note the software version levels that are necessary to support the software integration requirements.
- Determine the WebSphere Application Server profiles that you need to create and whether you will create them during or after installation. Decide on a location for the profile files (see 9.4, “Planning for disk space and directories” on page 240).
- Develop a naming convention that includes system naming and WebSphere Application Server component naming.
- Develop a strategy for managing certificates in your environment, including personal certificates and signed certificates.
- Develop a strategy for assigning TCP/IP ports to WebSphere processes.
- Select an installation method (wizard, silent, or centralized installation manager).
- Plan an administrative security strategy that includes a user repository and role assignment.
- Determine the user ID to use for installation and whether you perform the installations by using administrator, non-administrator, or group mode.
- Plan for the web server and web server plug-in installation. Determine whether the web server is a managed or unmanaged server, and note the implications. Create a strategy for generating and propagating the web server plug-in configuration file.

9.15 Resources

WebSphere Application Server ships with an installation guide that you can access through the Launchpad. For more information about the installation process, see the WebSphere Application Server V8.5 Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp

10

Performance, scalability, and high availability

This chapter provides information about the aspects to consider for implementing a capable, scalable, and highly available WebSphere Application Server V8.5 environment. These three requirements are interrelated. For example, to increase the performance of your environment, you need to add additional resources. To add additional resources efficiently, you need a scalable design and workload management to spread the requests across all available components. By adding additional resources, in most cases, you introduce redundancy, which is a prerequisite for high availability.

This chapter includes the following sections:

- Performance, scalability, and high availability features in WebSphere Application Server V8.5
- Scalability
- Performance
- WebSphere Application Server performance tools
- Workload management
- High availability
- Caching
- Session management
- Data replication service
- Highly available deployment manager
- Whole-system Analysis of Idle Time Tool
- Checklist for performance, scalability, and high availability
- References

10.1 Performance, scalability, and high availability features in WebSphere Application Server V8.5

WebSphere Application Server V8.5 provides features that improve the performance, scalability, and high availability of the application infrastructure. This section provides information about the following features:

- Default garbage policy gencon
- JVM garbage policy: Balanced
- JVM garbage policy: Metronome
- High Performance Extensible Logging
- Disabling WebSphere MQ functions
- Java Persistence API L2 cache provided by the dynamic cache provider
- Collecting Java memory dumps and core files
- Enabling request-level granularity of reliability, availability, and serviceability
- Resource workload routing
- External high availability framework for service integration
- High availability for a WebSphere MQ link

10.1.1 Default garbage policy gencon

The default garbage policy of WebSphere Application Server V8.5 is the generational concurrent or *gencon*. This policy replaces the `optthruput` policy. The gencon strategy manages objects by their lifetimes. The heap is composed of the following areas:

- Tenured space for old objects
- Nursery space for new objects

Objects are promoted from the nursery space to the tenured space based on their age.

Gencon is the garbage collector policy for transactional applications, where the objects do not survive after the transaction ends, and for applications with many short-lived objects.

Tip: The gencon policy might require more memory than the optthruput policy. You can begin sizing by setting the tenured area to the previous heap value, and then allocating additional memory to the nursery area.

10.1.2 JVM garbage policy: Balanced

Balanced Java virtual machine (JVM) garbage policy is available with WebSphere Application Server V8.5. The strategy of this policy is to divide the heap between potentially thousands of regions, with each region individually managed. Objects are allocated in these empty regions. This region area is called the *eden space*. Partial garbage collection runs when the eden space is full to free memory.

The balanced policy is designed for large heap sizes. It can be useful when you use the gencon policy with a heap size greater than 4 GB, or if you use large arrays.

For more information, see the Websphere Application Server V8.5 Information Center. Search for the phrase *Balanced Garbage Collection policy:*

`http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp`

10.1.3 JVM garbage policy: Metronome

A new Metronome garbage collection policy is included with the IBM SDK for Java 7 for evaluation purposes. It is a real-time incremental garbage collection policy and is supported through WebSphere Real Time.

The Metronome Garbage Collector consists of two types of threads: An alarm thread and a number of collection threads. The single alarm thread *wakes* at regular intervals to check whether insufficient free space is available and no garbage collection is taking place. If these conditions are met, the alarm thread triggers the collection threads to start garbage collection. A number of collection threads mark live objects so that unmarked objects are available for collection.

The benefit of the Metronome Garbage Collector is that the time it takes is more predictable. This predictability allows garbage collection to take place at set time intervals.

The key difference between Metronome garbage collection and standard garbage collection is that Metronome garbage collection occurs in small interruptible steps. The standard garbage collection stops the application as it marks and collects garbage.

For more information about Metronome Garbage Collector, see the WebSphere Real Time Information Center at:

http://publib.boulder.ibm.com/infocenter/realtime/v1r0/topic/com.ibm.rt.doc.10/realtime/rt_options_metro.html

10.1.4 High Performance Extensible Logging

WebSphere Application Server V8.5 provides a High Performance Extensible Logging (HPEL) logging and tracing feature. HPEL outperforms basic logging methods. It writes logs and traces to a log data repository and a trace data repository in a binary format. A text log file can also be generated, but doing so affects the performance. A log viewer is provided to view, filter, and format the log and trace data.

You can run performance tests with the logging enabled, which can improve application performance if your applications use log files intensely.

10.1.5 Disabling WebSphere MQ functions

By default, WebSphere MQ functions are enabled. To support these functions, application server resources are continually used. If you do not want to take advantage of these functions, you can disable them to improve performance.

10.1.6 Java Persistence API L2 cache provided by the dynamic cache provider

The Java Persistence API 2.0 has standardized the second level (L2) cache. WebSphere Application Server supports Java Persistence API 2.0. The dynamic cache service plugs in as an L2 cache provider to the Java Persistence API. The L2 cache improves performance by avoiding direct requests to the database. The L2 cache also uses additional memory, which limits its size.

10.1.7 Collecting Java memory dumps and core files

You can now produce Java memory dumps, Java core, and system dump files directly by using the administrative console. These files are useful when analyzing performance issues, such as memory, thread, and system behaviors.

10.1.8 Enabling request-level granularity of reliability, availability, and serviceability

WebSphere Application Server V8.5 provides a request-level granularity of reliability, availability, and serviceability (RAS). It is provided for HTTP, Internet Inter-ORB Protocol (IIOP), optimized local adapter, and certain message-driven bean (MDB) requests within the same application server. With this feature, you can define the granularity of your requests and improve the RAS of your application server. With RAS granularity, you can assign different sets of RAS attribute values (such as timeout values, timeout actions, and trace settings) to different sets of requests.

To set up RAS, you must configure a workload classification document and specify the document in the administrative console's environment variables.

10.1.9 Resource workload routing

WebSphere Application Server V8.5 provides a feature that enhances availability by configuring failover resources to a data source and connection factory. You create alternative resources for the data source and connection factory. These resources must be identical to the primaries and be compatible with the applications. The data source and connection factory can fail over when a failure occurs, and then fail back when the situation returns to normal. Only one resource can be used at a time, and the alternate is available only when the primary fails.

10.1.10 External high availability framework for service integration

WebSphere Application Server V8.5 allows a message engine to be managed by an external high availability framework such as IBM PowerHA. The message engine is enabled or disabled only when the external high availability framework through HA manager orders it.

This feature is mandatory when the message engine stores the data in a database that uses a high availability framework to recover. Both the message engine and database must be in the same external cluster.

10.1.11 High availability for a WebSphere MQ link

To improve the high availability connection between WebSphere Application Server and WebSphere MQ, you can configure a list of connection names for the WebSphere MQ link sender channel. If the active gateway queue manager fails, the Service Integration Bus can reconnect to a standby gateway queue manager. Resiliency of the bus improves with the use of the Intelligent Management feature. For more information, see 13.4, "Enhanced resiliency for the service integration bus in V8.5" on page 425.

10.2 Scalability

This section provides information about the scalability of the WebSphere Application Server environment.

10.2.1 Scaling overview

Scalability is the ability of the infrastructure to properly handle an increase of the load. In many cases, scalability means getting increased throughput by adding more resources.

Important: For your infrastructure to be scalable, your applications need to be scalable. If an application is not designed to be scalable, the scalability options are limited.

Understanding the scalability of the components in your WebSphere Application Server infrastructure and applying appropriate scaling techniques improves availability and performance. Modifying the scalability approaches of the infrastructure affects availability and performance.

Consider additional resources as a step to improve performance. You can scale every component of your architecture. By analyzing your workload characteristics, you can define the components that are used the most so you can give them priority.

You can improve performance when adding resources by using the following methods:

- Scaling up (or *vertical scalability*)

  Vertical scaling means increasing the application throughput by adding resources inside the server to extend processing capability. This concept is relatively easy to implement and can be applied to any server in an environment until you reach the hardware limits. You do not need to change your application code.

  For example, you can double the number of processors and memory for a server. By upgrading the system hardware with additional processors, the server can potentially reach a higher throughput.

- Scaling out (or *horizontal scalability*)

  Horizontal scaling means increasing the application throughput by adding additional servers to handle the load. Find the best configuration for one server and then multiply that configuration to get the number of servers required to handle the load.

  For example, instead doubling the number of processors and memory in the server, add a second, identical server.

  Scaling out is also used to improve high availability by limiting single points of failure (SPOFs). Scaling out is sometimes the only solution when you are limited by hardware resources. Horizontal scalability can require other infrastructure components (as load balancers) to share the load between the instances. In addition, administrators must support and maintain multiple systems.

Every additional component, such as processor, memory, or JVM, in your infrastructure increases the resources needed for management. You need to define the *scalability factor* for each new component. The scalability factor is the percentage of effective service for this component. For example, adding one server requires 15% processor capacity. The effective use of each new server is 85%, and the scalability factor is 0.85.

A scalability factor of one means that the scalability of your application is *linear*. You always have the same additional throughput improvement when you add resources. It is rare to have applications with a scalability factor greater than or equal to 1.

In addition, be aware that the law of diminishing returns plays a role when using either the vertical or horizontal scaling technique. The *law of diminishing returns* is an economics principle. This principle states, if one factor of production is increased and all other factors remain constant, the overall returns reach a peak and then decrease.

This law can be applied to scaling in computer systems as well. This law means that adding two additional processors will not necessarily grant twice the processing capacity. Nor will adding two additional horizontal servers in the application server tier necessarily grant you twice the request serving capacity. Additional processing cycles are required to manage those additional resources. Although the degradation might be small, it is not a direct linear function of change in processing power. Adding *n* additional systems does not result in *n* times the throughput.

For example, in a single-tier scenario, the web application and database servers are all running on the same system. You decide to scale by creating a cluster and spreading application servers across systems to improve the throughput. However, additional systems introduce new communication traffic and load to the database server. Consider the following questions:

- How much network bandwidth will this server configuration consume?
- What will be the performance improvement by adding more systems?

Scalability testing can be arranged as a part of the performance testing. It is crucial that you determine whether the scaling techniques are effective and whether they adversely affect other areas. Measure throughput and response time to ensure that the results meet your expectations.

Summary: You can implement both scaling approaches (scale up and scale out) to improve the performance with the following advantages and considerations:

- Scale up
  - Is easier and faster to implement
  - Does not need to change your application code
  - Can be limited by the hardware
- Scale out
  - Is more complex to implement
  - Brings servers high availability
  - Needs other components to share the load
  - Need to manage additional servers

10.2.2 Scaling the infrastructure components

This section highlights key points in scaling your application server components. Before investing in additional resources or making changes, examine the entire application environment to identify potential bottlenecks.

Network

When scaling at the network layer, such as with firewalls or switches, the most common solution is vertical scaling. Network devices have processing capacity and use memory much like any other hardware resource. Adding hardware resources to a network device increases

the throughput of that device, which positively impacts the scaling of that device. For example, moving from 1 Gb to 10 Gb connections can significantly increase performance at the network layer.

HTTP server

Both scaling approaches are viable. Scaling the HTTP server means creating more threads or processes to handle more requests in parallel.

You can implement one of the following solutions:

- Vertical scalability to create multiple instance of the web server on the same system, or add more threads or processes to your existing web server instances
- Horizontal scalability to create multiple instances of the web server on different systems

To support a configuration with multiple web server instances, use load balancers. Be careful when adding servers that the load balancer has adequate capacity. If it does not, you will shift the bottleneck from the web tier to the load balancing tier. Also, make sure that the additional request throughput can be handled by your application servers.

Opportunity: IBM HTTP Server for z/OS offers the unique feature of scalable mode. With scalable mode, WLM for z/OS can start additional interconnected clones of HTTP server. This configuration offers vertical scalability if performance goals are not met.

DMZ secure proxy

The DMZ secure proxy provides horizontal and vertical scaling capabilities in addition to the scaling activities on a per server basis. When scaling vertically, make sure that you have sufficient resources. Also, be aware that this configuration provides only limited high availability. In any scaling scenario, you need an IP sprayer, such as the Edge Components, to spread the incoming traffic across all proxy servers.

JVM

You can scale at the application server layer with vertical scaling, horizontal scaling, or both.

You can add resources, such as memory for the heap and more threads in the different containers to your existing JVMs. However, this approach might be limited by the size of the JVM heap and the available memory on the physical system.

You can also create multiple JVMs. A WebSphere Application Server JVM can be clustered by providing multiple copies of the same JVM. These copies scale vertically when on the same physical machine, horizontally when on different systems, and simultaneously when both scenarios are applied. When using the vertical approach, you might be limited by the system (number of processors or memory available).

Connection pools

To process the requests, many connections are managed between the infrastructure layers. To be scalable, design a solution that limits the number of connections, and avoids establishing additional connections. To minimize the impact, use connection pools. At the HTTP server layer, you can keep the connection between the browser and the HTTP server. WebSphere provides several pools to connect, for example, to the database or the Java Message Service (JMS) destinations.

Service integration bus

A service integration bus and a message engine are key components in an infrastructure. WebSphere Application Server offers the ability to scale messaging resources. You can use the scalability policy provided by WebSphere: One service integration bus is hosted by a WebSphere Application Server cluster. You can choose a horizontal cluster, a vertical cluster, or both. Each cluster member has one message engine. All the message engines are active at the same time, and each message engine can run only on its own JVM. If the JVM fails, the message engine is unavailable.

WebSphere Application Server manages the workload between the message engines. For the workload management, you must create partitioned destinations to enable a single logical queue to spread across multiple messaging engines. For *n* cluster members, the theory is that each member receives an *n*th number of messages. One key factor to consider in this design is that message order is not preserved. Not preserving messages might or might not be significant, depending on the nature of the application.

If the applications need high availability or scalability, WebSphere Application Server provides the high availability policies. Websphere Application Server V8.5 features a setup wizard for these policies that can be used for most topologies.

Database

You can use horizontal or vertical techniques to improve performance at the database layer. The most common technique is to scale up by adding more resources, such as memory or processors, to support the new load. Most of the databases provide a solution to scale out by adding multiple nodes. These solutions can be complex and expensive. Your applications must be multi-node aware to take advantage of the configuration, and must limit the network traffic between nodes.

10.3 Performance

To review the performance of the environment and the scalability techniques, you must first define the performance requirements. Then you must tune the environment to reach these requirements. This section provides information about the performance of the WebSphere Application Server components and the WebSphere Application Server performance tools.

Keep in mind that 80% of the tuning is made on the application, middleware, and database layers. The remaining 20% tunes the hardware and operating system layer.

10.3.1 Performance considerations

Performance is one of the most important nonfunctional requirements for any WebSphere environment. Application performance must be tracked continuously during your project.

Before switching your new environment to production, a real performance run campaign is mandatory to determine whether your infrastructure is correctly sized. Performance problems are by far the most user-visible problem that you can have. Most users are willing to accept small functional problems when a system is rolled out. However, performance problems are unacceptable to most users and affect everyone who is working on the system. Make sure to perform load tests that represent a realistic user load against your system.

This section provides information about how to manage a real performance run campaign activity.

Although performance is often subjective, performance requirements must be measurable for evaluation. Establish success criteria to evaluate the success of your scaling tasks. Consider the following targets:

- Throughput

  Throughput measures the number of requests in a period that the system can process. For example, if an application handles 10 client requests simultaneously and each request takes one second to process, the potential throughput is 10 requests per second.

- Response time

  Response time is the period from entering a system at a defined entry point until exiting the system at a defined exit point. In a WebSphere Application Server environment, this measurement is usually the time it takes for a request submitted by a web browser to be received at the web browser.

- A maximum time frame for batch style applications

  Batch applications often run during a defined time frame during the night to take advantage of low peak hours. This setup avoids disturbing application customers during the day. The *maximum time frame* is the time window for the batch application to run.

- Maximum used resources

  Another criteria, mainly for batch applications, is for the applications to use all of the resources in the system.

To measure the success of your tests, you need to generate a workload that meets the following characteristics:

- Measurable

  The metric must be quantifiable, such as throughput and response time.

- Reproducible

  The same results can be reproduced when the same test is run multiple times. Run your test in the same conditions to define the real impacts of the tuning changes. Change only one parameter at a time.

- Static

  The same results can be achieved regardless of how long you execute the run.

- Representative

  The workload must realistically simulate the stress to the system under normal operating considerations. Run your tests in a production-type environment with the same infrastructure and the same amount of data.

You can follow the tuning approach by using a top-down method to eliminate bottlenecks. For more information, see 10.3.6, “Tuning approach” on page 296.

10.3.2 Application design issues

Many performance problems cannot be fixed by using more hardware or tuning WebSphere parameters. Make performance testing and tuning part of your project schedule, development process, and release cycles to avoid problems later.

Important: Use application profiling techniques when you develop your application. With this approach, the development team can identify bottlenecks in the applications and hot spots where many resources are consumed. Hot spots can often be removed with little effort.

It takes much more effort and money to correct issues after they occur in production than to fix them up front. If performance testing is part of your development cycle, you can correct issues with your application design much earlier. Performance testing results in fewer issues when using your application in the production environment.

10.3.3 Establishing requirements

You must define the success criteria of performance. Without a goal or target, you cannot determine whether the performance campaign was successful. Also, avoid abstract success criteria, such as a “We need to achieve the best that we can have” goal. Keep in mind that performance testing can be endless if you do not have target figures to reach. Without specifics, each time you test, you will find a new bottleneck to solve and a new solution to discover. In the end, it will be impossible for you to define whether the test is a success or failure without tangible goals and outcomes.

The target objectives must be defined in cooperation with the functional team:

- Throughput, for example, transactions per second or payments per hour
- A combination of the number of users and a response time for HTTP pages
- A maximum time frame for batch-style applications
- Maximum number of resources used

Do not waste time performance tuning a system that was improperly sized and cannot withstand the load.

10.3.4 Tips for setting up the test environment

When running performance tests, follow these general tips:

- Run your tests in a production-like environment.

  By using an environment that is as close as possible to the production environment, you can extrapolate the test results to the production environment. If you are starting with a new environment, use the future production environment for your testing purposes before going live.

- Use the same amount of data as in production.

  For your database, use the same amount of data as in production. The size of the database has a significant impact on the performance. Do not take only a part of the data. The difference between the performances can bring an inappropriate result. After each test, you must restore the database to run the same test again in the same conditions.

- Ensure exclusive access to the environment for the test.

  Make sure that no one else is using the test systems. Also, ensure that no background processes are running that consume more resources than what you find in production. For example, if the intent is to test performance during the database backup, make sure that the backup is running.

  If you are using shared or virtual hardware components, make sure no one is using them during the performance run period. For example, if another application uses the storage box at the same time, your disk response times will be higher. The overall response time will also be higher.

- Isolate network traffic as much as possible.

  Make sure that your network isolates the testing environment as much as possible before starting. Performance degradation of the network can create unexpected results.

To limit the network impact, configure separate VLANs for your different usages:

- Administration
- Application
- Injection

- Use monitoring options.

  Use monitoring tools to check the health of the environment during performance tests. Two levels of monitoring must be performed:

  - Debug monitoring

    The goal of this type of monitoring is to identify possible bottlenecks or reasons for problems. The debugging level is detailed and uses additional resources. This level of monitoring affects the test results by more than 15%, depending on the type of log.

  - Production monitoring

    After identifying and solving performance issues by using the debug monitoring level, perform another test. Use the same set of monitoring options that you will use in your target environment. Use this setting to satisfy the service level agreements (SLAs).

- Monitor resource use.

  Check for processor, memory, and disk use before, during, and after each test run to look for any unusual patterns. If the target environment is using shared infrastructure, make sure that the shared component is running under the projected shared load.

- Perform repetitive tests.

  Reset the environment to a defined start state, which includes restoring the database and clearing the different caches. However, do not run your tests with your caches empty. You can fill the caches by running a part of the test before the real one.

- Change only one parameter at a time and document all changes.

Important: To be comparable, run each test in the same conditions. If you do not, you cannot determine the real impact of your tuning change.

10.3.5 Load factors

Your load scenarios reflect your future environment use as close as possible. The following factors are most important in determining how you conduct load tests. Choose from the following options based on the results of your performance tests:

- Online transaction processing (OLTP) workload
  - Request rate
  - Concurrent users
  - Usage patterns
- Batch workload
  - Number of input files
  - Size of the input files

This list is not complete, considering that other factors can become more important depending on the site that is being developed.

Request rate

The *request rate* represents the number of requests per time unit, which is mostly expressed as the number of HTTP requests per second.

Concurrent users

The number of *concurrent users* indicates the numbers of users who are concurrently requesting service from your environment. This number of users is actively sending requests to your system at a specific time.

In contrast to concurrent users, you might also consider the following types of users:

- Active users

  The number of *active users* indicates all users who are currently using resources (for example, in the form of session data) in your environment. It includes users who are reading the response, entering data, and so on.

- Named users

  Named users are users who are defined in the overall environment. The number of named users is usually a large number compared to the number of concurrent users.

Usage patterns

Consider how your users will use the site. You might want to use the cases that your developers defined for their application design as input to build your usage patterns. This information makes it easier to later build the scenarios that the load test will use.

Usage patterns consist of the following factors:

- Use cases modeled as click streams through your pages
- Weights applied to your use cases

Combining weights with click streams shows you how many users you expect in each of your application components and where they generate load.

Notify your developers of your findings so that they can apply them to their development effort. Make sure that the most common use cases are the ones where most of the performance optimization work is run.

To use this information later when recording your load test scenarios, write a report with screen captures or URL paths for the click streams (user behavior). Include the weights for your use cases to show the reviewers how the load was distributed.

Number and size of input files

The number of files that are currently processing determine your level of parallelism. If your application can handle multiple threads, you can determine the correct number of input files.

10.3.6 Tuning approach

Tuning the infrastructure is an iterative process that involves optimizations in each of the environment layers.

First, run your performance tests and then compare them with your requirements:

- Performance meets your objectives.

  If the performance meets your objectives, make sure that you plan for future growth and that you are meeting all of your performance goals. After that, document your findings in a performance tuning report and archive it. Include all of the settings that you changed to reach your objectives.

- Performance is slower than required.

  Clearly determine what is considered slow in your environment:

  – Does it include everything or only particular requests?
  – Does it include everyone or only particular users?
  – Does the slow response occur with just one request, or when under a heavy load?

  To find which components are impacted, start from the application and go down to the lower layers:

  – Application
  – Middleware
  – Operating system or hardware

  To analyze the performance, you must collect information such as logging and tracing. Each layer has monitoring tools or performance metrics. Based on your analysis, you can find the bottleneck and apply the correct tuning or application change. Sometimes, the solution is to add more hardware resources, such as processor and memory. Rerun the performance tests and redo the same process until your performance requirements are met.

  If performance issues persist, you must start over with the sizing and ask the following questions:

  – Were any of the application characteristics underestimated during the initial sizing? If so, why?
  – Was the workload underestimated?
  – Is it still possible to change parts of the application to improve performance?
  – Is it possible to obtain additional resources?

Figure 10-1 summarizes the performance testing approach in a flow chart format.

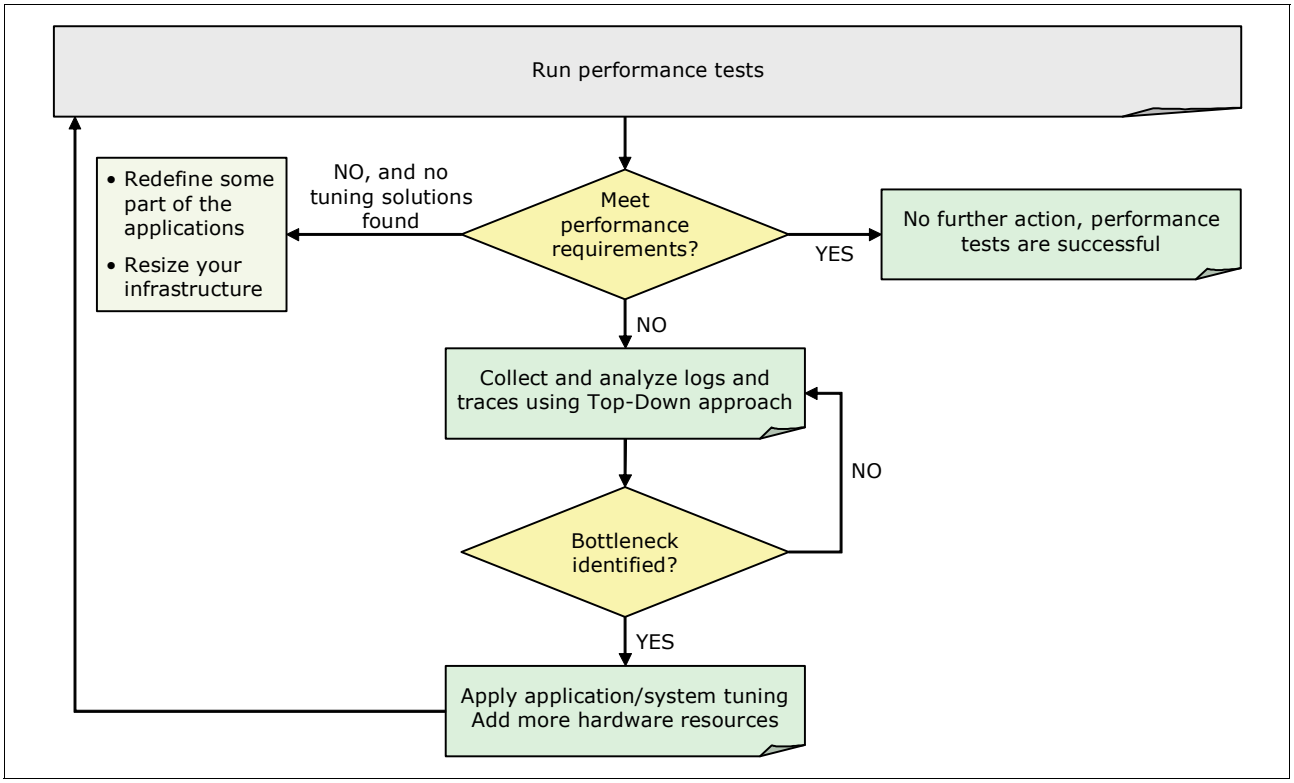


Figure 10-1 Performance approach

When your performance campaign is finished, carefully update your production environment.

10.3.7 Production system tuning

At the end of your tuning process, you must upgrade your production environment. This process is after you find the correct performance, scalability, and high availability balance between the application, middleware, and system.

When changing a production environment, use the following standard practices:

- Change only one parameter at a time.
- Document all changes.
- Compare several test runs to the baseline.

Important: Keep in mind that you often have only one chance to get performance tuning correct. After your environment is in production, you cannot run other performance tests because the production system cannot be taken offline. Normally, a production system is only tested if it is running in a severely degraded state.

10.3.8 Application tuning

The most important part of your tuning activities is spent on the application. Most performance-related problems are related to application design and development implementations. Only a well-designed application, developed with the preferred practices for programming, can provide good throughput and response times. Although environment-related tuning is important to optimize resource use and avoid bottlenecks, it cannot compensate for a poorly written application.

Review the application code itself as part of the regular application lifecycle. Ensure that it uses the most efficient algorithms and the most current application programming interfaces (APIs) that are available for external dependencies. For example, use optimized database queries or prepared statements instead of dynamic SQL statements. To help you in this task, you can optimize the application performance by using application profiling.

For more information about application design considerations, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-express-dist&topic=cprf_appdesign

10.3.9 WebSphere environment tuning

The WebSphere Application Server environment has many settings that can improve performance. This section provides a list of settings to consider for performance when designing a WebSphere Application Server environment, but does not directly explain the tuning parameters.

Web server

Tune the web server with the WebSphere plug-in carefully. Several configuration options can affect the performance. Such options include the number of concurrent requests, keep-alive settings, or Secure Sockets Layer (SSL) parameters. The number of concurrent requests is the most critical factor. The web server must facilitate sufficient concurrent requests to make full use of the application server infrastructure, and also act as a filter. The web server must

keep users waiting in the network and avoid flooding the applications servers with more requests than the system can handle.

You can set a rough initial start value for testing the maximum concurrent threads. One thread can handle one request at a time. In this case, the value 1.2 allows 20% of the threads to serve static content from the web server.

`MaxClients = (((`*`TH`* `+` *`MC`*`) * `*`WAS`*`) * 1.2) / `*`WEB`*

where:

`TH` is the number of threads in the web container

`MC` is the MaxConnections setting in the `plugin-cfg.xml`

`WAS` is the number of Websphere Application Server servers

`WEB` is the number of web servers

The web server configuration provides many processes, and each process has several threads attached. You must find a compromise between the number of processes and threads by process.

The keep-alive setting keeps the connection during a number of seconds between the web server and the browser. This interval avoids network negotiation for each new request between them. Keep in mind that, during this time, those threads cannot answer other requests.

For more information, see IBM HTTP Server Performance Tuning at:

`http://publib.boulder.ibm.com/httpserv/ihsdiag/ihs_performance.html`

DMZ Secure Proxy Server

The DMZ Secure Proxy Server is a possible replacement of the web server with the plug-in. The same tuning considerations apply for the DMZ secure proxy as they do for the web server with the plug-in loaded.

For the DMZ secure proxy, you must consider two additional main tuning areas:

- JVM tuning

  When tuning the JVM of the DMZ secure proxy, the same rules apply as for the application server JVM. For more information about JVM tuning, see “Application server and Java virtual machine” on page 300.

- Proxy tuning

  The proxy server also provides specific tuning capabilities. Review the following settings closely:

  - Proxy thread pool size
  - HTTP proxy server settings
    - Proxy settings (such as timeouts and connection pooling)
    - Routing rules
    - Static cache rules
    - Rewriting rules
    - Proxy server transports (persistent connections and pools size)
    - Proxy cache instance configuration
    - Denial of service protection

Application server and Java virtual machine

The most important aspects of tuning the JVM are to choose the correct garbage policy and to define the minimum and maximum heap sizes. You must define these parameters based on application behavior. A JVM that spends more than 10% of the time in garbage collection is not efficient and needs to be tuned. Time lost to free memory is time that is not spent to process application server requests.

For more information about the garbage collection policies and `-Xgcpolicy` option, see the Websphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=tprf_tunejvm_v61`

Starting a JVM with too little memory means that the application must immediately switch context to allocate memory resources. This switching can slow down server startup and the execution of the application until it reaches the heap size it needs to run. Conversely, a JVM with a size that is too large does not run garbage collection often enough. This can leave the system littered with unused objects and a fragmented heap that requires compacting later.

Adjust the levels during the testing phase to determine reasonable levels for both settings. In addition, the prepared statement cache and dynamic fragment caching also consume portions of the heap. You might be required to make additional adjustments to the heap when those values are adjusted.

For more information about tuning the JVM, see the Websphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=container_tune_jvm`

Thread pools

Inside the application server JVM, separate thread pools are used to manage different types of workload. Depending on your type of application and workload, define thread pool sizes carefully, as explained in this section.

Web container thread pools

Monitor the web container thread pool closely during initial performance runs. The web container thread pool is the most common bottleneck in an application environment. If you adjust the number of threads too low, the web server threads end up waiting for the web container. If you adjust the number of threads too high, the server can be inundated with too many requests. Both situations increase the response time.

Consider the following aspects when defining the web container size:

- The entire infrastructure chain in close cooperation with the web server
- The number of threads and the number of sessions in the database

Enterprise JavaBeans container thread pools

The Enterprise JavaBeans (EJB) container can be another source of potential scalability bottlenecks. The inactive pool cleanup interval is a setting that determines how often unused EJB are cleaned from memory. If the setting is set too low, the application can spend time instantiating a new EJB when an existing instance can be reused. If the setting is set too high, the application can have a larger memory heap footprint with unused objects remaining in memory. EJB container cache settings can also create performance issues if they are not properly tuned for the system.

Message listener thread pools

For JVMs that host MDBs, you can check and configure the message listener thread pool.

Mediation thread pools

If you want to run multiple mediations in your bus infrastructure concurrently, configure a mediation thread pool by using the **wsadmin** command-line interface (CLI).

Connection pools

Connection pools are used when the application needs access to a back-end tier (such as a database). For each connection pool, you can configure the number of connections, including the timeout connection and few other connection parameters.

Database connection pools

The database connection pool is another common location for bottlenecks, especially in data-driven applications. The default pool size is 10. Depending on the nature of the application and the number of requests, the default setting might not be sufficient. During performance runs, pay special attention to the pool usage, and adjust the pool size accordingly.

Connection factories connection pools

Applications use connection pools, such as connection factories, queue connections factories, and topic connections factories, to connect to JMS destinations. These resources present other potential bottlenecks that you need to monitor during performance runs.

Web services connection pools

Use HTTP transport properties for Java API for XML Web Services (JAX-WS) and Java API for XML-based RPC (JAX-RPC) web services. These functions manage connection pools for HTTP outbound connections. Configure the content encoding of the HTTP message, enable HTTP persistent connection, and resend the HTTP request when a timeout occurs.

Service integration bus

A service integration bus uses several pools and message threshold parameters, which you need to configure properly. Each bus has a high messages threshold that limits the number of messages that are currently processing. By default, this threshold is set to 50,000. You can adjust this parameter, if needed. JVM tuning is also possible for JVMs that host the bus. In addition, you can configure access to the message engine store and the storage itself for better performance.

For each connection resource as a connection factory, queue factory, or topic connection factory, you can configure the persistence or nonpersistence of the messages. An increase of the quality of service brings a decrease in performance because you must store and manage persistent messages.

You can also set the number of concurrent MDBs and the number of messages that are processed by MDB instances.

Large pages

If your platform can use a larger memory page size than the default of 4 KB, consider configuring the larger memory page size. JVM supports large pages, and Java applications benefit by using the large pages because they use less processor capacity.

10.3.10 System tuning

Bottlenecks also occur at the system level. To prevent these bottlenecks, tune your storage, network, and operating system adequately. The following aspects can potentially affect the performance:

- Storage

  If your applications run much I/O, directly by using read/write instructions or indirectly by using the database, check the storage box response times. Several storage improvements take place at the operating system, network, or storage level:

  - Increase the queue depth of disks
  - Adjust the number of possible paths to reach a disk
  - Reorganize data on the storage box
  - Use high performance disks

  A high performance disk example is a solid-state drive (SSD) and Fibre Channels.

- Network

  First check the throughput and the latency between your servers and network devices. Take the time to verify that port settings on the switches match the settings of the network interfaces. Many times, a network device is set to a specific speed, and the network interface is set to auto-detect.

  You can improve network performance by using more powerful links or Ethernet channel if your throughput is bounded. Check the operating system network parameters, especially the buffers. High-end servers with several partitions inside provide internal networks to improve the latency and throughput.

- Operating system

  Memory and processor can affect performance. You can configure several parameters to improve performance in this area. When the system is memory or processor bounded and the application stack is tuned, the solution might be to add resources.

  All tuning at the system layer must be defined in close cooperation with the infrastructure team. Changes at this level can affect applications and the entire environment.

  For more information about tuning the operating system for WebSphere Application Server, see the Websphere Application Server V8.5 Information Center at:

  http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-express-dist&topic=tprf_tuneopsys

10.4 WebSphere Application Server performance tools

When identifying bottlenecks or reviewing the application environment, you often need to go beyond the operating system layer and deeper into the behavior of the application. For example, you might need to determine the memory footprint of the application or analyze the threads that are used by the application. This type of evaluation requires the use of specialized tools to capture information.

WebSphere Application Server provides tools for the administrator to gather information related to the performance of various components in the Java 2 Platform, Enterprise Edition (J2EE) environment:

- IBM Support Assistant Data Collector
- IBM Monitoring and Diagnostic tools for Java
- IBM Tivoli Performance Viewer

- WebSphere Application Server performance advisors
- Request metrics in WebSphere Application Server
- WebSphere Performance Monitoring Infrastructure

10.4.1 WebSphere Performance Monitoring Infrastructure

WebSphere Performance Monitoring Infrastructure is the core monitoring component for WebSphere Application Server. WebSphere Performance Monitoring Infrastructure complies with the Performance Data Framework as defined in the J2EE 1.4 standard.

By collecting runtime application server and system data, this component provides interfaces that allow external applications to monitor the performance data. Performance Monitoring Infrastructure data can be gathered in different ways, such as by using Java Management Extensions (JMX) with MBeans or the Performance Servlet. With these two interfaces, you can develop your own monitoring applications. WebSphere Application Server also provides an integrated graphical monitoring tool, Tivoli Performance Viewer, to capture data from Performance Monitoring Infrastructure.

By using these interfaces, you can capture information about the following resources:

- Application resources
  - Applications counters
  - Custom Performance Monitoring Infrastructure
  - Enterprise bean counters
  - J2C connections counters
  - Java Database Connectivity (JDBC) connections counters
  - Servlets or JavaServer Pages (JSP) counters
  - Session Initiation Protocol (SIP) counters
  - Web services counters
- System resources
  - Total free memory
  - Processor usage
  - Components that are controlled outside the WebSphere environment, but that are vital and in a healthy application state
- WebSphere runtime resources
  - Database connection pools
  - Dynamic caching
  - JVM memory
  - Object Request Broker (ORB) counters
  - Proxy counters
  - Session persistence
  - Thread pools
  - Transactional counters
  - Workload management counters

Important: Performance Monitoring Infrastructure offers the custom Performance Monitoring Infrastructure API. With this interface, you can insert custom metrics and have them captured and available to the standard monitoring tools.

When determining the metrics to capture, you can select from the following monitoring statistics sets:

- All
- Basic (enabled by default)
  - Processor usage
  - HTTP session information
  - J2EE components
- Custom (select your own mix of metrics)
- Extended (basic +)
  - Dynamic cache
  - Workload Manager (WLM)

System monitoring effects: Monitoring a system changes the nature of the system. Introducing performance metrics consumes more resources. Thus, the more statistics that you capture, the more processing power is required.

Java Virtual Machine Tool Interface

The Java Virtual Machine Tool Interface (JVMTI) is a native programming interface that provides tools to inspect the state of the JVM. With JVMTI, you can collect the garbage collection and thread-state information of a JVM. The statistics that are gathered through the JVMTI are different than ones gathered by the JVM provided by IBM. The JVMTI statistics are also different from those gathered by the Sun HotSpot technology-based JVM. Sun HotSpot JVMs include Sun HotSpot JVM on Solaris and the Hewlett-Packard JVM for HP-UX.

Enabling the JVMTI involves enabling the JVM profiler for the application server and selecting the appropriate metrics by using the custom settings.

IBM Tivoli Composite Application Manager for WebSphere Application Server counters

WebSphere Application Server V8.5 offers an optional enhancement to Performance Monitoring Infrastructure, a web resources data collector called *eCAM*. eCAM is a separate data collector that is boot strapped at startup of the application server. It monitors class loads and instruments at the web and EJB container-level only. This data collector allows gathering of request-oriented data, elapsed time, processor data, and counters.

The data that the eCAM data collector gathers is exposed by using an MBean that is registered in Performance Monitoring Infrastructure. You can view the collected performance data through the standard Tivoli Performance Viewer of WebSphere Application Server.

The following counters that are collected by eCAM are exposed through Tivoli Performance Viewer:

- RequestCount
- AverageResponseTime
- MaximumResponseTime
- MinimumResponseTime
- LastMinuteAverageResponseTime
- 90%AverageResponseTime
- AverageCPUUsage
- MaximumCPUUsage
- MinimumCPUUsage
- LastMinuteAverageCPUUsage
- 90%AverageCPUUsage

For details about the data collected by eCAM, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=rprf_tpmcounter

10.4.2 IBM Tivoli Performance Viewer

Tivoli Performance Viewer is included with WebSphere Application Server V8.5, and is used to record and display performance data. Using Tivoli Performance Viewer, you can perform the following tasks:

- Display the following Performance Monitoring Infrastructure data that is collected from local and remote application servers:
  - Summary reports showing key areas of contention
  - Graphical or tabular views of raw Performance Monitoring Infrastructure data
- Provide configuration advice through the performance advisor section and tuning advice that is formulated from Performance Monitoring Infrastructure and your configuration data.
- Use Tivoli Performance Viewer to log real-time performance data so you can review it at a later time.
- View server performance logs. You can record and view data that was logged by Tivoli Performance Viewer by using the administrative console.

To avoid gathering too much information with Tivoli Performance Viewer, you can select the specific performance modules that you want to monitor. You can also use the log analysis tools to detect trends over time. Tivoli Performance Viewer can save performance data for later analysis or problem determination.

Because Tivoli Performance Viewer runs inside the administrative console, the performance impact depends on which edition of WebSphere Application Server is run. When running the single server edition, Tivoli Performance Viewer runs in the same JVM as your application. In the Network Deployment edition, Tivoli Performance Viewer runs in the JVM of the deployment manager. However, certain functions (such as the advisor) require resources in the node agents or in the application servers.

10.4.3 WebSphere Application Server performance advisors

After you gather runtime information, performance advisors for WebSphere Application Server can determine diagnostic advice about the environment. The advisors can determine the current configuration of an application server. Also, by trending the runtime data over time, they can determine potential environmental changes that can enhance the performance of the system. Determinations are hard-coded into the system, and are based on IBM preferred practices for tuning and performance.

The advisors do not implement any changes to the environment. Instead, they identify the problem, and the system administrator decides whether to implement the changes. Always perform tests after changes are implemented.

Two types of advisors are available:

- Performance and Diagnostic Advisor
- Performance Advisor in Tivoli Performance Viewer

Performance and Diagnostic Advisor

The Performance and Diagnostic Advisor is configured through the administrative console. It writes log files to the application server and to the console in monitor mode. To minimize the impact of this logging, configure the server to use High Performance Extensible Logging instead of using `SystemOut.log`. The interface can be configured to determine how often data is gathered and advice is generated.

Performance and Diagnostic Advisor offers advice about the following components:

- J2C Connection Manager
  - Thread pools
  - LTC Nesting
  - Serial reuse violation
- Web Container Session Manager
  - Session size with overflow enabled
  - Session size with overflow disabled
  - Persistent session size
- Web Container
  - Bounded thread pool
  - Unbounded thread pool
- ORB Service
  - Unbounded thread pool
  - Bounded thread pool
- Data source
  - Connection pool size
  - Prepared statement cache size
- JVM

  Memory leak detection

If you need to gather advice about items outside of this list, use the Performance Advisor in Tivoli Performance Viewer.

Performance Advisor in Tivoli Performance Viewer

The Performance Advisor in Tivoli Performance Viewer is slightly different from the Performance and Diagnostic Advisor. The Performance Advisor in Tivoli Performance Viewer is started only through the Tivoli Performance Viewer interface of the administrative console. It runs on the application server that you are monitoring, but the refresh intervals are based on the refresh option selected through the console. Also, the output is routed to the user interface instead of an application server output log. In addition, this advisor captures data and provides advice about more components.

Specifically, the Performance Advisor in Tivoli Performance Viewer can capture the following types of information:

- ORB service thread pools
- Web container thread pools
- Connection pool size
- Persisted session size and time
- Prepared statement cache size
- Session cache size
- Dynamic cache size

- JVM heap size
- DB2 performance configuration

Running the Performance Advisor in Tivoli Performance Viewer requires resources and can affect performance. Use this advisor with care in production environments.

10.4.4 Request metrics in WebSphere Application Server

Performance Monitoring Infrastructure for WebSphere Application Server provides information about average system resource usage statistics. However, it does not provide any correlation between the data. Request metrics, in contrast, provide data about each individual transaction and correlate this data.

Request metrics gather information about single transactions within an application. The metrics track each step of a transaction and determine the process time for each of the major application components.

The following components support this transaction metric:

- Web server plug-ins
- Web container
- EJB container
- JDBC calls
- Web services engine
- Default messaging provider

The amount of time that a request spends in each component is measured and aggregated to define the complete execution time for that transaction. Both the individual component times and the overall transaction time can be useful metrics when trying to gauge user experience on a site. The data allows for a hierarchical view by response time for each individual transaction. When debugging resource constraints, these metrics provide critical data at each component. The request metric provides filtering mechanisms to monitor synthetic transactions or to track the performance of a specific transaction. By using test transactions, you can measure performance of the site from end to end.

From a performance perspective, using transaction request metrics can aid in determining whether an application is meeting service level agreements (SLAs) for the client. The metrics can be used to alert the user when an SLA target is not met.

Request metrics help administrators answer the following questions:

- Which performance areas need focus?
- Is too much time spent on any specific area?
- Do response times for transactions meet goals so they do not violate the SLAs?

Several methods are available for implementing request metrics. This section briefly explains the methods that are currently available.

Request filtering

The most common method of implementing request metrics is to use request filtering. In this method, you use filters to limit the number of transactions that are logged. It captures only those transactions that you want to monitor. For example, you can use an IP address filter to monitor synthetic transactions from a single server.

The following filters are available:

- HTTP requests: Filtered by IP address, Uniform Resource Identifier (URI), or both
- Enterprise bean requests: Filtered by method name
- JMS requests: Filtered by parameters
- Web services requests: Filtered by parameters
- Source IP filters

The performance impact is less than 5% when all incoming transactions are being instrumented.

Tracing

By setting the trace depth, you can control the amount of information gathered through the metric and the overall performance impact on the system. The higher a tracing level is set, the greater the performance penalty the system takes.

The following trace levels are available:

- None: No data captured
- Hops: Process boundaries (web server, servlet, EJB over RMI-IIOP)
- Performance_debug: Hops + 1 level of intraprocess calls
- Debug: Full capture (all cross-process/intraprocess calls)

Output for request metrics

The data that is captured by request metrics is placed in several levels, depending on the nature of the metric that is selected:

- For web requests, the HTTP request is logged to the output file that is specified in the `plugin-cfg.xml` file on the web server.
- For application server layers, servlets, web services, EJB, JDBC, and JMS, the information is logged to the application server log files.

To minimize the writing impact, configure the server to use High Performance Extensible Logging instead of using the `SystemOut.log` file. The data can also be output to an Application Response Measurement (ARM) agent. It can be visualized by using an ARM management software, such as IBM Tivoli Monitoring for Transaction Performance or IBM Enterprise Workload Management.

If you currently use a third-party tool that is ARM 4.0 compliant, the data can be read by that agent as well. You can access data from the logs, the agent, or both at the same time.

Important: Do not use metric logging when implementing the ARM agent monitoring because the disk I/O can negatively affect performance.

Application Response Measurement

ARM is an Open Group standard that defines the specification and APIs for per-transaction performance monitoring. Request metrics can be configured to use ARM, by using the ARM API to gather data.

For more information about ARM, see:

`http://www.opengroup.org/tech/management/arm/`

WebSphere Application Server does not provide an ARM agent, but supports the use of an ARM 4.0 or ARM 2.0 agent.

10.4.5 IBM Monitoring and Diagnostic tools for Java

IBM also provides IBM Monitoring and Diagnostic tools for Java. By using IBM Support Assistant, a workbench offering, and a single point to access to these tools, you can analyze these objects:

- Applications
- Garbage collection logs
- Java heap memory dumps
- Java cores

For more information about IBM Monitoring and Diagnostic tools for Java, see:

http://www.ibm.com/developerworks/java/jdk/tools/

Health Center

With Health Center, you can monitor real-time running applications. Health Center provides useful information about memory, class loading, I/O, object allocations, and the system. This tool can help you to identify application memory leaks, I/O bottlenecks, and lock contentions. It also helps to tune the garbage collector. Health Center minimizes the performance impact of monitoring.

Memory Analyzer

The Memory Analyzer tool analyzes the Java heap of a JVM process, identifies potential memory leaks, and provides the application memory footprint. Memory Analyzer provides an object tree that you can use to focus on object interactions and to analyze memory usage.

Dump Analyzer

Dump Analyzer determines the causes of Java crashes by analyzing the operating system memory dumps. This tool can be useful in helping you to better understand application failures.

Garbage Collection and Memory Visualizer

Garbage Collection and Memory Visualizer helps you to analyze and tune the garbage collection. It also provides recommendations to optimize the garbage collector and to find the best Java heap settings. With Garbage Collection and Memory Visualizer, you can browse garbage collection cycles and better understand the memory behavior of an application.

10.4.6 IBM Support Assistant Data Collector

The IBM Support Assistant Data Collector for WebSphere Application Server V8.5 is a tool that can be run to gather data from the application server system for problem determination purposes. This tool focuses on automatic collection of problem data. It also provides symptom analysis support for the various categories of problems encountered by IBM software products. Information pertinent to a type of problem is collected to help identify the origin of the problem. The tool assists customers by reducing the amount of time it takes to reproduce a problem with the correct RAS tracing levels set. It also reduces the effort required to send the appropriate log information to IBM Support.

Remember: The collector tool of WebSphere Application Server is deprecated. It is replaced by IBM Support Assistant Data Collector in WebSphere Application Server V8.5.

10.4.7 IBM HTTP Server monitoring page

To monitor IBM HTTP Server, a web page called *server-status* is available. This page is disabled by default, but you can enable it in the `httpd.conf` configuration file of IBM HTTP Server. This web page shows a real-time view of the current IBM HTTP Server state.

You can visualize the following information:

- Processor usage
- The total number of requests served for the total time the server is up
- The total traffic size for the total time the server is up
- Average response time
- The number of requests currently running
- The number of idle threads
- List of requests that are being processed

10.5 Workload management

Workload management is the concept of sharing requests across multiple instances of a resource. Workload management is an important technique for high availability, performance, and scalability. Workload management techniques are implemented expressly for providing scalability and availability within a system. These techniques allow the system to serve more concurrent requests.

Workload management provides the following main features:

- *Load balancing* is the ability to send requests to alternative instances of a resource. Workload management allows for better use of resources by distributing loads more evenly. Components that are overloaded, and therefore, a potential bottleneck, can be routed around with workload management algorithms. Workload management techniques also provide higher resiliency by routing requests around failed components to duplicate copies of that resource.
- *Affinity* is the ability to route concurrent requests to the same component that served the first request.

10.5.1 HTTP servers

An IP sprayer component is used to run the load balancing and workload management function for incoming web traffic (Figure 10-2). The IP sprayer component can be the Edge Component Load Balancer or a network appliance.

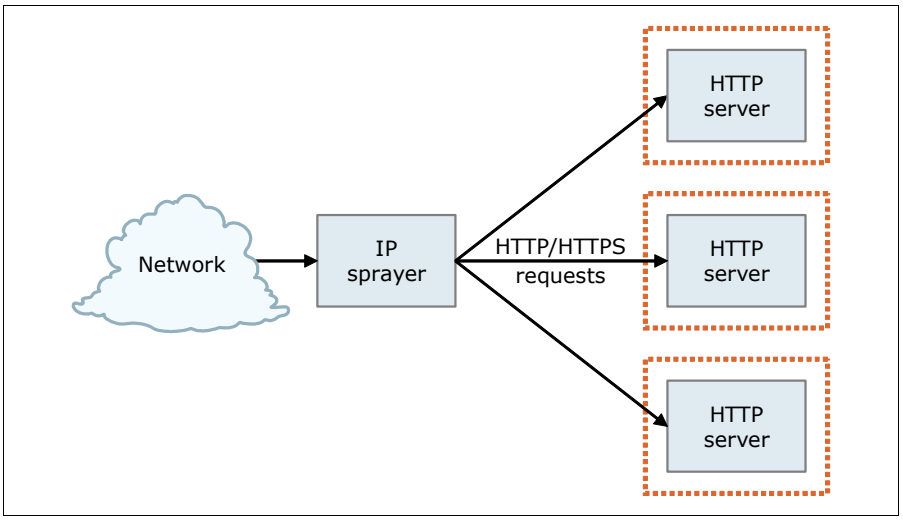


Figure 10-2 IP Sprayer or HTTP server workload management

Depending on which solution you implement, you have the following routing options:

- Dynamic weight, where the load balancer calculates the load of each HTTP server and routes requests dynamically to the one that is less busy
- Static weight, where each member has a weight and the load balancer spreads the requests based on this weight

The following affinity rules overwrite the routing options:

- Stickiness to source IP address

  You can configure the cluster member port to be sticky. This option allows client requests to be directly routed to the same server. A sticky time is also set to define the timeout of this association.

- Cookie affinity

  Based on the content of a cookie, the load balancer can route to the same server.

- URI affinity

  To improve the web server cache efficiently, you can use the URI affinity policy. The load balancer forwards the incoming requests with the same URI to the same web server.

- SSL session ID

  When SSL is enabled between the browser and the web server, you can avoid multiple SSL handshakes. You can do so by routing the HTTPS for the same client to the same server. To process, the load balancer needs an SSL session ID.

In addition, the WebSphere plug-in provides workload management capabilities for applications that are running in an application server.

10.5.2 DMZ proxy servers

As with HTTP servers, you can use an IP sprayer component to run load balancing and workload management for incoming web traffic. In addition, the DMZ proxy server provides workload management capabilities for applications that are running in an application server.

10.5.3 Application servers

In WebSphere Application Server, workload management is achieved by sharing requests across one or more application servers, each running a copy of the application. In more complex topologies, workload management is embedded in load balancing technologies that can be used in front of web servers.

Workload management is a WebSphere Application Server facility to provide load balancing and affinity between nodes in a clustered environment. Workload management can be an important facet of performance. WebSphere Application Server uses workload management to send requests to alternative members of the cluster. WebSphereApplication Server routes concurrent requests from a user to the same application server to maintain session state.

WLM for WebSphere Application Server for z/OS works differently from the WLM for distributed platforms. The workload management structure for incoming requests is handled by the WLM subsystem features of z/OS. Organizations can define business-oriented rules that are used to classify incoming requests, and assign SLA types of performance goals. This definition is done on transaction-level granularity rather than the server-level granularity used with distributed workload management. The system then assigns resources automatically in terms of processor, memory, and I/O to try to achieve these goals.

In addition to the response times, the system can start additional processes, called *address spaces*. Address spaces run the user application if performance bottlenecks occur due to an unpredictable workload spike.

The explanation provided in this section is an over-simplification of how workload management works in z/OS. For more information about workload management of z/OS and WebSphere Application Server for z/OS, see 16.1.7, “Workload management for WebSphere Application Server for z/OS” on page 509. You can also visit the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=crunwlmzos

10.5.4 Clustering application servers

Clustering application servers that host web containers automatically enable plug-in workload management for the application servers and the servlets that they host. Routing of servlet requests occurs between the web server plug-in and the clustered application servers by using HTTP or HTTPS (Figure 10-3).

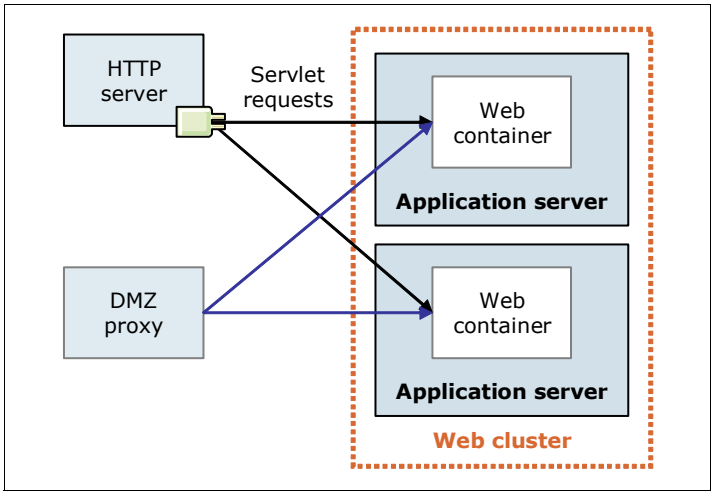


Figure 10-3 Plug-in (web container) workload management

WebSphere Application Server provides the following load balancing options:

- Round-robin

  This routing option is based on the weight that is associated with cluster members. If all cluster members have identical weights, the plug-in sends equal requests to all members of the cluster, assuming no strong affinity configurations. If the weights are scaled in the range 0–20, the plug-in routes requests more often to those cluster members with the higher weight value. No requests are sent to cluster members with a weight of 0 unless no other servers are available. Round-robin is the default load balance policy.

  Use the following guideline formula to determine the routing preference, where *n* cluster members are in a cluster:

  `% routed to Server1 = weight1 / (weight1+weight2+...+weightn)`

- Random

  With this option, a member of the cluster is picked randomly by the plug-in.

The load balancing options are impacted by session affinity. After a session is created at the first request, all the subsequent requests must be served by the same member of the cluster. The plug-in retrieves the application server that serves the previous request by analyzing the session identifier and trying to route to this server. For more information about these sessions, see 10.8, “Session management” on page 330.

On the z/OS platform, the assignment of transactions to cluster members is run on real-time achievement of defined performance goals. This assignment process allows the system to differentiate between light requests that use only a small fragment of performance and heavy requests.

For more information about web server plug-in workload management in a cluster, see:

`http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=plugin+workload+management&uid=swg21219567&loc=en_US&cs=utf-8&lang=en`

Workload management for EJB containers can be performed by configuring the web container and EJB containers on separate application servers. Multiple application servers with EJB containers can be clustered, enabling the distribution of EJB requests between the EJB containers, as illustrated in Figure 10-4.

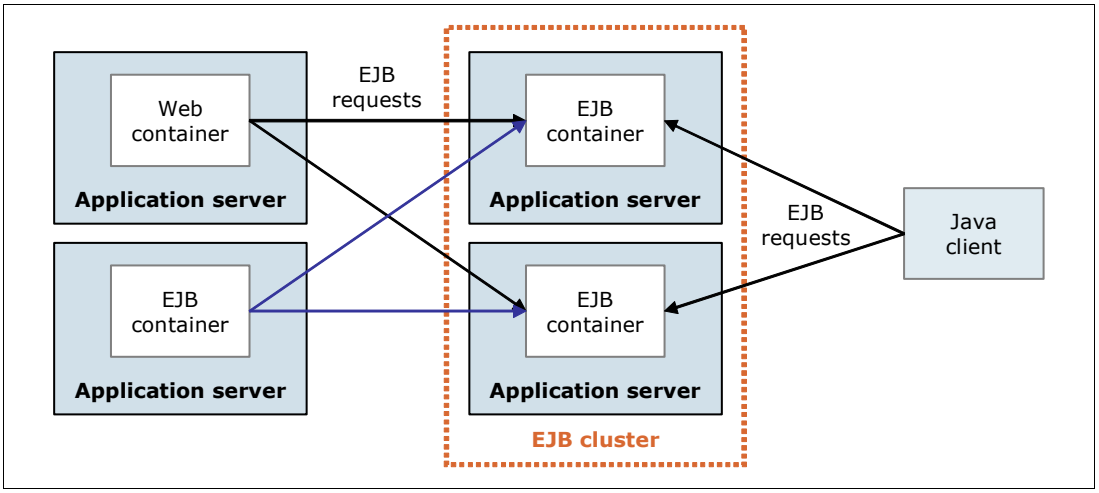


Figure 10-4 EJB workload management

To route the EJB requests, WebSphere Application Server provides the following main routing policies:

- Server weighted round-robin

  In this configuration, EJB client requests are routed to available EJB servers in a round-robin fashion based on assigned server weights. The EJB clients can be servlets that operate within a web container, stand-alone Java programs using RMI/IIOP, or other EJB.

  The server weighted round-robin routing policy ensures a distribution based on the set of server weights assigned to the members of a cluster. For example, if all servers in the cluster have the same weight, all servers should receive the same number of requests. If the weights for the servers are not equal, the distribution mechanism sends more requests to the higher weight value servers.

- Prefer local

  You can also choose to have EJB requests preferably routed to the same host as the host of the requesting EJB client. In this case, only cluster members on that host are chosen by using the round-robin weighted method. Cluster members on a remote host are chosen only if a local server is not available.

The following affinity policies also affect the routing:

- Process affinity

  If an EJB is available in the same cluster member as the client, all requests from that client are directed to the EJB in the same JVM process. One of the advantages of the policy is that there is no need for serialization for method calls.

- Transaction affinity

  All the requests from the same transaction are directed to the same cluster member. This policy overwrites all the other policies.

When planning for clustering, determine the number of application servers and their physical location. Determine the server weights to assign for application servers based on

considerations such as system stability and speed. When creating the cluster, consider using the *prefer local* setting. This setting ensures that, when a client calls an EJB, WLM attempts to select the EJB on the same system as the client. Staying on the same system eliminates network communication.

10.5.5 Dynamic clusters

WebSphere Application Server V8.5 provides integrated support for dynamic clusters. A dynamic cluster is an application deployment target that can expand and contract based on the workload in the environment. Dynamic clusters work with autonomic managers, including the application placement controller and the dynamic workload manager, to maximize the use of computing resources. A dynamic cluster uses weights and workload management. This management is used to balance the workloads of its cluster members dynamically, and is based on performance information that is collected from the cluster members. Dynamic clusters enable application server virtualization.

WebSphere Application Server V8.5 supports complete lifecycle management servers. In this mode, the product controls the creation and deletion of server instances, and can start and stop servers. By automatically defining cluster members with rules, you can create a subexpression that automatically selects nodes to host dynamic cluster members based on node properties. This subexpression is called a membership policy. After you create the membership policy, you can preview the node membership before you finish creating the dynamic cluster. After you create the dynamic cluster with a membership policy, dynamic cluster instances can start on any of the selected nodes.

WebSphere Application Server V8.5 also supports assisted lifecycle management servers. In this mode, the product can control the state of servers by stopping and starting servers from a pool of predefined server instances. When you manually define cluster members, you statically define which servers are cluster members by selecting servers to add to the cluster. Use this option instead of the membership policy if you have an existing static cluster that you want to convert to a dynamic cluster.

Operating modes

Dynamic clusters behave differently depending on the operating mode. The following modes of operation are available:

- Manual

  In manual mode, the dynamic cluster is no different from the standard application server environments with static clusters. Manual mode does not support application placement, or runtime task suggestions. The autonomic request flow manager and dynamic workload management (DWLM) can work with the cluster.

- Supervised

  In supervised mode, the environment provides information about required corrective actions by generating runtime tasks. You can accept or deny the recommendations of the autonomic managers.

- Automatic

  In automatic mode, the environment takes corrective actions automatically.

For more information about dynamic clusters, see Chapter 5, “Intelligent Management” on page 107.

10.5.6 Dynamic application placement

WebSphere Application Server V8.5 Intelligent Management feature provides a dynamic application placement capability. This capability is based on load distribution, service policy, and available resources. Dynamic application placement can use hardware more efficiently. It is unlikely that all applications are in high demand at the same time, assuming a varied assortment of deployed applications. Intelligent Management manages this situation by supporting resource allocation where needed, increasing the utilization of hardware. The enterprise no longer requires enough hardware to satisfy each application maximum load simultaneously. This can translate to a significant reduction of IT purchases.

For more information about dynamic application placement, see Chapter 5, “Intelligent Management” on page 107.

For more information about configuring dynamic application placement, see:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=twve_odapp

10.5.7 On-demand router

The on-demand router is a server that acts as an HTTP proxy or a SIP proxy. An on-demand router contains the autonomic request flow manager (ARFM). ARFM prioritizes inbound traffic according to service policy configuration and protects downstream servers from being overloaded. Traffic is managed to achieve the best balanced performance results, considering the configured service policies and the offered load.

The on-demand configuration component allows the on-demand router to sense its environment. On-demand configuration dynamically configures the routing rules at run time to support the on-demand router in accurately routing traffic to the application servers. An on-demand router can route HTTP requests to Intelligent Management servers, WebSphere Application Server Network Deployment servers, and servers that are not running WebSphere software. This router, like the web server plug-in for WebSphere Application Server, uses session affinity for routing work requests. After a session is established on a server, later work requests for the same session go to the original server. This system maximizes cache usage and reduces queries to back-end resources.

For more information about the on-demand router, see 5.3, “Intelligent routing and dynamic operations” on page 116.

10.5.8 Dynamic workload management

Dynamic workload management is a feature of the on-demand router. It applies the same principles as WLM such as routing based on a weight system, which establishes a prioritized routing system. With dynamic workload management, the system can dynamically modify the weights to stay current with the business goals. It also balances requests across the available nodes to regulate response times.

For more information about dynamic workload management, see 5.4, “Dynamic workload management” on page 121.

10.5.9 Scheduling tasks

WebSphere Application Server provides a scheduler service that can schedule actions to happen with the following frequencies:

- Only once
- Some time in the future
- On a recurring basis
- At regular intervals

The scheduler service can also receive notifications about task activity. Scheduler tasks can be stored in a relational database and be run for indefinite repetitions and long time periods. Scheduler tasks can be tasks based on EJB, or they can be triggered by using JMS.

The scheduler service can be a tool in workload management by scheduling maintenance tasks such as backups, cleanups, or batch processing during off-peak hours. When a task runs, the tool is run in the work manager that is associated with the scheduler instance. You can control the number of actively running tasks at a time by configuring schedulers with a specific work manager. The number of tasks that can run concurrently is set by the number of alarm threads parameter on the work manager.

10.6 High availability

High availability is also known as *resiliency*. High availability is the ability of a system to tolerate a number of failures and remain operational. This section provides several considerations for high availability.

10.6.1 Overview

High availability means that your infrastructure continues to respond to client requests regardless of the circumstances. Depending on the errors or failures, the infrastructure can run in a degraded mode. High availability is achieved by adding redundancy in the infrastructure to support failures. Availability affects both performance and scalability.

Depending on your needs, you must define the level of high availability of the infrastructure. The most common method of describing availability is by the “nines,” or the percentage of availability for the system. For example, 99.9% system availability represents 8.76 hours of outage in a single year.

Table 10-1 shows the level of availability and the calculated downtime per year.

Table 10-1 Availability matrix

Availability percentage	Downtime per year
99% (two 9s)	87.6 hours
99.9% (three 9s)	8.76 hours
99.99% (four 9s)	56.56 minutes
99.999% (five 9s)	315.36 seconds

Calculating availability by using the following formula:

`Availability = (MTBF/(MTBF + MTTR)) X 100`

where:

MTBF Is the mean time between failure
MTTR Is the maximum time to recovery

Keep in mind that the overall infrastructure is available only if all the components are available. A WebSphere Application Server infrastructure consists of several components such as load balancers, HTTP servers, application servers, and database servers. Availability is determined by the weakest component.

For most of the environment components, several degrees of high availability exist in an implementation. The cost of the infrastructure is directly linked to the level of availability. Evaluate the business loss of the infrastructure downtime, and ensure that the business case justifies the costs. Moving system availability from 99.9% to 99.99% can be expensive. It can also be true that the system is used only during regular business hours on regular working days. This assumption implies that an availability of 99.9% is more than adequate to meet the operational window.

Important: Availability features can have an impact on the cost of the solution. Be sure to evaluate this increment in the implementation cost against the cost of not having the application available.

For more information, see the following IBM developerWorks topic:

http://www.ibm.com/developerworks/websphere/techjournal/0312_polozoff/polozoff.html#sec1

In many facilities, the complete environment is made up of multiple systems. The goal is to make the entire system as available as possible. You do so by minimizing the number of SPOF throughout the system and by adding redundancy. Redundancy can be added at different layers, such as hardware, process, and data.

10.6.2 Hardware high availability

Although modern hardware is reliable and many components are fault tolerant, hardware can fail. Any mechanical component has an expected failure rate and a projected useful life until failure. Depending on the hardware, you have several high availability solutions. This section highlights a few ideas to improve hardware high availability.

At the server level, you can configure servers with duplicate components. For example, to mitigate power failures, you can have dual power supplies. With a dual power supply configuration, you can further mitigate power failures by plugging each power supply into separate circuits in the data center.

You can also configure multiple network interface cards (NICs) in adapter teaming. This configuration is done so a server can bind one IP address to more than one adapter and then provide failover facilities for the adapter. This configuration can be extended by plugging each adapter into separate switches to mitigate the failure of a switch within the network infrastructure.

At the storage level, you can use operating system disk mirroring with different internal disks or use external storage with multiple paths to access the data. Different options of Redundant

Array of Independent Disks (RAID) are available, depending on the needs. External storage also allows you to duplicate data in different locations.

Network hardware availability can be addressed by most major vendors. Now built-in support is available for stateful failover of firewalls, trunking of switches, and failover of routers. These devices also support duplicate power supplies, multiple controllers, and management devices.

10.6.3 Process high availability

Typically, process high availability is achieved by duplicating processes in a cluster or independently in one or more servers. Keep in mind that you must be able to share and manage the load between these processes.

In WebSphere Application Server, the concept of a singleton process is used. Although not a new concept in WebSphere Application Server V8.5, it is important to understand what this type of process represents in the environment.

A *singleton process* is an executing function that can exist in only one location at any time. In any system, singleton processes are likely to be key components of system functionality.

WebSphere Application Server uses a high availability manager to provide availability for singleton processes. For more information, see 10.6.7, “WebSphere Application Server high availability features” on page 321.

10.6.4 Data availability

In a WebSphere Application Server environment, data availability is important in multiple places. The following are the critical areas for data availability:

- Databases
- EJB session state
- EJB persistence
- HTTP session state

Most of these requirements can be satisfied by using facilities that are in WebSphere Application Server. These areas are explained in more detail in this section.

Database server availability

For many systems, a database server is the largest and most critical SPOF in the environment. Depending on the nature of this data, you can employ many techniques to provide availability for this data:

- If the data is read/write and there is no prevalence of read-only access, consider a hardware or a software clustering solution for the database node. Both require external shared disks through storage area network (SAN), network-attached storage (NAS), or other facilities to provide the exact same disks to different systems. For an active/passive solution, when a failure occurs, the disks are mounted on the standby node, and the database is restarted. In the active/active solution, all the nodes are active at the same time. When a failure occurs, the other members share the additional load.
- For read-only data, multiple copies of the database can be placed behind a load balancing device that uses a virtual IP. With this configuration, the application can connect to one copy of the data and fail over transparently to another working copy.

- If the data is mostly read-only, consider using replication facilities to keep multiple copies synchronized behind a virtual IP. Most commercial database management systems offer some form of replication facility to keep copies of a database synchronized.

Session data

WebSphere Application Server provides the following options for persisting the session data:

- Using memory-to-memory replication to create a copy of the session data in one or more additional servers (several options are available)
- Storing the session data in an external database

The choice of which option to use is up to you, and performance results can vary. External database persistence survives node failures and application server restarts, but it introduces a new SPOF. The SPOF must be mitigated by using an external hardware clustering or high availability solution. Memory-to-memory replication can reduce the effect of failure. Depending on the level of replication, if more than one server fails, the data held on those servers cannot be retrieved on other cluster members.

In contrast to the HTTP session persistence, stateful session EJB availability is handled by using only memory-to-memory replication. Using the EJB container properties, you can specify a replication domain for the EJB container and enable the stateful session bean failover by using memory-to-memory replication. When enabled, all stateful session beans in the container can fail over to another instance of the bean and still maintain the session state.

EJB persistence

When designing applications that use the EJB 2.1 (and later) specifications, the ability to persist these beans becomes available. If the beans participate in a clustered container, bean persistence is available for all members of the cluster. Using access intent policies, you can govern the data access for the persisted bean. This EJB persistence API is not to be confused with entity EJB.

10.6.5 Clustering and failover techniques

Clustering is a fundamental approach for achieving high availability. A *cluster* is a group of several redundant servers that are managed together and that participate in the workload management. *Failover* is the ability to detect the outage of a component automatically and route requests around the failed component.

Hardware-based clustering

On distributed platforms, clustering is deployed in a manner where only one of the servers is actively running system resources. Clustering is achieved by using an external clustering software, such as IBM PowerHA on Power systems. It can also be achieved by using operating system cluster capabilities, such as the Parallel Sysplex on the z/OS platform, to create a cluster of servers.

Each node is generally attached to a shared disk pool through NAS, a SAN, or by chaining SCSI connections to an external disk array. Each system has the base software image installed. The servers stay in constant communication with each other over several connections through the use of *heartbeats*. When a failure occurs, the clustering software switches the resources automatically from the active server to the standby server. The standby server becomes the active server, and the application stack is restarted on it. Configure multiple paths for these heartbeats so that the loss of a switch or network interface does not necessarily cause a failover.

You can use clusters with WebSphere Application Server with hardware-based clustering. Usually, you configure clustering by installing the binary files on both nodes and by creating profiles only one time on shared disks. When a failure occurs on the current node, the profiles are recovered in the standby node, and the processes are restarted.

Software-based clustering

With software-based clustering, you create multiple copies of an application component. All of these copies are available at the same time, both for availability and scalability.

In WebSphere Application Server Network Deployment, application servers can be clustered, which provides both workload management and high availability. WebSphere Application Server V8.5 provides dynamic clusters for dynamic work load management and the on-demand router for intelligently routing requests to nodes. The router is fully aware of the dynamic state of the cell. If one server in the cell fails, the requests are routed to another server. The routers themselves can be clustered to prevent single point of failure.

10.6.6 Maintainability

Maintainability is the ability to keep the system running before, during, and after scheduled maintenance. When considering maintainability in performance and scalability, remember to perform maintenance periodically. Consider maintenance on hardware components, operating systems, and software products in addition to the application components. Maintainability allows for ease of administration within the system by limiting the number of unique features found in duplicated resources. There is a delicate balance between maintainability and performance.

10.6.7 WebSphere Application Server high availability features

This section highlights the WebSphere Application Server features that facilitate high availability. It can help you to understand how the high availability features work and assist you in planning for high availability.

High availability manager

WebSphere Application Server uses a high availability manager to eliminate SPOFs. The high availability manager is responsible for running key services on available application servers rather than on a dedicated server (such as the deployment manager). It continually polls all of the core group members to verify that they are active and healthy.

This manager runs by default in each server, as illustrated in Figure 10-5.

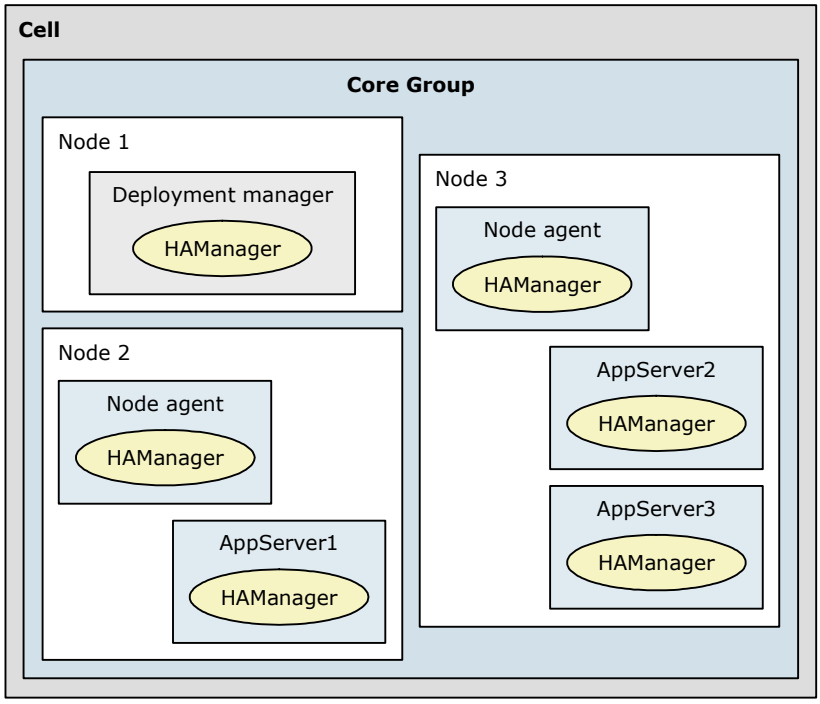


Figure 10-5 Conceptual diagram of a core group

For certain functions (such as transaction peer recovery), high availability manager takes advantage of fault tolerant storage technologies such as NAS. These technologies significantly lower the cost and complexity of high availability configurations. This manager also provides peer-to-peer failover for critical services by maintaining a backup for these services. WebSphere Application Server also supports other high availability solutions such as PowerHA and Parallel Sysplex.

High availability manager continually monitors the application server environment. If an application server component fails, the manager takes over the in-flight and in-doubt work for the failed server. This process introduces some additional processor usage, but significantly improves application server availability.

High availability manager focuses on recovery support and scalability in the following areas:

- Application servers
- Embedded messaging
- Memory-to-memory replication through Data Replication Service (DRS)
- On-demand routing
- Resource adapter management
- Transaction managers
- WebSphere partitioning facility instances
- Workload management controllers

To provide this focused failover service, high availability manager supervises the JVMs of the application servers that are core group members. It uses one of the following methods to detect failures:

- An application server is marked as failed if the socket fails.

  This method uses the KEEP_ALIVE function of TCP/IP. It is tolerant of poor performing application servers, which might happen if the application server is overloaded, swapping,

or thrashing. This method is preferred if you are using multicast emulation and running enough JVMs on a single application server to cause processor or memory starvation.

- A JVM is marked as failed if it stops sending heartbeats for a specified time interval.

  This method is called *active failure detection*. When it is used, a JVM sends out one heartbeat, or pulse, at a specific interval. If the JVM does not respond to heartbeats within a defined time frame, it is considered down.

WebSphere Application Server offers the ability to configure an alternative protocol provider to monitor and manage communication between core group members. In general, alternative protocol providers, such as the z/OS cross-system coupling facility (XCF)-based provider, uses less system resources than the default Discovery Protocol and Failure Detection Protocol. This savings is especially true when the core group members are idle.

In either case, if a JVM fails, the application server on which it is running is separated from the core group. Any services running on that application server are failed over to the surviving core group members.

A JVM can be a node agent, an application server, or a deployment manager. If a JVM fails, any singletons that are running in that JVM are activated on a peer JVM after the failure is detected. This peer JVM is already running, eliminating the normal startup time that potentially can be minutes. This time savings is a key difference to using operating system-based high availability. High availability manager usually recovers in seconds, but operating system-based solutions can take minutes.

When an application server fails, high availability manager assigns the work of the failing application servers to another eligible application server. Using shared storage for common logging facilities (such as the transaction logs) allows the manager to recover in-doubt and in-flight work if a component fails.

Additional resource: A testing routine is available for you to use to determine whether your shared file system is suitable for use with high availability manager. For more information, see IBM File System Locking Protocol Test for WebSphere Application Server at:

`http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=transaction+log+failover&uid=swg24010222&loc=en_US&cs=utf-8&lang=en`

Core group

A *core group* is a high availability domain that consists of a set of processes in the same cell that can directly establish high availability relationships. Highly available components can fail over only to another process in the same core group. Replication can occur only between members of the same core group.

A cell must contain at least one core group, although multiple core groups are supported. Each core group contains a core group coordinator to manage its high availability relationships. This coordinator also manages a set of high availability policies that are used to manage the highly available components within that core group.

WebSphere Application Server provides one standard core group, the DefaultCoreGroup, that is created during installation. New server instances are added to the default core group as they are created.

In most cases, one core group is sufficient for establishing a high availability environment. However, certain topologies require the use of multiple core groups. A basic rule is that all

members of a core group require full IP visibility. Therefore, if you spread the application servers of the cell across different firewall zones, you must create multiple core groups.

Tip: Having many application servers in a cell increases the resource impact of core group services and server start times. Consider creating additional core groups when you have more than 50 servers in a cell.

If you are using a DMZ Secure Proxy Server with dynamic routing, the routing information is exchanged by using core groups. In this case, create a tunnel access point group to establish a core group bridge tunnel between the core groups that run on either side of the firewall.

The core group contains a bridge service that supports cluster services that span multiple core groups. Core groups are connected by access point groups. A core group access point defines a set of bridge interfaces that resolve IP addresses and ports. It is through this set of bridge interfaces that the core group bridge provides access to a core group.

When moving core group members to new core groups, remember the following information:

- Each server process within a cell can be a member of only one core group.
- If a cluster is defined for the cell, all cluster members must belong to the same core group.

Network communication between all members of a core group is essential. The network environment must consist of a fast local area network (LAN) with full *IP visibility* and bidirectional communication between all core group members. IP visibility means that each member is receptive to the communications of any other core group member.

High availability groups

High availability groups are part of the high availability manager framework. A *high availability group* provides the mechanism for building a highly available component and enables the component to run in one of several different processes. A high availability group cannot extend beyond the boundaries of a core group.

A high availability group is associated with a specific component. The members of the group are the set of processes where it is possible to run that component. A product administrator cannot directly configure or define a high availability group and its associated set of members. Instead, high availability groups are created dynamically at the request of the components that need to provide a highly available function.

High availability groups are dynamically created components of a core group. A core group contains one or more high availability groups. However, members of a high availability group can also be members of other high availability groups. All of these high availability groups must be defined within the same core group.

Every high availability group has a policy associated with it. This policy is used to determine which members of a high availability group are active. The policies that the high availability groups use are stored as part of the core group configuration. The same policy can be used by several high availability groups. However, all of the high availability groups to which it applies must be part of the same core group.

Any highly available component for WebSphere Application Server can create a high availability group for its own usage. The component code must specify the attributes that are used to create the name of the high availability group for that component.

For example, establishing a high availability group for the transaction manager is as follows:

- The code included in the transaction manager component code specifies the attribute `type=WAS_TRANSACTIONS` as part of the name of the high availability group that is associated with this component.
- The high availability manager function includes the default Clustered TM Policy that includes `type=WAS_TRANSACTIONS` as part of its match criteria.
- When transaction manager code joins a high availability group, the high availability manager matches the match criteria of the Clustered TM Policy to the high availability group member name. In this example, the name-value pair `type=WAS_TRANSACTIONS` included in the high availability group name is matched to the same string in the policy match criteria for the Clustered TM Policy. This match associates the Clustered TM Policy with the high availability group that was created by the transaction manager component.
- After a policy is established for a high availability group, you can change some of the policy attributes. These attributes include the quorum, fail back, and preferred servers. You cannot change the policy type. If you need to change the policy type, you must create a policy and then use the match criteria to associate it with the appropriate group.

Remember: If you want to use the same match criteria, you must delete the old policy before defining the new policy. You cannot use the same match criteria for two different policies.

Application servers availability

With WebSphere Application Server, you can create clusters for application servers from the same or different nodes. Each member of a cluster must belong to the same cell and cannot belong to more than one cluster. Cluster members are required to have identical application components, but can be sized differently. The cluster is a logical view of the application servers and does not correspond to a process. The workload management is responsible for sharing the workload between the cluster members.

Default messaging provider availability

The bus provides high availability to the messaging system process. By using WebSphere Application Server, you can configure two policies to achieve message engine high availability. These policies are based on the following cluster utilization:

- High availability

  One message engine is created in the cluster and can fail over to any other server in the cluster. The message engine does not fail back to the previous server if this server becomes available again.

- Scalability with high availability

  One message engine is created for each application server on the cluster. Each message engine can fail over to any other server in the cluster.

All the messages set for high availability that were being processed or queued continue to be processed when the message engine is available in another server. Each message engine can fail back to the previous server when this server is available again.

To accomplish a seamless failover, the queue information and message data must be stored in a shared location. This location must be reachable by all the members of the cluster. A shared location can be either an external database or a shared disk environment.

For more information about the availability policy, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=cjt1005_

Restriction: For users who are using the embedded Derby database as a messaging data store, concurrent access can be a concern. The embedded Derby database does not support multiple servers that run the Derby engine. Therefore, you cannot have multiple servers that communicate with the same shared file system.

Resources availability

WebSphere Application Server V8.5 provides the possibility of configuring failover resources for a data source and connection factory. This *resource workload routing* improves the availability of the applications. The data source and connection factory can fail over when a default occurs, and fail back when the situation returns to normal. Only one resource can be used at a time, and the alternate is available only when the primary fails.

To use resource workload routing, you must create alternative resources for data source and connection factory. These resources must be identical to the primaries and be compatible with applications. Then, custom properties can be added to configure the availability behavior.

10.7 Caching

Caching is a facility to offload work to one or more external devices. The application server is no longer required to do all of the work that is associated with user requests. Caching options are available at many different layers in a complete system solution (from the client browser to the data layer). Caching improves performance and scalability.

From the caching point of view, there are two basic types of content:

- Static content

  The static content, such as HTML pages and images, does not change during a long period.

- Dynamic content

  Dynamic content changes repeatedly, such as personalized or custom data, and frequently updated data (exchange rates).

A combination of caching at different layers can improve performance by reusing some previous outputs and by avoiding multiple repeated requests. Be sure that all your cache components have synchronized data to avoid server content that is not up-to-date.

This section provides an overview of the different possibilities for caching within a system. It does not attempt to provide all options, or specific details, because the implementation types of caching are varied.

Important: To use the caching mechanisms that are provided by WebSphere Application Server and other components of your environment, the application must also be designed for caching. Work in close cooperation with the application architect to design your caching components.

This section provides information about the following caching capabilities:

- Edge caching
- Dynamic caching
- Data caching

10.7.1 Edge caching

Edge caching embraces various methods. Numerous software components can provide caching capabilities at the edge of the network:

- Proxy servers, such as the Caching Proxy of the Edge Components (*stabilized* in WebSphere Application Server V8), WebSphere Proxy Server, or the DMZ secure proxy
- Hardware appliances
- External caching proxy providers that can provide content offloading at points closer to the client
- Edge Side Include (ESI) fragment caching capabilities provided by the WebSphere plug-in

WebSphere Application Server can be used to configure and manage how these resources are accessed.

Caching Proxy

Edge Components Caching Proxy provides a caching function to cache both static and dynamic (only on a page level) content. By using Caching Proxy, you can offload additional work from the primary processing environment. Offloading is done by directly serving the response to the client without requesting web server or application server. Implementing this caching adds servers and cost to the solution, but can result in an improvement of the solution performance.

Remember: Caching Proxy is declared a *stabilized* feature. Stabilized means that no new features are delivered, but new platforms are supported.

WebSphere Proxy Server

WebSphere provides a proxy server with the ability to cache both dynamic and static content. This proxy is a part of the WebSphere Application Server cell, and is fully manageable by using the administrative console. WebSphere Proxy Server runs in the secure zone of your infrastructure, and offloads the processing of requests to the rest of the infrastructure. WebSphere Proxy Server can thus improve the performance of the infrastructure.

DMZ Secure Proxy Server

The DMZ Secure Proxy Server runs in a DMZ. You can use it to offload request processing from the core application servers. This proxy server can cache static and dynamic content at the edge of the network. DMZ Secure Proxy Server allows you to configure multiple security levels and routing policies. Depending on the routing policy used, it can dynamically determine the availability of applications on the application servers.

Hardware caching

Multiple network equipment providers offer hardware cache devices. These devices serve the same purpose as software caches do, namely to offload content. The main difference is that these appliances are not running full versions of an operating system. Instead, they use a specialized operating system that is dedicated to running the caching function. This operating system can include custom file systems that offer higher performance than the operating

system file system and a reduced instruction set. By placing dedicated appliances instead of software caching in your architecture, you can reduce your total cost of ownership. This reduction is because these appliances do not have to be managed as strictly as systems with full operating systems.

Caching services

Various providers sell caching as a service. This function can provide even higher performance gains because these providers generally have equipment positioned at Internet peering points throughout the world. Therefore, the user is not required to travel through the Internet to get to the network that serves the application. The providers bring the cached files as close as physically possible to the client.

Edge Side Include

ESI caching is an in-memory caching solution that is implemented through the web server plug-in. The ESI processor can cache pages or fragments of pages at the HTTP server layer.

Each time a new request is received by the plug-in, the ESI processor checks for it in the cache. If some fragments are already in cache, the plug-in can use it. If not, the ESI processor adds a specific header named *Surrogate-Capabilities* before forwarding the request to the application server. The application server then responds to the request. If servlet caching is enabled in the application server and the output is capable of edge cache, the application server adds a Surrogate-Capabilities header with caching information. The plug-in stores the application responses in the cache. The plug-in then builds the page with all the nested components, and returns the answer to the client.

10.7.2 Dynamic caching

WebSphere Application Server, by using the dynamic cache, provides the caching of the output of servlets, JSP, portlets, or web services. *Dynamic caching* is an in-memory cache with the ability to offload the content on disks. If you decide to offload the content, use fast I/O storage.

Dynamic caching is enabled at the container services level of the application server. Objects that can be cached are defined inside the `cachespec.xml` file. This file is stored inside the web module WEB-INF or enterprise bean META-INF directory. The caching options in the `cachespec.xml` file must include sufficient details to allow the dynamic cache service to build a unique cache-key. This cache-key is used to uniquely identify each object, which can be achieved by specifying request parameters, cookies, and so on. With the `cachespec.xml` file, you can define cache invalidation rules and policies.

You can also share the cache data with the other servers of the cluster. By using the functions provided by the DRS, you can replicate or copy the data to the other members of the cluster. This process saves execution time and resources. For more information, see 10.9, “Data replication service” on page 334. The cache consistency is maintained by the DRS. If a cache entry is invalidated by one server, the invalidation is propagated to all the members.

The *cache monitor* application is available to manage the data, and monitor and verify the configuration of the dynamic cache. It needs to be installed as a normal application.

10.7.3 Data caching

Data caching is used to minimize back-end database calls and assure the integrity of the data. In most cases, the decision for data currency is a business decision. Multiple methods are available to configure data caching:

- Keep a local copy in a database within the same network realm as the application
- Cache data from a localized database in memory to minimize database reads
- Use EJB persistence to keep the data in the memory space of the running application

Sometimes data is provided by an external provider. Making live calls to this data can prove to be a SPOF and a slower performer. If no strict dependencies are on the currency of the data, offloading this data to a local database can provide large performance, availability, and scalability gains. The data can be refreshed periodically, preferably during off-peak hours for the application.

Database data caching

To minimize direct reads from the database, database systems usually offer one or more of the following options:

- Fetch-ahead constructs attempt to anticipate that additional pages from that table are required and then preload those pages into memory pools.
- Buffer pools keep data loaded into memory, assuming that it is likely the same data will be requested again.

Both of these constructs reduce disk access, opting instead for reading the data from the memory, increasing performance. These facilities assume that the data is predominately read-only. If the data has been written, the copy in memory can be stale, depending on the write implementation of the database. Also, memory buffers can be used to store data pages, reducing disk access. The key is to make sure that the system has enough memory to provide to the database. The database also takes advantages of the storage cache to avoid physical disk access.

Application data caching

Another option is to cache some of the database or web page data inside an application. You can do so by creating objects that are instantiated when the application server is started. Those objects pull the necessary information in memory, improving performance because the query is against an object in memory. Ensure that some synchronous or asynchronous mechanism (or both) is available to update this cache on a timely basis according to the system requirements. However, this approach can create additional memory requirements, especially if a dynamic cache that might grow over time is implemented.

EJB persistence implies loading the data into an EJB after a call to the data provider. This method is similar to database caching, except that caching takes place in the application space, not in the database server memory. The EJB has an access intent, which indicates the rules used to determine the currency of the data in the bean. From a performance standpoint, avoiding a call to an external database in favor of a local bean creates significant gains.

10.8 Session management

This section introduces the session management concept and explains how you can manage the sessions with WebSphere Application Server.

10.8.1 Overview

Multisystem scaling techniques rely on using multiple copies of an application server. Multiple consecutive requests from various clients can be serviced by different servers. If each client request is independent of every other client request, it does not matter whether consecutive requests are processed on the same server. However, in practice, client requests are not always independent. A client often makes a request, waits for the result, and then makes one or more subsequent requests. The processing of these subsequent requests requires information about the data processed in previous requests. Session management links requests that belong together.

In terms of session management, two types of requests are possible:

- Stateless

  A server processes requests based solely on information that is provided with each request, and does not rely on information from earlier requests. Therefore, the server does not need to maintain state information between requests.

- Stateful

  A server processes requests based on both the information that is provided with each request and information that is stored from earlier requests. To achieve this processing, the server needs to access and maintain state information that is generated during the processing of an earlier request. For example, the information can be the shopping cart of a customer for an online retailer website. The website needs to keep the information about the customer's selected items during the entire time the customer is shopping online. This retention is needed to manage the order. It might also be used to determine the path through future menus or options to display content.

For stateless interactions, it does not matter whether different requests are processed by different servers. For stateful interactions, the server that processes a request needs access to the state information necessary to run that request. Either the same server processes all requests associated with the dedicated state information, or that information is shared by all servers that require it. From a performance view, it is better that the first server that served the request continue to serve the other ones. This setup avoids exchanging the state data across the servers, minimizing the resources needed for communications.

The load distribution facilities in WebSphere Application Server use several different techniques to maintain state information between client requests:

- Session affinity

  The load distribution facility (for example, the web server plug-in) recognizes the existence of a client session. It then attempts to direct all requests within that session to the same server.

- Transaction affinity

  The load distribution facility recognizes the existence of a transaction, and attempts to direct all requests within the scope of that transaction to the same server.

- Server affinity

  The load distribution facility recognizes that, although multiple servers might be acceptable for a client request, a particular server is best suited for processing it.

The session manager of WebSphere Application Server, which is part of each application server, stores client session information. It also takes session affinity and server affinity into account when directing client requests to the cluster members of an application server. The workload management service takes server affinity and transaction affinity into account when directing client requests among cluster members.

10.8.2 Session support

As explained previously, information that is entered by a user in a web application is often needed throughout the application. The information that is coming from multiple requests from the same user is stored in a *session*. A session is a series of requests to a servlet that originate from the same user and the same browser. Each request that arrives at the servlet contains a session ID. Each ID allows the servlet to associate the request with a specific user.

The WebSphere session management component is responsible for managing sessions, providing storage for session data, and allocating session IDs that identify a specific session. It is also responsible for tracking the session ID that is associated with each client request through the use of cookies or URL rewriting techniques. Replicating the sessions in memory between the cluster members or sharing them by using a database are also possible, and improve the availability of the solution. These techniques make the infrastructure more tolerant to application server failures.

Session management in WebSphere Application Server can be defined at the following levels:

- Application
- Application server
- Web module

When planning for session data, keep in mind the following basic considerations:

- Application design
- Session storage options
- Session tracking mechanism

The following sections outline planning for each of these considerations.

Additional resource: For more information about session management planning, see the WebSphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=cpersess`

Application design

Although using session information is a convenient method for the developer, store only the objects that are needed for processing subsequent requests in the session. You need to minimize the size of the sessions. Keep in mind that most sessions are stored in memory. Managing large sessions comes with a performance impact.

Session tracking mechanism

You can choose to use cookies, URL rewriting, SSL session IDs, or a combination of these mechanisms to manage session IDs.

Cookies

Using cookies as a session tracking mechanism is common. WebSphere session management generates a unique session ID and returns it to the user's web browser to be stored as a cookie.

URL rewriting

URL rewriting requires the developer to use special encoding APIs, and to set up the site page flow to avoid losing the encoded information. The session identifier is stored in the page returned to the user. WebSphere encodes the session identifier as a parameter on URLs that are encoded programmatically by the web application developer.

URL rewriting can be used only for pages that are dynamically generated for each request, such as pages generated by servlets or JSPs. If a static page is used in the session flow, the session information is lost. URL rewriting forces the site designer to plan the user's flow in the site to avoid losing their session ID.

SSL ID tracking

With SSL ID tracking, SSL session information is used to track the session ID. Because the SSL session ID is negotiated between the web browser and an HTTP server, it cannot survive an HTTP server failure. However, the failure of an application server does not affect the SSL session ID. In environments that use WebSphere components with multiple HTTP servers, use an affinity mechanism for the web servers when SSL session ID is used as the session tracking mechanism.

Consideration: SSL tracking is supported in IBM HTTP Server. Session tracking by using the SSL ID has been deprecated since the release of WebSphere Application Server V7.

When the SSL session ID is used as the session tracking mechanism in a clustered environment, use either cookies or URL rewriting to maintain session affinity. The cookie or rewritten URL contains session affinity information. This information enables the web server to properly route requests back to the same server after the HTTP session is created on a server. The SSL ID is not sent in the cookie or a rewritten URL. Rather, it is derived from the SSL information.

The disadvantage of using SSL ID tracking is the performance degradation due to the SSL resource requirements.

Selecting multiple tracking mechanisms

You can combine multiple options for a web application:

- Use of SSL session identifiers has a preference to cookie and URL rewriting.
- Use of cookies has a preference to URL rewriting.

If you select SSL session ID tracking, consider also selecting cookies or URL rewriting to maintaining session affinity. The cookie or rewritten URL contains session affinity information. This information enables the web server to properly route a session back to the same server for each request.

Storage of session-related information

You can choose to store the session data by using one of these options:

- Local sessions (non-persistent)
- Database persistent sessions
- Memory-to-memory replicated persistent sessions

The last two options in this list support session data being accessed by multiple servers. Consider them when planning for failover. Using a database or session replication is also called *session persistence*.

Storing session data external to the system can affect performance. The impact depends on the amount of session data, the method chosen, and the performance and capacity of the external storage. Session management implements caching optimizations to minimize the impact of accessing the external storage, especially when consecutive requests are routed to the same application server.

Local sessions (non-persistent)

If the session data is stored in the application server memory only, the session data is not available to any other servers. Although this option is the fastest and the simplest to set up, an application server failure ends the session because the session data is lost.

The following settings can help you manage local session storage:

- Maximum in-memory session count

  With this setting, you can define a limit to the number of sessions in memory. This setting prevents the sessions from acquiring too much of the JVM heap and causing out-of-memory errors.

- Allow overflow

  This setting allows an unlimited number of sessions. If you choose this option, monitor the session cache size closely.

  Tip: Session overflow is enabled by default in WebSphere Application Server V8.5.

- Session timeout

  This setting determines when sessions can be removed from cache.

Database persistent sessions

You can store session data in an external database. The administrator must create the database and configure the session database in WebSphere Application Server through a data source.

The *use multi-row schema* setting gives you the option to use multi-row sessions to support large session objects. With multi-row support, the WebSphere Application Server session manager breaks the session data across multiple rows if the size of the session object exceeds the size for a row. This setting also provides a more efficient mechanism for storing and retrieving session contents when session attributes are large and few changes are required to the session attributes.

Memory-to-memory replicated persistent sessions

Memory-to-memory replications enable the sharing of sessions between application servers. Using memory-to-memory replication eliminates the effort of maintaining a production database and eliminates the SPOF that can occur with a database. You can choose the number of replicas and the level of persistence. Depending on this configuration, replicating

the session impacts the performance. The service transfers copies of objects across the network, and these new objects reduce the memory heap that is available for the other objects.

Memory-to-memory replication is based on the generic DRS. To learn how DRS works, see the next section.

Availability: Memory-to-memory persistence is available only in a distributed server environment by using WebSphere Application Server Network Deployment.

10.9 Data replication service

The DRS is an internal WebSphere Application Server component that is designed for generic data replication. Session manager, dynamic cache, and stateful session EJB can all use the replication service. DRS can increase the availability of your solution by replicating the data across a *replication domain*.

A replication domain is a group of servers that share data such as session data). For each domain, you must define how the data is replicated:

- To one server (single replica)
- To every server (entire domain)
- To a defined set of servers

When adding an application server to a replication domain, you must specify the replication mode for the server:

- Server mode

  In this mode, a server stores only backup copies of other application server data. It does not send copies of its own data to other application servers.

- Client mode

  In this mode, a server broadcasts or sends only copies of its own data. It does not receive copies of sessions from other servers.

- Both mode

  In this mode, the server can send its own data and receive data from other application servers. Because each server has a copy of all of the data, this mode uses the most memory on each server.

The number of replicas can affect performance. Smaller numbers of replicas result in better performance because the data does not have to be transferred and copied by the network into many servers. However, configuring more replicas makes your system more tolerant of failure because the data is backed up in several locations.

10.10 Highly available deployment manager

This section addresses the scale-out administration enhancements in WebSphere Application Server, and highlights the high availability of the deployment manager.

Although it is not required to have deployment manager running at all times, you might require highly available administrative capability. This configuration is especially important in environments that have significant reliance on automated operations, including application

deployment and server monitoring. Having multiple instances of deployment manager removes the SPOF for cell administration. This aspect assures the attainability of the administrative console, **wsadmin**, and scripting features to manage your environment. WebSphere Application Server provides a mechanism for cloning your existing deployment manager.

High availability is achieved by employing redundant deployment managers with a hot-standby model and the use of a shared file system. In this paradigm, one of the deployment managers is elected as primary. As primary it is considered an active deployment manager that hosts the cell-wide endpoints for the administrative functions. Other deployment managers are considered backups and are kept in the standby mode. The administrative function does not support multiple concurrent server processes writing to the same configuration. These peer deployment managers are available to take over the active role in case of failure or termination of the primary.

A highly available deployment manager component runs in each deployment manager to control which deployment manager is elected as the active one. The deployment managers in standby mode, although fully initialized, cannot be used to perform administrative functions. Therefore, the standby rejects any login and JMX requests.

New elements of the Intelligent Management feature are at the heart of high availability for the deployment manager. One new element in particular is the on-demand router. It automatically recognizes the currently active deployment manager and has endpoint configuration knowledge for routing the administrative communication. Multiple on-demand routers can be configured on different systems fronted by an IP sprayer to eliminate SPOFs. On-demand routers are always started first so the primary deployment manager is recognized in the environment.

Figure 10-6 illustrates a common topology for the highly available deployment manager.

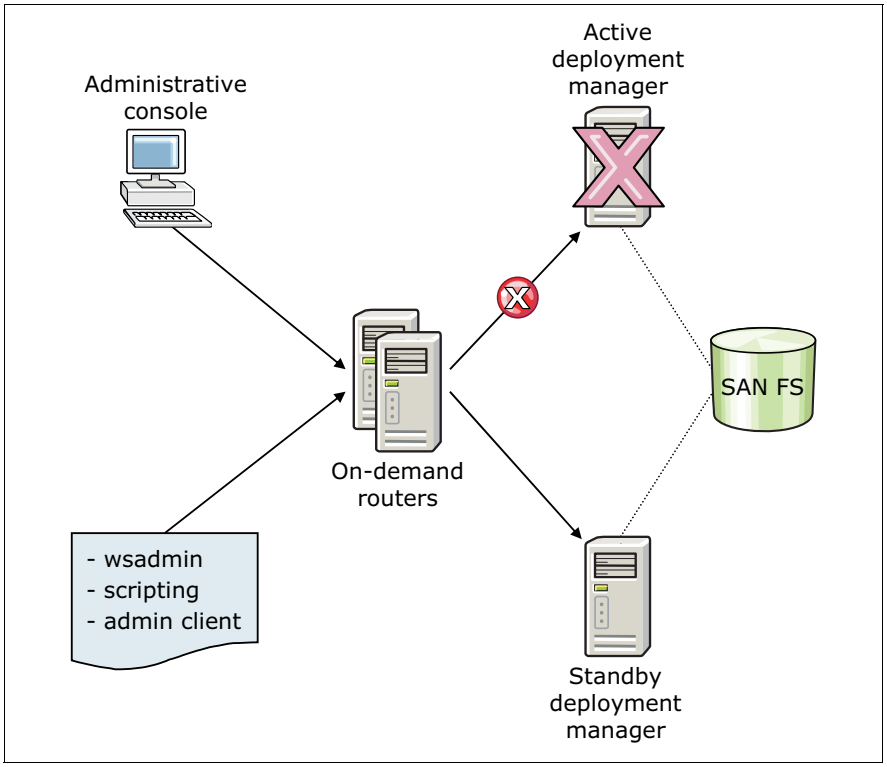


Figure 10-6 High availability deployment manager topology

When all the deployment managers in the cell are defined to the same core group, the routing information exposed to the on-demand router is consistent across all the deployment managers. If the deployment managers are in different core groups, core bridging must be employed. After configuring highly available deployment manager components, they need to be restarted for this solution to take effect. Include the on-demand routers and any active deployment managers in this restart.

The cell does not need to be homogenous, and deployment managers can be deployed on different operating systems and machines. However, like platforms are the preferred practice. High availability function is created by active and standby deployment managers that share an instance of the master configuration repository and workspace area.

When a deployment manager takeover occurs, work is not lost. This is ensured by installing deployment manager profiles on the same shared file system accessible by all the deployment manager instances. The WebSphere Application Server product binary files can be installed either locally or on the shared file system. Select your configuration based on the environment, performance, and your preference. The file system must support fast lock recovery because the safeguard for active deployment manager loss recognition is implemented as a file lock on the shared file system. The takeover can take a few seconds as the lock lease is released. For shared file systems, the IBM General Parallel File System (GPFS™) and Network File System Version 4 (NFS) are the preferred options. Whichever deployment manager is started first in your cell will be the active deployment manager, and the others will act as backup.

Important: The high availability (HA) deployment manager function supports use of only the JMX SOAP connector. The JMX RMI connector is not supported in this configuration.

The alternative for high availability deployment manager on z/OS is based on starting the deployment on a different logical partition (LPAR). For more information, see*:*

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101140

10.11 Whole-system Analysis of Idle Time Tool

IBM Whole-system Analysis of Idle Time is a lightweight tool that can be used in almost any WebSphere-based Java environment. It provides a quick performance analysis.

As the modern commercial enterprise-class applications become more complex and use a multitude of different components, debugging their performance becomes harder. These factors affect examining and diagnosing applications:

- Limited access to realistic scenarios in development or test systems
- Failure to install appropriate performance tools in resource constrained production environments

The IBM Whole-system Analysis of Idle Time tool was designed to help developers find bottlenecks in deployed workloads. This tool uses Javacores as its primary input. This type of input is available from any running JVM without command line or environment changes.

Enterprise-class multitier applications often suffer from problems that manifest not as hot spots, but as idle time that slows down completing an objective. The IBM Whole-system Analysis of Idle Time tool can be used to diagnose the root cause of idle time in server applications. The tool works by taking lightweight samples of Java activity on a single tier.

Using informative abstraction and compartmentalizing performance, it can often pinpoint the primary bottleneck on a multitier system. Many factors can contribute to undesirable idle time:

- Locking problems
- Excessive system-level activities like garbage collection
- Resource constraints
- Problems driving load

Example report showcases can be viewed at:

https://wait.ibm.com/wait/public/showcase/1/

IBM Whole-System Analysis of Idle Time tool has the following advantages:

- Identify bottlenecks
  - Gives high-level, whole-system, summary of performance inhibitors
- Zero installation time and convenient reporting
  - Uses built-in data collectors
  - Reports results in a browser
- Non-disruptive
  - No special flags, no restart
  - Can be used in any customer or development location
- Low-overhead
  - Uses only infrequent samples of an already-running application
- Simple to use
  - Usable by novices or experts with top-down level information approach
- Centralized knowledge base
  - Supports a rules and knowledge base to grow over time,
  - Can be adjusted quickly

This tool consists of data collector scripts related to platform. Linux, UNIX, and Windows based platforms support automated scripts to collect JVM and system performance information for analysis. Information about multiple processes can be captured, with sleep time between iterations. At the end of the collection interval, generated data is saved to a compressed file and prepared for upload to the data submission page. You need to register a user ID to be able to generate reports from your collected data.

For more information about using the tool and generating a report, see:

https://wait.researchlabs.ibm.com/

Restriction: This tool provides no scripting support to collect Javacores or system data on z/OS. Nevertheless, Javacores might still be collected manually by use of administrative console or the **MODIFY** operator command.

For more information about Whole Performance Analysis of Idle Programs, see:

http://researcher.ibm.com/files/us-sjfink/res0000076-altman.pdf

10.12 Checklist for performance, scalability, and high availability

Consider the following items as you plan performance, scalability, and high availability:

- Establish performance goals and identify workload characteristics (throughput, response time, and availability).
- Design your topology to meet the performance goals:
  - Determine your scalability techniques.
  - Plan for clustering:
    - Number of application servers
    - Physical location
    - Server weights
    - Affinity solutions
  - Determine whether the appropriate mechanisms are in place for workload management and failover. As part of this decision, consider where applications will be deployed (see 11.13, “Mapping applications to application servers” on page 379).
- Implement a monitoring system to watch for performance problems and to assist in determining whether adjustments are necessary.
- Monitor the following areas as potential physical bottlenecks:
  - Network load balancers
  - Firewalls
  - HTTP servers
  - Application servers
  - Database servers
  - Lightweight Third Party Authentication (LTPA) providers
- Examine initial settings for performance tuning parameters, adjust if necessary, and re-evaluate periodically:
  - JVM garbage policy, and heap maximum and minimum sizes
  - Web container
    - Thread pool
    - Maximum persistent requests
    - Timeout values
  - EJB container
    - Inactive pool cleanup interval
    - Cache size
  - Database connection pool
    - Maximum connections
    - Unused timeout
    - Purge policy
  - Database servers
    - Maximum database agents
    - Maximum connected applications
    - Query heap size
    - Sort heap size
    - Buffer pool size
    - Database memory heap

        - Application control heap
        - Lock timeout
    - Directory services
        - Database tuning
        - Authentication cache intervals
- Consider the scheduler service to run intensive tasks during off-peak hours.
- Evaluate session management needs:
    - Session ID mechanism (cookies, URL rewriting, or SSL)
    - Session timeout values
    - Session, transaction, and server affinity
    - Distributed session data store (memory-to-memory or database store)
- For messaging applications by using the default messaging provider, consider the following areas:
    - Quality of service settings
    - Bus topology

10.13 References

For more information, visit the WebSphere Application Server V8.5 Information Center:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=welc_content_cprf

11

Application development and deployment

The development and deployment of WebSphere applications involves developers, WebSphere infrastructure architects, and system administrators. Their involvement is a key factor for achieving a comprehensive and successful development and deployment plan. This chapter highlights important aspects and concepts that you need to consider during that planning.

This chapter includes the following sections:

- Application development and deployment features in WebSphere Application Server V8.5
- Recently supported programming models
- End-to-end lifecycle
- Development and deployment tools
- Naming conventions
- Source code management and collaboration
- Automated build process
- Automated deployment process
- Automated functional tests
- Test environments
- Managing application configuration settings
- Planning for application upgrades in production
- Mapping applications to application servers
- Planning checklist for applications
- Resources

11.1 Application development and deployment features in WebSphere Application Server V8.5

This section provides an overview of the features for application development and deployment that are provided in WebSphere Application Server V8.5:

- IBM Assembly and Deploy Tools for WebSphere Administration

  IBM Assembly and Deploy Tools for WebSphere Administration is the application assembly and deployment tool that is shipped with WebSphere Application Server V8.5. This tool replaces the previously available IBM Rational Application Developer Assembly and Deploy. With IBM Assembly and Deploy Tools for WebSphere Administration, developers can accomplish key assembly and deployment tasks. These tasks include editing of deployment artifacts, script development and testing, and application deployment and debugging. This tool is not intended for general application development.

- IBM WebSphere Application Server Developer Tools for Eclipse V8.5

  The IBM WebSphere Application Server Developer Tools for Eclipse V8.5 provides a development environment for developing, assembling, and deploying Java EE, OSGi, Web 2.0 and Mobile applications. You can use any of the following application servers:

  - WebSphere Application Server V8.5 Liberty profile
  - WebSphere Application Server V8.5 full profile
  - WebSphere Application Server V8.0
  - WebSphere Application Server V7.0

  When combined with Eclipse software development kit (SDK) and Eclipse Web Tools Platform, WebSphere Application Server Developer Tools for Eclipse provides a lightweight environment for developing Java EE applications.

- IBM Rational Application Developer for WebSphere Software, V8.5

  Rational Application Developer V8.5 provides a development environment for building applications that run on WebSphere Application Server V8.5. This tool supports all Java EE artifacts that are supported by WebSphere Application Server V8.5. Such artifacts include servlets, JavaServer Pages (JSP), JavaServer Faces (JSF), Enterprise JavaBeans (EJB), Extensible Markup Language (XML), Session Initiation Protocol (SIP), Portlet, and web services. It also includes integration with Open Services Gateway initiative (OSGi) programming model. The workbench contains wizards and editors that help in building standards-compliant, business-critical Java EE, Web 2.0, and service-oriented architecture applications. Integration with IBM Rational Team Concert™ provides a team-based environment that helps developers share information and work collaboratively. Code quality tools help teams find and correct problems before they escalate into expensive problems.

- IBM WebSphere Application Server for Developers

  WebSphere Application Server for Developers delivers an efficient development experience with the innovative features of WebSphere Application Server. This efficiency helps reduce the testing effort of developers and develop with confidence. They develop using a runtime environment that is identical to the production environment on which their applications will eventually run. This edition is available for not extra fee for the developer desktop.

- A broad set of integrated standards-base programming models

  Many of the core programming models in WebSphere Application Server V8.5 were available by using feature packs in the earlier versions. Now they are available for immediate use for WebSphere Application Server V8.5. The following programming models are included:

  - Java Batch programming model

    Use this model to build robust batch applications for running long running bulk transaction processing and computationally intensive work. WebSphere Application Server V8.5 includes efficiency and operational features that provide a unified batch architecture and a comprehensive batch solution.

    For more details, see Chapter 6, “WebSphere Batch” on page 137.

  - Communications enabled applications (CEA) programming model

    CEA is a programming model that allows you to add dynamic web communications to any application or business process. CEA provides Representational State Transfer (REST) and web service interfaces. The existing applications can quickly take advantage of communication features, involving phone calls and web collaboration.

  - XML programming model

    WebSphere Application Server V8.5 provides XML support to work with web applications that process data by using standard XML technologies. These technologies include Extensible Stylesheet Language Transformation (XSLT) 2.0, XML Path Language (XPath) 2.0, and XML Query Language (XQuery) 1.0. XQuery 1.0 allows you to query large amounts of data stored in XML outside of a database. Together, this technology simplifies application development and improves its performance and reliability.

  - Open Services Gateway initiative (OSGi) applications programming model

    The OSGi programming model allows development and deployment of modular applications that use both Java EE and OSGi technologies. This model provides control and flexibility to design and build applications and groups of applications from coherent, multiversion, and reusable OSGi bundles.

    The OSGi Enterprise Specification 4.2 Blueprint Container is used for declarative assembly of components. WebSphere Application Server V8.5 support for OSGi applications includes deploying web applications that use the Java Servlet 3.0 Specification and Java Persistence API. Support for the including EJB was introduced in WebSphere Application Server V8.5.

  - Service Component Architecture (SCA) programming model

    SCA accelerates application delivery and management in service-oriented architecture (SOA) environments. Service compositions can be created by using these methods:

    - Plain old Java objects (POJOs)
    - EJB 2.1, 3.0, and 3
    - OSGi applications
    - Spring components
    - Java servlets
    - JavaScript for Asynchronous JavaScript and XML (AJAX)

    This model is based on the open source Apache Tuscany project in conjunction with IBM.

    WebSphere Application Server supports the Open SOA Collaboration SCA specification. Additionally, WebSphere Application Server V8.5 provides support for the SCA OASIS programming model implementation.

- Session Initiation Protocol (SIP) programming model

  This programming model speeds the development of converged communication enhanced applications. It provides control over how messages are routed between applications. This model includes support for SIP Servlet Specification 1.1, also known as Java Specification Request (JSR) 289.

- Java Platform, Enterprise Edition (Java EE) 6 support

  WebSphere Application Server V8.5 supports the Java EE 6 specification. Java EE 6 expands the developer value that was introduced in Java EE 5 and continues to focus on developer productivity and ease-of-use enhancements. The following new features are included:

  - Enterprise JavaBeans (EJB) 3.1 (JSR 318)
  - Java Servlet 3.0 (JSR 315)
  - JavaServer Pages/Expression Language (JSP/EL) 2.1 (JSR 245)
  - JavaServer Faces (JSF) 2.0 (JSR 314)
  - Java Message Server 1.1 (JSR 199)
  - Java Contexts and Dependency Injection (JCDI) 1.0 (was Web Beans) (JSR-299)
  - Java Persistence API 2.0 (JSR 317)
  - Java EE Connector Architecture (JCA) 1.6 (JSR 322)
  - Java API for XML Web Services (JAX-WS) 2.2
  - Java API for RESTful Web Services (JAX-RS) 1.0 (JSR 311)
  - Java Authentication Service Provider Interface for Containers (JASPIC) 1.0 (JSR 196)
  - Bean Validation 1.0 (JSR 303)

  For more information, see the Websphere Application Server V8.5 Information Center at:

  `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=tovr_migrating_javaee`

- IBM WebSphere SDK Java Technology Edition Version 7.0

  WebSphere Application Server V8.5 supports IBM WebSphere SDK Java Technology Edition Version 7.0 as a pluggable JDK. Java 6 is installed with the product and used by default. Java 7 can be optionally installed and enabled by using the **`managesdk`** tool.

  This package provides a full-function SDK for Java. It is compliant with the Java Platform, Standard Edition (Java SE) 7 application programming interfaces (APIs). The SDK contains the Java application Runtime Environment and other tools that enable developers to create Java applications. This SDK includes the following features:

  - Project Coin (JSR 334) language enhancement features
  - NIO.2 (JSR 203) asynchronous I/O capabilities, extended file system attributes, and file system notifications
  - The `java.util.concurrent` capabilities by using a fork or join framework
  - Balanced garbage collection policy that targets short and consistent pause times on large heaps.
  - More detailed and efficient format for verbose garbage collection
  - Significant diagnostic improvements, including enhancements to `javacore.txt` contents (ulimits, native stacks, and native memory use)
  - Improvements to trace capabilities that enable Java stack traces to be captured at any tracepoint
  - Enhanced error logging to operating system logs, for example syslog on Linux

- Improved shared classes cache support:
  - Additional content
  - Better diagnostic files for corrupted caches
  - A programmable interface to find and delete caches
  - Better control of persistent cache file permissions
  - More control over displaying the cache contents

  For more information, see the IBM SDK Java Technology Edition V7 Information Center at:

  http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/index.jsp

- Simplified development of server-side REST applications by using JAX-RS

  JAX-RS offers a simpler way to develop, consume, and scale REST applications. It is composed by a collection of interfaces and Java annotations that simplifies the development process. With the annotations provided, you can declare resource classes and the data types they support. With this feature, developers can gain access to the runtime context. Through its extensible framework, it is also possible to integrate custom content handlers.

- Web services support

  Web services are web applications that improve the flexibility of business process and help to implement SOA. WebSphere Application Server V8.5 supports web services that are developed and implemented based on the Web Services for Java Platform, Enterprise Edition (Java EE) specification, V1.3. This specification supports WSDL Version 1.1, SOAP Version 1.1, and SOAP V1.2. The application server supports the JAX-WS programming model and the Java API for XML-based RPC (JAX-RPC) programming model. JAX-WS programming model simplifies application development through support of a standard, annotation-based model to develop web services applications and clients.

  The application server also supports Java Architecture for XML Binding (JAXB) 2.2 and JAX-WS 2.2. JAXB 2.2 provides an easy and convenient way to map Java classes and XML schema for simplified development of web services. JAX-WS 2.2 simplifies the development of web services with more platform independence for Java applications by using proxies and Java annotations. JAX-WS 2.2 requires JAXB 2.2 for data binding.

- Integrated WebSphere Application Server Web 2.0 and Mobile Toolkit support

  The WebSphere Application Server Web 2.0 and Mobile Toolkit simplifies the addition of AJAX rich desktop and mobile user interfaces. It also simplifies adding REST web services to Java web applications. Web 2.0 capabilities, such as AJAX and REST, help application developers to create more connected, interactive applications. These improvements result in higher customer satisfaction, user productivity, and enhanced decision making. New mobile AJAX components enable developers to create mobile web applications that run on devices such as smart phones and tablets.

- Monitored directory support

  By dragging applications into a defined and monitored directory, you can speed the process of editing, compiling, deploying, debugging, updating, and uninstalling applications. When an application is moved to the directory, after a defined interval, it is automatically installed and started. Likewise, if the application is removed from the directory, it is stopped and uninstalled. If the application or module is moved into the directory again, it is updated.

  The following file types are supported:

  - Enterprise archive (EAR)
  - Web archive (WAR)
  - Java archive (JAR)
  - SIP archive application resource

- IBM WebSphere Application Server Migration toolkit

  Application migrations can be performed faster by using the extensive set of tools that help to plan for and implement application migrations. The IBM WebSphere Application Server Migration toolkit is a suite of tools and knowledge collections provided at no extra fee. The toolkit allows you to quickly and cost-effectively migrate to WebSphere Application Server V7, V8.0, or V8.5. You can migrate from a previous version of WebSphere Application Server or competitive application servers that include Oracle WebLogic Server, Oracle Application Server, JBoss Application server, and Apache Tomcat. This toolkit provides a single solution for identifying, analyzing, and resolving application code quality and compliance requirements.

  For more information about application migration, see Chapter 17, “Migration” on page 547.

11.2 Recently supported programming models

This section provides more information about some of the WebSphere Application V8.5 supported programming models. Topics include Service Component Architecture, Open Services Gateway initiative, Communications Enabled Application, and Session Initiation Protocol. This section also explains the concept of a business-level application.

11.2.1 Service Component Architecture

Service Component Architecture (SCA) is a set of specifications that constitute a programming model for building applications using an SOA. SCA extends other SOA technologies, such as web services. It provides a platform and language-neutral component model that is based on open standards specified by the Open SOA Collaboration.

SCA allows the creation of complex composite applications based on previously existing service components. It is based on the following key principles:

- Service composition

  SCA offers a composition model that allows you to build new services from existing software components. SCA provides the metadata for describing these components and the connections between them while hiding their inner workings.

- Service development

  SCA has a language-neutral programming model. There are language-specific models for Java, Spring, C++, and other languages. Because SCA defines a common assembly mechanism, the language used for implementing a service does not need to be known by the service consumer.

- Service agility and flexibility

  The component model provided by SCA makes the composition and assembly of business logic simple, and allows for the flexible reusability of components. A component can be replaced easily by another component that provides the same service.

SCA contains the following key concepts:

- A *component* is the basic element of SCA and encapsulates business functions. It is a configured instance of an implementation, and has configurable services, references, and properties.
- *Implementation* is the actual code that provides the functions of the components.

- A *composite* is a combination of components. It is also called *composition* or *component assembly*.
- A *domain* contains one or more composites that runs in a single-vendor environment. Its components can be running on one or more processes, and on one or more systems.
- A *service* is the interface that is used by a consumer of the component. It specifies the operations that can be accessed by the component's client, but does not describe how the communication happens.
- A *property* is a configurable value that affects the behavior of a component.
- A *reference* describes the dependencies of a component on other software.
- A *binding* specifies how the communication with other components is accomplished.
- A *wire* represents a relationship between a reference and a service, and shows the existing dependency of a component on another component.
- *Promotion* is the process where a component's service is made available outside the composite. A promotion also occurs when a reference must become a reference for the composite.

Figure 11-1 illustrates the main concepts of an SCA domain.

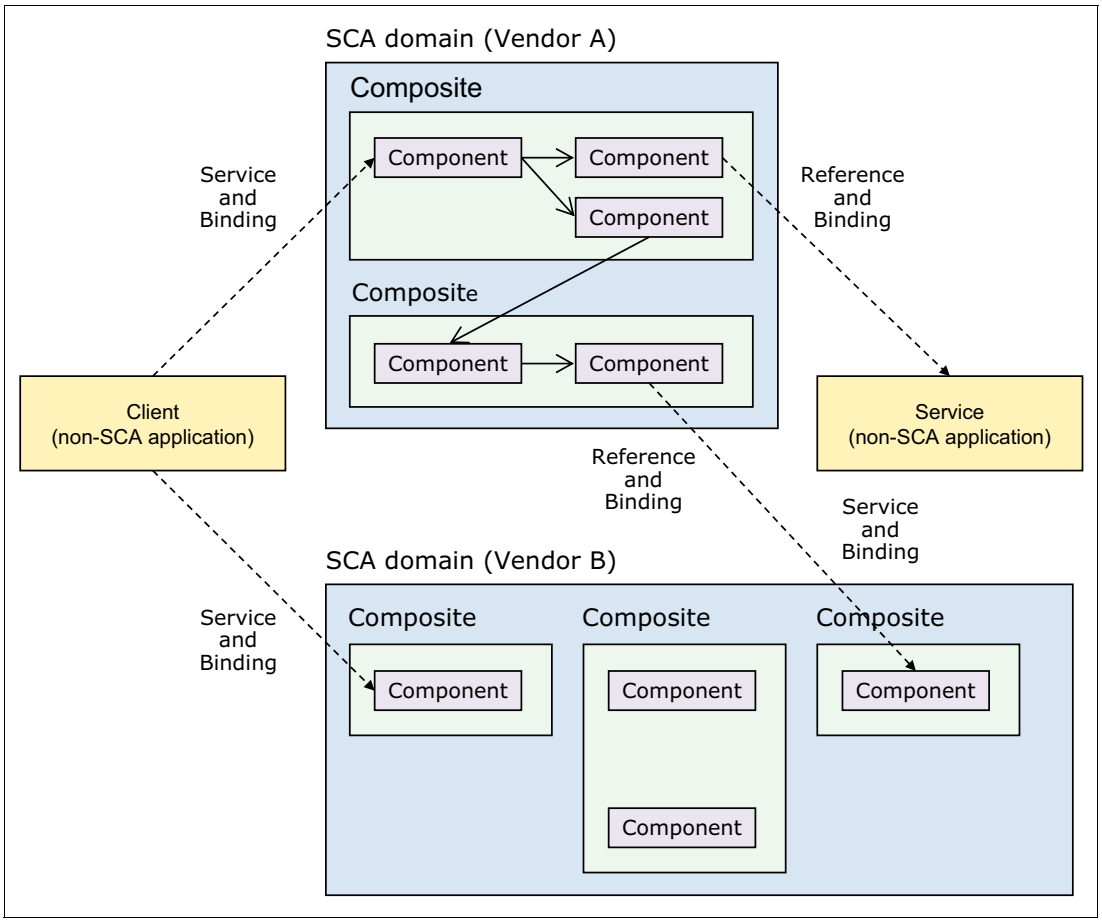


Figure 11-1 Key SCA concepts

For more information about SCA, see:

http://www.ibm.com/developerworks/library/specification/ws-sca/

SCA can increase programmer productivity and improve flexibility in application deployment. With SCA, you can focus on solving business problems, rather than worrying about the individual complexities of the technologies that connect service consumers and service providers.

WebSphere Application Server support for SCA is based on the Apache Tuscany open source technology. WebSphere Application Server V8.5 supports the Open SOA Collaboration specification and OASIS programming model for SCA.

For a complete list of specifications that are supported with WebSphere Application Server V8.5, see:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=csca_overview

Application development support

Rational Application Developer V8.5 provides a preview of SCA Tools for the development of OASIS SCA V1.1 applications, including deployment to WebSphere Application Developer V8.5. You can use the IBM Installation Manager to install Open SOA Collaboration SCA 1.1 or OASIS SCA 1.1 Tools.

Application packaging and deployment

WebSphere Application Server provides support for deploying SCA applications to the application server. Both JAR and WAR files are supported. Components that include service definitions must be packaged in a JAR file and deployed as assets for business-level applications. SCA WAR files can be deployed as well if they do not expose services over any binding type. WAR files must be deployed as WebSphere enterprise applications.

WebSphere Application Server also provides support in the administrative console and for the **wsadmin** command tool to install, delete, start, and stop SCA applications.

11.2.2 OSGi applications

The OSGi programming model defines a dynamic module system for Java. This system has a layered architecture, and is designed to run on various standard Java profiles. Eclipse Equinox is the reference implementation of the OSGi Service Platform Release 4 Version 4.2 Enterprise Specification. WebSphere Application Server uses Equinox as the framework for OSGi Applications.

These layers of OSGi architecture are supported with WebSphere Application Server V8.5:

- Modules layer

  The OSGi framework processes the modular aspects of a bundle in this layer. A *bundle* is a unit of deployment in OSGi.

- Lifecycle layer

  The bundle lifecycle management layer in OSGi enables bundles to be installed, started, stopped, and uninstalled dynamically. These processes are independent from the lifecycle of the application server.

- Services layer

  The services layer in OSGi intrinsically supports an SOA through its non-durable service registry component. Bundles publish services to the service registry, and other bundles can discover these services from the service registry.

OSGi application support in WebSphere Application Server allows the use of extensions to the Blueprint component model for declarative transactions and container-managed Java Persistence API. Deployment of web applications that use Java Servlet Specification Version 3.0 is supported.

Multiple versions of a class can be loaded simultaneously in the same application by using the OSGi mechanism. Additionally, a running application can be updated such that impact is only on those bundles that are affected by the update. Also, running applications can be extended and scaled as business demands it, without changing the underlying application. OSGi support includes using an integrated bundle repository and configuring the locations of external repositories to support reuse through the provisioning of bundles to applications.

WebSphere Application Server supports deploying applications in archive files that contain only application-specific content and metadata that points to shared bundles. This feature results in smaller application archive files. Also, when these shared libraries are shared by several OSGi applications, only one copy of the library is loaded into memory.

WebSphere Application Server V8.5 introduces support for OSGi programming model enhancements, including EJB support. WebSphere Application Server V8.5 also provides configuration of bean security in the Blueprint XML file of OSGi applications. Therefore, the methods of the bean can be accessed only by users that are assigned a specified role.

Enabling EJB OSGi bundles

An OSGi application can contain Enterprise JavaBeans (EJB). OSGi applications can access and start an enterprise bean directly. The enterprise beans in OSGi bundles can be developed from scratch. They can also be included from existing EJB assets and migrated to use OSGi modularity with minimal code changes. Stateful, stateless, and singleton enterprise beans are supported. OSGi application can also contain message-driven beans (MDBs).

Figure 11-2 depicts an EJB OSGi bundle. An EJB bundle includes the EJB module along with its OSGi metadata. It is then deployed as an OSGi bundle in an OSGi application.

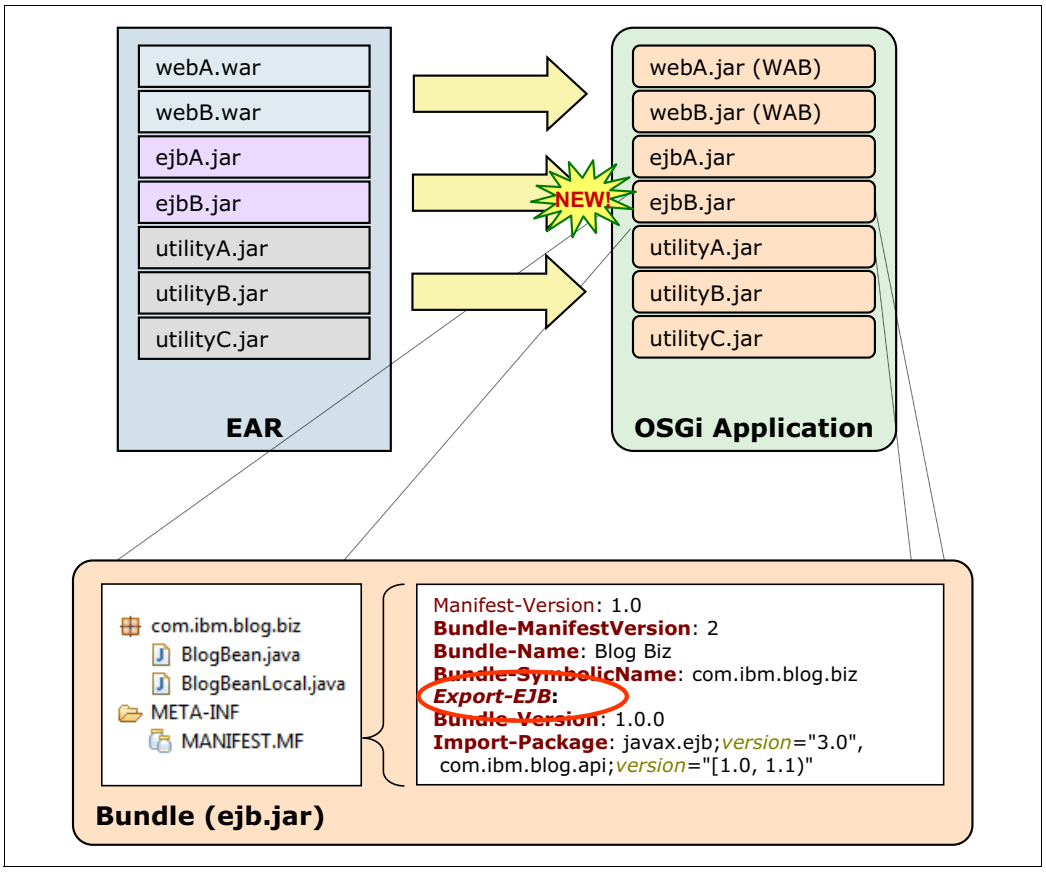


Figure 11-2 EJB OSGi bundle

Application development and deployment support

OSGi bundles are packaged as JAR files. A single OSGi application is packaged in an enterprise bundle archive (EBA) file, just as an enterprise application is packaged in an enterprise archive file. WebSphere Application Server also supports deploying existing WAR files as web application bundles to facilitate the use of an OSGi module system.

OSGi application development support is provided in IBM Rational Application Developer V8.5 and WebSphere Developer Tools. OSGi application can be deployed to WebSphere Application Server by using either the administrative console or **wsadmin** commands.

For more information about developing OSGi applications in WebSphere Application Server V8.5, see:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=thread_ta_dev_devdepeba

11.2.3 Business-level applications

A *business-level application* is an application beyond the Java EE definition. A business-level application is an administration model that provides the entire definition of an application as it makes sense to the business. A business-level application is a WebSphere configuration artifact, similar to a server or cluster, that is stored in the product configuration repository. This grouping notion for enterprise level applications includes WebSphere and

non-WebSphere artifacts, such as SCA packages, libraries, and proxy filters, under a single application definition. Business-level applications do not introduce new programming, runtime, or packaging models. You do not need to change application business logic or runtime settings. Figure 11-3 shows business-level applications.

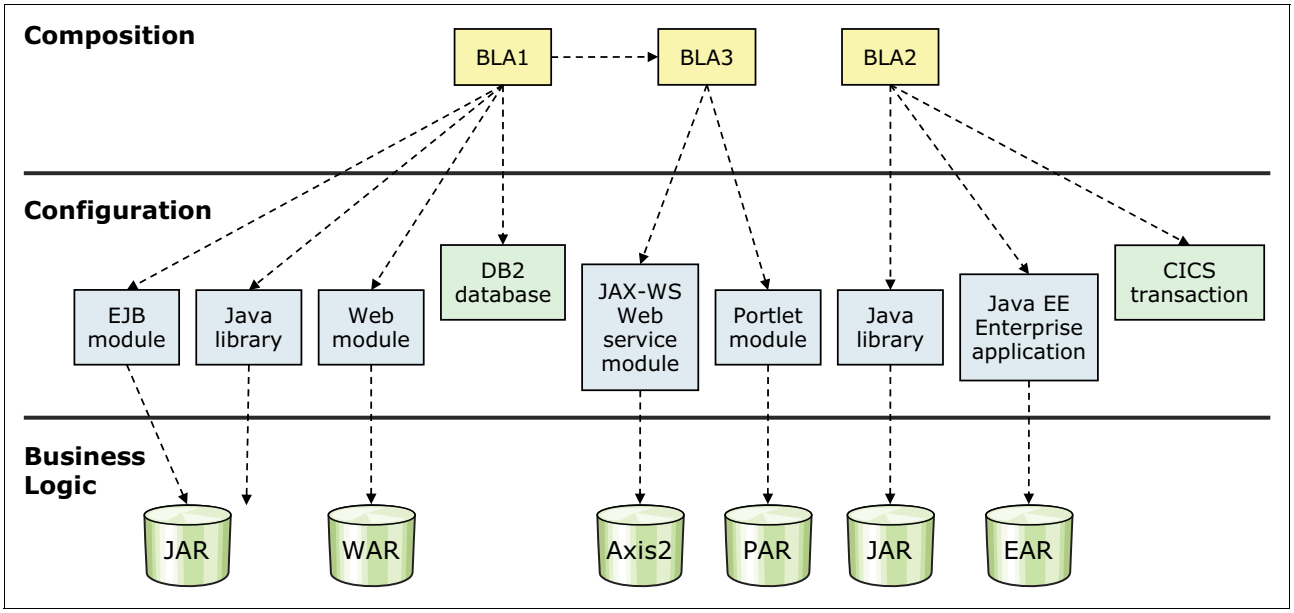


Figure 11-3 Business-level applications

A business-level application has the following characteristics:

- Lists one or more composition units that represent the application binary files.
- Might not explicitly manage the lifecycle of every artifact.
- Is a model that defines an application.
- Does not represent or contain application binary files.
- Can span more than WebSphere Application Server deployment target run times. Examples include a proxy server, a web server, and WebSphere Application Server Community Edition.
- Provides installation, distribution, activation, monitor, update, and removal management features for applications.
- Supports application service provider (ASP) scenarios by allowing single application binary files to be shared between multiple deployments.
- Aligns WebSphere applications closer with business as opposed to IT configuration.

In summary, a business-level application can be useful when an application has the following characteristics:

- Is composed of multiple packages.
- Applies to the post-deployment side of the application lifecycle.
- Contains additional libraries or non-Java EE artifacts.
- Includes artifacts that run on heterogeneous environments that include WebSphere and non-WebSphere run times.
- Is defined in a recursive manner (for example, if an application includes other applications).

11.2.4 Session Initiation Protocol applications

A *Session Initiation Protocol (SIP)* application is a Java program that uses at least one SIP servlet. A *SIP servlet* is a Java-based application component that is managed by a SIP servlet container and that runs SIP signaling. SIP servlets interact with clients by exchanging request and response messages through the servlet container.

SIP is used to establish, modify, and terminate multimedia IP sessions, including IP telephony, presence, and instant messaging. *Presence* in this context refers to the user status, such as active, away, or do not disturb. The standard that defines a programming model for writing SIP-based servlet applications is JSR 116.

WebSphere Application V8.5 complies with the Internet Engineering Task Force (IETF) and JCP SIP standards. A SIP application is packaged in a SIP application archive file. WebSphere Application Server V8.5 provides a SIP container to process SIP requests.

For more information about developing SIP applications in WebSphere Application Server V8.5, see:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=csip_sipwas

11.2.5 Communications enabled applications

Communications enabled applications (CEA) is a programming model that allows you to add dynamic web communications to any application or business process. CEA offers these functions:

- Establishing a call between two users
- Sharing sessions between two users
- Integrating communications features in applications with private branch exchange (PBX) systems
- Additional features that are required to support these functions

Enterprise developers do not need to have extensive knowledge of telephony or SIP to implement CEA. The CEA capability delivers call control, notifications, and interactivity, and provides the platform for more complex communications.

CEA is based on SIP-enabled services that use REST servlets and web services in a converged HTTP and SIP application. CEA includes a library of Dojo-style widgets for use in web applications. CEA widgets are extensible, allowing developers to customize them to handle more advanced tasks.

Examples of CEA application scenarios include:

- Click-to-call with co-browsing assistance from a customer service representative
- Shopping online with a friend
- Tracking and reporting call statistics

WebSphere Application Server V8.5 provides the following communication services:

- Telephony access allows you to incorporate telephony services in business applications, including making phone calls, receiving phone calls, and receiving call notifications within the web application.
- Multimodal web interaction allows you to provide session linking (*shared sessions*) between users who are browsing the same website from different locations. With session

linking, users can interact dynamically in collaborative ways, such as co-browsing or co-shopping web sessions.

For more information about CEA applications in WebSphere Application Server V8.5, see:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=ccea_overview

11.3 End-to-end lifecycle

The WebSphere Application Server V8.5 environment and its integration with Rational tools offers developers support at every stage of the application development lifecycle.

This lifecycle has the following key stages:

- Requirements gathering and analysis
- Prototyping
- High-level design
- Low-level design
- Implementation, coding, and debugging
- Unit testing
- Integration testing
- Functional verification testing
- Acceptance testing
- Performance testing
- Deployment
- Maintenance (including fixes, modifications, and extensions)

11.3.1 The Rational Unified Process

IBM Rational Unified Process (RUP) is a software engineering process. It is not a set of theoretical and idealistic practices. It is the result of many years of experience guiding many organizations and software projects to successful implementations.

RUP centers its practices on successful software projects that have the following characteristics:

- Adapt the process
- Balance stakeholder priorities
- Collaborate across teams
- Demonstrate value iteratively
- Elevate the level of abstraction
- Focus on quality

RUP provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its users within a predictable schedule and budget. It also helps improve team collaboration and facilitates communication across geographically distributed teams.

RUP is an iterative process, which means that the cycle can feed back into itself and that software grows as the lifecycle is repeated. The opposite is a waterfall model where the output of each stage spills into the subsequent stage.

This iterative behavior of RUP occurs at both the macro and micro levels. At a macro level, the entire lifecycle repeats itself. The maintenance stage often leads back to the requirements

gathering and analysis stage. At a micro level, the review of one stage might lead back to the start of the stage again, or to the start of another stage.

At the macro level, the Inception, Elaboration, Construction, and Transition phases can be identified in the process. These phases are periods of initial planning, more detailed planning, implementation, and finalizing and moving on to the next project cycle. The next cycle repeats these phases. At the micro level, each phase can go through several iterations of itself. For example, during a construction phase, coding, testing, and recoding can take place several times.

RUP identifies several disciplines that are practiced during the various phases. The first six disciplines (Business Modeling, Requirements, Analysis and Design, Implementation, Test, and Deployment) are known as *engineering workflows*. The three remaining disciplines (Project Management, Configuration and Change Management, and Environment) are known as *supporting workflows*.

These disciplines are practiced during all phases, but the amount of activity in each phase varies. The requirements discipline is more active during the earlier inception and elaboration phases, for example.

Figure 11-4 provides an overview of the RUP.

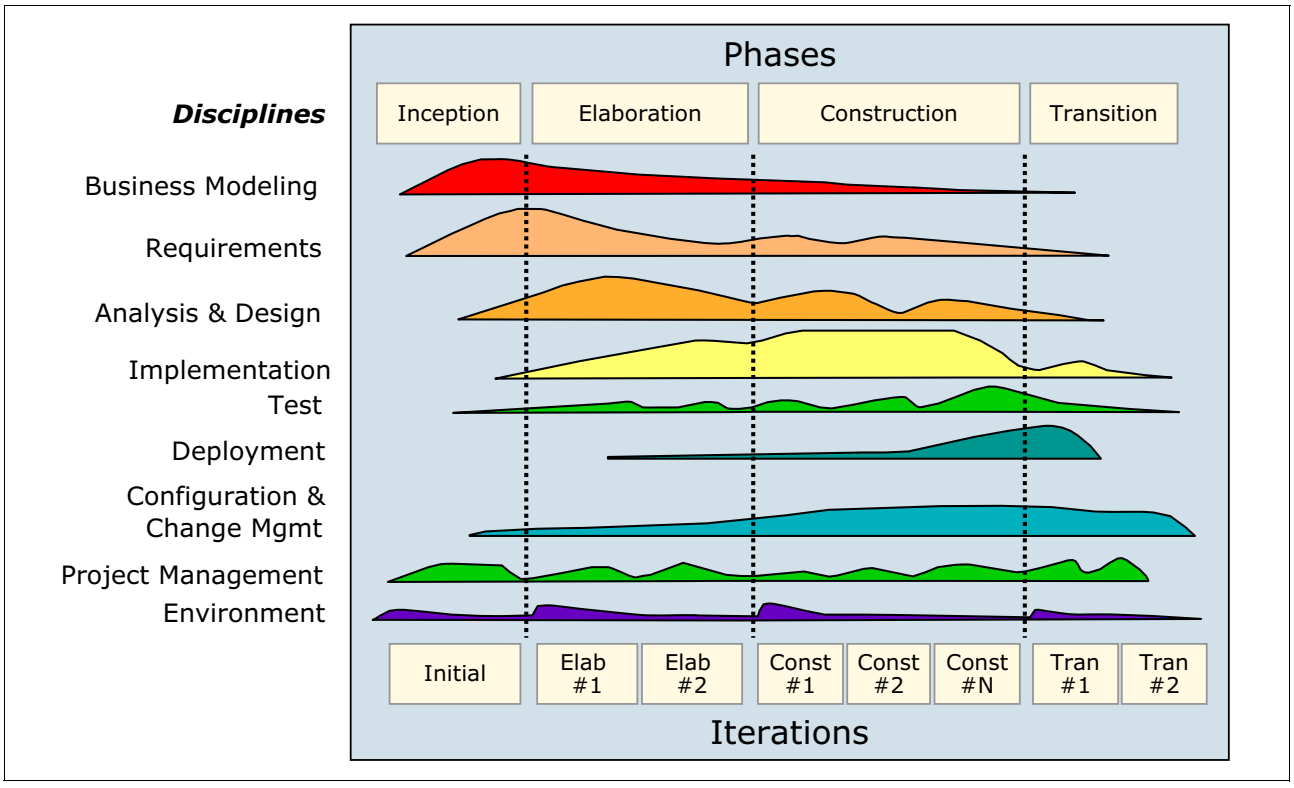


Figure 11-4 Rational Unified Process overview

RUP maps disciplines to roles. The roles break down into the following basic sets:

- Analysts
- Developers
- Testers
- Managers

Members of the team can take on more than one role. Also, more than one team member can have the same role. Each role might require the practice of more than one discipline.

RUP can be followed without using Rational Software, because after all, it is a process specification. However, RUP provides specific guidance (called *Tool Mentors*) on how to use Rational Software when following the process. Rational Method Composer is one of the tools that helps to customize RUP to meet the specific requirements of a project. The disciplines identified in RUP, such as requirements analysis, design, and testing, map to specific pieces of Rational Software and the artifacts that this software generates. RUP is a process that can be followed as much or as little as is required.

For more information about RUP, see the IBM Rational Unified Process (RUP) page at:

`http://www.ibm.com/software/awdtools/rup`

11.4 Development and deployment tools

Several tools in the WebSphere Application Server V8.5 environment help in the development and deployment of applications. All editions of WebSphere Application Server V8.5 include a full licensed version of the IBM Assembly and Deploy Tools for WebSphere Administration. They also include a trial version of the IBM Rational Application Developer for WebSphere Software V8.5.

Rational Application Developer for WebSphere Software V8.5 supports all features of WebSphere Application Server V8.5. It is a fully featured integrated development environment (IDE) for developing SIP, Portlet, web services, Java EE, and OSGi applications. It supports previous versions of WebSphere Application Server (V6.0, V6.1, V7.0, and V8.0) as an integrated test environment. It includes all Eclipse 3.6 features.

11.4.1 IBM Assembly and Deploy Tools for WebSphere Administration

IBM Assembly and Deploy Tools for WebSphere Administration helps in the assembly and deployment of applications only. It does not provide development capabilities. This tool has the following key components:

- Import and validate applications
- Edit deployment descriptors and binding files
- Edit EAR-level configuration (enhanced EAR)
- Create and debug Jython and **`wsadmin`** scripts
- Deploy EJB and web services
- Deploy applications to local or remote WebSphere Application Server V8.5 servers
- Debug applications on WebSphere Application Server V8.5

11.4.2 WebSphere Application Server Developer Tools for Eclipse, V8.5

The IBM WebSphere Application Server Developer Tools for Eclipse, V8.5 is a lightweight set of tools for developing, assembling, and deploying Java EE, OSGi, Web 2.0, and Mobile applications. The tool supports WebSphere Application Server V8.5 (including the Liberty profile), WebSphere Application Server V8.0, and WebSphere Application Server V7.0.

With the WebSphere Application Server V8.5 Liberty profile, this tool provides a fast and lightweight environment for the rapid development and unit testing of web, Web 2.0, Mobile, and OSGi applications.

The tool has the following features to manage the server:

- Starting and stopping the server and remote servers
- Application code can be published to the server
- Develop Java EE applications
- Provides several features to improve developer productivity, such as the use of annotation and deployment descriptor editors, code validations, quick fixes, and refactoring capabilities
- Develop JAX-RS and JAX-WS applications
- Rich Page Editor and a WYSIWYG editor are provided for developing Web 2.0 and mobile web applications

Developing and publishing OSGi applications to WebSphere Application Server V8.5 Liberty profile, V8.5, and V8.0 are also supported. The OSGi editors can be used to manage the metadata associated with bundles, bundle dependencies, and bundle fragments.

For more information, see:

http://publib.boulder.ibm.com/infocenter/radhelp/v8r5/topic/com.ibm.rad.install.doc/topics/wdt_overview.html

11.4.3 Rational Application Developer for WebSphere Software V8.5

Rational Application Developer for WebSphere Software V8.5 offers a more extensive set of tools that support enterprise development. IBM Rational Application Developer for WebSphere Software V8.5 can be used to design, develop, analyze, test, profile, and deploy high-quality web, SOA, Java, Java EE, and portal applications.

This product includes the following features:

- Fully integrated tools and support for IBM WebSphere Application Server V6.1 and later
- Tools, including many simple wizards and visual editors, that fully support the Java EE programming model, including web, Java, web services, and EJB applications
- Code quality, testing, and deployment tools, such as the enhanced runtime analysis to detect memory leaks or thread locks
- Web 2.0, OSGi, Java Persistence API 2.0, SCA, XML, CEA, portal, and web services development features
- IBM Workload Deployer (cloud) support
- Support for Java 7
- Ant scripting and JUnit testing framework
- WebSphere performance profiling and logging
- Agile development support with tools for refactoring code and unit testing
- Automated tools to manage server instances and server configurations, including automated creation and submission of **wsadmin** scripts
- Integration with IBM Rational Team Concert and IBM Rational ClearCase® so that management operations can be run within the development environment and increase collaboration and team productivity
- WebSphere Adapter Support for third-party products, such as SAP, PeopleSoft Enterprise, Siebel, Oracle E-Business Suite, and JD Edwards
- Unified Modeling Language (UML) modeling function

Rational Application Developer for WebSphere Software V8.5 provides support for batch and Java Persistence API development. The Liberty tool provides support for installing the Liberty run time from an archive file, and creating and editing Liberty bindings and extensions. The Liberty tool provides applications that can be created, compiled, profiled, and debugged by using Java 7.

For a complete list of the new features of Rational Application Developer for WebSphere Software V8.5, see:

http://www-01.ibm.com/software/awdtools/developer/application/

11.4.4 Monitored directory

The monitored directory feature in WebSphere Application Server V8.5 makes it easier to install applications. This version (and previous versions) also includes a deployment tool called *WebSphere Rapid Deployment Tools*. Table 11-1 compares the monitored directory feature and Rapid Deployment Tools.

Table 11-1 Comparison of monitored directory and Rapid Deployment Tools

Feature	Monitored directory	Rapid Deployment Tools
Deployment environments supported	Express, Base, Network Deployment, and z/OS environments	Base environment only
Process execution	Does not start a new process or daemon	Starts a separate process
Java EE support	Deployment of Java EE 5 and later modules	Assembly of J2EE 1.3 and 1.4 modules, and deployment of all Java EE module versions
Deployment options supported	Supports use of a properties file to specify deployment options	Does not support use of a properties file

With the monitored directory, developer productivity can be improved because all applications placed in the directory can be installed, updated, or uninstalled automatically.

Exception: Installing an enterprise application file by adding it to a monitored directory is available only on distributed platforms and z/OS. This option is not available on IBM i operating systems.

Configuring the monitored directory

To configure the monitored directory feature, perform these steps:

1. Log on to the administrative console.
2. Click **Applications** → **Global deployment settings**.
3. In the settings window, complete these steps:
   a. Select **Monitor directory to automatically deploy applications** to enable monitored directory deployment.
   b. For **Monitored directory**, specify a new value if you do not want to use the default. The path that you enter must exist because the product does not create it for you.
   c. For **Polling interval**, specify a different value in seconds if you do not want to use the default value of 5 seconds. The product changes 0 or negative values to 5 when the server starts.

d. Click **Apply** and save the changes.

You can also complete this configuration by using **wsadmin** scripting. For more information about the monitored directory configuration, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=trun_app_install_dragdrop

Installing, updating, and uninstalling an application

By default, the monitored directory uses the following paths:

- For base or stand-alone application servers, the *user_install_root*/monitoredDeployableApps/servers/*server_name* path
- For deployment managers, the *user_install_root*/monitoredDeployableApps/servers/*server_name* path

  For specific servers on a node or cluster, you must create the directory as in the following examples:

  - *user_install_root*/monitoredDeployableApps/nodes/*node_name*/servers/*server_name*
  - *user_install_root*/monitoredDeployableApps/clusters/*cluster_name*

If you add an EAR file, JAR file, WAR file, or SIP archive file to any monitored directory, the application is installed and automatically started. Keep in mind that the application server must be running for the application to start. If the node agent is stopped, the application is installed at the deployment manager level and synchronized when the node agent starts.

For deployment manager environments, the application must exist only in one monitored directory. If the application exists on another managed directory, you must first remove the application before adding it to a different monitored directory.

If the file you are moving exists in the directory, it might be updated. The application that is already deployed stops, the new module or application is deployed, and finally the updated module or application starts again.

Likewise, if you remove the file from the monitored directory, it is uninstalled. First the application stops, and then it is uninstalled.

Tip: The `SystemOut.log` file is updated every time a change in the deployment of the application occurs. The messages start with the CWLDD message key.

You can create a `deploymentProperties` directory under the `monitoredDeployableApps` directory to include a properties file to install, update, or uninstall applications. This alternative offers the option to specify application bindings. It runs the **wsadmin applyConfigProperties** command to run the action.

For more information, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=trun_app_install_dragdrop_prop

11.4.5 Which tools to use

The tool that you choose depends on your requirements. If you need to deploy and test applications on WebSphere Application Server V8.5 for fast turnaround times, choose IBM Assembly and Deploy Tools for WebSphere Administration or the monitored directory feature.

If you are developing applications, you can use WebSphere Application Server Developer Tools for Eclipse V8.5. If you want to take advantage of UML modeling, code quality testing, or change management operations, then consider Rational Application Developer for WebSphere Software V8.5.

Figure 11-5 illustrates the overview of features provided by the developer tools supported by WebSphere Application Server V8.5. It can be used as reference when choosing a developer tool.

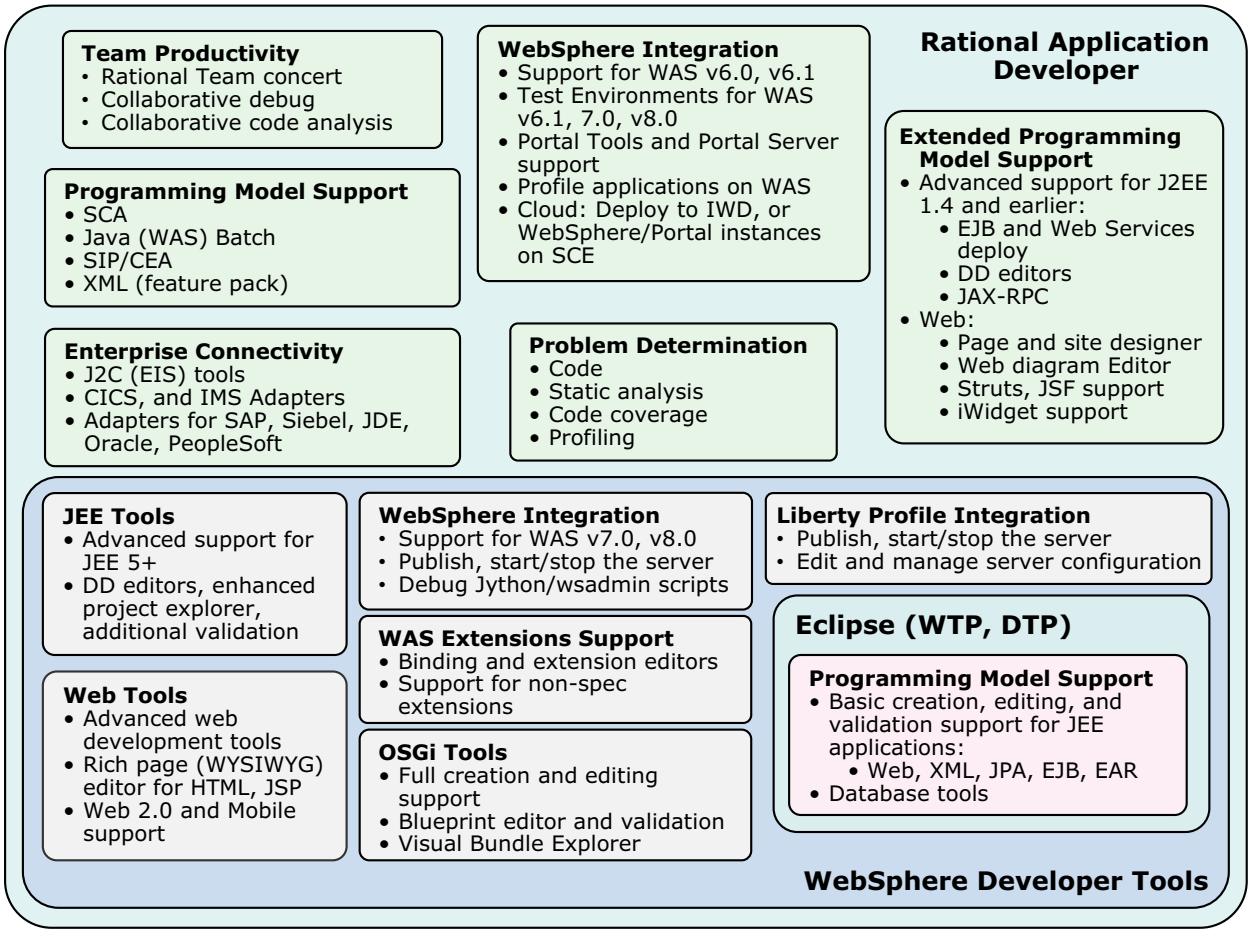


Figure 11-5 Overview of developer tools

11.5 Naming conventions

Spending extra time on application-related naming concepts pays off in practice. It can reduce the time spent on analyzing the source of issues during standard operations of future Java EE applications.

11.5.1 Naming for applications

Try to give the enterprise archives meaningful names that clearly indicate what the application is about. Choose a name that you, as a developer, can understand, but also that a system administrator, deployer, or tester, for example, can understand or interpret. The same guideline applies for the files or archives that are packaged within an application archive. Avoid including a number sign (#) in the name of the files, because doing so causes the deployment to fail.

Generally, a form of the version, release, modification, fix (VRMF) schema is used to organize code and builds. Commonly, a dotted number system, such as 1.4.0.1, is used. In this way, code management systems can be certain to identify, create, and re-create application builds accurately from the correct source code. Systems administrators and developers know exactly which version is used.

Append the version number to the EAR file name, such as in `OrderApplication-1.4.0.1.ear`. Consider appending only relevant information to the EAR file name to avoid names that are too long. You do not need to append the date if you correctly log the version number to that date.

Sometimes, the version number of included components, such as utility JAR files packaged in the EAR file, can also have version numbers in their file names. This practice can cause problems. Consider a utility JAR file with a version number in the file name, such as `log4j-1.2.4.jar`. If the number is updated to `log4j-1.2.5.jar`, each developer must update the class path settings in their workspace, which costs time. Instead, use a Source Code Management system and label the new JAR file as version 1.2.5. This allows you to keep the file name constant, such as `log4j.jar`.

To track all the versions of included components, consider including a bill of materials file inside the EAR file. The bill of materials file can be a simple text file in the root of the EAR file. This bill of materials file includes the following information:

- Versions of all included components
- Information about the tools used to build it
- The system on which the application was built

The bill of materials file can also include information about dependencies to other components or applications, and a list of fixes and modifications made to the release.

11.5.2 Naming for resources

When naming resources, associate the resource to both the application that uses it and the physical resource to which it refers. As an example, you can use a data source, but the concept holds also for other types of resources such as a messaging queue. Messaging queues can have names related to the business activity to which they are related. Remember, if your company already has a naming convention for other environments (non-WebSphere) in place, consider using the same naming convention in WebSphere.

For example, assume that you have a database called ORDER that holds orders placed by your customers. The obvious name of the data source is Order, and its Java Naming and Directory Interface (JNDI) name is `jdbc/Order`.

If the ORDER database is used only by a single application, the application name can also be included to further explain the purpose of the resource. The data source is then called `Order_OrderApplication`, and its JNDI name is `jdbc/Order_OrderApplication`.

Because the administrative console sorts resources by name, you might want to include the name of the application first in the resource, such as in `OrderApplication_Order`. This approach gives you the ability to sort or filter your resources according to the application that is using them.

To group and sort resources in the administrative console, you can also use the Category field, which is available for all resources in the administrative console. In this field, you can enter, for example, a keyword and then sort your resource on the Category column. Therefore, instead of including the name of the application in the resource name, you enter the application name in the Category field instead. If you have several different database vendors, you might also want to include the name of the database vendor for further explanation. The Category field is a good place to do that.

11.5.3 Naming resources in the Liberty profile

The Liberty profile has resources such as project, run time, profile server, application binary, and software development kit (SDK) files. These resources are packaged into a compressed file for deployment. The directory name of each resource in the Liberty profile compressed file is used as the resource name by the job manager. For example, if the directory name for the Liberty profile run time is `08.05.00.00`, its resource name is `08.05.00.00`, and its resource ID is `libertyRuntime/08.05.00.00`. Therefore, use directory and file names that are portable between operating systems in case you need to deploy the image to multiple operating systems. For example, do not name resources that differ only in capitalization so that you can deploy to Windows platform such as `jre_01.06.00` and `Jre_01.06.00`.

When using version numbers with major and minor numbers in resource names, such as 8.5.0.1, ensure that you allocate enough digits. Doing so allows you to use simple lexical string comparison to compare versions. For example, instead of using `8.5.0.1`, use `08.05.00.01`.

Finally, to avoid name conflicts in resources, use project names.

11.6 Source code management and collaboration

In development, you must manage generations of code. Carefully organize and track application builds and the source code that is used to create them to avoid confusion. In addition to tracking the version of the source code, track the version of the build tools and which system was used to generate a build. Not all problems are due to bugs in source code.

Developers usually use an IDE, such as Rational Application Developer for WebSphere Software V8, to produce code. Code in an IDE is stored in a workspace on the local file system of each developer. As the project continues, and possibly new members join the team, the code grows. Eventually, you must manage the code in a central master repository.

Regardless of the size of the developers group, code needs to be merged in an automatically, repeatable, and reliable way. It is common during development that two or more developers work on the same code or assets. Manually merging of all the changes can lead to bugs in the code, and can be time consuming.

Another important aspect during software development is how communication is done between developers, stakeholders, and other people relates to the development process. In large organizations, the development team or stakeholders are often dispersed around the globe. In such situations, it is difficult to keep clear visibility on how the development process is going and make sure it meets the business requirements.

Collaborative software helps to improve how the different teams or people communicate with each other. It improves team productivity by interconnecting people who can give valuable feedback, and helps identify defects when it costs less to fix them.

Source code management systems and collaborative systems help keep control of the source code of the application, and ensure efficient communication across the team.

11.6.1 IBM Rational ClearCase

IBM Rational ClearCase organizes its code repositories as a versioned object base (VOB). A VOB contains the versioned file and directory elements. Users of Rational ClearCase are organized according to their roles. Each user has their own view of the data in the VOB on which they are working. Rational ClearCase tracks VOBs and views. It coordinates the checking in and checking out of VOB data to and from views.

As the role-based model suggests, Rational ClearCase is a source code management system and a Software Asset Management (SAM) system, meaning that it manages code and other assets. These assets might be produced by the other Rational products with which Rational ClearCase integrates, such as libraries, documentation, binary files, and web artifacts. The asset only needs the ability to be represented as digital content for Rational ClearCase to manage it.

ClearCase integrates with the following Rational products:

- Rational Asset Manager (asset reuse software)
- IBM Rational Build Forge® (advanced assembly and build software)
- IBM Rational ClearQuest® (change management software)
- Rational Enterprise Suite Tools
- Rational IDEs
- Rational Unified Process

Artifacts, such as use cases generated by Rational IBM RequisitePro®, can be stored in Rational ClearCase. The artifacts can then be fed into an IBM Rational Rose® design model. In this model, they can be used to design Java components and generate Unified Modeling Language (UML) diagrams and documentation.

Rational ClearCase can also be used to implement the Unified Change Management (UCM) process. This change management process can be enhanced by using Rational ClearCase with Rational ClearQuest, which is a change and defect tracking software.

Rational ClearCase software is scalable. Rational ClearCase LT is a scaled down version of Rational ClearCase for small-to medium-sized teams. It can be upgraded seamlessly to Rational ClearCase as user needs change. Additionally, you can use an IBM Rational ClearCase MultiSite® add-on to support geographically dispersed development teams. This tool also supports a range of platforms (Linux, UNIX, Windows, and z/OS environments), allowing teams to use their preferred environment. It also allows you to keep audit trails of who changed the code or artifacts, and when those changes were made.

In short, although Rational ClearCase is a source code management system, it is also a part of the Rational toolset and RUP. For more information, see the Rational ClearCase page at:

http://www.ibm.com/software/awdtools/clearcase/

11.6.2 Concurrent Versions System

Concurrent Versions System (CVS) uses a branch model to support multiple courses of work that are isolated from each other but are still highly interdependent. By using branches, a development team shares and integrates ongoing work. A branch can be thought of as a shared workspace that is updated by team members as they change the project. With this model, individuals on a CVS team project can share their work with others as changes are made. They can also access the work of others as the project evolves. A special branch, called HEAD, represents the main course of work in the repository. HEAD is often called the *trunk*.

CVS has the following features:

- It is available to use at no charge under the GNU license.
- It is open source.
- It is widely used in the development community.
- Other source code management repositories can be converted to CVS.
- Many client applications, such as WinCVS, are available without additional charge.
- It can store text and binary files.
- It handles versioning and branching.
- It is a centralized repository.

11.6.3 Subversion

Subversion is an open source version control system that is available at no cost and tracks the entire file system and files. It creates versions of directories and individual files, and stores them in a repository. Each time a change is made to the files or directories, the change is recorded in the repository. You can track the history of changes on files or directories by reviewing the log files that are maintained by Subversion. Each file or directory has a corresponding log file.

Subversion is easy to configure and offers rich graphical and command-line interfaces to manage files and directories. For more information, see the Apache Subversion website at:

`http://subversion.apache.org/`

11.6.4 Rational Team Concert

Rational Team Concert is built on the IBM Jazz™ platform, which provides a common collaboration environment to improve communication across the teams in your organization. With efficient communication during the development of your applications, you can produce quality software that satisfies all requirements and stakeholder expectations more easily.

Collaboration facilitates customer or user involvement during the development of the applications. Customer and user feedback during this phase is valuable. If application development meets stakeholder expectations from the beginning, its probability of success is higher.

In a single integrated environment, Rational Team Concert offers the necessary tools to enhance productivity during the development lifecycle of your applications. It uses a Web 2.0 oriented portal, with customizable views to display relevant information about the project:

- News and events
- Current build status
- Work in progress
- Changes made and requested

- Comments from other teammates
- Current assigned work to other members of the team

This information can be useful for stakeholders who want clear visibility of the project status.

Rational Team Concert also has its own source code management system that can help support geographically distributed teams. Members of the same or different groups can work together on the same code or artifacts by communicating through its integrated instant messaging system. This software is suitable for large teams and also fits small development groups. It is available at no additional cost for groups of 10 or fewer developers.

The tool is widely integrated with other products in the areas involved in software development, such as the following examples:

- Development:
  - IBM Rational Application Developer for WebSphere Software (see “Rational Application Developer for WebSphere Software V8.5” on page 356)
  - Eclipse
  - NetBeans
- Requirements management:
  - IBM Rational Requirements Composer
  - iRise Connect for IBM Rational Requirements Composer
- Build and Process Automation:
  - IBM Rational Build Forge (see “Rational Build Forge” on page 367)
  - Maven
  - CruiseControl build system
- Version Control:
  - Rational ClearCase
  - CVS (see “Concurrent Versions System” on page 363)
  - Subversion (see “Subversion” on page 363)
- Collaboration:
  - IBM Lotus SameTime
  - GoogleTalk
  - Skype Internet phone service

For more information, see the following websites:

- Rational Team Concert

  http://www.ibm.com/software/rational/products/rtc/
- Rational software

  http://www.ibm.com/software/rational

11.6.5 Choosing the correct tools to use

The correct choice of tools depends on several factors, including your development environment and needs. The topics in this section help select the correct tools to manage your source code and meet the collaboration needs of your development projects.

Current software and processes

The choice of tools depends on the existing situation, the source code management, communication, and the current development process requirements and their requirements in the future. If a team uses CVS, Subversion, and communications tools, and an existing, successful development process is in place, then Rational ClearCase or Rational Team Concert might not be necessary. This is especially true if the size and complexity of requirements is not likely to grow in the future. Rational Team Concert is valuable when the development process does not allow clear visibility of the project status or makes it difficult for the team members to work together and coordinate.

Team size

Rational Team Concert or Rational ClearCase LT are good options for smaller teams. As mentioned previously, Rational Team Concert is available at no cost for groups of 10 or fewer developers. Both tools can be upgraded later to keep development control as the team continues to grow.

On large development projects, Rational ClearCase and Rational ClearQuest have a MultiSite option that allows for easier development by geographically dispersed development teams. Also, a collaboration tool, such as Rational Team Concert, delivers great value when team members are in separate geographical places. It also makes sense when the team becomes too large to coordinate easily. In this case, regular meetings or email messages are not agile enough to coordinate the development activities and track how the project is going. For small teams where communication and teamwork go smoothly, it might not be necessary to integrate a collaboration tool.

Complexity of requirements

RUP provides a holistic approach to the end-to-end development lifecycle. Use of the UCM process, which is part of the RUP, can shield users from complex code tagging and branching. CVS and Subversion do not offer this support. Alternatively, the collaboration capabilities of Rational Team Concert can help manage complex requirements and planning. Collaboration of the correct people must be a priority during the requirements analysis and tracking. Consider also that Rational Team Concert can be integrated with other specialized software for requirements management such as Rational Requirements Composer.

Cost

From the source code management perspective, CVS and Subversion are often the cheaper option because they are available at no cost. In terms of hardware, the hardware costs for hosting CVS or Subversion are usually cheaper because of their smaller footprint. However, these economies might be false. The limitations of CVS and Subversion can cause a team to change to Rational Team Concert or Rational ClearCase later. The same applies for collaboration software. The most important aspect when planning the cost factor is to evaluate the total cost of ownership of the solution. When buying a software solution and evaluating costs, consider factors such as performance, support, updates, migration processes, and other associated risks.

Change management process

If the development team uses CVS or Subversion rather than Rational ClearCase, the team does not get a prescribed change management process such as the UCM. If the team's organization does not have its own change management process, create such a process in the correct place. Likewise, Rational Team Concert can help improve change management processes, if any. If you do not have a change management process, it can help in putting one into place. Its tracking capabilities and collaborative change communication can help large organizations to gain control over the changes during the development phase.

Summary

In summary, the smaller the development team is and the less complex the requirements are, the more likely that CVS, Subversion, and Rational ClearCase LT are good choices. If the development team is less than 10 developers, Rational Team Concert is also a cost-effective option. In small environments, evaluate collaboration to see whether it can really improve the development process. As team size and complexity grows, Rational ClearCase, Rational ClearCase MultiSite, and Rational Team Concert become more attractive. Existing processes, software, and the budget for new software, hardware, and training are likely to affect the decision further. Consider all factors in matters of cost, as there might be false economies.

11.7 Automated build process

If the build process is not managed in the appropriate way, it can reduce team efficiency and provoke failed deployments in the production environments. Manual processes are not reliable, and you must avoid them, especially when they are related to critical operations in the organization.

When you do not have an automated process, you might run into the following problems:

- Failures occur on your test or production environment because the code was not packaged correctly.
- The wrong code was deployed, causing the application to fail.
- You must wait to get the code out to the test, staging, or production environments because the only person who has control over these areas is unavailable.
- You cannot reproduce a problem on production because you do not know what version of files are in production at the moment.
- Bottlenecks occur from different applications that need to be deployed.
- Requested application changes are not completed on time, resulting in customer dissatisfaction.
- A manual process requires a longer time to market of the product or service that your applications are trying to serve.

The time spent developing an automated build script will pay for itself over time. After you establish an automatic build process, you can virtually eliminate failures due to improper deployment and packaging, and reduce the build turnaround time. You can also easily re-create what is in each of your environments and ensure that the code base is under configuration management.

11.7.1 Apache Ant

Several tools, such Apache Ant, Apache Maven, and CruiseControl, are on the market to help you develop a build script. This section focuses on Apache Ant because WebSphere Application Server provides a copy of the Ant tool. Ant is a Java language-based build tool that extends Java classes and uses XML-based configuration files to run its job. These files reference a target tree in which various tasks are run. Each task is run by an object that implements a particular task interface. Ant has become a popular tool in the Java world.

WebSphere Application Server provides the Apache Ant tasks in the `com.ibm.websphere.ant.tasks` package. The Javadoc for this package contains detailed information about the Ant tasks and how to use them.

By using the Ant tasks included in WebSphere Application Server, you can perform the following tasks:

1. Install and uninstall applications.
2. Run EJB 1 (all versions), 2 (all versions), and 3 (all versions) deployment and JSP precompilation tools.
3. Start and stop servers in a base configuration.
4. Run administrative scripts or commands.

By combining these tasks with the tasks provided by Ant, you can create build scripts that pull the code from the source code management repository. The scripts can then compile, package, and deploy the enterprise application on WebSphere Application Server. To run Ant and have it automatically detect the WebSphere classes, use the **ws_ant** command. Ant tasks can be used for building application code. Apache Struts framework can be used to create an extensible development environment for applications.

For more information about Ant, see the Apache Ant website at:

http://ant.apache.org/index.html

11.7.2 Rational Build Forge

Another product to consider is IBM Rational Build Forge. IBM Rational Build Forge provides a framework to automate the software assembly process. It offers different versions according to the needs of an organization, so a build automation process can be implemented in teams of varying sizes.

The Rational Build Forge tool helps software development teams be efficient during build and deployment of applications by providing the following benefits:

- Automates build and deployment activities during the development lifecycle
- Allows integration of existing tools and assets used before, and links them together to improve efficiency
- Offers better utilization of hardware resources by simplifying build and deployment process
- Provides faster software releases
- Provides easy integration with IBM Rational Automation Framework for WebSphere that can be used to automate WebSphere Application Server or WebSphere Portal administrative tasks such as application deployment and configuration.
- Reduces associated costs as the process becomes more efficient

For more information, see the Rational Build Forge website at:

http://www.ibm.com/software/awdtools/buildforge/index.html

11.8 Automated deployment process

Automating application deployment is something to consider whether it is done more than one time. Successful automation provides an error-free and consistent application deployment approach. Most application deployment not only involves installing the application itself, but also creating other WebSphere objects, configuring the web servers, file systems, and other resources.

You can use the following approaches to automate the deployment process:

- Depending on the operating system, you can use shell scripting to deploy the applications with Java TCL (Jacl) and Jython (Java Python) scripts.
- WebSphere Application Server provides a script library with Jython script procedures that you can use to automate common administrative tasks. You can use the Jython scripting library code as a sample syntax to write custom scripts. Each script example in the script library demonstrates preferred practices for writing **wsadmin** scripts.

  For more information, see WebSphere Application Server V8.5 Information Center at:

  http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=welc_ref_adm_jython

  Also, see the following IBM developerWorks topic:

  http://www.ibm.com/developerworks/websphere/library/samples/SampleScripts.html
- You can use the built-in capability in WebSphere Application Server V8.5 for automated deployments. For more information, see 11.4.4, “Monitored directory” on page 357.
- IBM Rational Automation Framework for WebSphere makes it easier to accomplish the complex tasks that are involved in managing the WebSphere environment. It is designed to automate installation and patching, configuration change management, and application deployment.

  Rational Automation Framework for WebSphere provides a library of over 500 field tested configuration and installation scripts. They are organized in a configuration change management repository. This repository maintains a history of all deployment steps and application deployment automation, which allows organizations to deliver software faster with fewer resources.

  By automating and directing the execution of deployment steps, Rational Automation Framework for WebSphere ensures that steps are applied to the correct environment in the correct sequence. This process ensures that WebSphere deployments are accurate and reliable. This automation also allows WebSphere Application Server administrators to make changes consistently across cells, and mitigates the risk of human error while improving the quality of the environments.

  For more information, see the Rational Automation Framework for WebSphere Information Center at:

  http://publib.boulder.ibm.com/infocenter/rafhelp/v3r0/index.jsp?topic=/com.ibm.help.common.infocenter.raf/helpindex_raf.html

11.8.1 Application deployment in the Liberty profile

You can deploy web applications or OSGi applications to the Liberty profile by using one of the following methods:

- You can drop the application into a previously defined “drop-ins” directory.

  By default, the “drop-ins” directory is monitored automatically. If the application is dropped into this directory, the application is deployed on the server automatically. Similarly, if the application is deleted from the directory, the application is removed automatically from the server.

  You can use this directory for applications that do not require additional configuration, such as security role mapping. There is no requirement to include the application entry or any relevant information in the server configuration. You can also configure the name and location of this directory.

- You can add an application entry to the server configuration.

  For applications that are not in the “drop-ins” directory, you can specify the location of the application by using an application entry in the server configuration file. The location can be on the file system or at a URL. If the application is available at a URL, the application manager downloads the application to a temporary folder inside the server work area. The application manager then starts the application.

  If the application entry is added to the server configuration after the server starts, the changes are picked up dynamically.

Additionally, you can use the developer tools that are supported by WebSphere Application Server V8.5 to deploy an application in the Liberty profile.

11.9 Automated functional tests

A *functional test* verifies that an application is working as expected. Functional tests are made from the user perspective. They ensure that the application correctly fulfills the business needs it was intended for.

When the budget for the development process is tight, the testing phase is often sacrificed. However, testing your applications before placing them into production can mean a significant cost reduction. It can help avoid damaging your image when service is down because of nonfunctional applications.

Depending on the business size, a functional test can be incorporated to test critical and complex applications only. Keep in mind that testing involves certain levels of investment similar to any other process during the application development lifecycle. To make functional tests less error prone and less costly over time, consider automating such tests.

Automation of functional tests offers the following benefits:

- Reduced development and maintenance costs
- Faster test time
- Faster application availability
- Higher levels of accuracy and consistency throughout the tests

IBM offers a rich set of software tools for implementing automated test solutions. These solutions solve many common problems and, therefore, reduce complexity and cost. For more information, see Rational Functional Tester at:

http://www.ibm.com/software/awdtools/tester/functional/

11.10 Test environments

Before moving an application into production, you must test it thoroughly. Because many kinds of tests need to be run by different teams, an effective test environment often consists of multiple test environments.

Figure 11-6 shows an overview of an effective test environment setup.

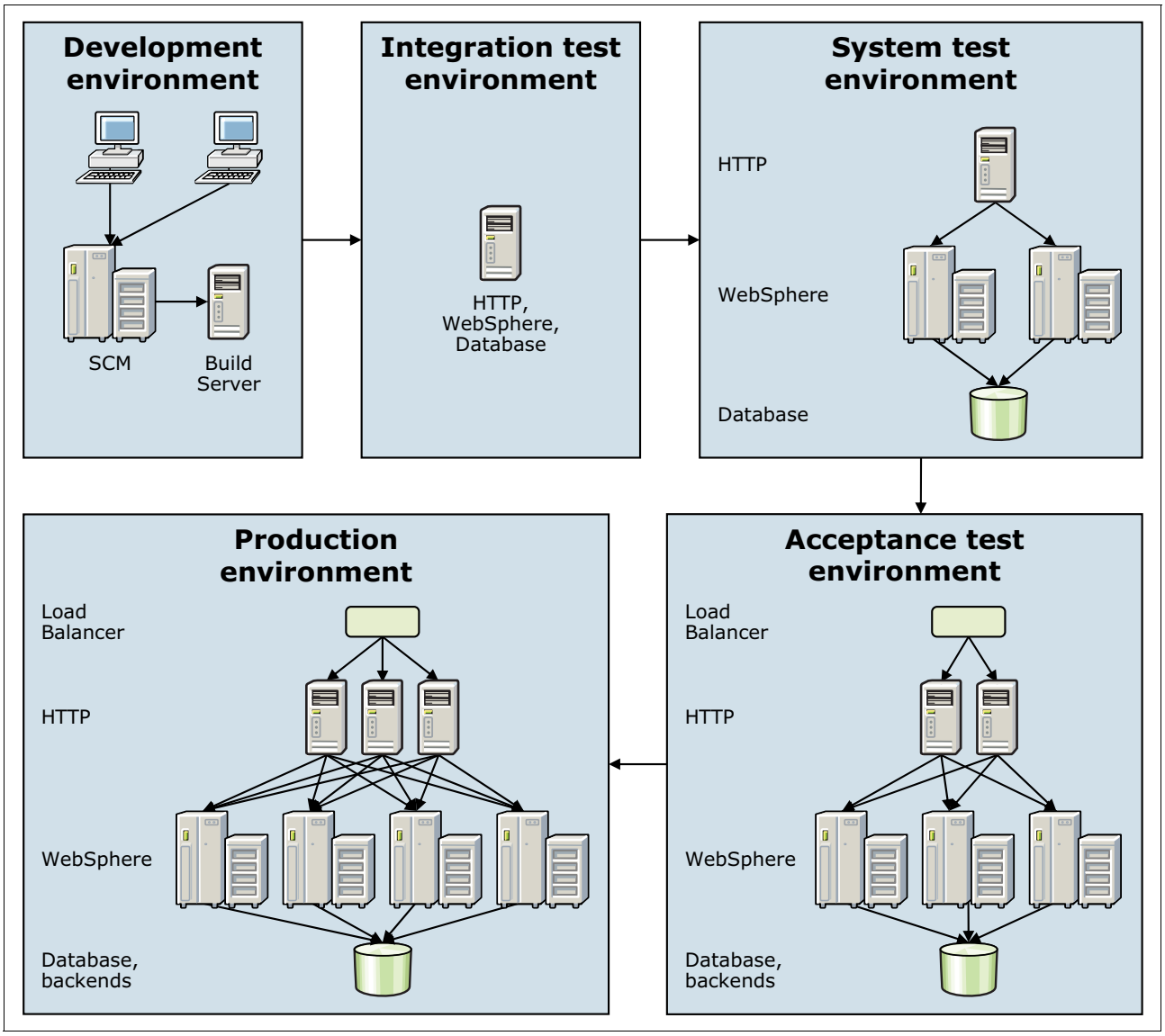


Figure 11-6 Test environments

Test cases must be developed according to system specification and use cases created before the application is developed. System specification and use cases must be detailed enough so that test cases can be developed. Test cases need to verify both functional requirements (such as application business logic and user interface) and nonfunctional requirements (such as performance or capacity requirements). After you create the test cases, and with sufficient developed functions in the application, start testing.

This section provides information about test environments, not servers. Depending on your organization size and business needs, you can have more than one environment on a physical server. The important point is to have a clear idea of the purpose of each environment, more than the topology or physical distribution of those environments.

Whether you choose to use some of these test environments, all of them, or additional test environments depends on the following factors:

- The system that is being developed
- The project size
- The budget constraints

Each environment is maintained as a separate cell to completely isolate the environments from each other. For smaller environments, a single application server profile is usually sufficient, whereas larger environments might need a deployment manager for a particular cell environment.

11.10.1 Development environment

Usually, developers have their own WebSphere test environment integrated in the development tool. This test environment is used for the daily work of a developer and is often active while the developer is coding. Whenever necessary, the developer can perform instant testing.

Because of the tight integration between WebSphere Application Server and the IBM development tools, the application server can run the application by using the resources in the workspace of the developer. This integration eliminates the need for developers to perform these steps for every small change:

1. Run build scripts
2. Export or otherwise package the application into an EAR file
3. Deploy that file on a test server

This capability makes it easy and quick to test applications while developing them, increasing developer productivity.

Developers are also responsible for performing unit testing of their own code. Most tests performed for the system are run in this environment. The primary goal is to remove obvious code bugs. The developers work against, and share code by using, the source code management system. The development environment is most often a powerful desktop system.

When developers commit their code to the integration stream in the source code management system, a development lead or integration team usually performs a clean build of the whole application. This build brings together code developed by different developers. This process is usually done on a special build server, and is controlled by automatic build scripts. For more information, see 11.7, “Automated build process” on page 366. This server might need a copy of the IBM Assembly and Deploy Tool or Rational Application Developer for WebSphere Software V8.5 installed.

The development team can also create a Build Verification Test process as described at:

http://www.ibm.com/software/awdtools/tester/functional/

With this process, each new build is run before making the build available to the team. A Build Verification Test covers test cases or scenarios that verify that critical paths through the code are operational.

Build Verification Test scripts are often controlled by JUnit. JUnit is a testing framework for the Java programming language that allows the development of repeatable test cases. For more information, see the JUnit website at:

http://www.junit.org/

Every developer is responsible for performing basic code profiling. By using the profiling tools in Rational Application Developer for WebSphere Software V8.5, a developer can discover methods that run poorly and find memory leaks or excessive creation of objects. Optionally, developers can also use other tools to profile the applications they develop if Rational Application Developer is not their development tool.

An alternative is IBM Monitoring and Diagnostic Tools for Java - Health Center. This open source tool is delivered in the IBM Support Assistant Workbench, which is also available at no additional cost. For more information about this tool, see the following developerWorks topic:

`http://www.ibm.com/developerworks/java/jdk/tools/healthcenter/`

11.10.2 Integration test environment

After a successful build and regression test, the application is deployed to the integration test environment. This environment is where the developers perform integration tests that include all system components. These tests are performed on a hardware and software platform that mirrors the production environment, although on a smaller scale. Before the integration tests, the only tests performed to the application are the unit tests made by the developers on their environment. Test the application only when it is free of explicit code issues, such as syntax or compilation errors.

Because the production environment is often not the same as the development environment, start testing on the target platform as early as possible. The integration environment can include the following tests:

- Access to the application by using the WebSphere plug-in on the web server
- Division between static content served by the web server and dynamic content served by the application server
- Incompatibilities between platforms (for example, hard-coded folder paths such as `C:\` versus `/usr`)
- Configurations on the WebSphere Application Server to access data (such as databases or service integration buses)
- Integration with directory services

The integration test environment is usually the first environment that is suitable for these types of tests.

For small projects, the integration test environment can often be shared between different projects. However, if the number of projects or developers is too large, the environment becomes difficult to manage. Avoid having more than 5–10 developers share a single integration test environment. If a developer needs to perform tests that might damage the environment, use a dedicated environment. If the system has enough resources in terms of processor and memory, consider using multiple WebSphere profiles to isolate different teams from each other. Using VMware virtualization is another option. The development team manages and controls the integration test environment.

11.10.3 System test environment

The purpose of the system test is to verify that the system meets both functional and non-functional requirements. After the development team tests the application in its controlled environment, the application is delivered to the system test team. When the application is delivered, the system test team deploys it by using the instructions given.

The system test team is responsible for verifying all aspects of the system, and ensuring that it conforms to the specifications.

This environment can include the following tests:

- Correct execution of business rules and logic
- Graphical interface evaluation
- Correct error handling
- Security access according to defined users and roles
- Security certificates and Secure Sockets Layer (SSL) configurations
- Correct load balancing across the servers in the cluster
- Failover of high available components
- Accurate installation and configuration instructions

The system test team completely controls the system test environment. The environment is usually a scaled down version of the real production environment, but with all of the important components in place.

The system test environment can also be used by other teams. For example, system administrators might need to test new patch levels for the operating system, WebSphere Application Server, and database, before rolling them out in production. In this case, they can use the system test environment to complete that task. If a patch is committed, ensure that it is applied to the other test environments to keep all environments synchronized.

11.10.4 Acceptance test environment

The acceptance test environment is the last stage where testing occurs before moving the application into production. The acceptance test environment is the one that most closely resembles the actual production environment. Hardware and software must be identical to the production environment.

Because of cost constraints, it is often not possible to have an acceptance test environment with identical capacity as the production environment. The acceptance test environment is, therefore, usually smaller than the production environment. However, it needs to contain the same components, brands, software patch levels, and configuration settings as the production environment.

The purpose of the acceptance test environment is to give the operations team a chance to familiarize themselves with the application and its procedures. It also provides an opportunity to test unrelated applications together, because previous environments focused on testing the applications independently of each other. This test is important because it helps to determine whether the server resources are enough to handle the expected workload for all of the deployed applications.

Because the acceptance test environment is almost identical to the production environment, this environment is the correct place to test the following aspects:

- Installation and configuration procedures
- Backup procedures
- Failover procedures
- Load tests (measures system behavior under expected load)
- Stress tests (measures system behavior under higher than expected load)
- Performance
- Session persistence

Typically, projects have successful performance tests where the results meet the requirements. Then, when the application is moved into production, the performance is poor.

When running performance tests, keep in mind the following considerations:

- Populate the database with the most similar production data as possible. Keep the same database structure, stored procedures, and volume of data if possible.
- If HTTP session persistence is enabled in production, enable it during the performance tests.
- If more than one application is running on the same production server, run them in the acceptance test environment.
- Try to replicate networking configurations on the acceptance environment such as firewalls, intrusion detection policies, access lists, and routing configurations.
- If running on Windows platforms, remember to configure antivirus software scanning policies to avoid scanning critical files, such as log files, that can affect server performance.

11.11 Managing application configuration settings

Almost all non-trivial applications require at least some amount of configuration to their environment to run optimally. Part of this configuration (such as references to EJB and data sources) is stored in the application deployment descriptors. It is modified by developers by using tools such as IBM WebSphere Developer Tool or Rational Application Developer for WebSphere Software V8.5. Other settings, such as the JVM maximum heap size and database connection pool size, are stored in the WebSphere Application Server configuration repository. These settings are modified by using the WebSphere administrative tools.

Finally, settings that are application-internal are usually created by the developers and are stored in Java property files. These files are then modified, usually by using a plain text editor, by the system administrators after deploying the application.

11.11.1 Classifying configuration settings

Configuration data can often fit into the following categories:

- Application-specific

  This category includes configuration options that are specific for an application regardless of its deployment environment. Examples include the number of hits to display per page for a search result and the EJB transaction timeout. The timeout option is needed if the application has long-running transactions. This category must move, unchanged, with the application between the different environments.

- Application environment-specific

  This category includes configuration options that are specific to an application and its deployment environment. Examples include log detail levels, cache size, and JVM maximum heap size.

  For example, in development, you might want to run the OrderApplication with debug-level logging, but in production, you want to run it with only warning-level logging. During development, the OrderApplication might work with a 256 MB heap. However, in the busier production environment, it might need a 1 GB heap size to run well. Do not move these options with the application between environments. They need to be tuned for the specific environment.

- Environment-specific

  This category includes configuration options that are specific to a deployment environment but that are common to all applications that run in that environment. This category includes, for example, the name of the `temp` folder if applications need to store temporary information. In the Windows development environment, this name might be `C:\temp`, but in a UNIX production environment, it might be `/tmp`. This category of options must not move between environments.

11.11.2 Managing the configuration settings

Managing the configuration settings is usually a major challenge for developers and system administrators. You might need to change configuration settings when the application is moved from one deployment environment to another. You must ensure that the settings are also synchronized among all application instances if running in a clustered environment.

You can manage the settings stored in the WebSphere configuration repository (such as the JVM maximum heap size). You can also develop scripts that run as part of an automatic deployment to configure the settings after the application is deployed. The values suitable for the application can be stored in a bill of materials file inside the EAR file. This file can then be read by scripts and used to configure the environment.

Settings stored in the deployment descriptors usually do not have to be changed when the application is moved between different environments. Instead, the Java EE specification separates the work of the developers from the work of the deployers. During deployment, the resources specified in the deployment descriptors are mapped to the corresponding resources for the environment. For example, a data source reference is mapped to a JNDI entry, which points to a physical database. Therefore, develop the applications by taking advantage of the configuration flexibility that the application server offers. Avoid using hardcoded connections to back-end systems, such as embedded direct Java Database Connectivity (JDBC).

However, application-internal configuration settings are often stored in Java property files. These files are plain text files with key-value pairs. Java provides support for reading and making them available to the application by using the `java.util.Properties` class since Java 1.0. You can use databases, Lightweight Directory Access Protocol (LDAP), JNDI, and so on, to store settings. However, plain Java property files are still the most common way to configure internal settings for Java applications. This method is an easy and straightforward way to accomplish this task.

You might want to protect sensitive information, such as passwords or IP addresses, that is stored in property files. In this case, use the **`PropFilePasswordEncoder`** utility, provided by WebSphere Application Server, to encode such information. Remember that encoding is not the same as encryption. Therefore, it is not enough to fully protect passwords.

Also consider whether to store sensitive information in property files. The **`PropFilePasswordEncoder`** is in the `profile_root/bin` path. Consider the simple property file in Example 11-1.

Example 11-1 Property file

```
userId=myUser
userPassword=myPassword
```

To encode the value of `userPassword`, use the following command:

```
PropFilePasswordEncoder path_to_property_file\myPropFile.props userPassword
```

Exception: The **PropFilePasswordEncoder** utility does not encode passwords that are in XML or XMI files.

For more information about the **PropFilePasswordEncoder**, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=rsec_propfilepwdencoder

In a clustered environment where the same application runs on multiple servers distributed across different systems, use care in determining how to package, distribute, and access the property files.

For packaging the property files, you have two approaches. You include the property files within the EAR file itself, or you distribute the files separately. To include the property files within an EAR file, the easiest approach is to create a utility JAR project. Next, add the property files to it, and then add that project as a dependent project to the projects that need to read the property files. The utility JAR project is then made available on the class path for the other projects.

However, a better approach is to centralize access to the property files by using a custom property manager class. This way, access to the properties is not scattered all over your code. For example, to load a property file by using the class loader, you can use the code snippet in Example 11-2.

Example 11-2 Loading a property file by using the class loader code snippet

```
Properties props = new Properties();
InputStream in =
MyClass.class.getClassLoader().getResourceAsStream(“my.properties”);
props.load(in);
in.close();
```

Package property files packaged in a JAR file in the EAR file for property files that cannot be modified after the application is deployed. The application-specific category is explained in 11.11.1, “Classifying configuration settings” on page 374.

If you want to make the property files easily accessible after the application is deployed, store them in a folder outside the EAR file. To load the property files, make the folder available on the class path for the application. Use the code snippet in Example 11-2. Alternatively, you can use an absolute path name and the code snippet in Example 11-3. In this example, the file to load is the `/opt/apps/OrderApp/my.properties` file.

Example 11-3 Absolute path name code snippet

```
Properties props = new Properties();
InputStream in = new FileInputStream(“/opt/apps/OrderApp/my.properties”);
props.load(in);
in.close();
```

Avoid using absolute path names because it tends to hard code strings into your code. Make the folder with the property files available on the class path for the application by defining a shared library to WebSphere Application Server. Instead of specifying JAR files, specify the name of the folder that holds the property files in the Classpath field for the shared library. For example, you might use the `/opt/apps/OrderApp` folder.

A lesser known, but better, approach to access property files is to use URL resources. Although this approach is not explained in detail here, the following steps describe it:

1. Create a folder on your system that holds the property file.
2. Using the administrative console, create a URL resource that points to the property file, and assign it a JNDI name.
3. In the application, create a URL resource reference binding that points to the chosen JNDI name.
4. In Java, use JNDI to look up the URL resource reference. Create an InputStream from the URL, and use that InputStream as input to the `java.util.Properties` class to load the property files.

This approach to access property files is also more compliant with Java EE. It does not rely on the `java.io` package for file access, which is prohibited by the Java EE specification. This method also gives you the opportunity to load the property files by using HTTP and FTP. This configuration allows you to set up an HTTP server that serves properties files from a central location.

Unless you are using the previous technique with the HTTP or FTP protocol, manage all property files in a central location on the deployment manager. However, property files that are stored in folders outside the EAR files are not propagated to the WebSphere nodes. The exception is folders that are created under the deployment manager cell configuration folder, which is *dmgr_profile_home*\config\cells*cell_name*.

By creating a folder under this folder, you can take advantage of the WebSphere file transfer service to propagate your files to the nodes. Because this folder is not known to the WebSphere Application Server infrastructure, the transfer does not happen automatically when the contents are changed. You need to force a synchronization with the nodes. This synchronization propagates the property files to the *profile_home*\config\cells*cell_name*\appconfig directory on each node. You can include that folder on the class path by using a shared library or pointing your URL resources to it.

Tip: When deciding on names for settings in property files, consider including the unit of the setting that is referred to in the name. Instead of using MaxMemory or Timeout, use MaxMemoryMB and TimeoutMS to indicate that the max memory be given as MB and the timeout as MS. This method can help reduce confusion for the system administrator who does not know the internal functions of the application.

Storing property files that need to be changed between different environments inside the EAR file can cause problems, especially in a clustered environment.

In a clustered environment when an enterprise application is deployed to WebSphere Application Server, it is distributed to each node in the cluster by using the WebSphere Application Server file transfer mechanism. At each node, the EAR file is expanded and laid out on the file system so that WebSphere Application Server can run it. A property file included in the EAR file is automatically replicated to each member of the cluster.

If you then need to change the property file, you must do it manually on each cluster member, which can be error prone. Alternatively, you can do it on the deployment manager and then distribute the updated file to each node again. However, WebSphere Application Server does not fully expand the contents of the EAR file to the file system on the deployment manager. It extracts from the EAR file only the deployment descriptors that are needed to configure the application in the WebSphere Application Server cell repository. Therefore, the property file is not readily accessible on the deployment manager.

As a result, you must manually unpack the EAR file, extract the property file, and modify it. Then you must re-create the EAR file again and redeploy the application. This approach results in complicated administration and limits flexibility. Therefore, be careful if you plan to package the properties file inside the EAR file.

An alternative when distributing the property files within the EAR file is to extract them from the EAR file after deployment. You then place them in a folder separate from the EAR file. An example of a folder name suitable for that is the *dmgr_profile_home*\config\cells*cell_name*\configData folder on the deployment manager system. Anything in that folder is replicated to each node in the cell when WebSphere Application Server synchronizes with the nodes. For the application to find the file, it must then refer to it on its local file system. However, because that folder name includes both the name of the profile and the name of the cell, it can quickly become messy. Depending on your environment, this approach can also be unfeasible.

11.12 Planning for application upgrades in production

When planning upgrades in production, consider the following questions:

- How is your application server topology designed?
- How flexible is your application design from the upgrade point of view?

You must consider several different aspects when planning the correct topology to minimize the outage of the applications during upgrades. Plan how to make your application server processes highly available in case the application upgrades need a server restart. Hide this process from the user. The topology selection criteria is provided in 8.2, “Topology selection criteria” on page 184.

Another important aspect when planning upgrades in production is how the application is developed. Naturally, the main actors here are the developers. Even though they might not always realize it, developers play a critical role in making the production environment stable and highly available. If an application is poorly written or developers introduce incompatible changes, you might be forced to bring down the whole system for an application upgrade.

Developers must consider the following areas when planning for new versions:

- Database schema compatibility

  If a change in database layout is introduced, you might have to shut down all instances of an application to migrate the database to the new layout and update the application. You might have to shut down multiple applications if they all use the same database. One way is to migrate a copy of the database to the new layout. You can then install the new applications on a new WebSphere cluster, and then switch to the new environment. In this case, all transactions committed to the hot database must be reapplied to the copy, which is the hot database again when switching back.

- EJB version compatibility

  If EJB interfaces do not maintain compatibility with earlier versions, and the application has stand-alone Java clients, you might have to distribute new versions of the Java clients. You might have to distribute new versions when the EJB clients are servlets, but they are not deployed as part of the same EAR file as the EJB. It might also be needed when these servlets are running in a container separate from the EJB. In this case, you might have to set up special EJB bindings. These bindings must allow version 1 clients to continue to use the version 1 EJB, whereas the version 2 clients use the new version 2 EJB.

- Compatibility of objects in HTTP session

  You might take a simple, straightforward approach and use the WebSphere Application Server rollout update feature, in which case, you also enable HTTP session persistence.

In this case, make sure that the objects stored in the HTTP session are compatible between the two application releases.

Consider a case where a user is logged on and has a session on one application server. That server is shut down for its application to be upgraded. The user is moved to another server in the cluster and the user's session is restored from the in-memory replica or from a database. When the first server is upgraded, the second server is shut down. The user is then moved back to the first server again. If the version 1 objects in the HTTP session in memory are not compatible with the version 2 application, the application might fail.

- User interface compatibility

  If a user is using the application and it suddenly changes the way it looks, the user might become frustrated. Users might require training to learn a new user interface or navigation system.

For more information about keeping the applications available during an update, see the following developerWorks topic:

http://www.ibm.com/developerworks/websphere/techjournal/0412_vansickel/0412_vansickel.html

This topic addresses this topic from the development and infrastructure perspective, and provides more detailed information and considerations.

11.13 Mapping applications to application servers

Two approaches are possible when deploying applications: Deploy each application in its own application server or deploy all applications in the same server or cluster. The correct choice depends on your environment and on the application needs. Table 11-2 compares both options from several perspectives.

Table 11-2 Deployment options comparison

Options	Applications per application server	
	One application	**Multiple applications**
Applications availability	If one server fails, one application fails, unless deployed to a cluster.	If one server fails, all the applications fail, unless deployed to a cluster.
Memory footprint	Around 130 MB of RAM per application server for its own processes.	Less memory footprint, because fewer application servers are needed.
Application configuration	Customized for each application: Heap size, log files, environment settings, EJB timeouts.	Configurations per server apply to all the deployed applications.
EJB calls	Remote calls if EJB modules also have their own application server. Can affect performance.	Local calls from one application to the other.
Security	More ports need must be opened in the firewall between the web server and application server.	Fewer opened ports in the firewall between the web server and application server.

Combining both options can also be the best approach for your environment. Decide whether deploying critical applications to one application server or cluster gives you the benefit of avoiding other faulty applications from interrupting their service. Other, less critical applications share the application server or cluster. For application deployment considerations that can help when planning how to accomplish this task, see 8.2.8, "Application deployment" on page 193.

11.14 Planning checklist for applications

As you plan, keep in mind the following checklist:

- Select the appropriate set of application design and development tools.
- Create a naming convention for applications and application resources.
- Implement a source code management system and a collaboration system if applicable.
- Design an end-to-end test environment.
- Create a strategy for maintaining and distributing application configuration data.
- Create a strategy for application maintenance.
- Determine where applications will be deployed (for example, all on one server).

11.15 Resources

For more information, see the Websphere Application Server V8.5 Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp

For more information about application development by using Rational Application Developer, see *Rational Application Developer for WebSphere Software V8 Programming Guide*, SG24-7835.

12

System management

This chapter provides an overview of the planning necessary for the system management of a WebSphere Application Server runtime environment. It focuses on developing a strategy to optimally use the multitude of system management capabilities in WebSphere Application Server. The operational efficiency of the overall system hinges on the correct implementation of the system management processes.

This chapter includes the following sections:

- System management features in WebSphere Application Server V8.5
- Administrative security
- Administration facilities of WebSphere Application Server
- Automation planning
- Configuration planning
- Repository checkpoints service
- Change management
- Serviceability
- Cross-component trace
- Planning checklist for system management

12.1 System management features in WebSphere Application Server V8.5

WebSphere Application Server V8.5 provides the following administrative tools and processes:

- Repository checkpoints service

  Repository checkpoints service enables the ability to track changes made to the application server configuration through the repository checkpoints service. For more information, see 12.6, “Repository checkpoints service” on page 398”.

- Cross Component Trace

  Cross Component Trace helps identify the root cause of problems across components with minimal cost. For more information, see 12.9, “Cross-component trace” on page 412”.

- Centralized installation manager

  Installs and applies maintenance on remote targets, and is available from the job manager. The following features are also offered:

  – Job scheduling
  – Removal of cell boundary limitations
  – Support for z/OS targets
  – Better scaling through the use of the Installation Manager
  – Support for Liberty profile installation management

  For more information, see 12.7.3, “Centralized installation manager” on page 402.

- High Performance Extensible Logging (HPEL)

  HPEL provides a convenient mechanism for storing and accessing log, trace, `System.err`, and `System.out` file information produced by the application server or the applications. However, HPEL does not replace the existing basic log and trace facility. HPEL provides greater flexibility for administrators to manage logging resources, and is easier to use than the basic logging and trace facility.

  HPEL includes the following benefits:

  – Log, trace, `System.err`, and `System.out` file information stored in collective repositories
  – Less impact on performance than basic logging
  – Better administration of resources used to collect and retain logging information
  – Enhanced capabilities to work with the logging and trace content

  WebSphere Application V8.5 adds the following enhancements for HPEL:

  – HPEL log/trace entries can now be extended with name-value pair extensions
  – Entries can be filtered by appName, requestID, or any other extension by using the HPEL logViewer command’

  For more information, see 12.8.1, “Log and traces” on page 405.

- Node management

  Nodes can be recovered or moved by using the **`-asExistingNode`** option with the **`addNode`** command. For more information, see “The addNode -asExistingNode command” on page 387.

- Properties file-based configuration

  With the portable format of the properties file, you can apply property files across multiple environments. Modifying environment-specific variables makes a properties file portable. The **`wsadmin`** tool allows you to extract a properties file from one cell, modify

environment-specific variables at the bottom of the extracted properties file, and then apply the modified properties file to another cell. The tool does not allow you to replicate a cell. It allows you to replicate a small part of the configuration at a time.

For more information, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=rxml_prop_file_syntax

- The **managesdk** command

  The **managesdk** command-line tool and associated **wsadmin** commands are used to manage the software development kits (SDKs) that are available to a WebSphere Application Server installation. The **managesdk** command provides a common API for all WebSphere Application Server platforms. For more information, see “The managesdk command” on page 389.

- Monitored directory deployment

  Monitored directory deployment updates and deploys applications automatically by using one of the following methods:

  – Adding enterprise application files to a monitored directory
  – Adding enterprise application files by adding properties files to a monitored directory

  For more information, see 12.3.8, “Monitored directory deployment” on page 392.

- Job manager

  Job manager actions are available from both a deployment manager and a job manager. The **jobs** link on the administrative console provides access to the following job manager options:

  – Submit a job
  – Review the status of a job
  – Manage job manager targets for jobs
  – Identify target resources that are used in jobs
  – Manage target groups for administrative jobs

  For more information, see 12.3.7, “Job manager” on page 391.

12.2 Administrative security

Enabling administrative security prevents unauthorized access to the administrative tasks. It secures only administration tasks, not applications.

After administrative security is enabled, a security check is run when the administrative console or other administrative facilities are accessed. The security check ensures that the accessing user is authenticated and mapped to one of the console security roles. Depending on the console role to which the user is mapped, different functions are available.

Effective planning for system management includes identifying the people who need access and their level of access to the administrative tools. Groups can be designed and preset for users and roles according to organizational needs.

For more information about the available roles and access levels, see the WebSphere Application Server V8.5 Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp

WebSphere Application Server offers the option to enable administrative security during profile creation. If this option is chosen during profile creation, a prompt to provide a user ID and password is displayed. The user ID and password are stored in a set of XML files and are mapped to the administrator role. Additional users can be added after profile creation by using the administrative tools.

12.3 Administration facilities of WebSphere Application Server

WebSphere Application Server V8.5 provides these administrative tools to configure and manage your runtime environment:

- The administrative console

  The administrative console is a browser-based client that uses a web application that runs in the web container to administer WebSphere Application Server.

- WebSphere scripting client (**wsadmin**)

  The **wsadmin** client is a non-graphical scripting interface that administers WebSphere Application Server from a command-line prompt. It can connect to WebSphere Application Server by using one of the two communication mechanisms:

  - SOAP by communicating with the embedded HTTP server in the web container
  - Remote Method Invocation (RMI) to communicate with the administrative services

- Task automation with Apache Ant

  Apache Ant is used to create build scripts that compile, package, install, and test applications on WebSphere Application Server.

- Administrative programming

  You can develop custom Java applications that use the Java Management Extensions (JMX) based on the WebSphere application programming interface (API).

- Command-line utilities

  WebSphere Application Server provides administrative utilities to help manage your environment. It includes the following features:

  - Called from a command line
  - Can be used to perform common administrative tasks such as starting and stopping WebSphere Application Server and backing up the configuration
  - Work on local servers and nodes only, including the deployment manager

The combination of administrative tools that you employ ultimately depends on the size and complexity of your runtime environment. If you have few resources but many tasks, consider using automation and scripts. If you have multiple administrators who perform different tasks, consider defining different access control roles. The use of different access control roles is important where you want non-administrators to perform limited roles such as application deployment.

Updates to configuration through the administrative console or the **wsadmin** client are kept in a private temporary area called a *workspace*. The changes are not copied to the configuration repository until an explicit save command is issued. The workspace is in the *profile_root*\wstemp directory. By using a workspace, multiple clients can access the configuration concurrently. Use care to prevent change conflicts. Clients can detect such conflicts and allow you to handle them. For example, the **wsadmin** client has a property called setSaveMode that can be set to control the default save behavior if a conflict occurs.

12.3.1 The administrative console

The administrative console connects to a running stand-alone server or, in a distributed environment, to a deployment manager. In WebSphere Application Server V8.5, it also connects to an administrative agent and a job manager.

Non-secure administration access

If administrative security is not enabled, the administrative console is accessed with a web browser through the following URL:

`http://<host>:<WC_adminhost port>/ibm/console`

You can gain access to the console without entering a user name. If you do enter a name, it is not validated. It is used exclusively for logging purposes and to enable the system to recover the session if it is lost while running administrative tasks.

Secure administration access

If administrative security is enabled, the administrative console is accessed with a web browser through the following URL:

`https://hostname:WC_admin_secure port/ibm/console/Logon.jsp`

Tip: Notice the use of `https://` versus `http://`. You must enter an authorized user ID and password to log in. The actions that you can perform within the console are determined by your role assignment.

12.3.2 WebSphere scripting client (wsadmin)

With the WebSphere scripting client (**wsadmin**), you can run scripts. You can use the **wsadmin** tool to manage a WebSphere Application Server V8.5 installation and configuration. This tool uses the Bean Scripting Framework (BSF), which supports several scripting languages to configure and control your WebSphere Application Server installation.

The **wsadmin** launcher makes Java objects available through language-specific interfaces. Scripts use these objects for application management, configuration, operational control, and for communication with Managed Beans (also referred to as *MBeans*) running in WebSphere server processes.

You can run the **wsadmin** tool in interactive and unattended mode. Use the **wsadmin** tool to perform the same tasks that you perform with the administrative console.

WebSphere Application Server provides command assistance in the administrative console that maps your administrative activities to **wsadmin** scripting commands written in Jython. You can view these commands from the administrative console, and can log the command assistance data to a file. You can also allow command assistance to emit JMX notifications to IBM Assembly and Deploy Tools for WebSphere Administration. These tools include Jython development tools that help you develop and test Jython scripts.

Consideration: The stabilized process for Java TCL (Jacl) syntax that is associated with **wsadmin** has been in place since the release of WebSphere Application Server V7. WebSphere Application Server supports Jacl syntax for **wsadmin**, and there is no plan to deprecate or remove this capability in a subsequent release of the product. However, future investment will be focused on Jython.

12.3.3 Task automation with Ant

WebSphere Application Server V8.5 provides a copy of the Ant tool and a set of Ant tasks that extend the capabilities of Ant to include product-specific functions. Ant has become a popular tool among Java programmers.

Apache Ant is a platform-independent, Java language-based build automation tool. It is configurable through XML script files and extensible through the use of a Java API. In addition to the base Ant program and tasks, WebSphere Application Server provides several tasks that are specific to managing and building applications in WebSphere Application Server.

In the Ant environment, you can create platform-independent scripts that compile, package, install, and test your application on WebSphere Application Server. It integrates with **wsadmin** scripts and uses Ant as their invocation mechanism.

For information about Apache Ant, see:

http://ant.apache.org

12.3.4 Administrative programming

WebSphere Application Server V8.5 supports access to the administrative functions through a set of Java classes and methods. You can write a Java application that runs any of the administrative features of the WebSphere Application Server administrative tools. You can also extend the basic WebSphere Application Server administrative system to include your own managed resources.

JMX is a Java specification part of Java Platform, Enterprise Edition (Java EE). It and the specification for the Java EE Management API (JSR-077) are the core of the management architecture for WebSphere Application Server. For information about JMX, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=cxml_javamanagementx

You can prepare, install, uninstall, edit, and update applications through programming. Preparing an application for installation involves collecting various types of WebSphere Application Server technology-specific binding information to resolve references that are defined in the application deployment descriptors. This information can also be modified after installation by editing a deployed application. Updating consists of adding, removing, or replacing a single file or a single module in an installed application. It can also consist of supplying a partial application that manipulates an arbitrary set of files and modules in the deployed application. Updating the entire application uninstalls the old application and installs the new one. Uninstalling an application removes it entirely from the WebSphere Application Server configuration.

12.3.5 Command-line tools

With command-line tools, you can perform management tasks that include starting, stopping, and checking the status of WebSphere Application Server processes and nodes. These tools work only on local servers and nodes. They cannot operate on a remote server or node. To administer a remote server, use the administrative console or a **wsadmin** script. The script must connect to the deployment manager for the cell in which the target server or node is configured.

All command-line tools function relative to a particular profile. If you run a command from the *was_home*/WebSphere/AppServer/bin directory, the command runs within the default profile when no profile option is specified.

WebSphere Application Server includes the following command-line tools:

- **addNode -asExistingNode**
- **managesdk**

The addNode -asExistingNode command

You can recover or move nodes by using the **-asExistingNode** option with the **addNode** command.

You can recover a damaged node as illustrated in Figure 12-1. You can use the **-asExistingNode** option of the **addNode** command to recover nodes of a deployment manager. By using the **-asExistingNode** option, you federate a new custom node to a deployment manager as an existing node. During federation, the product uses information in the master configuration of the deployment manager to transform the custom node into the existing node.

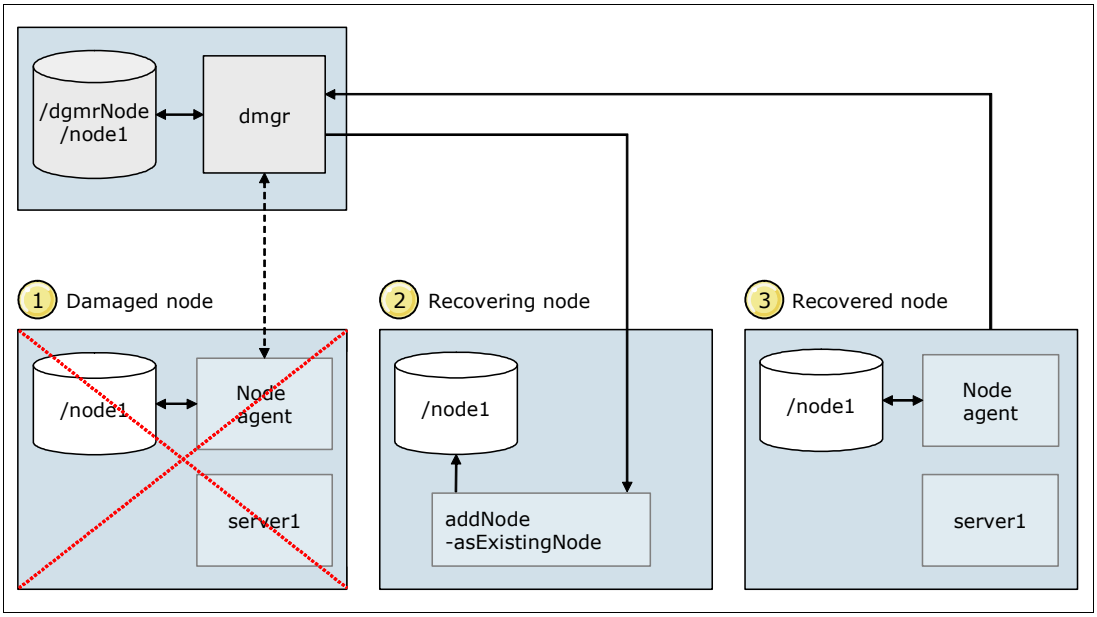


Figure 12-1 Recovering a damaged node

You can move a node to an installation of WebSphere Application Server on a different computer, with the same path or with a different path, as shown in Figure 12-2.

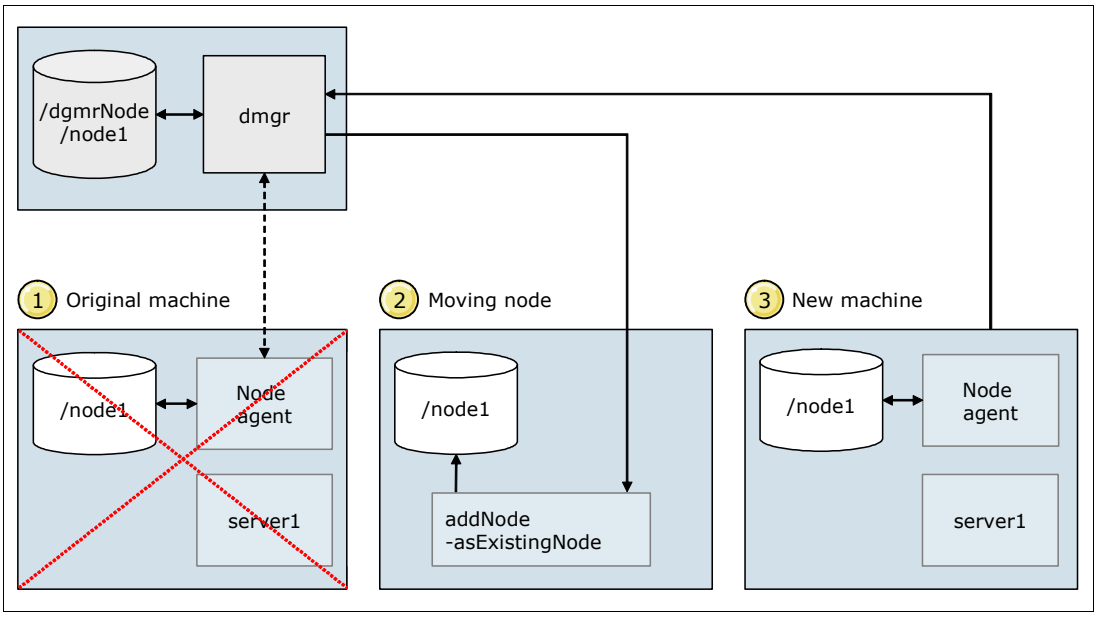


Figure 12-2 Moving a node

You can create a cell from a template cell as shown in Figure 12-3 on page 389 and configure it by using the following steps:

1. Run the **backupConfig** command to create a `template.zip` file of the configuration files.
2. For every new environment, install WebSphere application server.
3. Create deployment manager and node profiles.
4. Run the **restoreConfig** command to restore the configuration.
5. Customize the configuration of the deployment manager.
6. Run the following command:

   ```
   addnode -asExistingNode
   ```

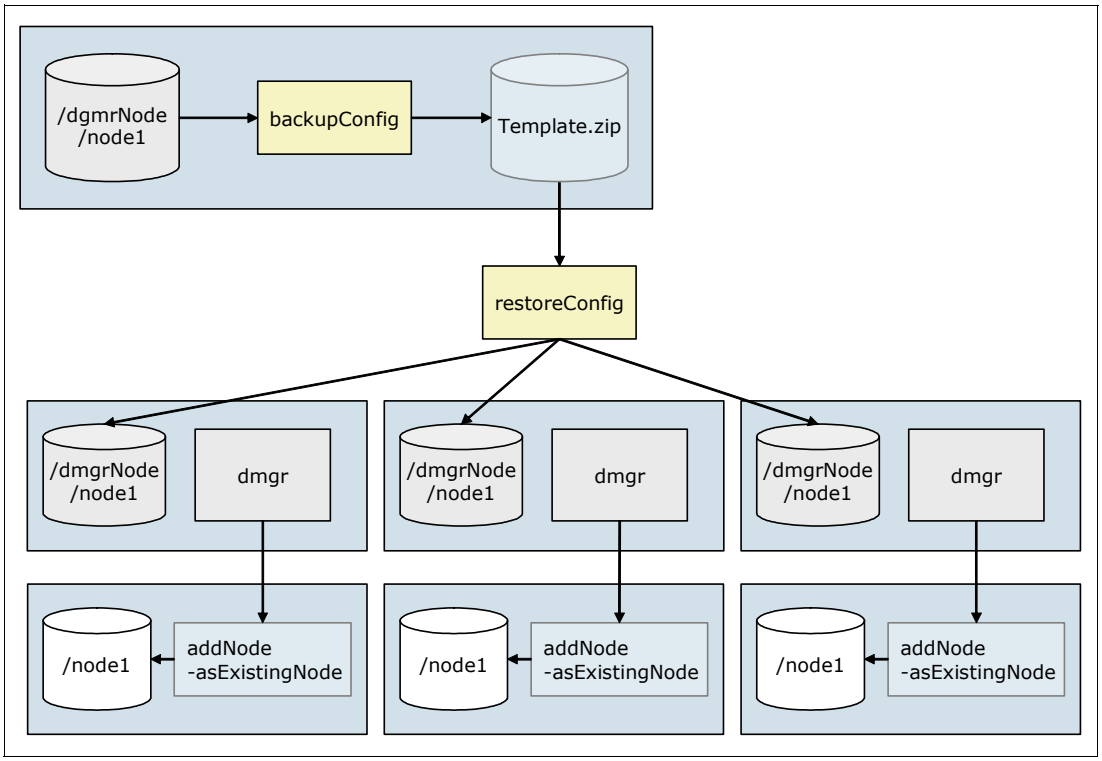


Figure 12-3 Creating cells from a template

For more information about the **addNode -asExistingNode** command, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tagt_addNode_asExistingNode

The managesdk command

WebSphere Application Server provides the **managesdk** command to manage the SDKs that are available to an installation of WebSphere Application Server. The **managesdk** command provides a common API for all WebSphere Application Server platforms. You can use the **managesdk** command to perform the following tasks:

- List the SDK names that are available to a product installation.
- List the SDK names that a specified profile or all profiles in an installation are currently configured to use.
- Enable a specific profile or all profiles in an installation to use a specified SDK name.
- Get the SDK name that is used to configure new profiles.
- Change the default SDK name that profiles use.
- Get the SDK name that is used by scripts called from the product *bin* directory.
- Change the SDK name that scripts in a product *bin* directory use by default.

The **managesdk** command also introduces the following SDK terminology, which is compatible with the existing WebSphere Application Server infrastructure:

Node default SDK	The default SDK for application servers on the node, as defined by node level JAVA_HOME variable map.
Server SDK	The SDK used by the application server. The default is the node default SDK. Each application can override the default by using the server level JAVA_HOME variable map.

Restriction: In WebSphere Application Server V7, V8 and V8.5, only IBM i and z/OS platforms support multiple SDKs. The **enablejvm** command used on IBM i platforms is deprecated since V8.

For more information about the **managesdk** command-line tool and associated scripting APIs, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=rxml_managesdk

12.3.6 Administrative agent

The administrative agent provides a single administration interface for multiple unfederated instances of WebSphere Application Server in the same physical server. Providing administrative agent capabilities involves creating an administrative agent profile and registering the node you want the administrative agent to manage by using the **registerNode** command. A **deregisterNode** command is available to undo the use of the administrative agent.

Non-secure administration access

If administrative security is not enabled, the administrative console is accessed with a web browser through the following URL:

`http://`*`hostname`*`:`*`WC_adminhost`*`/ibm/console/profileSelection.jsp`

Select a node that you want to manage. You can gain access to the console without entering a user name. If you enter a name, it is not validated and is used exclusively for logging purposes. It is also used to enable the system to recover a lost session while running administrative tasks.

Secure administration access

If administrative security is enabled, the administrative console is accessed with a web browser through the following URL:

`https://`*`hostname`*`:`*`WC_adminhost_secure`*`/ibm/console/profileSelection.jsp`

Reminder: Notice the use of `https://` versus `http://`. Select a node that you want to manage. You must enter an authorized user ID and password to log in. The actions that you can perform within the console are determined by your role assignment.

For more information, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=cagt_adminagent

12.3.7 Job manager

In a flexible management environment (Figure 12-4), the job manager allows the management of multiple WebSphere Application Server domains (multiple deployment managers and administrative agents) through a single administration interface. The Job manager supports Liberty profile management.

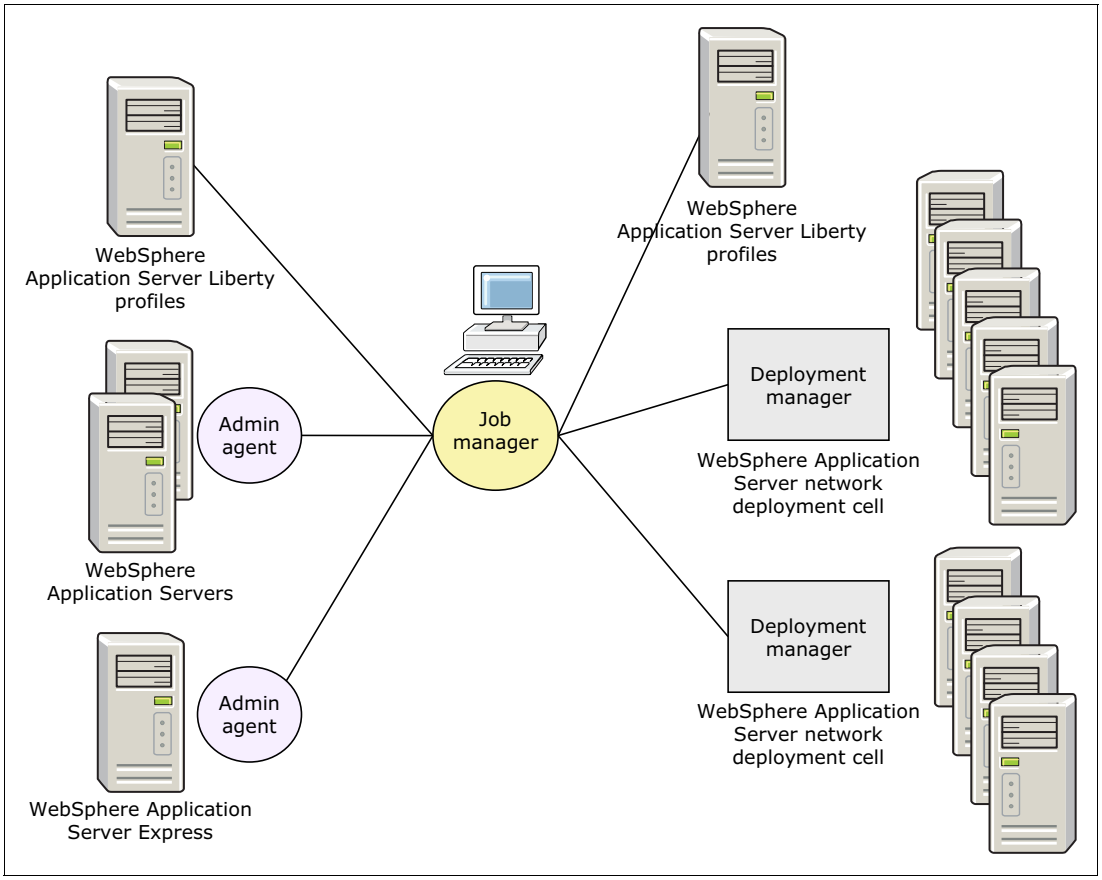


Figure 12-4 Flexible management with the job manager

Flexible management involves creating a job manager profile and using the **wsadmin registerWithJobManager** command to register the deployment manager or administrative agent with the job manager.

You can complete job manager actions and run jobs from a deployment manager. Like the jobs manager, the deployment manager administrative console has a list of job tasks that are available in a navigation tree. The administrative console of the deployment manager provides access to the following job manager options:

- Submit a job
- Review the status of a job
- Manage job manager targets for jobs
- Identify target resources used in jobs
- Manage target groups for administrative jobs

Remember: The job manager supports Liberty profile management. This support includes installing, uninstalling, and updating the Liberty profile. For the Liberty profile servers, support includes starting and stopping servers.

For more information, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=cagt_jobmanager

12.3.8 Monitored directory deployment

With monitored directory application deployment, you can automatically deploy and update applications. The applications can be deployed and updated by adding files to a monitored directory in the following ways:

- Adding enterprise application files
- Adding enterprise application files by adding properties files

By default, monitored directory application deployment is not enabled. You can use the administrative console or **wsadmin** scripts to enable or disable it. When monitored directory deployment is enabled, a monitored directory is created automatically based on the installation. By default, this directory is named `monitoredDeployableApps`:

- For base application servers, the monitored directory is *profile_root*/*profile_name*/monitoredDeployableApps/servers/*server_name*.
- For deployment managers, several monitored directories are in the deployment manager profile directory:
  - monitoredDeployableApps/servers/*server_name*
  - monitoredDeployableApps/nodes/*node_name*/servers/*server_name*
  - monitoredDeployableApps/clusters/*cluster_name*
- For properties files, the monitored directory is monitoredDeployableApps/deploymentProperties.

The polling interval specifies the number of seconds that the monitored directory is scanned for new applications.

Restriction: Using monitored directory for application deployment is available only on distributed and z/OS operating systems. It is not supported on IBM i operating systems.

Adding enterprise application files

You can install or update an application file by dragging or copying any of the following files to a monitored directory:

- Enterprise archive (EAR)
- Web archive (WAR)
- Java archive (JAR)
- Session Initiation Protocol (SIP) archive module

The monitored directory is scanned at a time interval based on the polling interval parameter. The status of an application file determines the action that is performed:

- Installation

  If an application file is added to the monitored directory, the application is installed and started.

- Update

  If an existing application file is updated in the monitored directory, the application is stopped, the update is installed, and then the updated application is started.

- Uninstallation

  If an application file is removed from the monitored directory, the application is stopped and uninstalled.

Remember: Adding an application file to a monitored directory does not change the existing Java Naming and Directory Interface (JNDI) and other application bindings. If binding values need to be set, install the files by using one of the following methods:

- The administrative console application installation wizard
- A **wsadmin** script
- A properties file that sets bindings

For more information about installing enterprise application files by adding them to a monitored directory, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=trun_app_install_dragdrop

Adding enterprise application files by adding properties files

You can install, update, or uninstall an EAR, WAR, JAR, or SIP archive file by dragging or copying an application properties file to a `monitoredDeployableApps/deploymentProperties` monitored directory.

Properties files can contain all the parameters in **wsadmin**:

- Application deployment actions:
  - Install
  - Update
  - Edit
  - Uninstall
- Application installation options
- Application installation bindings and extensions

The monitored directory is scanned at a time interval based on the polling interval parameter. If a new properties file is found, the **wsadmin applyConfigProperties** command runs automatically to install and start the application.

Exception: The properties files that are added to monitored directories differ slightly from typical properties files that are used to install, update, or uninstall applications. Consider the following examples:

- Statements such as `CreateDeleteCommandProperties=true` are not specified in the header of the properties section.
- To uninstall an application, specify `DELETE=true` in the header of the properties section.

For more information, see the Websphere Application Server V8.5 Information Center at:

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp

12.4 Automation planning

To emphasize the need for automated administration, consider that companies typically have multiple WebSphere Application Server environments. Multiple environments support activities in the different phases of the software development lifecycle. Each environment requires the same types of administrative tasks. With automation, a task can be run manually only one time and then have subsequent requests run automatically or with less effort.

Automating common procedures and actions is one of the keys to maintaining a stable and efficient WebSphere Application Server environment. You can reduce the possibility of error by eliminating human intervention in complicated tasks or by automating mundane procedures that are prone to mistakes. Automating WebSphere Application Server installation and configuration also allows an administrator to schedule recurring maintenance and backup procedures and other types of administrative tasks.

You can automate every action that you can run manually by using the administrative console and the WebSphere Application Server **wsadmin** tool or command-line utilities. You can automate the following tasks:

- Installation response files
  - Specify installation options one time, then use those options for multiple installations of WebSphere Application Server.
  - Enable silent execution mode.
- Command-line utilities
  - Use shell scripts on UNIX or batch files on Windows systems.
  - Run from a standard shell or command prompt.
  - Control different aspects of the WebSphere Application Server environments.
- WebSphere Ant tasks
  - Facilitate build and deploy processes to WebSphere Application Server.
- JMX framework
  - Provides standards-based capabilities to control and manage a WebSphere Application Server.
  - Creates custom Java clients to access managed resources.
- The **wsadmin** scripting tool
  - Starts administrative commands interactively or by running a script of commands.
- Centralized installation manager
  - Combines the installation of WebSphere Application Server with maintenance packages and fix packs in a single step.

Although scripting requires up-front development costs, in the long term it provides savings through automation and increases reliability. In addition, in many organizations, the administrative console is prohibited by security policy and infrastructure constraints. Scripted administration provides an alternative way to manage the WebSphere Application Server environment.

12.5 Configuration planning

This section provides information about global configuration planning topics. Configuring and managing the WebSphere Application Server runtime environment can be complex. This section addresses the following items to consider at the initial installation time:

- Configuration repository location and synchronization
- Configuring application and application server start behaviors
- Custom application configuration templates
- Planning for resource scope use

12.5.1 Configuration repository location and synchronization

WebSphere Application Server uses one or more configuration *repositories* to store configuration data. In a stand-alone server environment, one repository exists within the application server profile directory structure. In a distributed server environment, multiple repositories exist. The master repository is stored within the deployment manager profile directory structure. Each node also has a repository that is tailored to that node and its application servers. The deployment manager maintains the complete configuration in the master repository and pushes changes out to the nodes by using the file synchronization service. Repositories are in the *profile_home*/config subdirectory.

From a planning perspective, consider the actual location of the profile directory structures. The location can affect the performance and availability of the configuration file. The location is chosen during profile creation. If you run WebSphere Application Server for z/OS, consider using a separate hierarchical file system (HFS) for each node.

Consider whether to use automatic synchronization to push out changes to the nodes or to synchronize changes manually. In an environment where numerous administration changes occur, automatic synchronization might have a performance impact in the network.

12.5.2 Configuring application and application server start behaviors

With WebSphere Application Server, you can manage the start of applications and application servers. By default, applications start when their server starts.

By using the following settings, you can fine-tune the start speed and order in which the applications start automatically. You can access these settings in the administrative console by clicking **Applications** → **Application Types** → **WebSphere enterprise applications** → ***your_application*** → **Startup behavior**.

- Startup order

  By using this setting for an application, you can specify the order in which to start applications when the server starts. The application with the lowest “startup order” setting starts first. Applications with the same “startup order” setting start in parallel. Start order can be important for applications that are split into subapplications that need to start in a certain order because of dependencies between the applications.

- Start the application before server completes startup

  With this setting, you can specify whether an application must initialize fully before its server is considered started. Background applications can be initialized on an independent thread, allowing the server start to complete without waiting for the application.

- Create MBeans for resources

  Specify whether to create Managed Beans for resources such as servlets or JavaServer Pages (JSP) files within an application when the application starts.

Use the “parallel start” setting for an WebSphere Application Server to specify that the server components, services, and applications in an application server start in parallel. This option can shorten the startup time for a server. Access this setting by clicking **Servers** → **Server Types** → **WebSphere application servers** → ***your_server***.

The deployment manager, node agents, and application servers can start in any order they are discovered. The exception is the node agent, which must start before any application server on that node. Communication channels are established as they start, and each has its own configuration and application data to start.

You can prevent an application from starting automatically at application server start, so that you can start it manually later. To prevent an application from starting when a server starts, click **Applications** → **Application Types** → **WebSphere enterprise applications** → ***application_name*** → **Target specific application status**. Then disable auto start for the application.

12.5.3 Custom application configuration templates

With WebSphere Application Server, you can create a customized server template that is based on an existing server configuration. Then you can use that server template to create new servers. This template provides a powerful mechanism to propagate the server configuration, both within the same cell and across cell boundaries. To propagate the server configuration across cell boundaries, it must be exported to a configuration archive. The server configuration can then be imported to another cell.

You might need more than one application server, and the characteristics of the server might be different from the default server template. In this case, it is more efficient to create a custom template and use that template to create your WebSphere Application Server. When creating a cluster, use this template when you add the first member to the cluster. Create subsequent servers in the cluster by using the same template. This process reduces the scope for error and makes the task of creating the server cluster much faster.

12.5.4 Planning for resource scope use

Resource scope is a powerful concept to prevent duplication of resources across lower-level scopes. For example, if a data source can be used by multiple servers in a node, define that data source one time at the node level. Defining at the node level, rather than creating the data source multiple times, reduces the possibility of errors. Also, if the data source definition needs to change (for example, due to changes to an underlying database), you need to change it only once. It is visible to all servers within the node, saving both time and cost.

Consider outlining the resources that you need for all the applications to be deployed and at what scope to define each resource. Select the scope of a resource when you create it.

The following list describes the scope levels in order of granularity with the most general scope first:

- Cell scope

  The *cell scope* is the most general scope, and does not override any other scope. Consider making cell scope resource definitions granular at a more specific scope level. When you define a resource at a more specific scope, you provide greater isolation for the

resource. When you define a resource at a more general scope, you provide less isolation. Greater exposure to cross-application conflicts occurs for a resource that you define at a more general scope.

The cell scope value limits the visibility of all servers to the named cell. The resource factories within the cell scope are defined for all servers within this cell. They are overridden by any resource factories defined within application, server, cluster, and node scopes that are in this cell and have the same JNDI name. The resource providers that are required by the resource factories must be installed on every node within the cell before applications can bind or use them.

- Cluster scope

  The *cluster scope* value limits the visibility to all the servers on the named cluster. The resource factories defined within the cluster scope are available for all the members of this cluster to use. They override any resource factories that have the same JNDI name defined within the cell scope. The resource factories defined within the cell scope are available for this cluster to use, in addition to the resource factories defined within this cluster scope.

- Node scope (default)

  The *node scope* value limits the visibility to all the servers on the named node. This scope is the default scope for most resource types. The resource factories defined within the node scope are available for servers on this node to use. These factories override any resource factories that have the same JNDI name defined within the cell scope. The resource factories defined within the cell scope are available for servers on this node to use. They are available in addition to the resource factories defined within this node scope.

- Server scope

  The *server scope* value limits the visibility to the named server. This scope is the most specific scope for defining resources. The resource factories defined within the server scope are available for applications that are deployed on this server. They override any resource factories that have the same JNDI name defined within the node and cell scopes. The resource factories defined within the node and cell scopes are available for this server to use. They are available in addition to the resource factories defined within this server scope.

- Application scope

  The *application scope* value limits the visibility to the named application. Application scope resources cannot be configured from the administrative console. Use IBM Assembly and Deploy Tools for WebSphere Administration or the **`wsadmin`** tool to view or modify the application scope resource configuration. The resource factories defined within the application scope are available for this application to use only. The application scope overrides all other scopes.

You can define resources at multiple scopes, but the definition at the most specific scope is used.

When selecting a scope, the following rules apply:

- The application scope has precedence over all the scopes.
- The server scope has precedence over the node, cell, and cluster scopes.
- The cluster scope has precedence over the node and cell scopes.
- The node scope has precedence over the cell scope.

When viewing resources, you can select the scope to narrow the list to just the resources defined at the scope. Alternatively, you can select to view resources for all scopes. Resources are always created at the currently selected scope. Resources created at a scope might be

visible to a lower scope. For example, a data source created at a node level might be visible to servers within the node.

Explanation: A common source of confusion is the use of variables at one scope and the resources that use those variables at a different scope. Assuming that the correct definitions are available at a scope that the server can detect, variables do not have to be the same scope during run time.

However, consider the case of testing a data source. A data source is associated with a Java Database Connectivity (JDBC) provider. JDBC providers are commonly defined by using variables to point to the installation location of the provider product.

The scope of the variables and the scope of the JDBC provider do not have to be the same to be successful during run time. When using the test connection service to test a data source by using the provider, the variable scope and the scope of a JDBC provider must be the same.

For more information, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=ctestcon

12.6 Repository checkpoints service

WebSphere Application Server V8.5 introduces the repository checkpoints service to improve administration configuration changes. The repository checkpoints service helps an administrator to use checkpoints to track changes made to their application server configuration**.** Repository checkpoints represent saved images of the repository before configuration changes are made. The following are the checkpoints types:

- Full checkpoint

  The full checkpoint is created manually by the administrator and is a copy of the entire configuration repository. You can configure a checkpoint to back up copies of files from the master configuration repository.

- Delta checkpoints

  A delta checkpoint is created automatically when configuration changes are made and saved to the configuration repository. The delta checkpoint is formed by making a copy of the configuration documents affected by the configuration change before changes are applied.

  Remember: Delta checkpoints are optional, and are not enabled by default.

In WebSphere Application Server V8.5, you can perform the following actions on the repository checkpoints:

- Creating repository checkpoints

  You can create new repository checkpoints by clicking **System administration** → **Extended repository service** → **Repository checkpoints**. While the checkpoints are being created, the repository is locked. You have read access only to configuration data while the checkpoint is being created. Any attempt to make a configuration change during this period fails.

Consideration: Privileges for managing repository checkpoints are different depending on the administrative role of the user. Roles include monitor, operator, configurator, and administrator. If you are a user with either a monitor or an operator role, you can view only the repository checkpoint information. If you are a user with either a configurator or an administrator role, you have all configuration privileges for repository checkpoints.

- Enabling or disabling automatic checkpoints

  You can enable or disable the automatic checkpoints by clicking **System administration → Extended repository service → Repository checkpoints.**

- Archiving or Deleting checkpoints

  You can reduce clutter and free disk space by archiving or deleting old checkpoints periodically. When automatic delta checkpoints are enabled and checkpoint depth is high, the number of checkpoints that are stored to disk adds up. When the number of checkpoints reaches the checkpoint depth, WebSphere Application Server V8.5 automatically deletes delta checkpoints. If you want to preserve delta checkpoints, you must archive them before they are automatically deleted. Checkpoints can be archived easily by moving the checkpoint directories to a separate disk location.

- Restoring checkpoints

  WebSphere Application Server V8.5 allows you to restore the configuration repository back to the state it was in at the time the checkpoint was made. Using this function, you can reduce recovery time for problems that are caused by configuration changes. Depending on your needs, you can restore the total configuration repository or just delta checkpoints.

  Requirement: Delta checkpoints must be restored in descending sequence number order only. Selecting multiple checkpoints for restoration is not supported. Restore checkpoints one at a time. Select the latest delta checkpoint (the one with the largest sequence number), then restore it.

- Finding configuration changes in delta checkpoints

  If automatic repository checkpoints are enabled, the product creates a delta checkpoint whenever a change is made to the configuration repository. The delta checkpoint compressed file contains before and after versions of configuration files that were changed. You can extract the contents of the compressed file and then examine the extracted files to determine what changed in the configuration.

  Examine the extracted files to determine changes in the configuration. For more information, see the Websphere Application Server V8.5 Information Center at:

  `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=twve_checkpoint_changes`

- Enable audit records when saving changes to the master repository

  In WebSphere Application Server V8.5, you can enable security audit to track the repository configuration changes on the following areas:

  – Who made the changes
  – When the changes were made
  – What were the changes
  – Which configuration file was changed

After the security audit is enabled, a new audit record will be generated whenever the configuration repository changes. Example 12-1 shows a sample of an audit log.

Example 12-1 Sample audit log

```
Seq = 42
      | Event Type = ADMIN_REPOSITORY_SAVE | Outcome = SUCCESSFUL |
OutcomeReason = SUCCESS | OutcomeReasonCode = 109 | SessionId = null
      | RemoteHost = null | RemoteAddr = null | RemotePort = null | ProgName =
adminRepositorySave | Action = createDeltaCheckpoint
      | AppUserName = user1 | ResourceName = Delta-1328459402156 |
RegistryUserName = null | AccessDecision = authzSuccess
      | ResourceType = delta checkpoint | ResourceUniqueId = 0 |
PermissionsChecked = null | PermissionsGranted = null
      | RolesChecked = null | RolesGranted = null | CreationTime = Sun Feb 05
10:30:21 CST 2012 | GlobalInstanceId = 0
      | EventTrailId = -1444791282 | FirstCaller = user1 | Realm =
defaultWIMFileBasedRealm | RegistryType = WIMUserRegistry
```

The sample audit log details are explained as follows:

- Event Type = ADMIN_REPOSITORY_SAVE indicates a configuration save to the repository. Only successful saves cause an audit record to be generated.
- ResourceName = Delta-1328459402156 indicates the name of the checkpoint.
- AppUserName=user1 means user1 did the changes.
- CreateTime=Sun Feb 05 10:30:21 indicates when the change was made.

From this audit log, you know which configuration file was changed, who made the changes, when the changes were made, and other useful information.

For more information about the repository configuration checkpoints, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=twve_xdappedcfg

12.7 Change management

Effective change management is important to the longevity of any application environment. WebSphere Application Server contains several technologies to aid with the change management process. This section highlights topics to consider when planning for changes to the WebSphere Application Server V8.5 operational environment:

- Application update
- Changes in topology
- Centralized installation manager

12.7.1 Application update

WebSphere Application Server V8.5 permits fine-grained updates to applications. Application components are supplied and restart is limited to only the required parts of the application. This fine-grained approach preserves application configuration during the update process.

You can use the following options to update application files that are deployed on a server or cluster:

- Administrative console update wizard

  Use this option to update enterprise applications, modules, or files that are already installed on a server. The update can be entire EAR files, single or multiple modules (such as WAR or JAR files), or single or multiple file updates.

- **wsadmin** scripts

  Use **wsadmin** scripts to perform the same updates as the administrative console wizard.

- Hot deployment and dynamic reloading

  Hot deployment and dynamic reloading require you to directly manipulate the application or module file on the server where the application is deployed. The new files are copied directly to the installed EAR directory on the relevant server or servers.

When an application is deployed in a cluster, you can perform an automatic application rollout. This option provides a mechanism where each member in the cluster is stopped and updated with the application changes one at a time. When a server is updated, the next server is updated. Where clusters span multiple nodes, only one node at a time is updated. In this process, the cluster can operate uninterrupted as work is diverted from the node that is being updated to the other nodes. The process continues until the entire cluster receives the update. If only a single node is involved, that node is stopped and updated.

In WebSphere Application Server for z/OS, you can use the z/OS console **Modify** command to perform these tasks:

- Pause the listeners for an application server
- Update the application
- Resume the listeners

If you use this technique, you do not have to stop and then start the server to update the application.

12.7.2 Changes in topology

In a distributed server environment, the deployment manager node contains the master configuration files. Each node has its required configuration files available locally. Configuration updates must be done on the deployment manager node. The deployment manager process then synchronizes the update with the node agent. File synchronization is a one-way task, from the deployment manager to the individual nodes. Changes made at the node level are temporary and will be overwritten by the master configuration files at the next file synchronization. If security is turned on, HTTPS is used instead of HTTP for the transfer.

File synchronization

File synchronization settings are customizable by cell. Each cell can have distinct file synchronization settings. File synchronization can be automatic or manual:

- Automatic

  You can turn on automatic synchronization by using the administrative console. The default file synchronization interval is 60 seconds, and starts when the application server starts.

- Manual

  You can perform manual synchronization by using the administrative console, the **wsadmin** tool, or the **syncNode** command. The command is in the *install_root*/bin directory of the node that is synchronized.

The file synchronization process must coincide with the entire change management process. In general, define the file synchronization strategy as part of the change management process.

12.7.3 Centralized installation manager

Centralized installation manager is used to manage V8.5 and previous versions of WebSphere Application Server. You can install, update, and uninstall WebSphere Application Server remotely and apply maintenance packages by using the administrative console.

The process for managing WebSphere versions before V8 is different from the process for managing V8 and later, as shown in Table 12-1.

Table 12-1 Functional differences between centralized installation manager product versions

Function	CIM V6 and V7 (all releases)	CIM V8 and CIM V8.5
Scope	Install, update, and uninstall V7 (all releases). Update V6.1 (all releases).	Install, update, and uninstall V8 and all Installation Manager installable products. Targets can be added outside of the cell.
Installation software used	Integrated system management processor (ISMP) and Update Installer.	Installation Manager
Repository	Maintains a private repository on the deployment manager.	Maintains an installation kit directory and uses Installation Manager repositories.
Administrative console	Accessible from the deployment manager.	Accessible from the job manager, which is also accessible from the deployment manager.
Command line	Centralized installation manager **AdminTask** commands.	The job manager **submitJob** command.

Centralized installation manager for V8.5

Centralized installation manager for V8.5 is used to install and apply maintenance on remote targets, and is integrated into the job manager. Using this feature, you can manage multiple product offerings, such as the following products, in an agentless manner across cells:

- DMZ Secure Proxy Server
- IBM HTTP Server
- WebSphere Application Clients
- WebSphere Application Server
- WebSphere Application Server Liberty profile *(new in V8.5)*
- WebSphere Customization Toolkit
- Web server plug-ins

The centralized installation manager functions are accessed through the job manager or deployment manager. Because the functions are implemented as jobs, the process supports job scheduling. Using centralized installation manager jobs, you can perform the following tasks:

- Perform an inventory
- Install, update, and uninstall Installation Manager
- Manage offerings include, install, update, and uninstall WebSphere Application Server
- Manage offerings include install, update, and uninstall WebSphere Application Server Liberty profile
- Manage profiles
- Manage Liberty profile includes, start, and stop for the Liberty servers
- Run command
- Install SSH public key
- Distribute, collect, and delete files
- Test connection
- Add or search Installation Manager agent data locations

In z/OS environments, the centralized installation manager has some restrictions. For more information about available tasks for z/OS targets, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=tagt_jobmgr_imjobs

As shown in Figure 12-5, the centralized installation manager does not push the product binary files from the server. Instead, Installation Manager on the targets pulls the product binary files from the network repository directly. This process reduces the network traffic between the server and the targets, and reduces the processor utilization on the server.

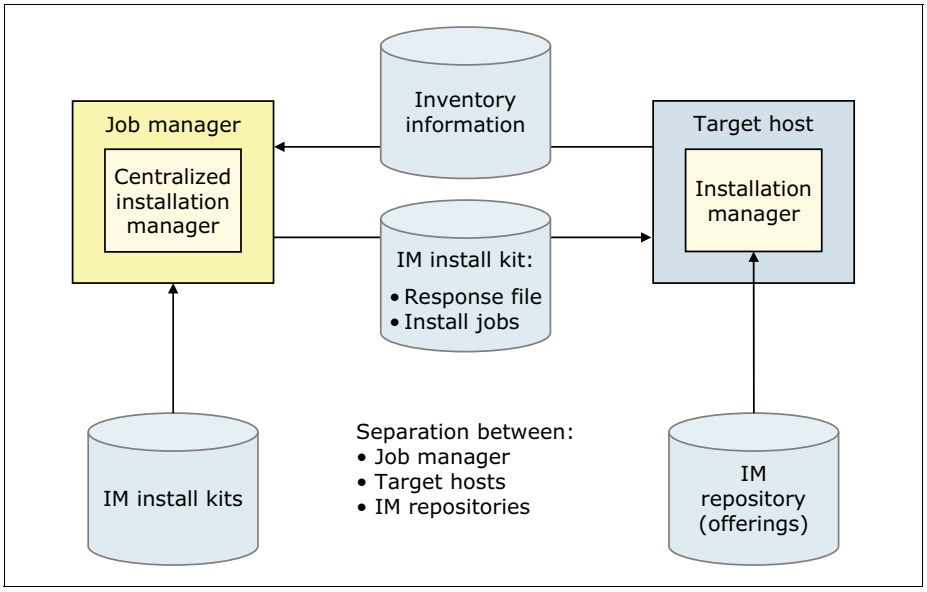


Figure 12-5 Centralized installation manager for WebSphere Application Server V8.5

The centralized installation manager for V8.5 is supported by the following operating systems:

- AIX
- HP-UX
- IBM i
- Linux
- Solaris

- Windows
- z/OS

Support: IBM Installation Manager V1.4.3 and later is also supported.

For more information about centralized installation manager for V8.5, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=tins_cim_overview

Centralized installation manager for V6.1 and V7

The functions for V6.1 (all releases) and V7 (all releases) are still available with the deployment manager. For more information, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=tins_cim

12.8 Serviceability

A major challenge of problem management is dealing with unanticipated issues. Much like detective work, you need to find clues, make educated guesses, and verify suspicions. The most important skills are common sense, focus, thoroughness, and rigorous thinking. A proactive approach to problem management is always the best choice. This section outlines general practices to follow.

Perform the following checks to avoid issues with the runtime environment:

- Check that you have the necessary prerequisite software up and running.
- Check that the correct authorizations are in place.
- Check for messages that signal potential problems. Look for warnings and error messages in the following sources:
  - Logs from other subsystems and products, such as TCP/IP, Resource Access Control Facility (RACF), and Windows Event Viewer
  - WebSphere Application Server `SystemOut.log` and `SystemErr.log` files
  - SYSPRINT of WebSphere Application Server for z/OS
  - Component trace output for the server
- Check the ports used by WebSphere Application Server. The ports that WebSphere Application Server uses must not be reserved by any other system component.
- Check that enough disk space for dump files is available.
- Check your general environment:
  - System memory
  - Heap size
  - System space requirements for archive data sets
- Make sure that all prerequisite fixes are installed. A quick check for a fix can save hours of debugging.
- Become familiar with the problem determination tools that are available in WebSphere Application Server and what these tools provide.

12.8.1 Log and traces

WebSphere Application Server V8.5 includes the following modes of logging:

- High Performance Extensible Logging mode
- Basic mode

High Performance Extensible Logging mode

Starting in WebSphere Application Server V8, you can configure the server to use the HPEL log and trace infrastructure. In prior versions, `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities were used for this logging. By default, HPEL is not enabled. You can enable it from the administrative console or by using **`wsadmin`** scripting.

For more information about enabling HPEL by using the administrative console or **`wsadmin`** scripting, see the Websphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=ctrb_HPELCompat`

After HPEL mode is enabled, the logs used in basic mode are no longer written to. HPEL keeps log and trace data stored in a proprietary binary format in two repositories and a text log file as illustrated in Figure 12-6 on page 406:

- Log data repository

  The log data repository stores log records from applications or servers written to the `System.out`, `System.err`, or `java.util.logging` file at the *Detail* level or higher. Data stored in the log data repository is useful to administrators the most often.

- Trace data repository

  The trace data repository stores trace records from applications or servers that are written to `java.util.logging` files at levels lower than *Detail*. Data stored in the trace data repository is most often useful to application programmers or by the WebSphere Application Server support team.

- Text log

  The text log file content is redundant because the data in the text log file is also stored in the log data and trace data repositories. The text log file is provided so that log content can be read without using the **`LogViewer`** command-line tool to convert the log data repository content to plain text. To improve server performance, the text log file can be disabled if the **`LogViewer`** tool is always used.

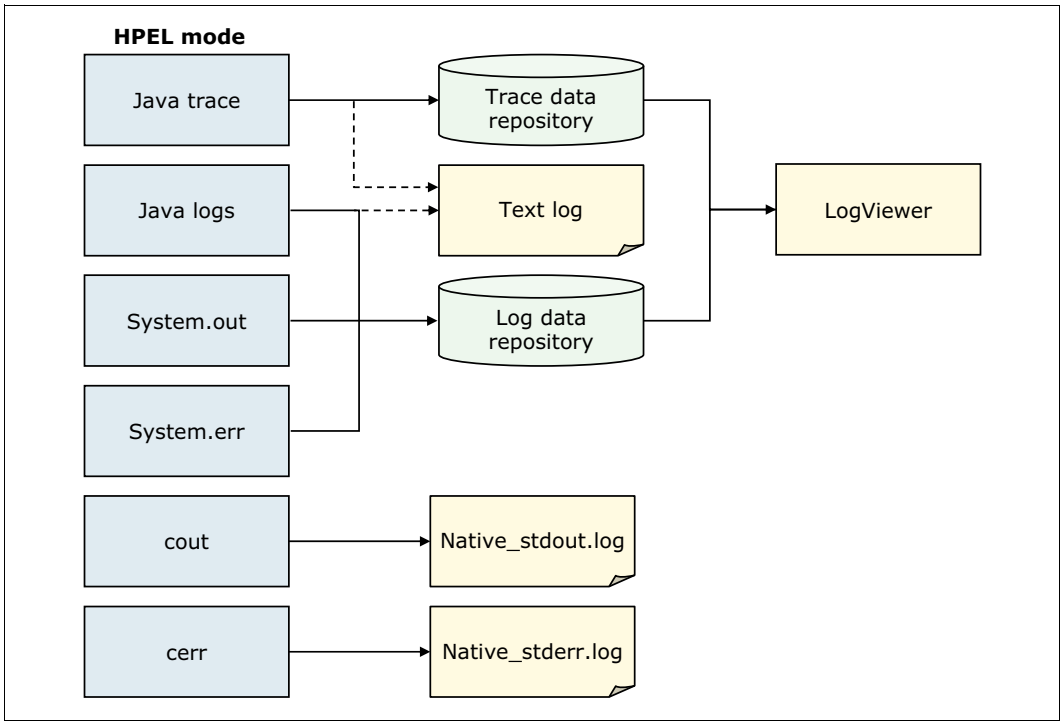


Figure 12-6 HPEL mode content and routing

Log and trace performance is greatly enhanced with HPEL because of the following factors:

- Log and trace events are stored in one place.

  Log and trace events are stored a single place and not redundantly in several locations. Log events, `System.out`, and `System.err` information is stored in the log data repository. Trace events are stored in the trace data repository. If the text log file is disabled, data is written only to these two repositories.

- Repositories are not shared across processes.

  Each server process has its own repository and text log file. The server environment, therefore, does not need to synchronize with other processes when writing data to the repositories or text log file.

- Data is not formatted until it is viewed.

  Log and trace data is stored in a proprietary binary format in the repositories rather than being formatted at run time. The log and trace data is not formatted until it is viewed by using the **`LogViewer`** tool.

  Restriction: Log and trace data stored in the repositories cannot be read by using text file editors. To view log and trace data, enable the text log file or convert it into a plain text format with the **`LogViewer`** command.

- Log and trace data is buffered before being written to disk.

For efficiency, HPEL buffers log and trace data in large blocks (8 KB) before writing it to disk. The size of the buffer and how often the buffer is written to disk are configurable. For more information about the configurable parameters, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=ttrb_usinghpel

Administrators can easily configure the resources that are required to store the log and trace data. The administrative console or **wsadmin** scripts are used to configure the following settings:

- The trace specifications
- The size of the repositories
- The location of the repositories
- Log record buffering
- Length of time to retain data
- Out of space actions

You can view, filter, and format the log and trace data by using these methods:

- **LogViewer** command

  The HPEL LogViewer is a simple command-line tool for HPEL users to work with the log and trace data repositories. The LogViewer provides filtering and formatting options that make finding important content in the log and trace data repositories easy. For example, a user can filter all log and trace entries that occurred within 10 seconds of a key error message. This filtering can be done on the same thread. You can use the LogViewer command-line tool to filter records based on the content of log and trace record extensions.

  Filter records based on extensions in V8.5:

  – The application server automatically creates an appName extension for each log and trace record related to a Java Platform or Enterprise Edition (Java EE) application. The appName extension indicates the name of that application.

  – The application server also automatically creates a requestID extension for each log and trace record created during the processing of certain types of requests. Requests like HTTP or Java Message Service (JMS) are examples of these types. The requestID extension indicates the unique ID of that request.

  – The requestID extension is added only to log and trace records when Cross-Component Trace is enabled. HPEL also provides the ability for developers to add custom extensions to log and trace records by using a log record context API.

- Administrative console as shown in Figure 12-7.

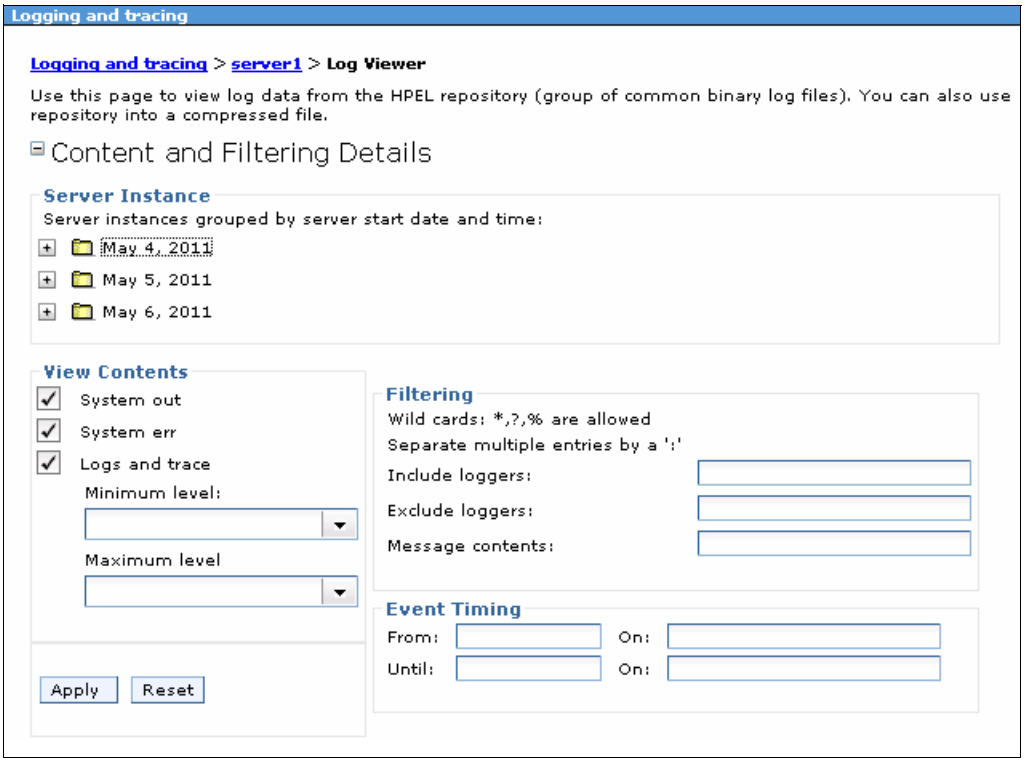


Figure 12-7 Filtering log and trace data by using HPEL

Developers can also use log and trace data to create log handling programs by using the available HPEL API. A message bean interface is also available to access log and trace data and to configure the repositories remotely. The log and trace data repositories can be read by using several methods, as illustrated in Figure 12-8:

- From a **`wsadmin`** script, using the HPELControlService JMX MBean (remotely or locally)
- From a Java program, using the HPELControlService JMX MBean (remotely or locally)
- From a Java program, using the `com.ibm.websphere.logging.hpel` API (locally)

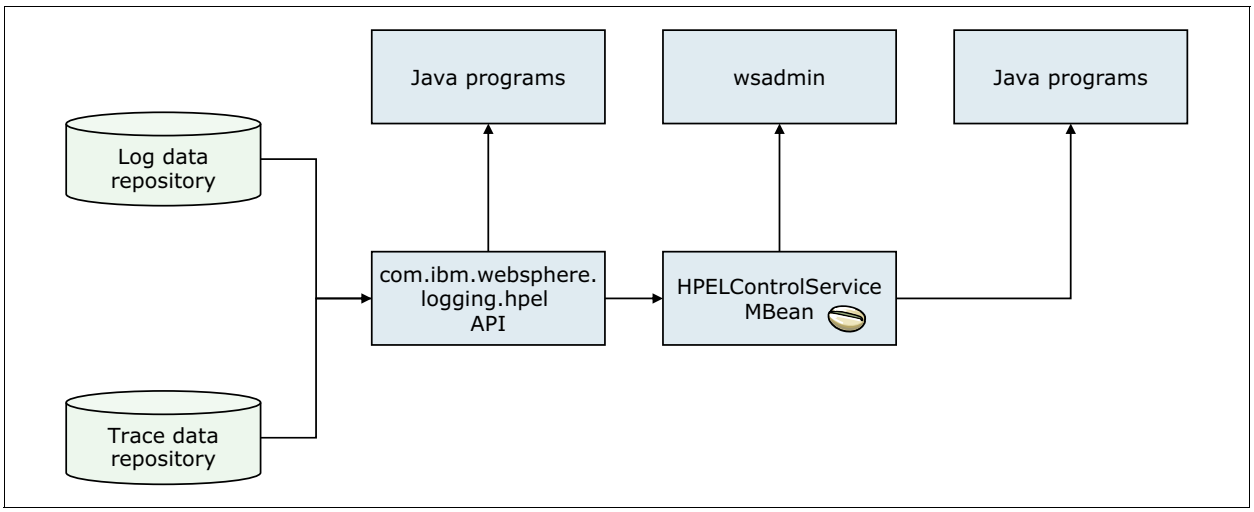


Figure 12-8 HPEL development resources

- Log and trace record extensibility

  Developers can use HPEL to add custom extensions to log and trace records through a log record context API, for example, `com.ibm.websphere.logging.hpel.LogRecordContext`. When HPEL stores log and trace records, it includes any extensions present in the log record context on the same thread. For example, a developer might write a servlet filter to add important HTTP request parameters to the log record context. While that servlet runs, HPEL adds those extensions to any log and trace records created on the same thread, for example, `.newfeat`.

  As with other log and trace record fields, developers can access the record extensions by using the HPEL API. This access is useful when writing tools that read from log and trace repositories. Developers can also use the log record context API to access extensions in custom log handlers, filters, and formatters at run time.

Basic mode

Basic mode is the log and trace function provided in previous releases of WebSphere Application Server. Basic mode is the default mode. No configuration changes are necessary to use basic mode. Any existing scripts and tools that you used with previous versions of WebSphere Application Server continue to function without modifications.

WebSphere Application Server can write the following system messages to several general-purpose logs, as illustrated in Figure 12-9 on page 410:

- JVM logs

  The JVM logs are written as plain text files, named `SystemOut.log` and `SystemErr.log`, and are written to the *`profile_home`*`/logs/`*`server_name`* directory.

  You can view the JVM logs from the administrative console, including logs for remote systems. You can also use a text editor on the system where the log files are stored.

- Process logs

  WebSphere Application Server processes contain two output streams that are accessible to native code that runs in the process. These streams are the standard output (stdout) and standard error (stderr) streams. Native code, including JVM, can write data to these process streams.

  By default, the stdout and stderr streams are redirected to log files at server startup. The stdout and stderr streams contain text written by native modules, including dynamic link libraries (DLLs), executables (EXEs), UNIX system libraries, and other modules.

  By default, these files are stored with the following names:

  - *`profile_home`*`/logs/`*`server_name`*`/native_stderr.log`
  - *`profile_home`*`/logs/`*`server_name`*`/native_stdout.log`

- IBM service log (`activity.log`)

  The service log is a special log file written in a binary format. You cannot view the log file directly with a text editor. Never directly edit the service log file because doing so can corrupt the log.

  You can view the service log by using one of the following methods:

  - Log Analyzer tool

    Use this tool to view the service log. This tool provides interactive viewing and analysis capabilities that are helpful in identifying problems.

– Showlog tool

  If you cannot use the Log Analyzer tool, use the **Showlog** tool to convert the contents of the service log to a text format. Select a text format that can write to a file or to the command shell window.

The IBM service log is in the *profile_home*/logs/ directory.

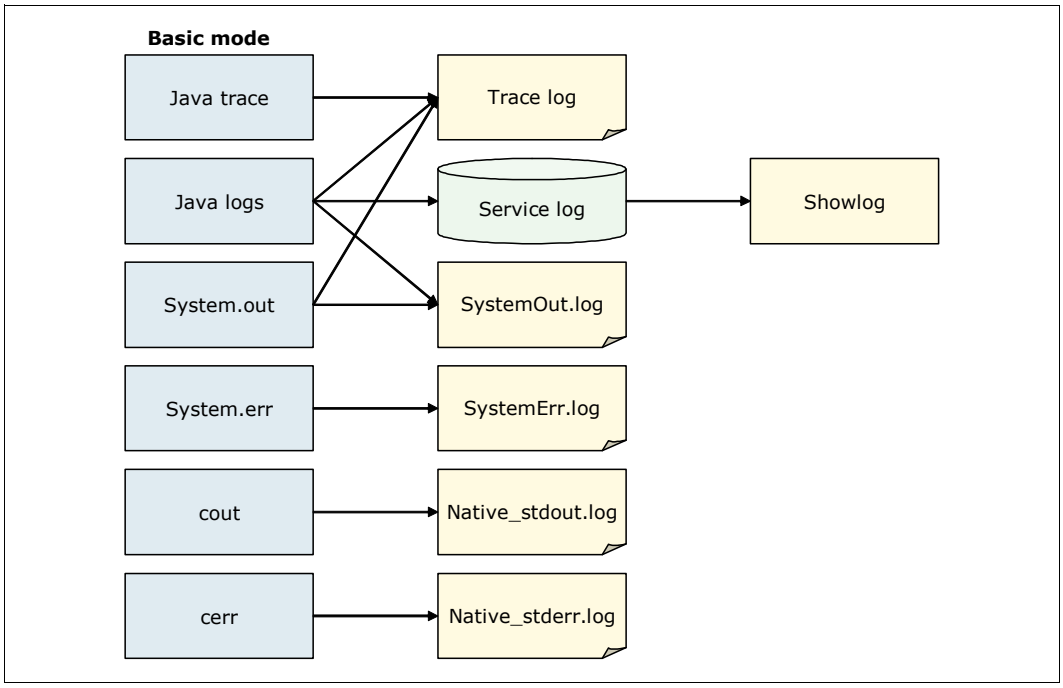


Figure 12-9 Basic mode content and routing

Log files and traces need to be named properly. Consider naming log files according to the application to which they belong and group them in different directories. Clean log files periodically, save them to media, and then delete them.

Explanation: On z/OS targets, the log files are in the job logs of the application server.

12.8.2 Fix management

Applying regular fixes is a key factor in reducing the probability and impact of problems. A fix plan establishes how fixes are applied on a regular basis. In addition to regular scheduled fixes, you might also need to perform emergency changes or fixes to a system in response to a newly diagnosed problem. The emergency fix plan outlines how to apply fixes safely and effectively. Overall, have a strong fix plan that outlines regular fix updates and reasonable retesting before each fix.

For more information, see the WebSphere Application Server support page at:

http://www.ibm.com/software/webservers/appserv/was/support/

12.8.3 Backing up and restoring the configuration

Back up the WebSphere Application Server configuration to a compressed 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. Do not run the **backupConfig** utility at the node level of a cell.

The **restoreConfig** command restores the configuration of your stand-alone node or cell from the compressed file that you created by using the **backupConfig** command.

Consider running the **backupConfig** utility before each major change to the WebSphere Application Server configuration.

12.8.4 MustGather documents

MustGather documents provide instructions on how to troubleshoot a problem and gather information to provide to IBM Support if opening a Problem Management Report (PMR). You can access MustGather documents from within IBM Support Assistant or on the IBM Support website. For more information, see:

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=troubleshooting&uid=swg21201625&loc=en_US&cs=utf-8&lang=en

On the IBM Support site, many MustGather documents are categorized as troubleshooting and analyzing data. Check the troubleshooting documents before you decide that you need to go through the MustGather document. The analyzing data document provides pointers for how to interpret the information that you collected from the MustGather document.

A majority of MustGather documents for WebSphere Application Server now have a corresponding AutoPD script in IBM Support Assistant. You can either follow the steps from the MustGather document manually, or run the AutoPD script, which does the work more or less automatically.

12.8.5 IBM Support Assistant

IBM Support Assistant improves your ability to locate IBM Support, development, and educational information through a federated search interface (one search, multiple resources). It provides quick access to the IBM Education Assistant and key product education road maps. It also simplifies access to the following IBM resources through convenient links:

- Product home pages
- Product support pages
- Product forums or news groups

In addition, problems can be submitted to IBM Support by collecting key information, then electronically creating a PMR from within IBM Support Assistant.

IBM Support Assistant includes a support tool framework that allows for the easy installation of support tools associated with different IBM products. It also provides a framework for IBM software products to deliver customized self-help information into the different tools within it. You can customize the workbench through the built-in updater feature to include the product plug-ins and tools that are specific to your environment.

The IBM Support Assistant Data Collector tool focuses on automatic collection of problem data, and is included in WebSphere Application Server V8.5.

The tool also provides symptom analysis support for the various categories of problems encountered by IBM software products. Information pertinent to a type of problem is collected to help identify the origin of the problem under investigation.

The tool can assist customers by reducing the amount of time it takes to reproduce a problem with the correct reliability, availability, and serviceability (RAS) tracing levels set. It also reduces the effort required to send the appropriate log information to IBM Support.

For more information about IBM Support Assistant and Data Collector, see the IBM Support website at:

http://www.ibm.com/software/support/isa

12.8.6 WebSphere Application Server Information Center

For troubleshooting information, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=welc6toptroubleshooting

12.9 Cross-component trace

WebSphere Application V8.5 introduces cross-component trace (XCT) to identify the root cause of problems across components.

Cross-component trace is built into the WebSphere Application Server log and trace framework. When enabled, cross-component trace annotates the logs so that log entries related to a request are identified as belonging to the same unit of work. This annotation is important because a request can be serviced by more than one thread, process, or server. In WebSphere Application V8.5, the cross-component trace brings the following benefits:

- Cross-component trace enables administrators and support teams to follow the flow of a request from end-to-end. Cross-component trace follows the request as it traverses thread or process boundaries and travels between stack products and WebSphere Application server.
- Helps to resolve questions about which component is responsible for a request that failed.

  In WebSphere Application Server V8.5, the following reflects the content of a cross-component trace log record:

  - XCT type (BEGIN / END): Demarcates the beginning and ending of work for a particular request on a particular thread.
  - XCT parent correlator ID: Demarcates when work is about to be transferred to or returned from another thread or process.
  - XCT current correlator ID: Demarcates when work is about to be transferred to or returned from another thread or process.
  - XCT annotations: Demarcates when work moves from major component to major component even if work continues on the same thread. For example, it can show transfer of control from application server code to application code.

Example 12-2 shows the cross-component trace log record in the log file.

Example 12-2 Cross-component trace log record

```
3/18/11 14:50:17:391 EDT] 00000031 XCT I BEGIN BJrcVPo+Yk4-AAAAAAA8zAA
00000000000-ccccccccc2 HTTPCF(OutboundRequest /index.html
RemoteAddress(127.0.0.1) RequestContext(36001645))?
```

The WebSphere Application Server V8.5 also provides the following to administer cross-component trace:

- Different cross-component trace modes for capturing the cross-component trace information:
  - Fully disabled
  - With cross-component trace, request IDs are added to existing log and trace records
  - With cross-component trace, request IDs are added to existing log and trace records and cross-component trace log records are added to log files
  - With cross-component trace, request IDs are added to existing log and trace records, cross-component trace log records are added to log files, and data snapshots are captured.

  Consideration: Cross-component trace adds the same request ID to every log or trace record when that record is a part of the same request. This addition occurs regardless of which thread or Java virtual machine (JVM) produces the log or trace entry.

- Viewer tool to view the cross-component trace logs easily and clearly:
  - When cross-component trace is used with the HPEL log and trace infrastructure, use the HPEL logViewer tool to view the request IDs.
  - When rendering log and trace content, XCT Log Viewer can also take advantage of cross-component trace log records or cross-component trace request IDs. The XCT Log Viewer is available as a tool add-on for the IBM Support Assistant.

For more information about how to use cross-component trace, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=ctrb_XCTOverview

12.10 Planning checklist for system management

Consider the following items as you plan system management:

- Create a strategy for administrative security. Identify the possible administrators and their roles. Determine the type of user registry that you will use for WebSphere security. If you do not want to use a federated repository, delay enabling administrative security until after installation.
- Review the administration facilities that are available (such as scripting and administrative console) and create an overall strategy for configuration and management of WebSphere Application Server resources.
- Determine where the profile directories (including the configuration repositories) will be located.

- Define a strategy for automation.
- Consider whether to use automatic or manual synchronization for nodes.
- Plan for starting the application server:
  - Set the start order.
  - Allow applications to start before the server completes startup.
  - Create Managed Beans for resources.
  - Set a parallel start.
- Create application server templates for existing servers if you plan to create multiple servers with the same customized characteristics.
- Create a strategy for scoping resources.
- Create a strategy for change management, including the maintenance and update of applications. This strategy includes changes in cell topology and updates to WebSphere Application Server binary files.
- Create a strategy for problem management. Use HPEL logging unless you have a special need for using basic logging mode. If using basic logging mode, identify a location and naming convention for storing WebSphere Application Server logs. Configure the processes to use those locations.
- Create a strategy for backup and recovery of the installation and configuration files.

The WebSphere Application Server Information Center contains useful information about system management. For a solid entry point to system management topics, see the Websphere Application Server V8.5 Information Center. The information center is available at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=welc6topmanaging

13

Messaging and service integration

This chapter provides information about planning for a WebSphere Application Server V8.5 environment that uses messaging facilities to connect to other applications.

This chapter includes the following sections:

- Messaging overview
- Service integration technology
- Messaging and service integration in WebSphere Application Server V8.5
- Enhanced resiliency for the service integration bus in V8.5
- Messaging options
- Messaging topologies
- Security and reliability of messaging features
- Planning checklist for messaging

This chapter briefly describes the concepts that are required to understand messaging. For more information, see the WebSphere Application Server V8.5 Information Center at the following address. Search for the phrase *messaging resources*:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=welcome_nd

13.1 Messaging overview

Generically, the term *messaging* describes the exchange of information between two or more interested parties. Messaging can take many forms. For example, sending a fax message from one point to another is point-to-point messaging. Sending a single message to many destinations, such as sending an email to a mailing list, is an example of the publish/subscribe messaging concept.

However, for the purposes of this chapter, messaging is defined as a synchronous or asynchronous method of communicating between processes on a computer. It provides reliable, secured transport of requests between applications. These applications can be on the same server, different servers, or different networks across a global application environment. The basic premise of messaging is that an application produces a message that is placed on a destination or queue. The message is retrieved by a consumer, who then does additional processing. The result can be that the producer receives data back from the consumer, or that the consumer performs a processing task for the producer.

Messaging is a popular facility for exchanging data between applications and clients of different types. It is also an excellent tool for communication between heterogeneous platforms. WebSphere Application Server implements a powerful and flexible messaging platform within the WebSphere Application Server environment, called the *service integration bus*.

13.2 Service integration technology

Service integration is a set of technologies that provide asynchronous messaging services. In asynchronous messaging, producing applications do not send messages directly to consuming applications. Instead, they send messages to destinations. Consuming applications receive messages from these destinations. A producing application can send a message and then continue processing without waiting until a consuming application receives the message.

13.2.1 Service integration buses

A service integration bus, or just a *bus*, is a group of one or more *bus members* in a WebSphere Application Server cell. This group cooperates to provide asynchronous messaging services. A cell requires only one bus, but a cell can contain any number of buses. The server component that enables a bus to send and receive messages is called a *messaging engine*.

A service integration bus provides the following capabilities:

- Any application can exchange messages with any other application by using a *destination*. The destination is where one application sends, and from which the other application receives.
- A message-producing application, called a *producer*, can produce messages for a destination regardless of which messaging engine the producer uses to connect to the bus.
- A message-consuming application, called a *consumer*, can consume messages from a destination (whenever that destination is available). It can consume messages regardless of which messaging engine the consumer uses to connect to the bus.

To configure a service integration bus, you use the administrative console. In the navigation pane, click **Service Integration** → **Buses**.

13.2.2 Bus members

A service integration bus can have the following members:

- Application servers
- Server clusters
- WebSphere MQ servers

Bus members that are application servers or server clusters contain messaging engines. These application server components provide asynchronous messaging services. Bus members that are WebSphere MQ servers provide a direct connection between a service integration bus and queues on a WebSphere MQ queue manager.

To configure a bus member, you use the administrative console. In the navigation pane, click **Service Integration** → **Buses** → ***bus name***, and select **Bus members** as shown in Figure 13-1.

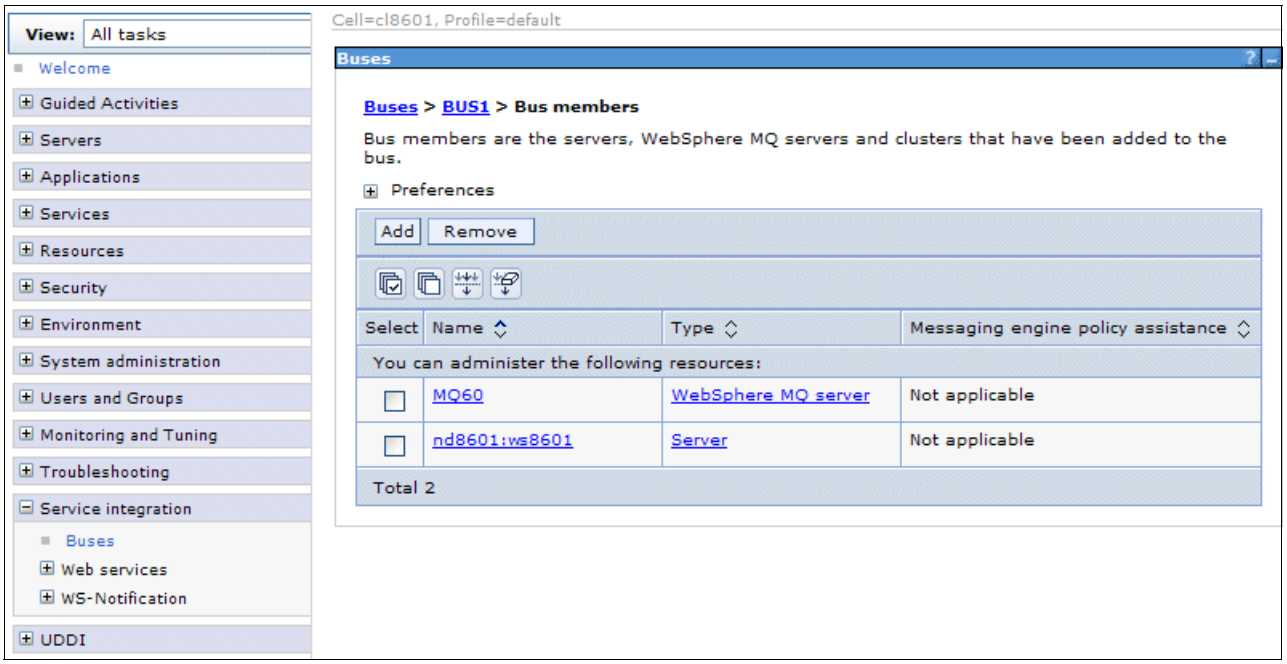


Figure 13-1 Bus members

To use WebSphere MQ as a bus member, you must first define WebSphere MQ as a WebSphere MQ server. To do so, use the administrative console in the navigation pane. Click **Server** → **Server Types**, and then select **WebSphere MQ server**.

13.2.3 Messaging engine

A messaging engine is a component that is responsible for processing messages, sending and receiving requests, and hosting destinations. To host queue-type destinations, the messaging engine includes a *message store* that can hold messages until consuming applications are ready to receive them. It can also preserve messages if the messaging engine fails. If the bus member is a WebSphere MQ server, it does not have a messaging engine.

The following types of message stores are available:

- A *file store* directly uses files in a file system through the operating system. This configuration is used to preserve operating information and to persist the objects that messaging engines need to recover if a failure occurs. It is split into the following levels:
  - The log file
  - Permanent store file
  - Temporary store file
- A *data store* uses a relational database. A messaging engine uses the operating information of a data store in the database to preserve essential objects. It needs these objects to recover if a failure occurs. It consists of a set of tables that a messaging engine uses to store persistent data in a database. A messaging engine uses an interface of a Java Database Connectivity (JDBC) data source to interact with that database.

Consideration: A messaging engine is created automatically when you add an application server or server cluster as a bus member.

For more information about the type of message store to use, see 13.7.3, “Planning for reliability” on page 448.

Figure 13-2 illustrates the components of the bus.

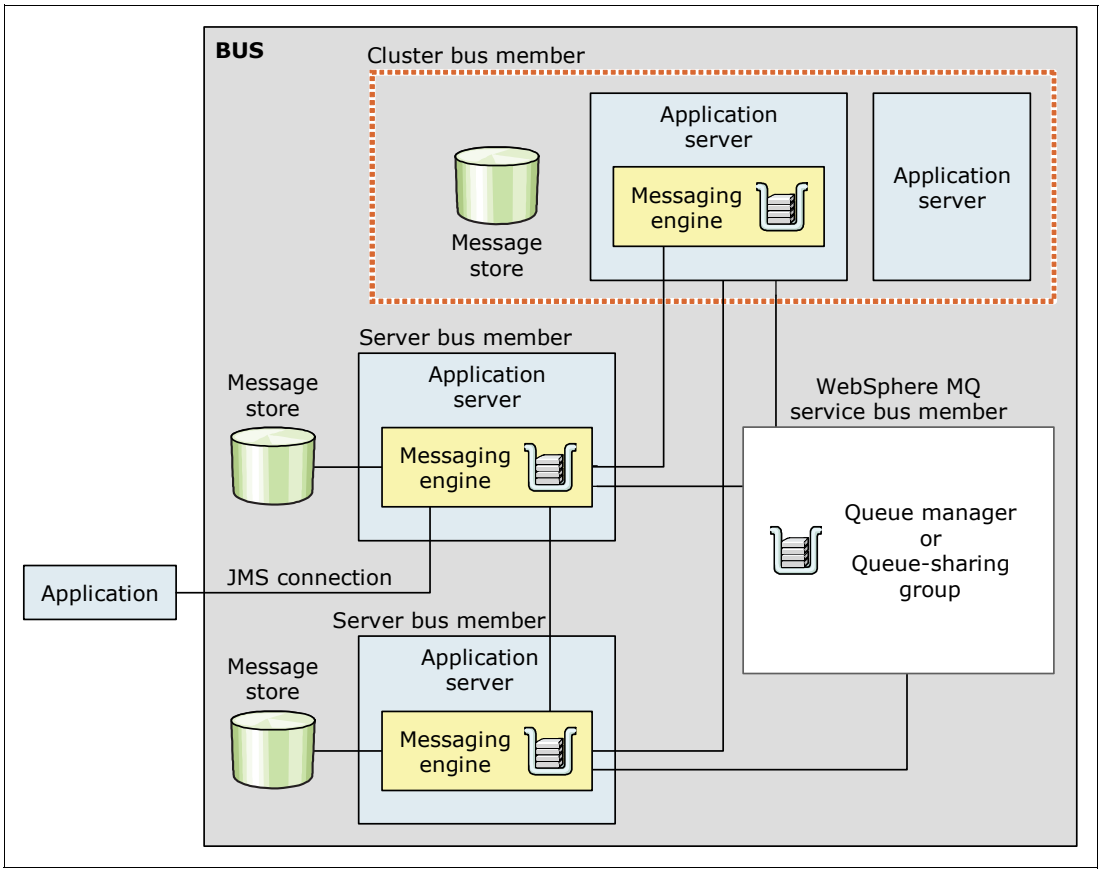


Figure 13-2 Main components of a service integration bus

13.2.4 Messaging provider

WebSphere Application Server applications start asynchronous messaging services by using the Java Message Service (JMS) application programming interface (API) to interface with a messaging provider. WebSphere Application Server supports the following types of messaging providers:

- Default messaging provider
- Third-party messaging provider
- WebSphere MQ messaging provider

For more information about the messaging provider, see 13.5, “Messaging options” on page 430.

Important: The Version 5 default messaging provider is deprecated. For compatibility with earlier releases, WebSphere Application Server continues to support this default messaging provider. Applications that use these resources can communicate with Version 5 nodes in mixed cells in later versions.For more information, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=tm_other

13.2.5 Other service integration concepts

This section provides information about other components that are related to service integration technology.

Bus destinations

A destination is defined within a bus, and represents a logical address to which applications can attach as message producers, consumers, or both. Destinations are associated with a messaging engine by using a message point.

Service integration has the following types of bus destinations, each with a different purpose:

- Queue destination

  Represents a message queue, and is used for point-to-point messaging.

- Topic space destination

  Represents a set of publish and subscribe topics, and it is used for publish/subscribe messaging.

- Foreign destination

  Represents a destination that is defined in another bus (a foreign bus). You can use a foreign destination for point-to-point messaging. The foreign bus can be another service integration bus or a WebSphere MQ network (one or more interconnected WebSphere MQ queue managers or queue-sharing groups).

- Alias destination

  Maps an alternative name for a bus destination that can be a queue destination or a topic space destination.

Bus destinations can be either permanent or temporary. A *permanent destination* is configured by an administrator, and has its runtime instances created automatically by the messaging provider. A *temporary destination* exists only while an application is using it. It can be used for queues (temporary queues) or topics (temporary topics).

Message point

A *message point* is the location on a messaging engine where messages are held for a bus destination. A message point can be a queue point, a publication point, or a mediation point (a specialized message point):

- Queue point

  The message point for a queue destination. When creating a queue destination on a bus, an administrator specifies the bus member that holds the messages for the queue. This action automatically defines a queue point for each messaging engine that is associated with the specified bus member.

  If the bus member is an application server, a single queue point is created and associated with the messaging engine on that application server. Messages that are sent to the queue destination are handled by this messaging engine. In this configuration, message ordering is maintained on the queue destination.

  If the bus member is a cluster of application servers, a queue point is created and associated with each messaging engine defined within the bus member. The queue destination is partitioned across the available messaging engines within the cluster. In this configuration, message ordering is not maintained on the queue destination.

- Publication point

  The message point for a topic space. When creating a topic space destination, an administrator does not need to specify a bus member to hold messages for the topic space. Creating a topic space destination automatically defines a publication point on each messaging engine within the bus.

- Mediation point

  A location in a messaging engine in which messages are stored and mediated. When an administrator associates a mediation with a bus destination, one or more mediation points are created on the bus member. The number of points created depends on the type of destination. For a mediated queue, a mediation point is created for each queue point on the bus member. For a mediated topic space, a mediation point is created for each publication point on the bus member.

Foreign bus and link

A *foreign bus* is an external messaging product that is either another bus or a WebSphere MQ network. You can set up a link to it so that messages traverse from one bus to another. The WebSphere MQ network can be seen as a foreign bus by the default messaging provider by using a WebSphere MQ link.

Mediations

A *mediation* is a Java program that extends the messaging capabilities of WebSphere Application Server. Mediations can be used to simplify connecting systems, applications, and components that use messaging. Mediations are used to process in-flight messages. Mediation can run the following types of processing:

- Transforming a message from one format to another
- Routing messages to one or more additional target destinations
- Adding data to a message from a data source
- Controlling message delivery based on some conditional logic in the mediation

You can use a mediation to process messages as an alternative to using message-driven beans (MDB). A mediation has the following advantages:

- It preserves message identity. If an MDB resends a message after processing its body, it sends a new message with a new message ID and message properties. By preserving the message identity, mediation makes it easier to track messages.
- It is independent of the messaging technology. The mediation programming model provides a Service Data Objects (SDO) Version 1 interface to all messages and a common API for accessing properties and metadata.

When a message arrives at the mediation point, the mediation consumes the message and then transforms, subsets, aggregates, or disaggregates the message. The message is then forwarded to another destination or returned to the same destination. The message then goes to the queue point where it can be consumed by the messaging application.

Figure 13-3 illustrates the mediation process flow.

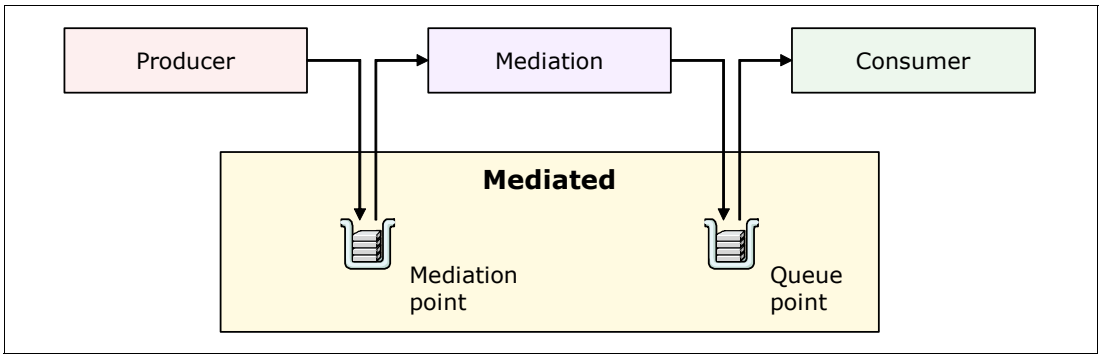


Figure 13-3 Mediation process flow

You can configure a destination so that the mediation point, the queue point, or both are WebSphere MQ queues. If both are WebSphere MQ queues, then a WebSphere MQ application, such as WebSphere Message Broker, can act as an external mediation, as illustrated in Figure 13-4.

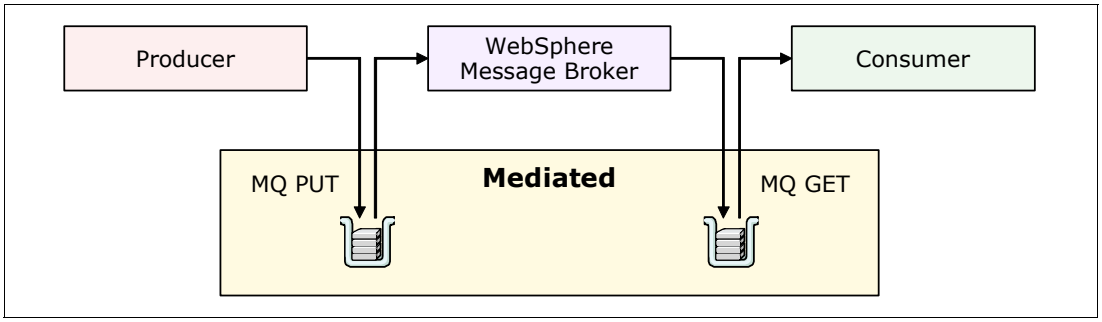


Figure 13-4 Mediation by using WebSphere MQ

Transport chains

The term *transport chain* describes the process and mechanism that a messaging engine uses to communicate. Communication can be with another messaging engine, external messaging provider, or a messaging application that runs outside of a server with a messaging engine. Transport chains are divided into inbound and outbound. They encompass encryption and communication protocols (for example, TCP/IP).

13.3 Messaging and service integration in WebSphere Application Server V8.5

WebSphere Application Server V8.5 includes the following resiliency enhancements for the service integration bus:

- Improvements to the recovery of messaging engine errors
- Enable the messaging engine to restart after a failure
- Retain the count of failed deliveries after the messaging engine is restarted
- Improvement to the messaging engine to prevent holding long running database locks
- Improvements to service integration bus performance
- Recovery of the messaging engine configuration from the message store

The following improvements introduced from V8 are still available in V8.5:

- Support for connecting to multi-instance WebSphere MQ

  WebSphere Application Server provides support for connecting to multi-instance WebSphere MQ queue managers. You can provide host and port information in the form of a connection name list. The list is used by a connection factory or activation specification to connect to multi-instance queue managers.

  To define multi-instance WebSphere MQ, perform these steps:

  a. In the administrative console, click **Resources** → **JVM** → **Activation specification** to display an existing WebSphere MQ resource provider.

  b. Select the provider that you want to specify as multi-instance WebSphere MQ.

  c. In the panel with the WebSphere MQ activation specification (Figure 13-5), select **Enter host and port information in the form of a connection name list**.

  d. Configure the multi-instance WebSphere MQ in the format `host[(port)][,host(port)]`.

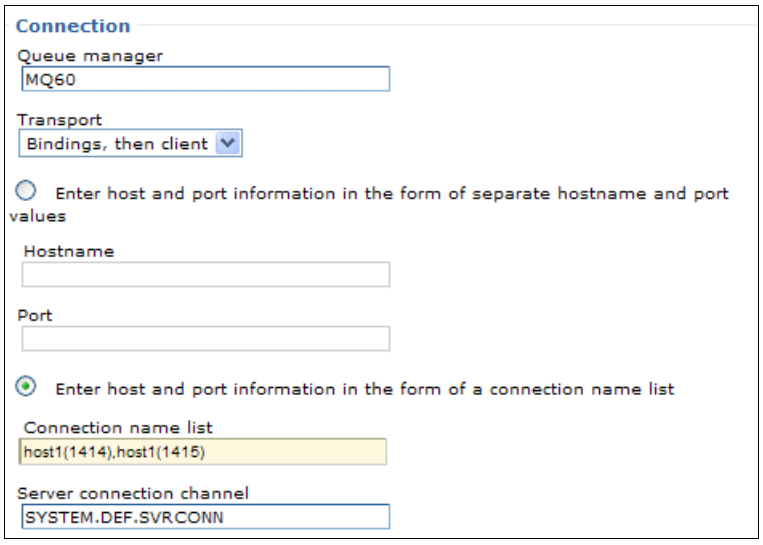


Figure 13-5 Defining a multi-instance WebSphere MQ

- Additional WebSphere MQ destinations properties

  With client reconnection properties for connection factories, a client node connection can be specified to reconnect automatically. This automatic reconnection is useful in the event of a communications or queue manager failure. You can specify a timeout value for reconnection attempts.

  For more information about client reconnection properties for connection factories, see the WebSphere Application Server V8 Information Center at:

  http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=umj_pjcfm_advprops

  In the WebSphere MQ queue or topic destination properties, you can specify the following information:

  – Whether an application processes the RFH version 2 header of a WebSphere MQ message as part of the JMS message body
  – The format of the **JMSReplyTo** field
  – Whether an application can read or write the values of MQMD fields from JMS messages sent or received by using the WebSphere MQ messaging provider
  – Message context options when sending messages to a destination

  For more information, see the WebSphere Application Server V8 Information Center at:

  http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=umj_MQQueueAdvancedProps

- Disabling WebSphere MQ functionality

  When a WebSphere Application Server process or an application process is running, processing is run to support WebSphere MQ-related functionality, such as the WebSphere MQ messaging provider. By default, this processing is run regardless of whether the

functionality is used. If you do not need any of the WebSphere MQ functionality, disable this process to increase performance. To disable WebSphere MQ functionality, select **Disable WebSphere MQ** (Figure 13-6) in the JMS provider general properties.

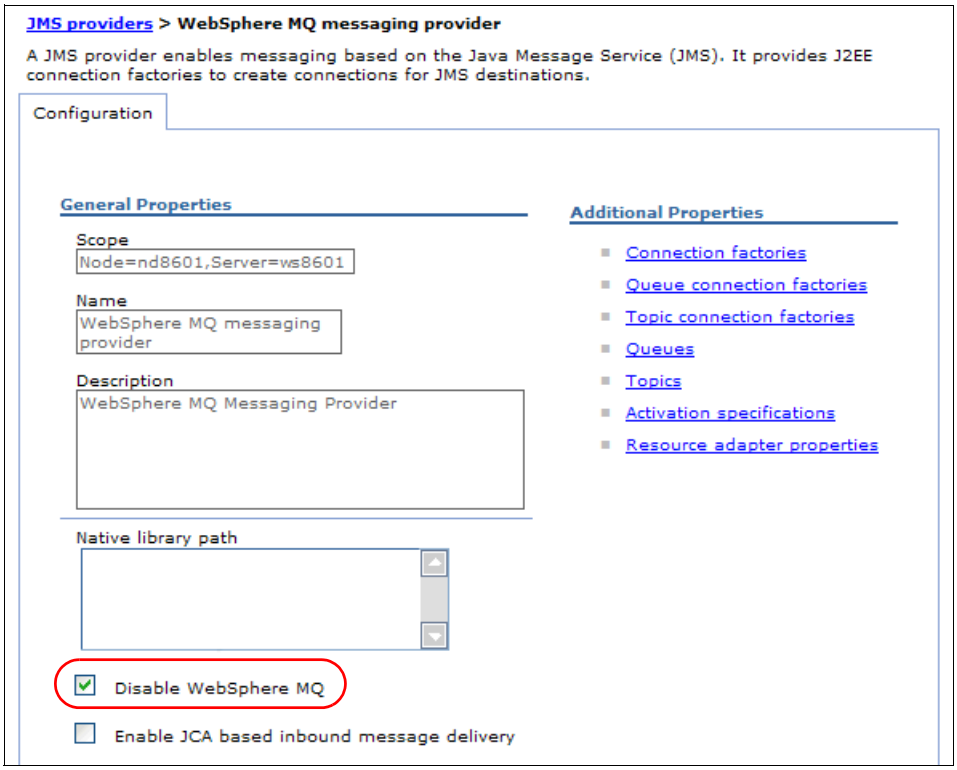


Figure 13-6 Disabling WebSphere MQ functionality

- Additional WebSphere MQ connection properties

  These properties were created to configure the WebSphere MQ resource adapter used by WebSphere MQ messaging provider as shown in Figure 13-7 on page 425. The following properties affect the connection pool that is used by the activation specification:

  - maxConnections

    The maximum number of connections to a WebSphere MQ queue Manager.

  - connectionConcurrency

    The maximum number of MDBs that can be supplied by each connection.

  - reconnectionRetryCount

    The maximum number of attempts made by the activation specification of a WebSphere MQ messaging provider to reconnect to a WebSphere MQ queue manager if a connection fails.

  - reconnectionRetryInterval

    The time, in milliseconds, that an activation specification of a WebSphere MQ messaging provider waits before attempting to again reconnect to a WebSphere MQ queue manager.

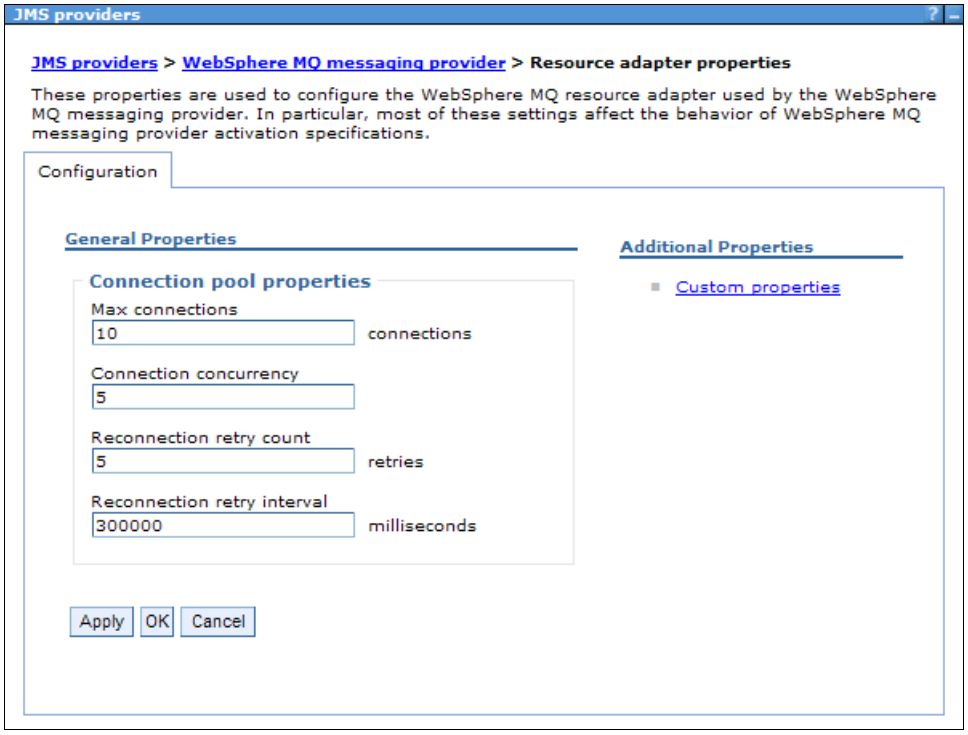


Figure 13-7 New WebSphere MQ connection properties

Websphere Application Server V8.5 has several other improvements and additions to messaging. For a full list, see WebSphere Application Server V8.5 Information Center at the following address, and click **What is new in this release**. Then search for *messaging*.

`http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp`

13.4 Enhanced resiliency for the service integration bus in V8.5

For service integration bus environments, the enhancements in WebSphere Application Server V8.5 include improved resiliency. This includes both failures of the messaging engine and the effects on the application server on which the messaging engine is running. In a messaging engine failure, the number of unsuccessful message deliveries is persisted. The messages retain the redelivery count even after the messaging engine is restarted.

The following enhancements have been made in WebSphere Application Server V8.5:

- Improvements to the recovery of messaging engine errors

  When a recoverable database error is detected by the high availability (HA) manager, the messaging engine is stopped, and the standby messaging engine is started automatically. The failure of the messaging engine does not affect the JVM, which ensures other applications that run in the application server are unaffected.

- Enable the messaging engine to restart after a failure

  When a messaging engine fails due to recoverable database problems, it is disabled and failed over to another messaging engine in the cluster. For this failover to occur, high availability must be configured. With WebSphere Application V8.5 the disabled messaging engine is automatically re-enabled without administrator intervention. Before WebSphere

Application Server V8.5, the messaging engine would remain disabled until the administrator manually enabled it.

- Retain the count of failed deliveries after the messaging engine is restarted

  When a message delivery fails, the messaging engine attempts to redeliver the message repeatedly and the delivery count increases incrementally each time.

  Before WebSphere Application Server V8.5, the redelivery count was not persisted to the message store after the messaging engine was restarted (Figure 13-8).

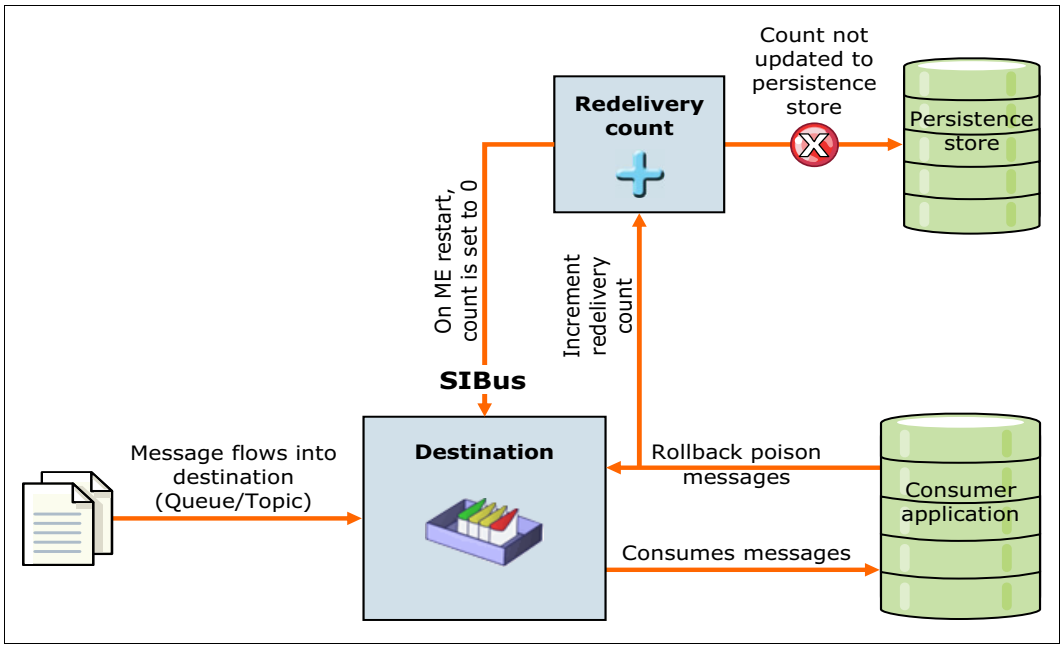


Figure 13-8 Redelivery count cannot be updated to the persistence store

In WebSphere Application Server V8.5 the redelivery count can be updated to the persistence store after the messaging engine is restarted (Figure 13-9).

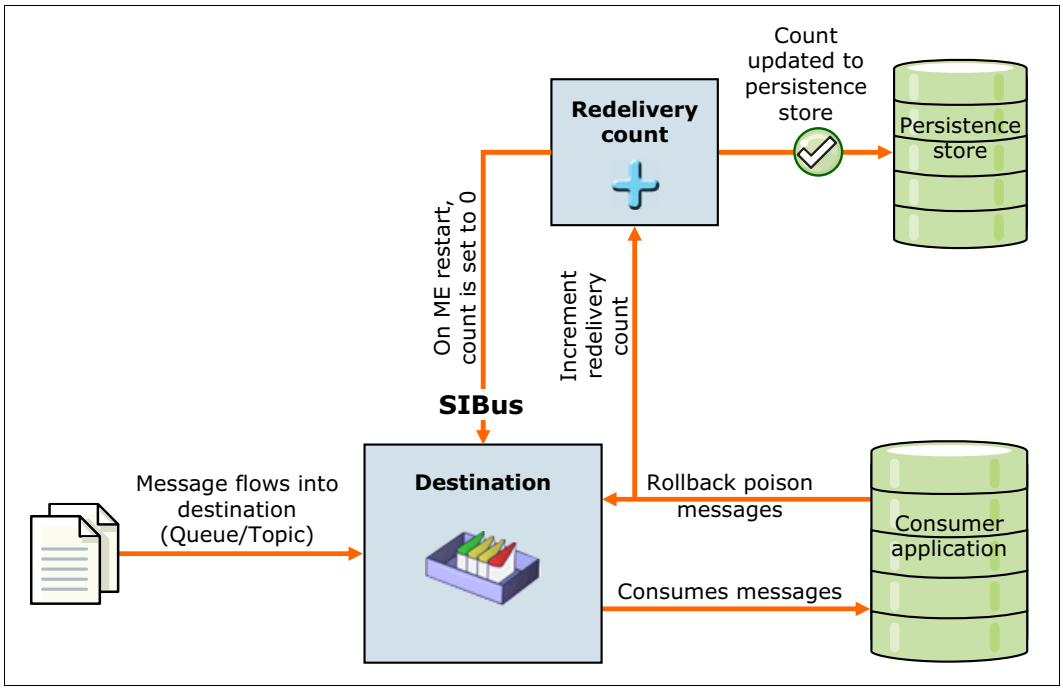


Figure 13-9 Redelivery count can be updated to the persistence store

- Improvement to the messaging engine to prevent holding long running database locks

  When the messaging engine uses a database as the message store, the messaging engine can be configured to acquire only short duration locks on the database. This configuration avoids long running locks on the database. The short duration locking mechanism creates locks only for the duration of a transaction. If the primary messaging engine becomes unresponsive, the standby messaging engine is able to acquire the database ownership without running into any stale locks.

Figure 13-10 illustrates the locking behavior before WebSphere Application Server 8.5 and the locking behavior with WebSphere Application Server V8.5.

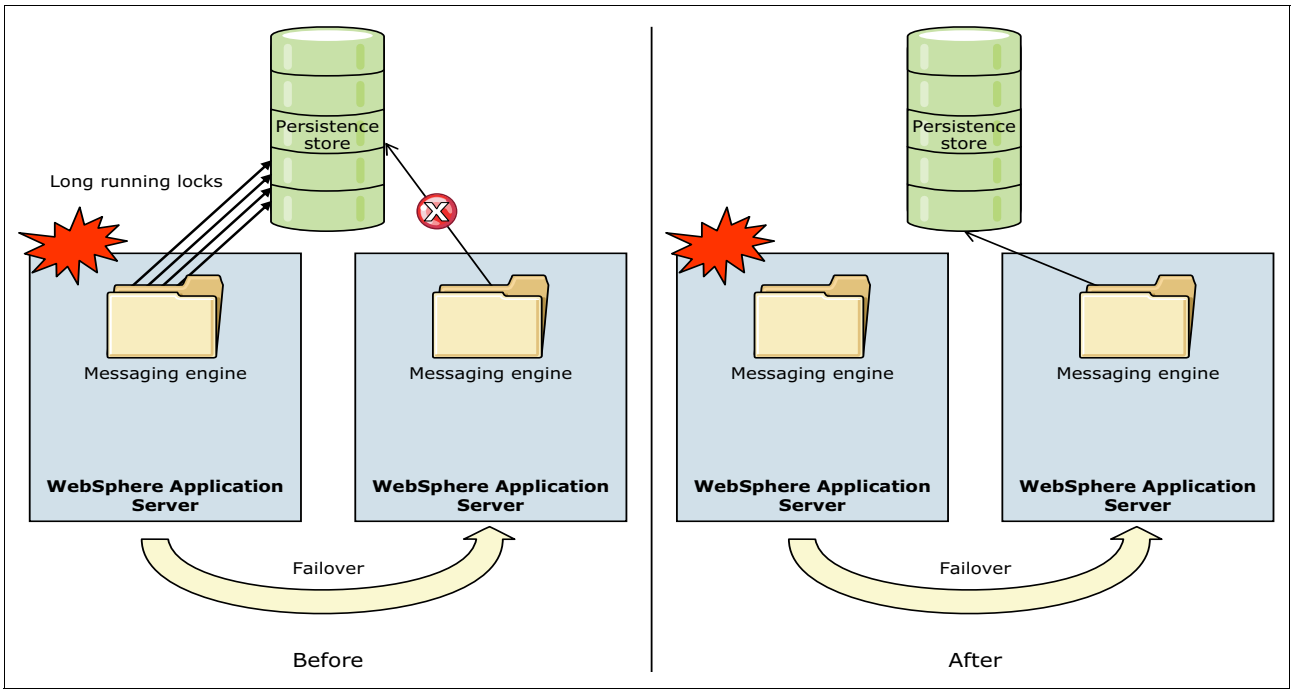


Figure 13-10 Locking behavior before and after WebSphere Application Server v8.5

- Improvements to service integration bus performance

  Messaging engine startup time is improved by pre-loading the destinations concurrently in a multi-core architecture. This concurrent loading is possible only if the message store is configured with a database that supports parallel reads by multiple threads. The performance improvement is directly proportional to the parallel processing capability of the database, and to the system capacity that runs the messaging engine.

- Recovery of the messaging engine configuration from the message store

  When a bus member of the messaging engine fails, the messaging engine still has persistent messages and configuration data in the bus message store. This data can now be recovered in the event of a messaging engine failure by using the **`recoverMEConfig`** command. The message store can either be a database or file store system to which the previous messaging engine was connected.

Figure 13-11 illustrates the issue with recovering the messaging configuration before WebSphere Application Server v8.5. Each messaging engine has a Universal Unique Identifier (UUID) that is stored at the WebSphere Application Server configuration level and also stored in the database. Creating a messaging engine and pointing the new messaging engine to the existing database fails. This failure occurs because the Universal Unique Identifier (UUID) of the new messaging engine would not match the UUID stored in the database.

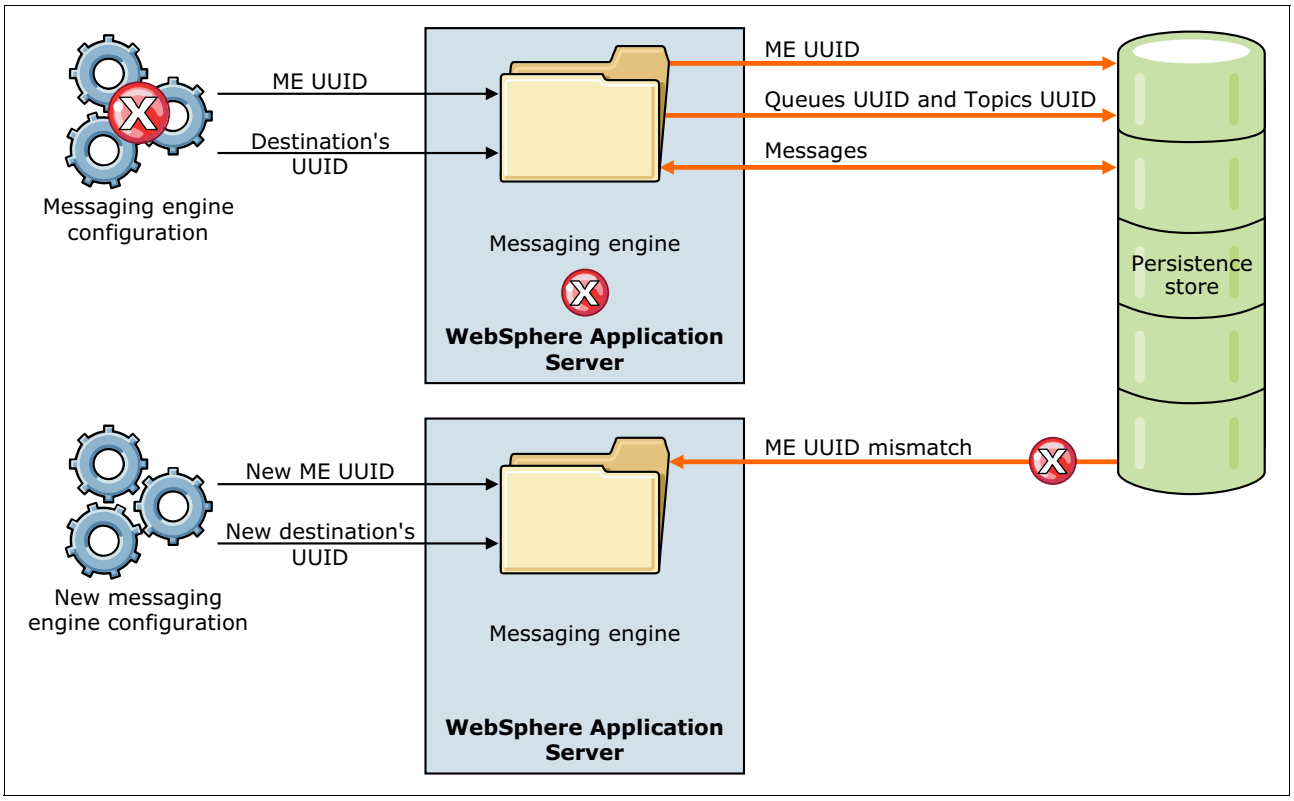


Figure 13-11 Messaging recovery behavior before WebSphere Application server v8.5

In WebSphere Application Server v8.5, the **`recoverMEConfig`** command can be used to recover the stored UUID of the failed messaging engine. It also recovers the queues and topics from the message store. Using this information, the new messaging engine is re-created with the prior configuration data and UUID values. When started, the new messaging engine can read all persistent messages.

Figure 13-12 illustrates this behavior.

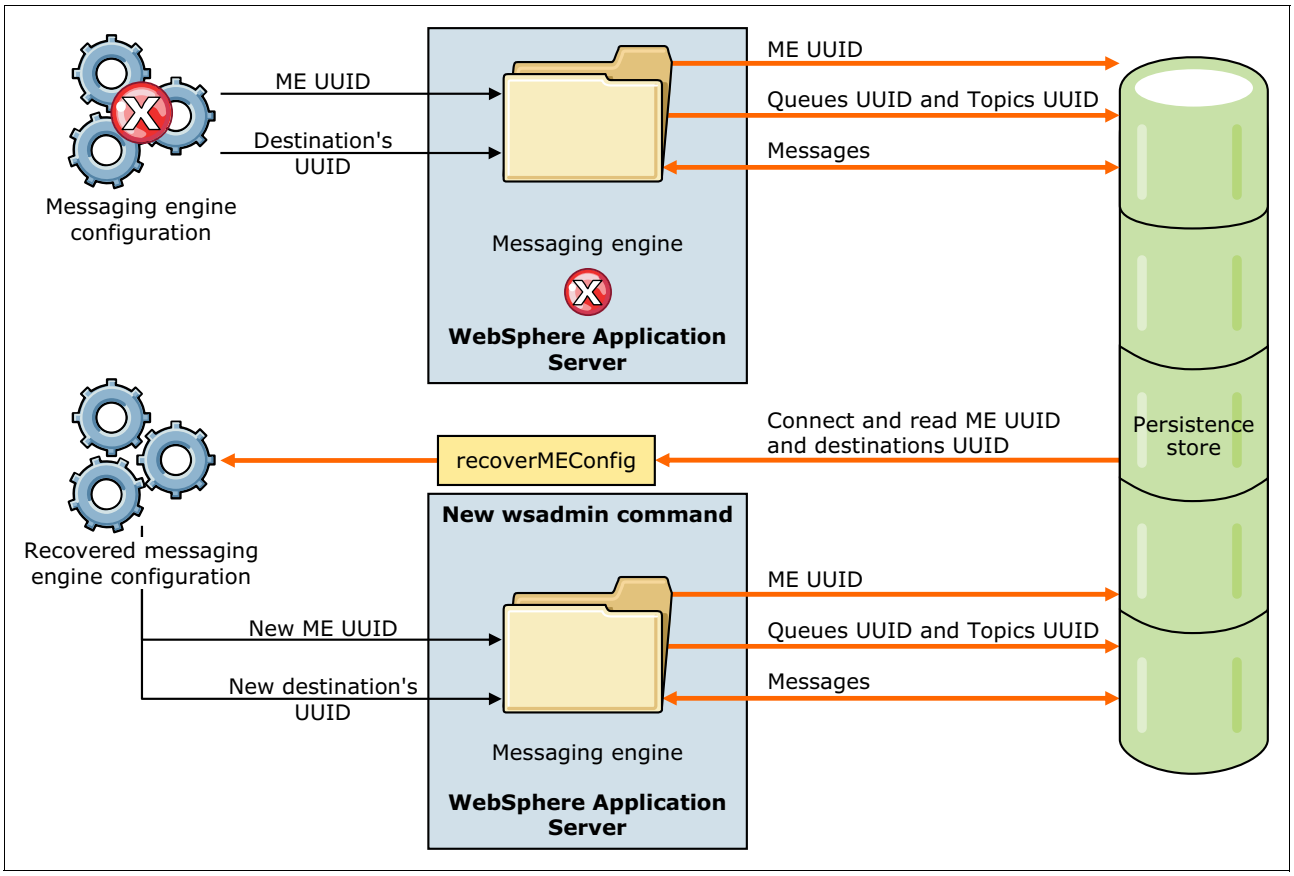


Figure 13-12 Restoring the configuration of a failed messaging engine

For more information about resiliency enhancements for the service integration bus, see the WebSphere Application Server V8.5 Information Center at the following address. Search for the phrase *enhanced resiliency for the service integration bus*.

`http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp`

13.5 Messaging options

This section addresses, at a high level, how messaging is implemented in WebSphere Application Server. It describes the options available as you design your messaging applications and infrastructure. This section includes the following information:

- Messaging provider standards
- Styles of messaging in applications
- Default messaging provider
- WebSphere MQ messaging provider
- Third-party messaging provider (generic JMS)
- Application design for retrieving messages

13.5.1 Messaging provider standards

To implement messaging within your application as a message producer, consumer, or both, your application needs to communicate with a messaging provider. Examples of messaging providers include the default messaging provider in WebSphere Application Server, WebSphere MQ, Oracle Enterprise Messaging Service, and SonicMQ.

Your application code can interact with these providers in several ways. Consider using the JMS API, but you can also use vendor-specific client libraries or the J2EE Connector Architecture (JCA) API. This section briefly provides information about the JMS and vendor-specific client library options.

Java Messaging Service

JMS is the standard API for accessing enterprise messaging systems from Java language-based applications. It provides methods and functions that are directly implemented by the underlying messaging provider. WebSphere Application Server V8.5 supports version 1.1 of the specification, which forms part of the overall Java Platform, Enterprise Edition 6 (Java EE 6) specification. For more information about the JMS V1.1 specification, see the Sun Developer Network Java Message Service website at:

`http://java.sun.com/products/jms`

Consider using JMS when writing an application to run within WebSphere Application Server for the following reasons:

- It is a tried-and-tested, consistent, and non-proprietary API that has been around for enough time to have plenty of skilled resources available.
- Applications that use it remain portable across many messaging providers.
- The API, though specific to messaging, has been expanded to support many message types and architectures, providing flexibility and versatility in the vast majority of applications.

Remember: For the rest of the chapter, JMS is the chosen method to access the messaging middleware provider.

Vendor-specific client libraries

Vendor-specific client libraries are libraries that are supplied by a software vendor so that applications can interact with their software. These libraries are similar to resource adapters, with the following important exceptions:

- They are proprietary and do not usually conform to any open standard.
- Use of the client libraries renders your applications non-portable across enterprise systems, and probably also across platforms.
- Support might not be available for certain languages such as Java, and these libraries have no direct support in WebSphere Application Server.

Avoid using these libraries whenever possible. They are usually only used in small, platform-specific utilities that do not run inside any type of application server.

13.5.2 Styles of messaging in applications

Applications can use the following styles of asynchronous messaging:

- Point-to-point messaging

  Point-to-point applications typically use queues to pass messages to each other. An application sends a message to another application by identifying, implicitly or explicitly, a destination queue. The underlying messaging and queuing system receives the message from the sending application and routes the message to its destination queue. The receiving application can then retrieve the message from the queue.

- Publish/subscribe messaging

  Publish/subscribe messaging has two types of applications: Publisher and subscriber:

  - A *publisher* supplies information in the form of messages. When a publisher publishes a message, it specifies a topic that identifies the subject of the information inside the message.
  - A *subscriber* is a customer of the information that is published. A subscriber specifies the topics it is interested in by sending a subscription request to a publish/subscribe broker. The broker receives published messages from publishers and subscription requests from subscribers. It then routes the published messages to subscribers. A subscriber receives messages from only those topics to which it is subscribed.

  This publish/subscribe style of messaging can be used in the following ways:

  - *One-way*: An application sends a message and does not want a response. A message such as this type can be called a datagram.
  - *One-way and forward*: An application sends a request to another application, which sends a message to yet another application.
  - *Request and response*: An application sends a request to another application that expects to receive a response in return.

13.5.3 Default messaging provider

The fully featured default messaging provider is available at no cost with WebSphere Application Server. It supports JMS 1.1 domain-independent interfaces. It is a robust and stable messaging platform that can handle point-to-point queues, topics in a publish-subscribe environment, and web service endpoints.

You can use the WebSphere Application Server administrative console to configure JMS resources for applications. The console can also manage messages and subscriptions that are associated with JMS destinations. The following resources are needed to configure the default messaging provider:

- A JCA activation specification to enable an MDB to communicate with the default messaging provider
- A JMS connection factory to create connections to JMS resources on a service integration bus
- A JMS queue or topic that is used to refer to the JMS destination with which applications interact. The administrator configures the JMS queue or topic as a JMS resource of the default messaging provider

Java EE applications (producers and consumers) access the bus and the bus members through the JMS API. JMS destinations are associated with bus destinations. A bus destination implements a JMS destination function. Session Enterprise JavaBeans (EJB) use

a JMS connection factory to connect to the JMS provider. MDBs use a JMS activation specification to connect to the JMS provider, as illustrated in Figure 13-13.

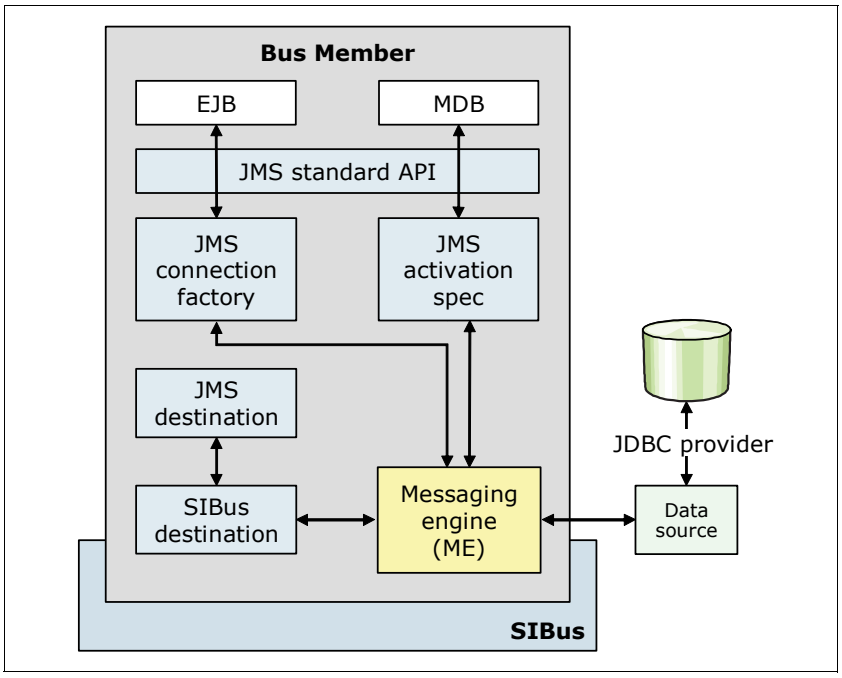


Figure 13-13 WebSphere default messaging provider and JMS

13.5.4 WebSphere MQ messaging provider

If your business uses WebSphere MQ, and you want to integrate WebSphere Application Server messaging applications into a WebSphere MQ network, the WebSphere MQ messaging provider is a logical choice. If you are unsure about which provider combination is suited to your requirements, see WebSphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tmj_jmsp_mixed`

The configuration in WebSphere Application Server for a WebSphere MQ provider is similar to using the default messaging providers. Configure a JMS provider for WebSphere MQ and the JMS resources required for the application to send messages to the queue or topic.

You also can use WebSphere Application Server to coordinate global transactions that include WebSphere MQ without configuring the extended transaction client. In this configuration, you administer security through WebSphere MQ.

Tip: If you do not need to take advantage of any WebSphere MQ functionality, disable it in an application server or client process to increase performance. For more information, see “Disabling WebSphere MQ functionality” on page 423.

The following sections compare, at a high level, the three ways that you can send messages between WebSphere Application Server and a WebSphere MQ network. They highlight the advantages and disadvantages of each approach.

WebSphere MQ as an external messaging provider

The WebSphere MQ messaging provider does not use service integration. Rather, it provides direct access to WebSphere MQ for applications by using JMS messaging. The WebSphere MQ messaging provider makes point-to-point messaging and publish/subscribe messaging available to WebSphere Application Server applications.

Table 13-1 conveys the advantages and disadvantages of using WebSphere MQ as an external messaging provider.

Table 13-1 Advantages and considerations of WebSphere MQ as an external messaging provider

Advantages	Considerations
► You do not have to configure a service integration bus or messaging engines. ► You can connect directly to WebSphere MQ queue managers. ► You manage a single JMS messaging provider. ► You can connect to queue managers in client mode or bindings mode. ► You can use point-to-point messaging and publish/subscribe messaging. ► You can use WebSphere Application Server clusters.	► If you are using message-driven beans, performance is lower. ► Interaction between WebSphere Application Server and WebSphere MQ is not seamless. ► You cannot use mediations for modifying messages, routing, or logging.

Figure 13-14 shows a JMS application in WebSphere Application Server sending messages to WebSphere MQ as an external messaging provider.

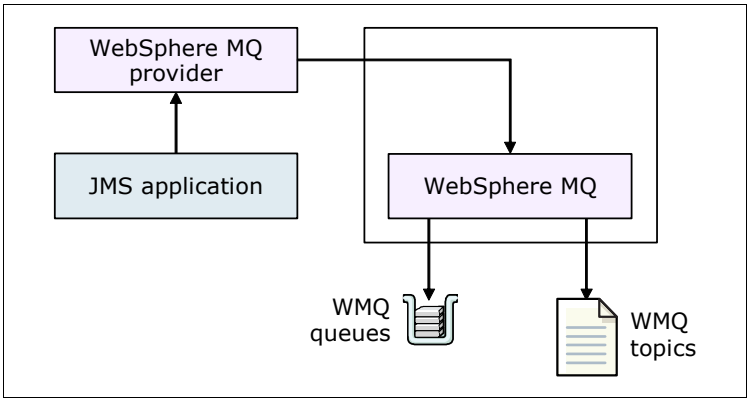


Figure 13-14 WebSphere MQ as an external messaging provider

WebSphere MQ network as a foreign bus (using WebSphere MQ links)

A WebSphere MQ link provides a server-to-server channel connection between a service integration bus and a WebSphere MQ queue manager or queue-sharing group (z/OS). The link acts as the gateway to the WebSphere MQ network. In this link, the WebSphere MQ network views the message bus as a virtual queue manager. The service integration bus views the WebSphere MQ network as a foreign bus.

With a WebSphere MQ link, WebSphere Application Server applications can send point-to-point messages to WebSphere MQ queues that are defined as a destination in the service integration bus. In addition, WebSphere MQ applications can send point-to-point messages to destinations in the service integration bus. Buses are defined as remote queues to WebSphere MQ.

You can also set up a publish/subscribe bridge. WebSphere Application Server applications can subscribe to messages published by WebSphere MQ applications, and WebSphere MQ

applications can subscribe to messages published by WebSphere Application Server applications. The link ensures that messages are converted between the formats used by WebSphere Application Server and those formats used by WebSphere MQ.

Table 13-2 lists the advantages and considerations of using the WebSphere MQ network as a foreign bus.

Table 13-2 Advantages and considerations of using WebSphere MQ network as a foreign bus

Advantages	Considerations
► A WebSphere MQ client facility is not required on the gateway WebSphere MQ queue manager. ► Each end of the link is displayed in natural form to the other. WebSphere MQ is displayed to service integration as a (foreign) bus. Service integration is displayed to WebSphere MQ as a (virtual) queue manager. ► Increased performance over the link is possible when compared with WebSphere MQ servers or direct connection to WebSphere MQ as an external JMS messaging provider. ► A managed connection from one node to another is created, but not from every application server in the cell. ► You do not have to define individual WebSphere MQ queues to the service integration bus. ► Security support is provided. For example, you can control which users are allowed to put messages onto queues. ► WebSphere Application Server and WebSphere MQ can exist on separate hosts. ► Interaction between WebSphere Application Server and WebSphere MQ is seamless. ► You can configure a publish/subscribe bridge. Through this bridge, WebSphere Application Server applications can subscribe to messages published by WebSphere MQ applications, and WebSphere MQ applications can subscribe to messages published by WebSphere Application Server applications. ► You can join publish/subscribe domains across the service integration bus and WebSphere MQ.	► You must configure a service integration bus and messaging engines. ► You cannot connect to queue managers in bindings mode. ► Optimum load balancing is more complicated to achieve because messages must be "pushed" from either end of the link. ► You cannot use mediations for modifying messages, routing, or logging.

Figure 13-15 shows a JMS application in WebSphere Application Server sending messages to WebSphere MQ by using WebSphere MQ network as a foreign bus.

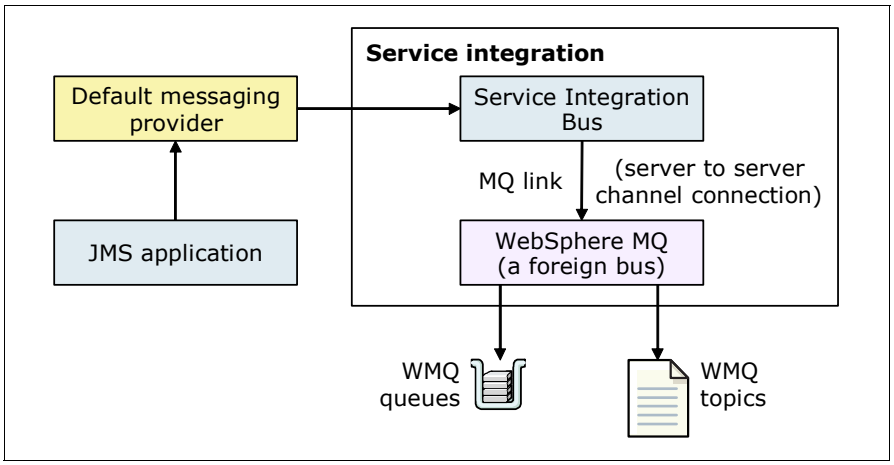


Figure 13-15 WebSphere MQ network as a foreign bus

WebSphere MQ server as a bus member

A WebSphere MQ server provides a direct client connection between a service integration bus and queues on a WebSphere MQ queue manager or queue-sharing group (z/OS). A WebSphere MQ server represents queues for point-to-point messaging only, and ensures that messages are converted between the formats used by WebSphere Application Server and WebSphere MQ.

Table 13-3 lists the advantages and considerations of using a WebSphere MQ server as a bus member.

Table 13-3 Advantages and considerations of WebSphere MQ server as a bus member

Advantages	Considerations
► WebSphere Application Server and WebSphere MQ can exist on separate hosts. ► Each end of the connection is displayed in natural form to the other. WebSphere MQ queue manager is displayed to service integration as a foreign bus. Service integration is displayed to WebSphere MQ as a client. ► Close integration of applications is possible. Service integration applications can consume messages directly from the WebSphere MQ network. ► You can connect to queue managers in client mode or bindings mode. ► You can use mediations for modifying messages, routing, or logging. ► Good security support is provided. For example, you can control which users are allowed to put messages onto queues. ► You can get messages from WebSphere MQ queues. ► Interaction between WebSphere Application Server and WebSphere MQ is seamless. ► Queues on the WebSphere MQ network are automatically discovered.	► You must configure a service integration bus and messaging engines. ► The queue managers and queue-sharing groups must be accessible from all the messaging engines in the bus. ► A topic for publish/subscribe messaging cannot be represented as a WebSphere MQ server. ► WebSphere MQ for z/OS Version 6 or later, or WebSphere MQ (distributed platforms) Version 7 or later, is a prerequisite. ► If you are using different nodes with WebSphere MQ for z/OS, you might require the Client Attachment feature (CAF) on z/OS. The need for CAF is dependent on the number of nodes and your version of WebSphere MQ for z/OS, ► You must explicitly define all destinations.

Figure 13-16 shows a JMS application in WebSphere Application Server sending messages to WebSphere MQ. The application uses WebSphere MQ (a queue manager or queue-sharing group) as a bus member.

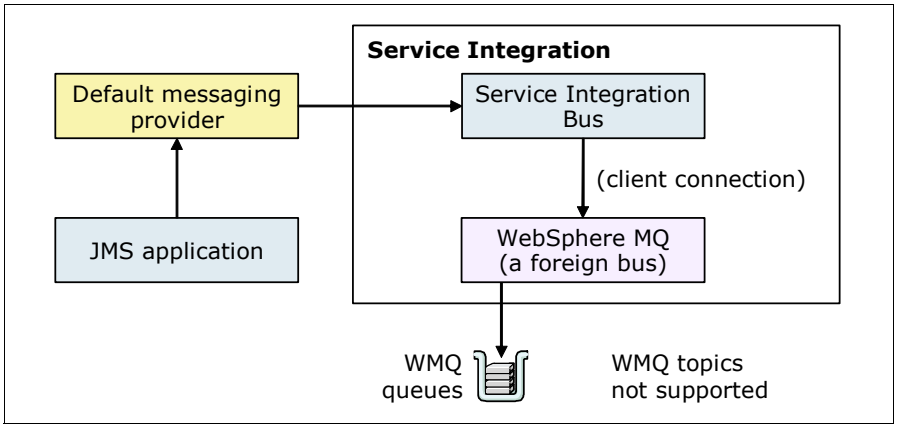


Figure 13-16 WebSphere MQ server as a bus member

13.5.5 Third-party messaging provider (generic JMS)

The third-party messaging provider is the catch-all for any external messaging providers other than WebSphere MQ. WebSphere Application Server works with any JMS-compliant messaging provider after it is defined to WebSphere. However, administrative support is limited.

Consider using a third-party messaging provider only if you have an existing investment in a one already. Keep in mind that much greater support is available in the WebSphere Application Server default messaging provider and WebSphere MQ messaging provider.

13.5.6 Application design for retrieving messages

WebSphere Application Server applications can be producers or consumers of messages. When the application is the consumer, it needs a way of receiving messages sent to it. This section addresses two different ways an application can receive messages.

Using JMS interfaces: Explicit polling for messages

Applications can use JMS to explicitly poll for messages on a destination, and then retrieve the messages for processing by business logic beans (enterprise beans). Figure 13-17 shows an enterprise application that is polling a JMS destination to retrieve an incoming message. The application then processes it with a business logic bean. The business logic bean uses standard JMS calls to process the message to extract data, or to send the message to another JMS destination.

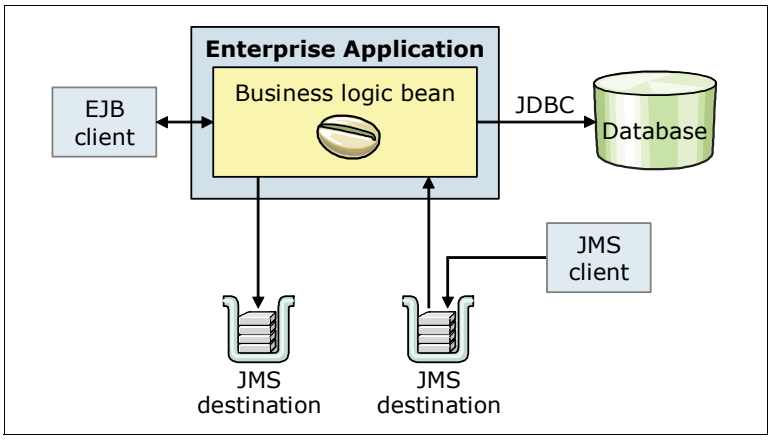


Figure 13-17 Using JMS as asynchronous messaging

WebSphere Application Server applications can use standard JMS calls to process messages, including any responses or outbound messaging. A response can be handled by an enterprise bean that acts as a sender bean, or in the enterprise bean that receives the incoming messages. Optionally, this process can use two-phase commit within the scope of a transaction.

Using a message-driven bean: Automatic message retrieval

WebSphere Application Server supports the use of MDBs as asynchronous message consumers. Figure 13-18 shows an incoming message that is passed automatically to the `onMessage()` method of an MDB that is deployed as a listener for the destination. The MDB processes the message, in this case forwarding the message to a business logic bean for business processing.

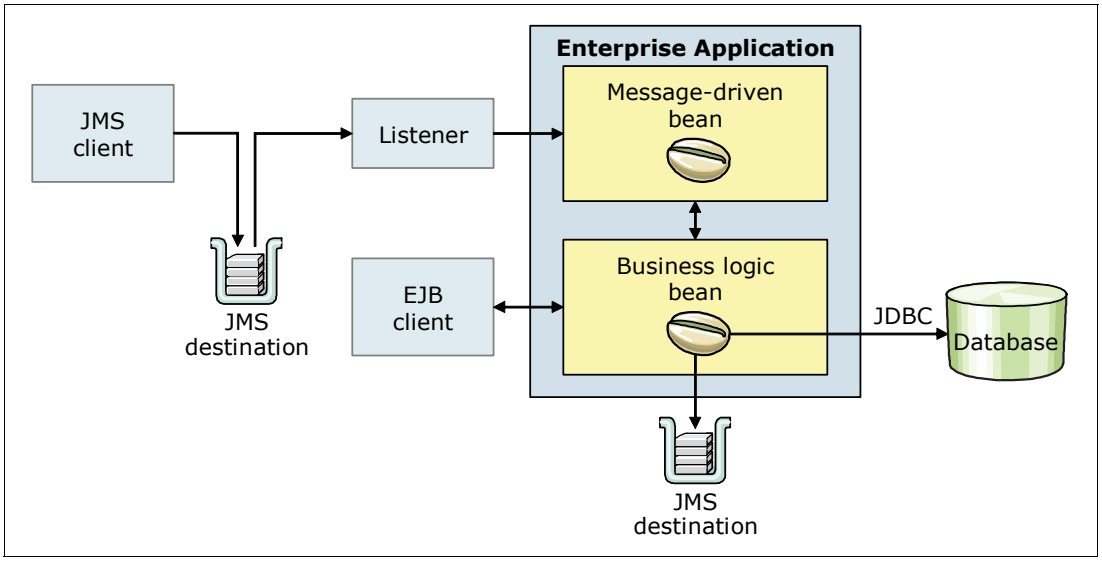


Figure 13-18 Using MDB as asynchronous messaging

MDBs can be configured as listener on JCA 1.5 resource adapter or against a listener port. With JCA 1.5 resource adapters, MDBs can handle generic message types, not just JMS messages. Having this ability makes MDBs suitable for handling generic requests inbound to WebSphere Application Server from enterprise information systems through the resource adapter. In the JCA 1.5 specification, such MDBs are commonly called *message endpoints* or *endpoints*.

13.6 Messaging topologies

Choosing a topology depends largely on the answers to questions about the topology of the application and your own messaging requirements. Consider the following important questions:

- What is the topology of your application?
- Can you break it up into logical parts that can be separately deployed?
- Which parts need to communicate with others?
- Does the application have natural divisions that are autonomous, needing separate communication channels?
- Does the application need to communicate with external systems?
- Do you need to balance the messaging workload for each part?
- Are there any critical parts that need high availability?
- Will you need application server clustering, or do you have it already?

The following sections outline common topologies to consider. The selection depends largely on the answers to the previous questions. Multiple topologies will probably fit your needs. However, usually the simplest choice produces the best results.

Important: This section provides a high-level look at messaging topologies, focusing on the default messaging provider. Before you design anything, even the simplest topology for messaging, you must understand how the default messaging provider handles messages.

The following topologies are implemented by using the default messaging provider. They are arranged in increasing complexity.

13.6.1 One bus, one bus member (single server)

The one bus, one bus member topology is the simplest and most common topology. This topology is used when applications deployed to the same server need to communicate among themselves. Additional application servers that are not members of the bus and use bus resources infrequently, can connect remotely to the messaging engine (Figure 13-19).

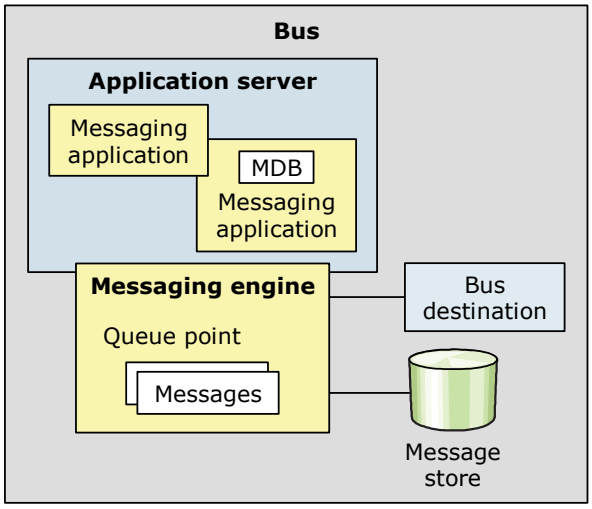


Figure 13-19 Single bus with an application server member

Although this topology is simple to set up, message producers and consumers that connect to the messaging engine remotely might experience a performance impact. Because the single messaging engine runs on a non-clustered application server, no high availability or workload management is supported.

An application can connect to a messaging engine on a bus in any of the following situations:

- The application is running on the same server as the messaging engine or another server in the same cell
- The application is running in a different cell or in a client container.
- The application uses a client connection to use the bus or in-process call.

Figure 13-20 shows the possible connections:

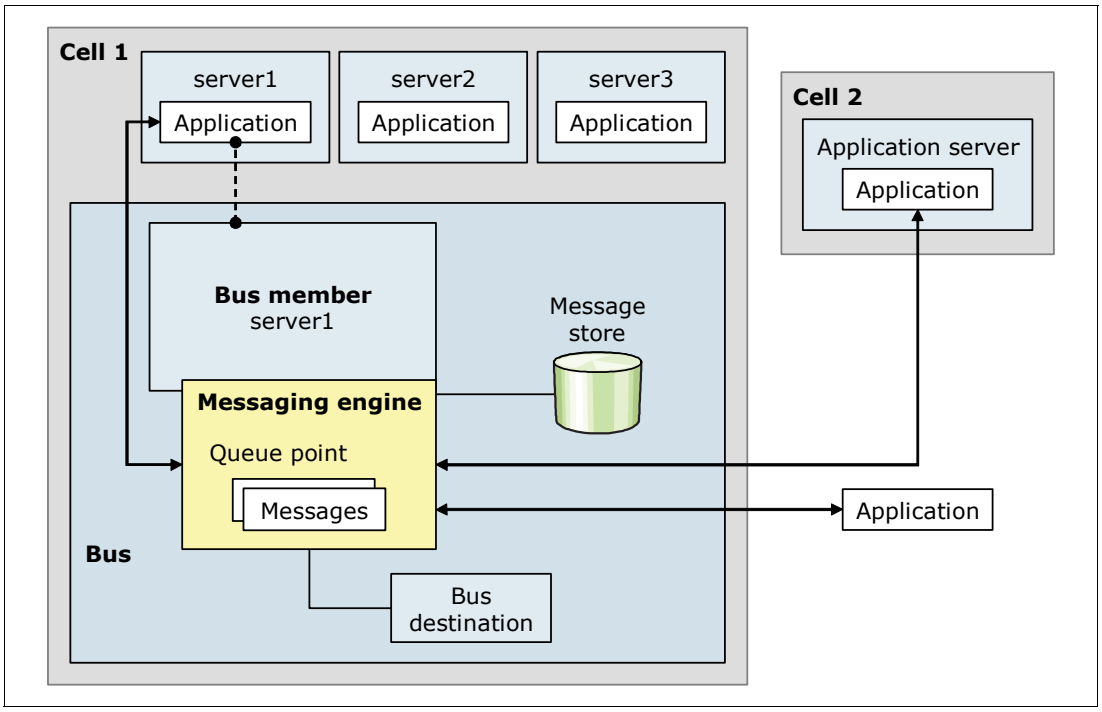


Figure 13-20 Applications connecting to a messaging engine

13.6.2 One bus, one bus member (a cluster)

With the one bus, one bus member variation, the bus member is a cluster. By default, only one application server in a cluster has an active messaging engine on a bus. If the server fails, the messaging engine on another server in the cluster is activated, which provides failover but no workload management.

The server with the active messaging engine has local access to the bus. However, the rest of the servers in the cluster access the bus remotely by connecting to the active messaging engine. Servers that access the bus remotely can consume asynchronous messages from a

remote messaging engine. However, an instance of an MDB that is deployed to the cluster can consume only from a local messaging engine (Figure 13-21). Because everything is tunneled through one messaging engine, performance might still be an issue.

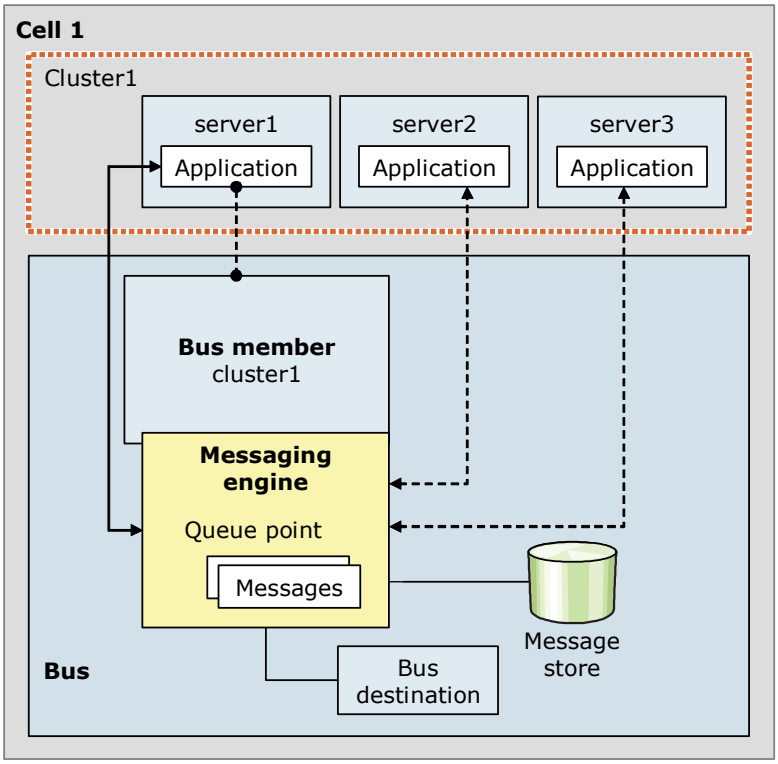


Figure 13-21 Single bus with a cluster member (high availability)

For greater control of where the messaging engine is activated, consider using a preferred server configuration. Explicitly define, for example, a primary server and a backup server in the same cluster. It is also possible to define that only preferred servers are used. This setup might circumvent the high availability advantages of the cluster if no more preferred servers are available.

With additional configuration, you can create a topology where each server in the cluster is configured to have an active messaging engine. This configuration provides workload management and failover abilities (Figure 13-22). Because messaging engines can run on any server, if one server goes down, both messaging engines run on the remaining server.

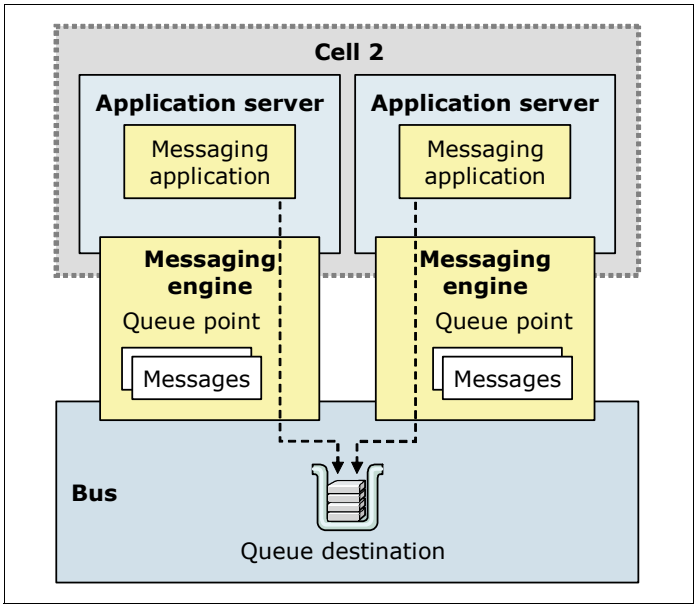


Figure 13-22 Single bus with a cluster member, providing workload management

In this topology, a queue destination assigned to the cluster is partitioned. Each active messaging engine in the cluster owns a partition of the queue. A message sent to the queue is assigned to one partition, and the messaging engine that owns the partition is responsible for managing the message. That is, requests sent to a destination can be served on any of the messaging engines that run on any of the servers in the cluster.

13.6.3 One bus, multiple bus members

In the one bus, multiple bus members topology, multiple non-clustered application servers are connected as members of the bus (Figure 13-23). In this topology, most, if not all servers are bus members. Locating the queue points on the same application server as the messaging application that is the primary user of the queue maximizes the use of local connections. This configuration enhances performance.

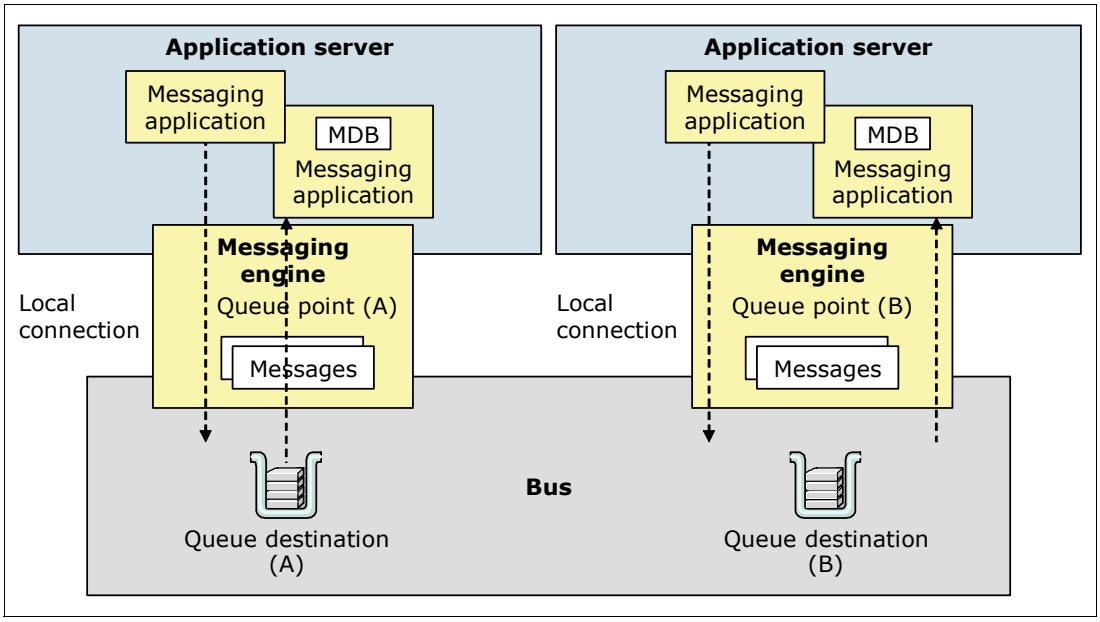


Figure 13-23 Single bus with multiple application server members

13.6.4 Multiple buses

Many scenarios require simple bus topologies, even just a single server. When integrating applications are deployed to multiple servers, it is often appropriate to add those servers as members of the same bus. However, servers do not have to be bus members to connect to a bus. In more complex situations, multiple buses can be interconnected to create more complicated networks.

A bus cannot expand beyond the edge of a cell. When you need to use messaging resources in multiple cells, you can connect the buses of each cell to each other. An enterprise might also deploy multiple interconnected buses for organizational reasons. For example, an enterprise with several autonomous departments might want separately administered buses in each location. Alternatively, separate but similar buses exist to provide test or maintenance facilities.

If you use messaging resources in a WebSphere MQ network, you can connect the bus to the WebSphere MQ network, where it appears to be another queue manager. This approach is achieved through the user of an WebSphere MQ link.

Figure 13-24 illustrates how a bus can be connected to another bus and to a WebSphere MQ network. The remote buses are considered to be foreign buses.

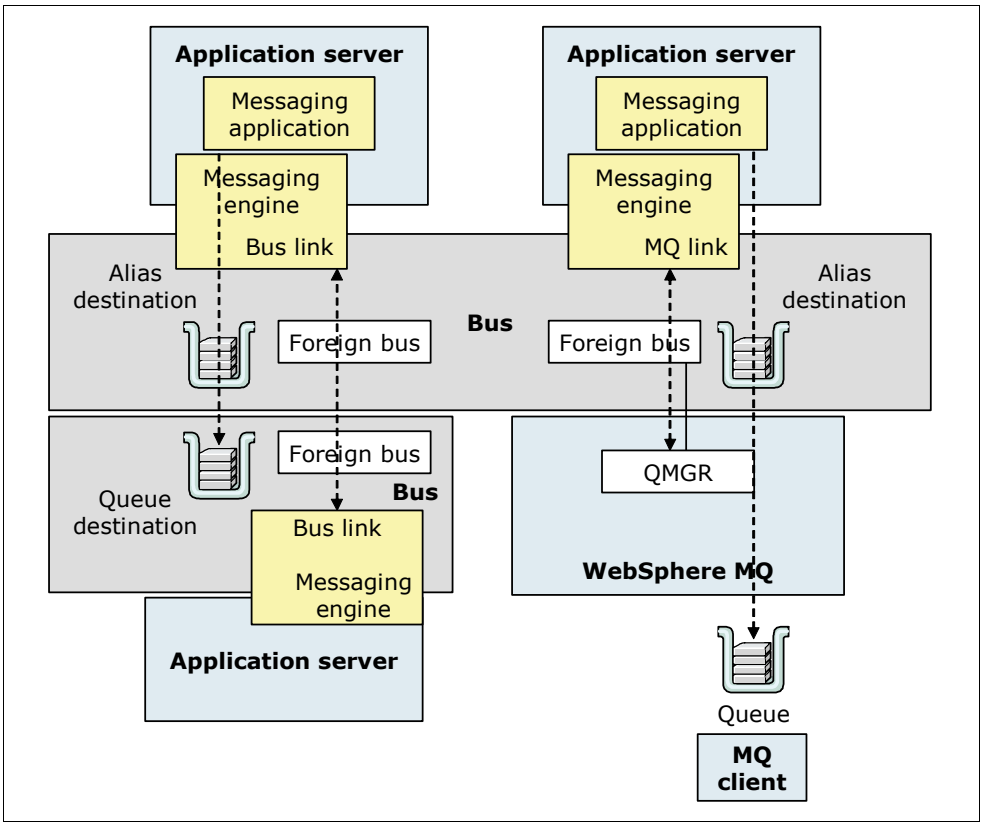


Figure 13-24 Multiple bus scenario that connects through the WebSphere MQ network

For the connection between the two service integration buses, each messaging engine contains a service integration bus link configuration. The configuration defines the location of the messaging engine on the remote bus. For the WebSphere MQ connection, the messaging engine contains an WebSphere MQ link configuration. This configuration defines the queue manager on WebSphere MQ, and identifies a queue manager name for the WebSphere MQ network.

When an application sends a message to a queue on the remote bus, it can send it to an alias destination. This destination is defined on the local bus that points to the queue destination on the second bus. Because there is a single link to a foreign bus, there is no workload management capability. It is important to note that an application cannot consume messages from a destination in a foreign bus.

13.6.5 Connecting to WebSphere MQ on z/OS

A second option for connecting to WebSphere MQ is to create a WebSphere MQ server definition. This definition represents a queue manager or queue sharing group on WebSphere MQ running on z/OS (Figure 13-25). The WebSphere MQ server defines properties for the connection to the queue manager or queue sharing group. With Websphere Application Server V8.5, this construct can also be applied to distributed (non z/OS platforms) queue managers.

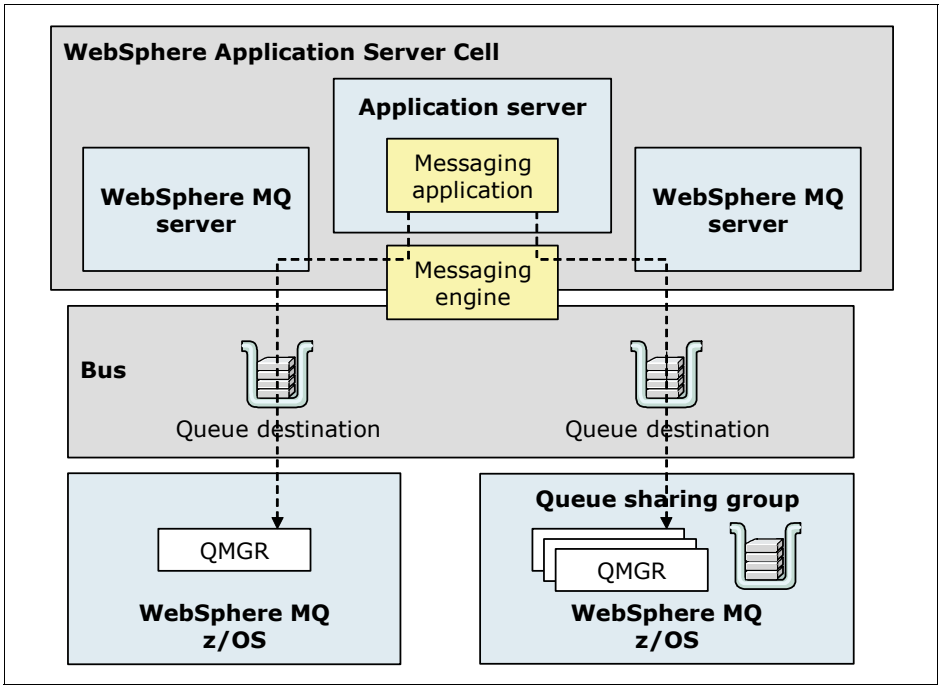


Figure 13-25 Multiple bus scenario that uses a WebSphere MQ server definition

When you add a WebSphere MQ server as a member of the bus, the messaging engines establish connections to that WebSphere MQ server to access queues on WebSphere MQ.

To the WebSphere MQ server, the WebSphere MQ queue manager or queue sharing group is regarded as a mechanism to queue messages for the bus. The WebSphere MQ server is regarded by the WebSphere MQ network as another WebSphere MQ client that attaches to the queue manager or queue sharing group.

WebSphere MQ server provides the following advantages over a WebSphere MQ link:

- With the WebSphere MQ server, applications can use the higher availability and optimum load balancing provided by WebSphere MQ on z/OS.
- With WebSphere MQ link, messages from WebSphere MQ are delivered to a queue destination in the bus. When a messaging engine fails, messages at destinations in the messaging engine cannot be accessed until that messaging engine restarts. When you use a WebSphere MQ server that represents a queue sharing group, the bus can continue to access messages on the shared queue. This access occurs even when a queue manager in the queue sharing group fails. This access is possible because the bus can connect to a different queue manager in the queue sharing group to access the shared queues.
- Messages are not stored within the messaging engine. Messaging applications directly send and receive messages from the queues in WebSphere MQ, making the WebSphere

MQ server tolerant of a messaging engine failure. With this process, message beans can be configured to immediately process messages as they arrive on a WebSphere MQ queue. Similarly, any bus mediations take place immediately upon a message that is displayed in a WebSphere MQ queue.

- With WebSphere MQ link, applications must push messages from the WebSphere MQ network end of the link. With WebSphere MQ server, applications can pull messages from the WebSphere MQ network. WebSphere MQ server, therefore, provides a better proposition than WebSphere MQ link in situations that require optimum load balancing.

13.7 Security and reliability of messaging features

This section provides information about some of the details and requirements of messaging. It addresses security, high availability, and reliability, which are important points in any planning.

This section includes the following topics:

- Planning for security
- Planning for high availability
- Planning for reliability

13.7.1 Planning for security

Messaging security has two main areas:

- Authorization and authentication of users and groups that want to connect to a bus
- Securing the transportation of the message from source to destination.

Authentication and authorization

All access to a service integration bus must be both authorized and authenticated if bus security is turned on. Authentication is done through an external access registry, such as a Lightweight Directory Access Protocol (LDAP) server, a custom database, or the local operating system. The user or group must have their credentials validated before they can access the bus.

After the user or group is authenticated, they must still be authorized to access bus resources. The user or group must be assigned to the *bus connector role*. Otherwise, they are denied access even if their credentials are valid.

The following roles also affect permissions for users and groups:

- Sender: The user/group can send (produce) messages to the destination.
- Receiver: The user/group can read (consume) messages from the destination.
- Browser: The user/group can read (non-destructive) messages from the destination.

When considering authentication and authorization, address the following questions:

- What users or groups, or both, do you need to define or have already been defined?
- What are the minimum permissions you need to assign to each one?

Secure message transportation

A messaging engine uses a particular transport chain to connect to a bus and exchange a message with another messaging engine. The transport chains have attributes such as security encryption (SSL or HTTPS) and the communication protocol used (TCP/IP).

Encryption is more secure, but can have performance impacts. The same is also true for protocols, although your choice of protocol is usually decided for you by what you are trying to communicate with. For each bus, select the particular transport chains that have the attributes you need.

Ask the following questions when designing secure message transportation solutions:

- What types of messages do you need secured?
- Where do you need to use encryption, and to what extent?
- What are the connection requirements (in terms of security) of the party you are trying to communicate with?

13.7.2 Planning for high availability

An application server has only one messaging engine for each bus of which it is a member. No option is available for failover. An application server that is clustered will, by default, have one active messaging engine. If the server that hosts the messaging engine fails, the messaging engine is activated on another server in the cluster.

To ensure that the messaging engine runs on one particular server in the cluster, specifically configure it by defining the preferred server for the messaging engine. For example, consider a situation where you have one primary server and one backup server. Another example is if you want the messaging engine to run only on a small group of servers within the cluster.

Each messaging engine on a bus belongs to one high availability group. A policy assigned to the group at run time controls the members of each group. This policy determines the availability characteristics of the messaging engine in the group, and is where preferred servers are designated. Be careful not to reduce or remove the high availability of the messaging engine by having a list of preferred servers that is too restrictive.

To obtain workload management across a bus with a cluster, create additional messaging engines and assign the messaging engines to a preferred server. The messaging engines run simultaneously with queues partitioned across them.

13.7.3 Planning for reliability

The JMS specification supports two modes of delivery for JMS messages:

- Persistent
- Non-persistent

The WebSphere administrator can select the mode of delivery on the JMS destination (queue/topic) configuration:

- Application (persistence determined by the JMS client)
- Persistent
- Non-persistent

Messages can also have a quality of service (QoS) attribute that specifies the reliability of message delivery. Different settings apply depending on the delivery mode of the message. The reliability setting can be specified on the JMS connection factory and, for the default messaging provider, on the bus destination. Reliability settings set at the connection factory

apply to all messages that use that connection factory. However, you can set the reliability settings individually at the bus destination. Each reliability setting has different performance characteristics. You can select from the following settings:

- Best effort non-persistent
- Express non-persistent
- Reliable non-persistent
- Reliable persistent
- Assured persistent

You must consider the trade-off between reliability and performance. Increasing reliability levels of a destination decrease the performance or throughput of that destination. A default setting is configured when the destination is created, but this setting can be overridden by message producers and consumers in certain circumstances.

For more information, see WebSphere Application Server V8.5 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp`

Search for the following topics:

- *JMS delivery mode and service integration quality of service*

  This topic includes a table that outlines what happens to a message under various circumstances. These circumstances depend on delivery mode and reliability setting.

- *Reliability*

  The following questions apply here:

  - What is more important for each type of message: Reliability or performance?
  - How heavy is the workload for the messaging engines?
  - What are the implications of message loss due to server failure?
  - What is the performance expectation?

Selecting a message store type

Another consideration is the message store that each messaging engine employs. A message store is where the messages are persisted according to the reliability levels of the messages. The message store and the reliability levels directly affect the performance of the messaging engine.

Message stores can be implemented as either of the following types:

- File stores (flat files)

  File stores are flat files that can be administered by the local operating system. They are the default type of message store. File stores are generally faster and cheaper than data stores because of the absence of the database. File stores have no extra licensing fees, fewer administration costs, and no database administrator.

- Data stores (tables inside a database)

  Data stores are the equivalent of file stores, but are implemented inside a relational database as a series of tables. They are administered by the facilities provided by the database. You can use any supported database product. Data stores might be preferable for larger organizations with an existing database infrastructure and skills.

Both types of message stores can be subject to security, such as file system or database encryption and physical security access.

13.8 Planning checklist for messaging

Use the following checklist to guide you as you plan for messaging:

- Determine if and how messaging will be used.
- Choose a JMS messaging provider (default messaging, WebSphere MQ, or generic).
- Design a messaging topology. If using the default messaging provider, determine the number of buses to be used and if connections to other buses or WebSphere MQ are required.
- Determine what destinations (queues, topics) are required initially, and the reliability levels for those destinations.
- Determine the type of message data store to use.
- Design a security strategy for messaging:
  - Bus security
  - Transport security
- Plan for high availability. If you are clustering application servers, decide whether to use one messaging engine (high availability) or multiple engines (workload management).

14

Web services

This chapter addresses traditional and RESTful web services. It highlights the decisions that administrators make when planning to use these services on a Websphere Application Server V8.5 architecture.

This chapter includes the following sections:

- Overview of web services
- Considerations when using web services
- Web services architecture
- Support for web services in WebSphere Application Server
- RESTful web services
- Planning checklist for web services
- Resources

14.1 Overview of web services

Web services are self contained, modular applications that can be described, published, located, and started over a network. More specifically, a web service can be an application or function that can be programmatically started over the Internet. For example, buyers and sellers all over the world can discover each other, connect dynamically, and run transactions in real time with minimal human interaction.

Web services have the following properties:

- Web services are self contained. No support beyond Extensible Markup Language (XML) and SOAP is required on either the client or server sides to realize a web service
- Web services are self describing. The definition of the message format travels with the message itself. No external metadata repositories are needed
- Web services can be published, located, and started across the Internet
- Web services use existing network infrastructure and Internet standards such as HTTP
- Web services are modular. Simple web services can be chained together or grouped into more complex services to run higher-level business functions
- Web services are interoperable across platforms, and are language independent. The client and the server can be on different platforms, on different systems, or in different countries. The language used has no restrictions if it supports XML and SOAP
- Web services are based on mature and open standards. The major underpinning technologies, such as XML and HTTP, were developed as open source standards themselves, with no proprietary technologies. As such, they are widely used and understood
- Web services are dynamic, loosely coupled and are easily reconfigured into new services. Therefore, web services must be able to be dynamically discovered in an automated fashion. This feature allows additions and changes to be implemented with minimal impact to other web service clients
- Web services can wrap existing applications with a programmatic interface. Older applications can implement a web service interface, extending the life and usefulness of these applications

Web services promote component reusability and a service-oriented approach to development. Thus, they are commonly used as part of a service-oriented architecture (SOA).

SOA is an approach to building enterprise applications that focuses on services, or loosely coupled components, that can be composed dynamically. With the SOA approach to application architecture, existing applications can be converted to services that can be used by new or existing applications. As the architecture grows and more applications are added to this portfolio, applications can be orchestrated in flexible business workflows. Businesses can then better react to changes. These changes include the introduction of a new partner or supplier, shifts in the business model, and the streamlining of several application services into one.

Any implementation of an SOA, including web services, must have the following characteristics:

- Interoperability between platforms, systems, and programming languages
- Clear and unambiguous service interface description language
- Dynamic search and retrieval capabilities of a service at run time
- Security

14.2 Considerations when using web services

This section addresses the business and technical issue considerations when using web services. The questions listed here represent the strategic thinking that needs to happen if you want to provide or use web services.

14.2.1 Business issues

The following business issues might affect your decision about the use of web services:

- Do you have business functionality that is common and can be shared?

  The typical reason to use a web service is to save time and effort by reusing existing infrastructure. Over time, with reuse, the entire IT infrastructure of an enterprise can reduce redundancy and consist of mature, well-tested components. Does your application have this functionality? Can you reduce the complexity of your application by using other web services?

- Do you need a more consumable interface for an existing exposed function?

  You can use web services as an easier way to expose application programming interfaces (APIs) to consumers. Wrapping existing APIs in web services provides a more friendly interface to users.

- What business functionality do you want to expose to external parties?

  You can expose as much or as little of your application as you want. This exposure can range from single business functions exposed as services to the entire application wrapped as a single web service. The exposure depends largely on your business strategy. There are no technical constraints. Does the architecture of your application allow individual business functions to be exposed in this manner?

- Do you need to promote your business functions in a common and non-proprietary way?

  Web services offer a common, non-proprietary level of abstraction between the client and the service provider. The key benefits here are that the client can easily discover and use business services that you provide. Web services generate goodwill and business opportunities, and give you the flexibility to alter or replace the back-end logic without the knowledge of the client. The importance of this function varies with the type of clients that are targeted. What do you know about your potential clients? Are your clients internal or external to your enterprise? Is there a limited set of clients?

14.2.2 Technical issues

The following technical issues might affect your decision about the use of web services:

- Does the business logic you want to expose have state dependency?

  If you intend to expose your application over the Internet, you can use the HTTP communications protocol. HTTP is a stateless protocol with no guarantees about message delivery, order, or response. It has no knowledge of prior messages or connections. Multiple request transactions that require a state to be maintained (for example, for a shopping cart or similar functionality) need to address this shortcoming. The solution is to use messaging middleware based on Java Message Service (JMS) or other protocols that provide for the maintenance of state.

  The bottom line is that you need to consider stateful web services carefully. Keep web services as simple and stateless as possible.

- Do you have stringent non-functional requirements?

  The basic mechanisms that underlie web services have been around for some time. However, newly adopted standards, such as security and transaction workflows, are still in flux with varying levels of maturity. Ensure that you use only industry-adopted standards. This consideration might influence your decisions about candidate business functions for web service enablement.

  For information about the status of the different available web services standards, see the following IBM developerWorks topic:

  `http://www.ibm.com/developerworks/webservices/standards/`

- What are you using web services for?

  Web services provide interoperability, not performance. Use web services in the context of providing exposure to external parties, not internally in the place of messaging between parts of your application. Web services use XML to represent data as human readable text for openness and interoperability. When compared to a binary format, web services are inefficient, especially where the use of parsers and other post-processing are required.

14.3 Web services architecture

The web services architecture is determined by the W3C Web Service Architecture Working Group. This section highlights the components of the architecture and explains how to use the architecture.

14.3.1 Components of the architecture

The basic SOA consists of the following primary components:

- The *service provider* creates a service, optionally publishes its interface, and provides information to the service broker. Another name for the service provider is the *service producer*. The terms are interchangeable
- The *service requestor* locates entries in the broker registry by using various find operations. It then binds to the service provider to start one of its services. Another name for the service requestor is the *service consumer*. The terms are interchangeable
- The *service broker* is responsible for making the service interface and implementation access information available to any potential service requestor. The service broker is not necessary to implement a service if the service requestor already knows about the service provider by other means

A component can act as both a service provider and service requestor. For example, if a service provider needs more information that it can acquire only from some other service, it acts as a service requestor. It still serves the original request during this process.

Figure 14-1 shows the operations that each SOA component can run. This example illustrates publishing, discovery, and request and response operations.

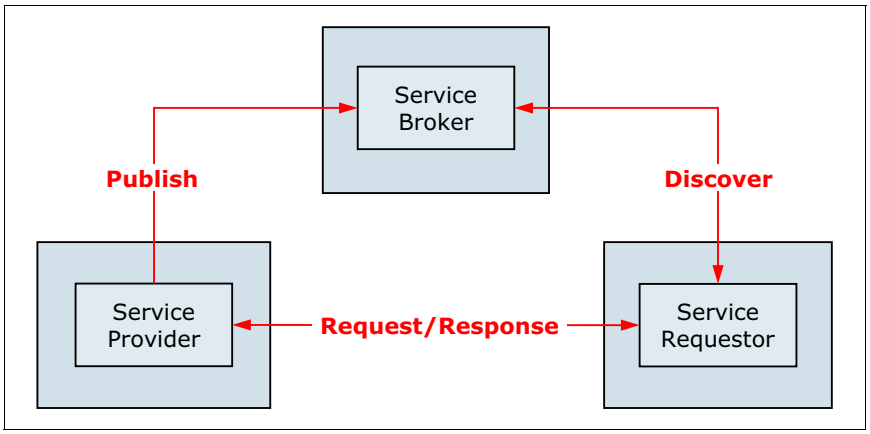


Figure 14-1 SOA components and operations

Before looking at the architecture from a web services-specific view, be familiar with the following terms:

- *XML* is a generic language that can be used to describe any content in a structured way, separated from its presentation to a specific device
- *SOAP* is a network, transport, and programming language. It is also a platform-neutral protocol that enables a client to call a remote service. The message format is XML
- *Web Services Description Language (WSDL)* is an interface based on XML and implementation description language. The service provider uses a WSDL document to specify the operations that a web service provides, the parameters, and the data types of these operations. A WSDL document also contains service access information
- *Universal Description, Discovery, and Integration (UDDI)* is a client-side API and a server implementation based on SOAP. It can be used to store and retrieve information about service providers and web services
- *Web Services Invocation Language (WSIL)* is a specification based on XML about how to locate web services without using UDDI. However, WSIL can be also used with UDDI, and does not necessarily replace it

Figure 14-2 shows a lower-level view of an SOA with specific components and technologies. The UDDI and WSIL, separately or together, become the service broker.

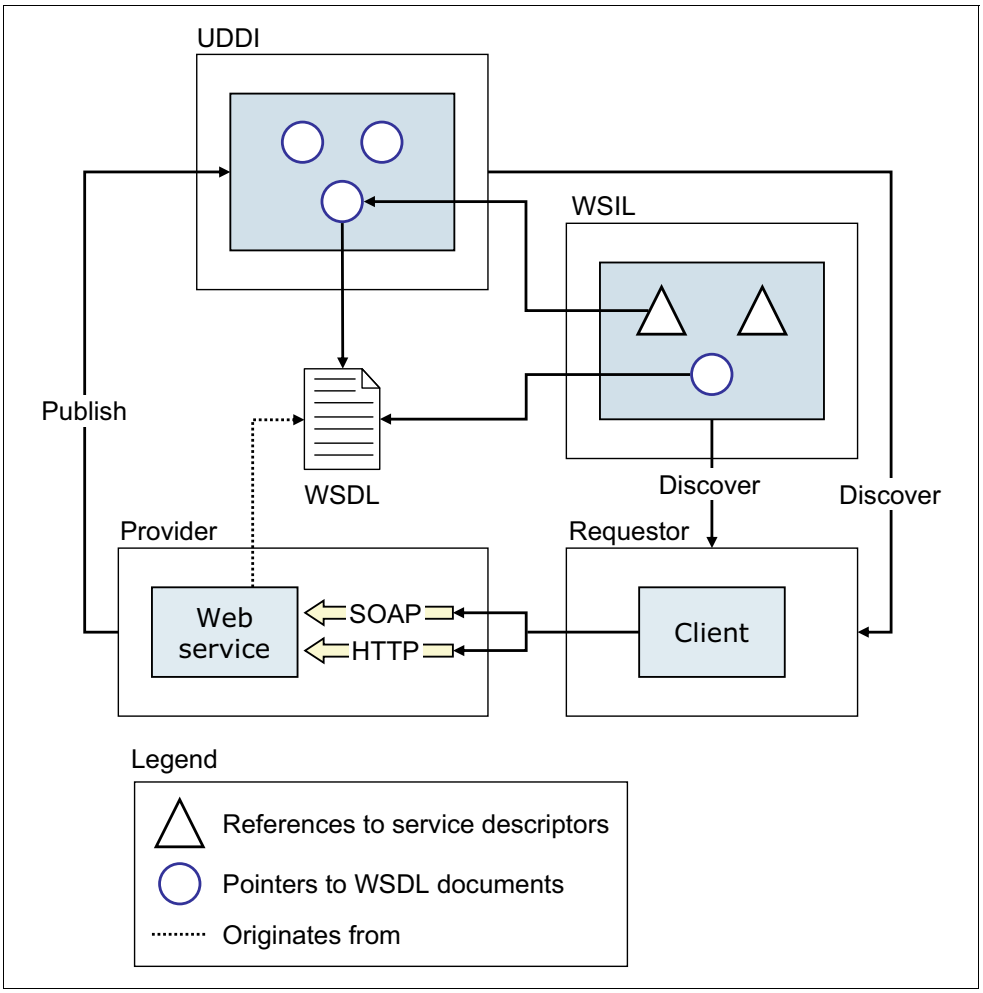


Figure 14-2 Main building blocks in an SOA approach based on web services

14.3.2 How to use this architecture

This section highlights the common message exchange patterns, often called *interaction patterns*, that you can employ. These patterns use the web services architecture that was explained in the previous section. However, some of these patterns might affect the type of transport that you use and whether you even need to use a web service.

This section addresses other options important to an administrator. These options include the use of web service gateways to implement logging and other functions at an infrastructure level.

Message exchange patterns

Some transport protocols are better adapted to some message exchange patterns than others. For example, when using SOAP/HTTP, a response is implicitly returned for each request. An asynchronous transport, such as SOAP/JMS, is probably more proficient at handling a publish-subscribe message exchange pattern.

The remainder of this section provides information about the following common message exchange patterns in the context of web services and considerations for their use:

- One-way simple message exchange pattern
- Asynchronous two-way message exchange pattern
- Request-response message exchange pattern
- Workflow-oriented message exchange pattern
- Publish-subscribe message exchange pattern
- Composite message exchange pattern

One-way simple message exchange pattern

In the one-way simple message exchange pattern, messages are pushed in one direction only, as shown in Figure 14-3. Whether the destination accepts the message, with or without error conditions, is not important to the source. The service provider or service producer implements a web service to which the requestor or consumer can send messages. This pattern is a candidate to use messaging instead of a web service, depending on your interoperability and reliability requirements.

An example of a one-way message exchange pattern is a resource monitoring component. Whenever a resource changes in an application, called the source, a new value is sent to a monitoring application called the destination.

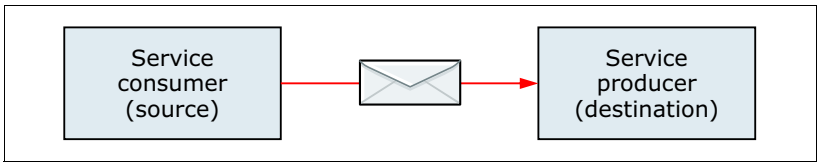


Figure 14-3 One-way messaging exchange pattern

Asynchronous two-way message exchange pattern

Figure 14-4 shows an asynchronous two-way message exchange pattern. The service requestor expects a response, but the messages are asynchronous in nature. Asynchronous here means that the response might not be available for many hours. Both sides must implement a web service to receive messages. In general, the web service provided by the Service 2 Producer component must relate a received message to the corresponding message sent by the Service 1 Consumer component.

Technically this message exchange pattern is the same as the one-way pattern, but with an additional requirement. There must be an additional mechanism to associate response messages with their corresponding request message. This mechanism can occur at the application level or by using the SOAP protocol.

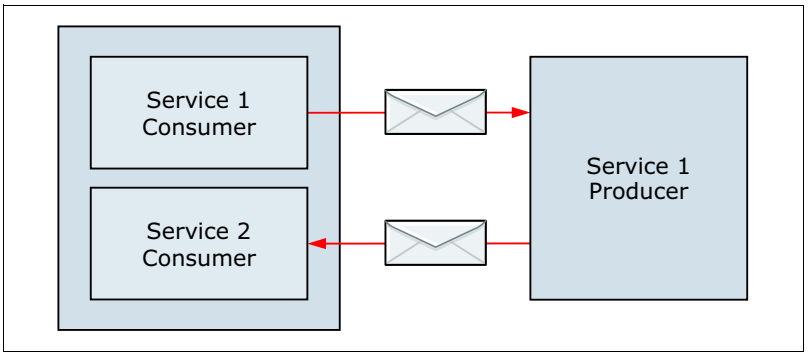


Figure 14-4 Asynchronous two-way messaging pattern

Request-response message exchange pattern

Probably the most common message exchange pattern, a Remote Procedure Call (RPC) or request-response pattern involves a request message and a synchronous response message. Figure 14-5 shows a simple request response exchange pattern. In this message exchange pattern, the underlying transport protocol provides an implicit association between the request message and the response message.

When the message exchange pattern is truly synchronous, such as when a user is waiting for a response, there is little point in decoupling the consumer and producer. In this situation, the use of SOAP/HTTP as a transport provides the highest level of interoperability. In cases where reliability or other quality of service requirements exist, such as prioritization of requests, you might need to consider alternative solutions.

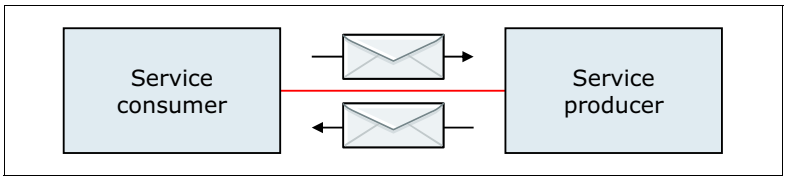


Figure 14-5 Request-response message exchange pattern

An example of this message exchange pattern is requesting an account balance on a bank account.

Workflow-oriented message exchange pattern

You can use a workflow-oriented message exchange pattern to implement a business process where multiple service producers exist. In this scenario, the message that is passed from web service to web service maintains the state for the workflow, as shown in Figure 14-6. Each web service plays a specific role in the workflow.

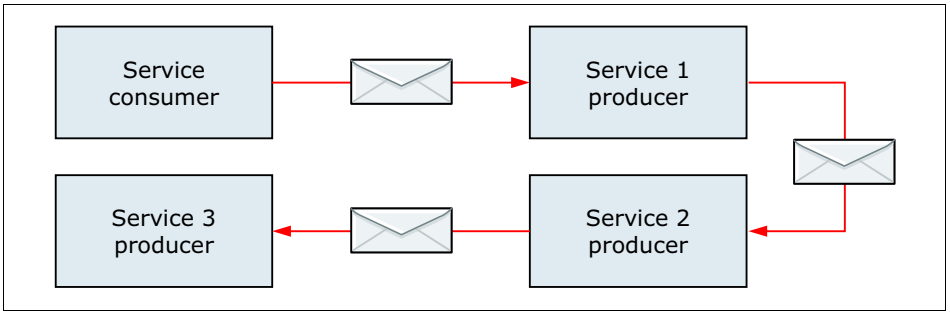


Figure 14-6 Workflow-oriented message exchange pattern

This message exchange pattern is inflexible and does not facilitate reuse. The workflow, or *choreography,* is built into each of the web services, and the individual web services can no longer be self-contained.

Publish-subscribe message exchange pattern

The publish-subscribe message exchange pattern is generally used in situations where information is pushed out to one or more parties. This pattern is also known as the *event-based* or *notification-based* pattern. See Figure 14-7 on page 459.

Implementation of this pattern at the application level is one possible architecture. Alternatively, the Service 1 Producer component can publish SOAP messages to a messaging infrastructure that supports the publish-subscribe paradigm.

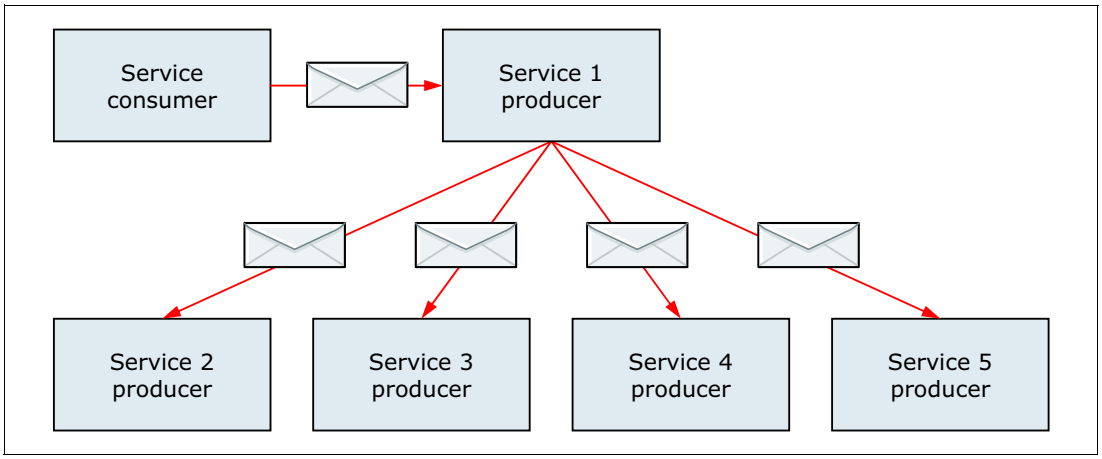


Figure 14-7 Publish-subscribe message exchange pattern

An example of a publish-subscribe message exchange pattern is a news syndication system. A news source publishes an article to the Service 1 Provider web service. The Service 1 Provider web service, in turn, sends the article to all interested parties.

Composite message exchange pattern

The composite message exchange pattern is where a web service is composed by making requests to other web services. The composite service producer component controls the workflow and generally includes business logic (Figure 14-8).

This pattern is a more flexible architecture than the workflow-oriented message exchange pattern because all of the web services are self-contained. The composite service producer component might be implemented in the conventional manner, or can be implemented by using a business process choreography engine.

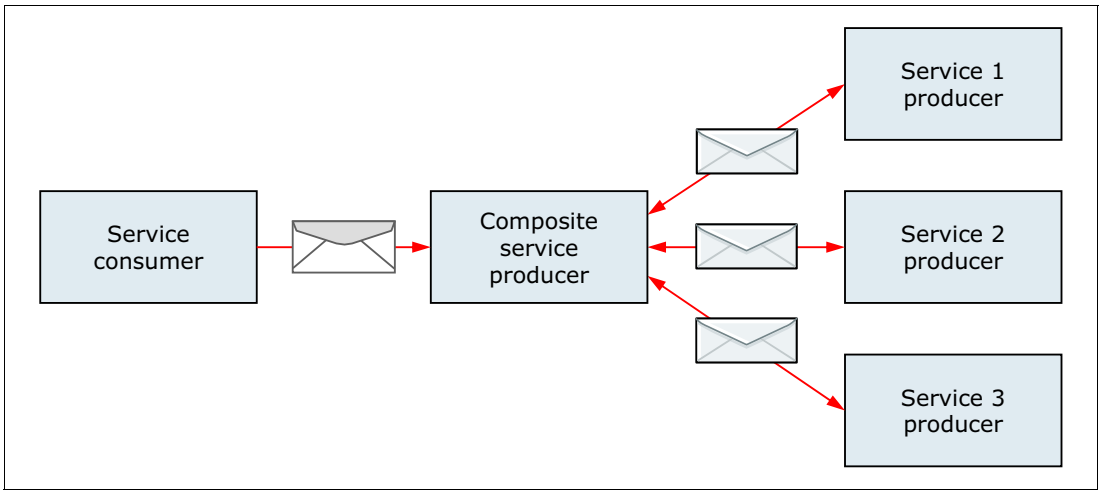


Figure 14-8 Composite message exchange pattern

An example of a composite message exchange pattern is an online ordering system. The service consumer represents a business partner application that places an order for parts. The composite service provider component represents the ordering system that is exposed as a web service to consumers and business partners through the Internet. The business process might involve using the Service 1 to check for the availability of parts in the warehouse. It might use Service 2 to verify the credit standing of the customer. It might also

use Service 3 to request delivery of the parts to the customer. Some of these services might be internal to the company, and other services might be external.

SOAP processing model

At the application level, a typical web service interaction occurs between a service consumer and a service provider, optionally with lookup to a service registry. At the infrastructure level, additional intermediary SOAP nodes might be involved in the interaction (Figure 14-9).

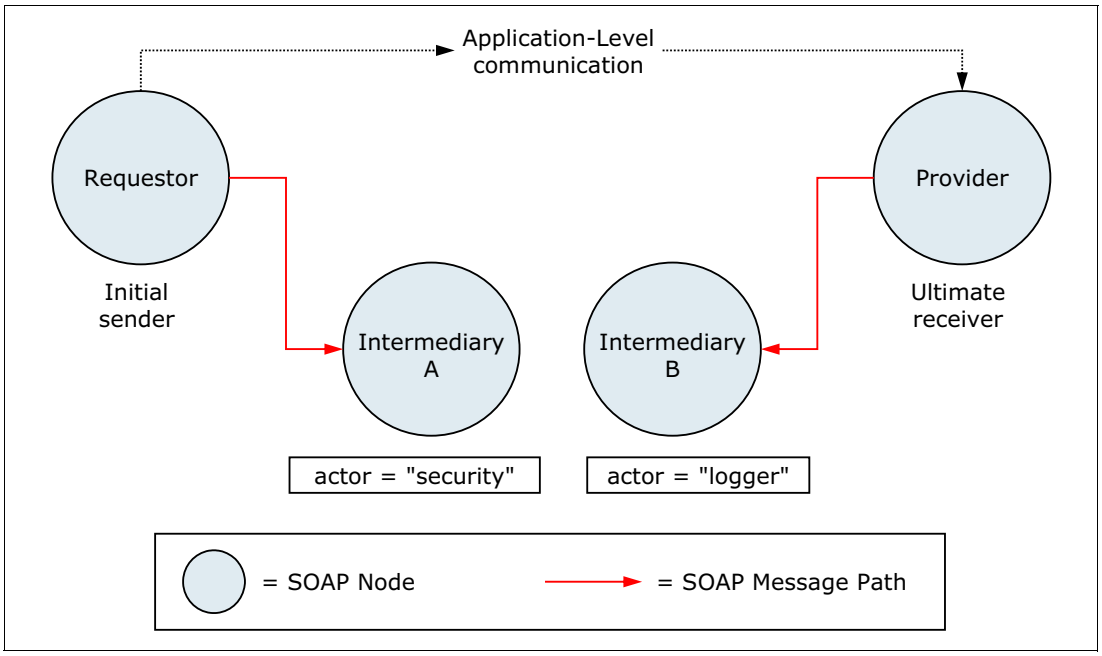


Figure 14-9 SOAP processing model

These intermediary nodes might handle quality of service and infrastructure functions that are non-application specific. Examples include message logging, routing, prioritization, and security. In general, intermediaries do not alter the meaning of the message body.

A typical situation where you need to use intermediary SOAP nodes is where you want to externally expose an existing internal web service implementation within your enterprise. New requirements might be associated with requests that originate from outside of your organization. These requests might be additional interoperability requirements, increased security requirements, auditability of requests, or contractual service-level agreements. These requirements can be implemented by using an intermediary SOAP node or a web service gateway.

Web service gateways

A *web service gateway* is a middleware component that bridges the gap between Internet and intranet environments during web service invocations. It can be used internally to provide the SOAP node functions as described previously. It can also be used at the network boundary of the organization. Regardless of where it is placed, it can provide some or all of the following functions:

- Automatic publishing of WSDL files to an external UDDI or WSIL registry
- Automatic protocol or transport mappings
- Security functions
- Mediation of message structure
- Proxy server for web service communications through a firewall

- Auditing of SOAP messages
- Operational management and reporting of published interfaces
- Web service threat detection and defense

14.4 Support for web services in WebSphere Application Server

WebSphere Application Server V8.5 supports the Web Services for Java EE V1.3 specification. This specification defines the programming model and runtime architecture to deploy and look up web services in the Java EE environment. Specifically, these tasks are run in the web, Enterprise JavaBeans (EJB), and client application containers.

14.4.1 Supported standards

Web services support in WebSphere Application Server V8.5 includes a number of standards. For more information about supported standards and specifications, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=rovr_specs

14.4.2 Service integration bus

The service integration bus (bus) is the communication infrastructure that provides service integration through messaging. This administrative concept is used to configure and host messaging resources. Bus capabilities allow you to take advantage of WebSphere security, administration, performance monitoring, trace capabilities, and problem determination tools.

Using a bus that applies both to the application and to the enterprise at large has the following advantages:

- Securely externalizing existing applications

  The bus can be used to expose existing applications as web services regardless of the implementation details of the application. The applications can be deployed deep inside an enterprise, but still be available to customers or suppliers on the Internet in a standard, secure, and tightly controlled manner

- Cost savings by reuse of infrastructure

  When the bus is in place, any application that is web service-enabled can reuse this infrastructure

- Messaging support

  The bus is built around support for JMS, allowing exposure of messaging artifacts such as queues and topics as web services. There is also a provision for advanced options such as asynchronous communication, prioritized message delivery, and message persistence.

- Support for standards

  The bus is part of the Java EE 6 implementation. Therefore, it supports the following major web services standards that are also part of Java EE 6:

  – WS-I Basic Profile 1.1
  – JAX-WS (JSR-224) 2.2
  – JAX-RPC (JSR-101) 1.1

  - UDDI V3
  - WS-I Security
  - WS-Transaction
- Support for complex topologies

  Tight integration with the WebSphere administrative model means that complex topologies with the bus are options for use by web services. These complex topologies include clustering for high availability.

For more information about bus, see Chapter 13, “Messaging and service integration” on page 415.

14.4.3 UDDI registries

UDDI is a specification that defines a way to store and retrieve information about a business and its technical interfaces, such as web services. A UDDI registry makes it possible to discover the technical programming interfaces provided for interacting with a business for electronic commerce or information retrieval. Essentially, UDDI is a search engine for application clients rather than human beings. However, many implementations provide a browser interface for human users.

UDDI helps broaden and simplify business-to-business interaction. A manufacturer who needs to create many relationships with different customers, each with its own set of standards and protocols, would benefit from UDDI. UDDI provides a highly flexible description of services using virtually any interface. The specifications allow the efficient and simple discovery of a business and the services it offers by publishing them in the registry.

One type of implementation for UDDI is the Business Registry, which is a group of web-based UDDI nodes that form a public UDDI registry. These nodes run on separate sites provided by several companies, including IBM and Microsoft. They can be used by anyone who wants to make information available about a business or entity, or who wants to find that information.

The use of public registries has issues. Companies often do not want to show all their interfaces to the entire world. Doing so invites the world to communicate with their service with unknown and possibly malicious intent. Second, because the registry is accessible by anyone, it often possesses inaccurate, obsolete, wrong, or misleading information. No expiration dates are given for published information. Nor are any type of quality review mechanisms provided. Users of the registry are often automated processes, not humans with the intuitive ability to separate good and bad content. Therefore, these issues can cause severe problems.

In this type of situation, companies can opt for private or protected registries. A private UDDI registry can be placed behind the firewall for the internal use of the organization. A protected registry can be a public registry that is managed by the organization with access to that registry limited to previously screened users. Private registries provide control over these aspects:

- Who is allowed to explore the registry
- Who is allowed to publish to the registry
- Standards that govern exactly what information is published

Given the cleanliness of the data in a private registry, compared to a public registry, successful hit rate, for clients dynamically searching it increase dramatically.

14.4.4 Web services gateway

With web services gateway functionality, users expose an existing web service as a new service that appears to be provided by the gateway. Gateway functionality is supplied only in WebSphere Application Server Network Deployment.

The gateway can act as a single point of control for incoming web services requests. It can be used to run protocol transformation between messages. For example, it can expose a SOAP/JMS web service over SOAP/HTTP. It can also map multiple target services to one gateway service. It also can create proxy services and administer handlers for services it manages, providing infrastructure-level facilities for security and logging, among others.

Using the gateway provides the following benefits:

- A gateway service is at a different location (or *endpoint*) from the target service. This configuration makes it possible to relocate the target service without disrupting the user experience
- The gateway provides a common starting point for all web services that you provide. Users do not need to know whether they are provided directly by you or externally
- You can have more than one target service for each gateway service

14.4.5 Security

WebSphere Application Server V8.5 includes many security features for web services. Several areas can be configured within the bus to enforce security for a web service:

- WS-Security configuration and binding information specifies the level of security that is required for a web service. This security includes the requirement for a SOAP message to be digitally signed and the details of the keys involved. The WS-Security specification focuses on the message authentication model
- WS-SecureConversation provides session-based security, allowing secure conversations between applications by using web services
- The endpoint for a web service can be configured to be subject to authentication, security roles, and constraints
- The underlying transport can be encrypted (for example HTTPS)
- The bus can be configured to use authenticating proxy servers. Many organizations use these proxy servers to protect data and services
- A Java API for XML Web Services (JAX-WS) client application can be also secured by using the Web Services Security API

14.4.6 Performance

With web services comes a trade-off between performance and interoperability. Specifically in the use of XML encoding, marshalling and unmarshalling for SOAP/HTTP-bound web services. XML encoding provides a high degree of interoperability, but can also affect performance of a system.

HTTP and HTTPS-bound web services have the concept of web service *dynamic caching*. Dynamic caching requires only a configuration change to enable a significant performance improvement. No application changes are required to implement caching on either the client or server side.

When planning to apply dynamic caching, one of the main tasks is to define the service operations that are cacheable. Operations that cannot be cached, for example, are dynamic or sensitive data. This planning can be a complex task, depending on the size of the application and the number of operations that are exposed. Over a slow network, client-side caching can be especially beneficial.

For SOAP, some performance improvements can be achieved with the Message Transmission Optimization Mechanism (MTOM) standard through the optimization of the messages it provides. Avoiding the use of large messages can also help.

14.5 RESTful web services

You can use Java API for RESTful Web Services (JAX-RS) to develop services that follow Representational State Transfer (REST) principles. REST defines a set of architectural principles by which you can design web services that focus on the resources of a system. It includes how resource states are addressed and transferred over HTTP by a range of clients written in different languages.

REST has gained acceptance as a simpler alternative to SOAP and WSDL-based web services. Many Web 2.0 service providers have either deprecated or foregone SOAP and WSDL-based interfaces in favor of the easier to use, resource-oriented REST model.

WebSphere Application Server V8.5 implements technologies that support RESTful architectural principles. Many of these technologies support Ajax.

14.5.1 Ajax

Ajax is a set of techniques and technologies that are used to build rich, interactive web applications. This technology supports the development and deployment of RESTful web services. The following defining principles of Ajax ensure that user interaction with an application is fluid and continuous:

- The browser hosts an application, not content
- The server delivers data, not content

With Ajax, the browser is not a dumb terminal that can render only a web page produced by the application server. The browser is considered a client-application runtime environment that can host complex JavaScript applications.

With the Ajax technique, the web page is dynamically updated by the JavaScript code as shown in Figure 14-10. The technique does not retrieve prerendered web pages, JavaServer Pages (JSP), from the application server. Instead, the JavaScript application acts as a full fledged client application, similar in architecture to the web services model.

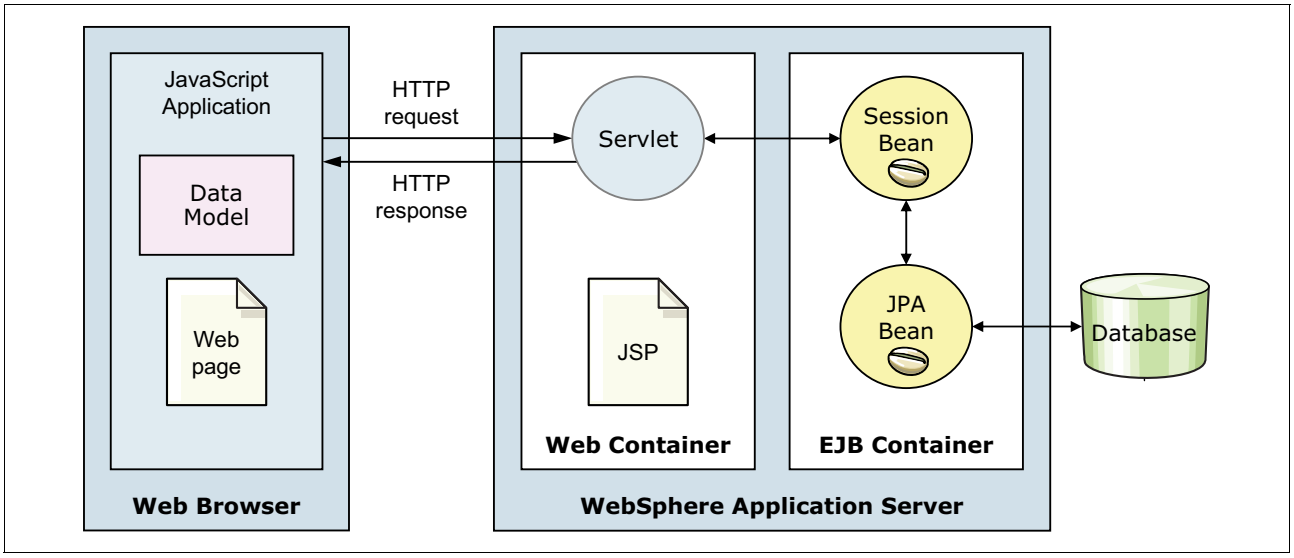


Figure 14-10 Ajax technique that moves presentation logic into the browser

Ajax is supported by Web 2.0 and Mobile V1.1.0, which provides server enhancements to support common Web 2.0 application patterns.

14.5.2 Key Ajax technologies

Ajax includes the following key technologies:

- *JavaScript* is a cross-platform, object-oriented scripting language that is supported by most browsers. Although similar in name, JavaScript is different from the Java programming language. Its syntax resembles Java, but certain syntax rules have been relaxed to make it easier to use.
- *XML* is a set of syntax rules and specifications that are used to define data. XML enables the interchange of data and structured text across dissimilar systems. XML has these characteristics:
  - Human-readable data format. It is visually similar to HTML but closer in structure to Standard Generalized Markup Language (SGML)
  - Understood by systems because its tree data structure can be easily parsed
  - Supports hierarchical (structured) data with any degree of complexity. The data represented in the structure is considered to be database neutral
- *Web services* are explained in 14.1, “Overview of web services” on page 452
- *REST* is a server-side architectural style that relies on HTTP methods (GET, PUT, POST, and DELETE) to access resources. REST is used to define flexible applications based on the notion of resources. A resource is any data that you want to share on the web that you can identify by a Uniform Resource Identifier (URI). A representation of the resource is typically a document that captures the current or intended state of a resource
- *Web Remoting* is a service-side concept that provides a web endpoint for exposing operations of enterprise Java assets. These assets include EJB and plain old Java objects (POJOs). Through the configuration of an RPCAdapter, HTTP GET and POST methods

are mapped to Java asset operations. This configuration allows JavaScript to call Java operations without having to modify back-end assets

- *JavaScript Object Notation (JSON)* is a data format that is used to exchange information between a browser (client) and a service. JSON is considered platform and language neutral. It is not a markup language like XML, because it does not use descriptive tags to encapsulate its data. JSON can be evaluated as JavaScript code. Thus, no deserialization is needed on the client.

14.5.3 Support for RESTful web services in WebSphere Application Server

Primary support for RESTful web services is provided through the WebSphere Application Server Web 2.0 and Mobile Toolkit.

The WebSphere Application Server support for Web 2.0 and Mobile Toolkit provides developers ready-to-use components. These components can extend an SOA by connecting web services and Java EE objects into interactive desktop and mobile user interfaces. With this toolkit, WebSphere Application Server applications developed initially for desktop browsers can be adapted and deployed to mobile devices such as smart phones and tablets.

An advantage of the Web 2.0 and Mobile Toolkit is that it provides an IBM supported distribution of the following open source technologies:

- *IBM Dojo Toolkit* is an open source framework to accelerate the development of cross-platform, JavaScript, and Ajax technology-based applications and websites. The Dojo toolkit has been adopted as the internal standard for IBM. The Web 2.0 and Mobile Toolkit provides the basic Dojo toolkit 1.6 libraries and several IBM extensions.
- *IBM Dojo Diagrammer* is a diagramming and graph layout widget built upon the Dojo Toolkit. It allows Ajax applications to display graphs, or networks, of nodes connected by links. The diagramming component can be run through a RESTful service or on the client by using JavaScript.
- *RESTful web services* are described in 14.5.2, “Key Ajax technologies” on page 465. For an example of a RESTful web service, see the topic “A RESTful Web service, an example” at:

  `http://www.peej.co.uk/articles/restfully-delicious.html`
- *Apache Wink* is a framework for building RESTful web services. It includes the Wink Server module and the Wink client module. The Wink Server module is a complete implementation of the JAX-RS v1.1 specification. On top of this implementation, the Wink Server module provides a set of additional features that facilitate the development of RESTful web services. The Wink Client module is a Java language-based framework that provides functionality for communicating with RESTful web services
- *Ajax Development Toolkit* is the name given to the IBM adoption of the open source Dojo toolkit. The Dojo toolkit has become a widely adopted standard for creating GUIs of RESTful web services. The adoption of the toolkit by IBM includes the following enhancements:
  - Atom data access to connect to Atom services and use Atom feeds as a data source
  - SOAP connectivity to facilitate starting public web services based on SOAP from Ajax applications
  - OpenSearch data store, which allows the invocation of any OpenSearch compliant service and then the binding of search results to Ajax application widgets
  - Atom feed widgets and gauge widgets

14.6 Planning checklist for web services

Consider the following checklist as you plan for web services:

- Determine if and how web services will be used
- Determine how web service clients will call providers. Calls can be directly, through the service integration bus, or through an enterprise service bus (ESB)
- Determine whether a web services gateway will be required
- Determine whether a UDDI service will be used. If so, decide whether you will subscribe to a public UDDI service or set up a private UDDI
- Design a security strategy for web services:
  - WS-Security for applications
  - Transport-level security
  - HTTP basic authentication
- Determine whether you will use web service dynamic caching

14.7 Resources

For more information about developing and deploying web services in WebSphere Application Server, see *IBM WebSphere Application Server V7.0 Web Services Guide*, SG24-7758. Consider having a copy of this book available as you plan your web services environment. This book is based on V7.0, and does not cover the additions and changes made to WebSphere Application Server V8 and later.

For an entry point to web services topics in the information center, see the WebSphere Application Server V8.5 Information Center at this web address. Search for the phrase *web services*:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp`

For examples of using web services in an SOA solution, see *Patterns: SOA Foundation Service Creation Scenario*, SG24-7240.

15

Security

WebSphere Application Server provides a security infrastructure and mechanisms to protect sensitive resources and address enterprise end-to-end security requirements. This chapter highlights the most important aspects that are inherent in planning security for a WebSphere Application Server installation. It provides information about the concepts and considerations to keep in mind.

This chapter includes the following sections:

- Security features in WebSphere Application Server V8.5
- Security in WebSphere Application Server
- Authentication
- User registries
- User roles in WebSphere
- Authorization
- Internal and external trusted relationships
- Security trace
- Auditing
- Securing the Liberty profile
- Resources

15.1 Security features in WebSphere Application Server V8.5

This section highlights the major security features in WebSphere Application Server V8.5.

15.1.1 Audit changes in configuration repository

WebSphere Application Server V8.5 introduces the repository checkpoints service to improve administration configuration changes. Repository checkpoints represent saved images of the repository before configuration changes are made.

To track those changes, an event is added to the security auditing component that is emitted when a checkpoint is saved in the extended repository service. For more information about extended repository and the content of this event, see 12.6, “Repository checkpoints service” on page 398.

15.1.2 SAML Web SSO Post binding profile

Security Assertion Markup Language (SAML) is a standard that is based on XML. It defines the framework for exchanging security information (assertions) between systems. It is used in single sign-on (SSO), identity federation, and web services security solutions.

Previously, to support an SAML SSO for Web applications, you had to install and configure an additional product (Tivoli Federated Identity Manager, available with limited license in WebSphere Application Server Network Deployment). In WebSphere Application Server V8.5 (also available in fix packs 7.0.0.23 and 8.0.0.4), a function enables support for SAML 2.0 HTTP post binding profile.

The SAML concepts Identity Provider (IdP) is a producer of assertions that authenticates a principal, as shown in the following examples:

- Tivoli Federated Identity Manager
- Microsoft Active Directory Federation Services
- Entrust GetAccess
- Novell Access Manager
- SAP NetWeaver Identity Management

A service provider is a consumer of assertions that relies on the identity provider to identify and provide a principal. The service provider receives an SAML Assertion containing the principle and security attributes to be used for the request.

Web SAML SSO uses the following flow, which is illustrated in Figure 15-1 on page 471:

1. A user accesses a web application, which can be on an identity provider (IdP), a service provider (SP), or elsewhere.
2. The web application redirects the user to the identity provider, and the user authenticates to the identity provider.

3. The identity provider redirects the user to an Assertion Consumer Service (ACS) in the service provider by sending an SAML response, as shown in Example 15-1.

Example 15-1 SAML response

```
<form method="post" action="https://mySP.com/TAI/SAML/POST ...>
   <input type="hidden" name=SAMLResponse" value="response" />
   <input type="hidden" name=RelayState" value="token(see next chart)" />
   <input type="submit" value ="submit">
```

4. The ACS processes the SAML response and creates the WebSphere Application Server security context.
5. The ACS adds a Lightweight Third Party Authentication (LTPA) cookie to the response, and redirects the request to the web resource (business application).
6. The web container intercepts the request. The web collaborator maps the LTPA cookie to the security context and authorizes the user's access to the requested web resource.
7. The web container sends HTTP responses back to user.

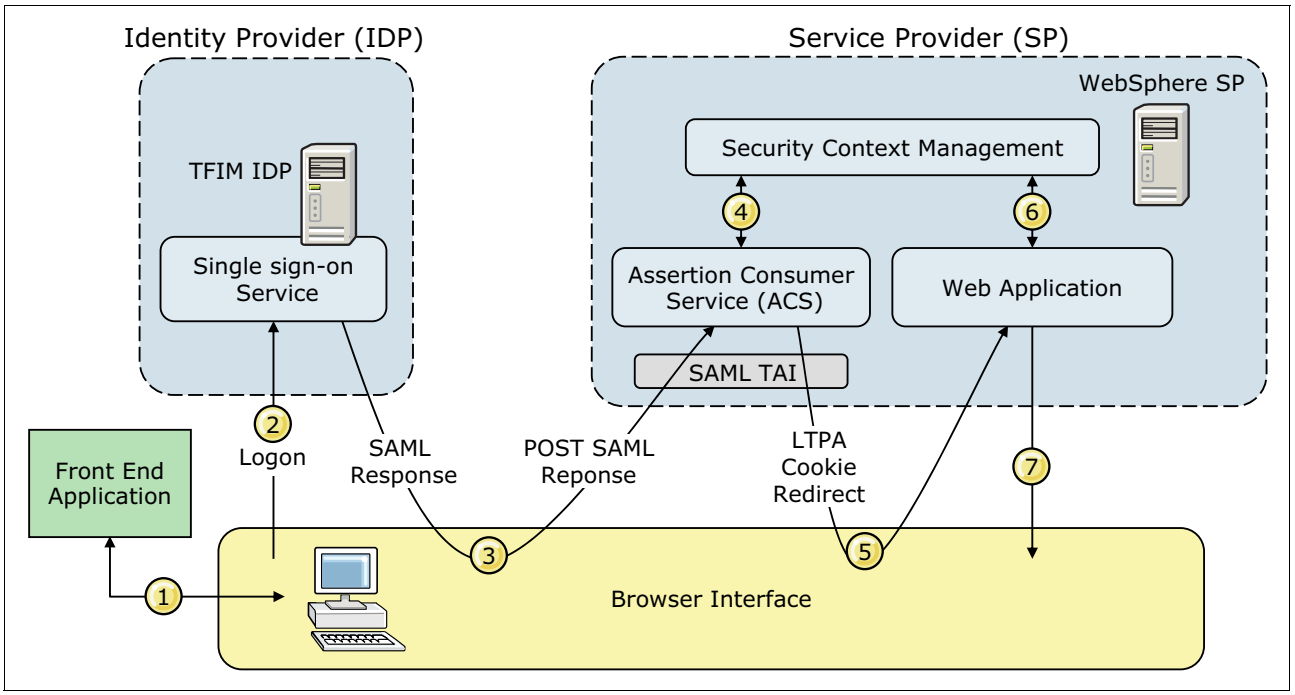


Figure 15-1 Web SAML SSO

For more information, see:

http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf

The identity provider must be present before setup and before the service provider role is implemented. An advantage of this profile is that it is based on browser requests. Thus, no connections are made from the service provider to the identity provider, which might cause problems when used behind a firewall.

The following features are available when implementing SAML service provider in WebSphere Application Server V8.5:

- Single sign-on with multiple identity providers
- Options for identity assertion, and mapping the assertion identity to the user registry of the service provider
- Mapping or asserting SAML token attributes to their realm, principal, and unique ID, and then grouping them into the service provider security context
- Plug point to allow for customized identity mapping
- Option to retrieve the group membership of the identity from the registry of the service provider and populate the security context
- Identity provider selection filter that routes the request back to the correct identity provider if the request did not come from the identity provider
- RSA-SHA1 and RSA-SHA256 signature algorithms
- Preserves the SAML token in the subject of the service provider for access by the application, and makes it available for downstream authenticated Enterprise JavaBeans (EJB) or web service call
- Business application URL can act as an AssertionConsumerService URL so that the identity provider can send a SAMLResponse directly to the business application URL
- Auditing of key SAML assertions, including Issuer and NameID

15.1.3 Security standards support

Cryptography is an important component of security in each system. As new algorithms are invented, the compute power that is needed to attack (decrypt information) has become more affordable. In addition, some protocols have proven security flaws. To protect sensitive data and provide interoperability, US government agencies have developed security standards such as NIST (FIPS 140-2, SP800-131) and NSA (Suite B).

In WebSphere Application Server V8.5, support is available in addition to FIPS 140-2 for NIST SP800-131 and Suite B (which adds extra constraints to SP800-131). WebSphere Application Server V8.5, when running in specified security standard mode, ensures that additional restrictions are put on algorithms, key lengths, and protocols that are used by SSL configuration.

Important: Most clients and servers (using older SSL implementations) might not be able to connect if one of those standards is enforced in WebSphere Application Server. Before implementing NIST SP800-131 or Suite B standard, double-check standards compatibility. Avoid a situation where the server cannot connect to Lightweight Third Party Authentication (LDAP), a database, or other servers in a mixed version cell.

WebSphere Application Server V8.5 permits SP800-131 to run in *transition* mode, which supports a mixture of old and new settings. Transition mode allows you to resolve those issues before implementing *strict* mode.

15.2 Security in WebSphere Application Server

WebSphere Application Server is part of an overall secure design principle called *defense in depth*. This principle is a military strategy that attempts to delay rather than prevent the advancement of an attacker, thus buying time by yielding space.

In computing terms, the concept is used today to increase IT protection with multiple lines of defense. By using computer security techniques at varying depths of penetration, you help mitigate the risk of the defense being compromised or circumvented. This type of security is becoming more important. Cyber attacks in which hackers work to steal credit card numbers or attempt to steal sensitive military secrets are becoming more frequent.

WebSphere Application Server is in one of the defensive layers. It takes responsibility and offers the capability to protect and defend itself in mitigating risk. Figure 15-2 illustrates these security layers and how WebSphere Application Servers fits into the layer of defense.

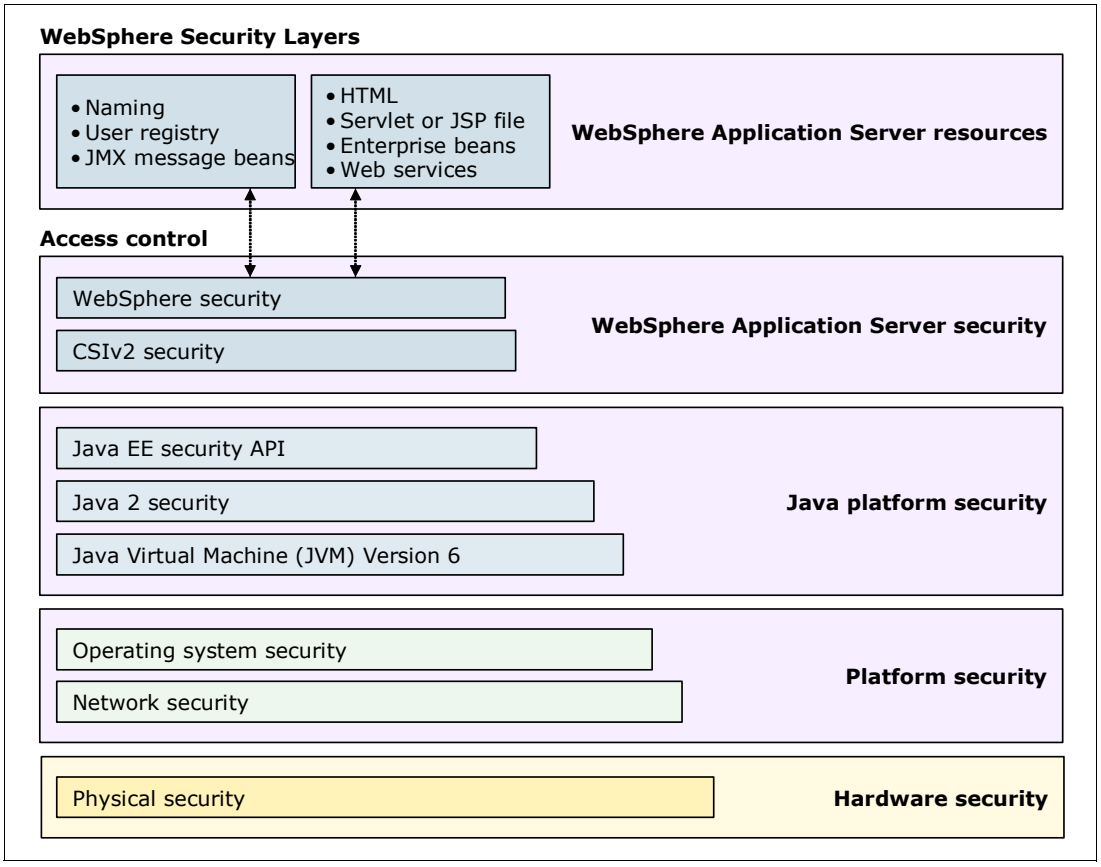


Figure 15-2 Security layers in WebSphere Application Server

WebSphere Application Server includes the following security layers (from bottom to top in Figure 15-2):

- Physical security

  Physical security encompasses the area where the environment is located. The major concerns at this level are access to the site and protection against environmental conditions. Commonly, such areas are physically secured, and access is limited to a few individuals. If an intruder can walk up to the physical server, no data on that server is secure.

- Network security

  The network security layers provide several technologies, such as firewalls, to provide a protection against network-based attacks. They are also responsible for transport level authentication, confidentiality, and integrity.

- Operating system security

  The security infrastructure of the underlying operating system provides certain security services for WebSphere Application Server. These services include access to the command-line tools and file system security support that secures sensitive files used by WebSphere Application Server. The administrator can configure WebSphere Application Server to obtain authentication information directly from the operating system user registry.

  Consider using this option only for z/OS systems. When you select the local operating system as a registry on z/OS, System Authorization Facility (SAF) works with the user registry to authorize applications to run on the server.

  If you are interested in protecting your system from applications, run WebSphere Application Server as a non-root user in distributed platforms. Set it so that access to root files and resources is not allowed. Keep in mind that, in this case, the operating system registry cannot be used.

- Java virtual machine (JVM)

  The JVM provides a set of standards-based security services for Java applications, and an installation layer between Java applications and operating system services. It provides an isolated environment for the Java application that is running in it. In this case, the application is WebSphere Application Server. In addition, the JVM protects memory from unrestricted access, creates exceptions when errors occur within a thread, and defines array types.

- Java 2 security

  The Java security model offers access control to system resources, including file system, system property, socket connection, threading, and class loading. Application code must explicitly grant the required permission to access a protected resource.

  This type of security model is called *Java 2 security*, because this type of security was first introduced in Java Version 2. It replaces the signed code and sandbox model that was used in earlier versions. Java 2 uses security policy files, which can control the access to the resources by applications. A WebSphere Application Server application has its own policy files, so that it can use files and directories on the host operating system. Also the deployed applications inside WebSphere Application Server can use their own policy files.

  Tip: When Java 2 security is disabled, access to local resources is not restricted. If you want to use the Java 2 security policies for your application, enable Java 2 security in the administrative console.

  Consider using Java 2 security only for specific situations where one or more application resources need this type of protection. Enabling Java 2 security can cause a significant impact on performance.

- Java EE security API

  The security collaborator enforces Java EE-based security policies and supports Java EE security application programming interfaces (APIs). The Java EE standard API describes a few methods with which the application can obtain the user's name and role membership. WebSphere Application Server never returns the password of any user to anyone using the API methods.

The Java EE security policy describes how application resources are accessed. The developer, when creating the application, has no information about real users of the application. Instead, the developer defines *user roles*, for example a client, clerk, or manager. During development, the user roles are mapped to access rights. For example, the user in the clerk role is allowed to access the `registerNewClient` method. The rule set is stored in the descriptor files of the application. Then, when the application is deployed, the deployer is responsible for mapping users and groups to the security roles.

- CSIv2 security

  CSIv2 is a three-tiered security protocol based on Internet Inter-ORB Protocol (IIOP) that is developed by the Object Management Group (OMG). This protocol provides message protection, interoperable authentication, and delegation in the following layers:

  – A base transport security layer
  – A supplemental client authentication layer
  – A security attribute layer

  Any calls made among secure Object Request Brokers (ORBs) are started over the CSIv2 security protocol. This protocol sets up the security context and the necessary quality of protection. After the session is established, the call is passed up to the enterprise bean layer.

  Important: Secure Association Service (SAS) is supported only between WebSphere Application Server V6 and previous version servers that are federated in a V6.1 cell.

- WebSphere security

  WebSphere security enforces security policies and services that govern access to its resources. It covers a wide range of features. It begins with the administrator user management in the administrative console, controlling which administrative user is allowed to do what on the administrative console. Administrative security is enabled by default.

  WebSphere security provides security services for applications that are running in WebSphere Application Server. WebSphere Application Server supports the J2EE security standards, and provides a means for applications to focus on business logic. WebSphere security handles the authentication, authorization, secure communications, and security auditing needs of the applications.

If all these security layers are passed, the user is allowed to access a WebSphere Application Server resource.

15.3 Authentication

Authentication is the process of confirming a user or system identity. The authentication mechanism in WebSphere Application Server uses a user registry to run this validation. A successful authentication results in the creation of a *credential*, which is the internal representation of a successfully authenticated client user. The abilities of the credential are determined by the configured authorization mechanism.

The WebSphere Application Server supports the following types of web login authentication mechanisms:

- Basic authentication
- Certificate-based authentication
- Form-based authentication

WebSphere Application Server V8.5 supports several authentication mechanisms, but not all of them can be directly selected in the administrative console:

- LTPA
- Kerberos
- Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO)
- Rivest-Shamir-Adleman algorithm (RSA) token authentication
- Web services security SAML Token Profile
- SAML Web SSO post binding profile

Figure 15-3 shows the authentication mechanism selection list.

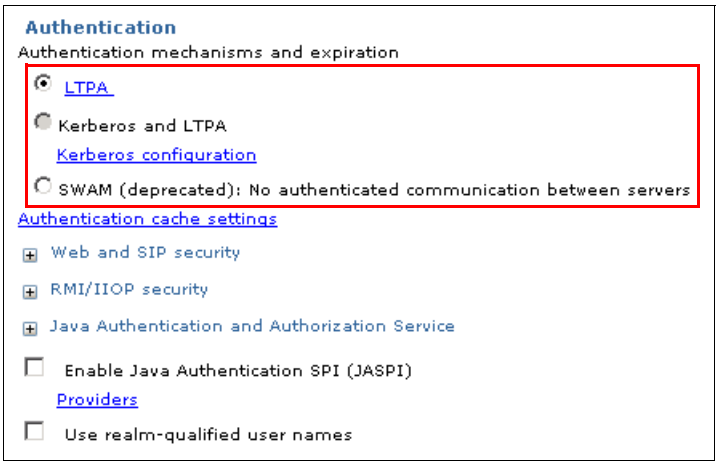


Figure 15-3 WebSphere Application Server V8.5.5 selectable authentication mechanisms

Restriction: This panel is shown only on a single server edition. WebSphere Application Server Network Deployment edition does not offer the Simple WebSphere Authentication Mechanism (SWAM) as a selectable authentication mechanism.

15.3.1 Lightweight Third-Party Authentication

LTPA is intended for single and multiple application server and system environments as the default user authentication protocol. It supports credentials that can be forwarded and SSO. LTPA can support security in a composite environment through cryptography. The LTPA token contains authentication-related data that is encrypted, digitally signed, and securely transmitted. Later, at the receiving side, the information is decrypted, and the signature is verified.

When using LTPA, a token is created with the user information and an expiration time. This token is then signed by the keys. The LTPA token is time sensitive. All product servers that participate in a protection domain must have their time, date, and time zone synchronized. If they are not synchronized, LTPA tokens might prematurely expire and cause authentication or validation failures. When SSO is enabled, this token is passed to other servers through cookies for web resources.

If the server and the client share keys, the token can be decrypted to obtain the user information. The data is then validated by WebSphere Application Server to ensure that data is not expired and that the user information in the token is valid. On successful validation, the resources in the receiving servers are accessible after the authorization check. All WebSphere Application Server processes in a cell (deployment manager, node agents, or application servers) share a set of keys.

If key sharing is required between different cells, export them from one cell and import them to the other. For security purposes, a password is necessary to access the keys.

Tip: When security is enabled during profile creation time, LTPA is configured by default.

15.3.2 Kerberos

Kerberos is a standard network authentication protocol. It is used to provide proof of identity between a client and server, or between servers. Kerberos takes advantage of cryptography as a way to secure identity during an identity exchange. It offers SSO interoperability with other applications that support Kerberos authentication. Kerberos technology allows a user to log in one time and then access other applications that support Kerberos authentication without logging in again.

Kerberos is composed of the following main parts:

- The client that needs access to a service
- The key distribution center (KDC) that is the actual authentication center
- A service server that provides the service for the client

The KDC consists of an authentication server and a ticket-granting server. The authentication server that connects to a user repository, typically a directory server, checks the user identity. The ticket-granting server generates service tickets so that the client can use a service.

The Kerberos *realm* or administration domain includes users, servers, services, and network resources that are registered within the KDC database. Alternatively, Kerberos authenticates *principals*, which can be a user or a server. The granting tickets are assigned to principals.

The *ticket* is the key term in Kerberos. Tickets are encrypted data structures that use shared keys. Tickets are issued by the KDC server. The first ticket, the *ticket-granting ticket*, is created when the user is authenticated with the authentication server. The authentication server returns the ticket-granting ticket to the principal, whose ticket in turn is used to request a service ticket from the *ticket-granting server*. The ticket-granting server generates a new ticket, a *service ticket*, which grants access to the service. The ticket-granting ticket is a long-term ticket that can be reused by the client to request several services. That way, the user is not forced to provide their user credentials each time the user wants to access a service.

In WebSphere, a Kerberos authentication token called *KRBAuthnToken* is created when the client authenticates. The KRBAuthnToken includes the Kerberos principal and the realm name that the client is using to authenticate. If the user sends a delegate authentication request, the KRBAuthnToken contains the delegate principal credentials.

WebSphere Application Server V8.5.5 supports both LTPA and Kerberos. They can be used simultaneously. Applications that use LTPA and applications that use Kerberos or SPNEGO can run together in WebSphere.

Keep in mind the following considerations when using Kerberos:

- It is a real advantage that the actual password never leaves the user system. The user authenticates and obtains a Kerberos ticket-granting ticket from a KDC by using a one-way hash value of the user password. During the subsequent communications, this ticket-granting ticket is used instead of the user password hash.
- The ticket generating algorithm is highly dependent on the synchronized clocks of the domain members. The tickets have a time availability period. If the host clock is not synchronized with the Kerberos server clock, the authentication might time out and fail.

The default configuration requires that clock times are no more than 5 minutes apart. Consider using Network Time Protocol daemons to keep the host clocks synchronized.

- A Java client can participate in Kerberos SSO by using the Kerberos credential cache to authenticate to WebSphere Application Server.
- Because the secret keys for all users are stored on the central server, a compromise of that server compromises all secret keys. The KDC system must be secured and protected.
- The KDC server must be a clustered server. Otherwise, if it is down, no one can log on to any of the managed systems.

15.3.3 Rivest-Shamir-Adleman algorithm token authentication

The RSA token authentication mechanism is used to simplify the security environment for the flexible management topology. It allows you to securely and easily register new servers by using the flexible management feature. The RSA authentication mechanism is used only for server-to-server administrative authentication, such as administration connector and file transfer requests. The RSA authentication mechanism does not replace LTPA or Kerberos for use by applications.

Restriction: You can use the RSA token authentication mechanism only for administrative requests. The authentication mechanism choices for administrative authentication are part of the global security panel of the administrative console. They are under the **Security global security administrative authentication** option.

After the RSA root signer certificate (15-year lifetime) is exchanged between two administrative processes, security information among disparate profiles for administrative requests does not need to be synchronized. The RSA personal certificate (1-year lifetime) is used to run the cryptographic operations on the RSA tokens. It can be verified by the long-lived RSA root. RSA token authentication is different from LTPA, where keys are shared and if one side changes, all sides need to change. Because RSA token authentication is based on a public key infrastructure (PKI), it benefits from the scalability and manageability of this technology in a large topology.

An RSA token has more advanced security features than LTPA. It includes a nonce value that makes it a one-time use token, a short expiration period (because it is a one-time use token), and trust. A trust is established based on certificates in the target RSA truststore. RSA token authentication does not use the same certificates that are used by Secure Sockets Layer (SSL). Thus, RSA has its own keystores. To isolate the trust established for RSA, the truststore, keystore, and root keystore need to be different from the SSL configuration.

15.3.4 Single sign-on

With SSO support, web users can authenticate one time when accessing both WebSphere Application Server and Lotus Domino resources. WebSphere Application Server resources include HTML, JavaServer Pages (JSP) files, servlets, and enterprise beans. Lotus Domino resources include documents in a Domino database and accessing resources in multiple WebSphere Application Server domains.

LTPA provides the SSO feature where a user is required to authenticate only once in a Domain Name System (DNS) domain. The user can then access resources in other WebSphere Application Server cells without prompting. Web users can authenticate one time to a WebSphere Application Server or to a Domino server. This authentication is

accomplished by configuring WebSphere Application Server instances and the Domino servers to share authentication information.

You can enable SSO by configuring it in the global security panel. To enable SSO between WebSphere Application Server and Domino servers, configure SSO for both types of servers.

The following list describes requirements for enabling SSO by using LTPA. Other authentication mechanisms might have different requirements.

- All SSO participating servers must use the same user registry (for example, the LDAP server).
- All SSO participating servers must be in the same domain name system. Cookies are issued with a domain name, and do not work in a domain other than the one for which it was issued.
- All URL requests must use domain names. No IP addresses or host names are allowed because they cause the cookie to work improperly.
- The web browser must be configured to accept cookies.
- Server time and time zone must be correct. The SSO token expiration time is absolute.
- All servers that participate in the SSO scenario must be configured to share LTPA keys.

SSO for HTTP requests is also possible with SPNEGO web authentication. For more information about SPNEGO, see 15.3.5, “Simple and Protected GSSAPI Negotiation Mechanism” on page 479. Microsoft Windows users can access WebSphere Application Server resources without requiring an additional authentication process after being authenticated by a domain controller.

For more information about SPNEGO web authentication, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=csec_SPNEGO_explain

15.3.5 Simple and Protected GSSAPI Negotiation Mechanism

SPNEGO is used when a client application wants to authenticate to a remote server, but neither can detect which authentication protocol the other supports. WebSphere offers SPNEGO support, which negotiates SSO between Microsoft and WebSphere web-based applications. Many customers use SPNEGO as an SSO solution between the Microsoft Windows desktop and WebSphere.

15.3.6 Java Authentication and Authorization Service

The Java Authentication and Authorization Service (JAAS) extends the Java security architecture with additional support to authenticate and enforce access control with principals and users. It implements a Java version of the standard Pluggable Authentication Module (PAM) framework. It extends the access control architecture of the Java platform in a compatible fashion to support user-based authorization or principal-based authorization. WebSphere Application Server fully supports the JAAS architecture. It also extends the access control architecture to support role-based authorization for Java EE resources, including servlets, JSP files, and EJB components.

JAAS is typically used by external applications that want to connect to the WebSphere Application Server and extend its functionality.

Tip: The JAAS has been a part of standard Java since Version 1.4.

Although the applications remain unaware of the underlying authentication technologies, they need to contain specific code to take advantage of JAAS. If a new JAAS module is plugged-in, the application works without a single modification of its code.

A typical JAAS-secured application has the following parts:

- The main application that handles the login procedure and runs the secured code under the authenticated subject
- The action that is started from the main application under a specific subject

When using JAAS to authenticate a user, a *subject* is created to represent the authenticated user. A subject consists of a set of principals, where each principal represents an identity for that user. You can grant permissions in the policy to specific principals. After the user is authenticated, the application can associate the subject with the current access control context. For each subsequent security-checked operation, the Java run time automatically determines whether the policy grants the required permission to only a specific principal. The operation is supported if the subject associated with the access control context contains only the designated principal.

Java Authentication Service Provider Interface

WebSphere Application Server V8.5 supports JSR 196: Java Authentication for Service Provider Interface (JASPI) for containers, sometimes referred to as *JASPIC*. With JASPI, third-party security providers can handle the Java Platform Enterprise Edition (Java EE), authentication of HTTP request and response messages. The JASPI specification extends the pluggable authentication concepts of JAAS.

15.3.7 Trust associations

Web clients can also authenticate by using a trust association interceptor (TAI). A trust association enables the integration of WebSphere Application Server security and third-party security servers. More specifically, a reverse proxy server can act as a front-end authentication server. The product then applies its own authorization policy to the resulting credentials passed by the reverse proxy server.

Important: SPNEGO TAI is deprecated in WebSphere Application Server V8.5 and been replaced by integrated SPNEGO and Kerberos support.

Demand for such an integrated configuration has become more compelling. This is especially true when a single product cannot meet all of the client needs or when migration is not a viable solution. In this configuration, WebSphere Application Server is used as a back-end server to take advantage of its fine-grained access control. The reverse proxy server passes the HTTP request to the WebSphere Application Server that includes the credentials of the authenticated user. WebSphere Application Server then uses these credentials to authorize the request.

15.3.8 Web Services Security SAML Token Profile

The Web Services Security SAML Token Profile OASIS standard specifies how to use SAML assertions with the Web Services Security SOAP Message Security specification. The standard describes the use of SAML assertions as security tokens. It is described in the

<wsse:Security> header, as defined by the Web Services Security, SOAP Message Security specification. The SAML Token Profile has been fully supported since WebSphere Application Server V7.0.0.9.

An XML signature can be used to bind the subjects and statements in the SAML assertion to the SOAP message. Subject confirmation methods define the mechanism by which an entity provides evidence (proof) of the relationship between the subject and the claims of the SAML assertions. The Web Services Security, SAML Token Profile, describes the use of the following subject confirmation methods:

- Bearer

  Because no key material is associated with a *bearer* token, protect the SOAP message by using a transport-level mechanism. Also, message-level protection can be run by other security tokens, such as an X.509 or Kerberos token.

- Holder-of-key

  When using the *holder-of-key* subject confirmation method, proof of the relationship between the subject and claims needs to be established. This proof is established by signing part of the SOAP message with the key specified in the SAML assertion. Because key material is associated with a holder-of-key token, this token can be used to provide message-level protection (signing and encryption) of the SOAP message.

- Sender-vouches

  The *sender-vouches* confirmation method is used when a server needs to propagate the client identity with SOAP messages on behalf of the client. This method is similar to identity assertion. However, this method has the added flexibility of using SAML assertions to propagate the client identity and client attributes. The attesting entity must protect the vouched for SAML assertions and SOAP message content. This process allows the receiver to verify that it has not been altered by another party.

  Two usage scenarios of the sender-vouches confirmation method are supported to ensure message protection either at the transport level or the message level. A receiver verifies that one of the following scenarios occurs:

  – A sender sets up an SSL session with a receiver using client certificate authentication.

  – A sender digitally signs SAML assertions with the containing SOAP message by using a security token reference transformation algorithm. A sender can use either SSL or SOAP message encryption to protect confidentiality.

  In either case, the SAML assertions are either issued by an external Security Token Service, or are self-issued by the application server.

15.4 User registries

The information about users and groups is in a user registry. In WebSphere Application Server, a user registry is used to authenticate a user. It contains information about users and groups so that security-related functions, including authentication and authorization, can be run.

Although WebSphere Application Server supports different types of user registries, only one can be active in a certain scope. WebSphere Application Server supports the following types of user registries:

- Local operating system
- Stand-alone Lightweight Directory Access Protocol

- Custom registry
- Federated repository

15.4.1 Local operating system

With the registry implementation for the local operating system, the WebSphere Application Server authentication mechanism can use the user accounts database of the local operating system. WebSphere Application Server provides implementations for the Windows local accounts registry and domain registry. It also provides implementations for the Linux, Solaris, and AIX user accounts registries.

Restriction: A local operating system registry can be used only in single server installations. WebSphere cell configuration does not support the use of operating system registry.

When the system that hosts the WebSphere Application Server process is a member of a Windows operating system domain, both the local and the domain user registries are used by default. The domain user registry takes precedence over the local user registry. By using the `com.ibm.websphere.registry.UseRegistry` property, you can set the registry to `local` or `domain` registry only.

On UNIX platforms, the process ID that runs the WebSphere Application Server process needs root authority to call the local operating system APIs for authentication. It also needs root authority to obtain user or group information. These platforms include AIX, Linux, Solaris, and HP-UX.

15.4.2 Stand-alone Lightweight Directory Access Protocol

The stand-alone LDAP user registry setting supports authentication of users from a single LDAP tree. This authentication can be a single LDAP server or a single server with one or more stand-by failover servers. To provide high availability, all the LDAP server instances must have the same LDAP content. WebSphere Application Server tries to connect to the first server on the configuration list. If the current active LDAP server is unavailable, WebSphere Application Server security attempts to fail over to the next available LDAP host in the specified host list.

When you first create a profile, WebSphere Application Server is configured to use a federated repositories security registry option with the file-based registry. You can change this security registry configuration to use other options, including the stand-alone LDAP registry.

Consider using the federated repositories option, which provides the following benefits for LDAP configuration:

- The ability to have one or multiple user registries
- Federating one or more LDAPs, in addition to the file-based and custom registries
- Improved failover capabilities
- A robust set of member (user and group) management capabilities

You must use the federated repositories option when using the new member management capabilities in these applications:

- WebSphere Portal V6.1 and later
- WebSphere Process Server V6.1 and later

You must use the federated repositories option for LDAP referrals, which is a common requirement in some LDAP server environments, such as Microsoft Active Directory.

The stand-along LDAP registry option is functionally stabilized. IBM has no plans to further enhance this option. Generally, migrate from the stand-along LDAP registry option to the federated repositories option. If you plan to move to WebSphere Portal V6.1 and later or WebSphere Process Server V6.1 and later, migrate to the federated repositories option before these upgrades.

For more information, see these resources:

- Federated repositories

  http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=cwim_fedrepos

- Migrating a stand-alone LDAP repository to a federated repositories LDAP repository configuration

  http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=twim_migrate_standaloneldap

Tip: You can set up highly available and performance balanced LDAP servers by using a load balancer.

For a list of the supported LDAP servers, see System Requirements for WebSphere Application Server V8.5 Base and Network Deployment at:

http://www.ibm.com/support/docview.wss?uid=swg27021246#AIX_LDAP_Servers_using_Stand_Alone_LDAP_Registry_Configuration_ww

For LDAP servers that are not listed but that are supported by the LDAP V3 specification, configure the LDAP server using a custom LDAP feature with an appropriate filter. You must obtain the appropriate filter information from the LDAP vendor.

WebSphere Application Server supports the use of nested groups and dynamic groups in single LDAP and in federated repositories:

- *Nested groups* enable the creation of hierarchical relationships that are used to define inherited group membership. A nested group is defined as a child group entry whose distinguished name (DN) is referenced by a parent group entry attribute.
- *Dynamic groups* contain a group name and membership criteria. The LDAP server looks for the possible group members who satisfy the criteria:
  - Group membership information is as current as the information about the user object.
  - Members do not need to be manually maintained on the group object.

Dynamic groups are for applications that do not need a large amount of information from the directory to determine whether someone is a group member.

15.4.3 Custom registry

With the custom user registry, you can connect to any type of user repository. For the custom registry, you can implement the Security Policy Index (SPI). The SPI is the UserRegistry interface, which is the same interface used by the local operating system (OS) and LDAP registry implementations. Through this interface, the application server calls the repository handler class that you provide, which connects to the actual repository. The advantage and flexibility of this option is that you can implement the SPI, which handles the repository. That

way, WebSphere Application Server can connect to any needed repositories, such as flat, stanza, XML file, and database.

The UserRegistry interface is a collection of methods that are required to authenticate individual users by using a password or certificates. The interface also collects information about the user authorization purposes. This interface includes methods that obtain user and group information so that they can be given access to resources. When implementing the methods in the interface, decide how to map the information manipulated by the UserRegistry interface to the information in your registry.

15.4.4 Federated repository

Federated repositories provide a unified view of the user information that is owned by multiple user repositories. Federated repositories support the following types of repositories:

- File-based repository

  A file-based repository is the built-in WebSphere repository, which is used by default if you enable administrative security when you create the repository. The administrative users are then created in the WebSphere configuration repository XML structure.

  Although the passwords are encrypted in a file-based registry, the operating systems are responsible for avoiding unauthorized access to the file.

- LDAP (full or subtree) repository

  For information about the LDAP repository, see 15.4.2, “Stand-alone Lightweight Directory Access Protocol” on page 482.

- Database repository

  Database user repositories have been supported since WebSphere Application Server V7. The application server connects to a JDBC resource. This resource points to a database and a table, which must include the standard VMM entity types PersonAccount, Group, and OrgContainer.

  Restriction: The database repository is configurable only by using the **`wsadmin`** command-line interface (CLI).

- Custom registry

  For information about custom registry, see 15.4.3, “Custom registry” on page 483.

15.5 User roles in WebSphere

WebSphere Application Server differentiates the following user roles:

- *Operating system* users are the technical users who are created for the operating system. Operating system user accounts are stored and managed by the operating system itself. This type of user can log on to the host operating system. If access is granted by the system administrator, the user can issue WebSphere command-line commands, such as **`startServer.bat`**, **`stopServer.sh`**, and **`versionInfo.bat`**. This user can be a root or administrator, or a non-root or non-administrator user.
- *Administrative users* are those users who can manage the application server. Only administrative users can log on to the administrative console. However, they might not necessarily be an operating system user. Different roles are inside the administrative console, such as the administrator, operator, or auditor.

Administrative users must authenticate to issue commands by using the following command:

```
stopServer server1 -user wasadmin -password admin
```

For more information about the administrative console user role, see “Fine-grained administrative security” on page 487.

- *Application users* have no access to the operating system or the administrative console. They can log on only to the application, typically by using a web browser. These users need authorization to access the different parts of the application, as explained in “Security roles” on page 489.

The user accounts for the administrative and application users are stored in a user registry, such as in an LDAP server.

15.6 Authorization

Authorization is the process of checking whether a user has the privileges necessary to access a requested resource. WebSphere Application Server differentiates the following types of authorization based on user roles:

- Administrative security roles
- Application security roles

15.6.1 Administrative security roles

Administrative security in WebSphere Application Server controls access to the configuration and management interfaces. Administrative security covers a wide range of the security features:

- Administrative console security
- Authentication mechanism
- Authentication of HTTP clients
- Authentication of IIOP clients
- Common user registry
- Naming security
- Propagation of identities (RunAs)
- Role-based authorization checks of servlets, enterprise beans, and Managed Beans (MBeans)
- Use of SSL transports

The following security information also defines the behavior of a security domain:

- The authentication protocol (Remote Method Invocation over the Internet Inter-ORB Protocol (RMI/IIOP) security)
- Other miscellaneous attributes

Administrative user roles

For a user or a group to have administrative authority, the user or group must be assigned to one of the following roles (Figure 15-4 on page 487):

- Monitor

  The monitor role has the fewest permissions and restricts the user to viewing the configuration and current state.

- Configurator

  The configurator role has the same permissions as the monitor role, and can change the configuration. For example, the configurator can deploy an application.

- Operator

  The operator role has monitor permissions and can change the runtime state. For example, the operator can start or stop services.

- Administrator

  The administrator role has the combined permissions of the operator and the configurator. This role has permission to access sensitive data, including server password, and LTPA password and keys.

  The administrator role is the superuser of the WebSphere Application Server. A user in this role can perform all tasks, except (if revoked) those tasks that are associated with the auditor role.

- ISC admins

  An individual or group that uses the ISC admins role has administrator privileges for managing users and groups in the federated repositories. These privileges can be accessed only from within the administrative console.

  Important: The ISC admins role is available only for administrative console users. It is not available for **`wsadmin`** users.

- Deployer

  The deployer role can perform both configuration actions and runtime operations on applications.

- Admin security manager

  The admin security manager role separates administrative security administration from other application administration. By default, the server ID and admin ID, if specified, are assigned to this role in the cell level authorization table. This role implies a monitor role. However, an administrator role does not imply the admin security manager role.

  Only users who are assigned to this role can assign users to administrator roles. When fine-grained administrative security is used, only users who are assigned to this role at the cell level can manage authorization groups.

- Auditor

  The auditor role can view and modify the configuration settings for the security auditing subsystem. The auditor role includes the monitor role, allowing the auditor to view but not change the rest of the security configuration. For more information about the auditor role, see 15.9, “Auditing” on page 494.

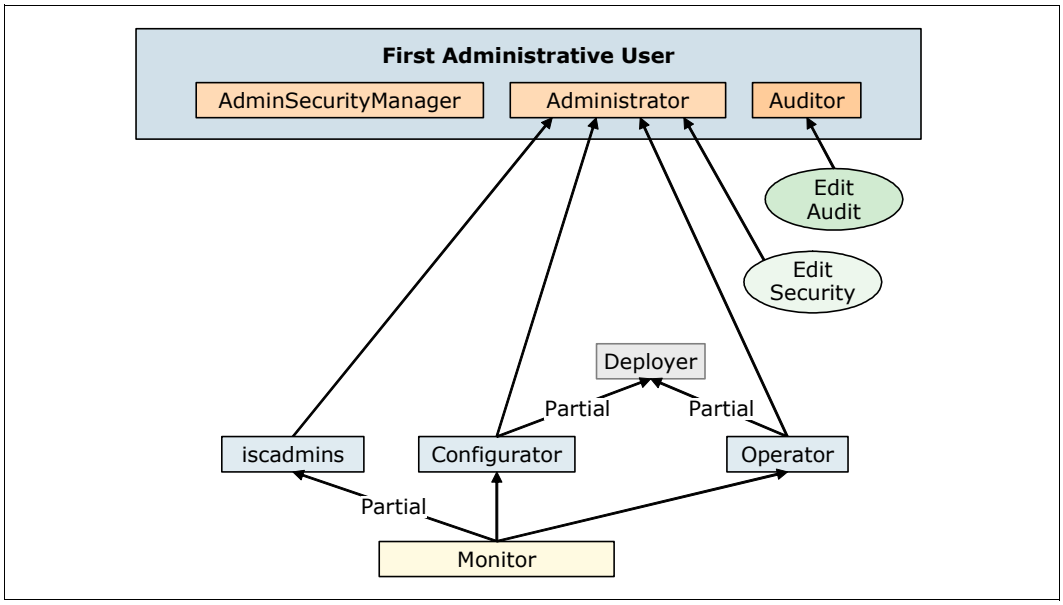


Figure 15-4 Administrative user roles

Fine-grained administrative security

The traditional all-or-none security model was extended in WebSphere Application Server V6.1 with optional fine-grained administrative security. In WebSphere Application Server V6.1, security was configurable only by using the **`wsadmin`** CLI. Since WebSphere Application Server V7, configuration from the administrative console is possible.

Fine-grained administrative security can grant access for each user role to each resource instance instead of granting access to all of the resources in the cell. With fine-grained administrative security, you can take advantage of better separation of administrative duties.

If no scope is mapped to the security roles, the scope is assigned automatically to the widest scope. For example, in a cell, the widest scope is the cell scope. In this case, the traditional authorization model is working.

Security domains

WebSphere security domains provide the flexibility to use different security configurations in a WebSphere Application Server cell. WebSphere security domains are also referred to as *multiple security domains* or simply *security domains*. With security domains, you can configure different security attributes, such as the user registry, for different applications in the same cell.

The global security configuration applies to all administrative functions, naming resources, and MBeans. This configuration is the default security configuration for user applications. A global security configuration must be defined before the security domains can be created. If no security domains are configured, all of the applications use the global security configuration.

When a security domain is created and associated with a scope, only user applications in that scope use the security attributes defined in that domain. The administrative applications and the naming operations in that scope use the global security configuration. Each security domain must be associated with a scope (a cell or specific clusters, servers, and service integration buses) where it will be applied.

You can configure the following attributes at the domain level:

- Application security
- Audit
- Authentication mechanism attributes
- Authorization provider
- Custom properties
- JAAS logins (application, system, and J2C authentication data)
- Java 2 security
- Java Authentication SPI
- Federated repositories
- RMI/IIOP security (CSIv2)
- SPNEGO web authentication
- Trust association
- User realm (registry)
- z/OS properties

You do not need to configure all the attributes. Those attributes that are not defined in the domain are obtained from the global configuration. When planning for security, you must determine whether you need different security attributes for your servers or if they can use the global configuration settings. For example, you might want to use various user registries if you have different sets of users that cannot be mixed. This can occur when the responsibility for user administration of each registry falls on different teams.

15.6.2 Application security roles

The Java EE specification defines the building blocks and elements of a Java EE application. The specification provides details about security that are related to different elements. A typical Java EE application consists of an application client tier, a web tier, an EJB tier, and a web services tier. When designing a security solution, you must be aware of the connections between each of the modules.

Figure 15-5 shows the components of a Java EE application.

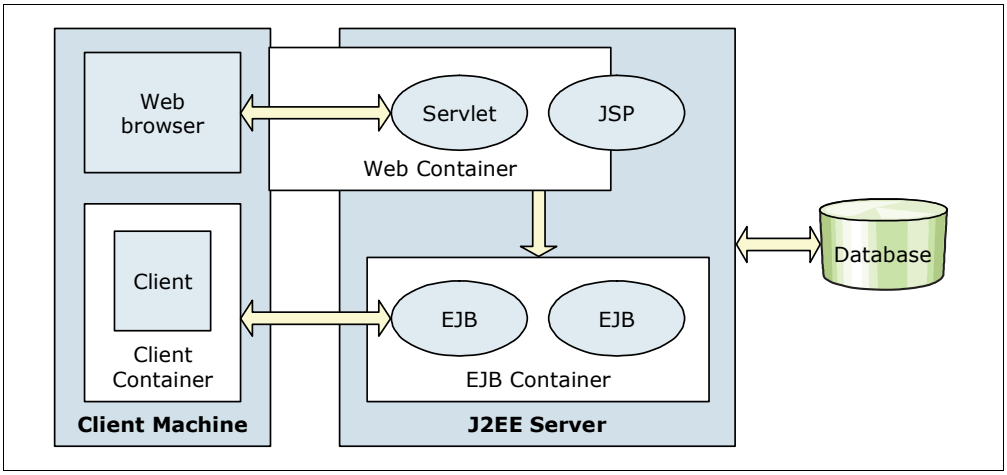


Figure 15-5 Java EE application components

For example, a user who is using a web browser can access a JSP or a servlet, which is a protected resource. In this case, the web container needs to check whether the user is authenticated and has the required authorization to view the JSP or servlet. Similarly, a thick

client can also access an EJB. When you plan for security, consider the security for every module.

Security roles

A *security role* is a logical grouping of users that is defined by the application assembler. It is not possible at development time to know all the users who are going to use the application. Security roles provide developers a mechanism through which to define the security policies for an application. Developers can then create named sets of users (for example managers, customers, and employees) who have specific levels of access to secure resources and methods. At application assembly time, these users are place holders. At deployment time, they are mapped to real users or groups.

Figure 15-6 shows an example of how roles can be mapped to users.

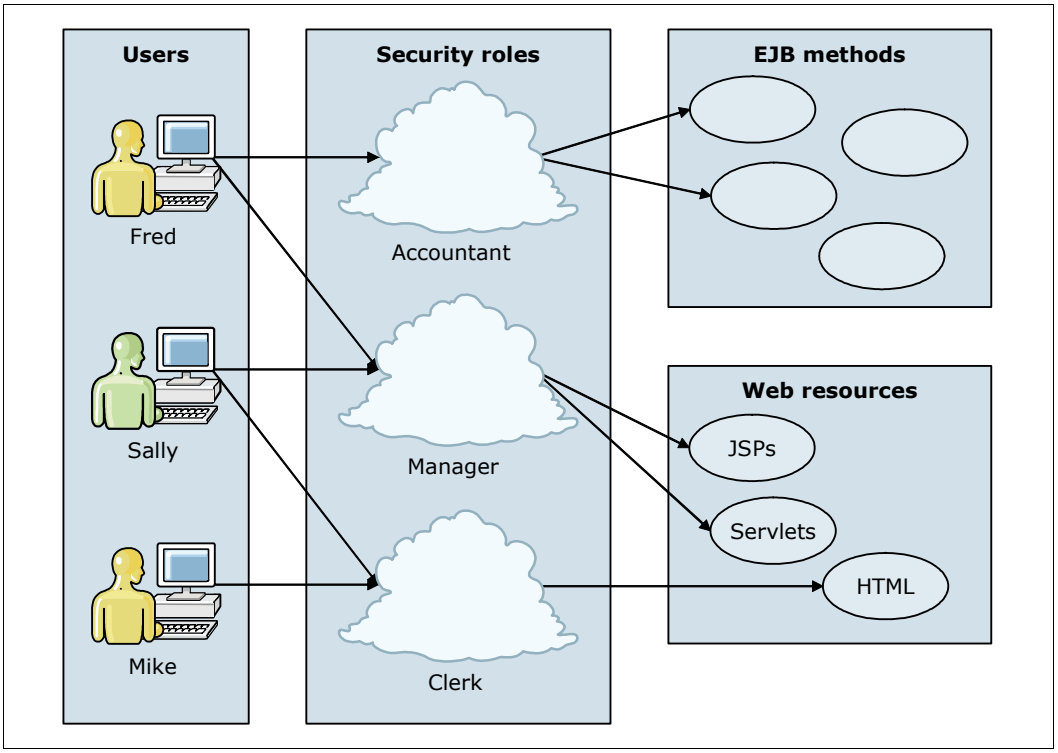


Figure 15-6 User role mapping

This two-phase approach to security gives a great deal of flexibility. Deployers and administrators have control over how their users are mapped to the various security roles.

Security for Java EE resources

Java EE containers enforce the following types of security:

- Declarative security
- Programmatic security

Declarative security

Declarative security is the means by which the security policies of an application can be expressed externally to the application code. At application assembly time, security policies are defined in an application *deployment descriptor*. A deployment descriptor is an XML file that includes a representation of the security requirements of an application. The

requirements include the security roles, access control, and authentication requirements of the application.

When using declarative security, application developers can write component methods that are unaware of security. By changing the deployment descriptor, the security environment of an application can be radically changed without requiring any changes in application code. The deployment descriptor can be created and modified by using Rational Application Developer for WebSphere Software V8.

Security policies can also be defined by using security annotations. Security annotations are included in Java code in a declarative manner. For more information, see “Security annotations” on page 490.

Programmatic security

Programmatic security is useful when the application server-provided security infrastructure cannot supply all the functions that are needed for the application. Using the Java APIs for security can be the way to implement security for the whole application without using the application server security functions. Programmatic security also provides the option to implement dynamic security rules for your applications.

Generally, the developer does not have to code for security because WebSphere Application Server provides a robust security infrastructure that is not apparent to the developer. However, sometimes the security model is not sufficient and the developer wants greater control over what the user has access to. For such cases, the developer can implement a few security APIs. For more information, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=tsecdesign

Java security

Java EE security guards access to web resources (such as servlets, JSP, and EJB) and to system resources (such as file I/O, sockets, and properties).

Important: Java security places requirements on application developers and administrators. Your applications might not be prepared for the fine-grain access control programming model that Java security can enforce. For more information, see the WebSphere Application Server V8.5 Information Center:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-dist&topic=csecrsecmgr2

Security annotations

Java annotations are powerful programming tools that resulted from the JSR-175 recommendation. They are a standard way to include supported security behaviors and continue to have the source code and configuration files generated automatically. In Java EE 6, the security roles and policies can be defined by using annotations and within the deployment descriptor. During the installation of the application, the security policies and roles defined by using annotations are merged with the security policies and roles defined within the deployment descriptor. This merge is run by the Annotations Metadata Manager (AMM) facility. Data defined in the deployment descriptor takes precedence over data defined in annotations.

Java annotations can be used in EJB 3.0 and 3.1, and Servlet 3.0 components and later. However, some security annotations are available only with EJB 3.0 components.

Java Authorization Contract for Containers

WebSphere Application Server V8.5 supports both a default authorization provider and an authorization provider that is based on the Java Authorization Contract for Containers (Java ACC) specification. WebSphere Application Server supports a Java ACC provider so that authorization can be administered externally by using a customer developed Java ACC implementation. With the Java ACC-based authorization provider, third-party security providers can handle the Java EE authorization.

When security is enabled, the default authorization is used unless a Java ACC provider is specified. The default authorization does not require special setup, and the default authorization engine makes all of the authorization decisions.

When a Java ACC provider is used for authorization, the Java EE application-based authorization decisions are delegated to the provider according to the Java ACC specification. Figure 15-7 shows the communications flow.

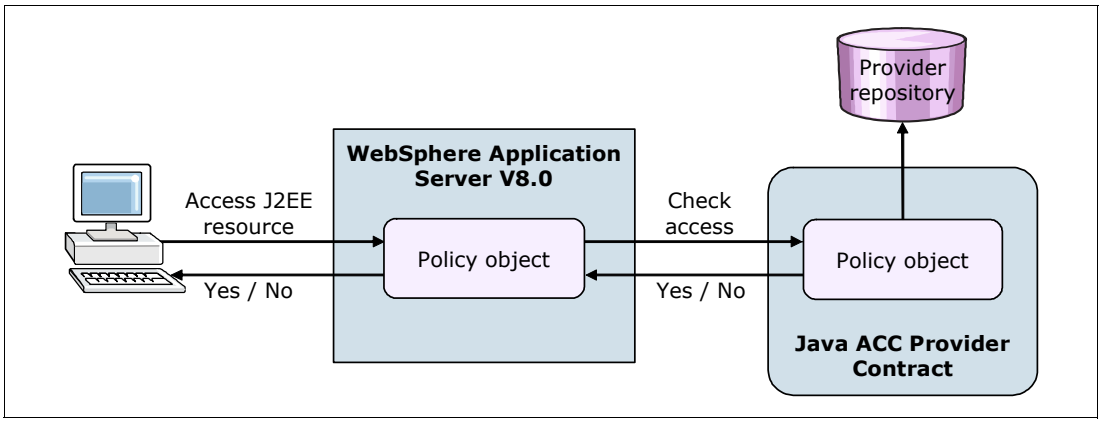


Figure 15-7 Java ACC provider architecture

Remember: All administrative security authorization decisions are made by the default authorization engine of WebSphere Application Server. The Java ACC provider is not called to make the authorization decisions for administrative security.

WebSphere Application Server handles the dynamic module update with respect to Java ACC for web modules. When the web module is updated, you must restart only that particular application in native authorization mode. If Java ACC is being enabled, it depends on the provider support to handle the dynamic module updates specific to the security modules.

15.7 Internal and external trusted relationships

WebSphere Application Server needs to communicate with different components inside the cell and to connect to external services. This section provides an overview of the security for these connections.

15.7.1 Secure communications

To prevent eavesdropping on communications, you must add security. WebSphere Application Server uses Java Secure Socket Extension (JSSE) as the SSL implementation for secure connections. JSSE is part of the Java Platform, Standard Edition (Java SE),

specification and is included in the IBM implementation of the Java runtime environment (JRE). JSSE handles the handshake negotiation and protection capabilities that are provided by SSL to ensure that secure connectivity exists across most protocols. JSSE relies on an X.509 standard public key infrastructure (PKI).

A PKI represents a system of digital certificates, certificate authorities, registration authorities, a certificate management service, and a certification path validation algorithm. A PKI verifies the identity and the authority of each party that is involved in an Internet transaction. This verification is done either financial or operational, with requirements for identity verification. It also supports the use of *certificate revocation lists* (CRLs), which are lists of revoked certificates.

Secure Sockets Layer

SSL is the industry standard for data interchange encryption between clients and servers. SSL provides secure connections through the following technologies:

- Communication privacy

  The data that passes through the connection is encrypted.

- Communication integrity

  The protocol includes a built-in integrity check.

- Authentication

  The server authenticates the client by exchanging digital certificates.

A *certificate* is an electronic document that includes the following information:

- Name of the certificate holder
- Public key for encryption or decryption
- Verification of the public key of a sender
- Name of the certificate authority
- Validity period for the certificate

The certificates in WebSphere Application Server are stored in password protected files, called *keystores*, except for z/OS SAF key rings. A certificate authority (CA) is an organization that issues certificates after verifying the identity of the requester.

Certificate management

You can create and manage certificates through the administrative console. WebSphere Application Server provides mechanisms to create and manage CA clients and keystores. It also includes mechanisms to create self-signed certificates and CA requests. Keystores in WebSphere Application Server profiles hold personal certificates, and the truststore holds signer certificates from other servers with which it is communicating.

A personal certificate stores the private and public key of the node with some identity information. A signer certificate contains a public key that is associated with same personal certificate.

15.7.2 SSL in cell management

The WebSphere Application Server uses SSL to communicate among nodes within the cell. It maintains certificates for each node in the cell.

When a new profile is created, including the deployment manager profile, a new unique chained certificate is also generated for the profile. This chained certificate consists of a signer certificate, which has a 15-year expiration, and personal server certificates, which have

a one-year expiration by default. WebSphere Application Server has its own, built-in mini CA with which it signs the certificates in the cell.

Alternatively, you can use your own certificate settings, in which case the following settings can be overridden:

- Keystore password
- Expiration time in years
- Distinguished name for both the signer and for the personal certificate
- Both the signer and personal certificates can be imported, replacing the defaults

The cell has its own certificate chain. It also has a cell truststore and a cell keystore. When a node is federated to a cell, the certificate is replaced with one signed by the cell root certificate. This new certificate is put into the cell truststore. Additionally, the default SSL configuration is modified automatically to point to the common truststore so that the node can access all other node signer certificates. With these certificates, the node can communicate with all other servers in the cell.

Certificate expiration

All certificates have an expiration. As mentioned previously, the personal server certificate has a default expiration of one year, and the signer certification has a default expiration of 15 years. This latter is long enough not to expire before you upgrade to the next release of WebSphere Application Server. You can replace the certificate with a newer certificate, when necessary, by using the administrative console.

The personal certificates are valid for one year by default. When it is necessary to replace these certificates, you can replace them manually. However, application server can replace the certificates automatically by using the built-in expiration manager. The expiration manager tracks the certificates. You can configure it to send notifications, and to automatically renew the certificates that are due to expire. If you do not want the expiration manager to renew the certificates automatically, then you must do it manually by using the administrative console. After the certificate renewal, the new certificates are propagated to the nodes automatically.

For more information about expiration manager, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=csec_sslcertmonitoring

Web server plug-in key ring

When the web server forwards HTTPS requests, it needs to communicate directly with the application servers. For example, it might need to communicate from one node to another node inside the cell. The plug-in configuration is a bit different from the nodes. The plug-in does not separate trust and keystore files. Rather, it maintains only one keyring file.

You can generate the plug-in personal keys by using the administrative console. Then you can add the node signer certificates to the keyring files. Finally, the manager can replicate the keystores to the web server directory structure.

15.7.3 External trusted relationships

WebSphere Application Server uses several other communication channels during production. These channels can transmit sensitive information. You need to configure these channels to use SSL communication.

The following external connections need SSL encryption:

- JDBC database connection
- LDAP directory protocol connection
- Messages channels
- Web services communication

15.8 Security trace

WebSphere Application Server V8.5.5 has a built-in tracing infrastructure for security components. If a security-related issue occurs in WebSphere Application Server, you might trace the security infrastructure to find the root cause of the problem.

The following classes implement WebSphere Application Server security:

- `com.ibm.websphere.security.*`
- `com.ibm.WebSphereSecurityImpl.*`
- `com.ibm.ws.security.*`
- `com.ibm.ws.wim.*`

  Explanation: The `com.ibm.ws.wim.*` class is for tracing with the federated repository.

- `SASRas`

The trace facility has different logging levels: Fine, finer, finest, and all levels. The trace data can be sent to the `trace.log` output file in the standard log directory of the process that is investigated. Alternatively, it can be collected in an in-memory buffer to create a dump file.

15.9 Auditing

The security auditing feature was new in WebSphere Application Server V7. With the audit service, WebSphere Application Server can log significant system and application events so you can later review these long-term logs.

Security auditing has the following primary goals:

- Confirming the effectiveness and integrity of the existing security configuration (accountability and compliance with policies and laws), most commonly by reviewing who did what operation
- Identifying areas where improvement to the security configuration might be needed (vulnerability analysis)

During run time, all code (except the Java EE application code) is considered to be trusted. Each time a Java EE application accesses a secured resource, any internal application server process with an audit point included can be recorded as an auditable event.

WebSphere Application Server auditing works through event logging. All security-related events are filtered with an audit filter and an event outcome filter. The captured events, which go through both filters, are added to the audit log.

The security auditing subsystem can capture the following types of auditable events:

- Audit subsystem-related runtime events such as start and stop
- Authentication

- Authentication termination (timeout, session termination, and logout)
- Authorization
- Delegation
- Principal or credential mapping
- Resource access (access to all file system, database, HTTP, and other resources)
- Security subsystem-related runtime events
- Signing and encryption
- User credentials modification

The different audit outcome filters are as follows:

- Challenge
- Denied
- Error
- Failure
- Info
- Redirect
- Success
- Warning

After the administrator selects the filter types from these two lists, WebSphere Application Server creates a Cartesian product and sets the filter definition.

WebSphere Application Server has a built-in auditor administrative role. Only the administrators in the auditor role can change settings related to the audit subsystem and review the audit logs. By default, the primary administrative user is a member of the auditor administrative role, but this role can be removed from this user. You can create a separate auditor user role and user principal. Assign these roles to a security team member for WebSphere Application Server. With this approach, only appropriate users have access to the audit data, and the audit subsystem and console administrator users cannot tamper with the audit content.

A user in the auditor role is necessary in WebSphere Application Server V8.5 to set up, configure, run, and review the auditing subsystem. Fine-grained security for the auditor role is not implemented. The auditor has full authority to read and modify the configuration information that is associated with the security auditing subsystem. Also, the auditor role includes the monitor role for the administrative console.

You can enable the audit subsystem in the administrative console by clicking **Security** → **Security auditing**, or by using the **`wsadmin`** interface.

The security audit log is added to the audit message log file, or it can send email to one or more addresses. The log message file is a text file, but it is not for human interpretation. The message log file is generated in the server log directory, by default in the `profile_root/logs/server_name` directory, with a name using the following pattern:

`BinaryAudit_`*`cellName_nodeName_serverName`*`.log`

See the following example:

`BinaryAudit_aNode01Cell_aNode01_server1.log`

Consider the performance and storage needs of the audit subsystem. WebSphere Application Server V8.5 adds controls to handle conditions when the audit flat files become full.

The following additional settings are available:

WRAP	The log file is written round-robin with the oldest file being overwritten.
NOWRAP	The server is quiesced.
SILENT_FAIL	Audit logging is stopped, but the server process continues.

You can encrypt the log file to avoid unauthorized read access. You can also sign the log file to block unauthorized write access. Encryption and signing are not enabled by default, but configuring them is a preferred practice. Encryption is managed by the auditor. The certificate that is used to encrypt the data records is managed within the audit subsystem and defined in the `audit.xml` file. Signing is managed by WebSphere Application Server. The certificate that is used to sign the data records is managed with WebSphere Application Server and is described in the `security.xml` file.

Default plug-in implementations are shipped with WebSphere Application Server V8.5 that capture and output the audit records to a binary audit log file. The security audit subsystem is built on the following plug-ins:

- Audit Event Factory, which captures data
- Audit Service Provider, which outputs the captured data to a back-end repository

If you need logging infrastructure, you can implement your own solution by using the plug-in architecture or by installing a third-party solution.

WebSphere Application Server V8.5 for z/OS uses System Authorization Facility (SAF) security to associate a SAF user ID with a distributed identity. When you use this feature, you can maintain the original identity information of a user for audit purposes and have less to configure in WebSphere Application Server. The SAF can send the audit record to System Management Facility (SMF). The SMF records all access violation and generates messages to the z/OS administrative subsystems. For more information about SAF and SMF, see *z/OS MVS System Management Facilities (SMF)*, SA22-7630, at:

`http://publib.boulder.ibm.com/infocenter/zos/v1r11/index.jsp?topic=/com.ibm.zos.r11.ieag200/abstract.htm`

The audit log is not displayed in the administration console. The logs can be read, if they are not encrypted, by using a text editor. However, these logs are not formatted for human interpretation. WebSphere Application Server V8.5.5 has an audit reader utility that reads the audit message log file and generates an HTML report. You can start this utility by using a **`wsadmin`** command.

The following Jython administrative script sample generates a basic audit report as shown in Figure 15-8:

```
AdminTask.binaryLogReader('[-fileName myFileName -reportMode basic -outputLocation
/binaryLogs.html]')
```

Audit Records

Hostname myHost . ReportTime 2011.05.05., 11:04:03

Record Number	Event Type	Outcome
0	**SECURITY_MGMT_AUDIT**	**SUCCESS**
CreationTime=Thu May 05 09:45:27 EDT 2011	Action=enable	ProgName=WASServer
RegistryType=WIMUserRegistry	Domain=null	Realm=defaultWIMFileBasedRealm
RemoteAddr=null	RemotePort=null	RemoteHost=null
RegistryUserName=null	AppUserName=null	NameInApp=null
FirstCaller=null	CallerList=null	TerminateReason=null
ResourceName=AuditSubSystem	ResourceType=process	ResourceUniqueId=0
1	**SECURITY_MGMT_AUDIT**	**INFO**
CreationTime=Thu May 05 09:45:27 EDT 2011	Action=auditNotificationChange	ProgName=WASServer
RegistryType=WIMUserRegistry	Domain=null	Realm=defaultWIMFileBasedRealm
RemoteAddr=null	RemotePort=null	RemoteHost=null
RegistryUserName=null	AppUserName=null	NameInApp=null
FirstCaller=null	CallerList=null	TerminateReason=null
ResourceName=AuditSubSystem	ResourceType=process	ResourceUniqueId=0
2	**SECURITY_MGMT_AUDIT**	**INFO**
CreationTime=Thu May 05 09:45:27 EDT 2011	Action=auditPolicyModify	ProgName=WASServer
RegistryType=WIMUserRegistry	Domain=null	Realm=defaultWIMFileBasedRealm
RemoteAddr=null	RemotePort=null	RemoteHost=null
RegistryUserName=null	AppUserName=null	NameInApp=null
FirstCaller=null	CallerList=null	TerminateReason=null
ResourceName=AuditEventFactoriesConfig	ResourceType=process	ResourceUniqueId=0

Figure 15-8 Audit utility report

15.10 Securing the Liberty profile

Security, as with other features in the Liberty profile, is optional. You can enable it when necessary. The following features are applicable to security in the Liberty profile:

- appSecurity-1.0 enables user registry support, authentication, and authorization
- ssl-1.0 enables SSL encryption for front-end and back-end (database, LDAP) connections
- zosSecurity-1.0 enables support for SAF Registry and Authorization on the z/OS platform
- restConnector-1.0 enables remote access by Java Management Extensions (JMX) client for Representational State Transfer (REST)-based connector

The Liberty profile is more than sufficient for many businesses. However, the WebSphere Application Server full profile includes the following features that are not available in the Liberty profile:

- EJB Security (RMI/IIOP, CSIv2, propagation)
- Federated Repository (VMM)
- Password Encryption
- Key and Certificate Management (except SSL)
- SPNEGO or Kerberos

- Complete SSO support
- z/OS Sync to thread
- Security Audit
- Multiple Security Domain
- Enhanced certificate and key management
- Custom User Registry
- Local OS Registry support (except z/OS)
- Java 2 Security
- Java ACC

15.10.1 SSL configuration

The Liberty profile includes support for the following SSL configuration:

- Creation of separate SSL configurations, one of which can be the default
- Client certification authentication, so that if the client certificate authentication fails, it is possible to fall back to the basic authentication (user ID and password)
- A minimal SSL configuration that requires only the specification of a keystore

  If no truststore is specified, it is assumed that it is the same file as the keystore.
- Trust and key managers configuration from the SDK configuration files, and cannot be overridden from those values

15.10.2 Authentication

The Liberty profile supports the following ways of establishing the user subject before passing the data to the authorization code:

- Based on simple one user security using the `quickStartSecurity` configuration element. This user is also granted automatically **Administrator** role.

  ```
  <quickStartSecurity userName="Bob" userPassword="bobpwd" />
  ```
- The default login module connects to the user registry to validate the password or map the client certificate to the user identity. The following types of user registries are supported in the Liberty profile:
  - Basic registry with user names, groups, and encoded passwords that are specified in the Liberty profile configuration file (`server.xml`)
  - LDAP registry with SSL support and custom filters for users and groups. A sample is provided for IBM Directory Server and Microsoft Active Directory.
  - SAF is available on z/OS with support for both authentication and authorization (when `zosSecurity-1.0` is enabled in addition to `appSecurity-1.0`)
- The use of LTPA tokens from cookies sent by browsers, which allows the creation of a single sign-on solution for a group of Liberty profiles
- Custom trust association interceptor (see 15.3.7, “Trust associations” on page 480)
- Custom JAAS login modules (see 15.3.6, “Java Authentication and Authorization Service” on page 479)

In the Liberty profile, the creation of a basic user registry supports the quick setup of a development environment. This environment is important when authentication and authorization is required by an application.

15.10.3 Authorization

The Liberty profile uses the same concept of mapping users and groups to roles as the full WebSphere Application Server profile. For more information, see 15.6.2, “Application security roles” on page 488. The authorization table (Figure 15-9) can be specified in two places:

- The `ibm-application-bnd.xml` file, which is only supported when the application is packaged as an enterprise archive (EAR)
- The Liberty profile configuration file (`server.xml`), which is always supported

```
<application type="war" id="myapp" name="myapp"
location="${server.config.dir}/apps/myapp.war">
   <application-bnd>
      <security-role name="user">
         <group name="students" />
      </security-role>
      <security-role name="admin">
         <user name="gjones" />
     <group name="administrators" />
      </security-role>
      <security-role name="AllAuthenticated">
         <special-subject type="ALL_AUTHENTICATED_USERS" />
      </security-role>
   </application-bnd>
</application>
```

Figure 15-9 Example mapping table

15.11 Resources

For more information about WebSphere Application Server security, see *WebSphere Application Server V7.0 Security Guide*, SG24-7660, at:

http://www.redbooks.ibm.com/abstracts/sg247660.html?Open

Consider having a copy of this book available for security planning of your environment. However, this book is written for WebSphere Application Server V7. Therefore, it does not cover the features and changes found in WebSphere Application Server V8.5.

For up-to-date information about securing applications and their environment, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=welc6topsecuring

16

WebSphere Application Server for z/OS

This chapter concentrates on the features of WebSphere Application Server for z/OS V8.5. The features and functions described in this chapter are available only with WebSphere Application Server for z/OS V8.5. WebSphere Application Server for z/OS is fully aligned with the platform to bring unique capabilities and enhancements only available on System z.

This chapter includes the following sections:

- WebSphere Application Server structure on z/OS
- Functions in WebSphere Application Server for z/OS V8.5
- Installing WebSphere Application Server for z/OS
- System programmer considerations
- Planning checklist
- Intelligent Management and WebSphere Batch on z/OS
- The Liberty profile on z/OS
- Resources

16.1 WebSphere Application Server structure on z/OS

This section shows the added value that the implementation for Websphere Application Server for z/OS offers compared to the distributed versions.

For those users who might not be familiar with the z/OS operating system, this section includes explanations of z/OS terms or techniques in general IT terminology. It explains how they might add value to your business environment.

16.1.1 Value of WebSphere Application Server for z/OS

WebSphere Application Server for z/OS V8 combines the leading application server from IBM with the z/OS high-end server platform. This combination offers the following unique features that can be of value to your environment and business:

- Service level agreements (SLAs) with workload management and local connections to back-end servers and enterprise systems

  WebSphere Application Server for z/OS uses the Workload Manager (WLM) component to assign resources to the application server automatically. This process helps achieve the performance goals that are set for the environment. You can set these goals on a transaction level. For example, you might want to ensure that platinum customers get the best response time. Balancing is done as part of the entire zEnterprise.

  Local connectors can be used to access databases and enterprise information systems that are running in the same operating system image. This configuration enhances throughput, eliminate network latency, and decreases the amount of processor resources used.

- High availability reduces downtime costs

  The proven technologies of the System z hardware and operating system have the highest availability in the industry. WebSphere Application Server for z/OS can directly benefit from this high availability. In addition, the structure of the application server expands this high availability into WebSphere Application Server itself. You can form a mini-cluster inside each application server, if activated by the administrator.

  Using a *Parallel Sysplex*, the z/OS cluster technique, increases the uptime of the environment significantly. If unplanned downtime occurs, System z and z/OS offer disaster recovery capabilities that bring the system back to a productive, industry-leading state.

- Reduced cost through manageability

  The management capabilities of the z/OS platform have evolved. The result is a platform with lower management costs and with a high degree of automation and transparency for administrators.

- Lower total cost of ownership (TCO)

  A System z platform provides a good TCO in the IT market. Independent consulting companies have shown that using a modern mainframe can outperform distributed environments that might be less expensive to purchase, but are more expensive to maintain. WebSphere Application Server for z/OS takes advantage of features, such as the IBM System z Application Assist Processor (zAAP), to reduce the software cost and the overall cost of the processing environment.

- Secure environment to stabilize operations and production

  With the use of a central security repository, the Resource Access Control Facility (RACF) can enhance the security model. It can be used for user authentication, authorization, and the role-based security model offered by Java. The security model of the operating system

prevents unauthorized user code from harming the system and bringing down the environment.

16.1.2 Benefits of using WebSphere Application Server for z/OS

This section highlights the benefits of using WebSphere Application Server V8.5 from a security, availability, and performance perspective.

Security

The use of a distinct area for user code offers more protection for other system components that run in the same logical partition (LPAR). In general, the application server has more rights than the applications that are running inside it. This level of security is necessary to ensure that the server can access all needed files, run scripts, and so on. In WebSphere Application Server for z/OS, these basic functions are run in the control region. However, the user code is run in the servant region, which generally has almost no rights. It is not possible to negatively influence system resources and services from inside the application.

Availability

The concept of a separate servant and control region greatly enhances the availability of a user application as follows:

- Multiple servant regions can form a “vertical cluster” running the application. If one servant region goes down, users with in-flight transactions in that servant receive an error. The other servant regions continue to work and respond to requests. Thus, the overall application is still available, and new requests can enter the system. z/OS starts the failed servant again automatically.
- Functions of the Intelligent Management feature provide autonomic computing abilities with self-healing and self-protecting attributes for your server environment. This feature reduces the possibility of server and component failure affecting your applications. The control region might be identified as a single point of failure (SPOF). Although the control region is unique for each application server, the risk of failure is low. Only WebSphere Application Server for z/OS product code is run in this region. To maintain availability for your application, create a WebSphere Application Server cluster as in a distributed environment.

Performance

From a performance point of view, the concept of different regions and the usage of WLM greatly enhances performance and scalability as follows:

- Performance improvements are achieved by creating multiple servant regions. This configuration allows more requests to be processed in parallel if the system has enough resources available.
- You can set detailed performance targets on a transactional level for the response time. Load balancing algorithms are available to spread work across the environment in accordance with current workload. The system adjusts resources automatically on a 24 x 7 year-round basis to ensure that the goals are kept.

16.1.3 Common concepts

Both the distributed and z/OS implementations of WebSphere Application Server V8.5 have the following common concepts:

- All Websphere Application Server components that are described in this book are common to both the distributed and z/OS platforms. These concepts include nodes, cells, clusters, core groups, job manager, administrative agent, deployment manager, and administrative console.
- Experience has shown that applications that run inside a Websphere Application Server platform on Windows, AIX, Linux, Solaris, and other systems can also run on WebSphere Application Server for z/OS. The application must meet the requirements that are common to both products. Minor modifications might be required when changing the underlying operating system.
- Websphere Application Server administrators will find the usual control options and web interfaces on z/OS.

  Using z/OS as the underlying operating system for WebSphere Application Server does not mean rebuilding your processes for administration, operation, and development. You also do not need to train administration staff on a new product. The WebSphere application programming interfaces (APIs) are the same. Also, the z/OS operating system offers additional capabilities that simplify administration and provide high availability, disaster recovery, performance settings, and management options.

16.1.4 The location service daemon

WebSphere Application Server for z/OS introduces the *location service daemon*, which is a WebSphere cell component that is exclusive to the z/OS platform. A daemon, in WebSphere Application Server for z/OS terminology, is the *location service agent*. It provides the location name service for external clients. One daemon is provided for each cell in each z/OS image (Figure 16-1). If a cell consists of multiple z/OS images, a daemon is created for each z/OS image where the cell exists. If two cells are on the same z/OS image, two daemons are created.

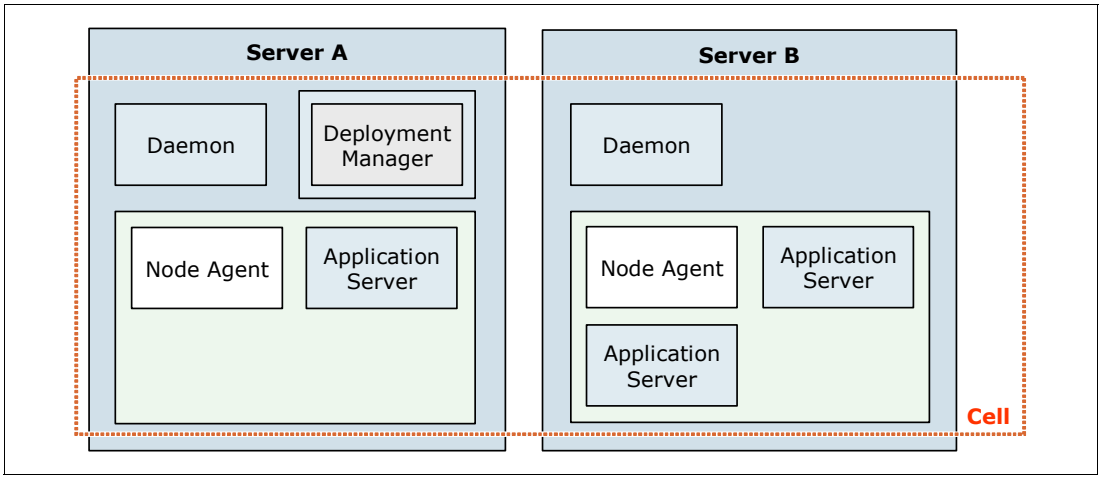


Figure 16-1 WebSphere Application Server for z/OS daemon usage in a cell

Daemon servers are started automatically when the first server for the cell on that z/OS image is started. Specifically, they are created when the first control region is started. If you terminate a daemon, all the Websphere Application Server components for the cell on that z/OS image terminate.

The daemon is created as part of the normal server customization process.

16.1.5 Structure of an application server

This section provides a conceptual view of an application server inside WebSphere Application Server for z/OS.

Overview

In WebSphere Application Server for z/OS, each application server (or instance of a profile) is built from the following building blocks:

- Control region
- Servant region
- Control region adjunct

Figure 16-2 shows these basic building blocks and how they form the application server. The communication between the control region and the servant regions is done by using *WLM queues*. Communication with the outside world ends in the control region.

Explanation: A WLM queue is used to queue work for further processing. Each queue uses a first-in first-out (FIFO) mechanism. Because it is possible to use different priorities for work requests, multiple queues exist, one for each priority. Servant regions are bound to a priority and, therefore, take work from the queue with the priority to which they are bound.

A WLM queue is a construct with which you can prioritize work requests on a transaction granularity, compared to server granularity on a distributed environment.

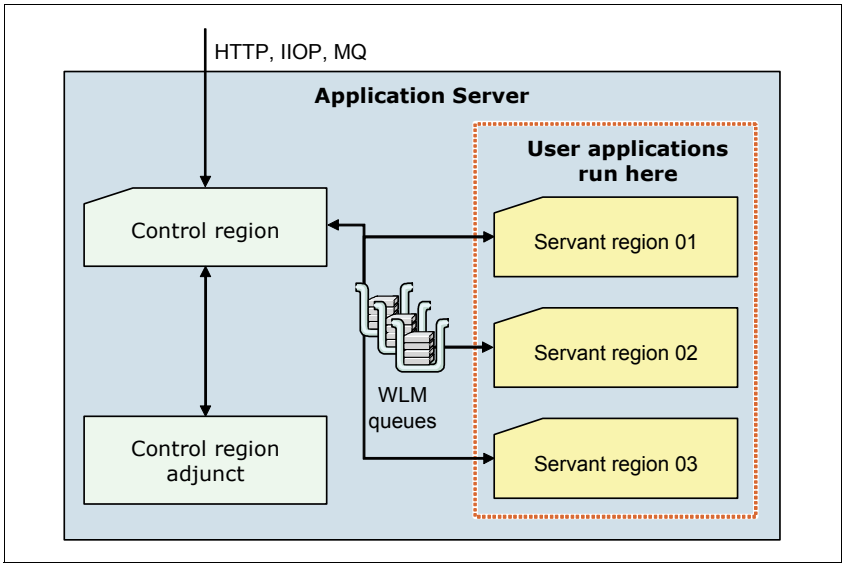


Figure 16-2 Building blocks of the WebSphere Application Server for z/OS V8.5

WebSphere Application Server profiles on z/OS are built by using multiple building blocks. However, they are still part of a single instance of an application server from an application developer, system administrator, and user perspective. Thus, nearly all WebSphere variables can be defined against a server rather than against the servant and control region adjuncts. However, some of the settings, such as heap sizes, must be defined for each component.

The following profiles are built by using the control region, servant region, and control region adjunct:

- Application server
- Deployment manager
- Job manager
- Administrative agent

Control region

The *control region* is the face of the application server to the outside world. It is the only component that is reachable from the outside world by using standard protocols and port communication. For communication with the servant regions, where the application is run, the control region is the endpoint for TCP transportation and switches to WLM queues.

Keep in mind the following points about control regions:

- An application server can have only one control region.
- The control region contains a Java virtual machine (JVM).
- The control region is the start and endpoint for communication.

Servant region

The *servant region* is the component of an application server on z/OS where the application runs and transactions are processed. The containers that run the applications are included here.

As shown in Figure 16-2 on page 505, you can have multiple servant regions for each application server. This concept is called a *multi-servant region* or *internal cluster*. This technique takes advantage of cluster benefits without the processor needs of a real cluster. For continuous availability and scalability, build a Websphere Application Server cluster that integrates these mini clusters. When creating a normal cluster, you can still use multiple servant regions for each cluster member.

Keep in mind the following information about servant regions:

- Each servant region contains its own, independent JVM.
- All servant regions are identical to each other.
- An application runs on all servant regions connected to an application server, because it is deployed at the server scope.
- An application must be Websphere Application Server cluster-ready to use the multiservant concept.
- The number of servant regions is not apparent to the user and the application.
- Servant regions can be started dynamically by the WLM component, if response times of user transactions do not meet the defined goals. The defined maximum is the limit.
- If a single servant fails, the remaining servants continue to run, keeping the application “alive.” Only the transactions of the failed servant region fail and deliver errors to the user. The other servant regions continue processing work.
- Failed servant regions are restarted automatically by the operating system, providing automation.

Tip: When determining the maximum number of servant regions, make sure that the system has enough resources to use them all.

Control region adjunct

The *control region adjunct* is a specialized servant that interfaces with new service integration buses to provide messaging services. The control region adjunct works in conjunction with the control region as the communication endpoint for messaging. It is only present when the server becomes a bus member and has a messaging engine created.

16.1.6 Runtime processes

This section describes the runtime behavior of WebSphere Application Server for z/OS V8.5.

Overview

The non-z/OS platforms are built on a single process model. Thus, the entire application server runs in a single JVM process. WebSphere Application Server for z/OS is built by using a federation of JVMs, each running in a different *address space*. Together, such a collection represents a single server instance, as illustrated in Figure 16-2 on page 505.

Explanation: An *address space* can be best compared to a process in the distributed world. Instead of running processes, the z/OS operating system uses a concept, called *address spaces*. Technically, an address space is a range of virtual addresses that the operating system assigns to a user or a separately running program, such as WebSphere Application Server for z/OS. This area is available for running instructions and storing data.

During run time, each building block of an application server or a deployment manager opens an address space such as a control region, servant region, or Deamon (Figure 16-3).

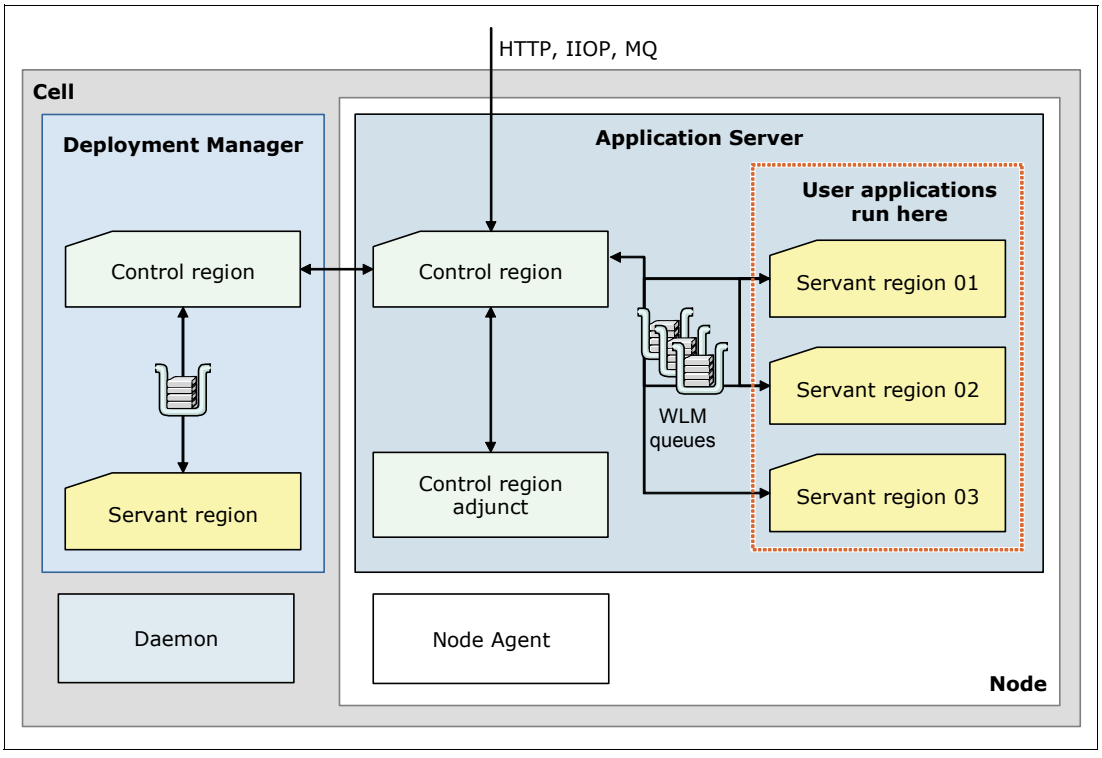


Figure 16-3 Runtime architecture of a z/OS Network Deployment cell

The WebSphere Application Server for z/OS environment shown in Figure 16-3 on page 507 includes the following address spaces:

- Application server control region
- Optional: Application server control region adjunct
- Application server servant region for each servant (the example shows three)
- Deployment manager control region
- Deployment manager servant region
- Location service daemon
- Node agent

A stand-alone server installation includes at least the following address spaces with the optional application server control region adjunct:

- Application server control region
- Application server servant (assumed that one servant is used)
- Location service daemon

Java virtual machine

Each control region and node agent, and each servant and adjunct region contains a JVM. The installation shown in Figure 16-3 on page 507 has eight JVMs: Two control regions, one node agent, four servant regions, and one adjunct.

These JVMs have special purposes. The control region JVM is used for communication with the outside world and for some base Websphere Application Server services. The servant region JVM runs the user application, and the control region adjunct hosts a bus service. In terms of specialized JVMs on z/OS, the maximum amount of heap storage defined for the various heaps is reduced. This reduction is possible because not all data and metadata needs to be loaded and kept inside the memory. It also separates the user data from most of the system data that is needed to run the Websphere Application Server base services.

The high number of JVMs has some implications on the system requirements and on the sizing of the heap:

- Amount of real storage
- Minimum and maximum size for the different heaps
- Shared class cache usage

> **Explanation:** The shared class cache is a construct introduced with Java Development Kit (JDK) 5.0. The shared class cache can be used to share the content of a JVM with other JVMs. For more information about z/OS settings for the shared class cache and its implications, see 16.4.2, “Java virtual machine settings” on page 535.

For more information, see 16.3, “Installing WebSphere Application Server for z/OS” on page 526.

16.1.7 Workload management for WebSphere Application Server for z/OS

This section focuses on how WebSphere Application Server for z/OS uses the WLM subsystem of z/OS.

Workload management overview

WebSphere Application Server for z/OS uses the WLM subsystem of z/OS in the following ways:

- Workload classification: Coarse-grained workload management on a server base.
- Transaction classification: Fine-grained workload management on a transaction level.
- Servant activation: Starts additional servant regions for application processing.

To fully use the provided capabilities of WLM, you need to configure your environment properly. For a detailed step-by-step approach, see the Websphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=crun_wlm_sessionplacement`

Before going into the enhancements that the WLM offers to WebSphere Application Server for z/OS, the following sections briefly explain the concepts of service classes, reporting classes, and enclaves.

Service classes

A *service class* is the z/OS implementation of a service level agreement (SLA). A service class is used to set performance goals for different work, such as incoming requests, applications, or operating system tasks.

For example, a user might define a service class to achieve a response time of 0.5 seconds 80% of the time for incoming requests. The WLM component of z/OS then assigns resources automatically (processor, memory, and I/O) to achieve these goals. It assigns resources by comparing the definitions of the service class to real-time data on how the system is performing.

You can have multiple service classes with multiple goals. The mapping of work to a service class is set up by the system programmer. This mapping can be based on many choices such as user ID, application, and external source.

Reporting classes

While the system is processing work, a reporting class monitors the resources that are spent processing work. A *reporting class* is an administrative construct that is used to track used resources. Each unit of work that is processed by the system is charged into one reporting class. The decision of what work is put into which report class can be defined by the z/OS system programmer (system administrator).

This grouping of used resources can then be used to tune the system or to create a charge-back to the departments that use the systems. You can create reports by using the IBM Resource Measurement Facility™ (RMF™).

Enclaves in an WebSphere Application Server for z/OS environment

An *enclave* is used to assign the user application a service class during run time. An enclave can be thought of as a container that has a service class and a reporting class attached to it. A thread can connect to this enclave and run the work of the thread with the priority of the enclave.

WebSphere Application Server for z/OS uses this technique to pass transactional work, on behalf of the user application, from a servant to an enclave. The application then runs with the priority of the enclave, and WLM can ensure that the performance goals for the application are achieved.

Workload classification

WebSphere Application Server for z/OS V8.5 and its previous versions can classify incoming work on a server basis. To begin, the control region of an application server determines which application server the request belongs to. It then assigns the request to a WLM queue. Each servant processes work for one service class at any point in time.

As shown in Figure 16-4, incoming work is assigned a service class, based on information of the user-work request. The granularity is on the application server level.

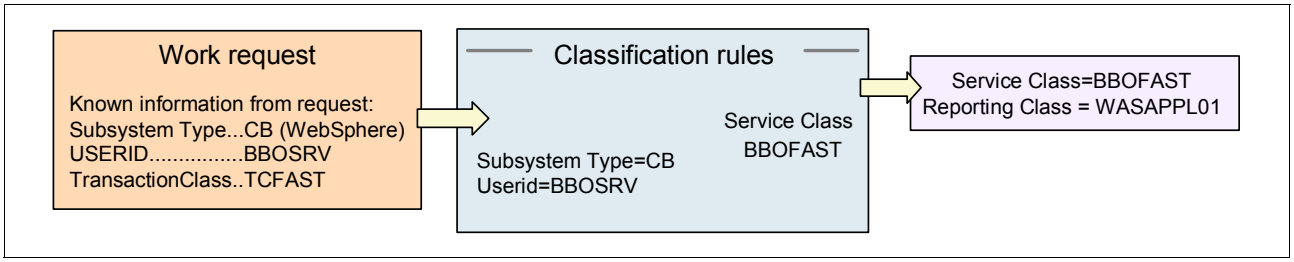


Figure 16-4 Workload classification for WebSphere Application Server for z/OS

Transaction classification

You can use transaction classification to classify the transactions that are handled by your application. You can use this technique to prioritize special requests. An example of this technique is a web store that classifies its customers as gold or platinum customers. The platinum customers are given a better response time than gold customers, as illustrated in Figure 16-5.

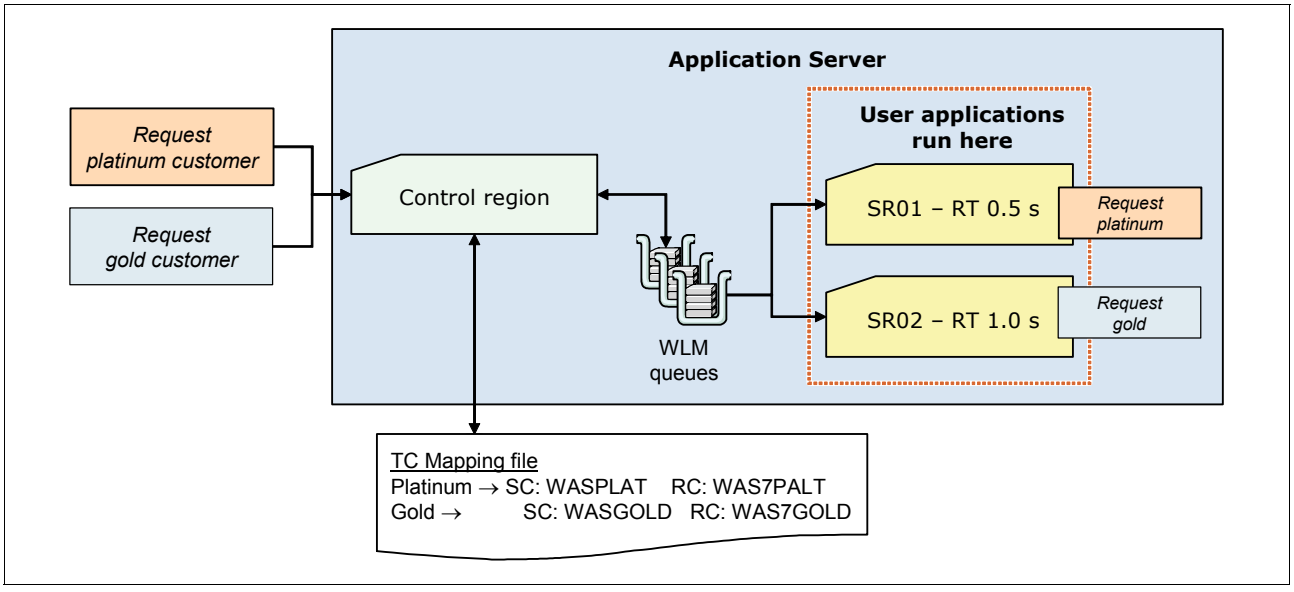


Figure 16-5 Transactional assignment of performance goals

A request that enters the system is assigned a transaction class. It is assigned by using request details such as the protocol that is used, the requested Uniform Resource Identifier (URI), or other metrics. The transaction class is then mapped to a service and reporting class

inside the WLM subsystem by using a *workload classification document*. This document is an XML file that classifies these types of requests and assigns them to a transaction class (TCLASS):

- Inbound HTTP
- Internet Inter-ORB Protocol (IIOP)
- Message-driven bean (MDB)
- Session Initiation Protocol (SIP)
- Optimized local adapter
- Mediation work requests

Remember: Use the common workload classification document method to classify work requests in a z/OS environment. Support for other WebSphere Application Server mechanisms for classifying work in a z/OS environment is deprecated.

The TCLASS value, if it is assigned, is passed to the IBM MVS Workload manager (WLM). WLM uses the TCLASS value to classify the inbound work requests and assign a service class or a report service class to each request.

For more information about the transaction classification, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=rweb_classservers

Workload classification document

A *workload classification document* is an XML file that contains classification tags that classify work requests and assign them to one of the following transaction classes:

- Inbound classification
  - HTTP classification
  - IIOP classification
  - Internal classification
  - MDB classification
  - Optimized local adapter classification
  - SIP classification
- SibClassification
  - JMS RA classification
  - Mediation classification
- WMQRAClassification
  - WebSphere MQ messaging provider classification

Tip: WebSphere Application Server for z/OS V8 improves the granularity of the workload classification. For more information, see “Improved RAS granularity for work requests” on page 523.

Servant activation

As described in “Servant region” on page 506, an application server can have multiple servant regions defined that process user application requests. If the response time goals that are defined for the applications cannot be kept, WLM can start additional servant regions. As within a normal cluster, incoming or queued requests can now be processed faster. The minimum and maximum number of servant regions can be defined by the system programmer.

16.1.8 WebSphere Application Server on z/OS and 64-bit mode

Beginning with WebSphere Application Server for z/OS V7.0, a newly created application server is configured automatically to run in 64-bit mode. This setting removes the 31-bit storage limitation.

On a 31-bit server on a z/OS system, the maximum size of the JVM heap is limited to a value of 768–900 MB. This limitation comes from the size of the private virtual storage under the 2 GB line of the z/OS address spaces. The private region is limited to approximately 1.4 GB. This amount of memory is used for the heap of the JVM and other infrastructure.

The usage of 64-bit removes this limitation and allows the definition of much larger heap sizes.

Considerations when using 31-bit mode

Keep in mind the following considerations when using 31-bit mode:

- Although you can configure the 31-bit mode manually, avoid doing so because this mode is deprecated in V8.5.
- The migration of a server from V6.1 to V8.5 does not change the bit mode. New servers start in 64-bit mode, but migrated servers use the original bit mode for which they were configured.
- You can switch a server from 31-bit mode to 64-bit mode and back again. It is not a permanent decision made at configuration time.
- Support for running a server in 31-bit mode is deprecated. When a server that is configured to run in 31-bit mode is started, a warning message is issued to the system log (Example 16-1). The *server_name* is the name of the server that is running in 31-bit mode.

Example 16-1 The 31-bit deprecation message in the z/OS system log

```
BBOO0340W: 31-BIT MODE IS DEPRECATED FOR THE APPLICATION SERVER RUNNING ON THE
Z/OS OPERATING SYSTEM. CONSIDER USING 64-BIT MODE FOR server_name AS AN
ALTERNATIVE.
```

Planning considerations

Because the 31-bit operation mode for WebSphere Application Server is deprecated in V8.5, use 64-bit mode for all planning activities for new installations. Keep in mind the considerations in the following sections when planning an installation.

Support of all components for 64-bit JVM

Make sure that all components used in your architecture support the use of 64-bit JVM. Virtually all versions of purchased software support the usage of 64-bit mode. However, this point might be of concern for user-built applications that are migrated from a 31-bit environment.

Real and auxiliary storage

The use of 64-bit mode does not imply that the amount of storage used will increase significantly. In general, 64-bit implementation increases the needed storage by approximately 20% when compared to 31-bit mode using the same JVM heap size.

Consideration: Using 64-bit mode with WebSphere Application Server for z/OS does not mean that the server needs additional memory. The amount of memory that your environment uses is based on the following factors:

- The need of the applications for storage
- Memory settings that are defined by the administrator

The difference between the 64-bit and 31-bit addressing modes is that it is theoretically possible to use larger amounts of memory with the 64-bit addressing mode. However, the amount of storage needed increases significantly only if the heap sizes are increased significantly (for example, if your application needs large heaps).

The amount of storage is controlled through administrator interaction. If the WebSphere Application Server administrator does not change the JVM memory settings, you do not need to increase the amount of real and auxiliary storage.

If applications need to use larger heap sizes than 900 MB, make sure that enough real and auxiliary storage is available.

Effect on the system

WebSphere Application Server for z/OS is part of a larger system that includes both the z/OS operating instance and the broader sysplex. Therefore, consider the entire system when increasing WebSphere for z/OS JVM heaps significantly.

Administration considerations

This section describes the changes that you need to be make to run WebSphere Application Server in 64-bit mode (default).

JCL parameters

If the application server needs a large JVM heap, ensure that the following job card parameters do not restrain the system:

- REGION setting on the JCL JOB or EXEC statement

  This setting specifies the maximum size of the execution region (between 0 M and 2 GB) for the step in this job. A value of 0 M means that the execution region takes the amount that it needs within that range with no limit imposed. Specify `REGION=0M` so that you do not limit the size of the execution region.

- MEMLIMIT setting in the JCL or in the PARMLIB member SMFPRMxx

  This setting specifies the limit on the use of virtual storage above 2 GB for a single address space. If you specify a JVM heap greater than 2 GB, the JVM heap can extend into this range. A value of `MEMLIMIT=NOLIMIT` means that the JVM heap is not limited above the 2 GB bar.

  Message BBOO0331I is issued during server start to show the MEMLIMIT value that was used for the address space, and where the value came from. The value can come from an exit, job control language (JCL), and so on.

System exits

Verify that the IEFUSI- and JES2/JES3 exits that are defined on the z/OS operating system do not limit the virtual region size for the WebSphere Application Server address spaces.

Tip: Although this modification or check is a step in the installation guide, some installers skip over basic steps when WebSphere Application Server is already installed. However, due to the theoretically larger heap sizes, you need to adjust the values in most environments.

16.1.9 XCF support for WebSphere high availability manager

WebSphere Application Server for z/OS V8.5 allows you to use the cross-system coupling facility (XCF) system services to monitor the status of cluster components. XCF can be used instead of the default common code base technique. This type of implementation provides the following value:

- Reduced processor usage

  Using the XCF reduces the processor usage that comes through the ping packets that are sent by each core group member. This reduction is noticeable during processor idle times.

- Improved interval for failure detection

  The default interval that is used in the original protocol (180 seconds) is not convenient for every environment. Using the XCF system service reduces this time. The default settings provide information after 90 seconds. These values can still be adjusted by the system programmer.

Remember: Although using the XCF system service is an option on the z/OS platform, the default setting for the core group member failure detection is the heartbeat technique. This default setting is chosen because of the common code base.

To take advantage of the enhanced high availability manager discovery and failure mechanism, the following requirements must be satisfied. Otherwise, you must use the default policy.

- Configure the z/OS IBM VTAM® component to start `XCFINIT=YES` to enable TCP/IP to use the XCF service.
- Ensure that all core group members are at WebSphere Application Server V7 or later.
- Ensure that all core group members are running on the z/OS platform.
- If the core group is bridged to another core group, ensure that all bridged groups are on z/OS in the same sysplex.

From an architectural side, using XCF adds the components shown in Figure 16-6 to a WebSphere Application Server environment.

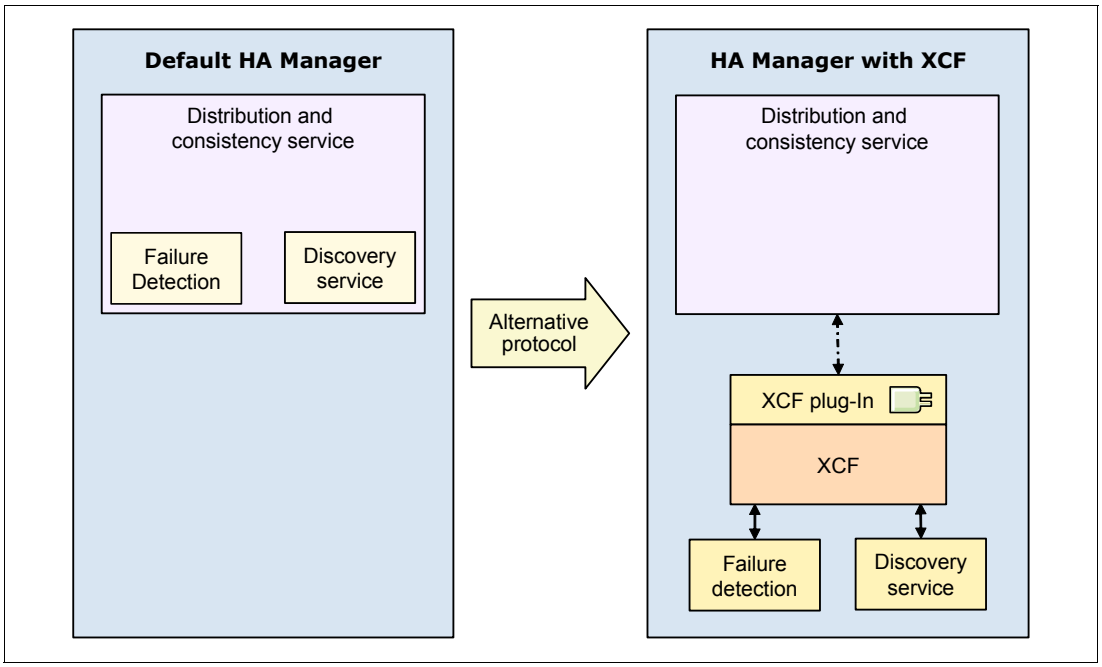


Figure 16-6 Architectural changes when using XCF support

Figure 16-6 illustrates the following main changes:

- The internal failure detection of the high availability manager is factored out of the DCS structure. Instead, the XCF failure detection is used to notify the high availability manager.
- The Discovery Service, which is used to communicate with the high availability manager, now communicates with the XCF component of z/OS.
- XCF plugs into the Distribution and Consistency Service to run the alive check and to disable the TCP/IP ping-based heartbeat.

16.1.10 z/OS Fast Response Cache Accelerator

You can configure WebSphere Application Server for z/OS to use the fast response cache accelerator (FRCA) facility of the z/OS Communications Server TCP/IP. The FRCA has been used for years inside IBM HTTP Server for z/OS to cache static content, such as pictures or HTML files.

You can use the high-speed cache to cache static and dynamic contents, such as servlets and JavaServer Pages (JSP) files, instead of using the WebSphere Application Server Dynamic Cache. FRCA also allows web traffic to be carried on an IPv6 network.

Attention: The FRCA function requires z/OS 1.9 or later.

The z/OS Communications Server TCP/IP service updates to the FRCA support are required for this function to work on z/OS Version 1.9. If the updated FRCA services are not available on the system, the application server issues a BBOO0347E or BBOO0348E error message. TCP/IP uses communications storage manager (CSM) storage to maintain the cache.

Figure 16-7 shows the changed flow of a request for a JSP that can be answered from the cache, assuming that IBM HTTP Server is also on z/OS:

- Without Fast Response Cache Accelerator, a request must be processed by TCP/IP and then by IBM HTTP Server for z/OS. This process lasts until WebSphere Application Server can answer the request from its dynamic cache.
- With Fast Response Cache Accelerator, a request to a cached JSP is recognized in the TCP/IP processing and is answered directly.

Compared to dynamic cache, the benefits of using the FRCA are a reduced response time and a reduced processor cost for the serving of requests. Tests have shown that a request served from the FRCA uses approximately 8% of the processor time as the same request in a dynamic cache environment. These advantages come from its structure, because the FRCA cache can serve incoming TCP/IP requests directly (Figure 16-7).

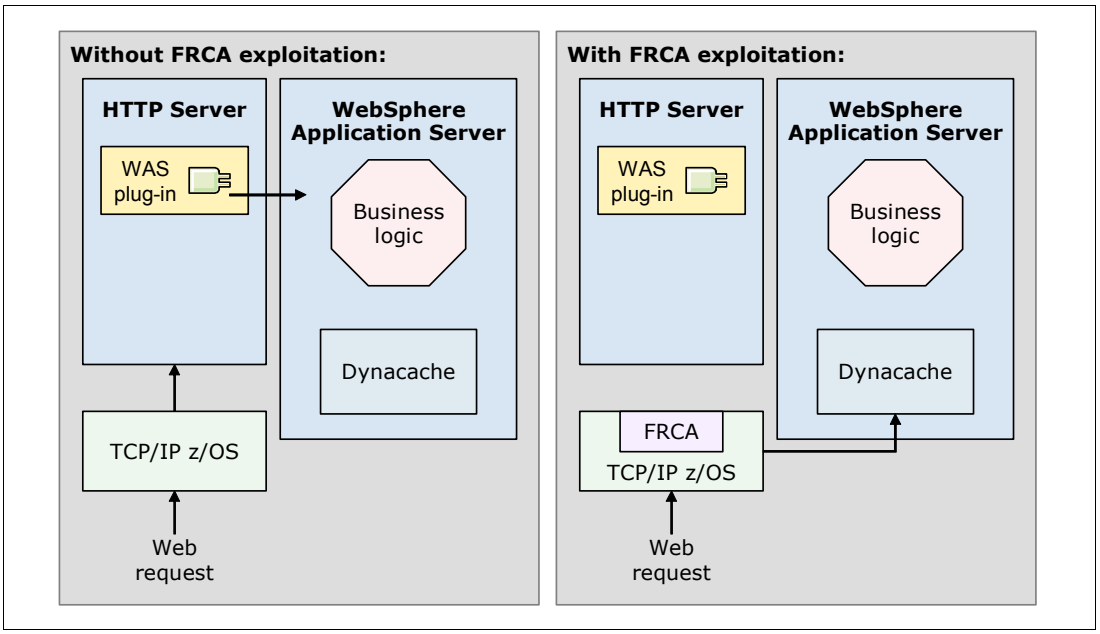


Figure 16-7 Overview of Fast Response Cache Accelerator

Restriction: FRCA cache supports only non-Secure Sockets Layer (SSL) connections.

For more information about FRCA, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tdyn_httpserverz

16.1.11 Thread Hang Recovery

Beginning with WebSphere Application Server for z/OS V7, a technique called *Thread Hang Recovery* is available. Hang detection policy can be configured for your applications and environment so that potential hangs can be reported. This process provides earlier detection of failing servers. A thread is identified as hung in one of the following situations:

- It hangs around, blocking only threads and application environment resources, such as connections and tables.
- It ends in a loop state, blocking other resources and using central processor or zAAP resources. The type of processor that is used depends on whether a zAAP is available, and on the step in the application where the error occurs.

Thread Hang Recovery directly addresses both of these issues. With this technique, you can specify thresholds for processor use and actions to run if a single request exceeds this value. This function is of real value if your environment uses high timeout values, due to long running transactions, but with few processor resources for each request. A transaction that suddenly uses a high amount of processor capacity would not have been detected in previous versions unless the normal timeout occurs. Not detecting the error in time can have a performance impact on the entire environment.

Technique for releases before WebSphere Application Server V7

In releases before Websphere Application Server V7, if a request runs into a timeout, the server assumes that the request is hung and begins to solve the situation. Depending on the recovery setting for your installation, the server has the following options:

- Terminate the servant with ABEND EC3.

  If `protocol_http_timeout_output_recovery=SERVANT` is set, the servant is terminated, and WLM starts a new servant. A dump file for the servant can be generated, and all work that was running in the servant is terminated. This option can result in penalizing work that was not having problems. In addition, server throughput is affected while the dump file is written and a new servant is started, which can take many seconds.

- Respond to the client and continue working.

  If `protocol_http_timeout_output_recovery=SESSION` is set, it is assumed that an unusual event occurred that caused the timeout and the request will eventually complete successfully. If this assumption is wrong, and the request is truly hung, the servant is left with one less thread for processing incoming work. In addition, by allowing the request to continue, deadlocks can occur because the request is holding locks or other resources. If this problem continues on subsequent requests, multiple threads become tied up. The throughput of the servant is affected, possibly to the point where it has no threads available to process work.

Current technique

Hung threads are determined by a timer valuer based on the request type. If a hung thread is detected, the servant can try to interrupt the request. To allow the servant to interrupt the request, a new registry of *interruptible objects* is introduced. Certain blocking codes can register. If too much time passes, the servant can call the interruptible object for it to attempt to unblock the thread. A Java interruptible object is always registered, so that the servant can attempt to interrupt the thread if all else fails.

Remember: WebSphere Application Server for z/OS provides the code that is used to unblock a thread. You do not need to implement code for the Interpretable Objects registry to use Thread Hang Recovery for your application serving environment.

This interruption can have the following results:

- The thread can be freed.

  In this case, the user whose request hung receives an exception. The administrator can define the type of action to take (none, svcdump, javacore, or traceback).

- The thread cannot be freed.

  If a thread cannot be freed, the system action depends on the administrator settings. The options are as follows:

  – Abend the servant.
  – Keep the servant up and running.
  – Perform a memory dump.

The basic options are still the same as in previous versions of WebSphere Application Server for z/OS. If a thread cannot be freed, the decision about whether a servant is abended or kept alive depends on the following factors:

- The amount of processor time that is used by the thread (looping or just hanging)
- Whether the servant is the last servant available
- The number of threads that are already in a hung state, within this servant

If a thread that was reported to the control region as hung completes, the control region is notified. The thread is no longer considered in the threshold determination.

The DISPLAY command

The **DISPLAY,THREADS** command shows the dispatch threads that are currently active. This command shows every dispatch thread in every servant region that is associated with the specified control region.

16.2 Functions in WebSphere Application Server for z/OS V8.5

This section highlights the functions in WebSphere Application Server for z/OS V8.5. See Chapter 2, “Concepts of WebSphere Application Server” on page 21, for an overview of the general concepts, functions, and features for WebSphere Application Server.

The following features are specific to WebSphere Application Server for z/OS V8.5:

- WebSphere optimized local adapter high availability support
- Resource Workload Routing
- Improved reliability, availability, and serviceability (RAS) granularity for work requests
- High Performance Extensible Logging (HPEL)
- Distributed identity mapping using System Authorization Facility (SAF)

16.2.1 WebSphere optimized local adapter

The WebSphere optimized local adapter is a high speed cross-memory exchange that was introduced with WebSphere Application Server for z/OS V7.0.0.4. It was enhanced with support for Information Management System (IMS) in the V7.0.0.12 fix pack and proxy function in V8.0.0.1. WebSphere optimized local adapter now allows for callers and targets to be located separately on operating system instances other than z/OS. Callers and targets include CICS regions, IMS regions, z/OS UNIX System Services processes, batch programs, and airlines line control (ALCS) regions.

The WebSphere optimized local adapter is built on a cross-memory service that WebSphere Application Server for z/OS uses for internal IIOP calls between servers on the same LPAR. This service bypasses the TCP/IP stack, avoiding network and serialization latency. This service is externalized so that programs in external address spaces can access cross-memory service to communicate with WebSphere Application for z/OS Java programs (Figure 16-8).

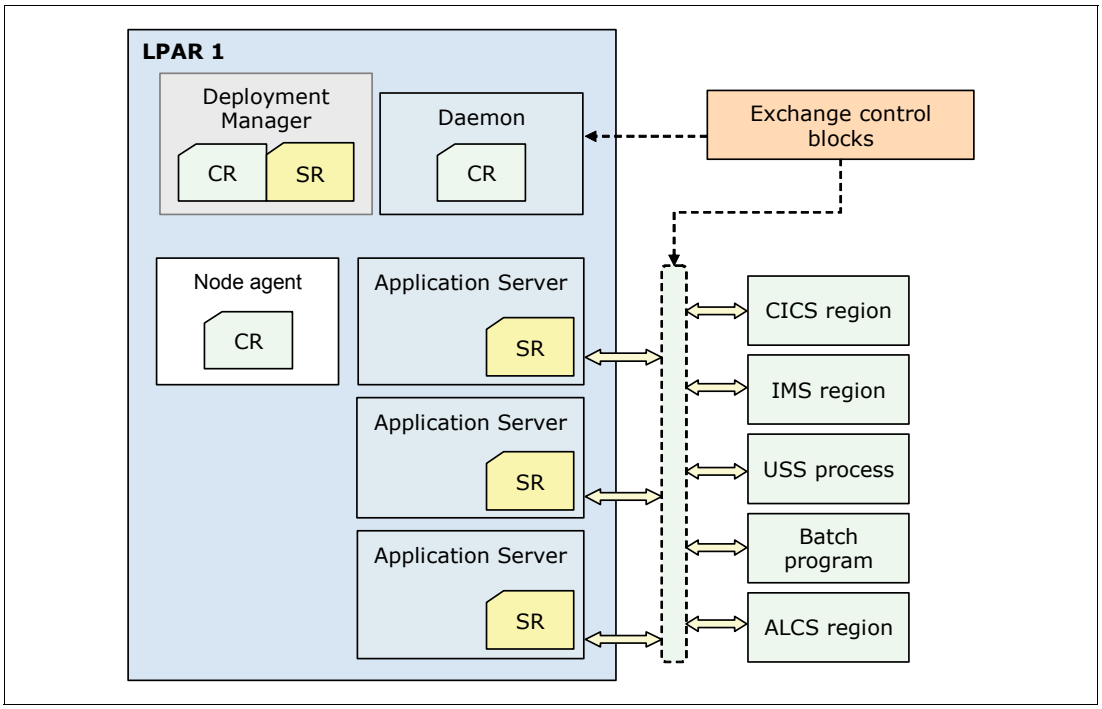


Figure 16-8 WebSphere optimized local adapter communication

The WebSphere optimized local adapter provides bidirectional communication from WebSphere Application Server to external address spaces (outbound) and from external address spaces to WebSphere Application Server (inbound). Shared memory space exchange control blocks owned by daemon are above the 2 GB bar and 64-bit native callable APIs for C/C++ are available.

Using the WebSphere optimized local adapter provides the following benefits:

- Performance improvement

  The ability to pass parameter data by using binary techniques provides a performance improvement. The transport-level support that the adapters provide uses z/OS cross-memory services. These services are used to optimize performance of calls to applications deployed on a locally accessible WebSphere Application Server for z/OS server.

  The optimized local adapters also provide a high performance local binding for existing application, middleware, and subsystems on z/OS platforms.

- Identity context propagation

  For inbound requests to WebSphere Application Server using optimized local adapter APIs, the user ID on the existing z/OS thread is always propagated and asserted in the WebSphere Application Server EJB container.

- Global transactions

  Global, two-phase commit transactions are supported with the optimized local adapters for inbound calls from CICS to WebSphere Application Server, and for outbound calls from WebSphere Application Server to CICS.

- Workload balance and availability

  With the workload balancing framework in the optimized local adapter support, the inbound call requests are passed to the target server control region. In the target server control region, the requests are queued by using z/OS workload management to an eligible servant region for execution.

- A gateway or proxy for existing assets on z/OS systems

  Built-in optimized local adapters provide the basis for you to begin to use the WebSphere Application Server for z/OS stack as an easily accessible set of capabilities. These capacities extend the life of application assets that might be difficult to replace.

- Audit and accountability

  WebSphere Application Server for z/OS produces detailed SMF records to provide record statistics structure and integrate into existing accounting processes. SMF 120 type 9 records are created with information about inbound WebSphere optimized local adapters (WOLA) calls. SMF creates 120 type 10 records with information about outbound calls.

Planning to use WebSphere optimized local adapter for z/OS systems

When using the WebSphere optimized local adapter for z/OS systems, keep in mind the following considerations:

- Review existing business and middleware applications in your environment to determine which process might benefit from using optimized local adapters.
- Make sure that you are running WebSphere Application Server in 64-bit mode.
- Make sure that WebSphere Application Server is using an SAF-based user registry if you plan to propagate an SAF user ID from WebSphere Application Server for z/OS to the enterprise information system (EIS).
- Review the optimized local adapter examples. Several samples are included when you install WebSphere Application Server for z/OS.
- Decide how to use optimized local adapters. You can use it to make inbound or outbound calls.

For more information about the WebSphere optimized local adapter, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=cdat_ola

16.2.2 Resource workload routing

Resource workload routing includes data source and connection factory failover and subsequent failback from a predefined alternate or backup resource. With this function, applications can recover easily from resource outages, such as database failures, without requiring you to embed alternate resource and configuration information. You can tailor the resource failover and failback flexible configuration options to meet your environment-specific and application needs. This feature is common across platforms.

WebSphere Application Server for z/OS V8.5 has three configurable actions, called *action notification*. You can configure an action notification for a resource. If requests for that

resource fail past a specified threshold value, the WebSphere Application Server for z/OS run time performs the action that you configured.

The various failure notification actions assist with high availability environments so that, when a resource failure occurs, work can be routed to other servers in a cluster. The following action codes can be used:

- Action code 1

  Action code 1 provides a notification to WebSphere administrators so that manual or automated mitigation actions can be configured outside of the application server. It issues a BBOJ0130I message to hardcopy in the control region that contains the following information:

  - The Java Naming and Directory Interface (JNDI) name that identifies the resource that has failed
  - The name of the server where the resource that has failed was used
  - The action that was taken, for example NONE or PAUSING LISTENERS

  When the resource is available again, a BBOJ0131I message is issued to a hardcopy in the control region, indicating that the resource is again available. BBOJ0131I contains the following information:

  - The JNDI name that identifies the restarted resource
  - The name of the server on which the resource is restarted
  - The action that was taken, for example NONE or RESUMING LISTENERS
  - The reason the action was taken:
    - Normal servant region availability notification
    - Unknown resource availability

- Action code 2

  Action code 2 pauses and resumes the listeners on the server where the resource is that this action was configured for. Server listeners are paused when the resource is deemed unavailable. When combined with a front-end router that supports high availability (a proxy server or an on-demand router), work for this server is routed to other servers in the cluster. As part of this action, BBOJ0130I messages are issued to a hardcopy in the control region when the resource is deemed unavailable.

  When the resource is available again, a BBOJ0131I message is issued to a hardcopy in the control region. The server listeners resume, which restores the ability of the server to receive incoming work.

- Action code 3

  Action code 3 stops and starts all applications with locally installed modules that use the resource for which this action was configured. Applications are stopped when the resource that these applications use is deemed unavailable. As part of this action, a BBOJ0130I message is issued to a hardcopy in the control region when the resource is deemed unavailable.

  Attention: The only applications for which a resource reference is defined are stopped on the server that experienced the resource failure. Therefore, if the application is installed in a cluster, the application continues running on the other servers in the cluster.

The BBOJ0130I message contains the following information:

– The JNDI name that identifies the resource that has failed
– The name of the server where the resource that has failed was used
– The action that was taken, for example NONE, PAUSING LISTENERS, and STOPPING APPLICATIONS THAT USE THIS RESOURCE

When the resource is available again, a BBOJ0131I message is issued to a hardcopy in the control region. Then all applications with locally installed modules that use this resource for which this action was configured are started.

The BBOJ0131I message contains the following information:

– The JNDI name that identifies the resource that is restarted
– The name of the server on which the resource is restarted
– The action that was taken, for example NONE, RESUMING LISTENERS, and STARTING APPLICATIONS THAT USE THIS RESOURCE
– The reason the action was taken:
  • Normal servant region availability notification
  • Unknown resource availability

In WebSphere Application server for z/OS V8.5, the WOLA participate in high availability support for WebSphere Application Server. With this support, you can specify an alternate connection factory JNDI name in the connection factory pool custom properties. Figure 16-9 shows a representation of the resource workload routing.

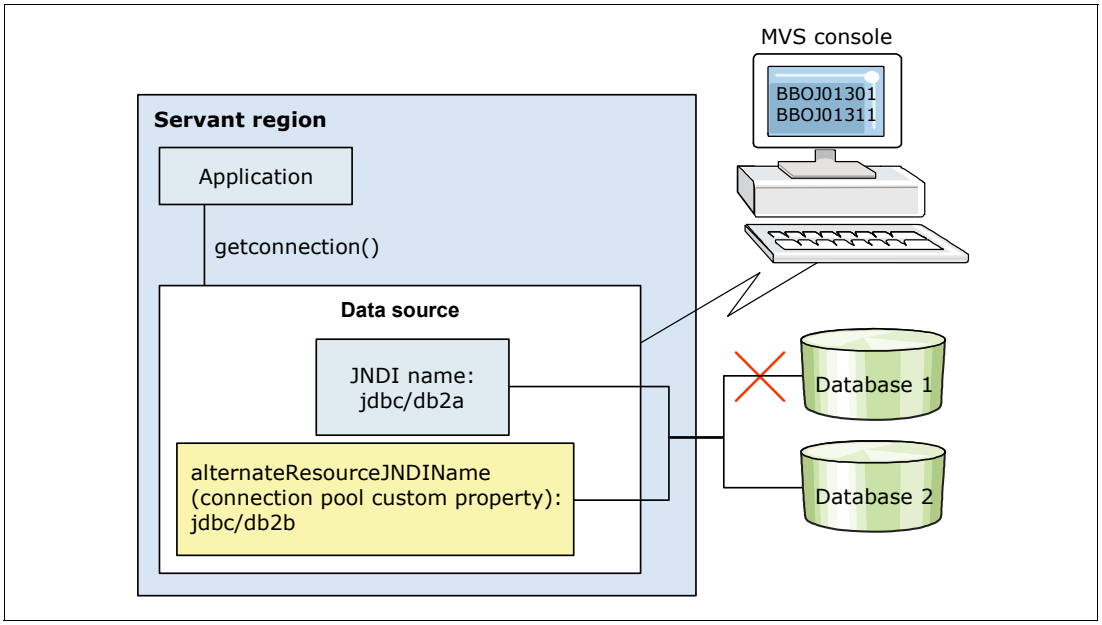


Figure 16-9 Resource workload routing

The optimized local adapter resource failover process is triggered the same way as for other resource adapters. When an application makes a `getconnection()` request for a resource that fails because the target registration is not available, the failover process is triggered. During this process, the alternate JNDI resource name is used for the `getconnection()` instead, as shown in Figure 16-10.

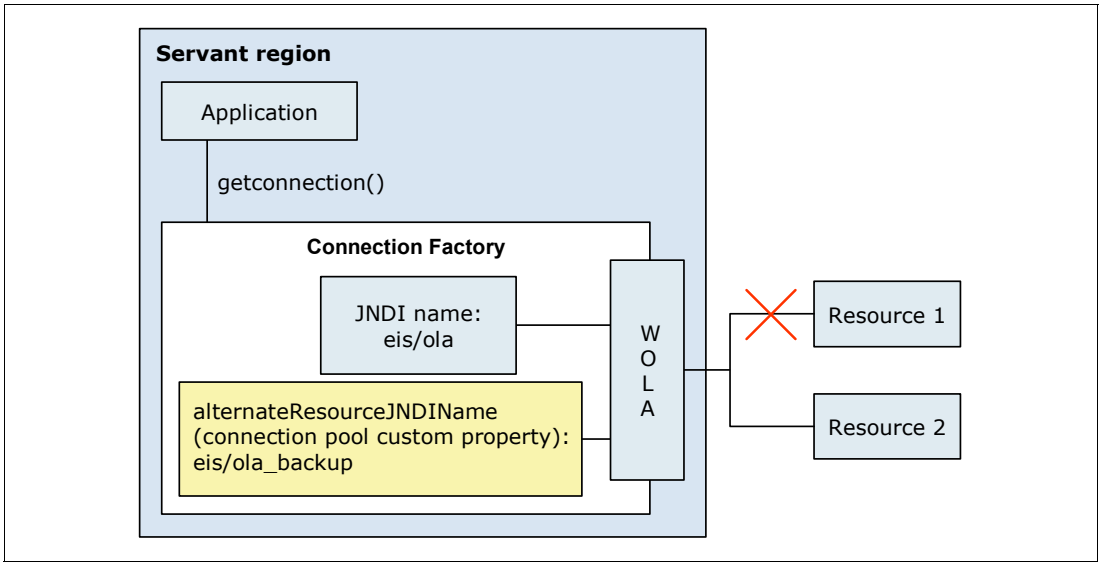


Figure 16-10 WebSphere optimized local adapter failover process

A failover event triggers a process to send subsequent requests to the alternate resource with the alternate JNDI name and register name. It initiates a polling process where WebSphere Application Server connection management sends a request every 10 seconds to determine if the primary resource is available again.

The WebSphere optimized local adapter resource adapter detects that the primary resource is available again or that the register name is active. When this detection occurs, WebSphere Application Server connection management is notified that future requests for the primary resource JNDI can be routed back to the primary connection factory.

Improved RAS granularity for work requests

RAS granularity is the ability to assign different sets of RAS attribute values to different sets of requests. The fineness of RAS granularity depends on how uniquely the application server can distinguish one set of requests from another. You can define RAS granularity in the workload classification file.

Previously, RAS granularity was limited to a per-server basis or, for a few RAS attributes, a per-protocol basis as follows:

- *Per-server RAS granularity* means that a single set of RAS attribute values is defined in the server configuration. These values apply to all requests that the application server processes.

  An example of a per-server RAS attribute is the trace setting. You can define only one trace setting for an application server. This trace setting applies to all requests that the application server processes.

- *Per-protocol RAS granularity* means that multiple sets of RAS attribute values can be defined in the server configuration, one set for each protocol. The application server divides requests into sets based on the request protocol, such as the HTTP or IIOP protocols. The application server then applies the set of RAS attribute values defined for that protocol to the requests for that protocol.

  An example of a per-protocol RAS attribute is the dispatch timeout.

Starting with WebSphere Application Server for z/OS V8, you can achieve finer RAS granularity by defining RAS attribute values on a per-workload-classification basis (request-level). *Per-workload-classification RAS granularity* means that you can define multiple sets of RAS attribute values in the server configuration. You can define one for each workload classification element in the workload classification file. The application server classifies requests based on the workload classification elements. It then applies the set of RAS attribute values that are defined for that workload classification element to those requests.

16.2.3 High Performance Extensible Logging and Cross Component Trace

HPEL provides a convenient mechanism for storing and accessing log, trace, `System.err`, and `System.out` information produced by the application server or your applications. It is an alternative to the existing log and trace facilities offered on the z/OS platform. z/OS platform uses Job Entry Subsystem (JES), LogStreams, Component Trace, hierarchical file system (HFS), or other facilities.

HPEL is easy to configure and understand. For example, administrators can configure how much disk space to dedicate to logs or traces. They can also configure how long to retain log and trace records, leaving the management of log and trace content up to the server.

For more information about HPEL, see 12.8.1, “Log and traces” on page 405.

Cross-component trace can be enabled with either of the logging types for trace correlation. This application allows entries that are serviced by more than one thread, process, or server to be identified as belonging to the same unit of work. For more information, see 12.9, “Cross-component trace” on page 412.

16.2.4 Distributed identity mapping using SAF

Distributed identity mapping is a new feature in SAF, introduced with z/OS 1.11. Distributed identity mapping in SAF provides the following major benefits:

- When a user is audited on the z/OS operating system using System Management Facilities (SMF), the audit record contains the distributed identity and the mapped SAF user ID. This information improves cross-platform interoperability and provides value for both host-centric and heterogeneous application environments.
- The mapping of distributed identities is handled by the z/OS security administrator. You do not need to configure mapping modules in the WebSphere Application Server configuration.

With this release of WebSphere Application Server, you can use z/OS SAF security to associate an SAF user ID with a distributed identity. You can log on to a WebSphere Application Server application with the distributed identity of the user. The filters defined in the z/OS security product then determine the mapping of the distributed identity to an SAF user, as shown on Figure 16-11.

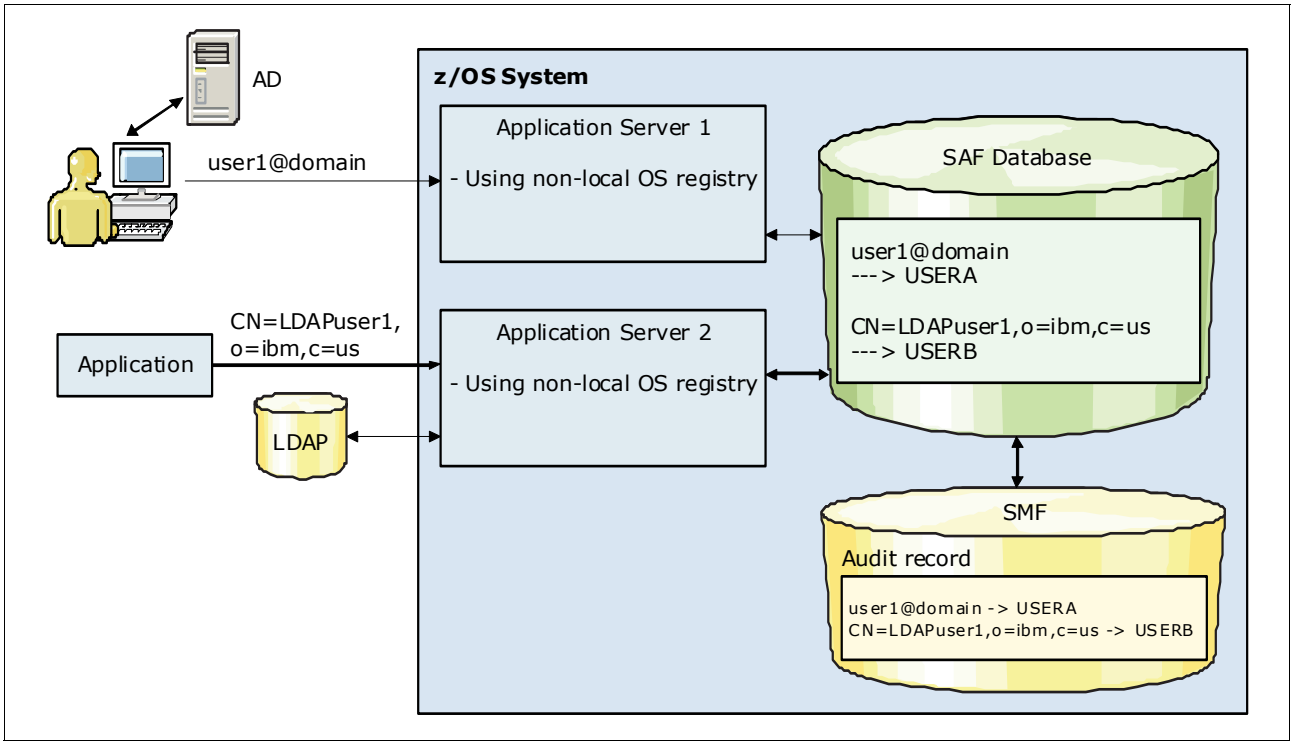


Figure 16-11 Distributed identity mapping using an SAF

When you use this feature, you can maintain the original identity information of a user for audit purposes, and have less to configure in WebSphere Application Server.

Restriction: The SAF distributed identity mapping feature is not supported in a mixed-version cell (nodes before WebSphere Application Server V8).

Considerations

When you configure distributed identity mapping, you must complete the following actions:

- Determine the SAF version. You must first ensure that your z/OS security version is at SAF V7760 or later. If you are using RACF, you must be at z/OS V1.11 or later. You can use the new AdminTask, `isSAFVersionValidForIdentityMapping()`, to determine the version. Additionally, the SECJ6233I informational message is printed in the server job log, which indicates the current SAF version.
- If you migrated your cell to WebSphere Application Server V8, remove unnecessary Java Authentication and Authorization Service (JAAS) login modules. Ensure that you do not have the `com.ibm.ws.security.common.auth.module.MapPlatformSubject` login JAAS module configured in the WebSphere configuration.

Scenarios for using distributed identity mapping for SAF

You can use the distributed identity mapping feature in SAF in the following scenarios:

- Scenario 1

  You have a non-local OS registry configured with SAF authorization, z/OS thread identity synchronization (SyncToThread), or the connection manager RunAs thread identity option. In this case, you can use this feature to map your registry user to an SAF user. To enable distributed identity mapping for this scenario, no further changes are needed in the security configuration on the WebSphere console.

- Scenario 2

  You have a local OS registry configured on z/OS operating systems with the Kerberos or Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) authentication mechanism. In this case, you might want to map a Kerberos user to an SAF.

- Scenario 3

  When you have a local OS registry configured, you can map an asserted certificate or an asserted distinguished name to an SAF user.

- Scenario 4

  When you have a Local OS registry configured, you can map a certificate received in the CSIv2 transport layer to an SAF user.

Tip: A new RACMAP command is available in the z/OS security product to configure a distributed identity filter. Use this filter to map multiple distributed users to one SAF user, or use a one-to-one mapping. The distributed identity filter consists of two parts:

- The distributed user name
- The realm name of the registry where the distributed user exists

For more information, see the Websphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tsec_use_identity_saf`

16.3 Installing WebSphere Application Server for z/OS

This section provides an overview of the installation and configuration process for WebSphere Application Server for z/OS V8.5.

16.3.1 Installation overview

Starting with WebSphere Application Server V8.0, IBM introduced IBM Installation Manager as the default z/OS product to install, update, and provide maintenance to WebSphere Application Server environment. SMP/E can be used to install the initial repositories used by Installation Manager to run the actual product installation. These repositories can be updated with SMP/E to contain newer fix pack levels. Fix packs can also be installed directly with Installation Manager from a web-based service repository. They can also be downloaded and installed on z/OS without requiring direct access from z/OS to the Internet.

IBM Installation Manager is also used to apply interim fixes, which replace ++APAR fixes in WebSphere Application Server V8.0 and V8.5

To install WebSphere Application Server for z/OS V8.5 on z/OS, perform these steps:

1. Prepare the system.
2. Install the product repositories by using SMP/E (ServerPac or CBPDO), or upload the compressed repositories (compressed files) from the product media to z/OS and extract the files.
3. Install the product by using the product repositories and IBM Installation Manager for z/OS.
4. Use the Profile Management Tool for z/OS (from WebSphere Customization toolbox) or `native zpmt.sh` script to configure the product and create the job control language (JCL) to define the profile. Then run the jobs to create the actual profile.

> **Remember:** Beginning with WebSphere Application Server for z/OS V7, the SBBOLOAD, SBBGLOAD, SBBOLD2, and SBBOLPA data sets no longer exist. The load modules are now in the product file system, which reduces complexity. To switch a configuration to using load modules in a data set, use the `server_dlls_in_hfs` custom property and the switchModules.sh tool. For more information, see the Websphere Application Server V8.5 Information Center at:
>
> `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=rins_switchmod`

For more information, including checklists for z/OS, see the Websphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=welc_howdoi_tins`

16.3.2 Installation considerations

This section includes general considerations when installing a WebSphere Application Server for z/OS V8.5.

General environment considerations

Planning your environment is critical. For more information, see Chapter 7, “Infrastructure” on page 161.

Naming convention

When installing WebSphere Application Server for z/OS V8.5, use a good naming convention. The operating system restriction to eight characters limits your naming convention to the use of abbreviations. Keep in mind that new administrative components were introduced in WebSphere Application Server for z/OS V7 such as the administrative agent and job manager. Ensure that your naming convention also reflects this information.

> **Tip:** The naming convention guidelines from the Washington System Center are included in the configuration tool for profile creation. For more information about Washington System Center, see “z/OS customization worksheet” on page 531.

Real memory defined

WebSphere Application Server for z/OS has a different blueprint than Websphere Application Server for distributed environments. Multiple heaps, one for every control and servant region, result in different memory requirements.

The heaps that are defined in a Websphere Application Server environment must fit into real memory. Not having the heap in real memory can have a negative performance impact due to paging during garbage collection. The garbage collection for a JVM heap requires all pages of the heap to be in exclusive access and in real memory. If any pages are not in real storage, they must first be paged in.

Ensure that the LPAR that is used for the installation has enough real storage defined for calculating the maximum heap sizes of the projected server's components. Account for WebSphere Application Server for z/OS address spaces that do not create JVMs. Also add storage for the operating system and other applications that run in this LPAR, such as DB2 and CICS.

Remember: Monitor your system and check for swapping. Swapping can have a major impact on performance.

Heap sizes (minimum or maximum) defined

Usually, the z/OS version needs smaller maximum heap sizes than the distributed version, because it has specialized heaps in its structure. This heap size is of interest when migrating an application from another platform to WebSphere Application Server for z/OS V8.5.

Often, the memory size from the distributed environment is carried on from the distributed environment and reused for the control and servant regions settings. This configuration can be a waste of memory resources, and it can affect performance. If the heap is sized too large, garbage collection runs less often, but when it runs, it takes up more time, reducing the general throughput.

Attention: If you migrate an application to WebSphere Application Server for z/OS V8.5 from another operating system family, perform a verbose garbage collection analysis. This analysis helps size the heap to a minimum and maximum value. Set these values so that the performance is not derogated and no resources are wasted.

IBM System z Application Assist Processor usage

zAAP is a processor that is dedicated to the execution of Java and XML work. Consider using zAAP in your WebSphere Application Server for z/OS environment for the following reasons:

- Reduced software cost

  A workload that runs on zAAP does not count toward the monthly z/OS software bill. Because Websphere Application Server is mainly written in Java, it can use zAAP. Most environments have about 80% of the WebSphere environment (WebSphere Application Server for z/OS and the applications inside) running on zAAP. The use depends on the amount of Java Native Interface (JNI) calls and other functions not based on Java that are used in the application.

- Performance gain

  The zAAP implementation offers a dedicated IBM Processor Resource/Systems Manager™ (IBM PR/SM™) processor-pool. Units of work that are dispatched run in their own world. Because fewer units compete for processor resources, units do not have to wait as long until they can access the processor.

You must configure zAAP in the LPAR profile through the Hardware Management Console (HMC) of the System z platform. zAAP can be used as a shared or a dedicated processor, depending on the LPAR setting.

If zAAP is not physically installed, you can use the RMF to project the amount of processor seconds that can be run on zAAP. For more information, see IBM System z Application Assist Processor (zAAP) at:

http://www.ibm.com/systems/z/advantages/zaap/resources.html

File system considerations

When installing WebSphere Application Server for z/OS, keep in mind the following considerations about the file system, regardless of whether an HFS or IBM zSeries file system (zFS) is used:

- Separate configuration file systems for each node

  Although file systems can be shared across multiple z/OS images in a Parallel Sysplex, create dedicated file systems for each node to improve performance.

- Product file system mounted read-only

  A read-only mount improves performance and prevents the change of file system contents.

- Separate logging file system

  A separate file system for the sole purpose of logging and tracing on a high-speed shared disk might be appropriate for some WebSphere Application Server features. These features include transaction recovery logging.

16.3.3 Function modification identifiers

Table 16-1 lists the function modification identifiers (FMIDs) for WebSphere Application Server for z/OS.

Table 16-1 FMIDs for WebSphere Application Server for z/OS V8.5

FMID	CompID	Component name
HBBO850	5655I3500	WebSphere Application Server for z/OS V8.5
HBBO850	5655N0212	DMZ Secure Proxy Server V8.5
HBBO850	5655I3511	Web server plug-ins V8.5
HBBO850	5655I3510	IBM HTTP Server for WebSphere V8.5
HBJA700	5655W6507	IBM JDK 7.0

Table 16-2 contains the upgrade and subset values for WebSphere Application Server for z/OS V8.5 and IBM Installation Manager.

Table 16-2 Preventive Service Planning upgrade and subset ID

Upgrade	Subset	Description
WASAS850	HBBO850	WebSphere Application Server for z/OS V8.5
WASAS850	HBJA700	IBM JDK 7.0
IIMZOSV1	HGIN140	Installation Manager for z/OS V1.4

16.3.4 Install repositories with SMP/E

The product repositories for WebSphere Application Server V8.5 can be installed on your z/OS system by using SMP/E.

Contact the IBM Software Support Center for information about Preventive Service Planning (PSP) upgrades for WebSphere Application Server for z/OS. For more information about PSP upgrades, see the WebSphere Application Server for z/OS: Program Directory. Although the Program Directory contains a list of required PTFs, the most current information is available from the IBM Software Support Center.

For more information about maintenance, see the Websphere Application Server V8.5 Information Center at:

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=/com.ibm.websphere.installation.zseries.doc/ae/rins_reqsdrive.html

16.3.5 Copy repositories from media (DVD)

The product repositories for WebSphere Application Server V8.5 can also be uploaded to your z/OS system from the product media (physical DVD or downloaded DVD image). A separate Program Directory, shipped with the product, describes this process.

16.3.6 Creating a product image with Installation Manager for z/OS

IBM Installation Manager is an Eclipse-based tool that allows you to manage all of the aspects of installing WebSphere software. Its behavior and methodology is consistent across platforms. It uses software packages and software package groups to install, uninstall, and modify products from repositories. It also provides complete lifecycle management of supported products.

For more information about Installation Manager concepts and terminology, see 9.6, “IBM Installation Manager” on page 242.

Installation manager uses the following locations for its data:

- Binary location: Where Installation Manager is installed.
- Agent data location: Where Installation Manager stores data associated to an application, including state and operations history. It is also known as appDataLocation.
- Object Cache location: Used by Installation Manager for shared resources and objects used for rollback.
- Command-line mode: Uses the Installation Manager command line (**imcl**) tool commands and arguments.
- Console mode: Uses the Installation Manager command line (**imcl**) tool in console mode. Use the Installation Manager console mode rlogin or Telnet to connect into the UNIX System Services shell session. Do not use OMVS shell due to square bracket translation.
- Silent mode: Silent mode uses the **imcl** command in conjunction with response files.
- Batch mode: Uses the BPXBATCH or other UNIX System Services utility to run **imcl** with parameters.

Attention: If you want the Liberty profile feature, install it as part of the package group with WebSphere Application Server for z/OS. It can be added at a later time, but it must be installed as another package group and into a separate installation directory. You cannot modify an existing WebSphere Application server for z/OS package group to add the Liberty profile feature.

16.3.7 Customization

After the SMP/E installation is complete, you can configure the product. Figure 16-12 illustrates the configuration process. The configuration includes creating the server profile and running it on the host by using the following tools and resources:

- The customization worksheet
- The WebSphere Customization Toolbox
- The `zpmt.sh` z/OS script (command line, batch style, or silent)

This section briefly describes these tools.

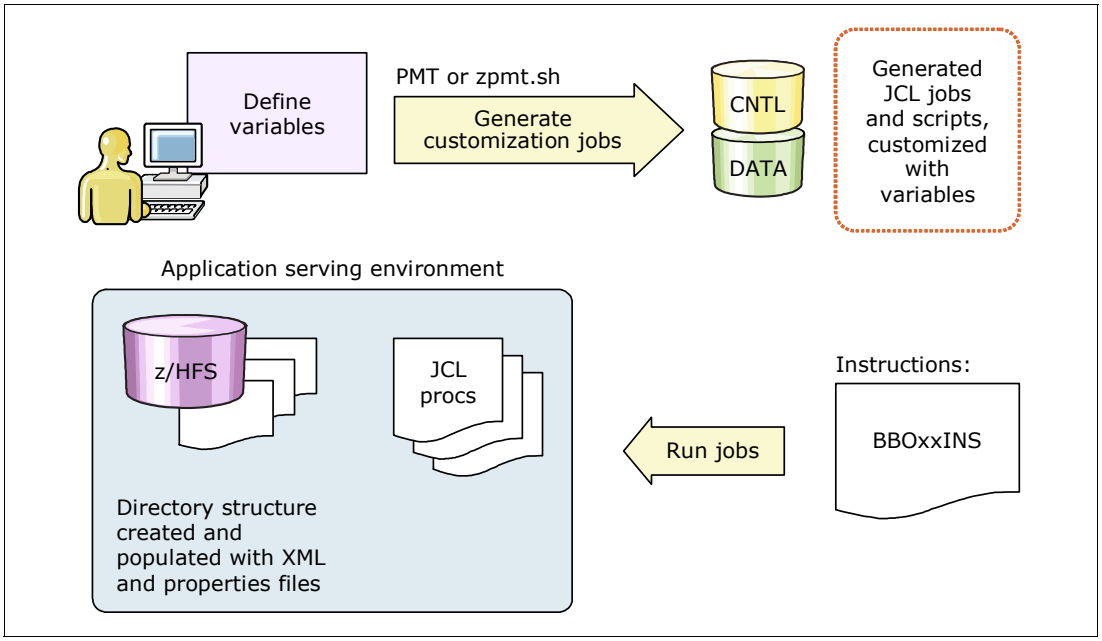


Figure 16-12 Configuration overview of WebSphere Application Server for z/OS

z/OS customization worksheet

Print the z/OS customization worksheet, and use it when collecting information about the customization variables. To obtain the configuration worksheet, see the Websphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tins_pmtstrt`

The following planning worksheets are available:

- Stand-alone application servers
- Deployment managers
- Managed (custom) nodes
- Federating application servers
- Network Deployment cells with application servers

- Job managers
- Administrative agents
- Secure proxy servers
- Secure proxy administrative agents

Tip: The Washington System Center created a version of the planning spreadsheet called *WebSphere for z/OS V8 - Configuration Spreadsheet* (Reference #PRS4686). You can use this spreadsheet during the customization process. You can obtain the spreadsheet at:

`http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS4686`

This resource includes the following planning spreadsheets:

- Network Deployment Cell
- Stand-alone server
- DMZ (Secure Proxy)

You can use the worksheet to enter multiple variables to define your installation. You can save the entered data and use it as a response file for the graphical WebSphere Customization Toolbox or the command-line `zpmt.sh` tool. These tools then generate the actual JCL that creates the profiles (Figure 16-13).

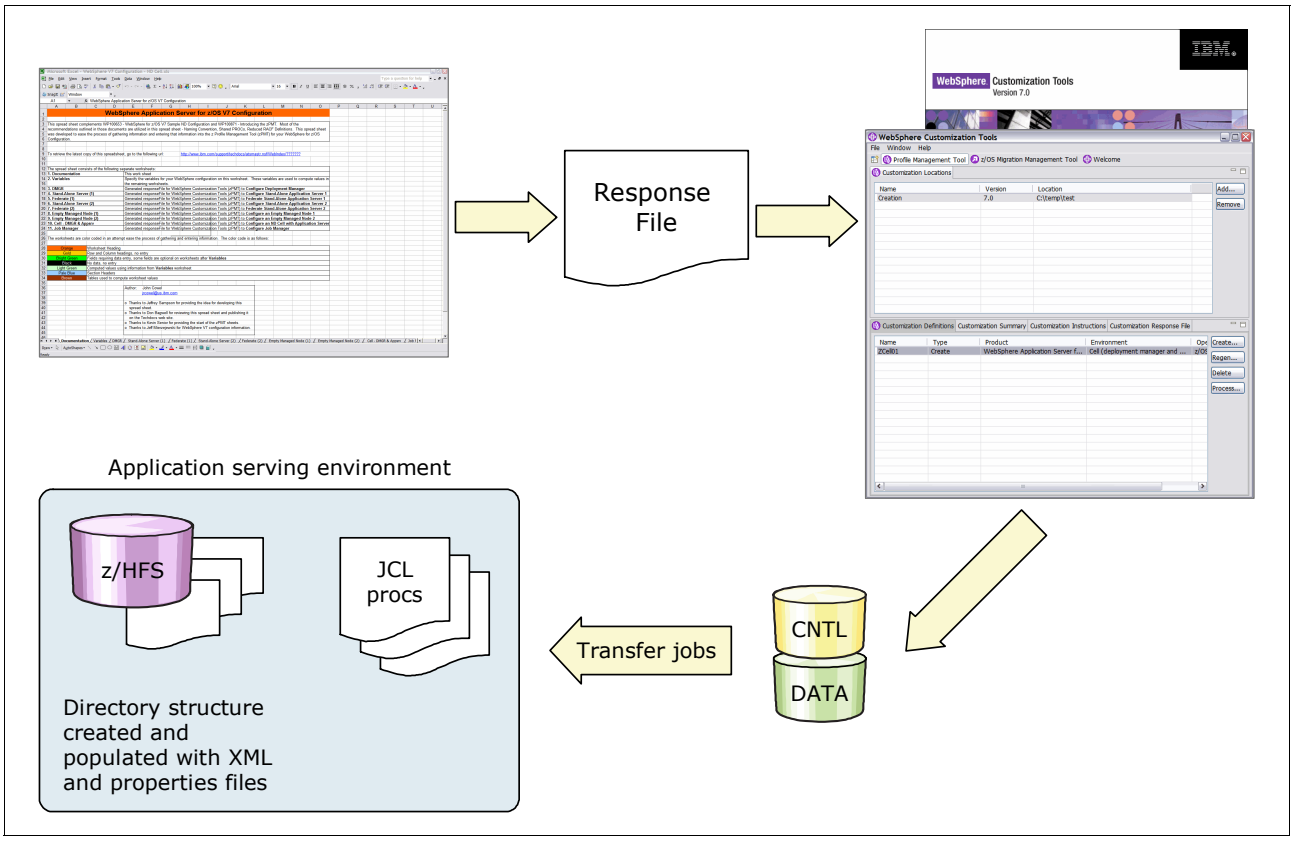


Figure 16-13 Using a planning spreadsheet for WebSphere Application Server for z/OS

WebSphere Customization Toolbox

In WebSphere Application Server V8.5, you must use the WebSphere customization tools for to configure the product. WebSphere Customization Toolbox can be installed by using the Installation Manager on supported platforms.

Deprecation: The Interactive System Productivity Facility (ISPF) panel configuration is no longer available in WebSphere Application Server for z/OS V8.5.

The following set of tools is available for Windows and Linux based workstations:

- Profile Management Tool (z/OS only) for creating profiles

  The Profile Management Tool for z/OS is an Eclipse-based plug-in that runs on a supported workstation. It is used to collect configuration values that it uses to generate data sets with the customization jobs that create the profiles. The z/OS customization worksheet or WSC configuration spreadsheet can be used as input in the form of a response file to this tool. The output of the Profile Management Tool is two data sets that contain sets of jobs that are uploaded to the z/OS system.

- z/OS Migration Management Tool for migrating nodes

  The z/OS Migration Management Tool is an Eclipse-based plug-in that runs on a supported workstation. It is used to collect configuration values for WebSphere Application Server node by node migration. It generates the data sets with JCL that migrate the profiles. The output of the z/OS Migration Management Tool is two data sets that contain set of jobs that are uploaded to the z/OS system.

For more information about how to install and use these tools, see the Websphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tins_wct`

The zpmt.sh script

The `zpmt.sh` script is the silent implementation of the Profile Management Tool on the z/OS host. You use a command-line call to the script, including various parameters and response files. It then creates the `.CNTL` and `.DATA` members that correspond to the response file and are necessary to build WebSphere Application Server. You can configure this script to allocate and copy the members from the z/OS file system to the z/OS data sets. Figure 16-14 shows an overview of this script.

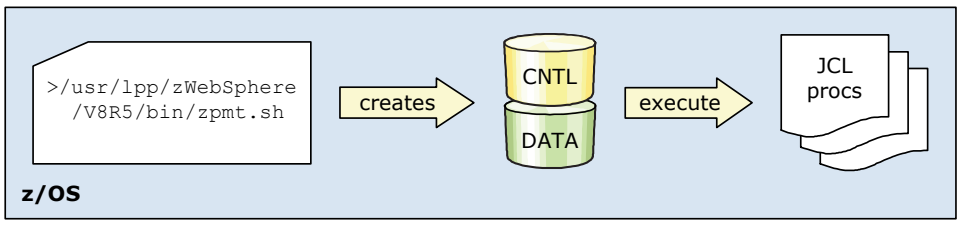


Figure 16-14 Overview of zpmt.sh configuration script

You can find the script in the `/usr/lpp/zWebSphere/V8R5/bin` default WebSphere Application Server for z/OS product image directory. Running the script opens the OSGi command shell.

Explanation: An OSGi command shell is an execution environment that allows remote management of the Java application and components. It is based on the OSGi open standard.

Although it might first look as though nothing is happening, the shell eventually shows status messages.

Running customization jobs

The second step in the customization is running the JCL created by using one of the techniques described in 16.3.7, “Customization” on page 531.

Table 16-3 shows the jobs that are necessary for the customization of WebSphere Application Server for z/OS V8.5. The jobs are similar to V6.1, but with a few notable differences.

Attention: Read the BBOxxINS member of the *hlq*.CNTL data set. It contains tasks that you must perform before you start the installation process. It also explains each job that you need to run.

Table 16-3 Installation jobs for WebSphere Application Server for z/OS V8.5

Job name	Description
BBOxxINS	Instruction member that contains the installation steps.
BBOSBRAK	Creates common groups and users for WebSphere Application Server for z/OS run time.
BBOSBRAM	Creates home directories for WebSphere users in the OMVS and sets ownership.
BBOxBRAK	RACF scripts are created and run in this JCL.
BBOxCFS	Sets up and mounts the file system (HFS or zFS).
BBOxHFSA	Populates the created HFS. The job creates intermediate symlinks automatically, based on the options chosen in the WebSphere Customization Toolbox.
BBOWWPFx	The HFSB job is no longer available. Instead, the file system initialization is included in WWPFD.
BBOxPROC	Copies the tailored start procedures to the cataloged procedure library.

16.4 System programmer considerations

This section includes additional hints and tips for system programmers to consider when installing and configuring a WebSphere Application Server for z/OS V8.5 environment.

16.4.1 WebSphere Application Server settings

This section addresses the following settings:

- Intelligent runtime provisioning
- Workload profile setting
- Addressing mode

Intelligent runtime provisioning

The intelligent runtime provisioning function is disabled by default. You might want to enable it in the administrative console to reduce startup time and resource consumption. You can see improvements in the startup time of up to 10–15%. For more information, see 2.2.8, “Intelligent runtime provisioning” on page 49.

Workload profile setting

WebSphere Application Server for z/OS V8.5 allows you to set a new value for the workload profile in the Object Request Broker (ORB) services advanced settings. You can now make a user-defined selection for the number of threads by using the CUSTOM setting.

For more information about this topic, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=urun_rorb_service

Addressing mode

The addressing mode (AMODE) is a JCL parameter introduced with V6.1. It is used in the START command to determine whether the server is started in 64-bit or 31-bit mode.

The AMODE parameter is still supported in V8.5. Do not modify the default value. In the generated procedures during the installation, the default value is 00. This value means that the value for the server bitmode defined in the XML files of the application server decides whether to run in 64-bit or 31-bit mode.

If you start the server with, for example, AMODE=64, and the XML files reflect a 31-bit installation, the server will not start.

Tip: Use the default value for the AMODE parameter (AMODE=00) in the startup JCL for the WebSphere Application Server components. Double-check your automation settings.

16.4.2 Java virtual machine settings

The settings described in this section are JVM or system settings that cannot be directly modified by the WebSphere Application Server V8.5 administrator, or need additional software or hardware prerequisites. However, servers will function based on these underlying settings.

For more information about the topics in this section, see the Diagnostics Guide for the SDKV6 in the Information Center at:

http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp

Shared class cache

This section provides information about shared class cache usage on z/OS. The shared class cache is used to share WebSphere Application Server and user classes between multiple JVMs. JVMs that use the shared class cache start quicker and have lower storage requirements than JVMs that do not. The overall cost of class loading is also reduced when JVMs use the shared class cache.

When a new JVM that shares the class cache is initialized, it uses the preinstalled classes instead of reading them from the file system. A JVM that shares the class cache still owns all the working data (objects and variables) for the applications that run in it. This configuration helps to maintain isolation between the Java applications that are processed in the system.

The first JVM, after an initial program load (IPL) or after the cache is deleted, takes 0–5% longer to fill the cache. The start time of subsequent JVMs decreases by 10–40%, depending on the number of classes that are loaded.

The z/OS implementation links pages in the private area of the address space that uses the cache to the frames of the original location of the cache. Because shared memory is used, the BPXPRMxx parmlib settings affect the cache performance.

Important settings

Consider these factors when using shared class cache in your environment:

- Cache size limits

  The maximum theoretical cache size is 2 GB. The size of cache that you can specify is limited by the amount of physical memory and swap space that is available to the system. The cache for sharing classes is allocated by using the System V IPC Shared memory mechanism. The virtual address space of a process is shared between the shared classes cache and the Java heap. Therefore, if you increase the maximum size of the Java heap, you might reduce the size of the shared classes cache that you can create.

- BPXPRMxx settings for shared memory

  The following settings affect the amount of shared memory pages that are available to the JVM:

  – MAXSHAREPAGES
  – IPCSHMSPAGES
  – IPCSHMMPAGES
  – IPCSHMMSEGS

  Consideration: The JVM uses these memory pages for the shared class cache. If you request large cache sizes, you might have to increase the amount of shared memory pages that are available.

  The shared page size for a z/OS UNIX System Service is fixed at 4 KB for 31-bit and 1 MB for 64-bit platforms. Shared classes try to create a 16 MB cache by default on both 31- and 64-bit platforms. Therefore, set IPCSHMMPAGES greater than 4096 on a 31-bit system.

  If you set a cache size by using `-Xscmx`, the JVM rounds up the value to the nearest megabyte. You must take this setting into consideration when setting IPCSHMMPAGES on your system.

For more information about performance implications and using these parameters, see *z/OS MVS Initialization and Tuning Reference*, SA22-7592, and *zOS UNIX System Services Planning Guide*, GA22-7800.

Persistence for shared class cache

WebSphere Application Server for z/OS V8.5 uses the IBM Java Standard Edition V6. This JVM implementation offers the shared class cache that allows multiple JVMs to access the same classes without loading them multiple times into memory. These include both application and system classes.

The IBM implementation for distributed platforms allows you to write the content to a file system so that it can survive an operating system restart. Platforms that are supported include AIX, Linux, and Windows systems. However, z/OS supports the use of only the non-persistent cache.

Compressed references

The use of compressed references improves the performance of many applications by making object headers and object references smaller. This reduction results in less frequent garbage collection and improved memory cache use. Certain applications might not benefit

from compressed references. Test the performance of your application with and without the option to determine whether it is appropriate.

When using compressed references, the following structures are allocated in the lower area of the address space:

- Classes
- Threads
- Monitors

Because this JVM technique is independent from the Websphere Application Server product, you can activate it only by using the JVM argument, *-Xcompressedrefs* on a JVM level. On the z/OS platform, the activation needs to be run for all components of an application server that have a heap (adjunct, control, and servant region).

As always, when changing JVM settings, restart the server after saving and synchronizing the modifications to activate them.

16.4.3 Basic WLM classifications

The usage of WLM classification for the control and servant region address spaces is a basic z/OS approach. It is part of the installation process of the WebSphere Application Server for z/OS V8.5.

The following considerations apply:

- Assign control regions a service class with a high priority in the system, such as the SYSSTC service class. A high priority is needed because control regions do some of the processing required to receive work into the system. This processing includes managing the HTTP transport handler, classifying the work, and running other housekeeping tasks.
- Do not set the servant classification higher in the service class hierarchy than more important work, such as the control region and CICS or IMS transaction servers. Use a high velocity goal instead.
- Classify enclaves for WebSphere Application Server for z/OS by using the Subsystem CB. The performance goals that you set here depend on your applications and the environment. Therefore, no quantitative recommendation can be made here. However, usually a percentile response time goal is advisable.
- Classify OMVS components of WebSphere Application Server for z/OS. Some OMVS scripts are run during server start. Therefore, if these scripts are not classified in the WLM, the server start time increases.

> **Remember**: A step in the control region start procedure starts the `applyPTF.sh` script by using BPXBATCH. Because the BPXBATCH program is classified according to the OMVS rules, several minutes might pass before this step is completed on a busy system.
>
> You can minimize the impact of the BPXBATCH step by changing the WLM Workload Classification Rules for OMVS work to a higher service objective.

For information about how to set WLM service class classifications, see *System Programmer's Guide to: Workload Manager*, SG24-6472.

For more information, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=trun_wlm_tclass

16.4.4 Address space identifier reuse

Address space identifier (ASID) reuse is an operating system function that was introduced with z/OS V1.9. This function allows the reuse of an address space ID that could not be reused in earlier releases of z/OS. These IDs include ones that are associated with cross-process services such as TCP/IP. Starting with V6.1, WebSphere Application Server for z/OS can use this function, allowing the reuse of the ASID for terminated control regions.

The REUSASID parameter is set to YES automatically for any new servers that are created in WebSphere Application Server for z/OS V8.5.

If the operating system runs with the ASID reuse option enabled, you can run the `updateZOSStartArgs` script in the `profile_root/bin` directory of each profile. Running the script enables the ASID reuse capability for a specific WebSphere Application Server for z/OS profile. The script adds the REUSEASID=YES argument to the servers started from the administrative console only. For servers started by command from a system console, add the argument to enable the function. Ask the system programmer whether the ASID reuse option is used in your installation.

For more information about ASID support, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=txml_configasid

16.4.5 Deprecated features WebSphere Application Server for z/OS

As with every new version, some features are deprecated. For a complete list of deprecated features, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=rmig_depfeat

16.4.6 Jacl stabilized

The Java TCL (Jacl) scripting language is stabilized. Stabilized means that, although no new development will be done for this language, it will coexist with Jython in Websphere Application Server V8.5. Administrative scripts that use Jacl do not need to be migrated to Jython. However, this stabilized status might change in future releases of Websphere Application Server.

16.4.7 Application profiling

With application profiling, you can analyze the application during run time. It graphically provides detailed information about how much each application step uses the processor. This information helps you to identify critical points inside the application. Although it is intended for developers, system programmers should encourage the development team to use profiling.

A profile tool for z/OS is *JinsightLive for z/OS*. With this tool, you can analyze 31- and 64-bit JVMs. To download this tool, go to the JinsightLive for IBM System z page on IBM alphaWorks® at:

`http://www.ibm.com/systems/z/os/zos/features/unix/tools/jinsightlive.html`

Another tool that provides application profiling is the Eclipse Test and Performance Tools Platform (TPTP), which is a project from the Eclipse platform. To download this tool, go to the Eclipse website at:

`http://www.eclipse.org/tptp/`

Consideration: Application profiling usually requires some level of experience with the tools. After you get used to the technique, application profiling is a powerful way of identifying CPU-intensive points in an application. Many of the critical points require only a few changes in the application itself.

As a starting point, ask your local IBM representative for assistance.

16.5 Planning checklist

Consider the following items as you plan for WebSphere Application Server for z/OS V8.5.

- Because the ISPF Customization Dialog has been removed, use the WebSphere Configuration Tools or the line-mode `zpmt.sh` script to create all profiles.
- Make sure that you have a convenient naming convention that can reflect the use, the job manager, and the administrative agent WebSphere Application Server V8.5 components. The z/OS Profile Management Tool uses the recommendations from the z/OS customization worksheet.
- Test the usage of XCF support for the high availability manager.
- Make sure that monitoring is in place.
- Use the IBM Support Assistant with the following plug-ins:
  - Visual Configuration Explorer (VCE)

    A graphical view of your environment to help track configuration changes. This tool is available for no extra fee.
  - Garbage Collection and Memory Visualizer

    To analyze verbose garbage collection information and identify a good heap size.
  - Thread Analyzer

    To analyze Java thread dumps (or Java cores) such as those from WebSphere Application Server.
- Check the amount of real memory provided for the LPAR where WebSphere Application Server for z/OS will be installed.
- Check the usage of Java compressed references, because most of the current applications have no need for heaps larger than 900 MB.
- Check with the application developers whether the application can use the shared class cache.
- Make sure that you selected an effective garbage collection policy and performed a verbose garbage collection analysis to identify and verify the heap size.

16.6 Intelligent Management and WebSphere Batch on z/OS

This section describes the internal differences of the Intelligent Management feature and WebSphere Batch implementation on z/OS, and their specialities in using the platform.

16.6.1 Intelligent Management on z/OS

Intelligent Management is a new robust component of WebSphere Application Server. It includes intelligent routing, health management, application edition management, and performance management capabilities. The new on-demand router server type is available as a Java based HTTP and Stateless SIP proxy. It provides automatic routing without needing to update your configuration. The application placement controller monitors process and node level processor usage to indicate current demand and makes informed decisions on application placements. The Intelligent Management functionality complements and improves the already existing application deployment options. It also provides elasticity of the environment with support for IBM z/VM®. Native monitoring tools are improved with additional metrics and Dojo charting technology. Health management introduces policies able to prevent or mitigate critical situations with automatic preset actions.

For more information about concepts, common features, and cross-platform behavior, see Chapter 5, “Intelligent Management” on page 107.

Restrictions: Federating middleware agents from distributed platforms into a z/OS deployment manager is not supported. SIP is not supported on the z/OS operating system.

16.6.2 WebSphere Batch on z/OS

Batch has been an important part of the mainframe processing for decades. But with pressure for global 24 x 7 support and increased online transaction processing (OLTP) workload, the batch window is shrinking. Customers need to run both side-by-side.

WebSphere Batch for z/OS is a mature component that delivers these capabilities. It provides a comprehensive execution environment for Java batch processing and unified batch architecture across the enterprise. It answers the need for batch modernization and parallelization by running the Java batch inside WebSphere Application Server for z/OS. It also alleviates the problem of processor usage by creating and deleting the JVM for every job run. Among other advantages on z/OS is the collocation and close proximity to the back-end data.

For more information about developing WebSphere Batch applications, see:

https://www.ibm.com/developerworks/wikis/display/xdcomputegrid/Home

WebSphere Batch is available on all platforms that support WebSphere Application Server V8.5. It can be used with stand-alone installations. However, it is typically run in a high-availability cell topology with two nodes (presumably across two LPARs), and two clusters: A cluster for the job scheduler servers and one for the endpoint servers.

COBOL support

COBOL has been a part of batch processing since the early days of computers and there is significant investment in mission-critical COBOL assets, especially on mainframes. The new COBOL container is a function that allows COBOL modules to be loaded into the WebSphere Application Server for z/OS address space and started directly. It provides the means of direct

integration of COBOL resources into WebSphere Java processing. The container itself is implemented as a handful of dynamic link libraries (DLLs) and Java archive (JAR) files, as shown in Figure 16-15.

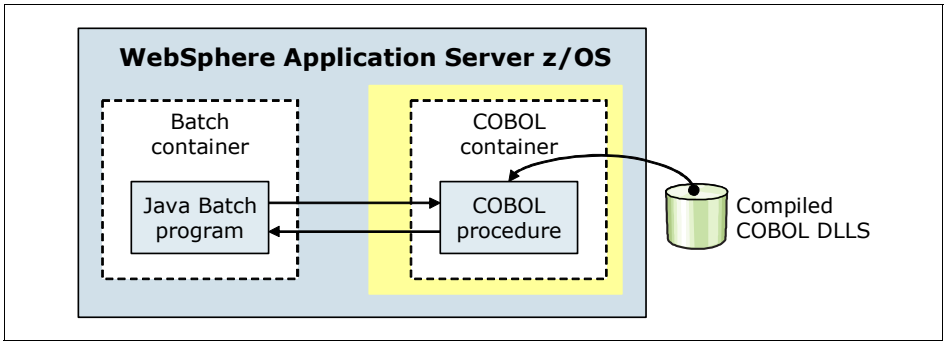


Figure 16-15 WebSphere Batch COBOL container

The container itself can be created and deleted multiple times within the lifecycle of a server. Each container is created with IBM Language Environment® enclave separate from that of a server. The container is assured of a clean Language Environment each time it is created. Java programs can pass parameters into COBOL and retrieve the results. A utility is supplied that creates the Java call stubs and data bindings based on the data and linkage definitions in the COBOL source. Further, JDBC Type 2 connections created by the Java program can be shared with the COBOL program under the same transactional context. The COBOL container supports a wide variety of data types beyond integers, including primitive and national data types. It also supports nested COPYBOOKs.

Java programs that intend to start COBOL programs complete the following steps:

1. Create the container. This phase creates the separate Language Environment environment within the address space of the server.
2. Create the procedure or procedures and initialize any parameters. This phase loads the COBOL module and prepares any data values to be passed.
3. Start the procedures. This phase runs the named COBOL procedures.
4. Retrieve the results. This phase processes return values.

A utility is supplied with the COBOL Container to assist in the generation of call stubs and data bindings. The utility is called the *Call Stub Generator*. It takes as input the COBOL source. It produces as output the generated call stub and any data bindings as seen in the LINKAGE section of the COBOL.

Requirement: COBOL procedures started using the COBOL container must be compiled as a *recursive DLL*. The DLL can be maintained in the hierarchical file system and referenced with a LIBPATH update. It can also be maintained in a PDSE and referenced with STEPLIB.

Integration with schedulers

The native WSGRID connector provides efficient integration with Tivoli Workload Scheduler for z/OS and other z/OS schedulers. It makes it possible for logs and data to be returned from the execution. The connector can be used in conjunction with the service integration bus and the WebSphere MQ queuing network. This configuration provides a means of scheduling work from the enterprise schedulers to the distributed WebSphere Application Server V8.5 running on other platforms.

For more information about common features and cross-platform behavior, see Chapter 6, “WebSphere Batch” on page 137.

16.7 The Liberty profile on z/OS

This section focuses on the exclusive features and architecture of the Liberty profile that run on System z.

16.7.1 Architecture of Liberty profile on z/OS

The Liberty profile on z/OS has some distinct differences in its runtime architecture. It is closely aligned with the platform, and can provide active z/OS exploitation. Functions and behavior of the Liberty profile are consistent across platforms. z/OS specific extensions are modeled as independently enabled feature sets, and therefore come into play only when configured.

The Liberty profile on z/OS includes the following process types:

- Angel process
- Server process

These runtime processes can be started in the background or foreground, or can be started as jobs or tasks controlled by MVS commands.

Angel process

The angel process runs in an authorized key. It provides facilities to the server process to load and access system services in a way that protects the integrity of the operating system. There is no code level dependency between the angel and the server processes. The angel does not need any configuration and exists independently of the server, although SAF profiles must be configured. All the Liberty profile servers that run on a z/OS image can share a single angel, regardless of the level of code that the servers are running. If no z/OS system authorized services are enabled for any server on a system, the angel does not need to be active. The angel process is not required for the command processing services because it uses unauthorized z/OS services for command support. If the server is configured to attempt to use authorized services but either of these statements are true, the authorized service is not available on that server:

- The angel is not available
- The effective owner of the process is not authorized to use the angel

In some cases, an unauthorized service can be used instead, but the processing path lengthens.

Server process

The server process is similar to the server on other operating systems. It is a JVM running the Liberty code in 64-bit mode, and provides single compacted environment for your applications. The server process can use JES output as STDOUT and STDERR for logging convenience. This means that Liberty servers can run with default behavior without requiring any configuration at all. Configuration values need to be supplied only when the default behavior is to be changed. Together with fast startup times of the server and applications, and a set of eclipse-based tools and portability, it is a perfect fit for your development or production environment.

Figure 16-16 shows a sample z/OS topology.

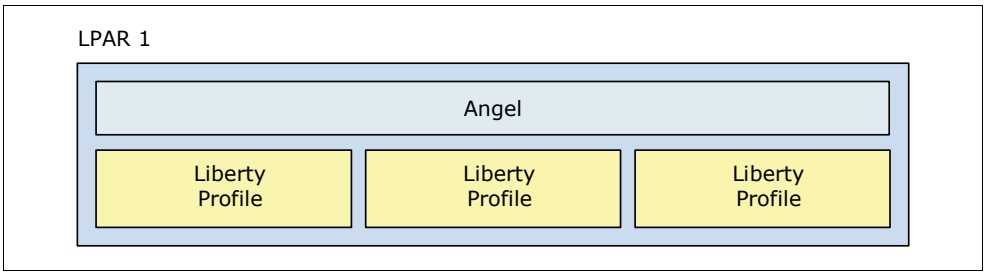


Figure 16-16 Liberty profile processes

For more information about Liberty profile common features and cross-platform behavior, see Chapter 4, “An overview of the Liberty profile” on page 91.

Requirement: To run the Liberty profile on z/OS, you must be using z/OS Version 1.11 or later. Minimum supported level of IBM JDK is Java 6 SR 1 64-bit.

16.7.2 Unique features of the Liberty profile on z/OS

The Liberty profile on z/OS allows for an active exploitation of the existing System z specific services. The profile thus provides an advantage in performance and quality of service (QoS) of the Liberty profile on z/OS.

Currently, the following optional features are available on z/OS only:

- zosSecurity-1.0

  The SAF registry holds information needed to run security-related functions. These functions include authenticating users and retrieving information about users, groups, or groups associated with users. It comes configured using the default configuration values. By default, the SAF registry uses unauthorized UNIX System Services services unless configured to use authorized SAFCRED resources. SAF-based key rings for SSL certificates are supported.

- zosTransaction-1.0

  This feature enables the application server to synchronize and appropriately manage transactional activity between the following applications:

  – Resource Recovery Services (RRS)
  – The application server's transaction manager
  – The resource manager

  It also allows for use of DB2 for z/OS JDBDC type 2 Native-API driver, which can speed up back-end database interactions. Transaction feature requires an angel process and a functional RRS subsystem to run.

- zosWlm-1.0

  This feature provides access to z/OS native WLM services. It allows classification of HTTP requests based on host, port, method, and resource in the `server.xml`. This classification includes transaction class that is mapped to service and report class by WLM. A response enclave is created and joined for each classified request. A collection name can be associated with classifying work requests by use of **`zosWorkloadManager`** configuration element.

For a complete list of features supported by Liberty profile, go to the following website:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=twlp_setup_feat

For more information about development-related resources with news and samples for download, visit:

http://www.wasdev.net

16.8 Resources

This section includes links and references to additional material to provide deeper insight on the workings of z/OS with the WebSphere Application Server for z/OS.

For information about planning and system considerations required to build a heterogeneous cell, see the *WebSphere for z/OS—Heterogeneous Cells* white paper at:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100644

This paper focuses on WebSphere Application Server V6.1, but the basic concepts are still valid for V8.5.

For a comprehensive overview of IBM Installation Manager for z/OS and its use with WebSphere Application Server for z/OS, see:

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102014

Benefits of collocation of the application layer with the data layer on z/OS are addressed in the white paper *The Value of Co-Location*, Document ID WP101476:

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101476

For more information about WebSphere configuration tools, including the z/OS Profile Management Tool and the z/OS Migration Management Tool, see the following websites:

- WebSphere Application Server V8.5 Information Center at:

  http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tins_installation_wct_gui

- *WebSphere Application Server for z/OS V7.0 - Introducing the WCT for z/OS*, Document ID PRS3357

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS3357

- *WebSphere for z/OS Version 7 - Configuration Planning Spreadsheet*, Document ID PRS3341

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS3341

- *Introducing the IBM Support Assistant for WebSphere on z/OS*, Document ID WP101575

  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101575

For deeper insight into the Java options and functions used by WebSphere Application Server V8.5, see the following websites:

- IBM Java 6.0 Diagnostics Guide

  http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp

- Java technology, IBM style: Garbage collection policies, Part 1

  http://www.ibm.com/developerworks/java/library/j-ibmjava2/index.html

- Java technology, IBM style: Garbage collection policies, Part 2

  http://www.ibm.com/developerworks/java/library/j-ibmjava3/

For a comprehensive look into application development of Java applications, see the following IBM Redbooks publications:

- *Java Stand-alone Applications on z/OS, Volume I*, SG24-7177
- *Java Stand-alone Applications on z/OS Volume II*, SG24-7291

Various tools are available to ease the daily life of developers and system programmers. The following tools are available at no charge:

- IBM Support Assistant

  This tool, together with some plug-ins, provides an easy way to check for configuration changes and a central repository for configuration values. In addition, you can use it to create graphical overviews of your environment. To download this tool, go to the IBM Support Portal at:

  http://www.ibm.com/software/awdtools/isa/support/

- JinsightLive for IBM System z

  To download this application profiling tool, go to:

  http://www.ibm.com/systems/z/os/zos/features/unix/tools/jinsightlive.html

- Eclipse Test and Performance Tools Platform (TPTP)

  To download this profiling tool plug-in for the well-known Eclipse project, go to the Eclipse website at:

  http://www.eclipse.org/tptp/

17

Migration

This chapter addresses migration considerations for moving to WebSphere Application Server V8.5. This chapter includes the following sections:

- Migration features in WebSphere Application Server V8.5
- Migration overview
- Migration plan
- Application development migration considerations
- Infrastructure migration considerations
- Migration considerations for WebSphere Application Server for z/OS

17.1 Migration features in WebSphere Application Server V8.5

WebSphere Application Server V8.5 provides features that support migration from older versions. This section highlights these features.

17.1.1 Configuration Migration Management Tool

The Eclipse-based graphical wizard is called the *Configuration Migration Management Tool*. This tool supports the migration of all management profiles of WebSphere Application Server, including admin agent and job manager. It also provides the option to run the generated Migration jobs as 64 bit. The tool highlights the potential changes due to the migration. It can also generate the commands that are run by the graphical wizard to create migration scripts.

17.1.2 Cross platform migrations

With WebSphere Application Server V8.5, you can migrate a node from one system to another, even if they have different operating systems (except for IBM i and z/OS). The **createRemoteMigrJar** tool creates a compressed file from the WebSphere V8.5 binary files. These files are able to run the migration backup command in a system that does not have WebSphere Application Server V8.5 installed.

17.1.3 Enhanced z/OS Migration Management Tool

The Websphere Application Server V8.5 z/OS Migration Management Tool supports the migration of all management profiles of WebSphere Application Server, including admin agent and job manager. It also supports 64-bit migration on z/OS.

17.2 Migration overview

Migration is the action of moving from an existing release to a newer release. A WebSphere Application Server migration is more than just applying a new product version. It is a project.

A WebSphere Application Server migration impacts the following components of your infrastructure:

- Applications
- Middleware
- Operating systems

For information about the removed, deprecated, and new features of WebSphere Application Server V8.5, see the following websites:

- http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=%2Fcom.ibm.websphere.base.doc%2Fae%2Fwelc6topnew.html
- http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=%2Fcom.ibm.websphere.base.doc%2Fae%2Frmig_deprecationlist.html

Reviewing this information can provide a better understanding of the areas that are impacted by migration.

17.3 Migration plan

You must create a migration plan to perform a migration from your existing environment to the new version of WebSphere Application Server. This plan covers the following core steps. Keep in mind that each migration is unique and might need to be adjusted.

1. Project assessment

   Create a migration team and review all aspects of the migration, such as education, hardware, application, testing, and risk factors.

2. Project planning

   Define a complete migration plan, from day one to the actual production migration, based on the assessments of step 1.

3. Skill development

   Plan for an education period to address new product features, tools, and the development standards in WebSphere Application Server V8.5.

4. Setup of development environment, application migration, unit test

   Test your applications in the new environment for compatibility and possible code modification.

5. Setup, migration, and test of additional runtime environments

   In parallel with the application migration process, iteratively migrate all of your other environments except the production environment, and create a migration path.

6. Testing

   Plan a functional, technical, and performance test campaign to validate your migration.

7. Production environment migration

   Before migrating, prepare a rollback plan. Follow your migration procedure to update your production environment.

8. Lessons learned session

   Following the migration, review the project outcome and processes that were used with the entire migration team to improve the migration process.

Figure 17-1 illustrates the steps that you might take in performing a migration.

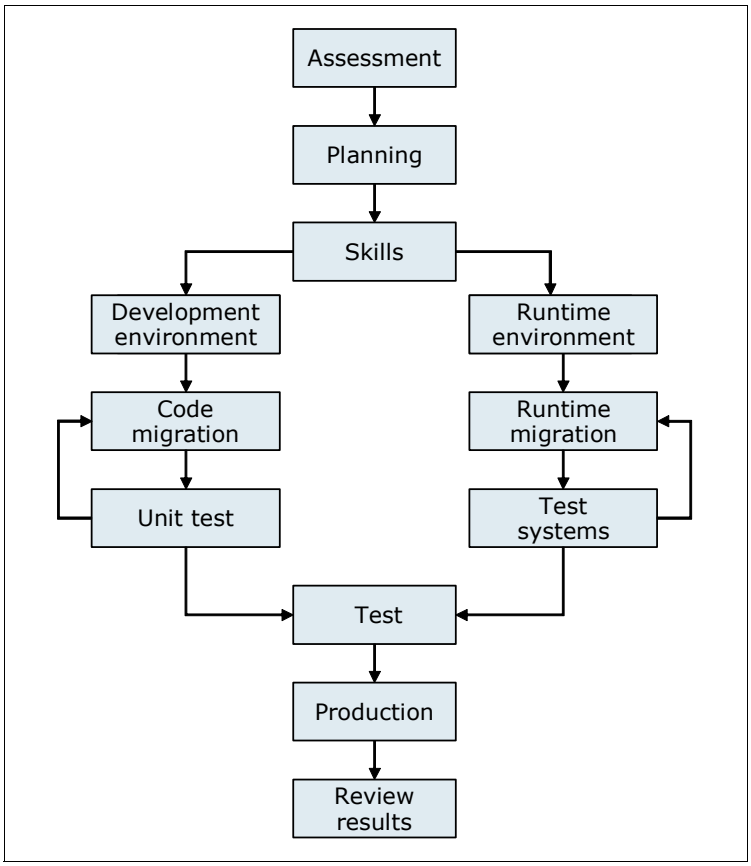


Figure 17-1 Migration path

Additional resource: For more information about WebSphere Application Server migration, see Knowledge Collection: Migration planning for WebSphere Application Server at:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27008724

17.4 Application development migration considerations

This section provides a general overview of considerations to make when migrating applications between WebSphere Application Server versions. WebSphere Application Server V8.5 supports Java Platform, Enterprise Edition 6 (Java EE 6). Consider the following points:

- Although newer J2EE versions support older versions, some minor exceptions might exist.
- Identify the deprecated application programming interfaces (APIs) and determine whether any of these APIs are used in your existing applications.
- Understand the new WebSphere Application Server V8.5 features.

For more information about deprecated APIs, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=cmig_apispec

See also the deprecated API list for the Java platform at:

http://download.oracle.com/javase/6/docs/api/deprecated-list.html

or

http://download.oracle.com/javase/7/docs/api/deprecated-list.html

For more information about how to migrate specific application components such as web services, EJB, OSGi, and asynchronous beans, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=welcome_migrating

IBM provides a separate tool called *WebSphere Application Migration Tool* based on Rational Software. With this tool, you can quickly analyze your applications and highlight the parts that are not compatible, such as deprecated APIs.

17.5 Infrastructure migration considerations

This section addresses topics to consider when migrating from an existing environment to WebSphere Application Server V8.5.

17.5.1 Coexistence

WebSphere Application Server V8.5 can be installed and configured to coexist with other WebSphere Application Server V8, V7, and V6.1 installations on the same system simultaneously without any conflict.

Consider the following factors before starting such a migration:

- The hardware and software of the system must be supported by all versions of Websphere Application Server that you plan to coexist.
- Each installation of WebSphere Application Server requires additional system resources.
- Plan for unique ports for every installed version of WebSphere Application Server.

17.5.2 Interoperability

WebSphere Application Server V8.5 is generally interoperable with WebSphere Application Server V8, V7, and V6.1. This interoperability means that different versions of WebSphere Application Server can exchange data and communicate.

Requirements exist for some functions that depend on the WebSphere version. For more information, see the WebSphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=welcome_migrating

17.5.3 Mixed-version-cell support

To ease the incremental upgrade of your environment, WebSphere Application Server V8.5 supports mixed-cells with nodes from V8, V7, and V6.1. A cell can contain nodes from different versions of WebSphere Application Server and different platforms. The version of your deployment manager must be at the highest level you use in your cell.

Although running in a mixed-cell configuration is supported, this situation is to be considered transitional and for a limited time. In the end, all your nodes should be at the same level for best results.

17.5.4 Configuration Migration Tools

WebSphere Application Server V8.5 provides Configuration Migration Tools to perform a migration.

With Configuration Migration Tools, you can perform these tasks:

- Migrate configurations, including the topology, customizations, and applications, while keeping your old environment running. The tools support the migration of V6.1, V7, and V8 security features, which include enhanced Secure Sockets Layer (SSL), security audit, Kerberos, and multidomain security.
- Migrate the applications from the old version to the new version without changing them. The tools support the migration of the business-level applications.
- Migrate one profile at a time because the process is iterative.
- Perform cross-platform migration (for distributed platforms only).
- Migrate V7 and V8 job manager and administrative agent profiles.

Figure 17-2 illustrates the migration process.

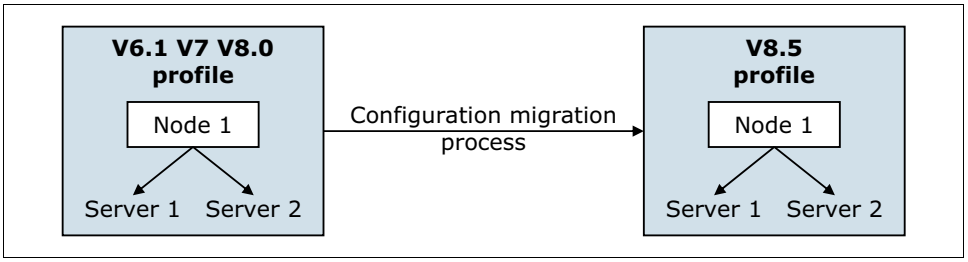


Figure 17-2 Migration process

The Configuration Migration Tools are available on the following platforms with the indicated features:

- Distributed
  - Configuration Migration Management Tool

    This tool provides a graphical interface used to run all the migration steps (Figure 17-3). The tool is based on the migration commands listed after the figure.

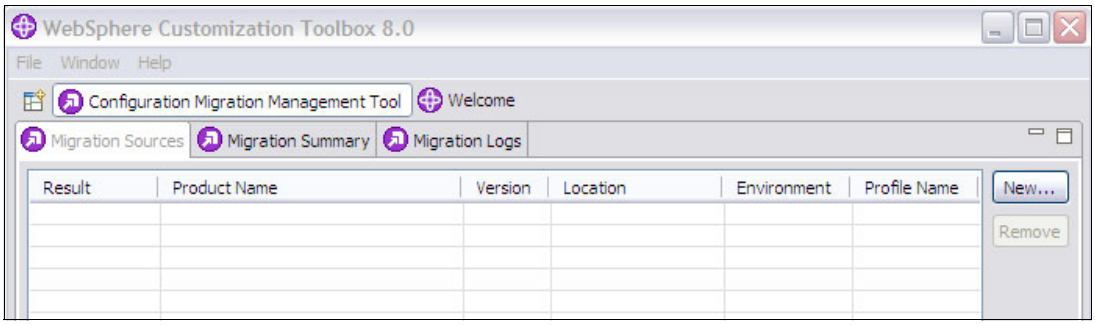


Figure 17-3 Configuration Migration Management Tool

  - The createRemoteMigrJar tool

    This tool creates a compressed file from the WebSphere Application Server V8.5 binary files. The compressed file contains the necessary files for running the migration backup command in a remote system that does not have WebSphere Application Server V8.5 installed. You use this tool when you want to perform a migration from one system to another. The compressed file that is created contains specific code, which makes it operating system dependent.

  - Migration commands
    - The **`WASPreUpgrade`** command saves the configuration of the old installed version into a migration-specific backup directory.
    - The **`WASPostUpgrade`** command applies the old version of the configuration to the new version by copying, replacing, merging, and deleting profile data.
    - The **`clientUpgrade`** command migrates previous versions of the client to the new version.

- IBM i

  Migration commands:

  - **`WASPreUpgrade`**
  - **`WASPostUpgrade`**
  - **`clientUpgrade`**

  For more information about these commands, see the Websphere Application Server V8.5 Information Center at:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp`

- z/OS

  z/OS Migration Management Tool

  For more information, see the Websphere Application Server V8.5 Information Center at:

  `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tmig_admin`

17.5.5 Properties files

The **wsadmin** tool provides a set of commands that enable you to export portions of the application server profile into properties files. You also can modify your cell by importing these properties files, enabling you to transfer parts of a cell configuration to a newly created cell.

For more information about the properties file based configuration, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=txml_property_configuration

17.5.6 Product configuration migration scenarios

This section describes these product configuration migration scenarios:

- Manual
- Stand-alone environment with the Configuration Migration Tools
- Multinode environment with all-node upgrade and Configuration Migration Tools
- Multinode environment migration with mixed-node and the Configuration Migration Tools
- Fine-grained approach for a stand-alone environment
- Administrative agent environment with the Configuration Migration Tools
- Job manager environment with the Configuration Migration Tools
- Cross-platform migration

For more details and complete migration scenarios, see the following resources:

- The Websphere Application Server V8.5 Information Center at:

  http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=welcome_migrating

- *WebSphere Application Server V7 Migration Guide*, REDP-4635

  The scenarios in this paper are available only for distributed platforms and IBM i. For z/OS migration, see 17.6, “Migration considerations for WebSphere Application Server for z/OS” on page 560.

Considerations: Before migrating, consider the following tips:

- Migrate one profile at a time.
- Migrate from a clean and functional profile to a clean profile.
- Back up all data before migrating.
- Always migrate the highest level profile first.
- The job manager can manage only servers at the same release or earlier.
- The administrative agent can register only Base application servers at the same release level and on the same system.
- The deployment manager can manage nodes only at the same release or earlier.

Manual

With the manual migration scenario, you start with a new WebSphere Application Server V8.5 environment and import all configurations and applications. Ideally, use scripts to perform this import. Because of the risk of human error, re-creating it manually by using the administrative console can be risky. Remember that you must migrate all of your environments. The manual approach provides the following advantages and disadvantages:

- Advantages
  - All of the migration tasks can be performed independently from the running environment.
  - The granularity of the migration is under the control of the project team.
  - All of the scripts are yours. You have full control over the migration and do not need to depend on WebSphere tools.
  - You can reuse the scripts for a disaster recovery.
- Disadvantages
  - Creation and continuous maintenance of these scripts can require considerable effort and be expensive. These scripts must be valid for the new WebSphere version.
  - Every change in the environment must be scripted.
  - It is easy to forget some configurations.

You can migrate the entire topology with the manual approach.

Stand-alone environment with the Configuration Migration Tools

You can migrate your complete stand-alone environment at the same time by using the Configuration Migration Tools provided by WebSphere Application Server.

Use the following procedure to perform this migration.

1. Back up the previous version of the profile with the migration tools.
2. Install WebSphere Application Server V8.5, and create a profile.
3. Import the profile configuration with the migration tools.

The schema in Figure 17-4 illustrates this migration.

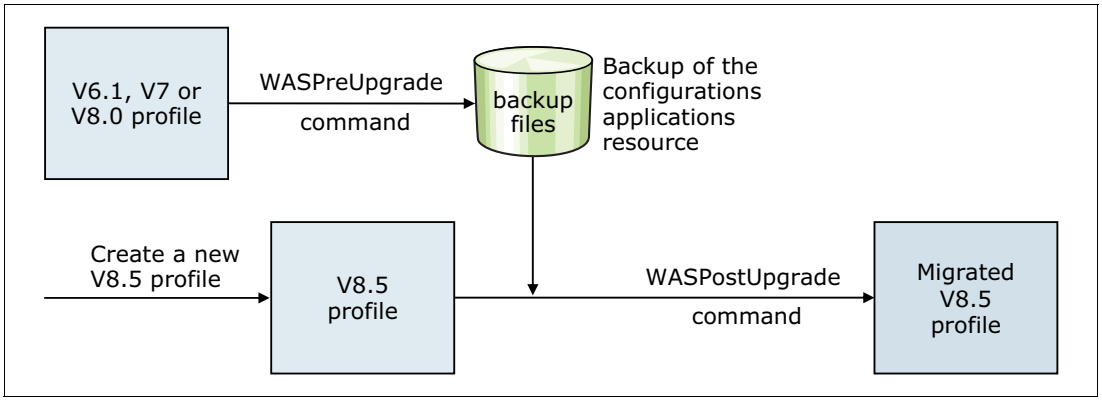


Figure 17-4 Automated migration approach

This approach provides the following advantages and considerations:

- Advantages
  - There is no need for self-written scripts. All migration is done by using the WebSphere Application Server migration tools.
  - All of the information in the current configuration is imported to WebSphere Application Server V8.5.
- Considerations
  - All applications that will be migrated must be ready at the same time.
  - This approach works only if you keep the same topology.

Multinode environment with all-node upgrade and Configuration Migration Tools

You can migrate your complete environment at the same time by using the migration tools provided by WebSphere Application Server. This approach is useful if you are not redesigning your environment. If you do not want to migrate your entire environment at the same time, see “Multinode environment migration with mixed-node and the Configuration Migration Tools” on page 556.

To perform this migration, complete these steps:

1. Back up the previous version of the deployment manager (dmgr) profile with the migration tools.
2. Install WebSphere Application Server V8.5, and create a deployment manager profile.
3. Import the deployment manager configuration with the migration tools.

   When the configuration is imported, you are now in a mixed-cell environment. The deployment manager profile is in V8.5, and the nodes are in the older version.
4. Finish the procedure by migrating all the nodes, one by one, using the same migration tools.

This approach provides the following advantages and considerations:

- Advantages
  - There is no need for self-written scripts. All migration is done by using the WebSphere Application Server migration tools.
  - All of the information in the current configuration is imported to WebSphere Application Server V8.5.
- Considerations
  - All applications that are being migrated must be ready at the same time.
  - This approach works only if you keep the same topology.

Multinode environment migration with mixed-node and the Configuration Migration Tools

You can perform node-by-node migration of your environment by using the migration tools provided by WebSphere Application Server. This approach is useful if you are not redesigning your environment. In this approach, you do not need to migrate all of your nodes at the same time.

To perform this migration, complete these steps:

1. Back up the previous version of the deployment manager (dmgr) profile with the migration tools.
2. Install WebSphere Application Server V8.5, and create a deployment manager profile.
3. Import the deployment manager configuration with the migration tools.

   When imported, you are now in a mixed-cell environment with the deployment manager profile in V8.5, and the nodes in the older version.
4. Decide whether to migrate your nodes independently of one another. All nodes must be migrated as a part of the project. Running in a mixed-cell configuration is considered a transitional state.

This approach provides the following advantages and considerations:

- Advantages
  - There is no need for self-written scripts. All migration is done by using the WebSphere Application Server migration tools.
  - This approach is flexible. Therefore, you can migrate your nodes iteratively without any time consideration.
  - All of the information of the current configuration is imported to WebSphere Application Server V8.5.
- Consideration
  - This approach works only if you keep the same topology.

Fine-grained approach for a stand-alone environment

With the fine-grained approach, you can migrate portions of the configuration by using the Configuration Migration Tools and the properties files commands (Figure 17-5 on page 558).

To perform this migration, complete these steps:

1. Install WebSphere Application Server V8.5, and create a temporary profile.
2. Back up the previous version of the profile with the migration tools.
3. Import the configuration with the migration tools into a temporary profile. You do not need any applications in the temporary profiles. Nevertheless, you must rebuild your applications using the WebSphere Application Server V8.5 classes to be able to install them in final profiles. An import command option is available to specify that applications will be built without installing them in the temporary profiles.
4. Create the final profile.
5. Using the **`extract properties files`** command, extract the temporary profile configuration.
6. Using the **`apply properties files`** command, import the temporary profile configuration into the final profile.
7. Install the applications created in step 3 into the final profile.

Figure 17-5 illustrates the flow of the fine-grained migration approach.

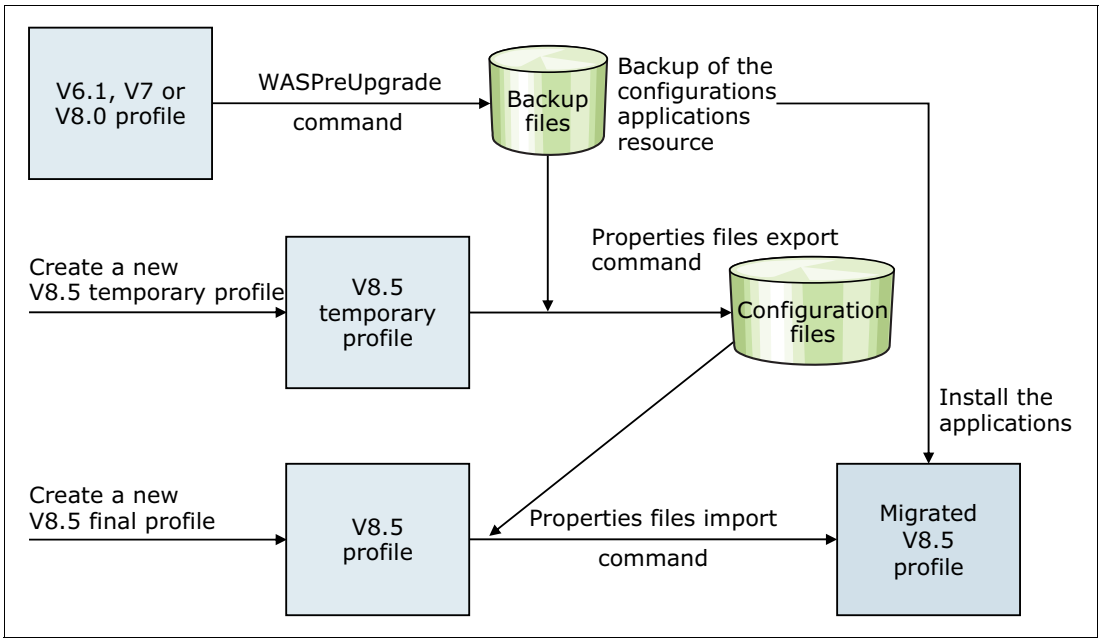


Figure 17-5 Fine-grained migration approach

The fine-grained approach has the following advantages and considerations:

- Advantages
  - There is no need for self-written scripts. All migration is done by using the WebSphere Application Server migration tools and properties file commands.
  - You can choose which information from the current configuration to import into WebSphere Application Server V8.5.
- Consideration
  - The migration requires considerable preparation.

You can also perform this migration approach on a multinode federated environment.

Administrative agent environment with the Configuration Migration Tools

To migrate an administrative agent environment, complete these steps:

1. Verify that no jobs are currently running, and back up the previous version of the administrative agent profile with the migration tools.
2. Install WebSphere Application Server V8.5, and create an Administrative Agent profile.
3. Import the administrative agent configuration with the migration tools.
4. After the configuration is imported and the new administrative agent is running, migrate all the registered base application servers one by one. This approach is explained in “Stand-alone environment with the Configuration Migration Tools” on page 555. You must use specific parameters with the **`WASPreUpgrade`** and **`WASPostUpgrade`** commands.

Job manager environment with the Configuration Migration Tools

To migrate a job manager environment, complete these steps:

1. Back up the previous version of the job manager profile by using the migration tools.
2. Stop the job manager.
3. Install WebSphere Application Server V8.5, and create a job manager profile.
4. Import the job manager configuration by using the migration tools.
5. After the configuration is imported and the new job manager is running, migrate all the registered servers one by one. This process is explained in “Multinode environment with all-node upgrade and Configuration Migration Tools” on page 556. Also, see “Job manager environment with the Configuration Migration Tools” on page 559.

Cross-platform migration

In this approach, you migrate from a stand-alone WebSphere Application Server V6.1, V7, or V8 instance installed on Linux to WebSphere Application Server V8.5 installed on Windows.

To perform this migration, complete these steps:

1. Install WebSphere Application Server V8.5 on the Linux system to generate the compressed file that contains the migration tools.
2. Extract the `compressed` file in the Linux system where WebSphere Application Server V6.1, V7, or V8 is installed.
3. Back up the previous version of the profile by using the migration tools provided in the `compressed` file.
4. Install WebSphere Application Server V8.5 on the Windows system, and create a profile.
5. Import the profile configuration by using the migration tools.

Figure 17-6 illustrates the flow of a cross-platform migration.

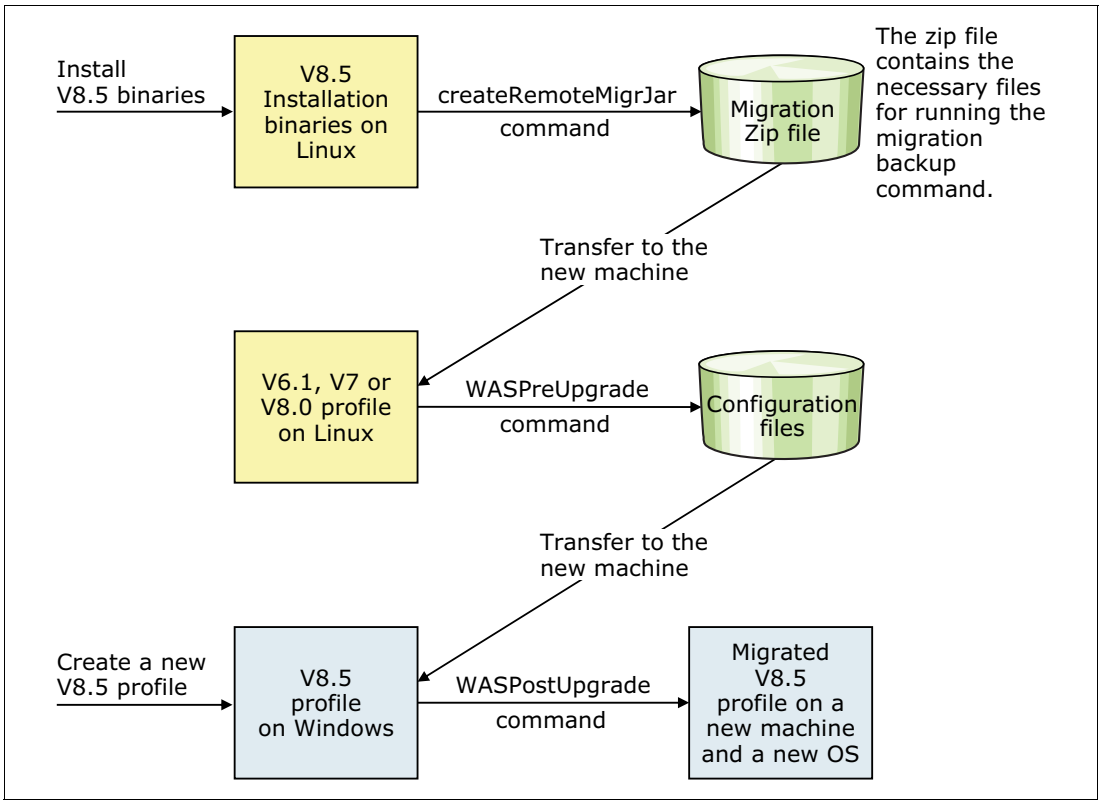


Figure 17-6 Cross-platform migration

17.5.7 Scripts migration

Since the release of WebSphere Application Server V5.1, two scripting languages for WebSphere Application Server are available: Java TCL (Jacl) and Jython.

Jacl is declared *stabilized* since WebSphere Application Server V7, meaning that it will not be removed but there will be no further development for it. Therefore, do not migrate your existing Jacl scripts to Jython. Instead, create your scripts by using Jython.

17.6 Migration considerations for WebSphere Application Server for z/OS

This section concentrates on the topics that you need to consider when migrating an existing WebSphere Application Server for z/OS to V8.5.

17.6.1 Migration and coexistence

Before attempting a migration, you must meet coexistence and prerequisite conditions. The earliest release level of Websphere Application Server that can be directly migrated to Websphere Application Server V8.5 is V6.1. Prior releases must be migrated by using a two-step migration. The first step is to migrate to a version that is supported by the migration tools. Then, in the second step, you migrate to V8.5.

Table 17-1 shows the minimum requirements for the supported releases.

Table 17-1 WebSphere Application Server for z/OS releases for direct migration

Current release	Target release	Minimum level
V6.1	V8.5	V6.1.0
V7.0	V8.5	V7.0.0
V8.0	V8.5	V8.0.0

Keep in mind that the deployment manager must always be at the latest version level. For example, when migrating to V8.5, the deployment manager must be at V8.5. With mixed versions in a cell, you can minimize application downtime during migration because you can migrate one node at a time. If you have applications that run in a clustered environment, those applications can typically continue to run while the migration of one node takes place.

Requirement: You must migrate the job manager to WebSphere Application Server V8.5 before migrating deployment managers or administrative agents that have servers registered to it.

17.6.2 General considerations

Before going into the migration process in more detail, keep in mind the following considerations when performing a migration:

- Use the same procedure names.
  - Before updating the StartedTasks procedures for V8.5, save your current procedures in case you need to fall back to the previous level.
  - If you choose to use different procedure names, update the RACF STARTED class profiles. You can find sample Resource Access Control Facility (RACF) commands to accomplish this task in the migration instructions that are provided.
- Automation changes might also be required when changing procedure names.
- Use a separate file system (HFS or ZFS) for each V8.5 node. This configuration might require new procedure names if you used a shared file system in previous versions.
- Review the guidance for *migrating, coexisting, and interoperating* in the Websphere Application Server V8.5 Information Center at:

  `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=migration_concepts`
- *Premigration considerations* are also an important point to review. For more information, see:

  `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=cmig_pre`

17.6.3 Overview of the migration process

The product code of WebSphere Application Server for z/OS V8.5 is brought into the system by using System Modification Program/Extended (SMP/E) or in an IBM Installation Manager repository format. The code is installed by IBM Installation Manager for z/OS. The migration is then performed by using a three-step approach:

Remember: The migration is always performed on a node basis. In the Network Deployment configuration, you must always start with the deployment manager node.

1. Back up the old environment to have a fallback option.
2. Create and transfer the job control language (JCL) jobs needed during the actual migration (CNTL and DATA data sets).
3. Run the JCL jobs to perform the migration.

To create the JCL, use the z/OS Migration Management Tool or the *zmmt.sh* script. Both techniques are addressed in the following sections. Other migration actions might be in place depending additional products installed in the environment.

17.6.4 z/OS Migration Management Tool

This section describes the z/OS Migration Management Tool that is used during the migration process on z/OS.

Overview

The z/OS Migration Management Tool is an Eclipse-based application that is available in WebSphere Customization Toolbox V8.5. This tool is used to create the JCL jobs for the migration. It uses *Migration Definitions*, a construct that contains all the data that is necessary to migrate a WebSphere Application Server for z/OS node from V6.1 (and later) to V8.5. It contains the Migration Instructions that are personalized for each Migration Definition. It can be used to transfer the JCL to the z/OS target system, if that system has a File Transfer Protocol (FTP) server.

z/OS Migration Management Tool is intended for use by system programmers or administrators who are familiar with the z/OS target system on which the migrated V8.5 nodes will run.

Figure 17-7 shows a high-level overview of the migration process using z/OS Migration Management Tool.

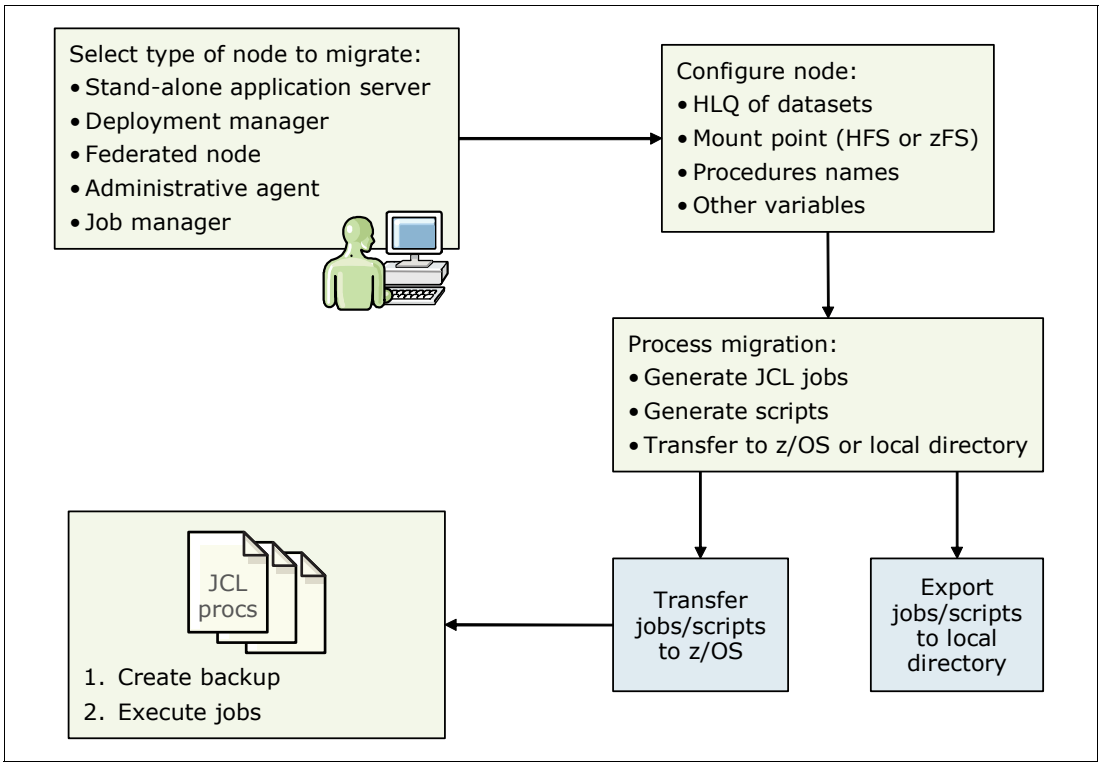


Figure 17-7 Migration process with z/OS Migration Management Tool

Installing the z/OS Migration Management Toolbox

The z/OS Migration Management Tool is available for Windows and Linux technology-based workstations. It is included in the Supplementary Material package. You can also download the WebSphere Customization Toolbox package from the IBM Installation Manager. For information about how to install Installation Manager, see 9.6, “IBM Installation Manager” on page 242. The WebSphere Customization Toolbox includes the z/OS Migration Management Tool, the z/OS Profile Management Tool, and the Web Server Plug-in Configuration Tool.

If WebSphere Customization Toolbox is not installed on your system, perform these steps:

1. Update the IBM Installation Manager with your preferred repository location.
2. Go to the main window of Installation Manager, and click **Install**. For the tool to access the IBM online repository, a user ID and password are required.
3. When you see the packages that can be installed, select **WebSphere Customization Toolbox** and click **Next**.
4. Read and accept the license agreement.
5. Identify the directory on your local file system in which you want to install the packages.
6. Select the tools from the WebSphere Customization Toolbox for installation as shown in Figure 17-8 on page 564.
7. After the installation process, you can see the packages that were installed.

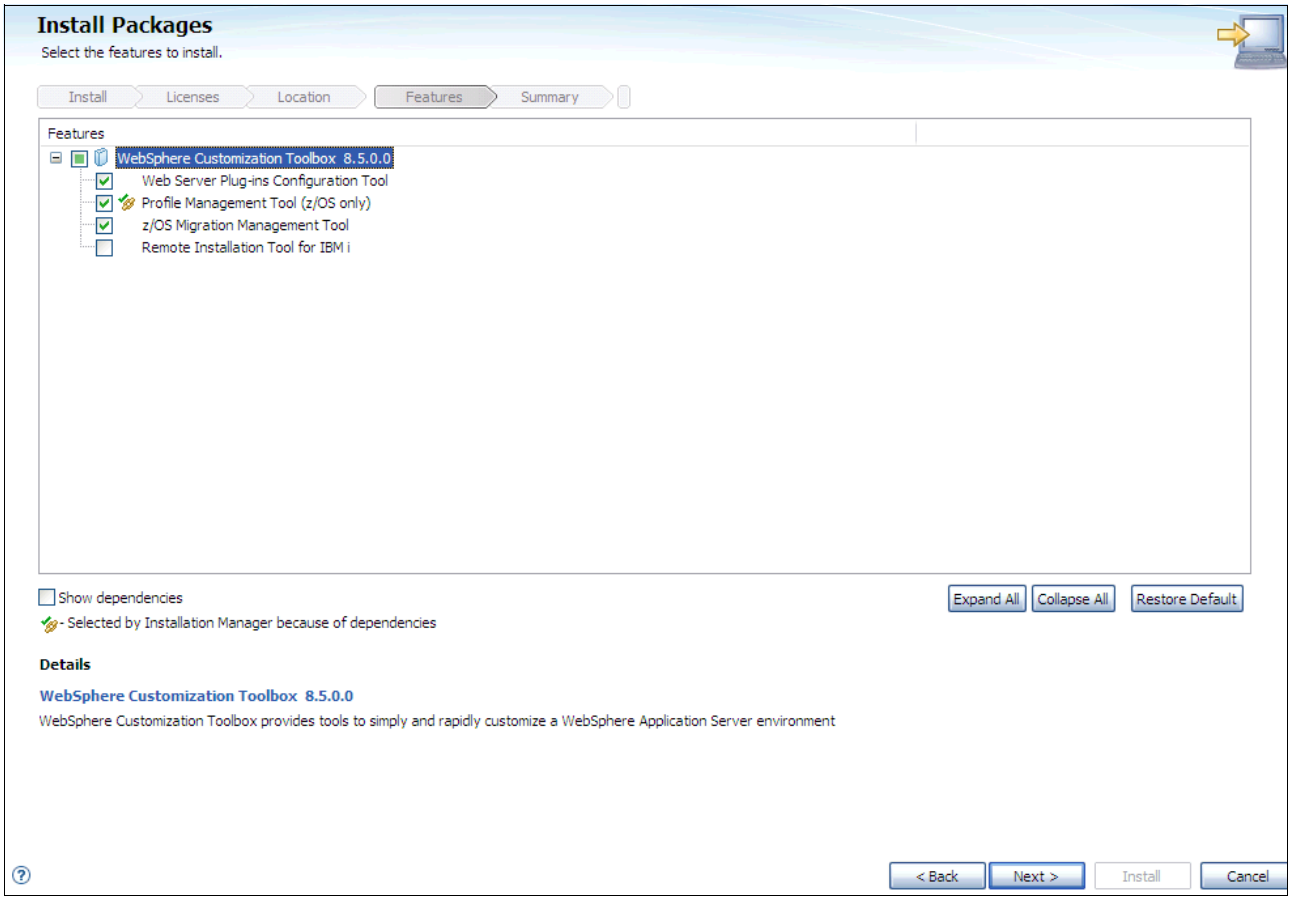


Figure 17-8 Selecting the WebSphere Customization Toolbox components to install

To access WebSphere Customization Toolbox, perform these steps:

1. On a Windows operating system, click **Start Programs** → **IBM WebSphere** → **WebSphere Customization Toolbox V8.5**.
2. Click **WebSphere Customization Toolbox** to start the program.

3. Click **z/OS Migration Management Tool** to open the WebSphere Customization Toolbox V8.5 **perspective** as shown in Figure 17-9.

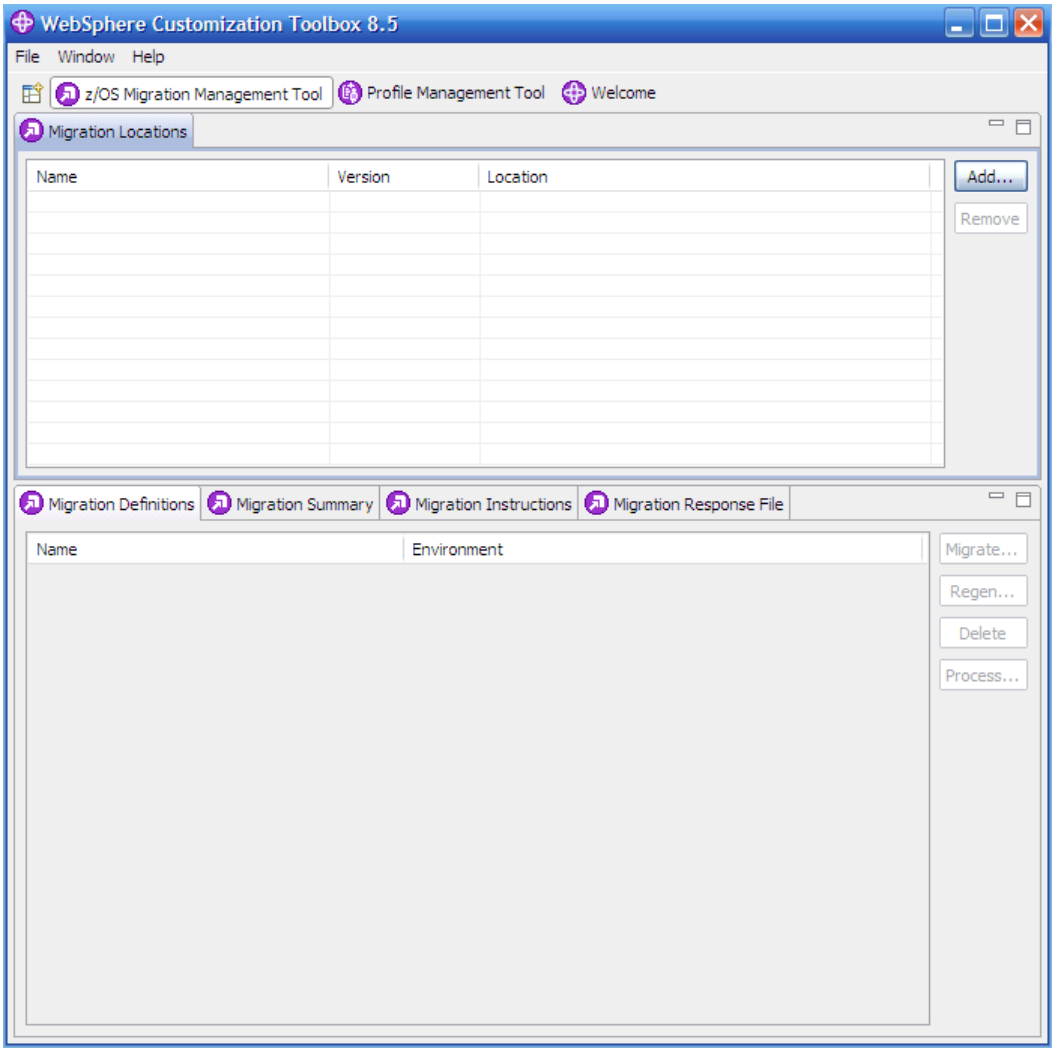


Figure 17-9 z/OS Migration Management Tool

Creating a migration definition

To create a migration definition, perform these steps:

1. Complete the configuration worksheet in the Websphere Application Server V8.5 Information Center at:

   `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tmig_zmmt_depmanwrk`

2. Start the z/OS Migration Management Tool.
3. Specify a location where you want Migration Definition files to be stored on your workstation, or add another migration location to the Migration Locations table:
   a. Click **Add** on the right side of the window.
   b. Enter the path name of the location where you want to store the migration data. The migration location directory must be empty when you create a migration location.
   c. Enter a name to be associated with the table entry.

d. Select the version of WebSphere Application Server to which you are migrating.

e. Click **Finish**.

4. Select the migration location that you created, and click **Migrate** as shown in Figure 17-10.

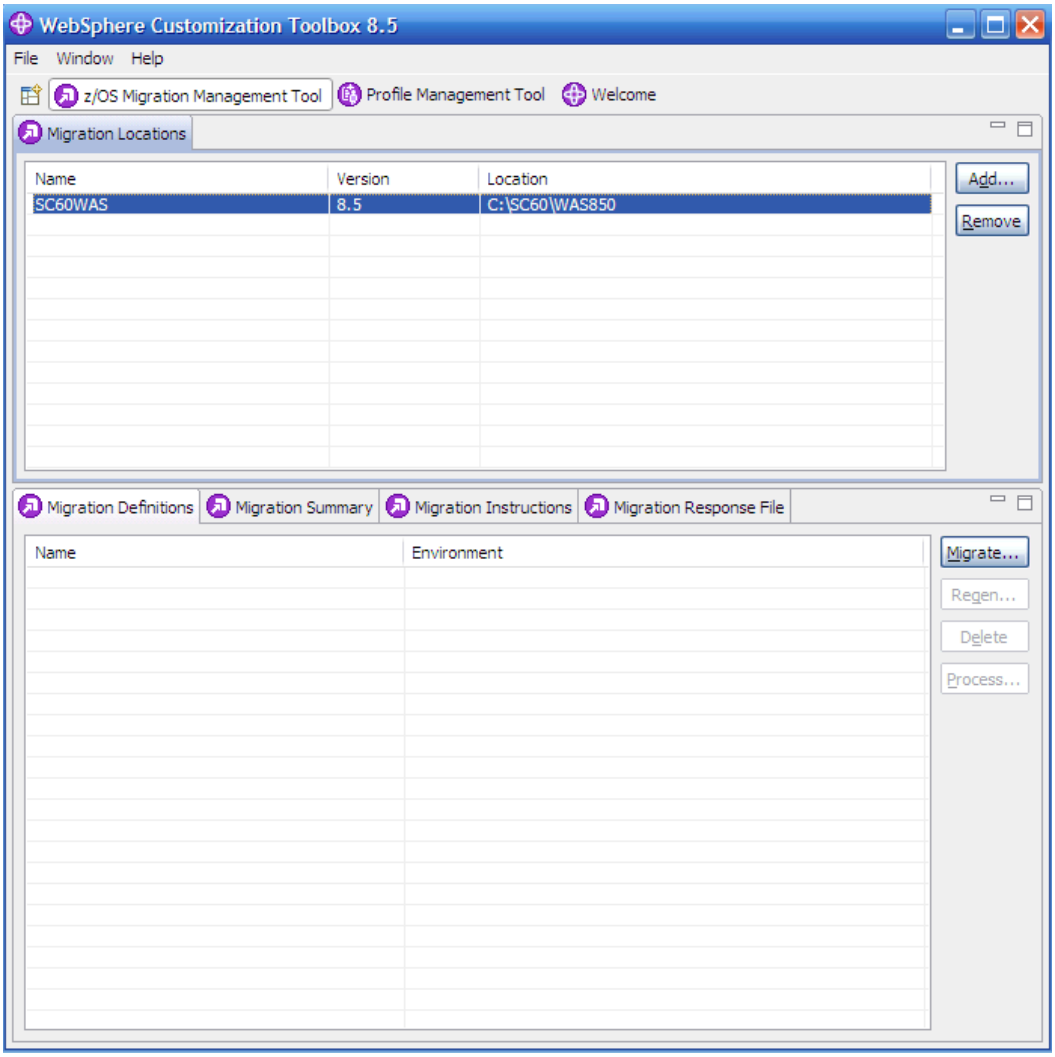


Figure 17-10 Migration process

5. In the Migration Node Type Selection window shown in Figure 17-11, select the type of node migration and click **Next**.

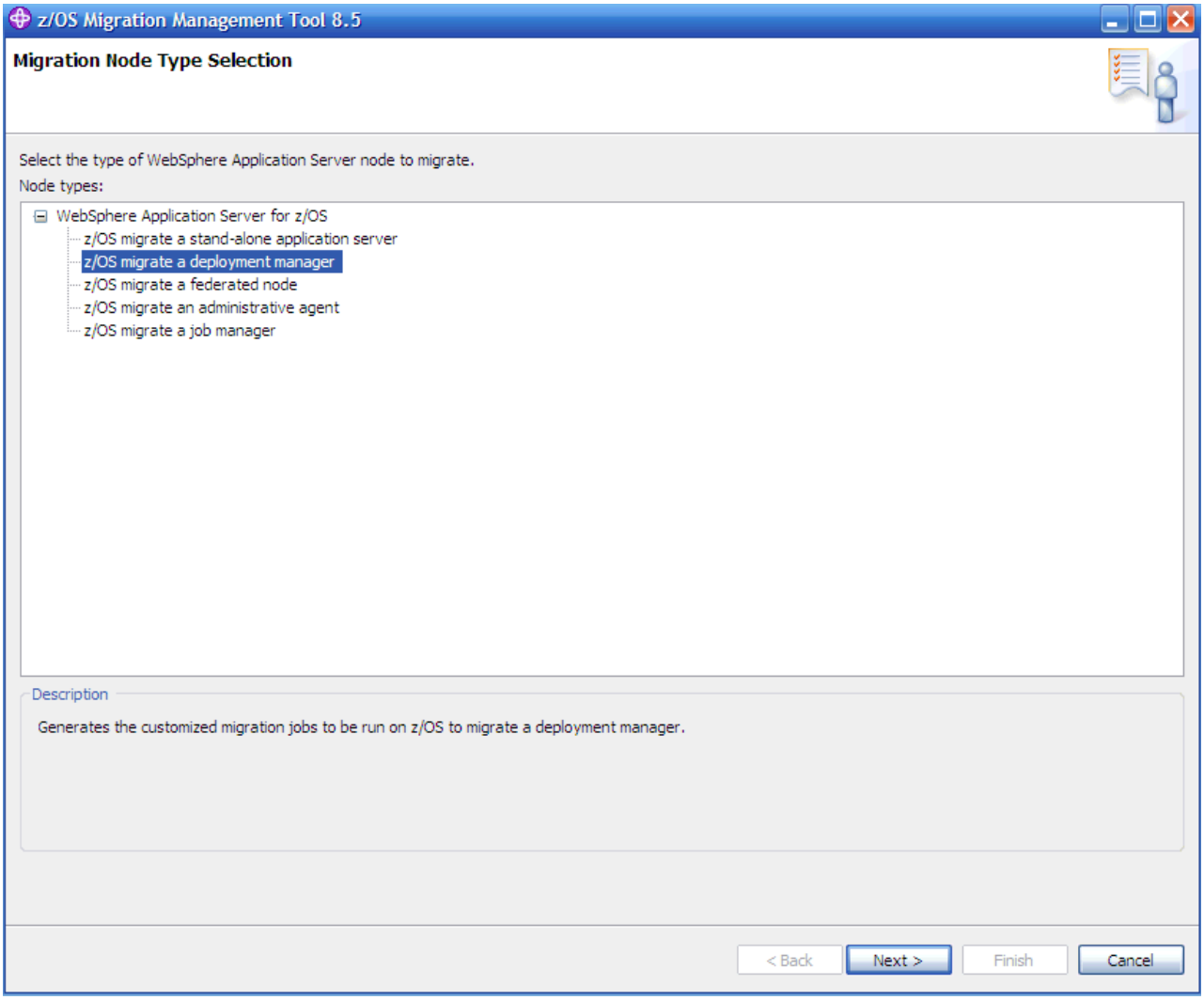


Figure 17-11 Migration Node Type Selection window

6. In the multi-panel window that opens, complete the fields by using the values that you entered for the variables on the configuration worksheet. Click **Back** and **Next** as necessary.

Considerations:

- The z/OS Migration Management Tool has a help file that is accessible by hovering the mouse over a field.
- In the Migration Process Options window, a Migration Definition identifier is shown. You might want to write down this number. This identifier is used to separate the output of individual node migrations. The identifier is also the name of the subdirectory where the JCL will be saved on your workstation.

7. After you successfully enter all of the necessary information for this type of Migration Definition, in the Migration Summary window, click **Create.** Doing so builds the Migration Definition on your workstation.
8. Check the definition type, location, and name information in the Migration Creation Summary window, then click **Finish**.

You will find a directory structure that populates the path that was specified to store the Migration Definitions. For the next steps, upload the migration jobs by using the z/OS Migration Management Tool, as explained in the following section.

For help regarding the z/OS Migration Management Tool, see the Websphere Application Server V8.5 Information Center at:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-zos&topic=tmig_zmmt_usemmt

Creating migration jobs

To create migration jobs, perform these steps:

1. To create the JCL jobs and scripts, select the definitions that you created under Migration Definitions and click **Process**, as shown in Figure 17-12.

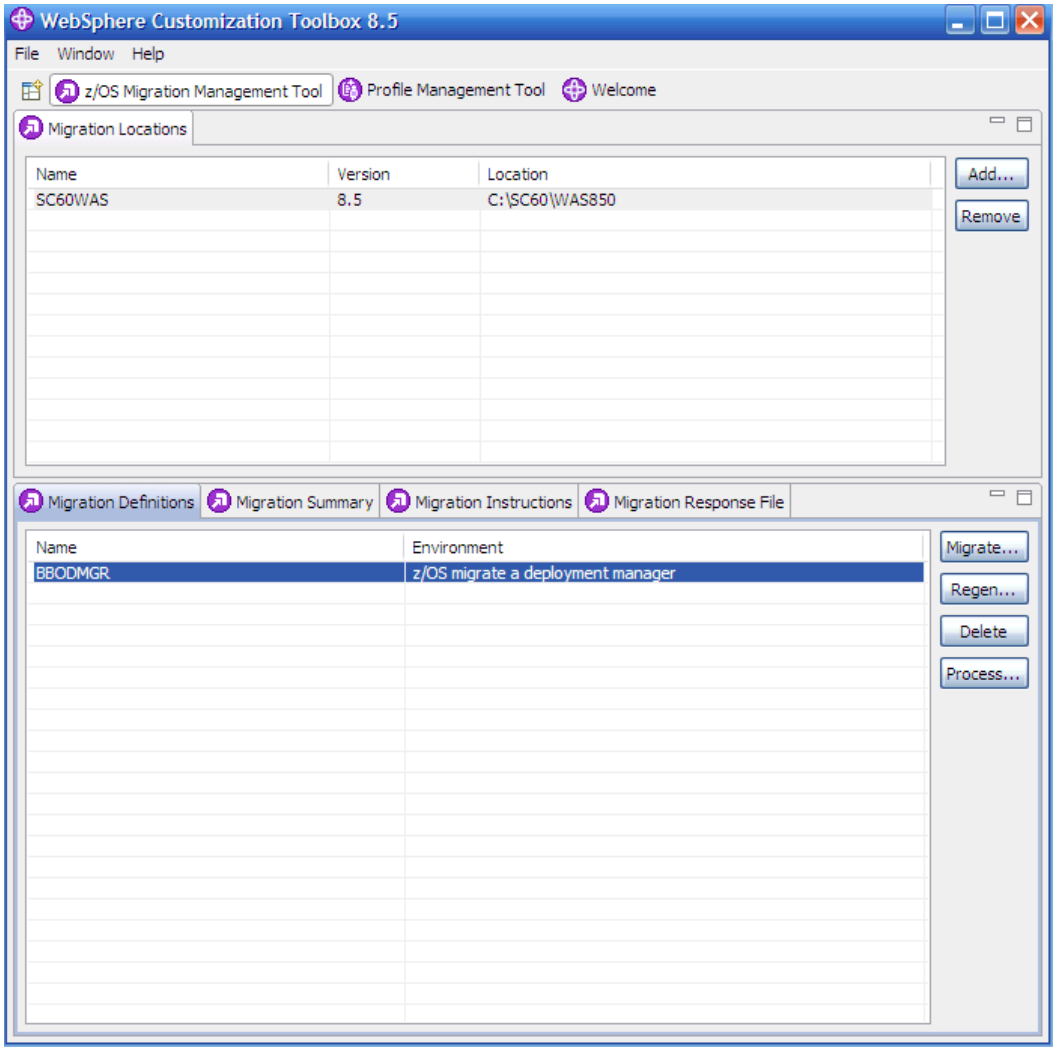


Figure 17-12 Processing the migration definition

2. Select the type of processing for the migration definition and click **Next**.
   - If you choose to upload to the target z/OS system, you must provide the host name or IP address, user ID, and password. The JCL and the scripts are then transferred to the z/OS system into the CNTL and DATA data sets named in the migration definition. The z/OS Migration Management Tool presumes that the data sets are preallocated. For CNTL and DATA data sets to be allocated during the transfer, select **Allocate target z/OS data sets** and specify the appropriate **Volume** and **Unit** fields.
   - If you choose to export to a local directory, the JCLs and scripts are generated on your local system.
3. Click **Finish**.

For detailed migration instructions, select a Migration Definition and then click the **Migration Instruction** tab. You can also find the instructions in the file system of your workstation in the BBOMxINS member. The path is displayed on the **Migration Instruction** tab. The instructions reflect the variables that were entered in the Migration Definition windows.

17.6.5 Migration Management Tool script

This section provides information about the z/OS Migration Management Tool script (`zmmt.sh`). It is used to create the JCL needed for a node migration.

Overview

You can create the migration jobs on z/OS by using the shell script `zmmt.sh`. This script is in the `bin` directory of the product image (default `/usr/lpp/zWebSphere/V8R5/bin`). The script creates the CNTL and DATA data sets and the corresponding JCL that is needed to perform the migration. You need a response file as input. This response file contains information about the node construction.

Requirement: Use the z/OS Migration Management Tool to build the response file. Make sure that any changes brought with new PTFs to the response file are used. You can achieve that by using the latest version of the WebSphere Customization toolbox. Do not copy the response files values directly from the z/OS Migration Management tool Migration Response File tab. Use the generated text file named `YourMigrationDefinitionName.responseFile`. This file is in a subdirectory of the profiles directory in your Migration Location.

The script

Example 17-1 shows how to run the script in the z/OS UNIX System Services environment.

Example 17-1 Migration management command

```
./zmmt.sh -workspace /xxx -transfer -allocate -responseFile
/xxx/YourMigrationDefinitionName.responseFile
```

You can use the following parameters to run the script:

- `-responseFile`

  Specifies the path to your response file. This file can be encoded in ASCII or EBCDIC. The shipped samples use ASCII. Some examples are in the `/usr/lpp/zWebSphere/V8R5/zOS-config/zpmt/samples` directory.

- `-profilePath`

  The fully qualified path name to an existing set of generated jobs. You cannot use this parameter with the `-responseFile` parameter.

- `-workspace`

  Specifies the Eclipse workspace directory.

- `-transfer`

  Copies generated jobs from a z/OS UNIX System Services file system to a pair of partitioned data sets. The `zmmt.sh` script first writes the customization jobs to a z/OS UNIX System Services file system.

- `-allocate`

  Attempts to allocate the target data sets.

This script runs the following tasks:

- Generates the migration jobs to the location specified by `profilePath` in the response file.
- Allocates the target CNTL and DATA data sets by using the high-level qualifier specified by target HLQ in the response file.
- Transfers the jobs from the file system to the CNTL and DATA data sets.

Runtime considerations

When using the `zmmt.sh` script to create the migration JCL, keep in mind the following points:

- The script is run in the **`osgi`** command shell.

  Tip: An **`osgi`** command shell is an execution environment that is used for remote management of Java applications and components. It is based on the OSGI open standard.

  Because the script takes a relatively long time to run, it might look as though nothing is happening. Eventually the script writes messages like those in Example 17-2.

  Example 17-2 osgi messages when issuing the zmmt.sh script

```
osgi> Customization definition successfully written to /tmp/ZDMgr01 Attempting
to allocate dataset: CUI.WAS85M.CNTL
Allocation successful.
Attempting to allocate dataset: CUI.WAS85M.DATA
Allocation successful.
Copying CNTL files to CUI.WAS85M.CNTL...
Copy successful.
Copying DATA files to CUI.WAS85M.DATA...
Copy successful.
```

- If you need to rerun the command, delete the `profilePath` directory. If the directory still exists, the **`osgi`** shell shows an error message as shown in Example 17-3.

  Example 17-3 zmmt.sh script error message

```
osgi> The following validation errors were present with the command line
arguments: profilePath: The profile path is not valid.
```

17.6.6 Migration jobs

Migration jobs are created by using the z/OS Migration Management Tool or the `zmmt.sh` script. This section provides a brief overview of these jobs.

Attention: Read the BBOMxINS module on the `hlq.CNTL` data set. It contains the tasks that you must perform before starting the migration process, and explains each job that you must perform.

Overview

Multiple jobs are created by the z/OS Migration Management Tool. Table 17-2 shows an overview of the jobs that are used for the migration. Jobs are relative to the node that must be processed. Check the detailed migration manual that is created during the JCL build step for the necessary user authorities.

Table 17-2 WebSphere Application Server for z/OS V8.5 migration jobs

Job name[a]	Job run
BBOMxZFS or BBOMxHFS	Allocates hierarchical file system (HFS) or zSeries file system (zFS)
BBOMxCP	Copies tailored JCLs to PROCLIB
BBOWMG1x	Clear transaction logs (for XA connectors only).
BBOWMG2x	Disable Peer Restart and Recovery mode (XA only).
BBOWxPRO	Creates a target profile in the new release.
BBOWxPRE	Creates a backup of the source profile.
BBOWxPOS	Migrates the backup profile into the new profile.
Alternatively BBOWMG3x	Runs BBOWxPRO, BBOWxPRE, and BBOWxPOS in a single step.

a. The value for x in the job names listed depends on the profile that you are migrating.

BBOWMG3x job

The BBOWMG3x job runs the actual migration. This job takes the longest time to run. The following tasks are included in the job:

1. Create a working directory (`/tmp/migrate/`*nnnnn*).

   A working directory in the `/tmp` directory is used to do much of the processing. The *nnnnn* is a unique number generated during the creation of your migration jobs. For normal migration, the space used in the `/tmp` directory is small. However, if you enable tracing, the space demand can become higher. Make sure that you have enough free space in the `/tmp` directory.

2. WRCONFIG: Copy the dialog generated variables to the HFS.
3. WRRESP: Create a profile creation response file from the dialog generated variables.
4. MKCONFIG: Gather information, such as the cell name and server name, from the existing configuration.
5. VERIFY: Verify the variables generated from the dialog.

   This step attempts to check that the information provided so that the migration does not fail because of bad input parameters.

6. CRHOME: Create a V8.5 WAS_HOME structure.

7. CRPROF: Create a V8.5 profile for the node that is being migrated.
8. PREUPGRD: Back up the files in the file system in preparation for migration.
9. UPGRADE: Run WASPostUpgrade to perform the migration (`serverindex.xml` renamed to `serverindex.xml__disabled`).

   This step is where the actual migration occurs, and takes the longest to complete.
10. FINISHUP: Run **`Config2Native`**, and update file permissions and attributes.

Troubleshooting for the BBOWMG3x job

Because a migration is complex, errors can occur. The main source for errors is the BBOWMG3x job, which was described in the previous section.

Here are some troubleshooting tips:

- If the BBOWMG3x job fails, check the output for errors:
  - `/tmp/migrate/nnnnn/BBOWMG3x.out` written to JOBLOG
  - `/tmp/migrate/nnnnn/BBOWMG3x.err` written to JOBLOG
  - `/tmp/migrate/nnnnn/logs` directory can contain log files, with a name such as the `WAS*Upgrade*timestamp.log file`.
- If you need more information, enable traces. The trace states are disabled by default. Be aware that `'xxxx.DATA(BBOWMxEV)'` must be updated to enable tracing:
  - `TraceState=enabled`
  - `profileTrace=enabled`
  - `preUpGradeTrace=enabled`
  - `postUpGradeTrace=enabled`
- If the job fails in the VERIFY step, it is likely that an error was made when specifying the information to use to create the jobs. Correct the information and rerun the job.
- If the job fails after the VERIFY step, delete the `WAS_HOME` directory. This directory is created during the failed run Delete the directory before rerunning the job. Check wether the original configuration for the `serverindex.xml` file has been renamed to `serverindex.xml_disabled`.

  The job failure is a signal that the configuration has already been migrated and to stop you from inadvertently migrating the node again. To change the default setting during the configuration phase, select **Disable previous deployment manager** in the Server Customization (Part 2) window. This window is a part of the z/OS Migration Management Tool. Alternatively, set the `keepDMGREnabled` parameter to `true` in the response file.

> **Tip:** The BBOWMG3x job can cause error conditions, such as an abend 522, because it runs for a long time. TIME=NOLIMIT on the JCL job card can avoid the problem. The BBOWMG1x and BBOWMG2x jobs are only necessary if you have any XA connectors defined in your configuration. They do not apply to the deployment manager node migration.

17.6.7 Migration considerations for 64-bit mode

Websphere Application Server V8.5 runs in 64-bit mode by default and 31-bit runtime mode is deprecated. Keep in mind the considerations highlighted in this section.

Application considerations

For code written in pure Java, the general experience is that no changes are necessary to the code for it to run in a 64-bit application server. If the application uses the Java Native Interface

(JNI) to call a native program, it must be a 64-bit program. Typically, these native programs are code written in C, C++, or perhaps an IBM Language Environment compliant assembler. This point is important to verify when using in-house applications that use older native programs.

For more information about how to convert applications to run in 64-bit mode, see the following resources:

- *C/C++ Code Considerations With 64-bit WebSphere Application Server for z/OS* (WP101095):

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101095

- *Language Environment Programming Guide for 64-bit Virtual Addressing Mode*, SA22-7569-06

  http://publibz.boulder.ibm.com/epubs/pdf/ceeam160.pdf

Larger heap sizes for applications

Use of the 64-bit addressing mode does not mean that the sizes for the various heaps need to be increased. In general, identify minimum and maximum heap sizes with a verbose garbage collection analysis. With this technique, you can identify values that reduce the garbage collection processor usage, saving processor time. Consider performing a verbose garbage collection analysis on a regular basis, especially if the number of users or the number of transactions have changed.

Explanation: In general, the structure of WebSphere Application Server for z/OS reduces the maximum heap size compared to those used by distributed platforms. For more information, see 16.1.6, “Runtime processes” on page 507.

A

Sample topology walkthrough

This appendix explores a complex topology and provides general guidance for setting it up. This appendix includes the following sections:

- Topology review
- Sample topology
- Installation
- Deploying the applications
- Configuring security
- Testing the topology
- Summary

Topology review

Figure A-1 illustrates one of the most common (and complex) topologies implemented in real scenarios. This configuration was provided by customers and IBM teams who are responsible for the implementation of WebSphere environments. This topology provides great resiliency because all points of failure are eliminated. It also takes advantage of almost all the components included in the WebSphere Application Server Network Deployment package.

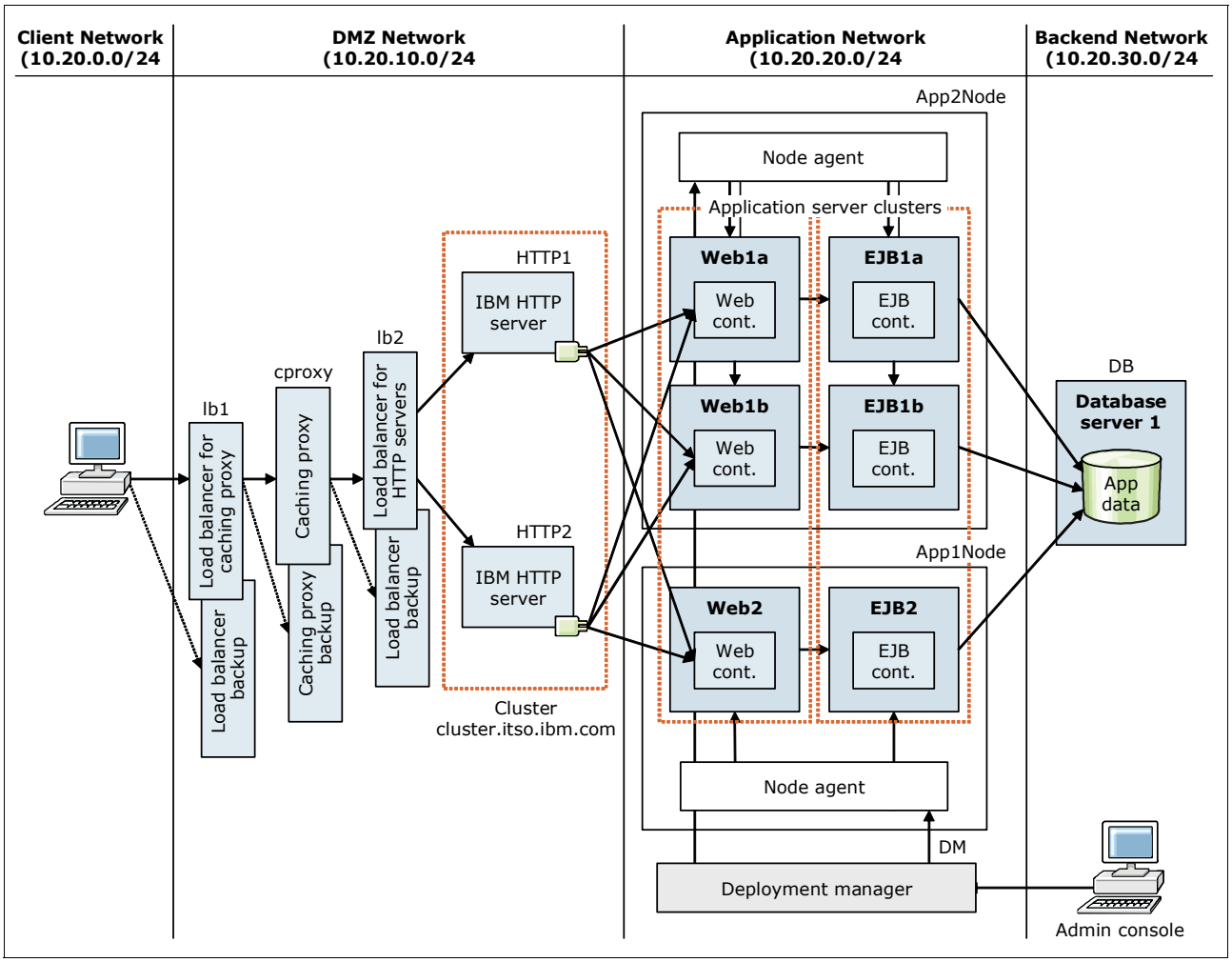


Figure A-1 Complex topology

This topology includes the following elements:

- A load balancer to direct incoming requests to the caching proxy, and a second load balancer to manage the workload across the HTTP servers.

  Load Balancer is included in the WebSphere Application Server Edge Component. Load Balancer distributes incoming client requests across servers, balancing workload and providing high availability by routing around unavailable servers. A backup server is configured for each primary Load Balancer to provide high availability.

- A Caching Proxy with a backup server in passive mode for high availability.

  Caching Proxy is included in WebSphere Application Server Edge Components. Cacheable content includes static web pages and JavaServer Pages (JSP) with dynamically generated but infrequently changed fragments. The Caching Proxy can

satisfy subsequent requests for the same content by delivering it directly from the local cache. This process is much quicker than retrieving it again from the content host.

- Two IBM HTTP Server Web servers configured in a cluster

  Incoming requests for static content are served by the web server. Requests for dynamic content are forwarded to the appropriate application server by the web server plug-in.

- A dedicated server to host the deployment manager

  The deployment manager is required for administration, but is not critical to the runtime execution of applications. Having a separate server for the deployment manager instead of placing it in on one of the node's servers leaves more resources available for the application servers. Also, if a problem occurs with the node server, the administration of the other nodes is still possible. Additionally, the deployment manager has a master copy of the configuration that must be backed up on a regular basis.

- Two clusters that consist of three application servers

  Each cluster spans two systems. In this topology, one cluster contains application servers that provide the web container functions of the applications (servlets and JSP). The second cluster contains the Enterprise JavaBeans (EJB) container functions. Whether you choose to use clusters is a matter of careful consideration. Although using clusters provides failover and workload management capabilities for web and EJB containers, it can also affect performance.

- A dedicated database server that runs the database.

Advantages

This topology has the following benefits:

- High availability and failover support

  The redundancy of the different elements eliminates single points of failure (SPOFs) and provides hardware and software failure isolation.

- Optimized resource use

  Vertical scaling provides the benefit for each Java virtual machine (JVM) to use a portion of the system's processor and memory. More JVMs can be created to take advantage of the available resources.

- Workload management

  In this topology, workload management is done by the Load Balancer, which distributes work among the web servers. In addition, the WebSphere Plug-ins load balance work across the application servers.

- Improved throughput and response time

  Multiple systems serve client requests without competing for resources, and the resources on the servers are optimally used.

- Scalability

  With this type of topology, you can add more JVMs if necessary. As more nodes or more web servers are added, the loader balancer distributes the load across the members added to the original topology as well.

Disadvantages

This topology has the following disadvantages:

- Complex administration

  Several different systems need to be administered, configured, and maintained. Consider the costs of such administrations in relation to the benefits of increased performance, higher throughput, and greater reliability.

- Increased cost

  More hardware, and, therefore more licenses, are required, which increases overall costs.

Sample topology

This section presents a simplified topology derived from the one illustrated in Figure A-1 on page 576. This sample topology demonstrates the necessary steps to implement the main elements of the complex topology as explained in “Topology review” on page 576. Figure A-2 shows the topology that is implemented in this section.

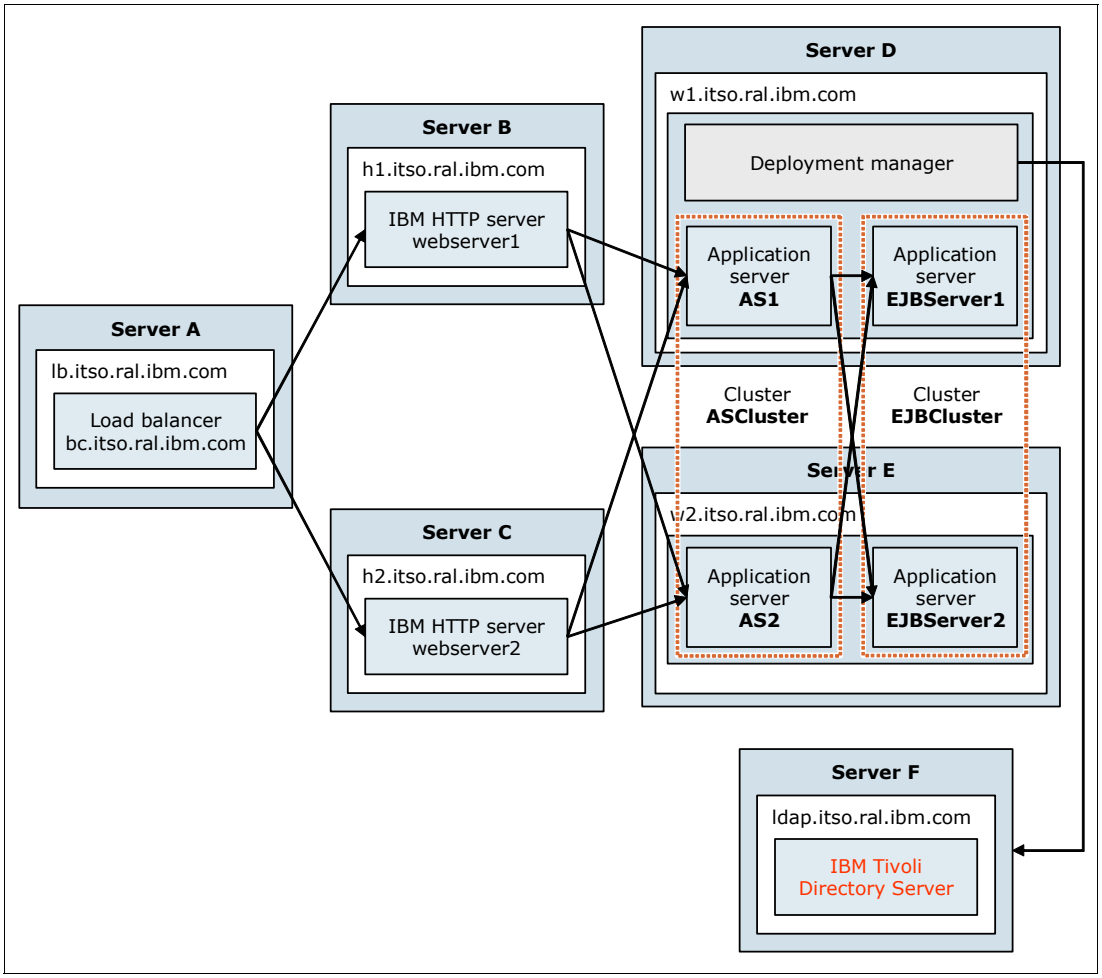


Figure A-2 Sample topology

Characteristics

This topology has the following characteristics:

- The topology allows for the deployment of the BeenThere sample application that is included in the material that is available for download for this book. For more information, see Appendix C, “Additional material” on page 601.

  The BeenThere sample application demonstrates the workload management and clustering capabilities of Websphere Application Server. Because its responses contain the application servers where requests are processed, you can test load balancing and availability with this application.
- It is common to have the administrative or application security integrated with a Lightweight Directory Access Protocol (LDAP) repository in production environments. Because of this integration, another element is added to the sample topology: The IBM Tivoli Directory Server offering for an LDAP server.
- A Load Balancer is installed to demonstrate its workload management capabilities for sending requests to the web servers. Because there is no backup server for the load balancer, this element is a SPOF, which is not a critical issue for this sample topology.
- No database is installed because, for testing purposes of this topology, it is not necessary to query data from a database.

Installation

This section explains the main steps for installing each component. In WebSphere Application Server V8.5, all elements are installed with the Installation Manager.

Tip: If the servers where the installation is going to occur do not have an Internet connection, use local repositories. To prevent Installation Manager from searching for the packages over the Internet, clear the remote repositories by clicking **File** → **Preferences**.

Installing Load Balancer (Server A)

The Load Balancer component runs in Server A, and is used to distribute traffic between the two web servers.

To install Load Balancer, complete these steps:

1. Install Load Balancer:
   a. Install the Installation Manager, accepting the default values.
   b. Start the Installation Manager, and select the **Install** option.
   c. From Installation Packages, select **IBM WebSphere Edge Components: Load Balancer for IPv4 and IPv6**, and click **Next**.
   d. Go through the remaining windows, accepting the default options for each one.
2. Configure the load balanced cluster:
   a. Verify that the load balancer service, IBMDisp(ULB), is started.
   b. Run the `lbadmin` command.
   c. Right-click **Dispatcher Connect to Host**.
   d. Right-click **Dispatcher Start Configuration Wizard**.

e. Following the wizard panels, create a cluster with the new web cluster IP address, and add the two web servers in it. This new IP address is the public address that clients must use to access the BeenThere application. It is a different IP address from the system addresses.

f. Right-click **Host** and select **Start Manager** to configure the Advisor.

g. Right-click **Manager** and select **Advisor**. Then monitor the HTTP protocol. This action detects when a web server is down and stops sending requests to that server.

3. Configure loopback adapters in web servers:

a. Add a loopback adapter interface in both HTTP server systems (Servers B and C). Installing this interface is necessary in Windows platforms only.

b. Configure the cluster address and the corresponding subnet mask in the loop back adapters.

Installing the HTTP servers (Servers B and C)

Both Servers B and C run a web server that directs the incoming requests to one of the application servers. To install the HTTP servers, complete these steps on both servers:

Perform the following steps on both servers:

1. Install the Installation Manager.
2. Install the IBM HTTP Server:

a. Start the Installation Manager, and select the **Install** option.

b. From Installation Packages, select **IBM HTTP Server for WebSphere Application Server** and click **Next**.

c. In the wizard panels, select the default options, and complete the installation.

d. Start the server by clicking **Start Programs** → **IBM HTTP Server for WebSphere Software V8.5** → **Start HTTP Server**.

3. Install the WebSphere Plug-in:

a. Start the Installation Manager, and select the **Install** option.

b. From Installation Packages, select **Web Server Plug-ins for IBM WebSphere Application Server**, and click **Next**.

c. In the wizard panels, select the default options, and complete the installation.

4. Install the WebSphere Customization Toolbox:

a. Start the Installation Manager, and select the **Install** option.

b. From Installation Packages, select **WebSphere Customization Toolbox**, and click **Next**.

c. Click to clear the **Profile Management Tool (z/OS only)** and **z/OS Migration Management Tool** options.

d. In the other wizard panels, select the default options, and complete the installation.

e. Start the WebSphere Customization Toolbox.

f. Add a WebSphere Plug-in Runtime Location. Use the `plugin_root` path as the location.

g. Create a WebSphere Plug-in Configuration for IBM HTTP Server V8.5.

Creating a deployment manager (Server D)

Server D hosts the deployment manager that is used to administer the servers, applications, and resources in the WebSphere Application Server cell.

To build this node, complete these steps:

1. Install WebSphere Application Server Network Deployment:
   a. Install IBM Installation Manager.
   b. From Installation Packages, select **IBM WebSphere Application Server Network Deployment**.
   c. Navigate through the panels in the installation wizard, selecting the default options.
   d. Leave the option to start the Profile Management Tool selected, and click **Finish**.
2. Create a deployment manager profile:
   a. Click **Create**.
   b. Create a management profile of type *deployment manager* with the default options.
3. Start the deployment manager by clicking **Start Programs** → **IBM WebSphere** → **Websphere Application Server Network Deployment V8.5** → **Profiles** → **Dmgr01** → **Start the deployment manager**.

Creating the application servers (Servers D and E)

Server D and E host the two application server clusters that are needed for the BeenThere application. The process to install and build the WebSphere Application Server components is the same for each application server node.

To create the application servers, complete these steps:

1. Install WebSphere Application Server Network Deployment:
   a. Install IBM Installation Manager.
   b. From the Installation Packages, select **IBM WebSphere Application Server Network Deployment**.
   c. Navigate through the panels in the installation wizard, selecting the default options.
2. Create a custom profile for the node:
   a. Start the Profile Management Tool, and select **Custom Profile**.
   b. Follow the prompts, accepting the defaults, including the federation of the node to the cell as part of the process. The deployment manager must be installed and running during this process.
3. Create the application server clusters:
   a. Log on to the administrative console.
   b. Click **Servers Clusters WebSphere application server clusters**.

c. Add the two new clusters with the names and member weights shown in Table A-1.

Table A-1 Cluster names and weights

Cluster name	Member name	Weights
ASCluster	AS1	2
	AS2	3
MyEJBCluster	EJBServer1	1
	EJBServer2	3

The chosen weights are the weights indicated in the BeenThere application installation instructions, and are intended to illustrate the workload management capabilities of the product.

Enabling the WebSphere configuration service

According to the application installation instructions, the WebSphere configuration service must be enabled for the application to read WebSphere Application Server configuration files to obtain environment information.

To enable the WebSphere configuration service, complete these steps:

1. Log on to the administrative console.
2. For both application servers from the ASCluster, complete the following steps:
   a. Click **Servers** → **Application Servers** → ***server_name*** → **Server Infrastructure** → **Administration** → **Administration Services** → **Custom Properties**.
   b. Create a property called `com.ibm.websphere.management.enableConfigMBean` with a value of `true`.

Deploying the applications

For the deployment of the application, the new monitored directories feature is used. To deploy the applications, complete these steps:

1. Enable the monitored directory:
   a. Log on to the administrative console.
   b. Click **Applications** → **Global Deployment Settings**.
   c. Select the **Monitor directory to automatically deploy applications** option.
   d. Leave the default values, and then click **Apply**.
   e. Restart the deployment manager.
   f. Create a directory, called `ASCluster`, in the `dmgr_profile_path/monitoredDeployableApps/clusters` path.
2. Deploy the BeenThere application:
   a. Make sure that ASCluster is running.
   b. Copy the BeenThere enterprise archive (EAR) file into the ASCluster directory.

c. After 5 seconds, log on to the administrative console. Then click **Applications** → **Application Types** → **WebSphere Enterprise Applications**. The BeenThere application should be listed with the status of *starting*.

3. Map the EJB module to the MyEJBCluster cluster:
   a. Click **BeenThere** → **Manage Modules**.
   b. Select the check box for the **BeenThereEJB** module and the **MyEJBCluster** row under Cluster and Servers. Then click **Apply**.

Explanation: You can also use a property file to deploy the web archive (WAR) module to the ASCluster and the EJB module to the MyEJBCluster automatically. However, for simplicity, the administrative console was used in this example. To create the property file and use it to deploy the modules, see the Websphere Application Server V8.5 Information Center at:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-base-dist&topic=trun_app_install_dragdrop`

4. Generate a new plug-in for the IBM HTTP Servers:
   a. Click **Environment Update global Web server plug-in configuration**.
   b. Click **OK** to update the plug-in file.
   c. Copy the new plug-in file to webserver1 and webserver2. The default target directory is `C:\Program Files\IBM\WebSphere\Plugins\config\`*`your_web_server`*.

Configuring security

Many of the production environments in today's organizations manage their users in LDAP directories. This section describes the necessary steps to connect the WebSphere Application Server security with IBM Tivoli Directory Server.

Because the profile for the deployment manager was already created with security enabled, complete these steps:

1. Change the administrative user registry:
   a. Log on to the administrative console.
   b. Click **Security** → **Global Security**.
   c. Under the User account repository, select **Standalone LDAP registry**, and click **Configure**.
   d. Enter the corresponding information for the following fields:
      - Primary administrative user name
      - Type of LDAP server (in this case IBM Tivoli Directory Server)
      - Host
      - Port
      - Base distinguished name (DN)
      - Bind distinguished name (DN)
      - Bind password
   e. Click **Test Connection** to make sure that the communication with the LDAP server has no problems.
   f. Click **OK**, and save the changes.

g. Make sure that the stand-alone LDAP registry is selected under **User account repository**. Click **Set as current**, and then click **Apply** to validate all the entered settings. If no errors are shown, save the changes.

> **Tip:** If you receive the following error, verify that the user filter is correct in the user registry settings:
>
> `Validation failed: SECJ7716E: Primary administrative user Id does not exist in the registry.`

2. In the Global security panel, select **Enable application security**.
3. Assign administrative roles.

   The user account that is being used to log on to the BeenThere application needs an administrative role. Otherwise the application will not work because the application needs to get the node name from Websphere Application Server. Therefore, the user role is *not* enough.

   To assign this role to the user, complete these steps:

   a. Click **Global security Administrative user roles**.
   b. Click **Add**.
   c. Search for the user that is going to be used with the application, and assign that user the *monitor* role.
   d. Click **OK** and save the changes.
4. Assign application roles.

   This application has a role (administrator) that must be assigned to the user. To assign application roles, complete these steps:

   a. Click **Applications** → **Application types** → **WebSphere enterprise applications**.
   b. Select **BeenThere,** and click **Security role to user / group mapping**.
   c. Map the user from the LDAP repository to the *administrator* role.
   d. Click **OK**, and save the changes.
5. Stop all the processes in the cell, and start the deployment manager, node agents, and application servers in the order listed.

Testing the topology

Because the tests were intended to demonstrate the clustering, load balancing, and high availability capabilities of the sample topology, the environment was challenged in different conditions. For each test case described in this section, the following address was entered in the web browser:

`http://bc.itso.ral.ibm.com/wlm/BeenThere?weights=false&count=4`

Normal functioning

In this test, every component was up and running to show the load balancing features in Websphere Application Server. Because this was the first test, the window shown in Figure A-3 on page 585 opened. Enter the credentials of the user stored in the LDAP repository and click **OK**.

Figure A-3 Authentication dialog box

Figure A-4 shows the response from the application. The request was processed by the servlet on node w2Node1. Three of the four iterations were processed by EJBServer2 and the other one by EJBServer1. This distribution of the requests to the EJB servers is because of the weights that were configured when those servers were created (Table A-1 on page 582).

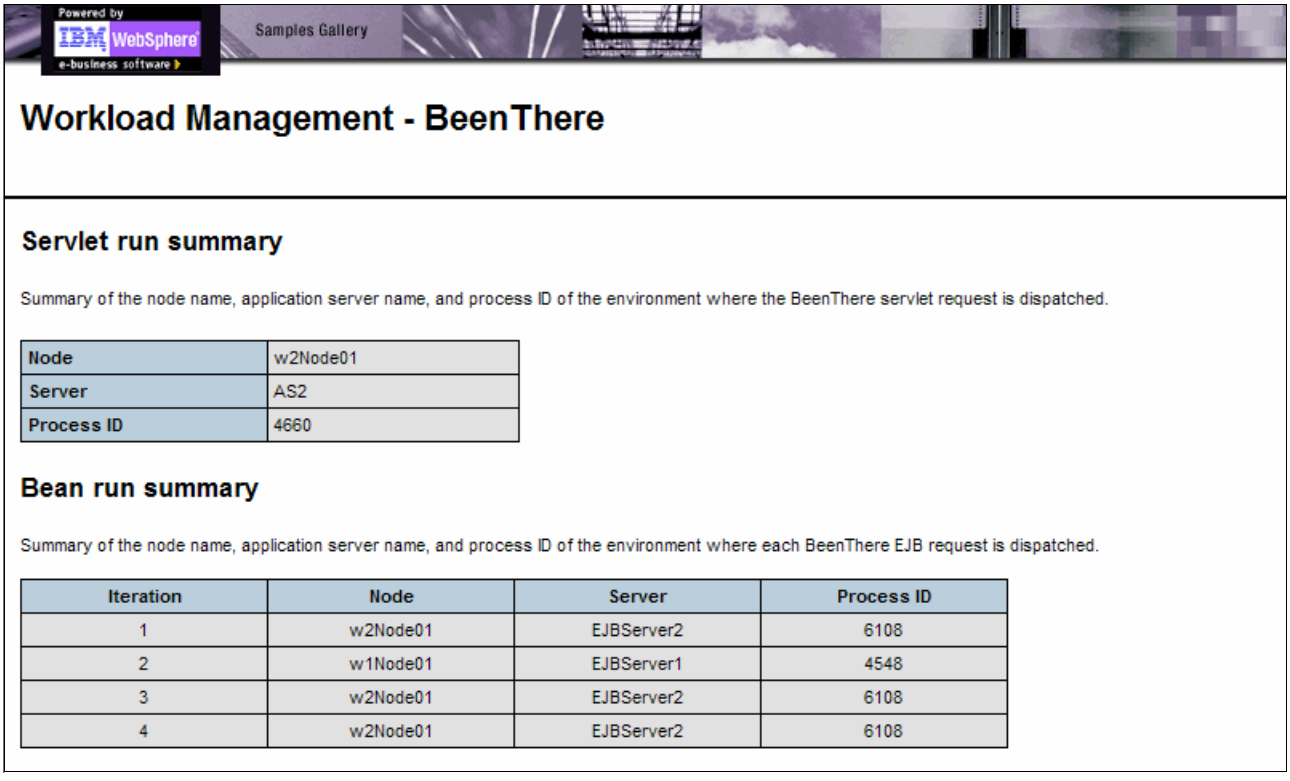


Figure A-4 Response from the BeenThere application

Remember: If the EJBServer is the same for the four iterations, but alternates between EJBServer1 and EJBServer2 after refreshing the page several times, the work load management is working. The *prefer local feature* in the MyEJBCluster is enabled by default, which causes all enterprise bean requests to be routed to the client host. You can disable this feature, restart the MyEJBCluster, and test again to see how the requests are load balanced according to the configured weights. Keep in mind that the prefer local feature improves performance and must remain enabled in production environments.

One web server down

In this test, webserver1 was stopped. Load Balancer detected this situation. In the Advisor Status window (Figure A-5), a value of -1 in the load column for the h1.itso.ral.ibm.com server indicates that it is not available.

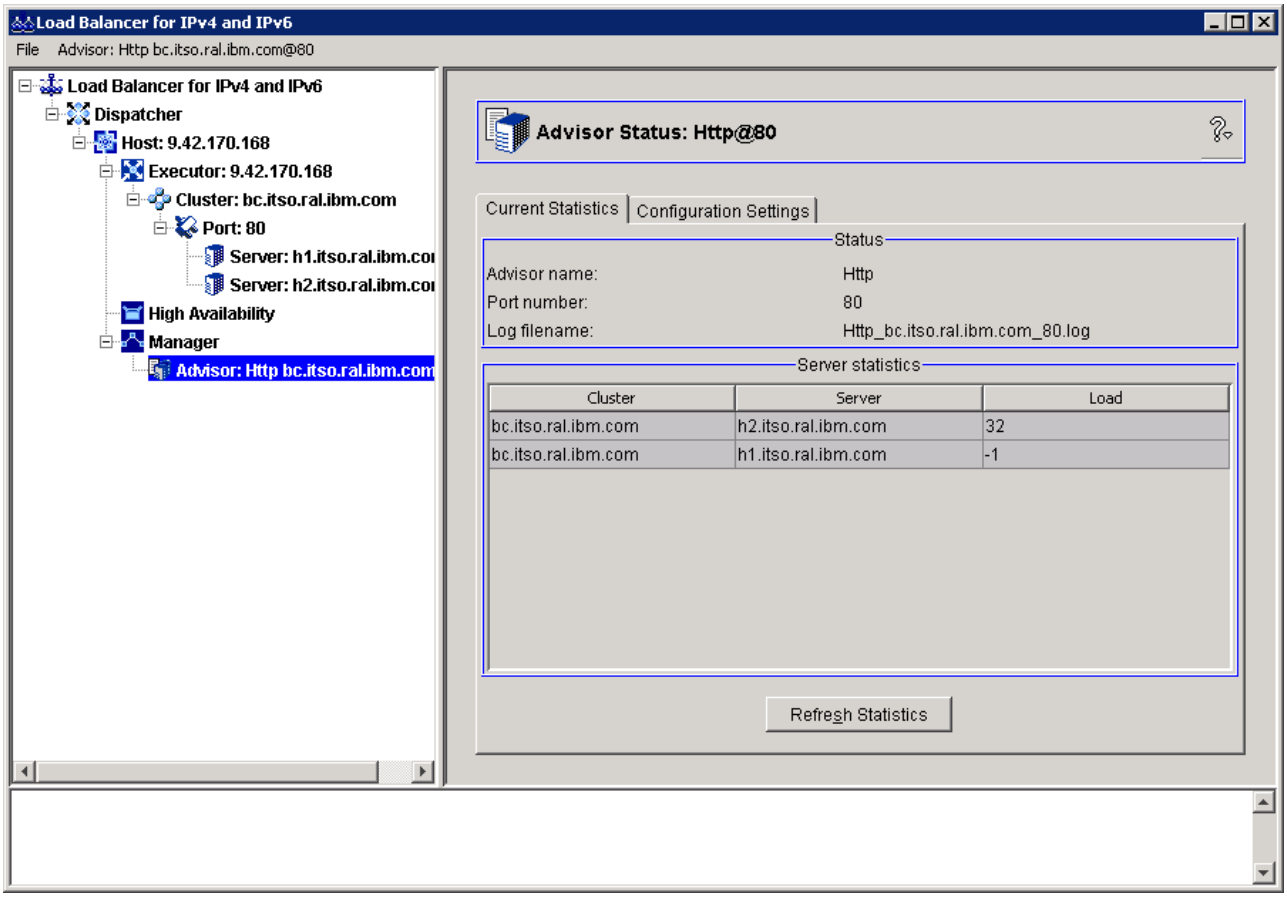


Figure A-5 Advisor Status window

Due to the load balancing mechanism, this environment was still working, and new requests to the system received the correct responses. From a user point of view, the environment behavior did not change.

One Websphere Application Server node down

In the last of the test scenarios, the servers at w2.itso.ral.ibm.com were stopped. The system still responded to requests, which were all processed by w1.itso.ral.ibm.com (Figure A-6).

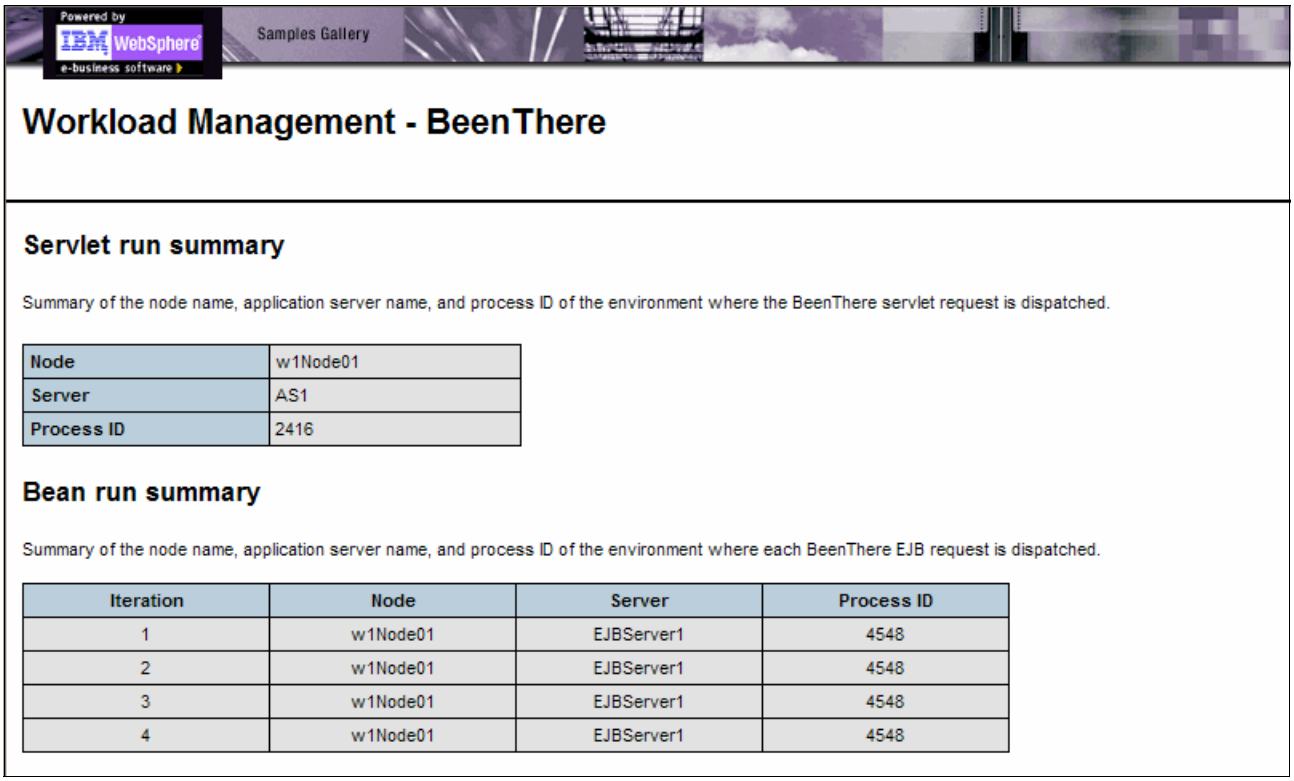


Figure A-6 All responses from EJBServer1

Summary

This appendix highlighted a commonly used complex topology. It provided a simplified version of this topology and used it to illustrate the installation of the different components. The scenario demonstrated its response to the challenges. Finally, this appendix highlighted the administration capabilities that can be done by running jobs from the deployment manager. These jobs include collecting files, creating profiles, and discovering installed resources on the job targets.

B

Sample topology using the job manager and a Liberty profile

This appendix shows how to set up a topology with the job manager and the Liberty profile. The topology was introduced in 8.3.3, “Liberty profiles managed by a job manager” on page 202.

This appendix contains the following sections:

Sample topology

This simple topology uses only one HTTP server. A setup with two HTTP servers and a load balancer is also possible, and removes the single point of failure (SPOF) at the HTTP Server level. This sample topology uses one job manager and three Liberty profiles on different nodes, as shown in Figure B-1.

Remember: The process described in this appendix is an example and shows one of many options you have when creating the Liberty profiles with a job manager.

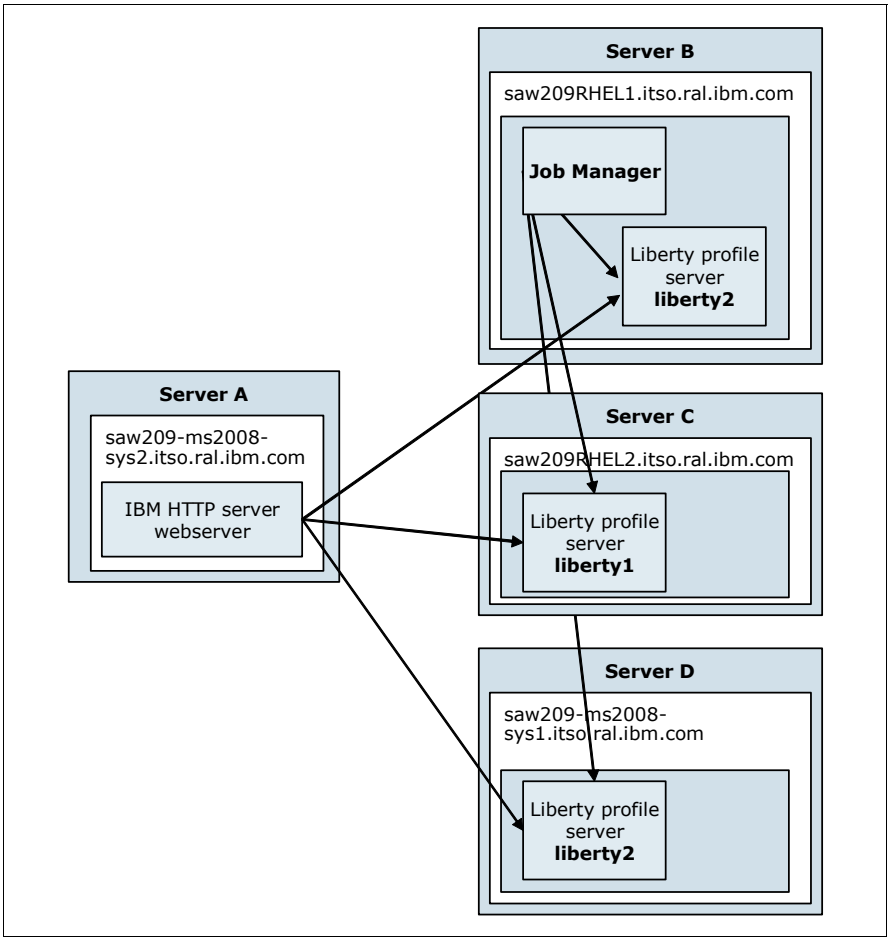


Figure B-1 Sample topology with Liberty profiles

Installing the HTTP server on Server A

To install the HTTP Server on System A, complete these steps:

1. Install the Installation Manager.
2. Install the IBM HTTP Server:
   a. Start the Installation Manager, and select the **Install** option.
   b. From Installation Packages, select **IBM HTTP Server for WebSphere Application Server**, and click **Next**.

c. In the wizard panels, select the default options, and complete the installation.

d. Start the server from **Start Programs** → **IBM HTTP Server for WebSphere Software V8.5** → **Start HTTP Server**.

3. Install the WebSphere Plug-in:

a. Start the Installation Manager, and select the **Install** option.

b. From Installation Packages, select **Web Server Plug-ins for IBM WebSphere Application Server**, and click **Next**.

c. In the wizard panels, select the default options, and complete the installation.

4. Install the WebSphere Customization Toolbox:

a. Start the Installation Manager, and select the **Install** option.

b. From Installation Packages, select **WebSphere Customization Toolbox**, and click **Next**.

c. Clear the **Profile Management Tool (z/OS only)** and **z/OS Migration Management Tool** options.

d. In the other wizard panels, select the default options, and complete the installation.

e. Start the WebSphere Customization Toolbox.

f. Add a WebSphere Plug-in Runtime Location. Use the `plugin_root` path as the location.

5. Create a WebSphere Plug-in Configuration for IBM HTTP Server V8.5.

Installing the WebSphere job manager on Server B

The WebSphere job manager needs a WebSphere Application Server Network Deployment installation. To build this node, complete these steps:

1. Install WebSphere Application Server Network Deployment:

a. Install IBM Installation Manager.

b. From Installation Packages, select **IBM WebSphere Application Server Network Deployment**.

c. Go through the panels in the installation wizard, selecting the default options.

d. Launch the Profile Management Tool, and click **Finish**.

2. Create a job manager profile:

Open a terminal and navigate to the `bin` subdirectory of the application server directory. If you installed the application server in `/opt/IBM`, navigate to `/opt/IBM/AppServer/bin`. Enter the following command:

```
./manageprofiles.sh -create -profileName JobMgr -templatePath
/opt/IBM/AppServer/profileTemplates/management -nodeName saw209RHEL1JobMgr
-cellName JobMgrCell -hostName saw209RHEL1 -serverName jobmgr -adminUserName
admin -adminPassword ******* -enableAdminSecurity true -isDefault -serverType
JOB_MANAGER
```

3. Start the job manager by entering:

```
./startServer.sh jobmgr
```

4. Call the job manager from a web browser:

```
http://saw209RHEL1:9960/admin
```

The IP Port might be different. You can find the ports for your profile in the `AboutThisProfile.txt` file, which is in the `AppServer/profiles/JobMgr/logs` directory.

Installing the Liberty profiles, servers, and applications on servers B, C, and D

You can install the Liberty profiles, servers, and applications by using the job manager. When this installation starts, there is no need for any WebSphere code on the systems on which the Liberty profiles run. However, if you do not include the JRE in the compressed file, make sure that a Java runtime environment (JRE) is installed on these systems.

Install a Java Runtime Environment on Servers B, C, and D

The minimum supported levels are IBM JDK 626 SR 6 or Oracle Java 6 SR 26. Higher levels (including Java 7) are supported but might have restrictions.

After you install a supported JRE, make sure that the jre/bin directory is in one of these locations:

- The JAVA_HOME
- The PATH variable for the administrative user that you used to install the Liberty profile

Create a compressed file that contains the servers and applications

You can download an archive file that contains the WebSphere Liberty profile at:

`http://wasdev.net`

You can use any extraction tool to create a package file of the server directory (`/wlp/usr/servers`) within the compressed file. The name of the directory is the name of your server. The directory you create must contain the server `server.xml` configuration file. The server directory contains an `apps` subdirectory that contains all applications you want to deploy as web archive (WAR) or enterprise archive (EAR) files.

Alternatively, you can also use the **`package`** command to package a server.

Deploy the Liberty profiles by using the job manager

To deploy a Liberty profile, log on to the job manager and complete these steps:

1. Create a target entry for the host:
   a. Click **Jobs** → **Targets**, then click **New Host**.
   b. Enter the host name and select the operating system.
   c. Enter the name and the password of a user with administrative rights on the target system.
   d. Enter a variable called `WLP_WORKING_DIR`. The value of this variable is an existing directory on the target host. Job manager creates the `wlp` subdirectory in the `WLP_WORKING_DIR` directory and installs the Liberty profile there.

Figure B-2 shows where to enter the data to create this target.

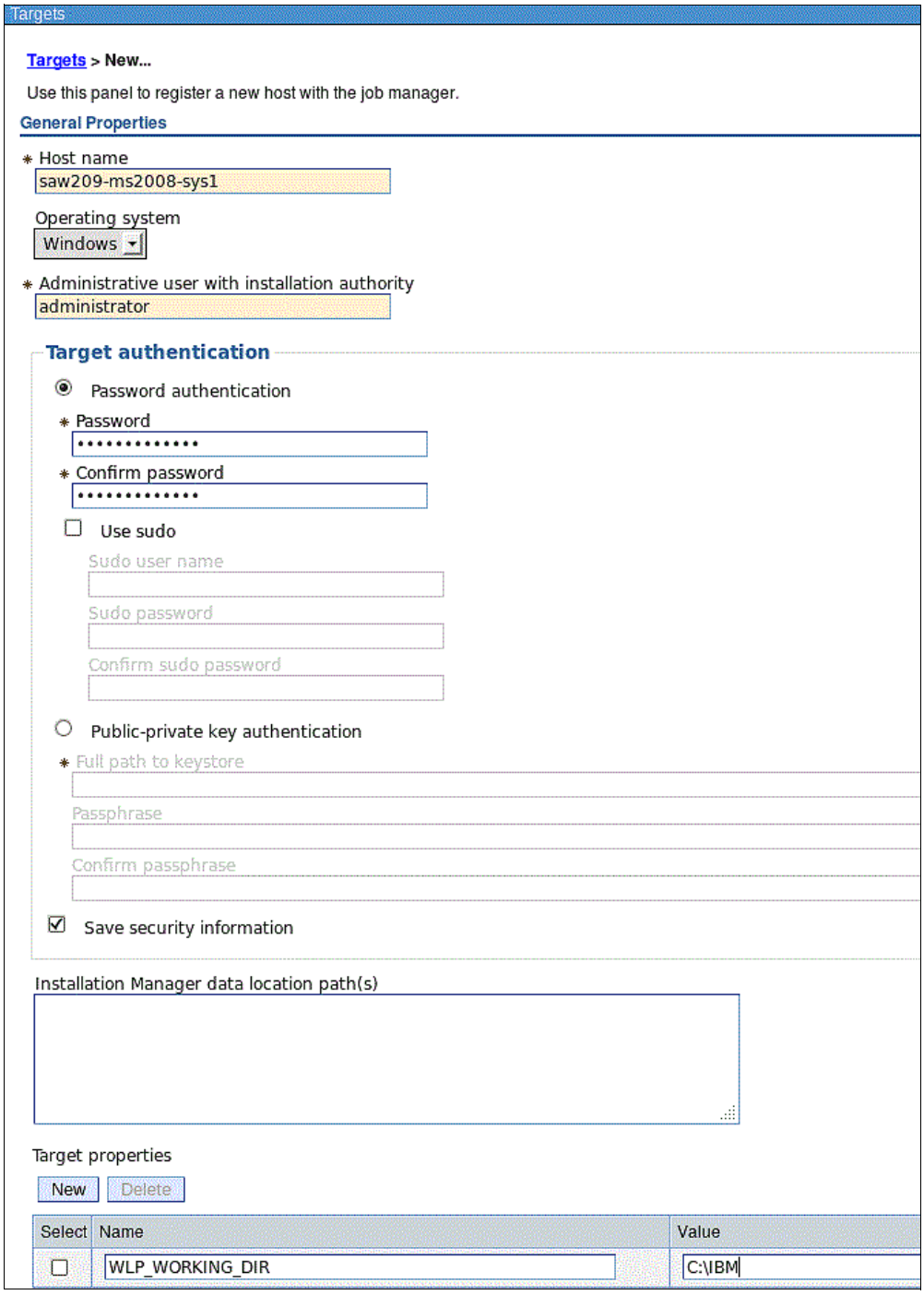


Figure B-2 Create job manager target

2. Deploy the prepared compressed file.

   Deploy the compressed file that you created in “Create a compressed file that contains the servers and applications” on page 592 to the target host. Click **Jobs** → **Submit**, and select **Install Liberty profile server resources** as shown in Figure B-3. Click **Next**.

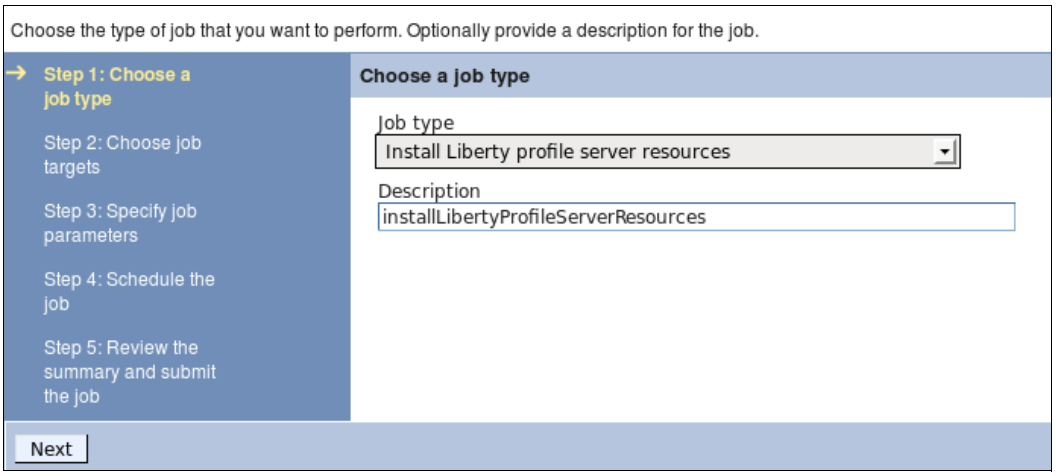


Figure B-3 Deploy Liberty profile, step 1

3. Select the target host and enter user name and password of a user with administrative rights as shown in Figure B-4. Click **Next**.

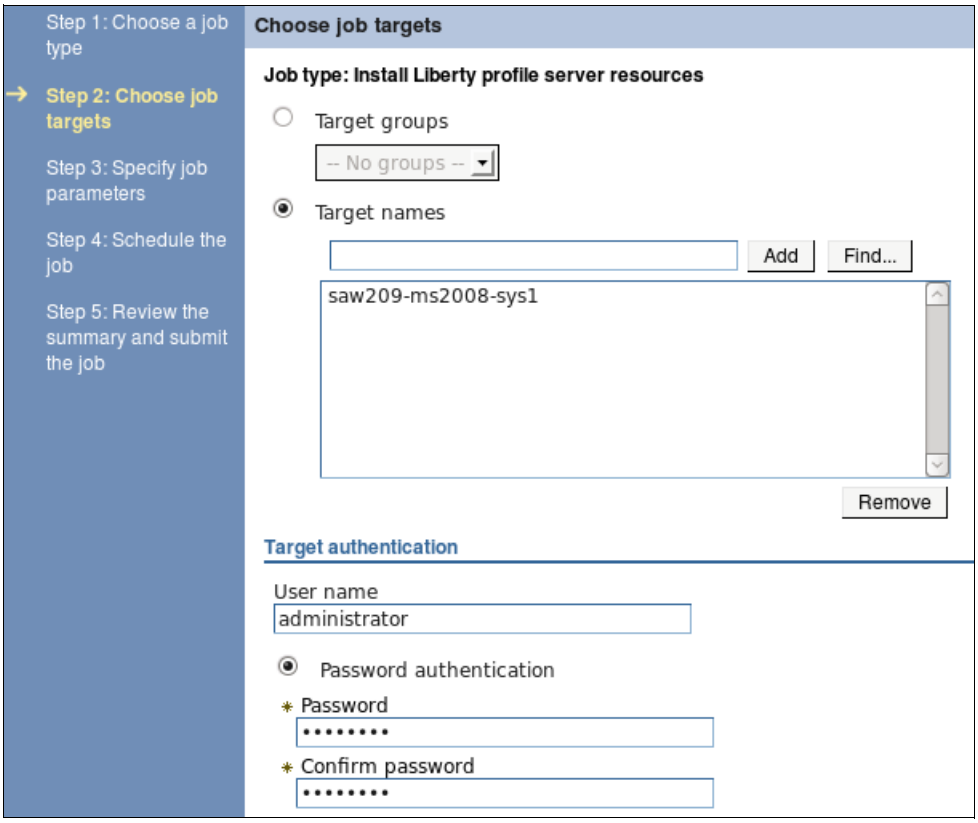


Figure B-4 Deploy Liberty profile, step 2

4. Specify the compressed file on the job manager host as shown in Figure B-5. You can also specify a URL where the file can be retrieved by using HTTP or FTP.

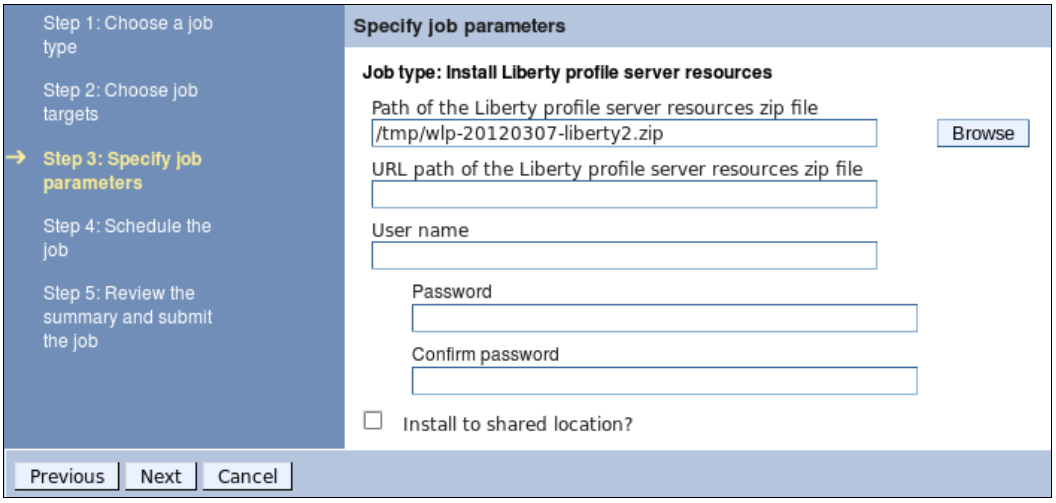


Figure B-5 Deploy Liberty profile, step 3

5. Click **Next** and accept the default values. The job is run immediately.
6. Click **Next** again, and review the settings. If the values are correct, click **Finish**.

   To view the finished job, click **Jobs** → **Status** as shown in Figure B-6.

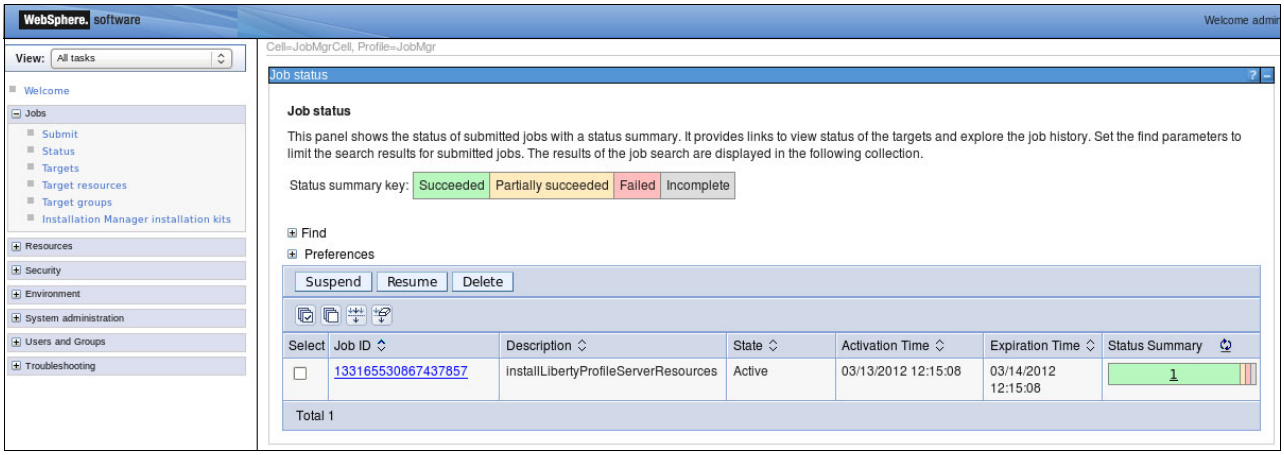


Figure B-6 Job status when deployment finishes successfully

After the deployment completes, check the directories on the target host. Figure B-7, shows where the run time, the server configuration, and the applications are deployed.

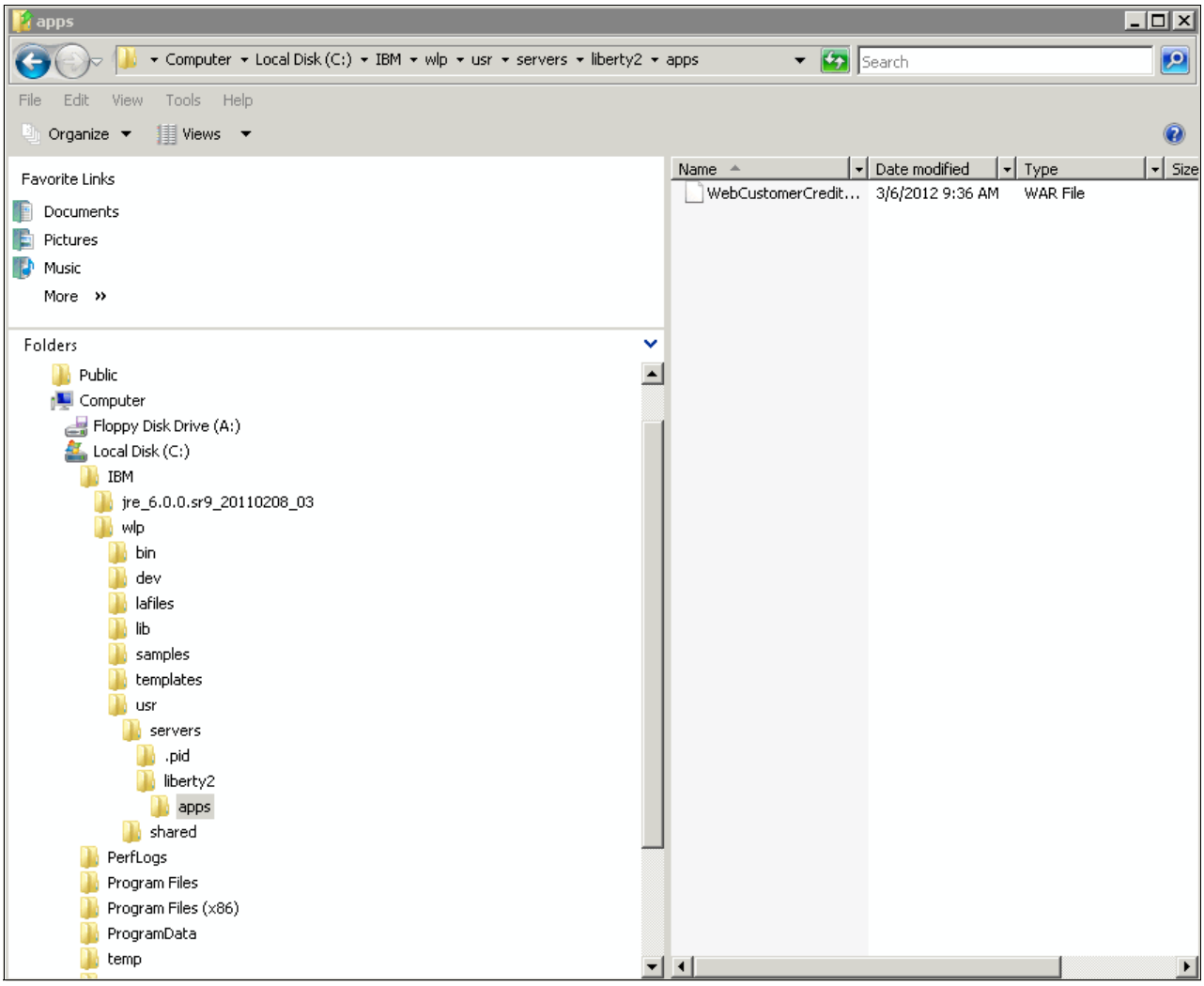


Figure B-7 Directories the job manager created on the target host

You can see the resources on the new host if you click **Jobs** → **Target resources** on the Job manager as displayed in Figure B-8.

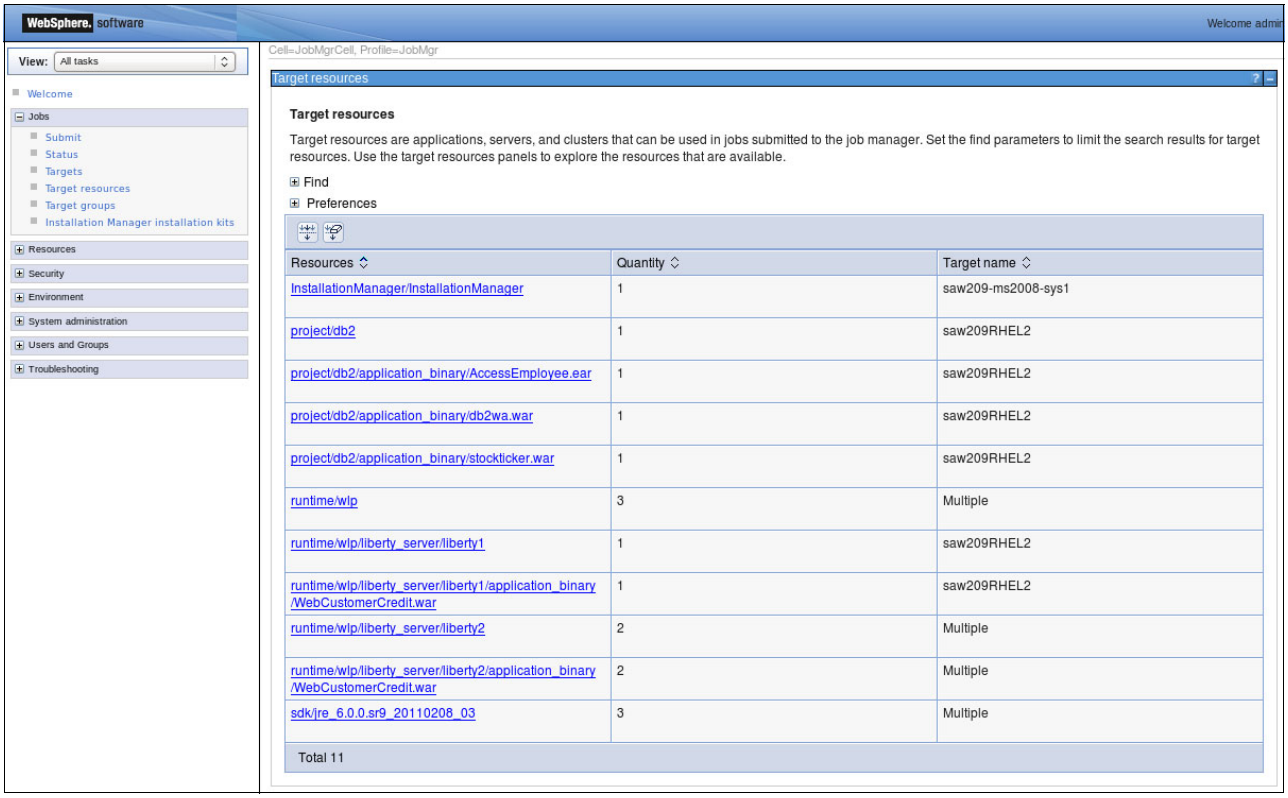


Figure B-8 Job manager resources

7. Start the Liberty profile server:

   a. Click **Jobs** → **Submit** and then **Start Liberty profile server** as shown in Figure B-9.

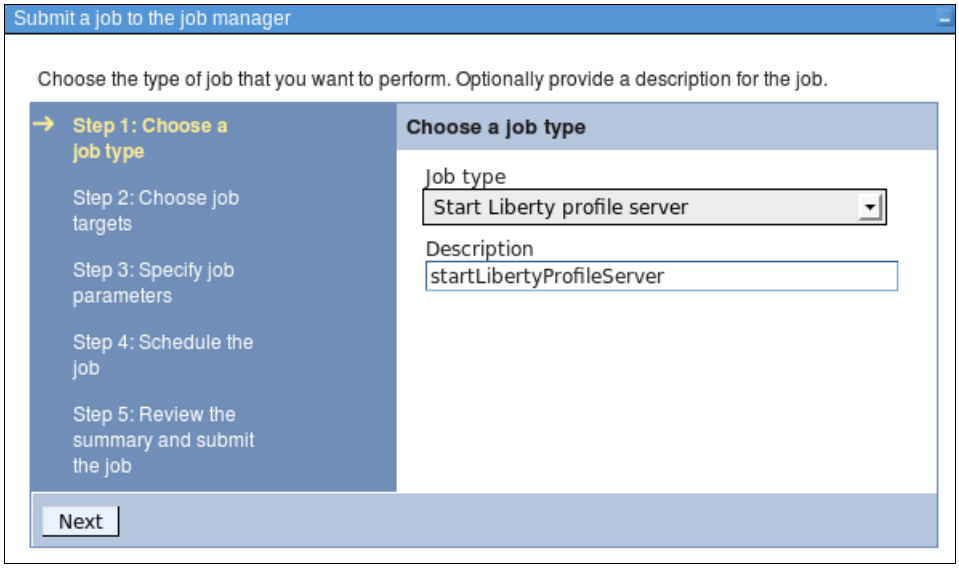


Figure B-9 Start Liberty profile server, step 1

   b. Select **Next** then select the target host as shown in Figure B-4 on page 594.

c. Click **Next** again and then click **Find** to find the target resource name of this server. Select the server name and press **OK**. The server to be started is displayed in the text box as shown in Figure B-10.

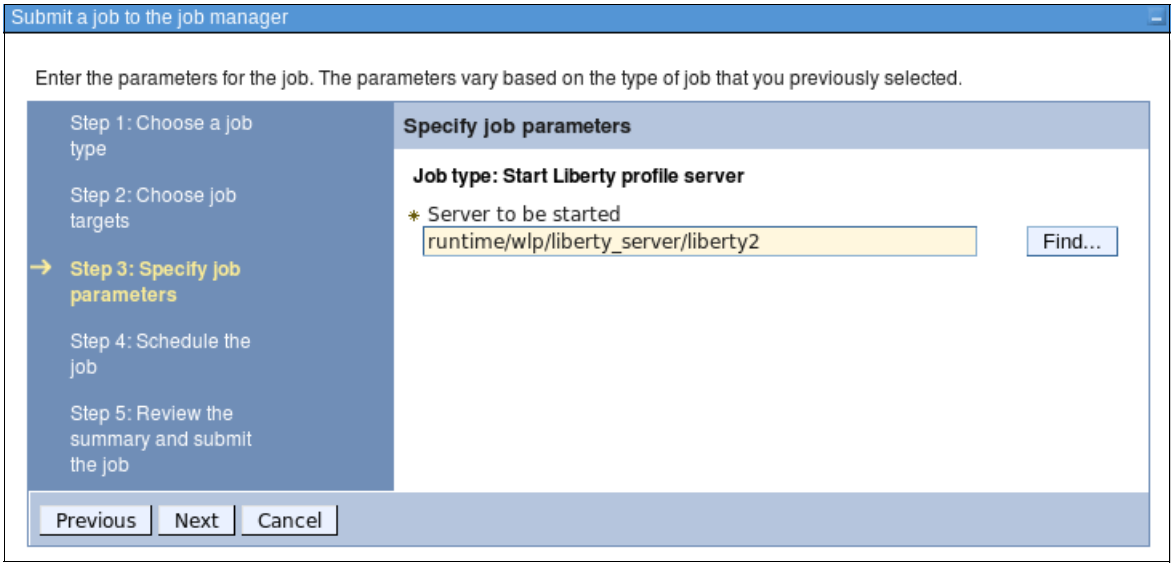


Figure B-10 Start Liberty profile server, step 2

8. Click **Next** twice, and then click **Finish**.

   To check the status of the job, click **Jobs** → **Status**. When the job finishes successfully, the status view looks like Figure B-11.

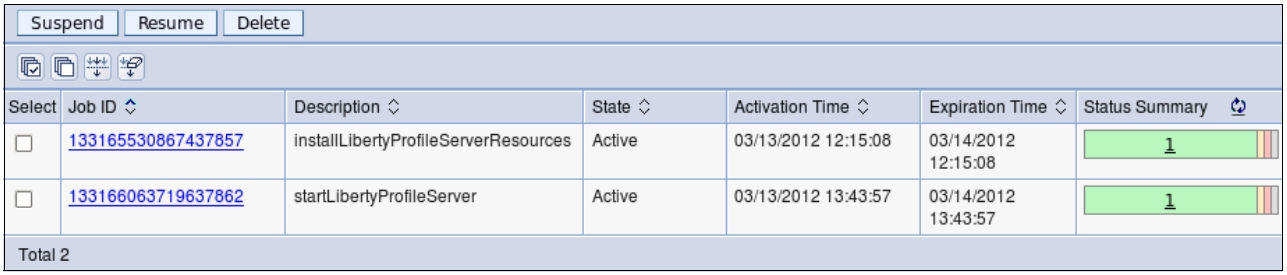


Figure B-11 Start Liberty profile server, step 3

You can check the `console.log` file on the target server `usr/servers/servername/logs` subdirectory of the Liberty profile directory, as shown in Figure B-12.

```
Launching liberty2 (wlp-1.0.0.20120307-0807/websphere-kernel_1.0.0) on IBM
J9 VM, version jvmwi3260sr9-20110203_74623
[AUDIT   ] CWWKE0001I: The server liberty2 has been launched.
[AUDIT   ] J2CA8000I: The jdbcDriver db2JDBCDriver is available.
[AUDIT   ] CWWKZ0058I: Monitoring dropins for applications.
[AUDIT   ] CWWKT0016I: Web application available (default_host):
http://saw209-ms2008-sys1.itso.ral.ibm.com:9080/WebCustomerCredit/*
[AUDIT   ] CWWKZ0001I: Application WebCustomerCredit started in 2.762
seconds.
[AUDIT   ] CWWKF0011I: The server liberty2 is ready to run a smarter planet.
```

Figure B-12 Liberty log file entries

Generating a common plug-in configuration for the Liberty profiles and deploying it to the HTTP server

Using job manager, you can create a common plug-in configuration for the Liberty profiles. This configuration enables the HTTP server to run load balancing and failover for the applications that are running on the Liberty profiles. This plug-in configuration is copied to the HTTP server.

C

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server at:

`ftp://www.redbooks.ibm.com/redbooks/SG248022`

Alternatively, you can go to the IBM Redbooks website at:

ibm.com`/redbooks`

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG248022.

Using the web material

The additional web material that accompanies this book includes the following files:

File name	*Description*
`SG248022.zip`	Compressed code samples

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material `.zip` file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *IBM Tivoli Composite Application Manager Family Installation, Configuration, and Basic Usage*, SG24-7151
- *Java Stand-alone Applications on z/OS, Volume I*, SG24-7177
- *Java Stand-alone Applications on z/OS Volume II*, SG24-7291
- *Patterns: SOA Foundation Service Creation Scenario*, SG24-7240
- *Rational Application Developer for WebSphere Software V8 Programming Guide*, SG24-7835
- *Solution Deployment Guide for IBM Tivoli Composite Application Manager for WebSphere*, SG24-7293
- *System Programmer's Guide to: Workload Manager*, SG24-6472
- *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257
- *WebSphere Application Server V7 Migration Guide*, REDP-4635
- *WebSphere Application Server V7.0 Security Guide*, SG24-7660
- *IBM WebSphere Application Server V8 Concepts, Planning, and Design Guide*, SG24-7957

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *Language Environment Programming Guide for 64-bit Virtual Addressing Mode*, SA22-7569
- *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- *z/OS MVS System Management Facilities (SMF)*, SA22-7630
- *z/OS UNIX System Services Planning Guide*, GA22-7800

Online resources

These websites are also relevant as further information sources:

- Websphere Application Server V8.5 Information Center

  http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp
- Java EE 6 specifications

  http://jcp.org/en/jsr/detail?id=316
- The WebSphere Application Server V8 Information Center

  http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp
- Tivoli Access Manager for e-business

  http://www.ibm.com/software/tivoli/products/access-mgr-e-bus/
- Tivoli Directory Server

  http://www.ibm.com/software/tivoli/products/directory-server/
- IBM Tivoli Workload Scheduler

  http://www.ibm.com/software/tivoli/products/scheduler/
- WebSphere MQ

  http://www.ibm.com/software/integration/wmq/
- IBM WebSphere Adapters

  http://www.ibm.com/software/integration/wbiadapters/
- Information about the DataPower appliances

  http://www.ibm.com/software/integration/datapower/
- IBM DB2 database software

  http://www.ibm.com/db2/
- DB2 pureScale product page

  http://www.ibm.com/software/data/db2/linux-unix-windows/editions-features-purescale.html
- What is DB2 pureScale? Going to extremes on scale and availability for DB2 article

  http://www.ibm.com/developerworks/data/library/dmmag/DBMag_2010_Issue1/DBMag_Issue109_pureScale/
- Integrating WebSphere Extreme Scale and WebSphere Application Server for Caching HTTP Sessions:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1112_shenoy/1112_shenoy.html?ca=drs-
- WebSphere Application Server—Express V8.5

  http://www.ibm.com/software/webservers/appserv/express/
- WebSphere Application Server V8.5

  http://www.ibm.com/software/webservers/appserv/was/
- WebSphere Application Server for Developers V8.5

  http://www.ibm.com/software/webservers/appserv/developer/index.html

- WebSphere Application Server Network Deployment V8.5

  http://www.ibm.com/software/webservers/appserv/was/network/
- WebSphere Application Server for z/OS V8.5

  http://www.ibm.com/software/webservers/appserv/zos_os390/
- WebSphere Application Server Community Edition

  http://www.ibm.com/software/webservers/appserv/community/
- WebSphere eXtreme Scale

  http://www.ibm.com/software/webservers/appserv/extremescale/
- Rational Application Developer for WebSphere Software V8

  http://www.ibm.com/software/awdtools/developer/application/
- WebSphere Portal Server - Enterprise portal software product page

  http://www.ibm.com/software/genservers/portal/server/index.html
- Integrating WebSphere Extreme Scale and WebSphere Application Server for Caching HTTP Sessions

  http://www.ibm.com/developerworks/websphere/library/techarticles/1112_shenoy/1112_shenoy.html?ca=drs-
- JSR 154,53 and 315 (Java Servlet 3.0 specification)

  http://jcp.org/en/jsr/detail?id=315
- JSR 318 (EJB 3.1 specification)

  http://jcp.org/en/jsr/detail?id=318
- Portlet 2.0 on the Java Community Process website

  http://jcp.org/en/jsr/detail?id=286
- JSR 289 SIP Servlet API 1.1 Specification

  http://jcp.org/en/jsr/detail?id=289
- RFC 3261 SIP Session Initiation Protocol

  http://www.ietf.org/rfc/rfc3261.txt
- Developing enterprise OSGi applications for WebSphere Application Server

  http://www.ibm.com/developerworks/websphere/techjournal/1007_robinson/1007_robinson.html
- IBM Education Assistance for online presentation about developing modular and dynamic OSGi applications

  http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.was_v8/was/8.0/ProgramingModel/WASV8_OSGi_part1/player.html
- Best practices for working with OSGi applications

  http://www.ibm.com/developerworks/websphere/techjournal/1007_charters/1007_charters.html
- Supported specifications for OSGi applications, visit the OSGi Service Platform Release 4

  http://www.osgi.org/Release4/HomePage
- Cloud computing

  http://www.ibm.com/cloud-computing/us/en

- Subscribe to the IBM Cloud YouTube channel for latest videos:

  http://www.youtube.com/user/IBMCloud
- Virtualization overview, YouTube video

  http://www.youtube.com/watch?v=IJM4GIfemT8
- On demand router hardware sizing requirements

  https://www.ibm.com/developerworks/wikis/display/xdoo/Best+practices+for+managing+the+on+demand+router?showComments=false>
- Design for Scalability: An Update:

  http://www.ibm.com/developerworks/websphere/library/techarticles/hipods/scalability.html
- IBM Systems Workload Estimator page at:

  http://www.ibm.com/systems/support/tools/estimator/index.html
- Information on rPerf

  http://www.ibm.com/systems/power/hardware/notices/rperf.html
- Information on SPEC

  http://www.spec.org/benchmarks.html
- Information on TPC

  http://www.tpc.org/information/benchmarks.asp
- IBM SmartCloud Enterprise as a way to access secure WebSphere environments:

  http://www.ibm.com/services/us/igs/cloud-development/
- Amazon Elastic Compute Cloud provides WebSphere Application Server images:

  http://aws.amazon.com/ec2/
- IBM Workload Deployer:

  http://www.ibm.com/software/webservers/workload-deployer
- Using virtual image templates to deploy WebSphere Application Server:

  http://www.ibm.com/developerworks/websphere/techjournal/0705_willenborg/0705_willenborg.html
- IBM white paper *WebSphere for z/OS -- Heterogeneous Cells*:

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100644

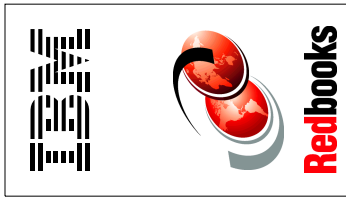
Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



WebSphere Application Server V8.5 Concepts, Planning, and Design Guide

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

WebSphere Application Server V8.5 Concepts, Planning, and Design Guide

Highlights end-to-end planning for WebSphere implementations

Defines WebSphere concepts and best practices

Addresses distributed and z/OS platforms

This IBM Redbooks publication provides information about the concepts, planning, and design of IBM WebSphere Application Server V8.5 environments. The target audience of this book is IT architects and consultants who want more information about the planning and design of application-serving environments, from small to large, and complex implementations.

This book addresses the packaging and features in WebSphere Application Server V8.5, and highlights the most common implementation topologies. It provides information about planning for specific tasks and components that conform to the WebSphere Application Server environment.

Also in this book are planning guidelines for Websphere Application Server V8.5 and Websphere Application Server Network Deployment V8.5 on distributed platforms. It also includes guidelines for WebSphere Application Server for IBM z/OS V8.5. This book contains information about migration considerations when moving from previous releases.

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

SG24-8022-00 ISBN 0738436933