IBM® WebSphere®

IBM WebSphere Application Server V8 Concepts, Planning, and Design Guide

Includes details about end-to-end planning for WebSphere implementations

Defines WebSphere concepts and preferred practices

Addresses distributed and z/OS platforms

Margaret Ticknor
Alan Corcoran
Balazs Csepregi-Horvath
Addison Goering
José Pablo Hernandez
Julien Limodin
Sergio Straessli Pinto

Redbooks

IBM

International Technical Support Organization

IBM WebSphere Application Server V8 Concepts, Planning, and Design Guide

August 2011

Note: Before using this information and the product it supports, read the information in "Notices" on page xiii.

First Edition (August 2011)

This edition applies to Version 8.0 of IBM WebSphere Application Server.

Contents

Notices

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

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

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

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

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

Any references in this information to non-IBM 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.

xiii

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®	IMS™	Rational Rose®
alphaWorks®	Informix®	Rational Team Concert™
BladeCenter®	iSeries®	Rational®
Build Forge®	Jazz™	Redbooks®
CICS®	Language Environment®	Redbooks (logo) ®
ClearCase MultiSite®	Lotus®	RequisitePro®
ClearCase®	MVS™	Resource Measurement Facility™
ClearQuest®	Parallel Sysplex®	RMF™
CloudBurst™	Passport Advantage®	System i®
Cognos®	PowerHA™	System z®
DataPower®	POWER®	Tivoli®
DB2®	PR/SM™	VTAM®
developerWorks®	Processor Resource/Systems	WebSphere®
Domino®	Manager™	z/OS®
i5/OS®	pureScale™	zSeries®
IBM®	pureXML®	
ILOG®	RACF®	

The following terms are trademarks of other companies:

Intel, Itanium, Pentium, 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.

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.

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

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

Preface

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

This book addresses the packaging and features in WebSphere Application Server V8 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 and WebSphere Application Server Network Deployment V8 on distributed platforms and for WebSphere Application Server for z/OS® V8. 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 (ITSO), Raleigh Center.

Margaret Ticknor is an IT Specialist at the ITSO Center in Raleigh, NC. She manages WebSphere projects for the development of Redbooks publications. Prior to joining the ITSO in 1997, Margaret worked in Endicott, supporting internal VM customers. Margaret attended the Computer Science program at State University of New York at Binghamton.

Alan Corcoran is an IT Specialist in Newport, Rhode Island, working for IBM in WebSphere Education since 1996. He has 15 years of experience developing and delivering education courses on WebSphere products. His focus is WebSphere Application Server administration and WebSphere Voice Response administration. Alan holds a bachelor degree in computer science and urban planning from the State University of New York at Albany.

Balazs Csepregi-Horvath is a Customer Engineer working for IBM Global Services in Hungary. He also acts as a Project Manager in IBM Hungary. He has ten years of expertise in supporting IBM WebSphere, Rational® and Information Management products. He is an authorized WebSphere and certified Portal instructor. Balazs holds bachelor degrees in heavy current automation and information technology, both from Kando Kalman Technical College in Budapest, Hungary.

Addison Goering is a Certified IT Specialist working for IBM since 1998 with WebSphere Education. He has over 12 years experience in developing and delivering administrative courses for the entire WebSphere portfolio, including WebSphere Application Server, WebSphere Enterprise Service Bus, and WebSphere Process Server. Addison's area of expertise is in course design and development.

José Pablo Hernandez is a WebSphere Software Specialist working since 2007 for GBM Costa Rica, an IBM Business Partner. His areas of expertise include implementation and support for WebSphere Application Server projects. José is also an IBM Certified Advanced

System Administrator. He holds a bachelor degree in system engineering from Universidad Latina de Costa Rica.

Julien Limodin is an IT Specialist working for IBM France since 2006. He is currently working on customer performance benchmarks, especially on WebSphere and IBM Power, in the Products & Solutions Support Center in Montpellier, and is part of the EMEA benchmark center. His main areas of expertise are design, implementation, performance tuning, and high availability of the WebSphere Application Server environment. Julien holds a Master of Science degree in information technology and management from a French engineering school.

Sergio Straessli Pinto is an IT Specialist working for Integrated Technology Delivery, Server Systems Operations, in Brazil. He has worked for IBM for 15 years. His areas of expertise include support in WebSphere MQ and WebSphere Message Broker on distributed and IBM z/OS environments and support in WebSphere Application Server for z/OS. He has also worked on developing software using COBOL/IBM CICS® command level and Visual Basic using Oracle database. He holds a bachelor degree in business administration and accounting from Instituto Católico de Minas Gerais, Brazil.

The team from left to right: Sergio Pinto, Balazs Csepregi-Horvath, Alan Corcoran, Margaret Ticknor, Julien Limodin, and José Pablo Hernandez; not shown: Addison Goering

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

Tamikia Barrow
Linda Robinson
Carla Sadtler
Stephen Smith
Debbie Willmschen
ITSO, Raleigh Center

Keys Botzum
IBM Bethesda, US

Michael Cheng
Jim Knutson
Bill O'Donnell
IBM Austin, US

Eric Covener
IBM Raleigh, US

Dana Duffield
Jenifer Servais
IBM Rochester, US

Frederic Dutheil
IBM France

Leho Nigul
Joshua Robbins
Patrick Tiu
IBM Toronto, Canada

The team who created *WebSphere Application Server V6.1: Planning and Design*, SG24-7305

The team who created *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708

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

IBM WebSphere is the leading software platform for On Demand Business and service-oriented architecture (SOA) for your enterprise. Providing comprehensive leadership, WebSphere is evolving to meet the demands of companies faced with challenging business requirements. Their requirements include performing the following tasks:

► Increasing operational efficiencies
► Strengthening client loyalty
► Integrating disparate platforms and systems
► Responding with speed to any market opportunity or external threat
► Adapting existing business to change quickly
► Creating agile environments

With IBM WebSphere, you can build business-critical enterprise applications and solutions and combine them with innovative new functions. WebSphere includes and supports a range of products that helps you develop and serve your business applications. These products make it easier for clients to build, deploy, and manage dynamic websites and other more complex solutions productively and effectively.

This chapter introduces WebSphere Application Server V8.0 for distributed platforms and z/OS and highlights other IBM software products related to WebSphere Application Server. This chapter includes the following sections:

► Java Platform, Enterprise Edition
► Overview of WebSphere Application Server
► Packaging
► Supported hardware, platforms, and software
► Related products

1

1.1 Java Platform, Enterprise Edition

WebSphere is the IBM brand of software products that is designed to work together to help deliver dynamic On Demand Business quickly. It provides solutions for connecting people, systems, applications, and services with internal and external resources. WebSphere is infrastructure software, or middleware, for dynamic On Demand Business and for enabling SOA for your enterprise. It delivers a proven, secure, robust, and reliable software portfolio that provides an excellent return on investment.

Java is the technology that powers WebSphere 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). This platform contrasts with the Java Platform, Standard Edition (Java SE), with which most clients are familiar.

Java SE supported the development of client-side applications with rich graphical user interfaces (GUIs). The Java EE platform, which is built on top of the Java SE platform, provides specifications for developing multitier enterprise applications with Java. It consists of application technologies for defining business logic and accessing enterprise resources such as databases, enterprise resource planning (ERP) systems, messaging systems, internal and external business services, and email servers.

> **Java Platform name changes:** Previously Java EE was named *Java 2 Platform, Enterprise Edition (J2EE)*, and Java SE was named *Java 2 Platform, Standard Edition (J2SE)*. Now, J2EE 1.5 has been renamed to *Java EE 5*. J2SE 6 has been renamed to *Java SE 6*. This book provides information about these standards with their new names.

Java EE provides the following benefits:

► An architecture-driven approach that allows application development helps reduce maintenance costs and allows for construction of an information technology (IT) infrastructure that can grow to accommodate new services.

► Application development 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.

► With industry standard technologies, clients can choose among platforms, development tools, and middleware to power their applications.

► Platform independence gives flexibility to create an application one time and to run it on multiple platforms, providing true portability to enterprise applications.

► With embedded support for Internet and web technologies, applications can bring services and content to a wider range of customers, suppliers, and others, without creating the need for proprietary integration.

Java EE 6, the latest release of the Java EE platform, represents a significant evolution in the Java enterprise programming model. It improves the application developer experience and productivity with following new features:

► Latest specifications

 – Enterprise JavaBeans (EJB) 3.1
 – Java API for XML Web Services (JAX-WS) 2.2
 – JavaServer Pages (JSP) 2.1

- – Servlet 3.0
- – JavaServer Faces (JSF) 2.0
- – Java Persistence API (JPA) 2.0
- – Java API for RESTful Web Services (JAX-RS) 1.1
- ▶ Java Development Kit (JDK) 6.0
- ▶ Use of progressive disclosure
- ▶ Annotations and injection support to reduce complexity
- ▶ EJB development as Plain Old Java Objects (POJOs)
- ▶ JPA to allow creation of simpler entities using an annotated POJO model

Another exciting opportunity for IT is web services. Web services allow for the definition of functions or services within an enterprise that can be accessed by using industry standard protocols (such as HTTP and XML) that are already in use today. These protocols allow for easy integration of both intrabusiness and interbusiness 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 enable businesses to address changing opportunities.

For more information about the Java EE 6 specifications, see Java EE at a Glance at:

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

1.2 Overview of WebSphere Application Server

WebSphere Application Server is the IBM runtime environment for Java language-based applications. This section gives an overview of the options and functionalities that WebSphere Application Server V8.0 offers:

- ▶ Application server purpose
- ▶ Evolving Java application development standards
- ▶ Enhanced management
- ▶ Broader integration
- ▶ Advanced tooling and extensions

1.2.1 Application server purpose

An application server provides the infrastructure for executing applications that run your business. It insulates the infrastructure from hardware, operating system, and the network (Figure 1-1). An application server serves as a platform to develop and deploy your web services and EJBs. It also serves as a transaction and messaging engine. An application server also delivers business logic to users on various client devices.
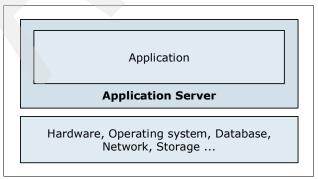
Figure 1-1   Basic presentation of an application server and its environment

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.

WebSphere Application Server provides the environment to run your solutions and to integrate them with every platform and system, as business application services conforming to the SOA reference architecture (Figure 1-2).
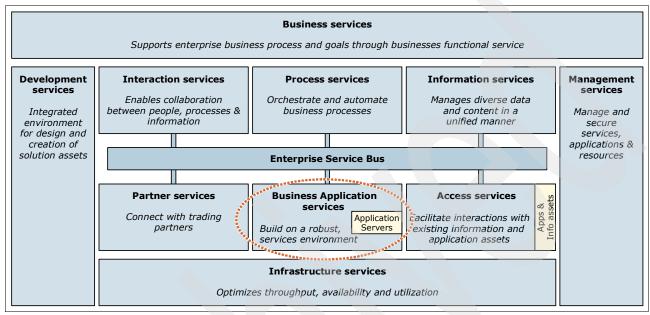
Figure 1-2   Position of business application services in an SOA reference architecture

WebSphere Application Server is a key SOA building block. From an SOA perspective, with WebSphere Application Server, you can perform the following functions:

► 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 the application server that is required to run specific applications, it also serves as the base for other WebSphere products and many other IBM software products. Such products include IBM WebSphere Enterprise Service Bus, IBM Business Process Manager, and WebSphere Portal.

As business needs evolve, new technology standards become available. The reverse is also true. Since 1998, WebSphere has grown and adapted itself to new technologies and to new standards to provide an innovative and cutting-edge environment so that you can design fully integrated solutions and run your business applications.

1.2.2 Evolving Java application development standards

WebSphere Application Server V8.0 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 invoke 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.

For universal data access and persistence, WebSphere Application Server supports the following specifications:

► Java Database Connectivity (JDBC) API 4.0

By using the JDBC API, you can connect to any type of data source.

► JPA 2.0

JPA delivers simplified programming models and a standard persistence API for building reusable persistent objects.

► Service Data Objects (SDO)

With SDO, application programmers can uniformly access and manipulate data from heterogeneous data sources as collections of tree-structured or graph-structured objects. WebSphere Application Server V8 extends the application server to support these tasks:

– Java Specification Requests (JSR) 286 (Portlet 2.0) compliant portlets

– Session Initiation Protocol (SIP) applications conforming to the JSR 116 specification and SIP Servlet specification 1.1 (JSR 289)

– Java Servlet 3.0 (JSR 315), JavaServer Pages 2.2 (JSR 245) and JavaServer Faces 2.0 (JSR 314) specifications for web applications

The enhanced support of WebSphere Application Server for application development standards delivers maximum flexibility and significantly improves developer productivity.

1.2.3 Enhanced management

WebSphere Application Server has several packaging options to meet your demands. With these packages, you can create basic scenarios with single application server environments. Furthermore, you can extend your environment to include multiple application servers that are administered from a single point of control, the deployment manager. These application servers can be clustered to provide scalable and high available environments.

Management features

WebSphere Application Server V8 contains the following effective management features in its packages:

► Flexible management tools:

With these tools, you can build advanced and large-scale topologies and reduce management and maintenance complexity.

– Administrative agent

Centralize node administration and manage multiple stand-alone servers from a central point.

– Job manager

Transfer management jobs, such as deploying, starting and stopping applications, and distributing files. You can include stand-alone and Network Deployment servers in such topologies.

► IBM Installation Manager

This component provides local and centralized product lifecycle management. It installs the desired level of service in one pass.

- ► Centralized installation manager

  This component provides the capability to perform centralized installations and apply maintenance to remote endpoints. It is available from the Job Manager and Deployment manager in distributed and z/OS environments.

- ► High Performance Extensible Logging (HPEL)

  This 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 for administrators to manage logging resources and work with log and trace content.

- ► Node Management

  Recovery or moving nodes is achieved with the **addNode** command and the **-asExistingNode** option, improving productivity and minimizing downtime.

- ► Business-level application

  This component simplifies the management and administration tasks of multicomponent applications. You can use it to group and manage Java EE and other related artifacts under a single application definition.

- ► Monitored directory support

  Monitored directory support accelerates edit, compile, and dedug tasks during the development life cycle. Through monitored directory-based application installation, update, and uninstallation of Java EE applications.

- ► Improved Console Command Assistant

  This component provides easier security configuration and database connectivity, wizards, and a stand-alone thin administration client that enable efficient management of the deployment environment.

- ► Enhanced administrative scripting for **wsadmin**

  An extended sample script library accelerates automation implementations. It also includes enhanced AdminTask commands.

- ► Consolidated administration feature for IBM WebSphere DataPower®

  With this component, you can manage and integrate your WebSphere DataPower appliances into your environment.

Runtime provisioning

With the runtime provisioning mechanism in WebSphere Application Server, the application server run time can select only the necessary functions for memory and space dynamically while running applications. Starting only the necessary components for an application reduces the server footprint and startup time.

Security management and auditing

WebSphere Application Server V8.0 adds value to installations by providing the following security management and auditing improvements:

- ► With the Auditing service provider setting, the WebSphere application server administrator can configure the behavior of the audit files when they reach maximum capacity.

- ► 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 be security configuration changes, key and certificate management, access control policy changes, bus and other system resources management, and so on. With this feature, you can hold administrative users accountable for configuration and runtime changes.

- With the DMZ Secure Proxy, a proxy server hardened for demilitarized zone (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 running on the cell.

1.2.4 Broader integration

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

Web services

WebSphere Application Server V8 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

To centrally define various quality of service policies, you might want to apply to web services that are already deployed. WebSphere Application Server V8 provides the following web service policy sets:

- Application policy sets

  These sets are used for business-related assertions, such as business operations that are defined in the Web Services Description Language (WSDL) file.

- System policy sets

  These sets are used for non-business-related system messages, such as messages that are defined in specifications that apply quality of service policies. Examples include security token (RequestSecurityToken) messages that are defined in WS-Trust or the creation of sequence messages that are defined in WS-Reliable Messaging metadata.

Rather than define individual policies and apply them on a web service base, policy sets can be applied one time to all web service applications to which they are applicable. This way ensures a uniform quality of service for a given type of web service. WebSphere Service Registry and Repository, which is an additional WebSphere Software product, can discover

WebSphere Application Server V8 JAX-WS policy sets and existing associations. It can also represent them as policy attachments.

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, and 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 consume these services as remote portlets.

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

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.

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. It provides a stand-alone client runtime environment for your client applications and enables your client to run applications in a Java EE environment that is compatible with EJB.

The Application Client for WebSphere Application Server V8 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).

Web server support

WebSphere Application Server can work with a web server (such as the IBM HTTP Server included in WebSphere Application Server packages) to route requests from browsers to the applications that run in WebSphere Application Server. A web server plug-in for WebSphere is provided for installation with supported web servers. This plug-in directs requests to the appropriate application server and performs workload balancing and failover among servers in a cluster.

1.2.5 Advanced tooling and extensions

This section provides information about WebSphere Application Server tooling and extension enhancements.

Application development and deployment tools

WebSphere Application Server V8.0 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 V8.0 servers.
► Debug applications on WebSphere Application Server V8.

Installation Manager

With WebSphere Application Server V8, the IBM Installation Manager replaces the Installshield MultiPlatform (ISMP) and the Update Installer. It also replaces the functions previously provided by the Installation Factory. The Installation Manager is a single installation tool that loads and installs product components from a structured collection of files known as a *repository*.

Centralized installation manager

The centralized installation manager (CIM) is used to consolidate and simplify the steps that are required to perform installations and to apply maintenance on systems. CIM can be used to install Installation Manager instances, update Installation Manager with a repository, and manage Installation Manager offerings using the administrative console or the `wsadmin` tool.

WebSphere Customization Toolbox

The WebSphere Customization Toolbox for WebSphere Application Server V8 includes tools for customizing various parts of your WebSphere Application Server environment. You can launch the Web Server Plug-ins Configuration Tool, Profile Management Tool, and the z/OS Migration Management Tool.

WebSphere Application Server feature packs

WebSphere Application Server feature packs simplify the adoption of new technology standards and make them available before the release of a new version of WebSphere Application Server. This strategy started with WebSphere Application Server V6.1 and continued with WebSphere Application Server V8.0.

The following previously available feature packs have been integrated into WebSphere Application Server V8, as part of the base:

► WebSphere application Server Feature Pack for Communications Enabled Applications
► WebSphere application Server Feature Pack for Modern Batch
► WebSphere application Server Feature Pack for OSGi Applications and JPA 2.0
► WebSphere application Server Feature Pack for SCA
► WebSphere application Server Feature Pack for XML
► WebSphere application Server Feature Pack for EJB 3.0
► WebSphere application Server Feature Pack for Web Services

Figure 1-3 illustrates the feature pack structure for WebSphere Application Server V8.
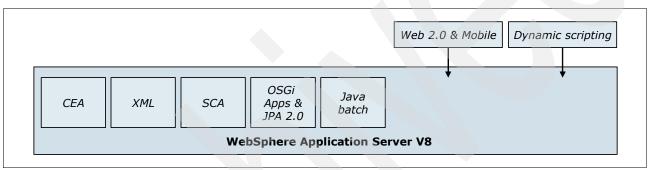
Figure 1-3   Feature pack structure for WebSphere Application Server V8

1.3 Packaging

Because different application scenarios require different levels of application server capabilities, WebSphere Application Server is available in multiple packaging options. 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
► WebSphere Application Server V8
► WebSphere Application Server for Developers V8
► WebSphere Application Server Network Deployment V8
► WebSphere Application Server for z/OS V8

Figure 1-4 summarizes the main components included in each WebSphere Application Server package.
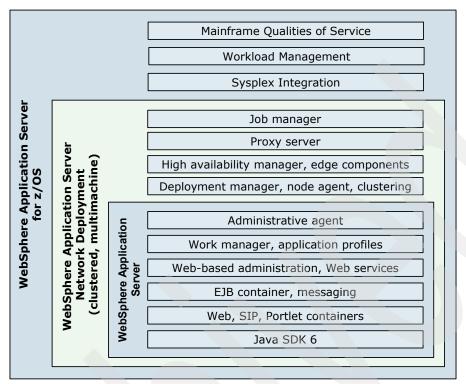
Figure 1-4   Packaging structure of WebSphere Application Server V8

1.3.1 WebSphere Application Server - Express V8

The WebSphere Application Server - Express V8 package is geared to those uses who need to get started quickly with a strong and affordable application server based on standards. It is targeted to medium-sized businesses or departments of a large corporation and is focused on providing ease of use and ease of application development. It contains portions of key Java EE 6, EJB 3.1, and web services support. However, it is limited to a single-server environment and a maximum of 480 Processor Value Units (PVUs) per server or virtualized partition, for licensing purposes.

The Express package also includes feature pack support. This package does not provide clustering and high availability features.

For more information about WebSphere Application Server - Express V8, see the product page at:

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

1.3.2 WebSphere Application Server V8

The WebSphere Application Server V8 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. It is available for both 32-bit and 64-bit platforms. This package is ideal for lightweight application solutions where cost

and simplicity are key. This package is also referred to as the *WebSphere Application Server V8.0 Base package*.

For more information about WebSphere Application Server V8.0, see the product page at:

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

1.3.3 WebSphere Application Server for Developers V8

The WebSphere Application Server for Developers package is functionally equivalent to the WebSphere Application Server 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.

For more information about WebSphere Application Server for Developers V8, see the product page at:

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

1.3.4 WebSphere Application Server Network Deployment V8

WebSphere Application Server Network Deployment (ND) V8 provides the capabilities to develop more enhanced server infrastructures. It extends the base package of WebSphere Application Server and includes the following features:

► Clustering capabilities
► Edge Components
► Dynamic scalability
► High availability
► Advanced centralized management features for distributed configurations

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

WebSphere Application Server Network Deployment 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. The usage of 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, see the product page at:

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

1.3.5 WebSphere Application Server for z/OS V8

IBM WebSphere Application Server for z/OS V8 is a full-function version of WebSphere Application Server Network Deployment. Although it offers all the options and functions common to WebSphere Application Server V8 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 with Workload Management in times of unpredictable peaks

► Uses z/OS functionality for billing based on used resources or transactions

► Uses one central security repository, including Java role-based security

► 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

For more information about WebSphere Application Server for z/OS V8.0, see Chapter 14, "WebSphere Application Server for z/OS" on page 385, or the product page at:

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

1.3.6 Packaging summary

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

Table 1-1   WebSphere Application Server V8.0 packaging

Features	Express	Base	Network Deployment	z/OS
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
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
Advanced clustering	No	No	Yes	Yes
Integration with IBM Rational Application Developer Assembly and Deploy	Yes	Yes	Yes	Yes
Job manager and deployment manager	No	No	Yes	Yes
Rapid Java Development and Deployment Kit 6.0	Yes	Yes	Yes	Yes
Runtime provisioning	Yes	Yes	Yes	Yes
Large-scale transaction support	No	No	Yes	Yes
Dynamic caching	Yes	Yes	Yes	Yes
Reporting and charge back: Granular reporting on resource consumption	No	No	No	Yes

Features	Express	Base	Network Deployment	z/OS
WebSphere Application Server feature packs	Yes	Yes	Yes	Yes
Administrative agent	Yes	Yes	Yes	Yes
Edge Components	No	No	Yes	Yes

1.4 Supported hardware, platforms, and software

This section provides details about the hardware, platforms, and software versions that WebSphere Application Server V8.0 supports at the time this book was written. For the most up-to-date operating system levels and hardware requirements, see WebSphere Application Server detailed system requirements at:

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

1.4.1 Hardware

The following hardware is supported for WebSphere Application Server V8:

► IBM POWER® family of processors
► IBM System z processors
► IBM System i® (iSeries®)
► Intel EM64T or AMD Opteron
► Intel Itanium 2 processor
► Intel Itanium 9300 series
► Intel Pentium IV or later processor at 1.2 GHz or faster
► Intel x86-64 or AMD x86-64 processor at 1.2 GHz or faster
► SPARC workstations

1.4.2 Operating systems

Table 1-2 shows the supported operating systems and their versions for WebSphere Application Server V8.

Table 1-2   Supported operating systems and versions

Operating systems	Versions
Microsoft Windows	Supported with a 32-bit WebSphere Application Server: ► Windows 7 Ultimate ► Windows 7 Enterprise ► Windows 7 Professional ► Windows XP SP3 Professional ► Windows Vista Ultimate ► Windows Vista Enterprise ► Windows Vista Business ► Windows Server 2003 R2 SP2 Standard Edition x86-32 ► Windows Server 2003 R2 SP2 Datacenter Edition x86-32 ► Windows Server 2003 R2 SP2 Enterprise Edition x86-32

Operating systems	Versions
Microsoft Windows (continued)	Supported with a 32-bit and 64-bit WebSphere Application Server: ► Windows Server 2003 SP2 Enterprise Edition x86-64 ► Windows Server 2003 R2 SP2 Enterprise Edition x86-64 ► Windows Server 2003 SP2 Standard Edition x86-64 ► Windows Server 2003 R2 SP2 Standard Edition x86-64 ► Windows Server 2003 SP2 Datacenter Edition x86-64 ► Windows Server 2003 R2 SP2 Datacenter Edition x86-64 ► Windows Server 2008 Standard Edition x86-64 ► Windows Server 2008 R2 Standard Edition x86-64 ► Windows Server 2008 Enterprise Edition x86-64 ► Windows Server 2008 R2 Enterprise Edition x86-64 ► Windows Server 2008 Datacenter Edition x86-64 ► Windows Server 2008 R2 Datacenter Edition x86-64
IBM AIX®	Supported with a 32-bit and 64-bit WebSphere Application Server: ► AIX 6.1 ► AIX 7.1
Sun Solaris on Sparc	Supported with a 32-bit and 64-bit WebSphere Application Server: ► Solaris 10 SPARC
Sun Solaris on x86-64	Supported with a 64-bit WebSphere Application Server: ► Solaris 10 x86-64
HP-UX on Itanium	Supported with a 64-bit WebSphere Application Server: ► HP-UX 11iv2.3 ► HP-UX 11iv31
Linux on x86	Supported with a 32-bit WebSphere Application Server: ► Red Hat Enterprise Linux, Version 5 with Update 6 ► SUSE Linux Enterprise Server (SLES) 11.0 ► SUSE Linux Enterprise Server, Version 10 with SP3 ► Asianux Linux 3.0
Linux on x86-64 and POWER	Supported with a 32-bit and 64-bit WebSphere Application Server: ► Red Hat Enterprise Linux (RHEL) 5.0 Update 6 Advanced Platform ► Red Hat Enterprise Linux 6.0 AS/ES ► SUSE Linux Enterprise Server 11.0 ► SUSE Enterprise Server, Version 10 SP3
Linux on System z and zSeries®	Supported with a 31-bit and 64-bit WebSphere Application Server: ► Red Hat Enterprise Linux 5.0 Update 6 Advanced Platform ► Red Hat Enterprise Linux 6.0 AS/ES ► SUSE Linux Enterprise Server 10.0 SP3 ► SUSE Linux Enterprise Server 11.0
IBM z/OS (Supported for WebSphere Application Server V8.0 for z/OS)	Supported with a 31-bit and 64-bit WebSphere Application Server: ► z/OS V1.10 ► z/OS V1.11 ► z/OS V1.12 ► z/OS.e ► z/OS.e V1.7 ► z/OS.e V1.8
IBM i	► IBM i V6R1 ► IBM i V7R1

1.4.3 Web servers

The following web servers are supported for WebSphere Application Server V8:

- ► Apache HTTP Server 2.2
- ► IBM HTTP Server V8.0
- ► IBM HTTP Server for i5/OS® any
- ► IBM Lotus® Domino® Enterprise Server 8.0
- ► Microsoft Internet Information Services (IIS) 6.0
- ► Microsoft Internet Information Services 7.0
- ► Microsoft Internet Information Services 7.5
- ► Sun Java System Web Server 6.1 SP12
- ► Sun Java System Web Server 7.0 Update 1

1.4.4 Database servers

Table 1-3 lists the database servers that WebSphere Application Server V8.0 supports.

Table 1-3   Supported database servers and versions

Databases	Versions
IBM Derby SDK	IBM Derby SDK 10.5
IBM DB2® [a]	DB2 Enterprise Server Edition 9.1 DB2 for i5/OS V5R4 DB2 for i V6.1 DB2 UDB for z/OS V8.1 DB2 9 for z/OS V9.1
Oracle	Oracle 10g Standard/Enterprise Editions Release 1 Oracle 11g Standard/Enterprise Release 1
Sybase	Sybase Adaptive Server Enterprise 15.0.3
Microsoft SQL Server	Microsoft SQL Server Enterprise 2005 SP3 and future releases
IBM Informix®	Dynamic Server Express Edition V11.10 and future releases
WebSphere Information Integrator	WebSphere Information Integrator 8.2 FP8 WebSphere Information Integrator 9.1 FP3 WebSphere Information Integrator 9.5 FP2

a. For a detailed list of supported fix pack levels and editions for DB2, see the #7013265 support document at: http://www.ibm.com/support/docview.wss?rs=180&uid=swg27013265

1.4.5 Directory servers

The following directory servers are supported for WebSphere Application Server V8:

► LDAP servers using a federated repository configuration:

– IBM Lotus Domino LDAP Server V7.0 and future releases
– IBM Tivoli Directory Server V6.0 and future releases
– Sun Java System Directory Server V6.0
– Microsoft Active Directory 2003 and future releases
– Novell eDirectory 8.7.3 SP9 and future releases
– IBM z/OS Integrated Security Services V1.8 and future releases
– IBM z/OS.e Integrated Security Services V1.8 and future releases

► LDAP servers using stand-alone LDAP user registry configuration:

– Microsoft Active Directory Application Mode (ADAM) 1.0 SP1 and future releases
– IBM z/OS.e Integrated Security Services V1.8 and future releases
– IBM z/OS Integrated Security Services V1.8 and future releases
– Novell eDirectory 8.7.3 SP9 and future releases
– Microsoft Active Directory 2003 and future releases
– Sun Java System Directory Server 6.0
– IBM Tivoli Directory Server V6.0 and future releases
– IBM Lotus Domino LDAP Server V7.0 and future releases

For more information about requirements, see WebSphere Application Server detailed system requirements at:

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

1.5 Related products

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

► WebSphere Application Server Community Edition
► WebSphere Extended Deployment
► Rational Application Developer Standard Edition for WebSphere Software V8
► Rational Application Developer for WebSphere Software V8
► Project Zero and WebSphere sMash

1.5.1 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 5 specifications such as the following examples:

 – Apache MyFaces
 – Apache OpenEJB
 – Apache Open JPA
 – Apache ActiveMQ
 – TranQL

► Supports the JDK from IBM and Sun

► 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

► Product binary files and source code as no-charge downloads from the IBM website

► Optional fee-based support for WebSphere Application Server Community Edition from IBM Technical support teams

► Can be included in advanced topologies and managed with WebSphere Extended Deployment

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

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

1.5.2 WebSphere Extended Deployment

WebSphere Extended Deployment (XD) is a suite of application infrastructure products that can be installed and used separately or as a package. The suite includes the following products:

► WebSphere Virtual Enterprise

This product, formerly called Operations Optimization, provides application infrastructure virtualization capabilities. Virtualization can lower costs required to create, manage, and run enterprise applications by using existing hardware resources. This package also contains additional management features such as application edition and dynamic application management, management of heterogeneous application servers, and service policy management. These features provide an enhanced management environment for an existing infrastructure.

► WebSphere eXtreme Scale

With this product, formerly called WebSphere Extended Deployment Data Grid, 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 transparent failover to ensure high availability, high reliability, and constant response times. WebSphere eXtreme Scale is a software technology for conducting extreme transaction processing.

► Compute Grid

With this product, organizations can schedule, execute, and monitor batch type jobs and compute-intensive tasks with service policy and workload management.

This composition of packages delivers enhanced qualities of service with features for optimizing IT resources. It can be used in collaboration with WebSphere Application Server packages.

WebSphere Extended Deployment provides the following benefits:

► Provides virtualization capabilities that can dynamically match available resources to changing workload demands. It enables more efficient resource use and improved application performance and scalability.

► Features virtualization, and workload and health management for the following application environments:
  – Apache Tomcat
  – BEA WebLogic
  – JBoss
  – PHP
  – WebSphere Application Server Community Edition

► Offers enhanced capabilities in job scheduling, monitoring, and management for batch-type workloads.

► Delivers customizable health policies and actions to help enhance manageability.

► Supports innovative and highly scalable data fabrics to accelerate data-intensive application performance.

For more information about WebSphere Extended Deployment, see the product page at:

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

1.5.3 Rational Application Developer Standard Edition for WebSphere Software V8

WebSphere Application Server V8 offers integration and support with Rational Application Developer Standard Edition V8. It provides developers the ability to design, develop, deploy, and maintain Java EE applications on the WebSphere Application Server V8.0 environment. It has support for all Java EE artifacts supported by WebSphere Application Server V8.0 such as servlets, JSPs, JSFs, EJBs, XML, SIP, Portlet, web services, and the new integration with Open Services Gateway initiative (OSGi) programming model. It also supports older versions of Java specifications (J2EE 1.2, 1.3, 1.4, and Java EE 5) and previous versions of WebSphere Application Server (V6 Remote, V6.1, and V7).

> **Trial and purchase:** Rational Application Developer Standard Edition for WebSphere Software is optionally installable as a 60-day trial with WebSphere Application Server V8. It is available for purchase as a separate product..

1.5.4 Rational Application Developer for WebSphere Software V8

Rational Application Developer for WebSphere Software is a full-featured Eclipse-based IDE and includes comprehensive 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, as follow:
  – Domain modeling
  – UML modeling
  – Web development

► Web services and XML productivity features

► Portlet development tools

► Relational data tools

► WebSphere Application Server V6, V6.1, V7, and V8 test servers

► Web 2.0 development features for visual development of responsive Rich Internet Applications with Ajax and Dojo

> **Dojo Toolkit:** The Dojo Toolkit is a modular open source JavaScript toolkit (or library) that is designed for the rapid development of cross-platform, JavaScript- or Ajax-based applications and websites.

► Integration with the Rational Unified Process and the Rational tool set, which provides the end-to-end application development life cycle

► 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 the product page at:

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

1.5.5 Project Zero and WebSphere sMash

Project Zero is an incubator project started by IBM that is centered around agile development and the next generation of dynamic web applications. The project brings a simple environment to create, assemble, and run applications based on popular web technologies. This environment includes a scripting run time for Groovy and PHP. It also includes APIs that are optimized to produce REST-style services, integration mashups, and rich web interfaces that correspond to Web 2.0 concepts. It gives access to the latest features by defaulting to experimental modules. Project Zero is available at no cost for development and limited deployment.

Project Zero creates a way to build commercial software, an approach that is called *Community-Driven Commercial Development* and that is powered by community feedback. The Community-Driven Commercial Development process served as a base for the following IBM products:

► WebSphere sMash Developer Edition

This product is a stable build of Project Zero and is available at no initial cost for development and limited deployment usage. It includes useful tools for developing applications in WebSphere sMash.

► WebSphere sMash

This product is the final commercial product and consists of stable modules for production deployment. WebSphere sMash also includes messaging and reliable communications features with Reliable Transport Extension for WebSphere sMash package.

WebSphere sMash includes the following features:

– It is an application-centric run time. You can create an application and run it. Everything needed to run the application is built in, including the HTTP stack.

– It is designed around Dynamic Scripting, REST, Rich Web Interfaces, AJAX, and Feeds, corresponding to Web 2.0 concepts.

– Application logic is created in one of two scripting languages:

  • Groovy, for people that prefer Java
  • PHP for existing PHP programmers

    PHP run time is provided directly in WebSphere sMash.

– It empowers developers to build applications directly on the web with an interface and to compose applications by wiring together REST services.

For more information, see the Project Zero website at:

http://www.projectzero.org/

For more information about WebSphere sMash, see the product page at:

http://www.ibm.com/software/webservers/smash/

2

Integration with other products

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

► IBM Tivoli Access Manager for e-business
► Tivoli Directory Server
► WebSphere MQ
► WebSphere adapters
► WebSphere DataPower
► DB2
► IBM Tivoli Composite Application Manager for WebSphere
► WebSphere Portal Server
► Business process management

2.1 IBM Tivoli Access Manager for e-business

IBM Tivoli Access Manager provides a more holistic security solution at the enterprise level than the standard security mechanisms that are found in WebSphere Application Server.

Tivoli Access Manager provides the following features:

► Defines and manages centralized authentication, access, and audit policy 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, to 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 Windows users with their Windows credentials in WebSphere, if they are connected to a Microsoft Active Directory

► Can be integrated with Tivoli Identity Manager, which supports administering large number 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 the product page at:

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

2.1.1 Integration with WebSphere Application Server

WebSphere Application Server provides its own security infrastructure. This infrastructure consists of some mechanisms that are specific to WebSphere Application Server and many that use open security technologies 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 12, "Security" on page 349.

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 across 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

- ► Faster development and deployment
- ► Reduced application development costs because developers do not have to develop bespoke security subsystems
- ► Built-in, centralized, and configurable handling of legislative business concerns such as privacy requirements

WebSEAL

The WebSEAL server is a resource manager in Tivoli Access Manager architecture for managing and protecting web content resources. WebSEAL works as a reverse HTTP/HTTPS proxy server in front of the web servers or application servers and connects to the policy server for the access control lists (ACLs) as shown on Figure 2-1. Because it handles the HTTP/HTTPS protocol, it is independent of the web server or application server implementation. With this feature, customers can authenticate and authorize clients in a distributed, multivendor integrated environment.
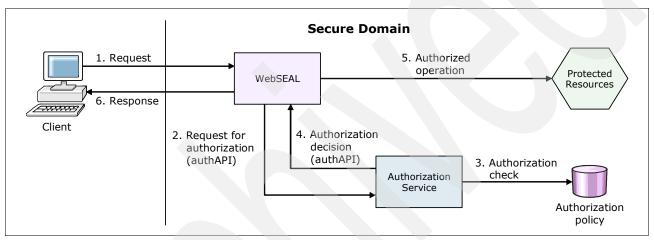
Figure 2-1   WebSEAL as a proxy in WebSphere integration

Repositories

In addition to WebSphere Application Server security, Tivoli Access Manager requires a user repository too. It supports many different repositories, such as IBM Tivoli Directory Server and Microsoft Active Directory. Tivoli Access Manager can be configured 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 then replicated across all local authorization servers.

Tivoli Access Manager for WebSphere component

The Tivoli Access Manager client is embedded in WebSphere Application Server and can be configured by using the scripting and GUI management facilities of WebSphere Application Server. To configure the embedded Tivoli Access Manager client, go to the WebSphere Application Server Version 8.0 Information Center, and search for *enabling embedded Tivoli Access Manager*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

The Tivoli Access Manager server is bundled with WebSphere Application Server Network Deployment. Tivoli Access Manager further integrates with WebSphere Application Server by supporting the special subjects AllAuthenticated and Everyone.

> **Special subjects:** *AllAuthenticated* and *Everyone* are subjects that are specific to WebSphere Application Server. The AllAuthenticated subject allows access to a resource to users who are authenticated, regardless of the repository user groups to which those users might belong. The Everyone subject allows access to a resource to all users regardless of whether they are authenticated.

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

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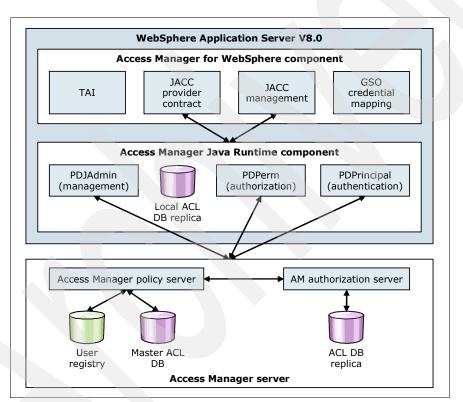
Figure 2-2   Integration of WebSphere Application Server with Tivoli Access Manager

Further advantages of using Tivoli Access Manager

In addition to the enterprise-level advantages mentioned earlier in this chapter, 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
- ► Provides tight integration with WebSphere Application Server

Security, networking, and topology considerations

The Lightweight Directory Access Protocol (LDAP) server contains sensitive data in terms of authentication, authorization, and privacy, and the 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 (privacy and data protection):** Storage of certain data types, such as personally identifiable data in the European Union, on IT systems, can be subject to legal or regulatory issues. You must consult your legal department before deploying such information about your systems. These considerations vary by geography and industry.

2.2 Tivoli Directory Server

In today's highly connected world, directory servers are the foundation of authentication systems for internal, and more commonly, 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 your web applications and identity management initiatives.

2.2.1 Lightweight Directory Access Protocol

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 look up a user principal, determine the attributes that the user has, and determine the groups of which the user is member. You can then make decisions regarding authentication and authorization by using this information.

> **LDAP server:** 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*. The LDAP is still a protocol, although the authentication happens by using the Lightweight Third Party Authentication (LTPA) mechanism. The LDAP carries data only 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). This optimization is possible because LDAP implementations are based on the assumption that the data changes relatively infrequently and that the directory is primarily for looking up data rather than updating data.

For more information about Tivoli Directory Server, see the product page at:

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

2.2.2 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, you can choose from the following LDAP-based repository solutions:

► 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)

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 so that the data is encrypted while transported between the application server layer and the data layer.

LDAPS: Sometimes when LDAP is used with the SSL protocol, it is noted as *LDAPS*.

Legal considerations (privacy and data protection): Storage of certain data types, such as personally identifiable data in the European Union, on IT systems, can be subject to legal or regulatory issues. You must consult your legal department before deploying such information about your systems. These considerations vary by geography and industry.

For a list of supported directory servers for WebSphere Application Server, see System Requirements for WebSphere Application Server Base and Network Deployment V8.0 at the following page:

http://www.ibm.com/support/docview.wss?uid=swg27021246

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

2.3 WebSphere MQ

IBM WebSphere MQ is an asynchronous messaging technology that is available from IBM. WebSphere MQ is middleware technology for application-to-application communication rather than application-to-user and user interface communication.

WebSphere MQ is available on many platforms and operating systems. It offers a fast, robust, and scalable messaging solution that 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, which 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 has been influenced by this dichotomy of generic JMS and proprietary WebSphere MQ access approaches.

For more information about WebSphere MQ, see the product page at:

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

2.3.1 Integration with WebSphere Application Server

WebSphere Application Server messaging is a general term for a group of components that provide the messaging functionality 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.

For WebSphere Application Server V8, the WebSphere MQ messaging provider has enhanced administrative options that support the following functions:

► WebSphere MQ channel compression

  Data sent between WebSphere Application Server and WebSphere MQ can be compressed, reducing the amount of data that is transferred.

► WebSphere MQ client channel definition table

  The client channel definition table reduces the effort required to configure a connection to a queue manager.

► Client channel exits

  Client channel exits are pieces of Java code that you develop and that are executed in the application server at key points during the life cycle of a WebSphere MQ channel. Your code can change the runtime characteristics of the communications link between the WebSphere MQ messaging provider and the WebSphere MQ queue manager.

► Transport-level encryption using SSL

  Transport-level encryption using SSL is the supported way to configure SSL for JMS resources associated with the WebSphere MQ messaging provider. The SSL configuration is associated with the communication link for the connection factory or activation specification.

► Automatic selection of the WebSphere MQ transport type

  Servers in a cluster can be configured automatically to select their transport.

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

► The default messaging provider
► WebSphere MQ
► Third-party JMS providers
► V5 default messaging provider (for migration purposes)

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* 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 service integration bus can connect to other service integration 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 in the following circumstances:

► You are currently using the WebSphere Application Server V5 embedded messaging provider for intra-WebSphere Application Server messaging.

► 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 gives you the benefit of tight integration between WebSphere and the default messaging provider (clustering) and the flexibility of WebSphere MQ.

For more information about messaging with WebSphere Application Server and about new features for WebSphere MQ connectivity, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *enhanced features of the WebSphere MQ messaging provider*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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 detailed description that can help with your decision, see the WebSphere Application Server Version 8 Information Center, and search for *choosing messaging providers for a mixed environment*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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

► Extend the WebSphere MQ and service integration bus networks. This approach is possible by defining a WebSphere MQ link on a messaging engine in a WebSphere Application Server that connects the service integration bus to a WebSphere MQ queue manager.

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

WebSphere MQ applications can send messages to queue destinations on the service integration 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. It cannot be used to consume messages from the other messaging network.

> **Considerations:**
>
> ► WebSphere MQ to service integration bus connections are supported only over TCP/IP.
>
> ► A service integration bus cannot be a member of a WebSphere MQ cluster.

► Integrate specific WebSphere MQ resources into a service integration bus for direct, synchronous access from default messaging applications running in WebSphere Application Servers. This method is achieved by representing a queue manager or *queue sharing group* as a WebSphere MQ server in the WebSphere Application Server cell and adding it to a service integration bus as a bus member.

> **MQ shared queue group:** An 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 running on z/OS, can be accessed in this way from any WebSphere Application Server that is a member of the service integration bus. Only WebSphere MQ queue managers and queue sharing groups running on z/OS can be accessed from a service integration bus in this way.

The WebSphere MQ server does not depend on any one designated messaging engine. This type of connectivity to MQ can tolerate the failure of any given message engine if another is available in the bus, increasing 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. It is not stored by the service integration bus for later transmission to WebSphere MQ when the WebSphere MQ queue manager is not currently available. When a WebSphere Application Server application receives a message from a WebSphere MQ queue, it receives the message directly from the queue.

Figure 2-3 shows a sample integration for WebSphere Application Server and WebSphere MQ.
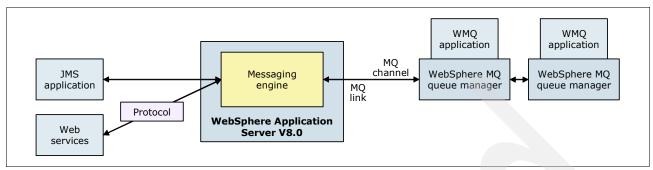
Figure 2-3   WebSphere Application Server integration with WebSphere MQ

For a more detailed description about the messaging features of the WebSphere Application Server V8, see Chapter 10, "Messaging and service integration" on page 299.

2.4 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.

> **WebSphere MQ resource adapter:** This topic is not relevant to the WebSphere MQ resource adapter. For more information about this adapter, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *WebSphere MQ resource adapter*:
>
> http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

IBM WebSphere Adapters provide a set of generic technology and business application adapters with wizards. They quickly and easily service enable EIS such as existing applications, enterprise resource planning (ERP), Human Resources (HR), customer relationship management (CRM), and supply chain systems. WebSphere Adapters can also integrate those systems to IBM business process management (BPM), enterprise service bus (ESB), and application server solutions in a service-oriented architecture (SOA).

WebSphere Adapters implement the Java Connector Architecture (JCA) and Enterprise MetaData Discovery specifications to provide a simple and quick integration experience with graphical discovery tools without resorting to writing code. WebSphere Application Server supports JCA versions 1.0, 1.5, and 1.6, including additional configurable features for JCA 1.5.

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

► Application adapters

  The following adapters integrate enterprise business application suites:

  – JD Edwards EnterpriseOne
  – Oracle E-Business Suite
  – PeopleSoft Enterprise
  – SAP Exchange Infrastructure
  – SAP Software
  – Siebel Business Applications

IBM provides the WebSphere Adapter Toolkit 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 providing an integrated development environment (IDE) and an Integrated WebSphere Test Environment. The WebSphere Adapter Toolkit is available for download at cost from the IBM developerWorks® site at:

http://www.ibm.com/developerworks/websphere/downloads/wat/

For more information about IBM WebSphere Adapters, see the product page at:

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

2.4.1 Integration with WebSphere Application Server

WebSphere Adapter plugs into WebSphere Application Server and provides bidirectional connectivity between enterprise applications (or Java EE components), WebSphere Application Server, and EIS.

Figure 2-4 shows the relation between WebSphere Application Server and a WebSphere Adapter.
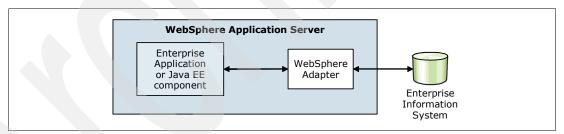
Figure 2-4   WebSphere Adapter integration with WebSphere Application Server

2.5 WebSphere DataPower

IBM WebSphere DataPower SOA Appliances represent an important element in the IBM holistic approach to SOA. IBM SOA appliances are purpose-built, easy-to-deploy network devices that simplify, secure, and accelerate XML and web services deployments while extending the SOA infrastructure. These new appliances offer an innovative, pragmatic approach to harness the power of SOA while simultaneously enabling you to use existing application, security, and networking infrastructure investments.

The IBM WebSphere DataPower SOA Appliances family offer the following features:

► DataPower Appliances that are rack-mountable 1U hardware devices or Blade servers that mount in an IBM BladeCenter® chassis

DataPower Appliances have a web-based graphical user interface (web GUI) as an administrative interface where administrators connect to the appliance from an Internet browser. In development environments, the web GUI is the primary interface to create services.

► A command-line interface that can be accessed by using Telnet, Secure Shell (SSH), or a serial connection.

The third administration interface is a SOAP-based XML interface. Its specifications include SOAP management, WS-Management, and Web Services Distributed Management (WSDM).

► XML/SOAP firewall, field-level XML security, data validation, XML web services access control, and service virtualization

► Lightweight and protocol-independent message brokering, integrated message-level security, fine-grained access control, and the ability to bridge important transaction networks to SOAs and ESBs

► High performance, multistep, wire-speed message processing, including XML, XML Stylesheet Language Transformation (XSLT), XPath, and XML Schema Definition (XSD)

► Centralized web services policy and service-level management

► Web services standard support:
  – WS-Security
  – Security Assertion Markup Language
  – Portions of the Liberty Alliance Protocol
  – WS-Federation
  – WS-Trust
  – XML Key Management Specification (XKMS)
  – Radius, XML Digital Signature
  – XML-Encryption
  – WSDM
  – WS-SecureConversation
  – WS-Policy
  – WS-SecurityPolicy
  – WS-ReliableMessaging
  – SOAP
  – Web Services Description Language (WSDL)
  – Universal Description Discovery, and Integration (UDDI)

► Transport layer flexibility, which supports HTTP/HTTPS, MQ, SSL, File Transfer Protocol (FTP), and others

► Scalable, wire-speed, any-to-any message transformation, such as arbitrary binary, flat text and XML messages, which include COBOL copybook, CORBA, CICS, ISO 8583, ASN.1, EDI, and others

DataPower appliances can meet the challenges that are present in an SOA network with the following features:

► Consumable simplicity

An easy-to-install and easy-to-maintain network appliance satisfies both application and network operational groups and that supports current and emerging standards and readily available XML web services standards.

► Enhanced security

Key support includes XML/SOAP firewall and threat protection, field-level XML security, data validation, XML web services access control, service virtualization, and SSL acceleration.

► Acceleration

A drop-in solution can streamline XML and web service deployments, helping to lower the total cost of ownership and accelerate a return on your assets, as you continue to move to SOA. SOA appliances are purpose-built hardware devices that can offload overtaxed servers by processing XML, web services, and other message formats at wire speed.

For more information about IBM WebSphere DataPower SOA Appliances, see the product page at:

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

For detailed technical information, see the WebSphere DataPower SOA Appliances Information Center at:

http://publib.boulder.ibm.com/infocenter/wsdatap/v3r8m2/index.jsp

2.5.1 DataPower appliance models

The WebSphere DataPower appliance family contains the following six model groups at the time this book was written. Each appliance has its own characteristics and fits various business needs.

► IBM WebSphere DataPower XML Accelerator XA35

This model is built with a highly efficient XML processing engine, which can help speed common types of XML processing by offloading the processing from servers and networks. It can perform XML parsing, XML schema validation, XPath routing, XSLT, XML compression, and other essential XML processing with wire-speed XML performance. It also supports Intelligent XML processing such as XML routing, pipelining, compression, and XML/XSL caching.

For more information about the DataPower XA35, see the WebSphere DataPower XML Accelerator XA35 product page at:

http://www.ibm.com/software/integration/datapower/xa35/

► IBM WebSphere DataPower XML Security Gateway XS40

XS40 provides a security-enforcement point for XML and web service transactions. It filters traffic at wire speed through stack layers 2–7. It can selectively share information through encryption or decryption and signing or verification. The appliance supports various access control mechanisms, including XACML, Security Assertion Markup Language (SAML), SSL, LDAP, RADIUS, and simple client URL maps. It can reject unsigned messages, validate signatures, and perform XML schema and message validation.

For more information about the DataPower XS40, see the WebSphere DataPower XML Security Gateway XS40 product page at:

http://www.ibm.com/software/integration/datapower/xs40/

► IBM WebSphere DataPower Integration Appliance XI50/XI52 and Integration Blade XI50B

These appliances provide many core functions to SOA deployments in a hardened device. They provide transport-independent transformations between binary, flat text files, and XML message formats. Visual tools are used to describe data formats, create mappings between different formats, and define message choreography. The XI50 appliances can

transform binary, flat text, and other non-XML messages to offer an innovative solution for security-rich XML enablement, ESBs, and mainframe connectivity.

The appliance integrates with various registry and repository, security, identity, and service management software applications. Such applications include IBM Tivoli Access Manager, IBM Tivoli Federated Identity Manager, IBM Tivoli Composite Application Manager for SOA, and WebSphere Registry and Repository (WSRR). The appliance also features deep integration with products such as WebSphere MQ, WebSphere Enterprise Service Bus, WebSphere Message Broker, and DB2 to help process SOA transactions in a faster, more secure, and simplified way.

The IBM WebSphere DataPower Integration Blade XI50B is a BladeCenter mountable edition of the XI50 device. The XI52 device is a new, stronger hardware version of the XI50. It is mounted in a 2U high rack mount box.

For more information about the DataPower XI50/XI52, see the WebSphere DataPower Integration Appliance XI50 product page at:

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

Figure 2-5 illustrates the flow of using the DataPower Integration Appliance XI50/XI52 appliances in the various tiers.
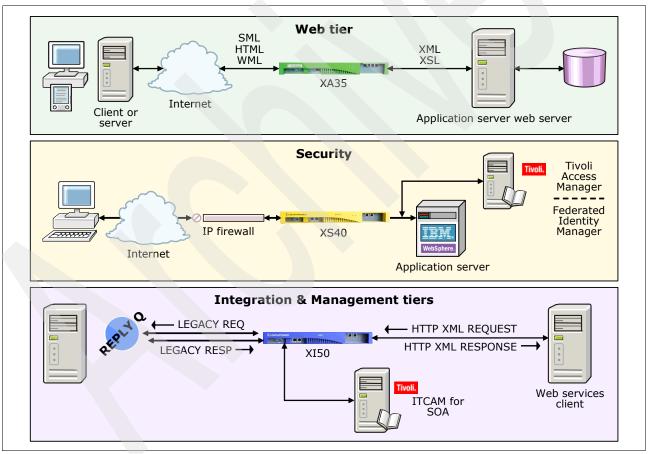
Figure 2-5   The XA35, XS40, and XI50 DataPower Appliances

▶ IBM WebSphere DataPower Integration Appliance XI50 for zEnterprise

The XI50 for zEnterprise edition is a zEnterprise enabled version of the standard XI50. It has some System z specific features as follows:

– VLAN support. Enforced isolation of network traffic with secure private networks and integration with Resource Access Control Facility (RACF) security.

– Improved support. Monitoring of hardware with "call home" for current or expected problems and support by a System z Service Support Representative.

– System z packaging. Increased quality with pre-testing of blade and zEnterprise BladeCenter Extension. Upgrade history available to ease growth.

– Operational controls. Monitoring rolled into System z environment from a single console.

For more information about DataPower XI50z, see the IBM WebSphere DataPower Integration Appliance XI50 for zEnterprise product page at:

http://www.ibm.com/software/integration/datapower/xi50z/

▶ IBM WebSphere DataPower B2B Appliance XB60/XB62

This appliance provides a high throughput, secure entry point at the edge for routing data into the enterprise. It helps manage and connect to trading partners to extend integration beyond the enterprise. It has a built-in business-to-business (B2B) transaction viewer so that you can monitor transactions.

The XB62 device is a new, stronger hardware version of the XB60, and it is mounted in a 2U high rack mount box.

For more information about DataPower XB60/XB62, see the WebSphere DataPower B2B Appliance XB60 product page at:

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

▶ IBM WebSphere DataPower Low Latency Appliance XM70

The low-latency appliance provides publish-subscribe messaging and content-based routing in high throughput messaging environments. It can act as a highly available messaging backbone for data feeds and other high-volume transactions. This appliance supports a wide area of standard messaging protocols (WebSphere MQ, WebSphere JMS, TIBCO EMS, TIBCO RV, HTTP, HTTPS). It also supports WebSphere MQ Low Latency Messaging clients and can connect to storage area networks (SANs) through Fibre Channel.

For more information about DataPower XM70, see the WebSphere DataPower Low Latency Appliance XM70 product page at:

http://www.ibm.com/software/integration/datapower/llm_xm70/

▶ IBM WebSphere DataPower XC10 Elastic Caching Appliance

The XC10 unit provides a large 240 GB built-in cache.This can accelerate application performance by putting the data closer to the applications/services. The data store source is accessible to Java applications through a native Java client or through a RESTgateway for applications that are not based on Java.

For more information about DataPower XC10, see the WebSphere DataPower XC10 Appliance product page at:

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

You can also find information in the WebSphere DataPower XC10 Appliance Information Center at:

http://publib.boulder.ibm.com/infocenter/wdpxc/v1r0/index.jsp

Figure 2-6 illustrates the flow of the B60, XM70, and XC10 DataPower Appliances.
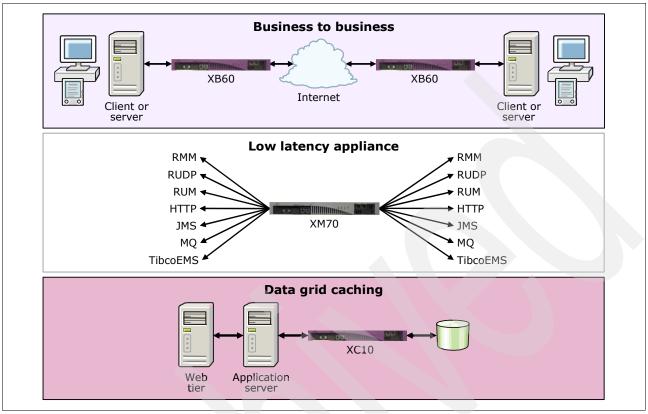
Figure 2-6   The XB60, XM70, and XC10 DataPower Appliances

2.5.2 Integration with WebSphere Application Server

In WebSphere Application Server V8, with the consolidated administration feature for WebSphere DataPower, you can manage and integrate appliances into your environment. The Integrated Solutions Console contains an administration interface (DataPower appliance manager) to manage multiple WebSphere DataPower boxes (Figure 2-7).

Figure 2-7   DataPower appliance manager interface of the Integrated Solutions Console

The Integrated Solutions Console is the single point of administration to manage both WebSphere Application Server and WebSphere DataPower and for 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 version, 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 have asked the DataPower appliance manager to process.

2.6 DB2

IBM 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.

DB2 has editions to work on Linux, UNIX, Windows, System i, and System z. The different editions scale from the one-user enabled Personal edition, through the Express Community edition, which is available at no cost, to the partitioned, massively parallel Enterprise Server edition.

Starting with DB2, the database supports two different data structures: 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 the DB2 database software page at:

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

IBM DB2 pureScale™ is a new high performance technology database solution. It implements in-memory transactions in a distributed server environment. For more information about pureScale, 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" article on IBM developerWorks

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

2.6.1 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 pureQuery runtime environment, DB2 provides an alternate 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 (JPA) of Java EE and Java SE environments.

► You can configure a service integration bus (messaging) member to use DB2 as a data store.

► The session management facility of WebSphere Application Server can be configured for database session persistence, by using DB2 as the data store. You can collect and store session data in a DB2 database.

► DB2 can be used 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 the WebSphere Application Server Version 8 Information Center at the following address and search for *data access resources*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

2.7 IBM Tivoli Composite Application Manager for WebSphere

Typical testing, staging, and production environments consist of several components, such as web servers, application servers (which might not be WebSphere), and databases. The application servers can be organized into one or more cells. Often stand-alone servers are used for specific functions. Also DataPower devices, Portal, and Process Servers can extend the system. In real production sites, all these components are organized into clusters.

The administrators need to monitor the performance, availability, and reliability of the system and must note any error that is occurring in time to fix it without downtime.

IBM Tivoli Composite Application Manager 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. IBM Tivoli Composite Application Manager for WebSphere provides in-depth WebSphere-based application performance analysis and tracing facilities. It provides detailed reports that you can use to enhance the performance of your applications.

For more information about IBM Tivoli Composite Application Manager for WebSphere, see the Tivoli Composite Application Manager for Application Diagnostics page at:

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

2.7.1 Integration with WebSphere Application Server

With IBM Tivoli Composite Application Manager for WebSphere, you can analyze the health of the WebSphere Application Server and the transactions that are invoked 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 invokes services from other application servers, including mainframe applications in IMS or CICS.

IBM Tivoli Composite Application Manager for WebSphere provides a flexible level of monitoring, from a non-intrusive production ready monitor, to detailed tracing for problems of locking or memory leaks. IBM Tivoli Composite Application Manager for WebSphere provides a separate interactive web console and allows monitoring data to be displayed on the Tivoli Enterprise Portal.

IBM Tivoli Composite Application Manager 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

2.7.2 Architecture of Tivoli Composite Application Manager for WebSphere

Tivoli Composite Application Manager for WebSphere is a distributed performance monitoring application for application servers. Its components are connected through TCP/IP communication. The central component of Tivoli Composite Application Manager for WebSphere, the managing server, is its heart and brain. It collects and displays various performance information from application servers.

The application servers run a component of Tivoli Composite Application Manager for WebSphere, called the *data collector* (DC), 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 machines where the data collector resides.

Figure 2-8 shows the overall architecture of Tivoli Composite Application Manager for WebSphere.
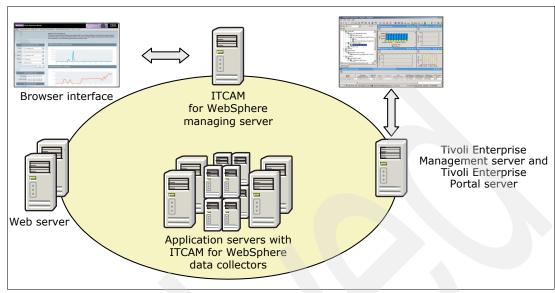
Figure 2-8   ITCAM for WebSphere architecture

For more information about IBM Tivoli Composite Application Manager 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

2.8 WebSphere Portal Server

WebSphere Portal is a web portal solution, which 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.

WebSphere Portal Server has the following main features among others:

▶ 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 (JSR) 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 might be customized for customer needs.

- It can be built in a cluster to provide a highly available environment, similar to WebSphere Application Server.

- Similar to most IBM applications that have a web interface, WebSphere Portal Server has a portlet interface such as Lotus Domino (email and database portlets), IBM Cognos® BI, and so on.

- 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 the WebSphere Portal Server - Enterprise portal software product page at:

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

2.8.1 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.

The 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 and render it by using only the Portal Server aggregation engine. Then, you can use performance needs between WebSphere Portal Server and WebSphere Application Server.

2.9 Business process management

This section provides information about IBM Business Process Manager and IBM WebSphere Enterprise Service Bus. Both products implement SOA environments, but at different levels of complexity.

IBM Business Process Manager helps customers to automate real-life business processes, including human interaction tasks, web interface actions, and business rule-based decisions. Mainly the process is controlled by a state machine engine or a simple business process. The building blocks of these process steps are stand-alone services, based on the Service Component Architecture (SCA) standard.

These products support a broad range of native bindings and adapters for service-oriented integration. They include web services, MQ and JMS messaging, HTTP, EJB, databases, files, file transfer, email, Lotus Domino, System i, CICS, IMS, SAP, Oracle, Siebel, PeopleSoft, JD Edwards, and others.

Announce date for Business Process Manager: IBM Business Process Manager was announced on 11 April 2011 as the follow-on offering for WebSphere Process Server and WebSphere Lombardi Edition.

For more information about the Business Process Manager and WebSphere Enterprise Service Bus products, see the following websites:

► IBM Business Process Manager product page

  http://www.ibm.com/software/integration/business-process-manager/

► WebSphere Enterprise Service Bus product page

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

2.9.1 Integration with WebSphere Application Server

Both the Business Process Manager and WebSphere Enterprise Service Bus products are built on top of the WebSphere Application Server infrastructure. WebSphere Application Server V7.0 included a feature pack for SCA. This feature pack is an integrated part of WebSphere Application Server V8. With this service pack, developers can construct applications based on the Open SCA specification. The SCA modules of Business Process Manager and WebSphere ESB Server can use import and export bindings to interoperate with Open SCA services running in WebSphere Application Server.

For more details about this topic, see the WebSphere Application Server Version 8 Information Center at the following address and search for *SCA in WebSphere Application Server: Overview*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

3

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 introduces common WebSphere Application Server concepts in the following sections:

- ► Core concepts of WebSphere Application Server
- ► Additional concepts for WebSphere Application Server
- ► Server configurations
- ► Security
- ► Service integration
- ► Clusters and the application server cluster
- ► Run times

3.1 Core concepts of WebSphere Application Server

The following concepts are at the core to understanding the architecture of WebSphere Application Server V8:

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

A person in an administrative role must manage these concepts on a regular basis. Understanding these concepts and how they apply to your environment will help facilitate any design and troubleshooting situations.

This section provides information about these concepts. You can find additional concepts about WebSphere Application Server that build on these core concepts in 3.2, "Additional concepts for WebSphere Application Server" on page 57.

3.1.1 Applications

At the heart of WebSphere Application Server is the ability to run *applications*. WebSphere Application Server V8 can run the following types of applications:

► Java EE applications
► Portlet applications
► Session Initiation Protocol applications
► Business-level applications
► OSGi applications

This section highlights these types of applications.

Java EE applications

The Java Platform, Enterprise Edition (Java EE), specification is the standard for developing, deploying, and running enterprise applications. WebSphere Application Server V8 provides full support for the Java EE 6 specification. The Java EE programming model has multiple types of application components:

► Enterprise beans
► Servlets and JavaServer Pages (JSP) files
► Application clients

The primary development tool for WebSphere Application Server Java EE 6 applications is Rational Application Developer for WebSphere V8. Rational Application Developer V8 or Rational Software Architect V8 ships with WebSphere Application Server and contains the tools that you need to create, test, and deploy Java EE 6 applications. Java EE applications are packaged as enterprise archive (EAR) files.

Portlet applications

The *Portlet* container in WebSphere Application Server V8 provides the runtime environment for Java Specification Requests (JSR) 268-compliant 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.

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 JSR 286, see the JSR page 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 servlet written to the JSR 116 specification. 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 has been 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 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 application archive (EAR file), similar to other Java EE components.

For more information about SIP applications, see the following references on the web:

► JSR 116 SIP Servlet API 1.0 Specification

   http://www.jcp.org/aboutJava/communityprocess/final/jsr116/

► JSR 116

   http://jcp.org/en/jsr/detail?id=116

► RFC 3261

   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. Enterprise-level applications have a grouping notion. It includes WebSphere artifacts and artifacts that are not based on WebSphere, such as 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 packages

► Applies to the post-deployment side of the application life cycle

► Contains additional libraries or artifacts that are not based on Java EE

► Includes artifacts that run on heterogeneous environments that include WebSphere run times and run times that are not WebSphere based

► Is defined in a recursive manner (for example, if an application includes other applications)

OSGi applications

The concept of *OSGi* evolved from managing software components that are deployed to network devices such as storage area network (SAN) arrays. When a storage module is removed from a system, the entire system does not fail. The system itself distributes responsibility to other modules. With this concept, software components on network devices

can be managed dynamically without disrupting the operation of the device. This OSGi concept has been carried over to enterprise applications. In the same fashion, the OSGI concept in software brings the concept of availability to users.

OSGi applications are built on an architecture for developing and deploying modular applications and libraries. An OSGi container specifically supports developing Java applications that can be broken up into modules. OSGi applications are built by using the OSGi API and deploying it into an OSGi container. WebSphere Application Server provides an OSGi container as part of its basic architecture.

From an administrator's and developer's perspective, OSGi provides these advantages:

► Different application modules can be in installed, uninstalled, started, and stopped without restarting the container.

► More than one version of an application module can run at the same time.

► OSGi provides the infrastructure for the development and deployment of service-oriented applications (SOA) and mobile and embedded applications.

Modular components and features that are created with OSGi technology are enabled in several ways. First, 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 Java virtual machine (JVM), a bundle behaves similar 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. Second, because each bundle is sand boxed, versions of logging libraries with one bundle do not conflict with different versions of the same product in different bundles.

The OSGi application feature of the WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API (JPA) 2.0 provides comprehensive run time and integrated administrative support for deploying OSGi applications to WebSphere Application Server. Support for OSGi applications applies to WebSphere Application Server V8.

Application integrity is maintained by isolating applications from one another while enabling the sharing of specific bundles to be directed by application assembly metadata. The application deployment process is augmented to enable application content to be provisioned from a combination of application-specific archives and shared OSGi bundle repositories. This augmentation reduces application archive size and disk and memory footprint.

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.

For a reference on developing enterprise OSGi applications for WebSphere Application Server, see the article "Developing enterprise OSGi applications for WebSphere Application Server: An overview of modular applications" on IBM developerWorks at:

http://www.ibm.com/developerworks/websphere/techjournal/1007_robinson/
1007_robinson.html?S_TACT=105AGY82&S_CMP=MAVE

3.1.2 Application servers

The *application server* (Figure 3-1) is the platform on which Java language-based applications run. It provides services that can be used by business applications, such as database connectivity, threading, and workload management.
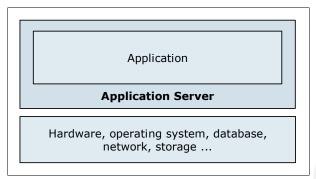
Figure 3-1 Relationship between an application and the application server

At the core of each product in the WebSphere Application Server family is an *application server*. The following product members of the WebSphere Application Server family are presented in this book:

► Express
► Base
► Network Deployment

Each member has essentially the same architectural structure shown in Figure 3-2. The application server structure for the Base and Express platforms is identical. Licensing terms and platform support differentiate the products.
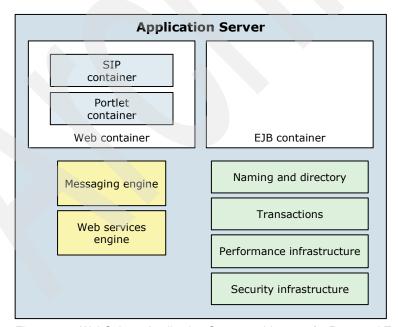
Figure 3-2 WebSphere Application Server architecture for Base and Express

The Base and Express platforms are limited to stand-alone application servers. With the Network Deployment platform, more advanced topologies provide workload management, scalability, high availability, and central management of multiple application servers, as shown in Figure 3-3. You can also manage multiple Base profiles centrally, but you do not have workload management, scalability, and high availability capabilities for those Base profiles.
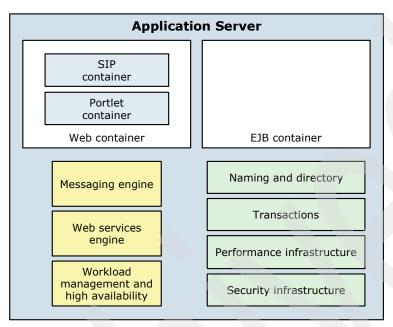
Figure 3-3   Application server architecture with Network Deployment capabilities

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. 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 (see Figure 3-6 on page 52).

A stand-alone server (Figure 3-4) can be managed from its own administrative console. It functions independent from all other application servers. You can also use the `wsadmin` scripting facility in WebSphere Application Server to perform every function that is available in the administrative console application.
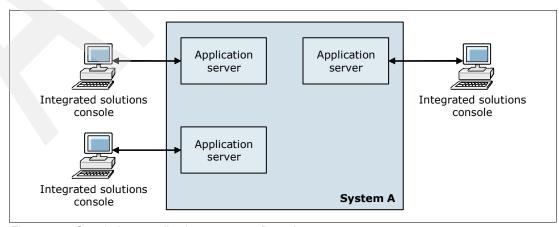
Figure 3-4   Stand-alone application server configuration

Multiple stand-alone application servers can exist on a machine, either through independent installations of the WebSphere Application Server product binaries or by creating multiple application server profiles within one installation. However, stand-alone application servers do not provide workload management or failover capabilities. They run isolated from each other.

With WebSphere Application Server for z/OS, it is possible to use workload load balancing and response time goals on a transactional base and on a special clustering mechanism, the *multi-servant region*, with a stand-alone application server. For more information about this concept, see 14.1.5, "Structure of an application server" on page 389.

Managing stand-alone servers from a central point: With WebSphere Application Server V8, you can manage stand-alone servers from a central point by using administrative agents and a job manager. For more information about this feature, see 3.3.2, "Flexible management" on page 66.

Distributed application servers

With the Network Deployment packaging, 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*, which is explained in 3.1.6, "Deployment manager" on page 56. The application servers can reside on the same machine as the deployment manager or on multiple separate machines. Administration and management are handled centrally from the administration interfaces of the deployment manager, as illustrated in Figure 3-5.
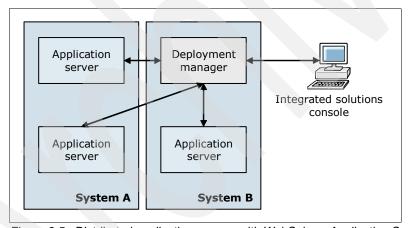
Figure 3-5   Distributed application servers with WebSphere Application Server V8

With a distributed server configuration, you can create multiple application servers to run unique sets of applications and to manage those applications from a central location. However, 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 that, when one server fails, another server in the cluster continues processing.

Workload is distributed among web and Enterprise JavaBeans (EJB) containers in a cluster by using a weighted round-robin scheme. In 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. With this configuration, requests can be dispatched to a cluster member according to real-time load and regardless of whether the member reaches it defined response time goals.

By creating various profiles (see 3.1.3, "Profiles" on page 52), you can create a distributed server configuration by using one of the following methods:

► Create a deployment manager profile to define the deployment manager. 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 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. Application servers can then be created by using the Integrated Solutions 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 system or z/OS image.

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 two profiles: a deployment manager profile and a federated application server profile. Both profiles reside on the same machine.

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 images.

3.1.3 Profiles

Runtime environments are built by creating *profiles*. A profile can define a deployment manager, a stand-alone application server, or an empty node to be federated (added) to a cell. 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.

WebSphere Application Server runtime environments are built by creating profiles. Profiles are sets of files that represent a WebSphere Application Server configuration. Two categories of WebSphere Application Server files are available, as shown in Figure 3-6:

► *Product files* are a set of read-only static files or product binary files that are shared by any instances of the WebSphere Application Server product.

► *Configuration files (profiles)* are a set of user-customizable data files. This file set includes WebSphere configuration, installed applications, resource adapters, properties, log files, and so on.
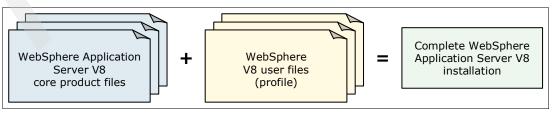
Figure 3-6   Anatomy of a profile

Administration is greatly enhanced when using profiles instead of multiple product installations. Not only is disk space saved, but updating the product is simplified when you maintain a single set of product core files. Creating new profiles is more efficient and less prone to error than full product installations, allowing a developer to create separate profiles of the product for development and testing.

Each profile is stored in a unique directory path, as shown in Figure 3-7, which is selected by the user at profile creation time. Profiles are stored in a subdirectory of the installation directory by default, but they can be located anywhere.
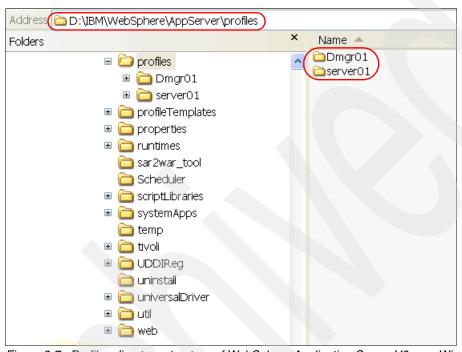
Figure 3-7   Profiles directory structure of WebSphere Application Server V8 on a Windows system

3.1.4 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. (Virtualization allows multiple operating systems on one machine.) 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, which are managed from one central administration server.

Figure 3-8 illustrates three nodes (Node01, Node02, and Node03).
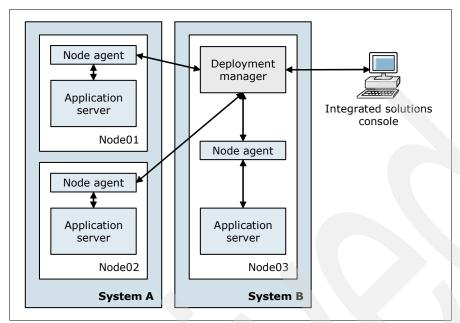
Figure 3-8   Node concept in a WebSphere Application Server Network Deployment configuration

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 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 3-8, each node has its own node agent that
communicates directly with the deployment manager. The node agent is responsible for
monitoring the application servers on the node and routing administrative requests from the
deployment manager to those application servers.

Node groups

A *node group* is a grouping of nodes within a cell that have similar capabilities. A node group
validates that the node can perform certain functions before allowing them. For example, a
cluster cannot contain both z/OS nodes and nodes that are not z/OS-based. In this case, you
can define multiple node groups: one for the z/OS nodes and one for nodes other than z/OS.
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.

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.

> **Sysplex:** A sysplex is the z/OS implementation of a cluster. This technique uses
> distributed members and a central point in the cluster, a *coupling facility*, for caching,
> locking, and listing. The coupling facility runs a special firmware, the Coupling Facility
> Control Code (CFCC). The members and the coupling facility communicate with each
> other by using a high-speed memory-to-memory connection of up to 120 Gbps.

Figure 3-9 shows an example of a node and a node group.
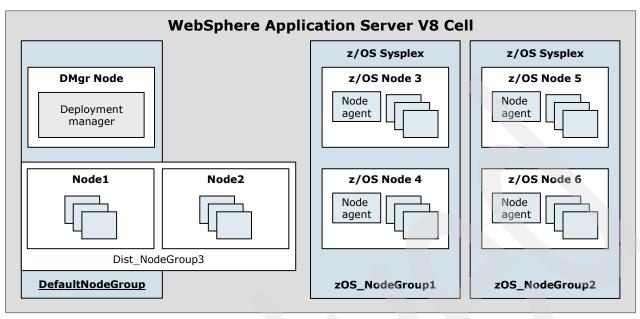
Figure 3-9   Cell, deployment manager, node, node agents, and node group concepts

3.1.5 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. Figure 3-9 shows a cell that contains multiple nodes and node groups.

In a Network Deployment environment, a cell can consist of multiple nodes (and node groups), which are all administered from a single point, the deployment manager (Figure 3-5 on page 51). 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.

This configuration is referred to as a *heterogeneous cell* and is illustrated in Figure 3-10.
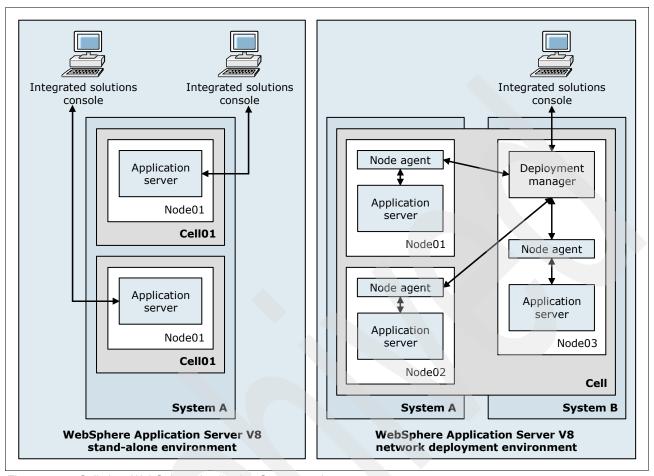
Figure 3-10   Cells in a WebSphere Application Server topology

3.1.6 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, similar to the configuration shown in Figure 3-9 on page 55. The deployment manager communicates with the node agent to manage the applications servers within one node.

A deployment manager provides management capability for multiple federated nodes and can manage nodes that span multiple systems and platforms. A node can be managed only 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 a master configuration repository. This centralized repository is managed by the deployment manager and synchronized with local copies that are held on each of the nodes.

3.2 Additional concepts for WebSphere Application Server

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

► Administrative agent
► Job manager
► Web servers
► Web server plug-in
► Proxy servers
► Generic servers
► Centralized installation manager
► Intelligent runtime provisioning

3.2.1 Administrative agent

An *administrative agent* (Figure 3-11) is a component that provides enhanced management capabilities for stand-alone (Express and Base) application servers. All configurations that are related to the application server are connected directly to the administrative agent that provides services to administrative tools.
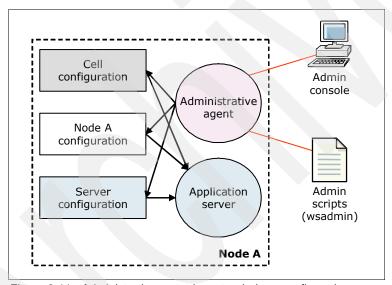
Figure 3-11   Administrative agent in a stand-alone configuration

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, therefore, 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 WebSphere Application Server Network Deployment.

3.2.2 Job manager

A *job manager* (Figure 3-12) 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 large geographical area.

Many of the management tasks that you can perform with the job manager are tasks that you perform with the product, such as application management, server management, and node management. With the job manager, however, tasks can be aggregated and performed across multiple targets.
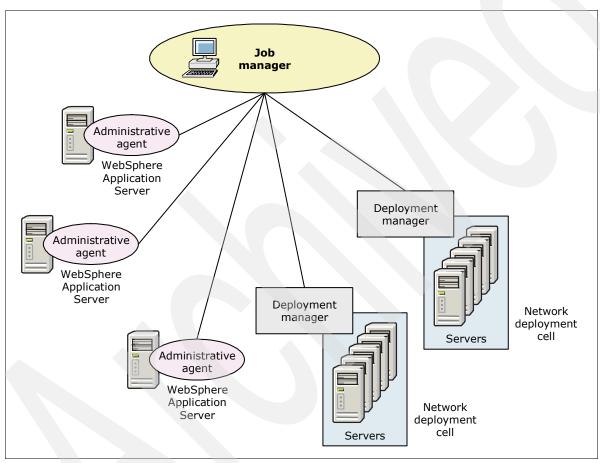
Figure 3-12   Job manager architecture

The job manager was introduced with WebSphere Application Server V7 and is only available with WebSphere Application Server Network Deployment and WebSphere Application Server for z/OS.

3.2.3 Web servers

Although web servers are independent products, they can be defined to and managed by the administration process of WebSphere Application Server. The primary purpose for this approach is to enable the administrator to associate applications with one or more defined web servers to generate the proper 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 resides on an unmanaged node. 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 machine 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 3-13 illustrates a managed server on a managed node.
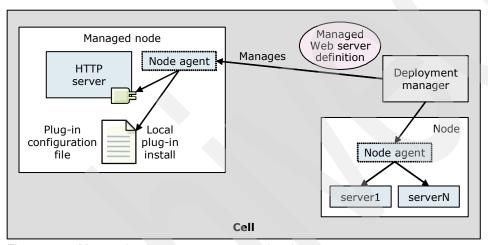
Figure 3-13   Managed web server on a managed node

Unmanaged nodes

Unmanaged nodes are not managed by WebSphere Application Server. You usually find these nodes outside the firewall or in the demilitarized zone (DMZ). You have to manually transfer the web server plug-in configuration file to the web server on an unmanaged node. In a z/OS environment, you have to use unmanaged nodes if the web server is not running on the z/OS platform. Figure 3-14 illustrates this configuration.
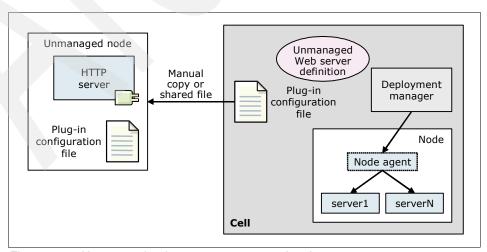
Figure 3-14   Unmanaged web server on a managed node

An IBM HTTP Server on an unmanaged node can be administered from the Integrated Solutions 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, as shown in Figure 3-15. This configuration does not require a node agent. IBM HTTP Server is shipped with all WebSphere Application Server packages.
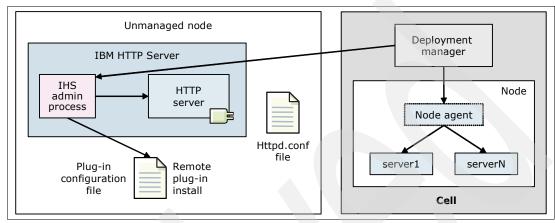
Figure 3-15   IBM HTTP Server on an unmanaged node

3.2.4 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 3-16.
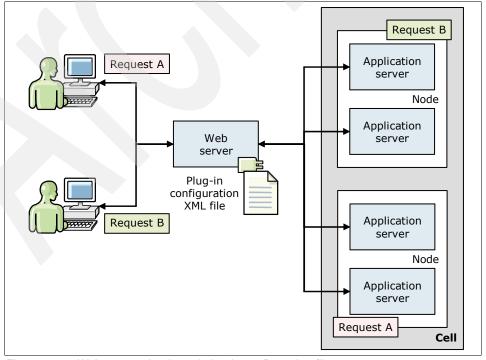
Figure 3-16   Web server plug-in and plug-in configuration file concept

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 if a request should be 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 that is generated by WebSphere Application Server, propagated to the web server, and stored in the plug-in directory of the web server.

3.2.5 Proxy servers

A *proxy server* is a specific type of application server that routes requests to content servers that perform the work. The proxy server is the initial point of entry, after the protocol firewall, for requests that enter the environment.

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

► WebSphere Application Server Proxy
► DMZ Secure Proxy Server

WebSphere Application Server Proxy

WebSphere Application Server Proxy provides many functions that a web server and plug-in have, but it is not a full replacement for a plug-in because it does not have web serving capabilities. (Static content can be served 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 Proxy supports two protocols:

► HTTP
► SIP

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 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 by protecting the content servers from surges in traffic.

You can modify an existing proxy server to perform 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 perform caching.

> **Proxy server:** When using WebSphere Application Server for z/OS V8, the proxy server uses the Workload Management component to perform 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 that you can use 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 that reside within the domain that is 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 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 and that provides security enhancements that allow deployments inside a DMZ as illustrated in Figure 3-17.
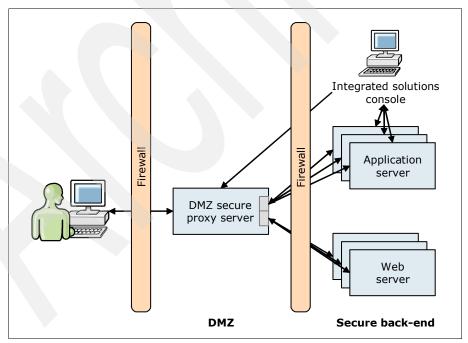
Figure 3-17   DMZ secure proxy simplified topology

The DMZ Secure Proxy Server improves security by minimizing the number of external ports that are opened and running as an unprivileged user when binding to well-known ports. It 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 reverse proxy, see 5.3.6, "Reverse proxy topology" on page 133.

3.2.6 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. With the generic servers function of 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 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

3.2.7 Centralized installation manager

With the *centralized installation manager,* single installation can be pushed out as an installation package from the deployment manager to a series of endpoints. The endpoint can be either a node that is not part of a Network Deployment cell or an existing Network Deployment node that might need a fix pack. The centralized installation manager simplifies the installation and maintenance of application servers within a Network Deployment cell.

With the centralized installation manager, an administrator can remotely install or uninstall product components or provide maintenance to specific nodes directly from the Integrated Solutions Console. This work can be done without having to log in and repetitively complete these tasks for each node. The centralized installation manager invokes the standard or update installer to perform the work. Using centralized installation manager can reduce the number of steps that are required to create and manage the environment and can simplify installation and patch management.

3.2.8 Intelligent runtime provisioning

Intelligent runtime provisioning is a concept that was first 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 3-18. 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.
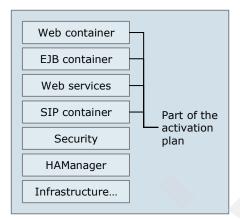
Figure 3-18   Intelligent runtime provisioning

3.3 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:

► Single cell configurations
► Flexible management

3.3.1 Single cell configurations

With a cell configuration (see 3.1.5, "Cells" on page 55), you can group 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 machine (physical machine) if it contains only one operating system

► An operating system virtual image, which is the case for virtualization purposes, if the host server machine contains multiple operating system images

► A z/OS image

Single system configurations

With the Base and Express configurations, you can only create a cell that contains a single node with a single application server as shown in Figure 3-19.

> **Multiple profile installations:** A system can contain multiple Base and Express profile installations and, therefore, can contain multiple cells.

A cell in a Network Deployment environment is a collection of multiple nodes. Each node can contain one or more application servers. The cell contains one deployment manager that manages the nodes and servers in the cell. A node agent in the node is the contact point for the deployment manager during cell administration. The deployment manager resides on the same system as the nodes.
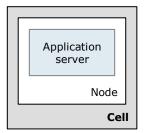
Figure 3-19   Single cell configuration in Base and Express packages

A single system configuration in a distributed environment includes all processes in one system as illustrated in Figure 3-20.
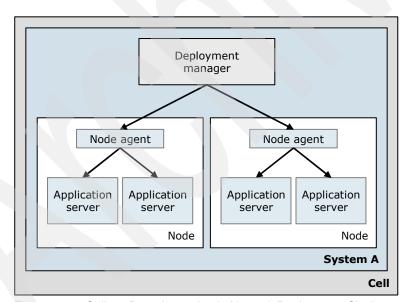
Figure 3-20   Cell configuration option in Network Deployment: Single system

Multiple system configurations

A Network Deployment environment provides the flexibility 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 3-20 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 the example, System A can be an AIX system, System B can be a Windows system, and System C can be a z/OS image.
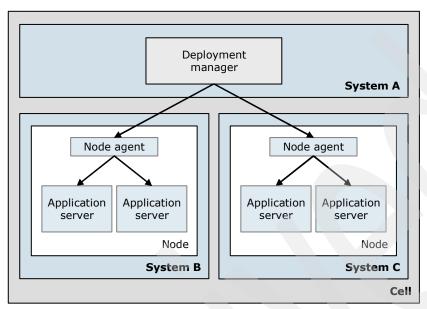
Figure 3-21   Cell configuration option in Network Deployment: Multiple systems

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

3.3.2 Flexible management

Flexible management is a concept that was introduced with WebSphere Application Server V7. With flexible management components, such as the administrative agent and the job manager, you can build advanced and large-scale topologies and 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 (Express and Base) application servers that are all installed in the same system. Figure 3-11 on page 57 shows the administrative agent management model.

It is possible to manage multiple administrative agents with a job manager. These administrative agents can reside 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 3-22).
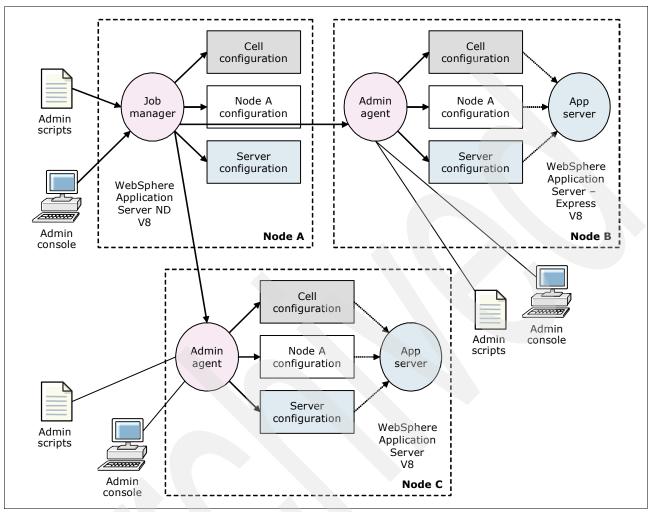
Figure 3-22   Job manager management model for multiple administrative agents

The job manager addresses the limitations that are inherent in the current management architecture by adopting 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 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 for WebSphere Application Server. In turn, the administrative agent or the deployment manager runs the jobs that update the configuration, starts or stops applications, and performs various other common administrative tasks.

To create a job manager and coordinate administrative actions among multiple deployment managers and administer multiple unfederated application servers, you need to create a management profile during the profile creation phase of the installation.

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.

> **Job manager versus deployment manager:** The job manager is not a replacement for a deployment manager. It is an option for remotely managing a Network Deployment deployment manager or, more likely, multiple deployment managers, removing the cell boundaries.

3.4 Security

WebSphere Application Server provides a set of features to help you to secure your systems and associated resources. Figure 3-23 illustrates the components that make up the operating environment for security in WebSphere Application Server.
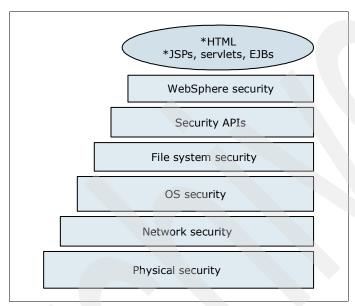
Figure 3-23   WebSphere Application Server security overview

From a broad perspective, a WebSphere security service runs locally in each process (deployment manager, node agent, and application server). This architecture, illustrated in Figure 3-24, distributes the security workload so that one process does not act as a bottleneck for the entire environment. If a security service failure occurs, then only a single process is affected. Additionally, the authentication mechanism and user registry are separated.
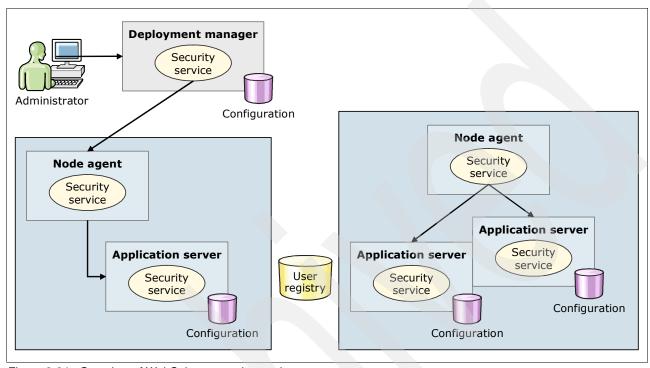
Figure 3-24   Overview of WebSphere security service

3.4.1 Security types

The security infrastructure of WebSphere Application Server is broadly divided into the following types of security (illustrated in Figure 3-25 on page 70), from the Integrated Solutions Console:

► *Administrative security* protects resources such as the Integrated Solutions Console, `wsadmin`, and scripts. When administrative security is enabled, naming security, authentication of HTTP clients, and use of SSL transports are enabled.

► *Application security* protects access to applications. It provides application isolation and requirements for authenticating application users which is done through security constraints to protect servlets and method permissions to protect EJBs. Application security can be applied to resources within an EAR through security roles that are 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. The lower-level, system-level resources that are protected include file I/O, network connections, property files, and so on. Through policy enforcement, overall system integrity can be increased by checking for permissions before allowing access to protected system resources.

Figure 3-25 Types of security

3.4.2 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 (fingerprints, retina scan, DNA, and so on).

The authentication mechanism in WebSphere Application Server typically collaborates closely with a user registry, as illustrated in Figure 3-26. When performing 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.
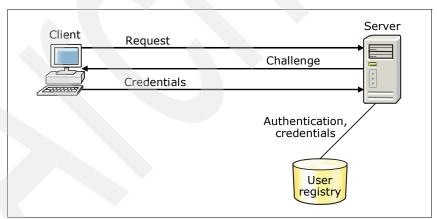
Figure 3-26 Simplified view of authentication

Although WebSphere Application Server provides support for multiple authentication mechanisms, you can configure only a single active authentication mechanism at a time. WebSphere Application Server supports the following authentication mechanisms:

► Lightweight Third Party Authentication (LTPA)

LTPA is intended for distributed, multiple application server and machine environments. It supports forwardable credentials and single sign-on (SSO). LTPA can support security in a distributed environment through cryptography. With this support, LTPA can encrypt,

digitally sign, and securely transmit authentication-related data, and later decrypt and 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 only be used for administrative requests.

Authorization

Authorization is the process of checking whether a given user has the privileges necessary to get access to or perform action on a requested resource (web pages, servlets, JSPs, and EJBs), as illustrated in Figure 3-27. Authorization controls access to resources through the following mechanisms:

► *Security lookup* determines the security privileges for a given user. This information is stored in a user registry.

► *Rule enforcement* obtains rules from a registry. Given the privileges of a given user and rules, access is determined.
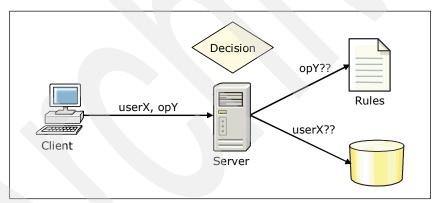
Figure 3-27   Simplified view of authorization

3.5 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 referred to as the *default messaging provider*.

► The service integration bus (referred to as the *bus* or *SIBus*)

The SIBus provides the communication infrastructure for the default messaging provider. The bus 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, so that you can control which web services are available to different groups of web service users.

3.5.1 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.

3.5.2 Service integration bus

The *service integration 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 3-28 illustrates the service integration bus and how it fits into the larger picture of an enterprise service bus.
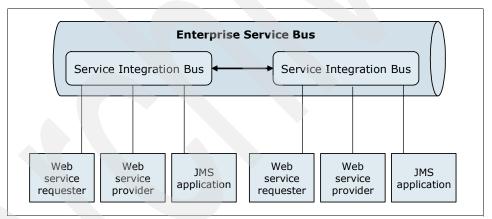
Figure 3-28 Service integration bus basics

The service integration bus supports the following application attachments:

► JMS messaging applications

JMS applications running 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 attaching through SOAP/HTTP or SOAP/JMS

3.5.3 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 3-29. 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)

► Performs protocol transformation, such as submitting incoming SOAP/HTTP messages to SOAP/JMS
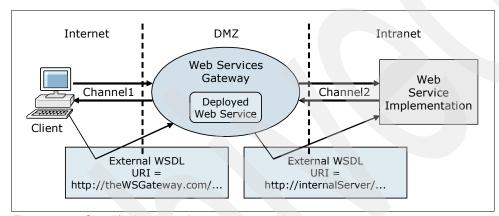
Figure 3-29 Simplified web services gateway architecture

The gateway acts as a proxy so that gateway service users do not need to know if 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.

3.6 Clusters and the application server cluster

A *cluster* is a collection of servers that is managed together. With clusters, enterprise applications can scale beyond the amount of throughput that an be achieved with a single application server. Also, enterprise applications can be 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 machines. 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

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, one node at a time, providing continuous availability of the application to the user.

Application server clusters have the following important characteristics:

► A cell can have multiple or no clusters.

► 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 given cluster. All cluster members in a cluster must be in the same node group.

3.6.1 Vertical cluster

All cluster members can reside on the same system. This topology is known as *vertical scaling* or *vertical clustering*. Figure 3-30 illustrates a simple example of a vertical cluster.
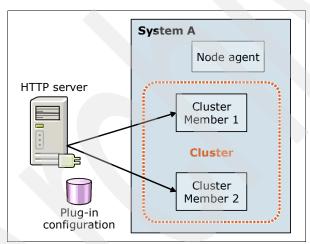
Figure 3-30   Vertical cluster

Vertical clusters offer failover support inside one operating system image, provide processor level failover, and increase resource utilization. For more information about vertical clusters and vertical scaling, see 5.3.3, "Vertical scaling topology" on page 125.

3.6.2 Horizontal clustering (scaling)

Horizontal scaling or *horizontal clustering* refers to cluster members that are spread across different server systems and operating system types (Figure 3-31). In this topology, each machine has a node in the cell that is holding a cluster member. The combination of vertical and horizontal scaling is also possible.
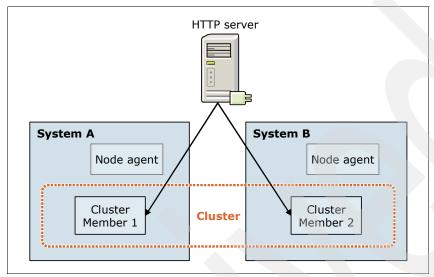
Figure 3-31 Horizontal cluster

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 machine failover. For more information about horizontal clusters and horizontal scaling, see 5.3.4, "Horizontal scaling topology" on page 127.

3.6.3 Mixed clustering

Figure 3-32 on page 76 illustrates a cluster that has four cluster members and is an example of combining vertical and horizontal clustering. The cluster uses multiple members inside one operating system image (on one machine) and that are spread over multiple physical machines. This configuration provides a mix of failover and performance.

This cluster also shows the following characteristics:

► Cluster members can reside in multiple nodes.
► Only some of the application servers in a node are part of the cluster.
► Cluster members cannot span cells.
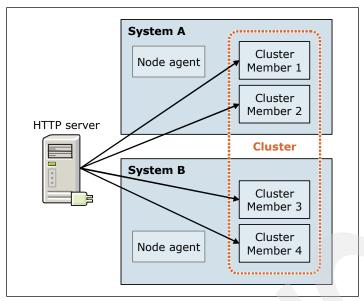
Figure 3-32   Vertical and horizontal clustering

3.6.4 Mixed-node versions in a cluster

A WebSphere Application Server Network Deployment V8 cluster can contain nodes and application servers from WebSphere Application Server V6, V6.1, V7, and V8. The topology illustrated in Figure 3-33 contains mixed version nodes within a cluster.

You can upgrade any node in the cell, while leaving the others at a previous release level. Consider using this feature only for migration scenarios.
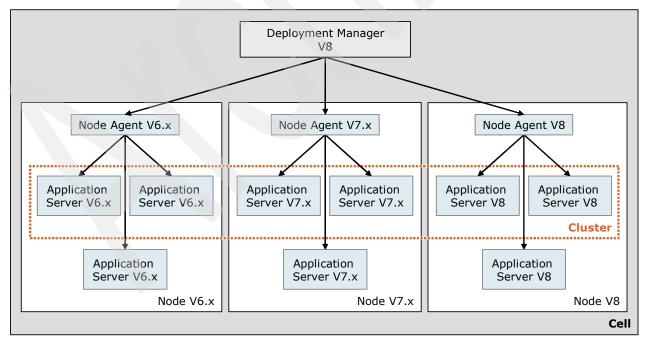
Figure 3-33   Mixed version nodes clustered in a cell

3.6.5 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, which is 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.)

Use an HTTP traffic-handling device, such as IBM HTTP Server. This method 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 based on the set of server weights that is assigned to the members of a cluster. In the case of horizontal clustering, where each node resides on a separate server system, the loss of one server system will not disrupt the flow of requests, because the deployment manager is used only for administrative tasks. 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 can be performed by configuring the web container and EJB containers on separate application servers. Multiple application servers can be clustered with the EJB containers, 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 to decide on which member a transaction should be processed.

> **Workload Manager and Sysplex Distributor:** 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 keeps track of 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.

It is possible to classify incoming requests in 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 processing the higher prioritized requests gets additional resources (such as CPU), thus protecting the response time of your most important work.

WLM changes: The WLM component can change the amount of CPU, 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 keeps its defined performance targets, and whether more important work is in the system. This technique is performed 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 14.1.7, "Workload management for WebSphere Application Server for z/OS" on page 392.

3.6.6 High availability

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

On the z/OS platform, the WebSphere Application Server V8.0 high availability manager uses native z/OS cluster technology, the cross-system coupling (XCF) services. This technology reduces the amount of processing used for the keep-alive check of clusters, while improving the time it takes to detect a failed member.

XCF services: 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 14.1.9, "XCF support for WebSphere HA manager" on page 399.

3.6.7 Core groups

In a high availability environment, a group of clusters can be defined as a 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. With the use of core groups, WebSphere Application Server can provide high availability for applications that must always be available to the user. You can also configure core groups to communicate with each other by using the *core group bridge*. The core groups can communicate within the same cell or across cells.

3.7 Run times

This section briefly explains how WebSphere Application Server processes execute 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.

3.7.1 Distributed platforms

On *distributed platforms* (AIX systems, Linux systems, Windows systems, and other operating systems), 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, as illustrated in Figure 3-34. 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 seen as a `java.exe` process.
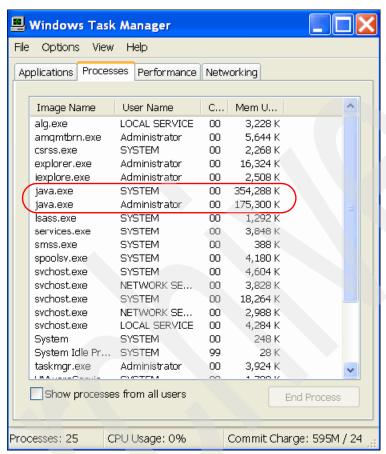
Figure 3-34   Deployment manager and server Java processes on a Windows system

3.7.2 WebSphere Application Server on z/OS

WebSphere Application Server for z/OS uses multiple runtime components to form the different WebSphere Application Server parts (such as the application server, deployment manager, and other components). One part, the *controller region*, runs some basic WebSphere Application Server functions, while the other part, the *servant region*, houses applications.

The z/OS address spaces option creates multiple processes to run applications. This form of mini-cluster, the *multi-servant cluster*, 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 14.1.6, "Runtime processes" on page 391.

4

Infrastructure

Planning and designing an infrastructure for a WebSphere Application Server environment requires considerations of many actions. This chapter describes the most important aspects to create a WebSphere Application Server infrastructure to run a successful WebSphere project.

This chapter includes the following sections:

- ► Infrastructure planning
- ► Environment planning
- ► Design considerations
- ► Sizing the infrastructure
- ► Performance considerations
- ► Monitoring
- ► Backup and recovery
- ► Cloud infrastructure

Terminology: The terms *machine* and *system* are synonyms for physical machines, logical partitions (LPARs), and operating system image.

81

4.1 Infrastructure planning

This section gives a general overview of the typical phases you have to go through during a project. It explains how to gather requirements and how to apply these 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 cater to 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, or 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 process time for batch.

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. Target developing the following designs:

▶ Application design

  To create your application design, use your functional and non-functional requirements to create guidelines for your application developers about how your application is built.

▶ Implementation design

  This design defines the target deployment infrastructure on which your 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, such as a database, a set of application servers, a set of web servers, and other components that are defined in the requirements phase. You can find helpful infrastructure design concepts in 4.3, "Design considerations" on page 84.

With these two draft designs, you can begin the process of formulating counts of servers, network requirements, and the other items related to the infrastructure. For more information, see 4.4, "Sizing the infrastructure" on page 91.

The last step in every deployment is to tune your system and measure whether it can handle the projected load that your nonfunctional requirements specified. For more details about how to plan for load tests, see 4.5, "Performance considerations" on page 92.

4.2 Environment planning

Your new 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

The environment is reserved for application developers to develop the future applications. This environment is often a simple stand-alone machine without high availability.

► Deployment

The environment is dedicated to application developers and integration team to test the application deployment procedure and the applications in a simple, highly available environment. This environment has to contain a minimum of high availability components to be sure that the applications are compatible with them.

► Technical qualification

This environment is reserved for the infrastructure team to develop and test the technical procedures, for example, backup and recovery and daily maintenance operations. It is also used to test the new hardware or middleware patches. Usually, the infrastructure team begins to build this environment to ensure all of the components are compatible and to create the 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 that test concurrently.

► Performance

The performance environment must mirror the production environment as closely as possible. The goal of this environment is to perform tuning and test 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 a temporary substitution solution.

► Production

The production environment is the final stage that is dedicated to run the business applications and server user requests. It is the most important environment of your infrastructure. You must clearly and strictly define specific rules and procedures to manage it.

> **Important:** 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 machines. At least, 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.

4.3 Design considerations

This section provides information about key infrastructure concepts to consider when designing a WebSphere Application Server environment. These concepts will significantly impact your design. This section includes the following topics:

- ► Scalability
- ► High availability
- ► Load balancing and failover
- ► Caching
- ► Security
- ► Application deployment

4.3.1 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 always linked to high availability and performance.

To determine your key infrastructure components and to identify scaling techniques that are applicable to your environment, follow these steps:

1. Understand the application environment.

   Applications are key to the scalability of the infrastructure. Ensure that the applications are designed for scaling. It is important to understand the component flow and traffic volumes that are associated with existing applications and to evaluate the nature of new applications. It is essential to 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, needs to focus on transaction performance and the scalability of databases that contain customer information that is used across sessions. These considerations might not typically be significant for a publish/subscribe site, where a user signs up for data to be sent to them, 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 key components of the infrastructure are the load balancers, the application servers, security services, transaction and data servers, and 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 most important components you determined in the previous step, you have different scaling approaches, which might include the following approaches:

   – Scaling up (or vertical scalability)

     Scaling up is done inside a component; you have to add more resources (such as memory and CPU) to this component to handle the load.

   – Scaling out (or horizontal scalability)

     Scaling out is an alternative to scaling up and means increasing the number of instances of the component. For example, instead of multiply by two the number of processors and memory to a machine, you keep your first machine and you add a second identical machine to the first one.

     Figure 4-1 illustrates the difference between scaling up and scaling out.
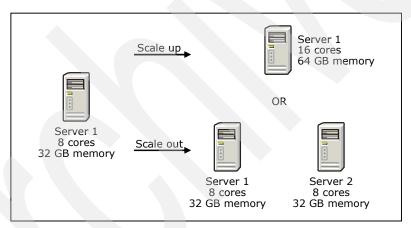
Figure 4-1   Scale up and scale out approaches

   – Using appliance servers

     You can use a dedicated appliance to perform a specific action for your workload. These machines 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 world regions.

   – 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 usually run during low peak hours. WebSphere Application Server V8 supports the development and deployment of Java batch applications.

- Aggregating user data

    To avoid applications accessing customer data from multiple existing applications, aggregate this data in a single back-end system and, thus, limit the number of connections to multiple systems.

- Managing connection

    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 the components' 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 should 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.

    > **Important:** 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 will need 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 will require examination of the machine configurations. Scalability is not a one-time design consideration. It is part of the growth of the environment.

For detailed information about this procedure, see the article "Design for Scalability: An Update" in developerWorks at:

http://www.ibm.com/developerworks/websphere/library/techarticles/hipods/scalability.html

For more information about some of the scaling techniques, see 7.2, "Scalability" on page 200.

4.3.2 High availability

Designing an infrastructure for high availability means that your environment can survive the failure of one or multiple components. High availability implies 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 the specific environment.

The following list helps to identify the high availability needs in your 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, high availability will increase 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. Carefully gather where and what type of high availability will be required to meet the service-level agreements (SLA) and non-functional requirements.

► After you gather the high availability requirements, review every component of the implementation design you developed in 4.1, "Infrastructure planning" on page 82. Determine how significant each component is for the availability of the service and how a failure might impact 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 will have an impact on the investments you are willing to take, 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 are in the architecture that 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. You must 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 using 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 are done with your high availability design, update your implementation design to include the high availability features.

4.3.3 Load balancing and failover

As a result of the design considerations for high availability, you might identify several components that need redundancy. Having redundant systems in an architecture requires you to consider how to implement redundancy to ensure getting 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, it is determined automatically by the system and transparently rejoins 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 and how you achieve these capabilities, additional hardware and software are 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.

4.3.4 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, that 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 third-party caching devices or external caching infrastructures provided by IBM and third parties. IBM provides the following caching software and appliance:

- 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 and 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.

4.3.5 Security

WebSphere Application Server V8 is a powerful Java language-based middleware application server that is ready to run your important applications and business transactions. For this reason, you must plan the security aspect in the beginning of your project. Do not underestimate the importance of security on your infrastructure design. Any change on this part might imply a serious redesign. You also have to test the whole security chain in your project as soon as possible. Choose the right level of security that will be required, remembering that security has an impact on performance.

Consider how security will affect your infrastructure:

- Understand the security policy and requirements for your future environment.
- Work with a security subject matter expert to develop a security infrastructure that adheres to the requirements and integrates in the existing infrastructure.
- Make sure that sufficient physical security is in place.
- Make sure that application developers understand the security requirements and code the application accordingly.
- Consider the user registry (or registries) that you plan to use. WebSphere Application Server V8 supports multiple user registries and multiple security domains.
- Make sure that the user registries do not break the high availability requirements. If the user registries that you are using are out of scope of the WebSphere Application Server project, consider the use of high availability and request it if needed. For example, make sure that your LDAP user registries are made highly available and are not a SPOF.
- Assess your current implementation design, and ensure that every possible access to your systems is secured.
- Consider the level of auditing required and how to implement it.

- ► Consider how you will secure stored data. Think of operating system security and the encryption of stored data.

- ► Define a password policy, including considerations for handling password expirations for individual users.

- ► Define the encryption (Secure Sockets Layer (SSL)) endpoints in your communications.

- ► Plan for certificates management such as expiration and secure storage.

After you finish your security considerations, update your implementation design to include all security-related components.

4.3.6 Application deployment

When planning for WebSphere Application Server infrastructure, do not delay application deployment thoughts until the application is ready to deploy. Start this planning task when you start designing your infrastructure. The way you deploy your application affects your infrastructure design in various ways:

- ► Performance.

   The way you deploy your application can affect performance significantly. Tests showed significant performance differences when deploying Enterprise Java Bean (EJB) modules to the same application server as the EJB client components. These differences were compared to deploying EJB modules to a separate application server than the EJB client components.

   However, deploying EJB modules to separate application servers provides more flexibility and avoids duplication of code.

- ► Number of application servers

   When you plan the deployment of multiple applications in your environment, you can either deploy multiple applications to the same application server or create one server for each application. Depending on that decision, you might end up with a higher number of application servers in your cell. Therefore, you will have a larger cell, which implies increased server startup times and complexity for your environment due to larger high availability views.

- ► Cluster communication

   The higher the number of application servers is, the higher communications and systems management are in your environment.

- ► Platforms

   WebSphere Application Server supports building cross-platform cells. Therefore, you can create one cell that contains servers on multiple platforms. Although this option increases the complexity of the environment and makes the administration more complicated, it provides flexibility when it comes to the integration of existing software components.

For more information about considerations when planning to deploy applications to the same or different servers, see 8.12, "Mapping applications to application servers" on page 268. After you know how your applications will be deployed, update your implementation design.

4.4 Sizing the infrastructure

After determining the initial application and infrastructure design, you need to determine the system resources required for the project to keep the SLAs.

You have to consider which hardware platforms you want to use. This decision depends on the following factors:

► The scaling capabilities of the platform
► The platforms that WebSphere Application Server supports
► The performance, security, and high availability requirements of your environment
► Integration with your own current infrastructure
► Scaling techniques: horizontal, vertical scalability, or others.

Sizing estimates are based solely 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: response time, throughput, or others?

► Is your workload steady, or does it peak? If it peaks, for which particular components?

Calculations based on the answers are performed to determine the estimated amount of hardware your infrastructure will require.

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. These results are obtained by running a simple, common workload on several platforms and give you an idea of the performance of the different machines. These reports, your experience, and your application inputs can help you make a decision.

The IBM Workload Estimator (WLE) tool helps you to size your infrastructure. Go to 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 the rPerf, TPC, and SPEC 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

4.5 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 runs campaign is mandatory to determine whether your infrastructure is well 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, but performance problems are unacceptable to most users and affect everyone 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 this activity.

4.5.1 Application design issues

Many performance problems cannot be fixed by using more hardware or tuning WebSphere parameters. As such, you have to make performance testing and tuning part of your development process and release cycles to avoid problems later. Performance testing and tuning must be considered in the project schedule.

> **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. Usually, many hot spots can 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, resulting in fewer issues when using your application on the production environment.

4.5.2 Requirements understanding

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. 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.

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.

4.5.3 Tips for setting up the test environment

When executing 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 and 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 (for example, storage box, virtual Ethernet, or Fibre Channel), make sure no one is using it during the performance runs period. For example, if another application uses the same storage box at the same time, your disk response times will be higher, and the whole 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, because 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 usually by more than 15% depending on the log.

– Production monitoring

After identifying and solving performance issues using the debug monitoring level, perform a test with the same set of monitoring options that are later used in your target environment. With this setting, you should be able to satisfy the 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 performing 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, you should not run your tests with your caches empty. You can complete 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.

4.5.4 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 depending 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 at a point in time. This number of users is actively sending requests to your system at a given point in 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

At this point in the project, you must 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 is important because it 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 performed.

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.

4.5.5 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 have planned 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 have to 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. Then, you have to rerun the performance tests and redo the same process until your performance requirements are met.

  Following this process will help you to resolve your main bottlenecks.

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 4-2 summarizes the performance approach.
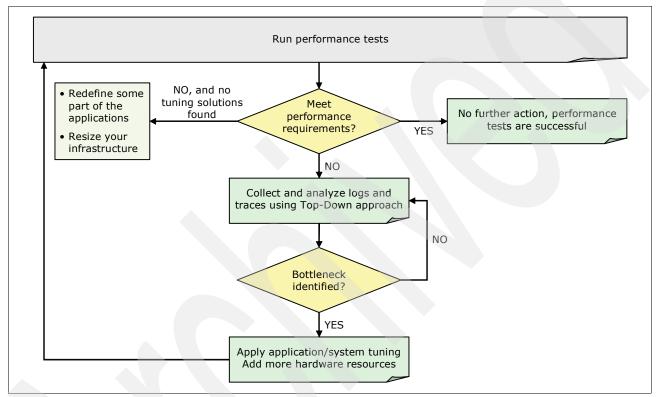
Figure 4-2   Performance approach

When your performance campaign is finished, you can carefully update your production environment.

4.5.6 Production system tuning

At the end of your tuning process and after you find the correct performance, scalability, and high availability combination between the application, middleware, and system, you must upgrade your production environment.

When changing a production environment, the following practices are among the standard ones:

► Change only one parameter at a time.
► Document all changes.
► Compare several test runs to the baseline.

Important: Keep in mind that you often only have one chance to get performance tuning right. Normally, as soon as your environment is in production, you cannot run other performance tests on this environment, because you cannot take the production system offline to run more performance tests. If a production system is being tested, it is likely that the system is running in a severely degraded position.

4.6 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, and users usually navigate to the next site if your site is not operable, meaning you lose a potential customer. Therefore, you must 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.

4.6.1 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 single component is overseen.

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

To provide efficient 24×7 monitoring, you are required to use monitoring software. Many organizations have some monitoring infrastructure already set. Determine if you can integrate your new WebSphere Application Server infrastructure on 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 your environment in almost all aspects. 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 keep track of 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, 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 popular 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.

4.6.2 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 unacceptable impact to 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, the more you

monitor, and the shorter the intervals are between your monitoring cycles, the higher the overhead you generate. 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 impact your environment. Even if something is wrong in the monitoring infrastructure, monitoring must never be the cause for any service outage.

4.6.3 Alerting and problem resolution

Monitoring alone is not enough to keep track of the health of your environment, because monitoring does not solve issues. You improve availability if you combine monitoring with proper alerting to the responsible problem resolvers. What is the use of monitoring if nobody knows that there is a problem? Consider the following thoughts among others 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.

4.6.4 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.

4.7 Backup and recovery

In general, IT hardware and software are reliable, but sometimes failures can occur and damage a machine, 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.

4.7.1 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 are prioritized 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 your components by risk level and then decide which backup or recovery techniques to use.

4.7.2 Recovery strategy

When critical areas have been identified, 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 proper strategy is to find the proper balance between recovery time and cost. The business impact is the determining factor in finding the proper balance. Business-critical processes need quick recovery time to minimize business losses. Therefore, the recovery costs are greater.

4.7.3 Backup plan

With 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 to 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 them. 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 just 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 performed 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 transparent to the users, thanks to the load balancing and the backup online.

4.7.4 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.

4.7.5 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 have to 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.

4.8 Cloud infrastructure

WebSphere Application Server is also available in both public and private clouds.

4.8.1 Public cloud

A public cloud is offered as a service for companies. Customers do not need to manage the whole infrastructure stack, but they must 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 only pay for what you need. If you need to test new application features for a few weeks and you do not have sufficient hardware, you do not need to buy hardware for short-time usage.

► 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; see the following website), which provides WebSphere Application Server images:

    http://aws.amazon.com/ec2/

4.8.2 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 appliance is a hardware appliance that provides access to IBM middleware virtual images and patterns 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), that 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 easily and rapidly build and manage a customized WebSphere Application Server infrastructure. For more information, see the IBM Workload Deployer product page at:

    http://www.ibm.com/software/webservers/workload-deployer

5

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 right topology for your environment, several aspects of your business must be taken into account. Keep in mind that choosing the right, flexible, and scalable topology from the beginning can determine the success of the WebSphere Application Server implementation.

This chapter addresses the most widely used topologies according to business size and needs. It offers the necessary information to help understand the different components that are involved in a topology and the best way to implement them according to the business needs.

This chapter includes the following sections:

► Topology selection criteria
► Terminology
► Topologies in detail

103

5.1 Terminology

Before you examine the topologies, you must understand and become familiar with the terminology that is highlighted in this section. These elements are in the diagrams that describe each topology later in this chapter.

5.1.1 Load balancers

A *load balancer*, also referred to as 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 the Network Deployment package. 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 overhead
- ▶ Load balancing of a private network
- ▶ Scalability

For more information about these features and the functions of Load Balancer, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *developing applications that use programmatic security*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

> **Sysplex Distributor on z/OS:** 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.

5.1.2 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 your servers from the clients. The caching proxy products provided by WebSphere Application Server V8 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, allowing faster response and decreasing the load on the servers and the internal network.

> **Stabilized:** Stabilized means that no new features will be delivered, but new platforms will be supported.

Edge proxy
The caching proxy provided with WebSphere Application Server V8.0 with the Edge Components can be configured as a reverse and a forwarding proxy. This proxy server supports the protocols HTTP, HTTPS, FTP, and Gopher.

DMZ secure proxy
With *DMZ secure proxy*, you can install proxy servers in the DMZ with reduced security risk, 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 functionality. 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 Network Deployment manager cell.

For the following features, you can select levels of security (low, medium, or high), but you can also customize these features, depending 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.

> **Enhancing protection:** You can also switch from a Java Development Kit (JDK) to a Java Runtime Environment (JRE) when using the DMZ secure proxy. This way removes the inclusion of a compiler on the installation, which can 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 and has access to cell information and the current state of all servers and applications inside the cell.

The current proxy server does not provide the following functions that were available in existing versions of the proxy server:

► Authentication and authorization on the proxy server
► Cache sharing
► Common Gateway Interface (CGI)
► Custom logging
► Forward proxy configuration, including SSL tunneling, transparent proxy, and FTP protocol
► National Center for Supercomputing Applications (NCSA) combined logging
► Request Uniform Resource Identifier (URI) rewrite (response URI rewrite is available)

5.1.3 Domain and protocol firewall

A *firewall* is a hardware and software 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, if those viruses attacks have to cross network boundaries protected by the firewall. Another typical usage of firewalls is to prevent denial-of-service (DoS) attacks against services.

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 role of a 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 increasing levels of protection at the expense of increasing computing resource requirements. The protocol firewall is typically implemented as an IP router.

5.1.4 Web servers and WebSphere Application Server plug-in

Most WebSphere Application Server topologies have a web server that receives HTTP-based 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 functionality in the topology:

► It serves requests for static HTTP content such as HTML files and images.

► Requests for dynamic content, such as JSPs, servlets, and portlets, are forwarded to the appropriate WebSphere Application Server through the WebSphere Application Server plug-in.

- ► The WebSphere plug-in offers load balancing and failover functionality between the application servers in the 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 ESI cache.
- ► It breaks 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.

> **More information:** For information about the web servers that are supported by WebSphere Application Server V8, see System Requirements for WebSphere Application Server Base and Network Deployment V8.0 at:
>
> 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 (for example, a new application is installed), the plug-in must be regenerated and propagated to the web server machine again.

If the plug-in configuration service is enabled (which it is by 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.

> **Stand-alone topology:** In a stand-alone topology, only unmanaged web servers are possible, meaning that the plug-in must be manually pushed out to the web server system. The exception is if you are using IBM HTTP Server. The application server can automatically propagate the plug-in configuration file to IBM HTTP Server, even though it is an unmanaged node, by using the administrative instance of IBM HTTP Server.

WebSphere Application Server V8 ships with IBM HTTP Server V8 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 Toolkit (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, see the WebSphere Application Server Version 8 Information Center at the following address, and search for these topics:

- ► *What is new in this release*:
- ► *Apache modules (containing directives) supported by IBM HTTP Server*

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

5.1.5 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 functionality offered by WebSphere Application Server, various profiles types are available. Some of the profiles 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.

5.1.6 Directory and security services

Directory and security services supply information about the location, capabilities, and attributes (including user ID and password pairs and certificates) of resources and users known to this WebSphere Application Server environment. This node can supply information for various security services (authentication and authorization) and can perform the actual security processing, for example, to verify certificates.

An example of a product that provides directory services is IBM Tivoli Directory Server, included in the Network Deployment package.

5.1.7 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 embedded messaging provider in WebSphere can run in any user-created application server.

For more information, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *messaging resources*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

5.1.8 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 running 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), a transaction monitor such as CICS and a web service.

5.2 Topology selection criteria

Regardless of the size of your business, you must carefully perform the topology selection and include the right people during its planning. WebSphere Application Server specialists are not the only ones who are responsible for choosing the right topology. Depending on your environment, security, networking or hardware specialists, and developers can also give valuable feedback about the correct topology.

This section provides an overview of the most important criteria for selecting a topology:

- ► High availability
- ► Disaster recovery
- ► Security
- ► Performance
- ► Scalability
- ► Application deployment
- ► Summary of topology selection criteria

5.2.1 High availability

High availability means that a system can tolerate a certain amount of failure and errors, but remain operational. Depending on which component of the topology failed, it is possible that the system continues to run but 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 physically separated nodes. If the hardware hosting one of the nodes fails, the application continues to run, but now the cluster can process half the workload it normally does. Keep this scenario in mind 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 involved in the topology.

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 power supplies, network cables, switches, processes, and machines). WebSphere Application Server Network Deployment provides various options to provide a highly available runtime environment for your applications. 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, see 5.2.5, "Scalability" on page 114. Often systems depend on other external systems that are beyond your control. If you experience this situation, consider alternatives for getting the data these systems provide, if possible, in case they run 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 another operating 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 the article "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 impact 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 machine 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 the Network Deployment package provide these features.

> **Important:** For Edge Component Load Balancer, to stop distributing a load across nonresponding web servers, the Advisor feature must be configured. By default, after the product is installed, this feature is not enabled. For information about how to configure this feature, see the WebSphere Application Server Version 8 Information Center at the following address and search for *developing applications that use programmatic security*:
>
> http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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. Proper tuning of the plug-in can help detect failures or problems in the application servers more efficiently.

► The Enterprise JavaBeans (EJBs) workload management mechanism, which is built into WebSphere Application Server to balance EJB workload across cluster members.

► The usage of partitioned queues. If the application permits, 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 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.

The following new features are provided in WebSphere Application Server V8 for high availability and failover:

► High availability support for an optimized local adapter

This support offers the possibility to specify an alternate connection factory Java Naming and Directory Interface (JNDI) name in case the primary one fails.

► External high availability frameworks and service integration

This feature makes it possible to specify alternate connection names for the link sender channel of WebSphere MQ. If a failure occurs in the active gateway queue, the service integration bus reconnects to a standby gateway queue manager by using that information.

► Resource workload routing

With this feature, resources, such as data sources and connection factories, can fail over and then fail back from previously defined backup resources. The backup resource must be compatible with all applications by using the primary resource. It is created the same way that the primary resource is, but applications can only use it 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.

5.2.2 Disaster recovery

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 your data remains consistent

For example, *cluster isolation* is a potential threat to your data consistency that you must avoid in all circumstances. After data consistency issues are resolved (problems with WebSphere data, such as the configuration repository and logs), you can resume your planning for WebSphere disaster recovery.

> **Cluster isolation:** In this context, cluster isolation refers to a condition where each member of a clustered system considers the other member to be gone and, therefore, taking over the service. The result is to have two systems that manipulate data.

No common solution exists for WebSphere Application Server in a disaster recovery scenario, because it depends on the existing environment, requirements, applications, and budget. Try to avoid running your cell across different data centers, because this approach can compromise performance. Also, the deployment might depend heavily on network reliability. Alternatively, 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 the IBM developerWorks article "Everything you always wanted to know about WebSphere Application Server but were afraid to ask, Part 5," at:

http://www.ibm.com/developerworks/websphere/techjournal/0707_col_alcott/0707_col_alcott.html

5.2.3 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.

Depending on your industry, you must comply with certain regulations. 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 includes tasks to protect 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 5-1 on page 113. From a security point of view, this architecture offers the benefit that, if a security breach occurs in one of the tiers, only that level is compromised, not the entire system. 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 and another firewall that filters information that the web servers forward to application servers in the second tier. This concept is known as a *demilitarized zone* (DMZ). The primary objective of the DMZ is to protect sensitive business logic or information that is hosted in the application servers or databases from possible attacks from untrusted networks on the Internet.
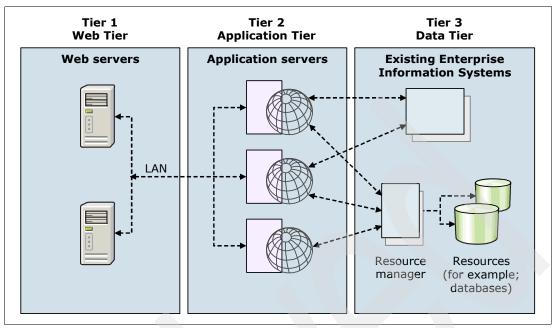
Figure 5-1   Three tier topology

Before you select and finalize a topology, for more information that can be useful for your scenario, see 4.3.5, "Security" on page 89, and Chapter 12, "Security" on page 349.

5.2.4 Performance

Performance determines the ability of an environment to process work in a given interval. The higher the performance is, the smaller the interval is that is needed to process a specified set of work, or the more work that can be processed in the same interval.

Choosing the right topology can help in getting good performance for your system. The most important aspect when planning for performance is to ensure that the applications running 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 should 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 and when the projected production load is used 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 as simply as possible. A multitier design can offer better throughput for heavy loads. Nevertheless, keep in mind that a large topology with many layers can pay a slight performance penalty 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 more information about how a single-tier topology performance differs from a split-tier topology, see the WebSphere Application Server V8 Information Center at the following website, and search for *choosing a topology for better performance*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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 and are available to handle incoming requests. Clustering generally provides improvements for performance, due to an optimized scaling.

5.2.5 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 machines to add processing power, improve security, maximize availability, and balance workloads. Scaling can be vertical or horizontal, as illustrated in Figure 5-2.
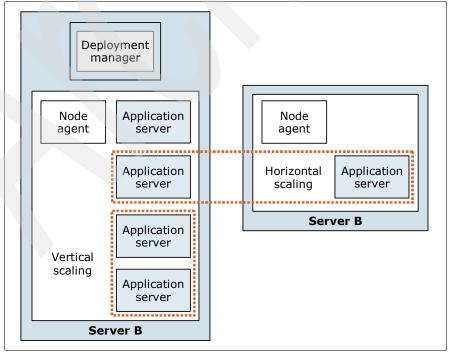
Figure 5-2   Vertical and horizontal scaling with WebSphere Application Server

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 machine or z/OS image, providing application server failover and load balancing across multiple application servers. This topology does not provide efficient fault tolerance because a failure of the operational system or the hardware on the physical machine might cause problems for all servers in the cluster.

► Horizontal scaling

Horizontal scaling involves creating application servers on multiple machines to take advantage of the additional processing capacity available on each machine. Using horizontal scaling techniques also provides hardware failover support.

> **Tip:** When planing horizontal scaling, you must decide if 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, because they need enough resources available to handle the workload that corresponds to the failing node.

When implementing horizontal scalability, using multiple smaller machines can be more cost effective. However, when planning to use vertical scalability, make sure that the machines have enough available resources to host multiple processes. Bigger machines can be more expensive, but vertical scalability takes advantage of the investment in resources.

Remember that, 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, as explained in "Failover support" on page 110, with your scalability schema for a more resilient environment.

Consider the following guidance when planing 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 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. Although the clients behave as though they are communicating directly with one web server, 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 and other TCP/IP-based servers whose protocol is understood by the IP sprayer.

5.2.6 Application deployment

Various application deployment decisions affect topology decisions. Clarify the following application deployment-related considerations before finalizing the architecture.

EJB deployment

The way you deploy EJB can significantly impact your application topology and the performance you expect. You can choose to deploy EJB to the same application servers and clusters as the client modules invoking the EJB or to separate dedicated application servers and clusters running the EJB only. Both of these options are valid approaches depending on your environment and requirements. For the explanation in this section, the EJB provide local and remote interfaces to be invoked.

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.

► Different tuning is required between enterprise applications and EJB.

► You need a faster start of the application servers that are hosting the web applications, because the EJB container is loaded only where the EJB are deployed.

► Optimize the utilization 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, it is probably better to deploy your EJB and enterprise applications on the same server:

► Performance issues with having both EJB and enterprise applications on the same server. Deploying your EJB on another server slows performance because a lot of serialization and deserialization must be done when sending the data through the network.

► Inexistent or incipient development methodology, because all enterprise applications must be compatible with the existing version of deployed EJB. This situation requires a lot of coordination during the development phase.

► Memory footprint issues on your environment. Additional servers for the EJB demand more memory to run.

► Complex cell. More servers to manage can make administration less practical.

► A lack of visibility of your deployed applications and how the EJB upgrades might affect them.

Assignment of applications to clusters

When you are running multiple applications in your environment, you must 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 avoid critical applications 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, you must 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.

If the following scenarios apply to your environment, carefully decide whether one application per server can be beneficial to your topology:

► Large environments with complex cell administration. A cluster or server per application will increase its complexity. It can also increase start times while the high availability managers establish connectivity across the cell.

► Performance issues. You are likely to experience slowed performance when calling EJB, because this process is an out-of-process call if the EJB have their own application server.

► Memory footprint issues. 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 should run. Whether you run the embedded messaging service on a separate application server and cluster or co-located on an application server running your applications depends on your needs.

To determine what is best for your environment, consider the advantages and disadvantages of running 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, focusing 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.

If the following scenarios apply to your environment, carefully choose whether locating the message infrastructure on separate servers will be beneficial to your topology:

- ► Large environments with complex cell administration. More servers for the messaging infrastructure will increase its complexity. They can also increase startup times while the high availability managers establish connectivity across the cell.

- ► Memory footprint issues. Each JVM has a basic memory footprint. More application servers will increase the overall footprint.

- ► Use of mediation modules. The deployment can become more complex by using this approach.

5.2.7 Summary of topology selection criteria

Table 5-1, Table 5-2 on page 119, and Table 5-3 on page 119 list the requirements for topology selection and possible solutions. Table 5-1 summarizes topology selection based on availability requirements.

Table 5-1   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) - 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 5-2 summarizes topology selection based on performance requirements.

Table 5-2   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 Advanced Fast Path Architecture (AFPA) or Edge Side Include (ESI) caching for external caching ► WebSphere plug-in tuning
Application server	► Clustering (in most cases horizontal) ► Deploy EJB to the same JVM as the invoking 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[a] ► 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

a. For more information, see *WebSphere Application Server V5: Separating Static and Dynamic Web Content*, TIPS0223.

Table 5-3 summarizes topology selection based on security requirements.

Table 5-3   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 depending on network security.

5.3 Topologies in detail

Because of the vast amount of configuration possibilities, WebSphere Application Server provides many configuration 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 or 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 close as possible. In some cases, testing applications on single server environments can have different results when tested on a distributed cell depending 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, you can create profiles graphically by using the Profile Management Tool or by using 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 the Linux for zSeries platforms, x86-based Linux and Windows platforms, Linux on Power PC platforms, and AIX Power PC platforms. 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 for WebSphere Application Server for z/OS, see 14.1.7, "Workload management for WebSphere Application Server for z/OS" on page 392.

5.3.1 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 and that you can use it for caching purposes.

Application server

A *stand-alone server topology* refers to the installation of WebSphere Application Server on one single (physical) machine 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.

> **Application image:** 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 multiservant regions, which are best thought of as an application cluster to build each application server. Multiservant regions provide one application image to the user while running multiple, independent instances of the application.
>
> The system administrator can determine if multiple application images are used. For more information, see 14.1.5, "Structure of an application server" on page 389.

Web server

Although you can install the web server on the same system as WebSphere Application Server, you need to have a web server in a DMZ as a front end to receive requests. The web server is in a DMZ to provide security, performance, throughput, availability, and maintainability, while the application server that contains business logic is securely in a

separate network. You can also direct requests to the application server, but it is discouraged for production environments.

Figure 5-3 illustrates a stand-alone topology with a web server in a DMZ.
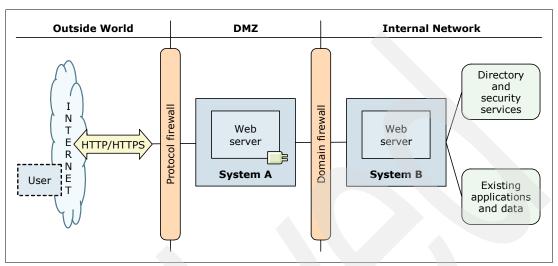
Figure 5-3   Stand-alone server topology with web server in a DMZ

Advantages

The stand-alone server topology has the following advantages:

► It allows the sizing and configuration of 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 the performance of the various components.

► 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 effect the resources (processor, memory, and disk) available to WebSphere. Nor does it affect the ability of WebSphere to service dynamic requests. The web server might have more resources available while serving dynamic content 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 machines 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 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.

Disadvantages

The stand-alone server topology has the following disadvantages:

► It requires additional deployment.

   The plug-in configuration file is generated on the WebSphere Application Server machine. You must copy it to the web server machine each time a configuration change occurs,

which affects requests for applications. Although WebSphere Application Server V8 provides tools to automate this step, not every environment is suitable to use these tools.

► It can have a possible drop of performance.

Depending on the network capacity and the distance of the web server, the network response time for communications between the application server and web server can limit the application response time. 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 to avoid sensitive data being "sniffed" in the network. This additional encryption introduces a performance penalty and increased resource use. From a security perspective, 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.

With the web server running 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 5-3 on page 121, you must install and configure the environment as explained 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:

   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 launch 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 Version 8 Information Center at:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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.

3. Create an application server profile by using the `app_server_root`/profileTemplates/default profile template.

4. Create a web server definition through the Integrated Solutions 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 machine is running Windows, the script is in the `plugins_root/bin/crossPlatformScripts` directory.

> **Web server definition:** 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.

5.3.2 Multiple stand-alone servers topology

The multiple stand-alone servers topology is a variant from the stand-alone server topology (see 5.3.1, "Stand-alone server topology" on page 120). The difference is that you have more than one profile on the same machine, and every profile has its corresponding web server. Figure 5-4 illustrates the multiple stand-alone servers topology. Notice the one-to-one relationship between the application servers and the web servers.
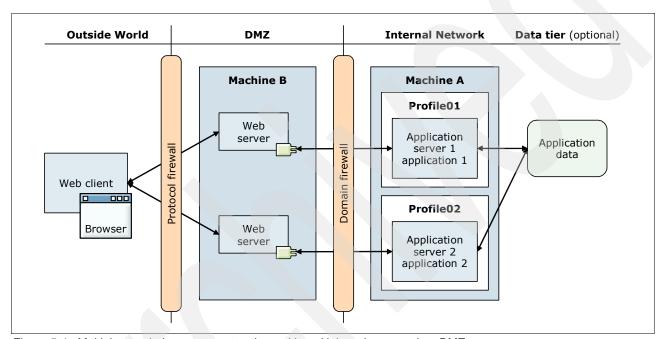
Figure 5-4   Multiple stand-alone servers topology with multiple web servers in a DMZ

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 machine, 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 while sharing the set of core product files. Also, each application has its own JVM that can help customizing 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.

Disadvantages

A multiple stand-alone servers topology has the following disadvantages in addition to the disadvantages that a stand-alone server topology has:

- ► 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 5-4 on page 123, complete the steps as explained 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 launch 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 Version 8 Information Center at:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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.

3. Create an application server profile by using the *app_server_root*/profileTemplates/default profile template.

4. Create a web server definition through the Integrated Solutions 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 machine 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.

5.3.3 Vertical scaling topology

A vertical scaling topology (illustrated in Figure 5-5) refers to a configuration with multiple application servers on a single machine or LPAR and a cluster of associated application servers that all host the same applications. All members of the cluster appear as one logical unit that serves the applications that are deployed to the cluster.

Keep in mind that a WebSphere Application Server cluster can only be implemented with the Network Deployment or the z/OS packages.
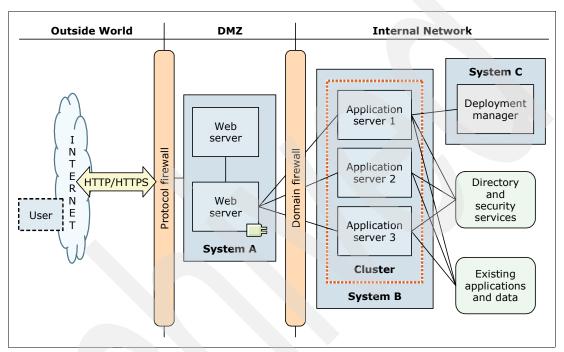
Figure 5-5   Vertical scaling topology with WebSphere Application Server

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 performed at the web server plug-in level based, by default, on a weighted round-robin algorithm.

> **SPOF:** The illustration in Figure 5-5 is intended to show a vertical scaling topology of application servers but still contains several SPOF.

Vertical scaling can be combined 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 running its own JVM uses a portion of the processor and memory of the machine. The number of application servers on a system can be increased or decreased to optimize the resource use of the machine.

- ► 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 because 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 will continue to process and realign the workload.

Disadvantages

Vertical scaling has the following disadvantages and possible drawbacks to consider:

- ► SPOF

  Unless you combine the vertical scaling architecture with horizontal scaling, you still have SPOF (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. For example, to benefit from a failover mechanism, you need to consider what is required for a successful failover (such as session data) and size carefully for all possible situations.

Setting up the topology

To set up an environment similar to the one illustrated in Figure 5-5 on page 125, complete the steps as explained 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:

   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 a supported web server.

3. Open the WebSphere Customization Toolbox, and launch 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 Version 8 Information Center at:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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 Network Deployment V8.

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 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.

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 Integrated Solutions Console or the **wsadmin** scripting interface.

5. Create a WebSphere Application Server cluster with three cluster members on System B.

5.3.4 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, first one without an IP sprayer and then one with the IP sprayer component (see "Horizontal scaling topology with an IP sprayer" on page 130).

Horizontal scaling topology without an IP sprayer

In the topology illustrated in Figure 5-6 on page 128, a single application spans multiple machines, while presenting 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.
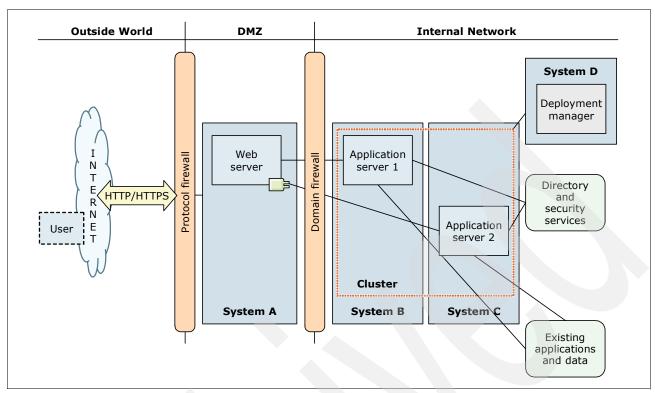
Figure 5-6   Horizontal scaling with cluster

The web server plug-in distributes requests to the cluster members on each server performing load balancing and 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.

> **SPOF:** The illustration in Figure 5-6 is intended to show a horizontal scaling topology of application servers but still contains SPOF (namely, the web server). To avoid these SPOF, you have to enhance the topology as explained and illustrated in "Horizontal scaling topology with an IP sprayer" on page 130.

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 machines, each cluster member can use the processing resources of the machine, avoiding bottlenecks and resource contention. Therefore, response times will 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 machines if a machine goes offline. The outage of a system or an operating system failure does not stop a service from working.

Disadvantages

Using horizontal scaling with clusters has the following disadvantages:

▶ 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 5-6 on page 128, complete the steps as explained 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:
   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 a supported web server.

3. Open the WebSphere Customization Toolbox, and launch 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 Version 8 Information Center at:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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.

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 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.

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 Integrated Solutions Console or through 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.

5.3.5 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 machines. The Load Balancer component of Network Dispatcher, for example, is an IP sprayer that performs intelligent load balancing among web servers based on server availability and workload.

Figure 5-7 on page 131 illustrates a horizontal scaling configuration that uses an IP sprayer to redistribute requests between web servers on multiple machines.

> **Edge Components:** To reduce the number of systems and to demonstrate the collocation capabilities of the Edge Component Load Balancer, in Figure 5-7 on page 131, Load Balancer and the web server are installed on the same system. The web servers can also be installed on separate systems in the DMZ. For more information about the supported network configuration for Edge Components, see the WebSphere Application Server Version 8.0 Information Center at the following address and search for *Load Balancer for IPv4 and IPv6*:
>
> http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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, Load Balancer keeps track of the web servers health and routes requests around web servers that are not available. To avoid having Load Balancer be a SPOF, Load Balancer is set up 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 and perform load balancing and failover between the application servers in the cluster through the web server plug-in. Any component on System C or System D that fails should be detected by the plug-in, and the other server can continue to receive requests.
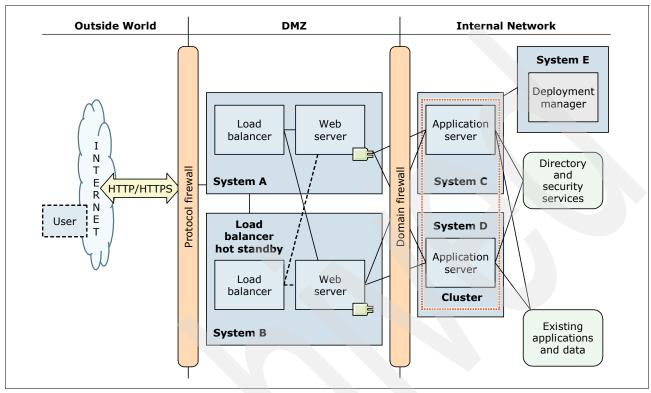
Figure 5-7   Simple horizontal scaling topology with an IP sprayer

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 (in this case, HTTP requests), among a group of web servers, spreads the workload among the available application servers, which also 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.

Disadvantages

Using Load Balancer has some disadvantages, most of which are cost related. It also comes with 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 Load Balancer runs on a separate system, there are more systems to manage and to operate, which in turn, increases the cost for the operation of the environment.

Setting up the topology

To set up a topology environment similar to the one illustrated in Figure 5-7 on page 131, complete the setup steps as explained 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: one for you web servers cluster and the other for the IP address of the machine.

   For more information about this task, see the WebSphere Application Server Version 8 Information at the following address, and search for *migrating, coexisting, and interoperating*:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

4. 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 a supported web server.

5. Open the WebSphere Customization Toolbox, and launch the Web Server Plug-ins Configuration Tool.

6. Configure the web server plug-in, and create the web server definition. For details about this task, see the WebSphere Application Server Version 8 Information Center (as referenced in step 3).

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.
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 running 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.

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 Integrated Solutions Console or the `wsadmin` scripting interface.

5. Create a WebSphere Application Server cluster by using either the Integrated Solutions Console or `wsadmin` with one cluster member on System C and one cluster member on System D.

5.3.6 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 illustrated in Figure 5-8 shows the use of DMZ Secure Proxy Server as the reverse proxy server for this example, because it offers a more secure option than the Proxy Server profile.
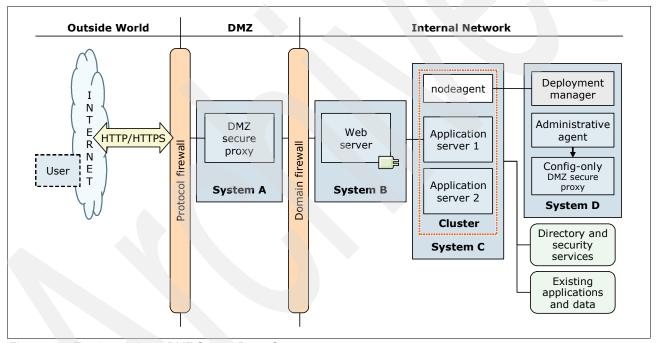
Figure 5-8   Topology using a DMZ Secure Proxy Server

> **High availability:** The illustration in Figure 5-8 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 and verifies that a valid copy of the requested object is 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 in the local cache, 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, hiding the web servers from the clients and allowing 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.

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 provide caching capabilities, offloading the web servers and the application servers if dynamic caching is also supported.

Disadvantages

Using a reverse proxy server in a DMZ configuration has the following disadvantages:

► 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 5-8 on page 133, complete the steps as explained 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:

   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 a supported web server.

3. Open the WebSphere Customization Toolbox, and launch 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 Version 8 Information Center at:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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.
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.

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 5-1 to export the static routing information.

   Example 5-1   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.

> **Profiles:** 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:** Name the proxy server and the node name in the configuration-only profile on System D the same as you did when creating 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 Integrated Solutions 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.

> **Updating the routing information:** 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 this 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 5-2.

   Example 5-2   Importing the proxy profile

   ```
   AdminTask.importProxyProfile('-archive directory/DMZProxy.car
   -deleteExistingServers true')
   AdminConfig.save()
   ```

4. Start the DMZ Secure Proxy Server.

5.3.7 Topology with redundancy of multiple components

To remove SPOF in a topology, redundant components are added. 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 redundancy of multiple components, the redundant 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 while being a failover component one for another. 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 illustrated in Figure 5-9 on page 138 shows the minimum WebSphere components that are used in an installation with the usual high availability requirements (where 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.
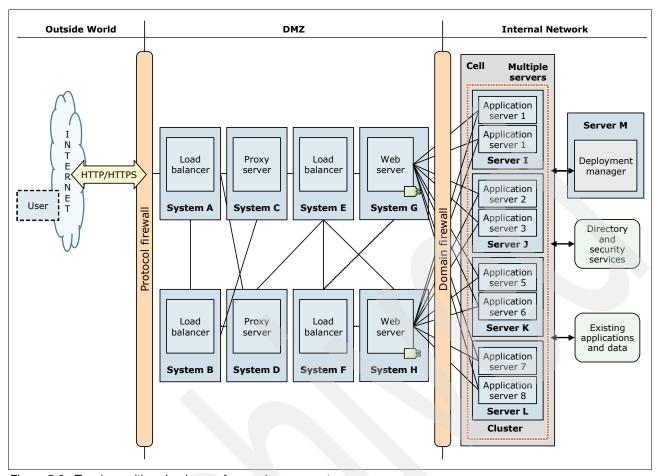
Figure 5-9   Topology with redundancy of several components

The following components are redundant in this example:

► Two clusters of load balancers

   The topology in illustrated Figure 5-9 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, provides the cluster address for the web servers. This cluster 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 "Horizontal scaling topology with an IP sprayer" on page 130.

   Cluster addresses: 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) are receiving requests from Load Balancer and 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 and shares the requests that come from the reverse proxies. Each web server is installed on a different machine 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 (as 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, which implies that cell-wide services, such as 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 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. To learn about the benefits of horizontal scaling, see 5.3.4, "Horizontal scaling topology" on page 127.

► Improved application performance

In most cases, application performance is improved by using the following techniques:

– Hosting application servers on multiple physical machines, 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 machine

► Usage of workload management technologies

Applications in this topology can benefit from workload management techniques. In this example, workload management is performed as follows:

– Load Balancer Network Dispatcher o 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.

Disadvantages

The major disadvantages of this topology with redundancy of multiple components is increased cost. This combined topology has the disadvantages of costs in hardware, complexity, configuration, and administration. Consider these costs in relation to advantages in performance, throughput, and reliability.

> **Additional disadvantages:** Because this topology is a combination of the topologies described earlier in this chapter, the disadvantages of the other base topologies also apply here.

Setting up the topology

To set up a topology environment similar to the one illustrated in Figure 5-9 on page 138, complete the steps as explained 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 IBM WebSphere Edge Components: Load Balancer for IPv6.

3. Configure the Load Balancer component according to your network topology.

> **Clusters:** Keep in mind that Systems A and B form one cluster and Systems E and F form another, different cluster.

For more information about this task, see the WebSphere Application Server Version 8 Information at the following address, and search for *migrating, coexisting, and interoperating*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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:

   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 a supported web server.

3. Open the WebSphere Customization Toolbox, and launch 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 Version 8 Information Center at:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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.

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 running 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.

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 definitions through the Integrated Solutions 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.

5.3.8 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 referred to as a *heterogeneous cell*. With WebSphere Application Server
V8, many different topologies are possible to compose a heterogeneous cell, as illustrated in
Figure 5-10.
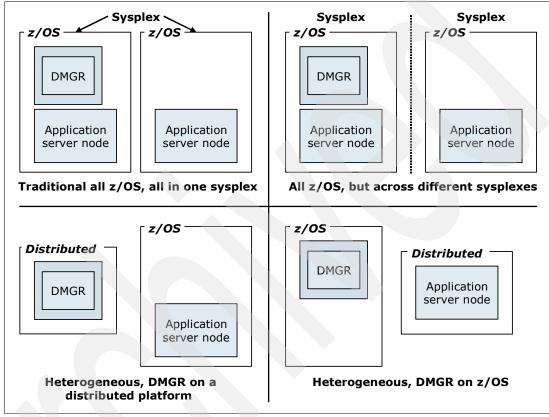
Figure 5-10   Different possibilities with a heterogeneous cell

WebSphere Application Server Version 8 products can coexist with the following supported
versions:

► IBM WebSphere Application Server Version 6.x
► IBM WebSphere Application Server Network Deployment Version 6.x
► IBM WebSphere Application Server Version 6.x for z/OS
► IBM WebSphere Application Server Version 7.0
► IBM WebSphere Application Server Network Deployment Version 7.0
► IBM WebSphere Application Server Version 7.0 for z/OS

WebSphere Application Server Version 6.x and Version 7 clients can coexist with Version 8
clients.

A WebSphere Application Server V8 Network Deployment cell can contain mixed releases of
Version 6.x or 7.x nodes, but no mixed-node management support is available for Version
6.0.0.x and Version 6.0.1.x. Upgrade all Version 6.0.0.x and Version 6.0.1.x nodes to at least
Version 6.0.2. By making this upgrade, the nodes can be administered by a Version 8
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 migration approach for migrating WebSphere Application Server versions. Although this environment is supported, mixed version cells are not a permanent solution.

Disadvantages

This topology has the following disadvantages:

► Complex administration

  Because of the heterogeneous environment, the administration is complex and requires administrator knowledge for all platforms.

► Increased administration and operational costs

  This combined topology has the disadvantages of hardware, configuration, and administration costs. Consider these costs in relation to gains in performance, throughput, and reliability.

For information about planning and system considerations 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

5.3.9 Multicell topology

The topologies introduced in 5.3.7, "Topology with redundancy of multiple components" on page 137, and in 5.3.8, "Heterogeneous cell topology" on page 142, provide a high level of availability and redundancy for all types of WebSphere components. Nevertheless, application software problems or malfunctioning of cell-level components such as high availability manager, although rare, are potential threats to the availability of your service.

A possible approach is a two cell architecture as outlined in Figure 5-11 on page 144. This topology is a duplication of the topology introduced in 5.3.7, "Topology with redundancy of multiple components" on page 137. However, here independent cells are implemented, both cells provide service. Even cell-level problems can be handled quickly in this topology, because full cells can be activated and deactivated as needed. Both cells can have installed the same applications, or 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. High availability and disaster recovery are two terms that you need to address differently.

If you are looking for disaster recovery, keep in mind that the topology illustrated in Figure 5-11 is not a complete solution for it. You must consider several factors when doing planning for disaster recovering. For more information, see 5.2.2, "Disaster recovery" on page 111.
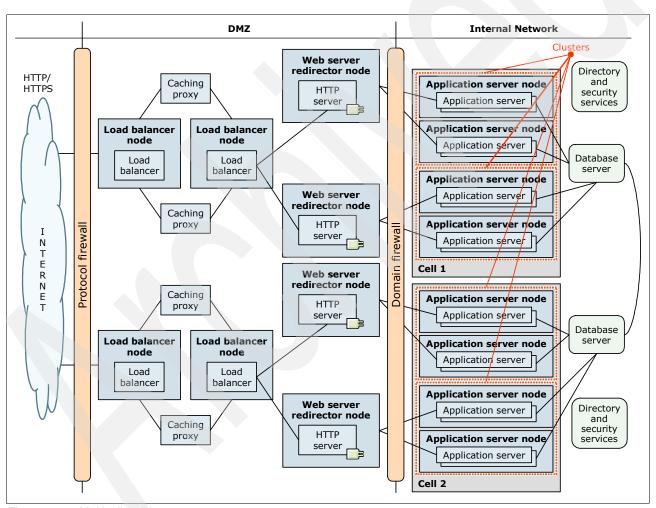
Figure 5-11   Multicell architecture

Advantages

The multicell topology has the following advantages:

► Provides all the advantages defined in 5.3.7, "Topology with redundancy of multiple components" on page 137.

► 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 the other 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 determine application problems in your production environment.

> **Compatibility:** Keep in mind that compatibility between application software releases is required.

► Possible approach for disaster recovery

Having the cells in different data centers is a possible approach for a disaster recovery solution from a WebSphere perspective.

> **Considerations for disaster recovery:** 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, including web requests, SIP requests, EJB requests, web services, and so on?

► 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.

Disadvantages

The multicell topology has a disadvantage 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 will likely have specific requirements to consider such an architecture.

> **Additional disadvantages:** Because this topology is a combination of topologies described previously, the multicell topology also has the disadvantages of those base topologies.

Setting up this topology

The steps for this topology are the same as in "Setting up the topology" on page 140, for a topology with redundancy of multiple components.

5.3.10 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 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 machine, but you can connect to the machine from a browser or the `wsadmin` tool on another machine.

> **Registering an application server node:** You can register an application server node with the administrative agent or federate the node with a deployment manager, but you cannot do both.
>
> **Security:** 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.

The topology in Figure 5-12 shows a possible topology using an administrative agent to manage all of the single server installations on System B. Instead of running the Configuration service, Integrated Solutions 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 the administrative access uses one central point. Therefore, you have one URL for the administration instead of multiple URLs.
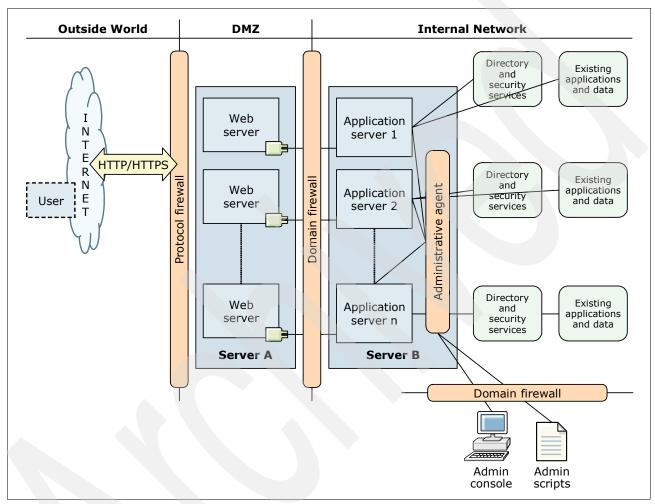
Figure 5-12   Topology containing administrative agent

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 (Integrated Solutions Console, `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 only need accessibility to the administrative agent to each application server.

Disadvantages

The implementation of an administrative agent profile has the disadvantage of requiring 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 5-12 on page 147, 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:

   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 a supported web server.

3. Open the WebSphere Customization Toolbox, and launch 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 Version 8 Information Center at:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

   > **Web server setup:** 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.

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 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 Integrated Solutions 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.

5.3.11 Advanced topology using a job manager

The job manager is an administration feature that was previously in WebSphere Application Server V7. The job manager addresses scalability issues of the administrative run time if the components are spread over multiple remote locations. An example of such a deployment is a typical branch deployment where central management is desired 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, you can know 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 5-13 shows the possible usage of a job manager for the central administration of multiple heterogeneous environments.
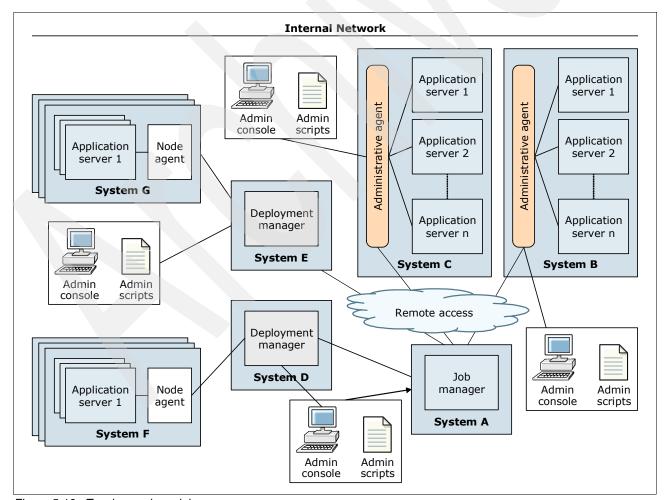
Figure 5-13   Topology using a job manager

The topology shown in Figure 5-13 on page 149 is a possible usage scenario for the 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) 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 execute 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

Disadvantages

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 5-13 on page 149, 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.

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 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.

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 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 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.

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.

3. Create an application server profile:

    – Use the *app_server_root*/profileTemplates/managed profile template. Federate this profile to the proper 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 proper deployment manager by using the **addNode** command.

6

Installation planning

This chapter provides general guidance for planning the installation of WebSphere Application Server V8 and many of its components. To effectively plan an installation, you need to select a topology, hardware, and operating system, and to prepare your environment for WebSphere Application Server installation.

> **Prerequisites:** For detailed information about prerequisite for WebSphere Application Server V8, see System Requirements for WebSphere Application Server Base and Network Deployment V8.0 at:
>
> http://www.ibm.com/support/docview.wss?uid=swg27021246

This chapter includes the following sections:

- ► Installation features in WebSphere Application Server V8
- ► 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 Load Balancer
- ► Planning for the DMZ secure proxy
- ► Planning for the HTTP server and plug-in
- ► WebSphere Customization Toolbox
- ► IBM Support Assistant
- ► Installation checklist
- ► Resources

For information about how to install WebSphere Application Server V8, which is not explained in the chapter, go to the WebSphere Application Server Version 8 Information Center, and search for *installing and configuring your application serving environment*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

6.1 Installation features in WebSphere Application Server V8

The following installation-related items are new or changed significantly for the WebSphere Application Server V8 editions for *distributed systems:*

► IBM Installation Manager

 – IBM Installation Manager replaces InstallShield MultiPlatform (ISMP) and the Update Installer for WebSphere Application Server V8.

 – The product installations, updates, and uninstallations with integrated prerequisite and interdependency checking are now done by Installation Manager. *Installation Manager* is a single installation program that loads and installs product components from a structured collection of files known as a *repository.*

 – With Installation Manager, you can easily identify product and maintenance packages, install packages after prerequisite and interdependency checking, and uninstall and roll back previously installed packages.

 – WebSphere Application Server V8 products and related components are installed, modified, or updated by using remote or local repositories:

 • Application Client for IBM WebSphere Application Server
 • Edge Components
 • IBM HTTP Server V8
 • Pluggable Application Client for IBM WebSphere Application Server
 • WebSphere Customization Toolbox
 • WebSphere DMZ Secure Proxy Server
 • Web server plug-ins

 – Installation Manager replaces the functionality previously provided by the Installation Factory.

► WebSphere Customization Toolbox V8

The WebSphere Customization Toolbox for WebSphere Application Server V8 includes tools for customizing various parts of the WebSphere Application Server environment. From the WebSphere Customization Toolbox, you can launch the following tools:

 – The Web Server Plug-ins Configuration Tool

 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 to communicate with the application server. If possible, this tool also creates a web server configuration definition in the application server.

 – The Profile Management Tool (z/OS only)

 You can use the Profile Management Tool (z/OS only) on an Intel technology-based Windows or Linux operating system to generate jobs and instructions for creating profiles for WebSphere Application Server on z/OS systems.

 – The z/OS Migration Management Tool

 You can use the z/OS Migration Management Tool on an Intel technology-based Windows or Linux operating system to generate definitions for migrating WebSphere Application Server for z/OS profiles.

► Centralized installation manager

The centralized installation manager (CIM), which is used to install and apply maintenance on remote computers, is now integrated with Installation Manager. You can use CIM to install Installation Manager instances, update Installation Manager with a repository, and

manage Installation Manager offerings by using either the administrative console or the `wsadmin` tool.

► Feature and setting changes

The following feature and setting changes apply to WebSphere Application Server V8:

– The EJBDeploy tool was installed automatically with the product in WebSphere Application Server V7 and earlier. It is now an optional feature, but is selected for installation by default.

– In WebSphere Application Server V8 and later, the default file-permission setting is 775.

6.2 Selecting a topology

Chapter 5, "Topologies" on page 103, provides information about common configurations. Each topology description contains information about the software products that are 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, and plan for the installation of the required products.

6.3 Selecting hardware and operating systems

After you select a topology, you must next 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, you can determine the configuration of each server by selecting the processor features, the amount of memory, the number of direct access storage device (DASD) arms, and 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 proper maintenance level installed for WebSphere Application Server to work properly and to get support. Keep in mind that not every product you receive with WebSphere Application Server V8 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, see System Requirements for WebSphere Application Server Base and Network Deployment V8.0 at:

http://www.ibm.com/support/docview.wss?uid=swg27021246

6.4 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, for example.

Although WebSphere Application Server products provide a default directory structure for their components, it might not be the best choice because 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, using different mount points. Generally, disk space management is more flexible and efficient if you split the installation into different file systems. By planning your directory structure and file systems, you can consider other criteria such as performance, backup requirements and capabilities, availability, and security.

Different file systems for the following purposes are useful:

► 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 have to change the server configuration for each server. You can change the server configuration by using scripting or by using 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, as a good practice, redirect the system 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) 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 number of JVMs dumping to this directory, their individual sizes, the number of dump files that you want to retain, and so on.

► Maintenance packages or Installation Manager repositories

   Installation Manager uses repositories for easier management of your installation packages and your fix packs. If you use the CIM, it uses the Installation Manager repositories to install and maintain WebSphere Application Server V8.

CIM: CIM also keeps a directory of Installation Manager installation kits on the job manager. To install Installation Manager, CIM 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.

6.5 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, and 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 proper 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. Experience has shown that it is best to avoid any special or national language-specific characters in the names.

6.6 IBM Installation Manager

WebSphere Application Server V8 is the first full version to be installed by IBM Installation Manager rather than by programs based on InstallShield MultiPlatform that are used to install previous versions.

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 reside in repositories to which Installation Manager can connect and acquire all required binary files that are relevant to user selections and the system environment.

6.6.1 Benefits of Installation Manager

Using Installation Manager provides the following benefits:

► Supports full product lifecycle management operations including the following operations:

  – Installation
  – Update
  – Modify
  – Rollback
  – Uninstallation

► A single tool for all WebSphere Application Server platforms

► A consistent user experience across multiple IBM products

► Several methods for performing lifecycle management activities:

  – Using a graphical user interface (GUI)
  – Using a command-line interface (CLI)
  – Silently

► One-pass installation

  – Installs the level of service desired without multiple steps to install GA level and updates
  – Can install multiple products

► Validation and system checking performed 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 GA version of the product to enable an optional feature is no longer necessary.

  – Optional installable features can be selected during the fixpack installation or from the modify option.

► Better management of files for rollback

6.6.2 Infrastructure topologies

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. These packages and package groups 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
► Local Installation Manager enterprise repository

With IBM Packaging Utility, you can manage repositories and packages. Often the IBM Packaging Utility is used to copy software packages from CD installation images or update packages from IBM repositories into local repositories, from which installations can be performed.

With 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.
► Copy packages from CD installation images or IBM hosted repositories.
► Delete packages.

> **Installation objects:** Installation objects, including product installations, updates, and fixes that can be manipulated or used by Installation Manager are known as *software packages*.

Local repository

As illustrated in Figure 6-1, a local Installation Manager repository resides on the same server where Installation Manager is installed and from which installations can be performed.
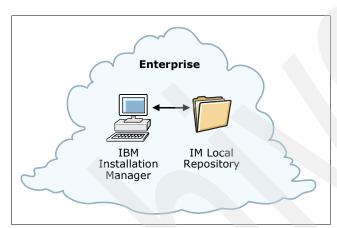
Figure 6-1   Local Installation Manager repository

IBM hosted repository

The Installation Manager can also download packages from IBM hosted repositories, such as IBM Passport Advantage®, as shown in Figure 6-2.
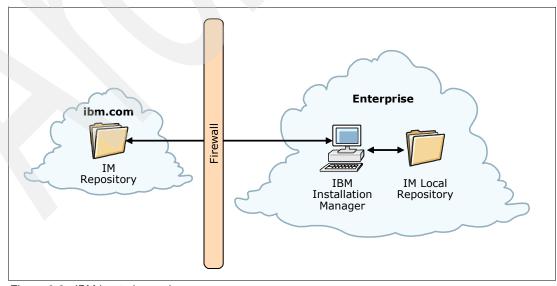
Figure 6-2   IBM hosted repository

Local enterprise repository

A local enterprise repository contains software packages and package groups that are shared with one or more computers within the enterprise as shown in Figure 6-3. A typical workflow, as shown in Figure 6-3, might involve the following tasks:

► Download images from an IBM hosted repository or other repositories to a local computer with the Packaging Utility.

Images might include compressed CD installation images that contain software package repositories, including software packages such as WebSphere Application Server.

► 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.

► Use Installation Manager to install the software package from the local or enterprise repository.
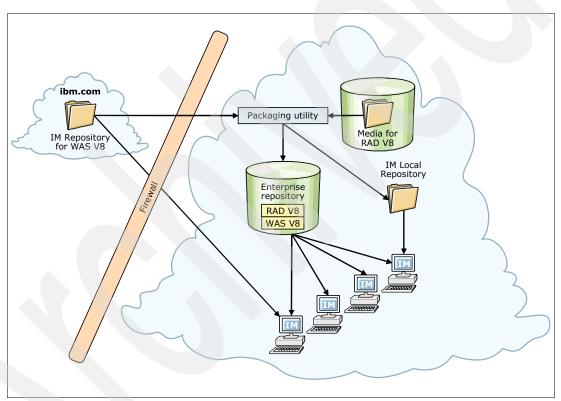
Figure 6-3   Enterprise repository

More information: For installation instructions and more details about IBM Installation Manager, see the IBM Installation Manager Information Center at:

http://publib.boulder.ibm.com/infocenter/install/v1r4/index.jsp?top

6.7 Planning for WebSphere Application Server

WebSphere Application Server V8 entails a full product installation, not an upgrade installation. Consider the best installation method to use based on the number of systems and the complexity of the installations.

Review the documentation. The WebSphere Application Server Information Center contains planning topics for all WebSphere Application Server packages that are tailored to each platform. This section provides a high-level view of the planning tasks that you need to perform.

Installation considerations z/OS: For information about installation considerations for WebSphere Application Server V8 for z/OS, see 14.3, "Installing WebSphere Application Server for z/OS" on page 417.

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, you can choose on which file systems you want to install the product and 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 one time on a system and to create multiple runtime environments 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. They include the size of the installation (the number of systems), the operating systems involved, the number of times you anticipate performing the same installation (using a GUI or performing a silent installation), and if you are performing remote installations with unskilled personnel. For some installations, 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 will 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 if writing scripts.

► TCP/IP port assignments

Each type of server (deployment manager, node agent, application server, and so on) 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 will 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).

6.7.1 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 if this satisfies your needs. For more information, see 6.4, "Planning for disk space and directories" on page 156.

6.7.2 Single installation or multiple installations

You can install WebSphere Application Server V8 multiple times on the same system in different directories. Alternatively, you can install WebSphere Application Server V8 in parallel to older versions of WebSphere Application Server on the same system. These installations are independent of each other.

This configuration facilitates fix management. If a fix is applied on a particular installation, it only affects that specific WebSphere Application Server installation, leaving the remaining installations on that machine 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, you can create multiple application server profiles. In this case, all profiles share the product binary files. Therefore, you have to 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 6-4 shows the difference between multiple installations and multiple WebSphere profiles in a stand-alone server environment (Base and Express).
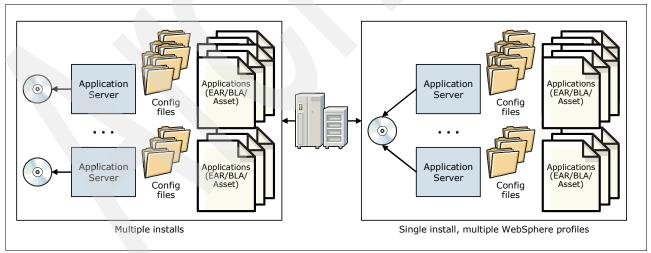
Figure 6-4   Stand-alone server installation options

Limits: There is no architectural limit for multiple installations or multiple profiles. The real limitation is ruled by 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, you can install Network Deployment one time and then create multiple profiles so that each profile is used to administer a different cell, as shown in Figure 6-5.
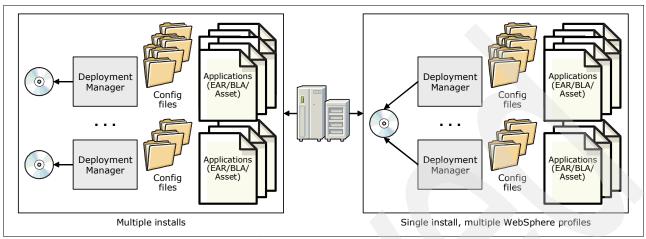
Figure 6-5   Deployment manager installation options

Another possibility is the combination of multiple installation instances and multiple profiles. Figure 6-6 illustrates a Network Deployment environment where multiple runtime environments have been created by using profiles. In Figure 6-6, cell 1 contains a deployment manager and application server on separate machines, using separate installation instances. Cell 2 contains a deployment manager and two application servers that span three installation instances.
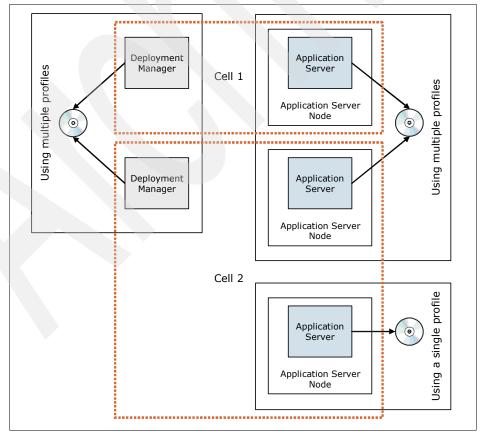
Figure 6-6   Cell configuration flexibility

6.7.3 Installation method

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. Wizard mode is suitable for installing WebSphere Application Server on a few systems. Executing wizard mode installs one system. You can start with the Launchpad, which contains a list of activities to select, or you can start the Installation Manager directly.

Installation Manager verifies the required operating system level, sufficient disk space, and user permissions before downloading any required packages.

Silent installation

To install WebSphere Application Server V8 on multiple systems or remote systems, use the silent installation. With this option, you can store installation and profile creation options in a single response file and then issue a command to perform the installation and (optionally) profile creation. The silent installation approach offers the same options as the graphical installer. Providing the options in a response file offers various 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:** Do not use the same response files that are used with WebSphere Application Server V7 or earlier versions to install or uninstall Version 8 silently. Use response files that are based on Installation Manager to install, update, or uninstall Version 8 or later.

Centralized installation manager

Another product feature that you can use to install and update WebSphere Application Server Network Deployment installations is the CIM. For more information about CIM, see 9.6.3, "Centralized installation manager" on page 288.

Verifying the installation

You can verify the installation of WebSphere Application Server V8 by using the installation verification features provided by the Installation Manager.

The installation can be verified by using one of the following methods:

► Use the `listInstallPackages` command to display a list of the packages that are installed by Installation Manager.

► Launch 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 `installed.xml` file and looking for the location element as shown in Example 6-1.

Example 6-1   The installed.xml file

```
<location id="IBM WebSphere Application Server V8.0" 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.

6.7.4 Installing updates

For WebSphere Application Server V8 maintenance tasks, use the Installation Manager or CIM. The Installation Manager is used to apply maintenance for WebSphere Application Server V8. Installation Manager uses repositories for easier management of your installation packages and your fix packs.

If you use the CIM, it uses the Installation Manager repositories to install and maintain WebSphere Application Server V8. CIM also keeps a directory of Installation Manager installation kits on the job manager. To install Installation Manager, CIM pushes the Installation Manager installation kit to a target system and installs it.

When dealing with fixes and fix packs, you must plan for and implement a proper policy on where to archive fixes and fix packs. This plan is best implemented by using the CIM. Using CIM allows quick configuration of identical systems and brings systems to the identical software level.

> **Feature packs:** Previous versions of WebSphere Application Server will continue to use the Update Installer. Feature packs for WebSphere Application Server V7, which are installed by Installation Manager, can also be serviced by Installation Manager.

6.7.5 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. Because all profiles of an installation share binary files, 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 166.

► How will you create the profiles?

   See "Creating profiles" on page 168.

► Where do you store the profile configuration files?

   See "Profile location" on page 176.

   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 as outlined in 6.4, "Planning for disk space and directories" on page 156.

Profile types

The types of profiles that are available to you depend on the WebSphere Application Server package that you have 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

The deployment manager profile creates an application server named `dmgr` and deploys the Integrated Solutions Console (recommended). 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 clustering, high availability, failover, and so on.

When using **manageprofiles**, you can create a deployment manager profile by using either of the following options:

– 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

The administrative agent profile creates the application server named `adminagent` and deploys the Integrated Solutions Console (recommended).

The administrative agent provides a centralized administration interface for multiple unfederated application server profiles on the same system without providing any support for clustering, high availability, failover, and so on.

When using **manageprofiles**, specify `-profileTemplate` *app_server_root*`/profileTemplates/management,` and then specify `-serverType ADMIN_AGENT`.

► Management profile with a job manager server

The job manager profile creates the application server named `jobmgr` and deploys the Integrated Solutions Console (recommended).

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

No additional capabilities for clustering, high availability, failover, and so on are provided.

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)
– Integrated Solutions Console (recommended)

The default name of the application server is server1, but 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 a deployment manager.

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 deployment manager either during profile creation or at a later time.

    When using **manageprofiles** specify -profileTemplate *app_server_root*/profileTemplates/managed.

- ► Cell profiles

    A cell profile creates a deployment manager profile and a federated application server profile at the same time on the system by using default naming conventions in certain areas. The result of this profile creation is a fully functional cell. The following applications can be deployed to the federated application server:

    - – Default applications (optional)
    - – Sample applications (optional)

    The deployment manager profile deploys the Integrated Solutions Console (recommended). From the functional perspective, approach is the same as creating a management profile with a deployment manager server and an application server profile, and then federating the application server profile to the deployment manager.

    When using **manageprofiles**, two portions of the cell 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 demilitarized zone (DMZ) and supports HTTP, Session Initiation Protocol (SIP), and the corresponding secure version of the protocols.

    The default name of the application server is proxy1, but 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 6-1 shows a list of the available profile types per WebSphere Application Server edition.

Table 6-1   Available profile types for the editions of WebSphere Application Server

Product	WebSphere profiles available
WebSphere Application Server Express V8	► Management profile with an administrative agent server ► Application server profile
WebSphere Application Server V8 (single server)	► Management profile with an administrative agent server ► Application server profile.
WebSphere Application Server Network Deployment V8	► 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 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 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

The installation on WebSphere Application Server for z/OS uses System Modification Program/Extended (SMP/E) and only installs the product binary files. After the installation, you create profiles by using the z/OS Profile Management Tool, which is available in the WebSphere Application Server Toolkit. For more information about z/OS installation and the configuration steps, see 14.3, "Installing WebSphere Application Server for z/OS" on page 417.

> **Profile Management Tool:** To have an operational product, you must create a profile by using the Profile Management Tool after installation.

After you create your first profile, you can create additional profiles to create additional runtime environments and to expand distributed server environments. When you use the Profile Management Tool and choose the advanced path or the `manageprofiles` command to create the profiles, you have more flexibility in the options that you can select.

> **Tips:**
>
> ► 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 6-2 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 6-2   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 `Dmgrxx` 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 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 have to specify a user name and password that are given 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 an existing 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 an existing keystore.
The default expiration date for the personal certificate is 1 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 are 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, you can choose whether the deployment manager will run 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 6-3 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 6-3   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 run without specifying a profile are run against the default profile.)
The cell name is *host*AACell*xx*. The node name is *host*AANode*xx*. 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 have to 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 an existing keystore.
Creates a new 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 an existing keystore.
Default expiration date for the personal certificate is 1 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 are 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, you can choose whether the deployment manager will run 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 6-4 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 6-4   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 `JobMgrxx` 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 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 have to 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 an existing 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 an existing keystore.
The default expiration date for the personal certificate is 1 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 are 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, you can choose whether the deployment manager will run 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)

Table 6-5 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 6-5   Options for the application server profile (non-Express V8)

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 `AppSrvxx` 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 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 have to specify a user name and password that are given administrative authority.	
Creates a default personal certificate for this profile 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 to import an existing default personal certificate from an existing 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 an existing keystore.
The default expiration date for the personal certificate is 1 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 are 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, you can choose whether the application server will run 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)

Table 6-6 summarizes the options that are available when creating a profile for an application server in WebSphere Application Server Express V8. The options depend on whether you choose the typical path or advanced path through the Profile Management Tool.

Table 6-6   Options for the application server profile (Express V8)

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 `AppSrvxx` 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 have to specify a user name and password that are given administrative authority.	
Creates a default personal certificate for this profile 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 an existing 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 an existing keystore.

The default expiration date for the personal certificate is 1 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 are 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, you can choose whether the application server will run 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 6-7 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 6-7   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 `Dmgrxx` by default, where *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 `AppSrvxx` by default, where *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 `hostCellxx`. The node name for the deployment manager is `hostCellManagerxx`. The node name for the application server is `hostNodexx`. 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 have to specify a user name and password that are given administrative authority.	
Creates a default personal certificate for this profile 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 an existing 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 being created or import an existing root signing certificate from an existing keystore.
The default expiration date for the personal certificate is 1 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 are 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, you can choose whether the application server will run 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 6-8 summarizes the options that are available when creating a custom profile.

Table 6-8   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 if 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 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 an existing 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 being created or to import an existing root signing certificate from an existing keystore.
The default expiration date for the personal certificate is 1 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 are 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 Base, Express, or Network Deployment V8 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 V8.0] Tools** → **Profile Management Tool**).

► By running the **pmt.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 to the *app_server_root*/**manageprofiles.bat(sh)** command. You can use this command directly to manage (create, delete, and so on) profiles without a 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 considerations about disk space and directory planning, see 6.4, "Planning for disk space and directories" on page 156.

6.7.6 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 close 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 and helps them to find the proper 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

  – Illegal 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 situation:

– 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) and, if appropriate, 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`, and a running number to enable growth if additional nodes need to be created.

► Naming application servers

In stand-alone environments, the default server name is `server1`, but can be overridden through the `manageprofiles.bat(sh)` command or by using the advanced profile creation options in the Profile Management Tool.

Cell naming scheme when federating multiple stand-alone application servers: When you federate multiple stand-alone application servers that were 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, it is more likely that new application servers are created on a federated node by using the Integrated Solutions Console or another administrative tool. In this case, you can name the server, assigning a meaningful name. Whether you choose to name servers based on their location, function, membership in a cluster, or some other scheme largely depends on how you anticipate your servers being used and administered.

Unique server name: The server name must be unique within the node.

If each application server will host only a single application, the application server name can include the name of the application. If several applications (each deployed on their own application server) make up a total system or project, that name can be used as a prefix to group the application servers, making it easier to find them in the Integrated Solutions Console.

If an application server hosts multiple applications, develop another suitable naming convention, such as 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.

6.7.7 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, it is highly desirable to 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 machine 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, discuss and 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 using the advanced options path, you can set the ports for your profile as needed. The Profile Management Tool identifies the ports that are used in the same installation on that system and the ports that are 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 Version 8 Information Center at the following address, and search for *port number settings in WebSphere Application Server versions*:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp`

6.7.8 Security considerations

To plan a secure WebSphere Application Server environment, you must have highly skilled security specialists that 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 will 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.

> **Signer and personal certificates:** In WebSphere Application Server V8, 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 to 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, 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 Version 8 Information Center at the following address, and search for *installing the product*:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp`

You can run Installation Manager in group mode. For more information about group mode, see "Installation modes" on page 180.

– Ensure that any policies on password expiration are well known to avoid disruption on the service (password expiration of root, administrator, or the password of the user to access a database).

Installation modes

The term *non-root* implies a Linux or UNIX installer, but also means a non-administrator group installer on a Windows system. You can install WebSphere Application Server V8 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 or IBM i platforms.

> **More information:** For detailed information about installing in group mode, see the WebSphere Application Server Version 8 Information at the following address, and search for *installing in group mode*:
>
> http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

Secure administration tasks

WebSphere Application Server provides a mechanism to secure the administrative interfaces. With WebSphere Application Server V8, you can enable security for administrative tasks during profile creation for an application server or deployment manager (including 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, and 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.

6.7.9 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/

6.8 Planning for Load Balancer

Before starting the installation of Load Balancer, you must complete several planning tasks. The first prerequisite to install Load Balancer is that you must finish the detailed network planning for your environment and have an exact understanding of the data flow in your environment. Edge Components V8 is shipped with two versions of Load Balancer:

► Load Balancer for IPv4
► Load Balancer for IPv4 and IPv6

> **Exception:** Unless you have a specific requirement to use Load Balancer for IPv4, use Load Balancer for IPv4 and IPv6.

6.8.1 Installation

Before you start installing Load Balancer, consult the WebSphere Application Server Version 8 Information Center (at the following web address) for these reasons:

► To ensure that all the required hardware and software prerequisites are met
► To understand the hardware and software requirements of Edge Components

Installation Manager is used to install Load Balancer. For detailed installation documentation about Load Balancer for IPv4 and IPv6, see the WebSphere Application Server Version 8 Information Center at the following address and search for *Load Balancer for IPv4 and IPv6*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

6.8.2 Configuration

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.

Load Balancer for IPv4 and IPv6 provides the following methods for configuration:

► Command line
► Configuration Wizard
► GUI
► Scripts

Forwarding method

Load Balancer provides different methods to forward packages to the servers to which they are dispatching:

► Media Access Control (MAC)-level routing
► Encapsulation forwarding

Advisors

Advisors are used to keep track of 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.

The Load Balancer product provides various built-in advisors that are ready to use, but that also allow the usage of custom advisors. You can write your own advisor and configure Load Balancer to react based of the response of your advisor.

6.9 Planning for the DMZ secure proxy

The DMZ secure proxy was a new feature of the WebSphere Application Server Network Deployment V7 product. In contrast to the WebSphere proxy that was introduced in WebSphere Application Server V6.0.2, this WebSphere software-based proxy solution is hardened and, therefore, suitable to run in a DMZ.

> **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.

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 running in the DMZ and that forwards requests to the content servers.

Because of the similarity of the installation of a DMZ Secure Proxy Server and WebSphere Application Server, this section only outlines the differences between the two products.

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 about these items, see 6.7, "Planning for WebSphere Application Server" on page 160.

6.10 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, and configure a supported web server to an installed web server plug-in.

> **Locating the applications:** WebSphere Customization Toolbox is in the supplements directory of WebSphere Application Server, along with IBM HTTP Server and the web server plug-in.

6.10.1 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 is unable to create the web server definition in the application server configuration directly, the tool creates a script that can be copied to the application server system. The script is run to create the web server configuration definition within the application server configuration.

The Web Server Plug-ins Configuration Tool is launched from the WebSphere Customization Toolbox. Figure 6-7 shows the main window of the Web Server Plug-ins Configuration Tool.
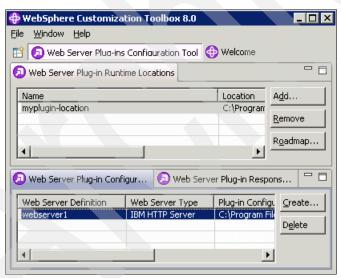
Figure 6-7   Main window of the Web Server Plug-ins Configuration Tool

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 6-8). As an alternative to using the Web Server Plug-ins Configuration Tool, you can use the command-line tool for Web Server Plug-ins Configuration Tool with a response file to configure a web server.
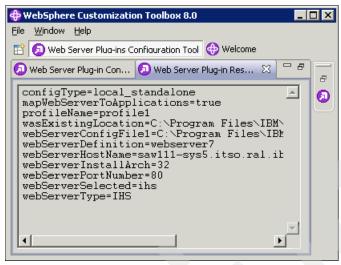
Figure 6-8   Response file for web server plug-in

6.10.2 Configuration process for WebSphere plug-in

The examples in this section outline the process that is required to create each sample topology.

> **Installation of WebSphere processes:** For each example, only the WebSphere processes shown in the diagrams are installed on each system and that the profile for the process is the default profile.

The examples in this section are provided to help you understand the process; they are not a substitute for using the product documentation. For detailed information about how the Web Server Plug-ins Configuration Tool works, see the *Plug-ins Installation Roadmap*, which is installed with the `roadmap/` subdirectory of the web server plug-in root directory. The installation road map provides information about the installation and configuration of the Web Server Plug-ins Configuration Tool.

During the plug-in configuration, you are prompted to respond whether the configuration is local or remote. Depending on your response, a path through the configuration occurs. Figure 6-9 on page 185 illustrates the plug-in configuration behavior in more detail.
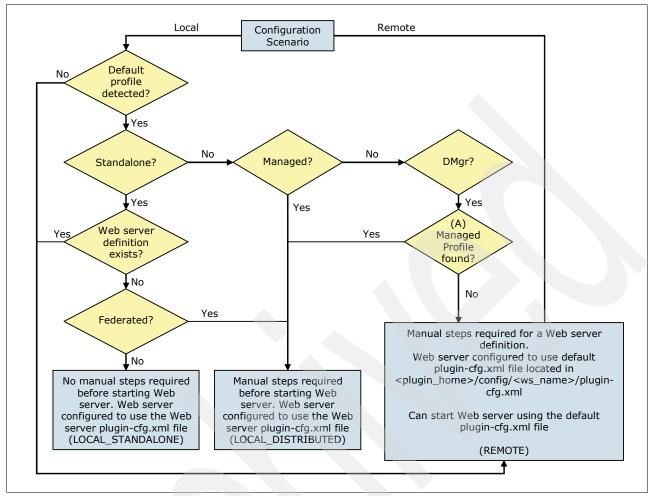
Figure 6-9   Configuration behavior of the web server plug-in

The plug-in configuration maps all possible configurations to three scenarios (LOCAL_STANDALONE, LOCAL_DISTRIBUTED, and REMOTE), as illustrated in Figure 6-9:

► LOCAL_STANDALONE

A LOCAL_STANDALONE plug-in configuration is a default unfederated stand-alone profile that has no existing web server definition. The Web Server Plug-ins Configuration Tool performs the following tasks in this case:

– Creates a web server definition for the default stand-alone profile
– Configures the web server to use the `plugin-cfg.xml` file

Web server definition: If the stand-alone profile is federated, you must recreate the web server definition.

What is next?

You can start the web server and WebSphere Application Server without any manual steps and access the snoop servlet through the web server to verify that everything is working.

► LOCAL_DISTRIBUTED

A LOCAL_DISTRIBUTED plug-in configuration has the following characteristics:

– A stand-alone profile federated into a deployment manager cell
– A managed node that is either federated or unfederated in a cell
– A managed profile found after a default deployment manager cell is detected (see (A) in Figure 6-9 on page 185)

In this case, the Web Server Plug-ins Configuration Tool Configures the web server to use the `plugin-cfg.xml` file in the web server definition directory that the user needs to create manually. You cannot start the web server until the manual steps are completed. The tool does not create a web server definition for the default distributed profile.

What is next?

– If the managed node is still not federated, federate the node first to prevent the web server definition from being lost after the federation has occurred.

– Run the manual web server definition creation script.

– Start the web server and WebSphere Application Server, and run the snoop servlet to verify that everything is working.

► REMOTE

A REMOTE plug-in configuration has the following characteristics:

– A default deployment manager profile

– No default profiles detected in the WebSphere Application Server directory given by the user

– A default, unfederated, stand-alone profile with an existing web server definition

In this case, the Web Server Plug-ins Configuration Tool configures the web server to use the `plugin-cfg.xml` file in *plugin_root*/config/*webserver_name*/plugin-cfg.xml. The tool does not create a web server definition for the default distributed profile.

What is next?

If the default `plugin-cfg.xml` file in the *plugin_root* directory is used, start the web server and WebSphere Application Server, and select the snoop servlet to verify that everything is working. This task requires that the `DefaultApplication.ear` application file is installed.

In this case, the snoop servlet is accessible at the following URL:

`http://`*hostname*`:`*web_server_port*`/snoop`

To benefit from the web server definition, perform the following steps:

a. Copy the configuration script to the WebSphere Application Server machine.

b. Run the manual web server definition creation script.

c. Copy the generated web server definition `plugin-cfg.xml` file back to the web server machine into the *plugin_root* directory tree. For IBM HTTP Server, you can use the propagation feature.

d. Start the web server and WebSphere Application Server, and select the snoop servlet.

6.10.3 Stand-alone server environment

In a stand-alone server environment, a web server can be either remote or local to the application server machine, but only one can be defined to WebSphere Application Server. The web server always resides on an unmanaged node.

Remote web server

In this scenario (Figure 6-10), the application server and the web server are on separate machines. The web server machine can reside in the internal network or more likely will reside in the DMZ.
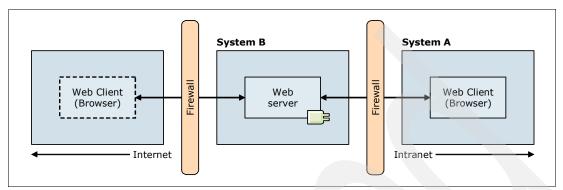
Figure 6-10   Remote web server in a stand-alone server environment

In this scenario, the application server is already installed and configured on system A. To create the environment shown in Figure 6-10, 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 on system B by performing the following steps:

   a. Select the type of supported web server.
   b. Identify the web server configuration file, and identify the web server port.
   c. Enter a name for the web server definition. The default is `webserver1`.
   d. Select **Remote** for the configuration scenario.

   > **Additional configuration tasks:** During configuration, the following tasks are performed 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/configure`web_server_name` directory.

e. After the configuration is complete, review the information in the Plug-in Configuration Result window (Figure 6-11). 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** option. Then click **Finish**.
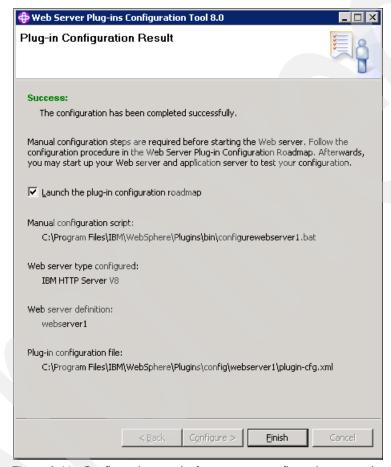
Figure 6-11   Configuration results for a remote configuration scenario

5. Follow the configuration procedure that is specified in the Web Server Plug-in Configuration road map.

Launch the plug-in configuration road map option: The **Launch the plug-in configuration road map** option is in the Plug-in Configuration Result window (Figure 6-11). If you do not select this option, click **Roadmap** in the main window (Figure 6-7 on page 183) of the Web Server Plug-ins Configuration Tool to access the road map.

a. Copy the script to the *app_server_root*/`bin` directory of the application server machine on System A.

b. Start the application server.

c. Execute 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.

Local web server

In this scenario (Figure 6-12), 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.
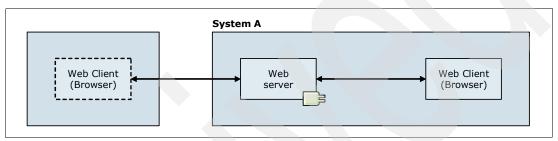
Figure 6-12   Local web server in a stand-alone server environment

In this scenario, the application server is already installed and configured. To create the environment shown in Figure 6-12, 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. By 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 identify 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.

> **Additional configuration tasks:** During configuration, the following tasks are performed 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 6-13). 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.
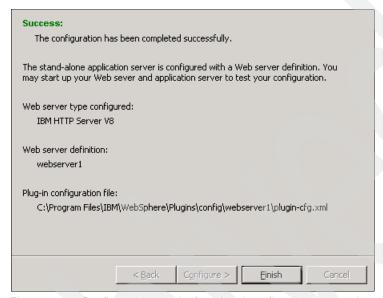
Figure 6-13   Configuration results for a local configuration scenario

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 6-7 on page 183).

6.10.4 Distributed server environment

Web servers in a distributed server environment can be local to the application server or remote. The web server can also reside on the deployment manager system. You can define multiple web servers. The web servers can reside 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 machines. 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, and you see an unmanaged node created for the web server node.

In Figure 6-14, the node is unmanaged because no node agent is on the web server system.
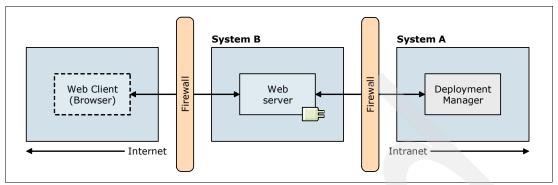
Figure 6-14   Remote web server in a stand-alone server environment

In this scenario, the deployment manager is already installed and configured on System A. To create the environment shown in Figure 6-14, 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 identify the web server port.
   c. Enter a name for the web server definition. The default is `webserver1`.
   d. Select **Remote** for the configuration scenario.

   > **Additional configuration tasks:** During configuration, the following tasks are performed 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 located in the `plugin_root`/bin/configure`web_server_name` directory.

   e. After the configuration is complete, review the information in the Plug-in Configuration Result window (Figure 6-11 on page 188). 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** option. Then click **Finish**.

5. Follow the configuration procedure specified in the Web Server Plug-ins Configuration road map.

> **Launch the plug-in configuration road map option:** The **Launch the plug-in configuration road map** option is in the Plug-in Configuration Result window (Figure 6-11 on page 188). If you do not select this option, click **Roadmap** in the main window (Figure 6-7 on page 183) of the Web Server Plug-ins Configuration Tool to access the road map.

  a. Copy the script to the *app_server_root*/bin directory of the application server machine, System A.

  b. Make sure that the deployment manager and the node agent are running.

  c. Execute 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.

> **Propagation for IBM HTTP Server:** 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 6-15), the web server is installed on a system that also has a managed node.
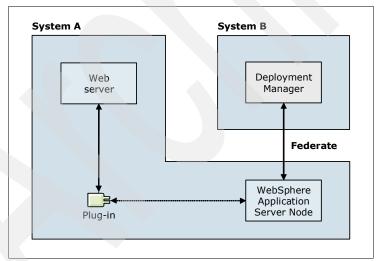
Figure 6-15   Web server installed locally on an application server system

In this scenario, the application server is already installed, configured, and federated to the deployment manager cell. To create the environment shown in Figure 6-15, 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. By 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 identify 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.

   > **Additional configuration tasks:** During configuration, the following tasks are performed 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/configure`web_server_name` directory.

   f. After the configuration is complete, review the information in the Plug-in Configuration Result window (Figure 6-13). 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. Then click **Finish**.

5. Follow the configuration procedure specified in the Web Server Plug-ins Configuration road map.

   > **Launch the plug-in configuration road map option:** The **Launch the plug-in configuration road map** option is in the Plug-in Configuration Result window (Figure 6-11 on page 188). If you do not select this option, click **Roadmap** in the main window (Figure 6-7 on page 183) of the Web Server Plug-ins Configuration Tool to access the road map.

   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. Execute 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.

   > **Managed web server:** For security reasons, avoid installing managed web servers in the DMZ.

6.11 WebSphere Customization Toolbox

The WebSphere Customization Toolbox for WebSphere Application Server V8 includes tools to manage, configure, and migrate various parts of your WebSphere Application Server environment. The WebSphere Customization Toolbox can be used to launch the following tools:

► Web Server Plug-ins Configuration Tool
► Profile Management Tool (z/OS only)
► z/OS Migration Management Tool

The WebSphere Customization Toolbox is installed by using the IBM Installation Manager.

6.12 IBM Support Assistant

IBM Support Assistant is available at no 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 your 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 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 such as Search, Product Information, Data Collection, Managing Service Requests, and Guided Troubleshooting. The Workbench can only perform these functions locally, for example, on the system where it is installed (except for the Portable Collector). For a list and description of 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 machine.

► 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 perform 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.

Installation instructions: 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/

6.13 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 installation or after installation. Decide on a location for the profile files (see 6.4, "Planning for disk space and directories" on page 156).

► 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 signer certificates.

► Develop a strategy for assigning TCP/IP ports to WebSphere processes.

► Select an installation method (wizard, silent, or CIM).

► Plan an administrative security strategy that includes a user repository and role assignment.

► Determine the user ID to use for installation and whether you will perform the installations by using administrator, non-administrator, or group mode.

► Plan for the web server and web server plug-in installation. Determine if 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.

6.14 Resources

WebSphere Application Server ships with an installation guide that you can access through the Launchpad. For additional information that guides you through the installation process, see the WebSphere Application Server Version 8 Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

7

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 environment. The nature of these three requirements make them related to each other. 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
► Scalability
► Performance
► WebSphere Application Server performance tools
► Workload management
► High availability
► Caching
► Session management
► Data replication service
► Checklist for performance, scalability, and high availability
► References

197

7.1 Performance, scalability, and high availability features in WebSphere Application Server V8

WebSphere Application Server V8 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
► High Performance Extensible Logging
► Disabling WebSphere MQ functionality
► JPA L2 cache provided by the dynamic cache provider
► Collecting Java dumps and core files with the administrative console
► Enabling request-level reliability, availability, and serviceability granularity
► Resource workload routing
► External high availability framework for service integration
► High availability for a WebSphere MQ link

7.1.1 Default garbage policy gencon

The default garbage policy of WebSphere Application Server V8 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

The objects are promoted from the nursery space to the tenured space, depending on their ages.

The gencon policy is the performing garbage collector policy for transactional applications, where the objects do not survive after the transaction ends, or for applications with many short-lived objects.

> **Tips for sizing the heap:** 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.

7.1.2 JVM garbage policy: Balanced

A new balanced Java virtual machine (JVM) garbage policy is also available with WebSphere Application Server V8. The strategy of this policy is to divide the heap between potentially thousands of regions, with each region being 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 and can be useful when you use the gencon policy with a heap size greater than 4 GB or if you use large arrays.

For additional information, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *Balanced Garbage Collection policy*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

7.1.3 High Performance Extensible Logging

WebSphere Application Server V8 provides a High Performance Extensible Logging (HPEL) logging and tracing feature. HPEL outperforms log writing when compared to the basic logging method. It writes logs and traces to a log data repository and a trace data repository in a binary format. (A text log file is also possible but impacts the performance.) A log viewer is provided to view, filter, and format the log data and trace data.

You can run performance tests with the logging enabled, which can improve application performance if your applications use log files intensely.

7.1.4 Disabling WebSphere MQ functionality

By default, WebSphere MQ functionality is enabled. To support this functionality, application server resources are continually used. If you do not want to take advantage of this functionality, you can disable it to improve the performance.

7.1.5 JPA L2 cache provided by the dynamic cache provider

The Java Persistence API (JPA) 2.0 has standardized the second level (L2) cache. WebSphere Application Server supports JPA 2.0. The dynamic cache service plugs in as an L2 cache provider to JPA. The L2 cache improves performance by avoiding direct requests to the database. The L2 cache also uses additional memory. Therefore, you must limit its size.

7.1.6 Collecting Java dumps and core files with the administrative console

You can now produce Java dump, Java core, and system dump files directly by using the administrative console. These files can be useful when analyzing performance issues, such as memory, thread, and system behaviors.

7.1.7 Enabling request-level reliability, availability, and serviceability granularity

WebSphere Application Server V8 provides a request-level granularity of reliability, availability, and serviceability (RAS) 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 (timeout values, timeout actions, trace settings, and so on) to different sets of requests.

To set up the RAS, you must configure a workload classification document and specify it in the administrative console.

7.1.8 Resource workload routing

WebSphere Application Server V8 provides a feature that enhances availability by configuring failover resources to a data source and connection factory. You create alternate resources for the data source and connection factory. These resources must be identical to the primaries and be compatible with applications. The data source and connection factory can fail over when a failure 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.

7.1.9 External high availability framework for service integration

WebSphere Application Server V8 allows a message engine to be managed by an external high availability framework (for example IBM PowerHA). The message engine is enabled or disabled only when the external high availability framework orders it by the HA manager.

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.

7.1.10 High availability for a WebSphere MQ link

To improve the high availability connection between WebSphere Application Server and WebSphere MQ, with WebSphere Application Server V8, 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 (SIBus) can reconnect to a standby gateway queue manager.

7.2 Scalability

This section provides information about the scalability of the WebSphere Application Server environment.

7.2.1 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 application needs 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 can greatly improve availability and performance. Modifying the scalability approaches of the infrastructure has an impact on the availability and the 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. Then, you can give priority to the most used and most important components.

You can improve performance when adding resources 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, increase by two 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. You have to find the best configuration for one server and multiply that configuration to get the number of servers required to handle the load.

  For example, instead of multiplying by two the number of processors and memory of the server, keep your first server, and add a second server that is identical to the first server.

  Scaling out is also used to improve high availability by limiting single points of failure (SPOF). Scale 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 have to support and maintain multiple machines.

Every additional component, such as processor, memory, or JVM, in your infrastructure has management overhead. 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% overhead. The effective use of each new server is 85%, and the scalability factor is 0.85.

A scalability factor of 1 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 that states that, if one factor of production is increased while all other factors remain constant, the overall returns will reach a peak and then begin to 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 to say that adding *n* additional machines will 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. We decide to scale by creating a cluster and spreading application servers across systems, hoping that this configuration will improve the throughput. At the same time, 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 should be a part of the performance testing. It is crucial that you determine if the scaling techniques are effective and whether they adversely affect other areas. You can 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
– Needs to manage additional servers

7.2.2 Scaling the infrastructure components

This section highlights key points to 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 additional 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 getting 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 machine or add more threads or processes to your existing web server instances

► Horizontal scalability to create multiple instances of the web server on different machines

To support a configuration with multiple web server instances, use load balancers. Be careful when adding servers, and make sure that the load balancer has adequate capacity to avoid shifting the bottleneck from the web tier to the load balancing tier.

Scalable mode: IBM HTTP Server for z/OS offers the unique feature of scalable mode. With scalable mode, the server can start additional instances of itself, offering 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 on the heap and more threads in the different containers (web, Enterprise JavaBeans (EJB), portlets, and so on) to your existing JVMs. By using this approach, you can be limited by the size of the JVM heap and the available memory on the physical machine.

You can also create multiple JVMs. A WebSphere Application Server JVM can be clustered vertically (providing multiple copies of the same JVM on the same physical machine), horizontally (providing multiple copies of the same JVM on the different machines), or both. By using this vertical approach, you can be limited by machine (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 to limit the number of connections, and avoid the additional connection establishment. 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 SIBus and a message engine are key components in a modern infrastructure. WebSphere Application Server offers the ability to scale messaging resources.

You can use the scalability policy provided by WebSphere: one SIB is hosted by a WebSphere Application Server cluster. You can choose either a horizontal cluster, a vertical cluster, or both types of clusters. Each cluster member has one message engine. All the message engines are active at the same time, and each message engine can only run on its own JVM. If the JVM fails, the message engine is unavailable.

WebSphere Application Server manages the workload between all 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 nth number of messages. One key factor to consider in this design is that message order is not preserved, which 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 and Scalability and High Availability policies. WebSphere Application Server V8 features a setup wizard for these policies that you can use 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 database provides a solution to scale out by adding multiple nodes. These solutions can be complex and expensive. Your applications must be multinode aware to take advantage of the configuration and must limit the network traffic between nodes.

7.3 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. You can find more information about the tuning method, the setup of your performance environment, and the load factors in 4.5, "Performance considerations" on page 92. 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.

7.3.1 Performance considerations

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 can handle 10 client requests simultaneously and each request takes one second to process, this site can have a potential throughput of 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 that is submitted by a web browser until the response is received at the web browser.

► A maximum time frame for batch style applications

  Batch applications often run during a defined time frame as a number of hours during the night, for example, to take advantage of low peak hours and to avoid 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 to use all the resources of the machine.

To measure the success of your tests, you need to generate a workload that meets the following characteristics:

► Measurable

  Measurable refers to a metric that can be quantified, such as throughput and response time.

► Reproducible

  The same results can be reproduced when the same test is executed multiple times. Execute 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 realistically represents the stress to the system under normal operating considerations. Execute your tests in a production-type environment with the same infrastructure and the same amount of data.

You can also follow the tuning approach by using a top-down method, described in 4.5.5, "Tuning approach" on page 95, to eliminate bottlenecks.

7.3.2 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 to avoid bottlenecks, it cannot compensate for a poorly written application.

Review the application code itself as part of the regular application life cycle to 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 using application profiling.

For additional information about the application design considerations, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *application design consideration*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

7.3.3 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 impact 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 allow for sufficient concurrent requests to make full use of the application server infrastructure, but it should 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) as follows. 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	Number of threads in the web container
MC	MaxConnections setting in the plugin-cfg.xml

WAS	Number of WebSphere Application Server servers
WEB	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 to avoid network negotiation for each new request between them. Keep in mind that, during this time, those threads cannot answer other requests.

For additional 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 206.

► 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 right garbage policy and to define the minimum and maximum heap sizes. You have to define these parameters based on application behavior. A JVM that spends more than 10% of the time in garbage collection is not efficient and can be tuned. Time lost to free memory is time that is not spent to process application server requests.

For more information about the garbage policies provided by IBM, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *garbage collection policy options*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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 perform garbage collection often enough, which can leave the machine 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 additional information about tuning the JVM, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *tuning the JVM*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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 can wait for the web container. If you adjust the number of threads too high, the server can be inundated with too many requests. For both situations, the consequence is an increase of the response time.

You have to define the web container size in consideration of the entire infrastructure chain in close cooperation with the web server number of threads and the number of sessions of the database.

Enterprise JavaBeans container thread pools

The EJB container can be another source of potential scalability bottlenecks. The inactive pool cleanup interval is a setting that determines how often unused EJBs are cleaned from memory. If the setting is set too low, the application can spend more time instantiating new EJBs 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 message-driven beans (MDB), you can check and configure the message listener thread pool.

Mediation thread pools

If you want to run multiple mediations in your SIBus infrastructure concurrently, you need to configure a mediation thread pool by using the `wsadmin` command-line interface (CLI).

Connection pools

Connection pools are used when the application needs to 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 to manage connection pools for HTTP outbound connections. Configure the content encoding of the HTTP message, enable HTTP persistent connection, and re-send the HTTP request when a timeout occurs.

Service integration bus

A Service integration bus (SIBus) uses several pools and message threshold parameters, which you need to configure properly. Each SIBus 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 have to 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 memory page size of 4 KB, consider taking advantage of this feature and configure the larger memory page size. JVM supports large pages, and Java applications often benefit by using the large pages because of less CPU consumption.

7.3.4 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 perform a lot I/O, directly using read/write instructions, or indirectly using the database, check the storage box response times. Several storage improvements take place at the operating system, network, or storage level. For example, increase the queue depth of disks, adjust the number of possible paths to reach a disk, reorganize data on the storage box, or use high performance disks, such as 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 Etherchannel if your throughput is bounded. Check the operating system network parameters, especially the buffers. If you have high-end servers with several partitions inside, they provide internal networks to improve the latency and throughput.

- ► Operating system

  Memory and CPU can affect performance, and you can configure several parameters to improve performance in this area. When the system is memory or CPU bounded and when the application stacked is tuned, the solution might be to add resources.

  All tuning at the system layers must be defined in close cooperation with the infrastructure team. Changes at this level can impact applications and the entire environment.

  For additional information about tuning the operation system for WebSphere Application Server, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *tuning operating systems*:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

7.4 WebSphere Application Server performance tools

When reviewing the application environment for overall daily monitoring or when identifying bottlenecks, you often need to go deeper into the behavior of the application than what is at the operating system layer. 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 the following tools for the administrator to gather information related to the performance of various components in the Java 2 Platform, Enterprise Edition (J2EE), environment:

- ► IBM HTTP Server monitoring page
- ► 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

7.4.1 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 using the Performance Servlet. With 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 JSPs counters
   – SIP counters
   – Web services counters

► System resources

   – Total free memory
   – Processor usage
   – Components that are controlled outside the WebSphere environment but that are vital in a healthy application state

► WebSphere runtime resources

   – Database connection pools
   – Dynamic caching
   – JVM memory
   – 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)

   – CPU usage
   – HTTP session information
   – J2EE components

► Custom (select your own mix of metrics)

► Extended (basic +)

   – Dynamic cache
   – WLM

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 from statistics that are gathered by the JVM provided by IBM and the Sun

HotSpot technology-based JVM, including 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 using the custom settings.

> **System monitoring effects:** Monitoring a system naturally changes the nature of the system. Introducing performance metrics consumes more resources. Thus, the more statistics that you capture, the more processing power that is required.

IBM Tivoli Composite Application Manager for WebSphere Application Server counters

WebSphere Application Server V8 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 counts.

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 counts 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 Version 8 Information Center at the following address, and search for *IBM Tivoli Composite Application Manager for WebSphere Application Server counters*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

7.4.2 IBM Tivoli Performance Viewer

Tivoli Performance Viewer is included with WebSphere Application Server V8 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 configuration data that is gathered.

► Log performance data by using Tivoli Performance Viewer, to log real-time performance data and to review the data at a later time.

► View server performance logs. You can record and view data that was logged by Tivoli Performance Viewer by using the Integrated Solutions Console.

To avoid gathering too much information, with Tivoli Performance Viewer, you can choose the specific performance modules that you want to monitor. You can use the log analysis tools to detect trends over time. Tivoli Performance Viewer can also save performance data for later analysis or problem determination.

Because Tivoli Performance Viewer runs inside the Integrated Solutions Console, the performance impact depends on which edition of WebSphere Application Server you 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.

7.4.3 WebSphere Application Server performance advisors

Gathering runtime information, performance advisors for WebSphere Application Server can determine diagnostic advice about the environment. The advisors can determine the current configuration for 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

The sections provide more information about these advisors.

Performance and Diagnostic Advisor

The Performance and Diagnostic Advisor is configured through the Integrated Solutions Console. It writes log files to the application server and to the console while 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 is configurable to determine how often data is gathered and advice is written.

Performance and Diagnostic Advisor offers advice about the following components:

► J2C Connection Manager

  – Thread pools
  – LTC Nesting
  – Serial reuse violation
  – Plus various different diagnostic advises

- ► 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 invoked only through the Tivoli Performance Viewer interface of the Integrated Solutions Console. It runs on the application server that you are monitoring, but the refresh intervals are based on selecting the refresh option through the console. Also, the output is routed to the user interface instead of to 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 can require resources and impact performance. Use this advisor with care in production environments.

7.4.4 Request metrics in WebSphere Application Server

Performance Monitoring Infrastructure for WebSphere Application Server provides information about average system resource usage statistics, but 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 metric tracks each step of a transaction and determines 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 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:

- ► On which performance area should you focus?

- ► Is too much time spent on any given area?

- ► How do you determine whether response times for transactions are meeting goals and 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, capturing only those transactions that you want to monitor. For example, you can use an IP address filter to monitor synthetic transactions that always come from the same 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 control the depth of information that is 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 also 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 to use metric logging while implementing the ARM agent monitoring, because the disk I/O can negatively impact 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 call across the ARM API to gather the data.

For more information about ARM, see "ARM" on the Open Group site at:

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.

7.4.5 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 a single point to access to these tools, you can analyze applications, garbage collection files, Java Heapdump, and Java core. This section provides more information about these tools.

For additional information about IBM Monitoring and Diagnostic tools for Java, see the IBM developer kits page at:

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 and 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 dump. 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.

7.4.6 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 the real-time view of the current IBM HTTP Server state.

You can visualize the following information:

- ► CPU usage
- ► The total number of requests served since the server is up
- ► The total traffic size since the server is up
- ► Average about the response time
- ► The number of requests currently running
- ► The number of idle threads
- ► List of requests being processed

7.5 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 alternate 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.

7.5.1 HTTP servers

You can use an IP sprayer component, such as the Edge Component Load Balancer or a network appliance, to perform the load balancing and workload management functionality for incoming web traffic, as illustrated in Figure 7-1.
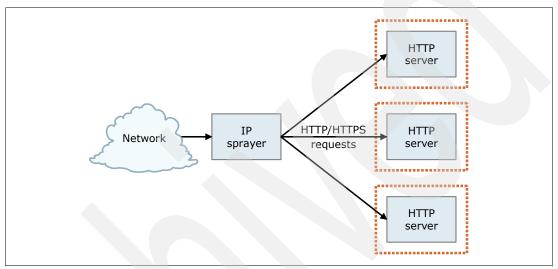
Figure 7-1   IP Sprayer or HTTP server workload management

Depending on which solution you implement, you might have these routing options:

► Dynamic weight, where the load balancer calculates the load of each HTTP server and route dynamically to the one that is less busy

► Static weight, where each member has a weight and the load balancer spreads the requests depending 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. It allows client requests to be directly route 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 efficient, 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, to avoid multiple SSL handshakes, which are resource consuming, you can route 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.

7.5.2 DMZ proxy servers

As with HTTP servers, you can use an IP sprayer component, such as the Edge Component Load Balancer, to perform load balancing and workload management functionality for incoming web traffic. In addition, the DMZ proxy server provides workload management capabilities for applications that are running in an application server.

7.5.3 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 facility to provide load balancing and affinity between nodes in a WebSphere clustered environment. Workload management can be an important facet of performance. WebSphere uses workload management to send requests to alternate members of the cluster. WebSphere routes concurrent requests from a user to the same application server to maintain session state.

Workload Manager (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 to assign SLA types of performance goals. This definition is done on transaction-level granularity compared to server-level granularity on the 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*, that 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 14.1.7, "Workload management for WebSphere Application Server for z/OS" on page 392, or the *WebSphere Journal* article "Understanding WAS for z/OS: Sophisticated J2EE platform is different and powerful" at:

http://websphere.sys-con.com/read/98083.htm

7.5.4 Clustering application servers

Clustering application servers that host web containers automatically enables 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 using HTTP or HTTPS, as illustrated in Figure 7-2.
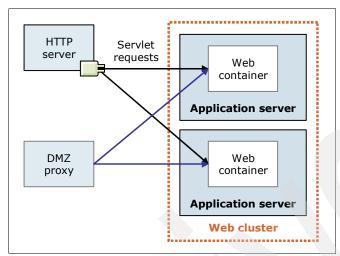
Figure 7-2   Plug-in (web container) workload management

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 to those cluster members with the higher weight value more often. 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 the 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 7.8, "Session management" on page 234.

On the z/OS platform, the assignment of transactions to cluster members is performed on real-time achievement of defined performance goals. This assignment process has the benefit that the system can differentiate between light requests that use only a small fragment of performance and heavy requests that use more processor capacity.

For more information about web server plug-in workload management in a cluster, see the technote *Understanding IBM HTTP Server plug-in Load Balancing in a clustered environment* at:

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=plugin+workload+ma
nagement&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 7-3.
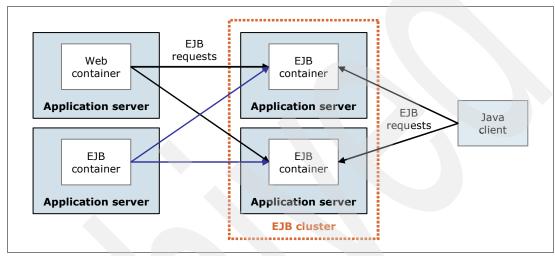
Figure 7-3   EJB workload management

To route the EJB requests, WebSphere 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 operating 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 that have been assigned to the members of a cluster. For example, if all servers in the cluster have the same weight, the expected distribution for the cluster is that all servers 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 than the lower weight value servers. This policy ensures the desired distribution based on the weights that are assigned to the cluster members.

► 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 impact the routing:

► Process affinity

   If an EJB is available in the same cluster member than 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 to ensure that, when a client (for example, a servlet) calls an EJB, WLM attempts to select the EJB on the same system as the client, eliminating network communication.

7.5.5 Scheduling tasks

WebSphere Application Server provides a scheduler service that can schedule actions to happen with the following frequencies:

► Only one time
► 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 executed for indefinite repetitions and long time periods. Scheduler tasks can be EJB-based tasks, 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 executed in the work manager that is associated with the scheduler instance. You can control the number of actively running tasks at a given 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.

7.6 High availability

High availability is also known as *resiliency*. High availability is the ability of a system to tolerate an amount of failures and to remain operational. This section provides several considerations for high availability.

7.6.1 Overview

High availability means that your infrastructure continues to respond 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 impacts 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% of system availability represents 8.76 hours of outage in a single year.

Table 7-1 shows the level of availability and the calculated downtime per year.

Table 7-1   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 is a simple process 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. For a WebSphere Application Server infrastructure that 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 will be 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 additional information about this topic, see the IBM developerWorks article "IBM WebSphere Developer Technical Journal: Planning for Availability in the Enterprise" at:

http://www.ibm.com/developerworks/websphere/techjournal/0312_polozoff/polozoff.html#sec1

Because it is likely that the complete environment is made up of multiple systems, the goal is to make the entire system as available as possible 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.

7.6.2 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 an adapter teaming to allow a server to bind one IP address to more than one adapter and then provide failover facilities for the adapter. This configuration should 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 offers the possibility to duplicate data on 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 for routers. These devices also support duplicate power supplies, multiple controllers, and management devices.

7.6.3 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, 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 given instance. 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 about this topic, see 7.6.7, "WebSphere Application Server high availability features" on page 225.

7.6.4 Data availability

In a WebSphere Application Server environment, data availability is important in multiple places. Critical areas for data availability are as follows:

- ► 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 an external shared disks through storage area network (SAN), network-attached storage (NAS), or other facilities to provide the exact same disks to different machines. For an active/passive solution, when a failure occurs, the disks are mounted on the standby node, and the database is restarted. In opposition to the active/active solution, all the nodes are active at the same time, and 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 to 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 that must be mitigated by using an external hardware clustering or high availability solution. Memory-to-memory replication can reduce the impact of failure. Depending on the level of replication, if more than one server fails, the data that is held on these servers cannot be retrieved on other cluster members.

In contrary to the HTTP session persistence, stateful session EJB availability is handled by using memory-to-memory replication only. Using the EJB container properties, you can specify a replication domain for the EJB container and enable the stateful session bean failover 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 should not be confused with entity EJB.

7.6.5 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, or 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, the idea is to create multiple copies of an application component and have all of these copies 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.

7.6.6 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 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.

7.6.7 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 (HA manager) to eliminate SPOF. The HA 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. HA manager runs by default in each server, as illustrated in Figure 7-4 on page 226.
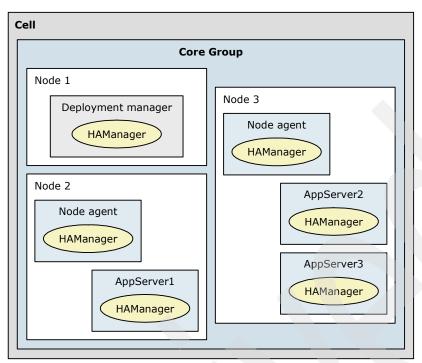
Figure 7-4   Conceptual diagram of a core group

For certain functions (such as transaction peer recovery), HA manager takes advantage of fault tolerant storage technologies such NAS, which significantly lowers the cost and complexity of high availability configurations. HA 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.

HA manager continually monitors the application server environment. If an application server component fails, HA manager takes over the in-flight and in-doubt work for the failed server. This process introduces some overhead but significantly improves application server availability.

HA 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, HA manager supervises the JVMs of the application servers that are core group members. HA manager 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 and is tolerant of poor performing application servers, which might happen if the application server is overloaded, swapping, or thrashing. This method is preferred for determining a JVM failure if you are using

multicast emulation and are running enough JVMs on a single application server to push the application server into extreme processor starvation or memory starvation.

▶ A JVM is marked as failed if it stops sending heartbeats for a specified time interval.

This method is referred to as *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, alternate 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, especially during times 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, and 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 running in that JVM are activated on a peer JVM after the failure is detected. This peer JVM is already running and eliminates the normal startup time, which potentially can be minutes, which is a key difference to using operating system-based high availability. HA manager usually recovers in seconds while operating system-based solutions can take minutes.

When an application server fails, HA manager assigns the failing application servers work to another eligible application server. Using shared storage for common logging facilities (such as the transaction logs) allows HA manager to recover in-doubt and in-flight work if a component fails.

> **Additional resource:** For a testing routine that you can use to determine whether your shared file system is suitable for use with HA manager, 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 and 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, which 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 have to create multiple core groups.

> **Core group with more than 50 servers in a cell:** Many application servers in a cell increases the overhead 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, you need to create a tunnel access point group to establish a core group bridge tunnel between the core groups that are running on both sides 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 only be a member of 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 entirely 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, if all of these high availability groups are 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 at a given time. 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, but 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 policy Clustered TM Policy that includes `type=WAS_TRANSACTIONS` as part of its match criteria.

- ► When transaction manager code joins a high availability group, the HA 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, such as 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.

Using the same match criteria: 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 SIBus 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 reliability that were being processed or queued will 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 the failover seamlessly, the queue information and message data must be stored in a shared location that is reachable by all the members of the cluster, by using either an external database or a shared disk environment.

For additional information about the availability policy, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *messaging engine policy assistance*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

> **Embedded Derby database:** 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 running the Derby engine. Therefore, there is no ability to have multiple servers communicating with the same shared file system.

Resources availability

WebSphere Application Server V8 introduces 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 alternate resources for data source and connection factory. These resources must be identical to the primaries and be compatible with applications. Then, add custom properties to configure the availability behavior.

Deployment manager high availability

The deployment manager (dmgr) process is not considered a SPOF. If the process fails, the application servers managed by the deployment manager do not fail, and the applications continue to run. However, it is impossible to manage the entire cell. This section presents one active/passive solution to make the deployment manager more fault tolerant. This solution is based on shared disks (disks that are accessible by both active and passive machines) that contain the deployment manager profile.

The WebSphere Application Server binary files are installed on both machines (current and standby). The deployment manager profile is created only one time on a specific file system. The disks of this file system come from a SAN or NAS and are accessible by the two machines. The profile must be installed with an IP alias.

You can perform the failover by using clustering software (such as PowerHA) or manually by completing the following steps:

1. On the current machine:

   a. Stop the deployment manager, if it is not already stopped.
   b. Release the disks.
   c. Remove the IP.

2. On the standby machine:

   a. Attach the same disks, and retrieve the deployment manager profile.
   b. Add the same IP.
   c. Restart the deployment manager process.

You can run in failover as long as you want with this configuration.

7.7 Caching

Caching is a facility to offload work to one or more external devices so that the application server is not 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 (for example exchange rates).

A combination of caching at the 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 to 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

7.7.1 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 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.

> **Stabilized feature:** 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 cell, and it is fully manageable by using the administrative console. WebSphere Proxy Server runs in the secure zone of your infrastructure and allows you to offload requests processing for the rest of the infrastructure. Then WebSphere Proxy Server can 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. By using the DMZ Secure Proxy Server, you 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 performing the caching function. This operating system can include custom file systems (that offer higher performance than the operating system file system) and a significantly reduced instruction set. By placing dedicated appliances instead of software caching in your architecture, you can reduce total cost of ownership, because these appliances do not have to be managed as strictly as machines 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 application serving network to return content. 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 responds to the request. If servlet caching is enabled in the application server and the output is edge cacheable, the application server adds a Surrogate-Capabilities header with caching information. The plug-in stores, in

the cache, the application responses, builds the page with all the nested components, and returns the answer to the client.

7.7.2 Dynamic caching

WebSphere Application Server, 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. Cacheable objects are defined inside the `cachespec.xml` file, which is 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 functionality provided by the data replication service (DRS), you can replicate or copy the data to the others members of the cluster to save execution time and resources (see 7.9, "Data replication service" on page 238). 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.

7.7.3 Data caching

Data caching is used to minimize back-end database calls while assuring 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, or a subset, 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 offer the ability to keep the 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 disks access.

Application data caching

Another option is to cache some of the database or web page data inside an application 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. The key is to 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.

7.8 Session management

This section introduces the session management concept and explains how you can manage the sessions with WebSphere Application Server.

7.8.1 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 allows linking 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 to manage the order or 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 execute that request. Either the same server will process all requests that are associated with dedicated state information, or the state information can be shared by all servers that require it. From a performance view, it is better that the first server that served the request continues to serve the other ones to avoid exchanging the state data across the servers and minimize the communications overhead.

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 and 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 given client request, a particular server is best suited for processing that request.

The session manager for 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.

7.8.2 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 the planning considerations for each of these considerations.

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 sessions are mainly 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 have been encoded programmatically by the web application developer.

URL rewriting can only be used 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, you must use an affinity mechanism for the web servers when SSL session ID is used as the session tracking mechanism.

SSL tracking: 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, either cookies or URL rewriting must be used to maintain session affinity. The cookie or rewritten URL contains session affinity information that enables the web server to properly route requests back to the same server after the HTTP session has been created on a server. The SSL ID is not sent in the cookie or a rewritten URL but is derived from the SSL information.

The disadvantage of using SSL ID tracking is the performance degradation due to the SSL overhead.

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 selecting SSL session ID tracking, consider also selecting cookies or URL rewriting to maintaining session affinity. The cookie or rewritten URL contains session affinity information that 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 whether to store the session data as follows:

► Local sessions (non-persistent)
► Database persistent sessions
► Memory-to-memory replicated persistent sessions

The last two options in this list allow session data to be accessed by multiple servers and should be considered 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 permits an unlimited number of sessions. If you choose this option, monitor the session cache size closely.

  Overflow: Session overflow is enabled by default in WebSphere Application Server V8.

► 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 through a data source.

The *use multirow schema* setting gives you the option to use multirow sessions to support large session objects. With multirow support, the WebSphere 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 enables 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 using WebSphere Application Server Network Deployment.

7.9 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 are the three consumers of 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 server sharing data (for example sessions). For each domain, you have to define how the data is replicated:

► To one server (single replica)
► To every server (entire domain)
► To a defined number 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 only stores 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 only broadcasts or sends 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 data, this mode uses the most memory on each server.

There is a relationship between the degree of replication and the performance. 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, by configuring more replicas, your system becomes more tolerant of possible failures of application servers because the data is backed up in several locations.

7.10 Checklist for performance, scalability, and high availability

Consider the following items as you plan performance, scalability, and high availability. It includes additional resources to assist you:

► 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 if the appropriate mechanisms are in place for workload management and failover. As part of this, consider where applications will be deployed (see 8.12, "Mapping applications to application servers" on page 268).

► Implement a monitoring system to watch for performance problems and to assist in determining if adjustments are necessary.

► Monitor the following areas as potential physical bottlenecks:

– Network load balancers
– Firewalls
– HTTP servers
– Application servers
– Database servers
– LTPA providers

► Examine initial settings for performance tuning parameters, adjust if necessary, and re-evaluate periodically:

– JVM garbage policy, 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 in 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 using the default messaging provider, consider the following areas:

  - Quality of service settings
  - Bus topology

7.11 References

For more information, see the WebSphere Application Server Information Center at the following address, and search for *monitoring* and *performance*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

8

Application development and deployment

The development and deployment of WebSphere applications is a topic that 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 and understand to help in that planning.

This chapter includes the following sections:

► Application deployment features in WebSphere Application Server V8
► End-to-end life cycle
► 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

241

8.1 Application deployment features in WebSphere Application Server V8

This section provides an overview of the features for application development and deployment that were introduced in WebSphere Application Server V8:

► 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. 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 needs including editing of deployment artifacts, script development and testing, and application deployment and debugging. This tool is not intended for general application development.

► Rational Application Developer Standard Edition for WebSphere Software V8

WebSphere Application Server V8 offers integration and support with Rational Application Developer Standard Edition V8. With this tool, developers can design, develop, deploy, and maintain Java Platform, Enterprise Edition (Java EE) applications on the WebSphere Application Server V8 environment.

This tool supports all Java EE artifacts that are supported by WebSphere Application Server V8. 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. They also include new integration with Open Services Gateway initiative (OSGi) programming model. It also supports older versions of Java specifications (Java 2 Platform, Enterprise Edition (J2EE) 1.2, 1.3, 1.4, Java EE 5) and previous versions of WebSphere Application Server (V6 Remote, V6.1, V7).

► Rational Application Developer for WebSphere Software V8

Rational Application Developer for WebSphere Software V8 offers the same development capabilities as the Standard Edition plus additional enhancements such as team productivity, problem determination, and enterprise connectivity. This tool uses collaboration during code analysis and debug operations, optimized unit testing, and static analysis and connectivity to enterprise information systems through WebSphere Adapter Support.

► A broad set of integrated standards-base programming models

Many of the core programming models in WebSphere Application Server V8 were available by using feature packs in the version 7. Now they are available for immediate use for WebSphere Application Server V8. The following programming models are included:

– Java Batch programming model

Use this model to build robust batch applications for performing long running bulk transaction processing and computationally intensive work.

– Communications Enabled Applications (CEA) programming model

CEA provides Representational State Transfer (REST) and web service interfaces so that existing applications can quickly take advantage of communication features, involving phone calls and web collaboration.

– XML programming model

With this programming model, you can develop applications that process data by using a new standard XML technology such as Extensible Stylesheet Language (XSLT 2.0), XML Path Language (XPath 2.0), and XML Query Language (XQuery 1.0). With this

new version of XQuery, you can query large amounts of data stored in XML outside of a database. All together, this technologies offer the benefit of simplifying application development and improving its performance and reliability.

– OSGi applications programming model

With this model, you can develop and deploy modular applications that use both Java EE and OSGi technologies. By using this model, you have 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. With WebSphere Application Server V8, support for OSGi applications includes the capability of deploying web applications that use the Java Servlet 3.0 Specification and Java Persistence API (JPA).

Additionally, you can update a running application that only impacts those bundles that are affected by the update. You can also extend and scale running applications as business demands it, without changing the underlying application. OSGi support includes the ability to use an integrated bundle repository and configure the locations of external repositories to support reuse through the provisioning of bundles to applications.

– Service Component Architecture (SCA) programming model

SCA accelerates application delivery and management in service-oriented architecture (SOA) environments. You can create service compositions by using Plain Object Java Objects (POJOs); EJB 2.1, 3.0, and 3.1; OSGi applications; Spring components; Java Servlets; and JavaScript for Asynchronous JavaScript and XML (AJAX). This model is based on the Open Source Apache Tuscany project with IBM.

– 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 EE 6 support

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:

– Support for the EJB 3.1 specification

– Support for the Context and Dependency Injection (CDI) 1.0 specification at a runtime level that uses the Apache OpenWebBeans 1.x implementation

– JPA 2.0

– Java Servlet 3.0

– Java API for RESTful Web Services (JAX-RS) 1.1

– JSF 2.0

– JSP 2.2

– Bean Validation 1.0

– Java Architecture for XML Binding (JAXB) 2.2

– Enterprise Web Services 1.3

– Java API for XML-Based Web Services (JAX-WS) 2.2

– Java EE Connector Architecture (JCA)1.6

For more information, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *what has changed in this release*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

► Simplified development of server-side REST applications 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.

► Integrated web services support

– JAXB 2.2 provides an easy and convenient way to map Java classes and XML schema for simplified development of web services. Version 2.2 provides minor enhancements to its annotations for improved schema generation and better integration with JAX-WS.

– 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.

► 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 (SAR) application resource

8.2 End-to-end life cycle

The WebSphere Application Server V8 environment and its integration with Rational tools offers developers support at every stage of the application development life cycle.

This life cycle 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)

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 gathering experiences and practices that have lead 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 feedback into itself and that software grows as the life cycle 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 life cycle 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, phases of Inception, Elaboration, Construction, and Transition 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. Clearly, the requirements discipline is more active during the earlier inception and elaboration phases, for example.

Figure 8-1 provides an overview of the RUP.
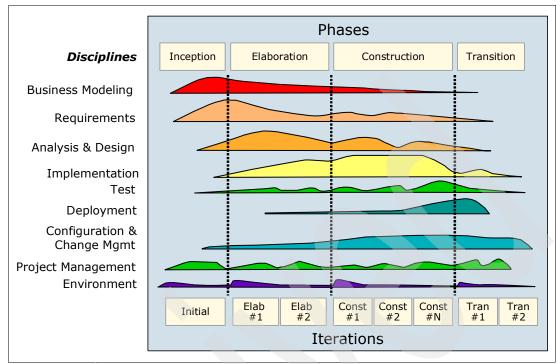
Figure 8-1 Rational Unified Process overview

RUP maps disciplines to roles. The various roles break down into the following basic sets:

► Analysts
► Developers
► Testers
► Managers

Members of the team can take on more than one role. 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, or testing, map to specific pieces of Rational software and 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

8.3 Development and deployment tools

Several tools in the WebSphere Application Server V8 environment help in the development and deployment of applications. All editions of WebSphere Application Server V8 include a full licensed version of the IBM Assembly and Deploy Tools for WebSphere Administration

and a 60-day trial version of the Rational Application Developer Standard Edition for WebSphere Software V8.

Rational Application Developer for WebSphere Software V8 supports all features of WebSphere Application Server V7 and 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 (V6.1 and V7) as an integrated test environment. V6 is supported as remote integration only. It includes all Eclipse 3.6 features.

8.3.1 IBM Assembly and Deploy Tools for WebSphere Administration

IBM Assembly and Deploy Tools for WebSphere Administration is targeted to help 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 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 servers.
► Debug applications on WebSphere Application Server V8.

8.3.2 Rational Application Developer Standard Edition for WebSphere Software V8

Rational Application Developer for WebSphere Software Standard Edition V8 includes all of the features of IBM Assembly and Deploy Tools for WebSphere Administration and the capabilities to design and develop your applications. Additionally, it provides productivity enhancement features to get started with application development and testing.

In addition to the features in IBM Assembly and Deploy Tools for WebSphere Administration, Rational Application Developer for WebSphere Software Standard Edition V8 brings the following features:

► Full support for Java EE 6, including EJB 3.1

► Full support for Java Development Kit (JDK) 6

► Portal application development with support for WebSphere Portal V6.1 and V7 beta test environments

► EJB universal test client and generated EJB test client

► Ant scripting and JUnit testing framework

► Web 2.0, OSGi, JPA 2.0, SCA, XML, CEA, and web services development features

► 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

► Find and deploy to WebSphere or Portal instances in the IBM Smart Business Development and Test (SBDT) Cloud

8.3.3 Rational Application Developer for WebSphere Software V8

Rational Application Developer for WebSphere Software V8 offers all the features in Rational Application Developer Standard Edition plus additional capabilities for team productivity, problem determination, and enterprise connectivity. It includes the following features:

► WebSphere Adapter Support for third-party products such as SAP, PeopleSoft Enterprise, Siebel, Oracle E-Business Suite, and JD Edwards

► Integration with IBM Rational Team Concert™ and IBM Rational ClearCase® so that you can perform management operations within the development environment, have an integrated view of projects, and increase collaboration and team productivity

► Code quality, testing, and deployment tools, such as the enhanced runtime analysis to detect memory leaks or thread locks

► Unified Modeling Language (UML) modeling function

8.3.4 Monitored directory

The monitored directory feature in WebSphere Application Server V8 makes it easier to install applications. Still present in this version (and in previous versions) is another deployment tool called *WebSphere Rapid Deployment Tools*. Table 8-1 compares the monitored directory feature and Rapid Deployment Tools.

Table 8-1   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.

IBM i exception: The possibility of 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, follow these steps:

1. Log in to the Integrated Solutions Console, and then click **Applications** → **Global deployment settings**.

2. 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 for in seconds if do not want to keep 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 instructions and more detailed information about the monitored directory configuration, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *setting monitored directory deployment values*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

Installing, updating, or 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 clusters, 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, Java archive (JAR), web archive (WAR), or SAR 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. First the application that is already deployed stops, the new module or application gets 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.

SystemOut.log file: 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 execute the desired action.

For more information about how to install, update, or uninstall applications with properties files or move them to monitored directories, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *installing enterprise application files by adding them to a monitored directory*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

8.3.5 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 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 Rational Application Developer Standard Edition for WebSphere Software V8. 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.

8.4 Naming conventions

Spending extra time on application-related naming concepts pays off in practice, because it can reduce the time spent on analyzing the source of issues during standard operations of future Java EE applications.

8.4.1 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 pound sign (#) in the name of the files, because this causes the deployment to fail.

Generally, a form of the Version, Release, Modification, Fix (VRMF) schema is used to organize code and builds, and 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, and 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. Appending the date of built or release to the EAR name can be substituted if a proper logging between the version number and the corresponding date is kept.

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, which can often 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 and the name is changed to `log4j-1.2.5.jar`,

each developer must update the class path settings in their workspace, which costs time. It is better to use a Source Code Management (SCM) system and label the new JAR file as version 1.2.5, but keep the file name constant, such as `log4j.jar`.

To keep track of 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 versions of all included components, information about the tools used to build it, and the machine 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.

8.4.2 Naming for resources

When naming resources, associate the resource to both the application using 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 Integrated Solutions 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 your resources according to the application that is using them.

To group and sort resources in the Integrated Solutions Console, you can also use the Category field, which is available for all resources in the Integrated Solutions Console. In this text 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.

8.5 Source code management and collaboration

In development, it is important to 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, it is equally important to track the version of the build tools and which machine was used to generate a build. Not all problems are due to bugs in source code.

Developers produce code and usually use an IDE, such as Rational Application Developer for WebSphere Software V8, to do that. Code in an IDE is stored in a workspace on the file system that is locally on the machine of each developer. As the project continues, and perhaps new members join the team, the code grows, and it becomes necessary to manage the code in a central master repository.

Regardless of the size of the developers group, it also becomes necessary to manage the code merge 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, can be time consuming, and sometimes can be nearly impossible.

Another important aspect during software development is how communication is done between developers, stakeholders, and other people who somehow relate to the development process. In large organizations, it is also common that the development team or stakeholders are dispersed around the globe. In such situations, it is difficult to keep clear visibility on how the development process is going and if it is really meeting the business requirements as expected.

Collaborative software helps to improve how the different teams or people communicate with each other. It improves team productivity by interconnecting the right people that can give valuable feedback on time and helps identify defects when it costs less to fix them.

Source code management (SCM) systems and collaborative systems are valid approaches to keep the right control on the source code of the application and ensure efficient communication across the team,

8.5.1 IBM Rational ClearCase

IBM Rational ClearCase organizes its code repositories as Versioned Object Bases (VOBs). VOBs contain 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, views, and coordinates the checking in and checking out of VOB data to and from views.

As the role-based model suggests, Rational ClearCase is an SCM 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 and 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 use of the software in 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 made changes on the code or artifacts and when those changes were made.

In short, although Rational ClearCase is an SCM 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/

8.5.2 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 make changes to the project. With this model, individuals can work on a CVS team project, share their work with others as changes are made, and access the work of others as the project evolves. A special branch, referred to as HEAD, represents the main course of work in the repository. (HEAD is often referred to as 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 SCM repositories can be converted to CVS.
► Many free client applications, such as WinCVS, are available.
► It can store text and binary files.
► It handles versioning and branching.
► It is a centralized repository.

8.5.3 Subversion

Subversion is an open source version control system that is available at no coast 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/

8.5.4 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.

One of the most beneficial aspects of collaboration is that it facilitates customer or user involvement during the development of the applications. Customer and user feedback during

this phase is valuable. If application development goes according to what they expect from the beginning, the success probabilities are higher.

In a single integrated environment, Rational Team Concert offers the necessary tools to enhance productivity during the development life cycle of your applications. It uses a Web 2.0 oriented portal, with customizable views where users can see relevant information about the project they are working on, including the following information:

► 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 SCM 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 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" on page 248)

    – 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 257)
    – Maven
    – CruiseControl build system

► Version Control:

    – Rational ClearCase
    – CVS (see "Concurrent Versions System" on page 253)
    – Subversion (see "Subversion" on page 253)

► 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

8.5.5 Choosing the right tools to use

The right choice of tools depends on several factors, including your development environment and needs. The topics in this section serve as a guide when planning for the right tools to assist in meeting the source code management and collaboration needs of your development projects.

Current software and processes

To some extent, the choice of tools depends on the existing situation (if any), the SCM, communication, and current development process requirements and their requirements in the future. If a team uses CVS, Subversion, and communications tools, and an existing and a successful development process exists, Rational ClearCase or Rational Team Concert might not be necessary, especially if the size and complexity of requirements is not likely to grow in the future. If the current development process does not allow clear visibility of the project status or makes it difficult for the team members to work together and coordinate the development, Rational Team Concert can be a solution in these situations.

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 group of 10 or fewer developers. Both tools can be upgraded later to keep development control as the team continues growing.

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 precisely when team members are in separate geographical places. It also makes sense when the team is too large and regular meetings or email messages are not agile enough to coordinate the development activities and keep track of 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 life cycle. 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 right 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 SCM perspective, CVS and Subversion are possibly a cheaper option because they are available at no cost and have a large user base, which means cheaper skills. In terms of hardware, it is likely that hardware costs for hosting CVS or Subversion are cheaper because of their smaller footprint. However, these economies might be false. The limitations of CVS and Subverison can cause a team to migrate 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, in addition to the initial cost, you must consider other 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 for CVS and Subversion such as the UCM. If the team's organization does not have its own change management process, create such a process in the proper place. Likewise, Rational Team Concert can help improve change management processes if any, and if not, can help in putting one into place. Its tracking capabilities and collaborative change communication can help large organizations to gain control over the changes going 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, 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 if 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. In matters of cost, there might be false economies.

8.6 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.

► The development team, testers, and customers have to wait to get the code out to 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 are occurring from different applications that need to be deployed.

► Requested application changes are not completed on time, resulting in customer dissatisfaction.

► From the business point of view, a manual process means 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 considerably reduce the turnaround time for a build. You can

also easily recreate what is in each of your environments and ensure that the code base is under configuration management.

Apache Ant

Several tools, such Apache Ant, Apache Maven, or 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 perform 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:

- ► Install and uninstall applications.
- ► Run EJB 1.x, 2.x, and 3.x deployment and JSP precompilation tools.
- ► Start and stop servers in a base configuration.
- ► 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 SCM repository, and 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.

For more detailed information about Ant, see the Apache Ant website at:

http://ant.apache.org/index.html

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 your organization, so that a build automation process can be implemented in different teams with varying sizes.

The Rational Build Forge tool helps software development teams be efficient during build and deployment of their applications by providing the following benefits:

- ► Automates build and deployment activities during the development life cycle

- ► 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

- ► Includes an add-on called IBM Rational Automation Framework for WebSphere that you can use 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

8.7 Automated deployment process

Automating application deployment is something to consider if 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 it also needs to create other WebSphere objects and configure the web servers, file systems and others.

Different approaches exist to automate the deployment process:

► Depending on the operating system, shell scripting can be used to deploy the applications with Java TCL (Jacl) and Jython (Java Python) scripts.

> **Jacl:** Jacl stabilized in WebSphere Application Server V7.
>
> **More information:** For information about administrative tasks (such as deployment and application management) that can be done with scripting using `wsadmin`, see WebSphere Application Server Version 8 Information Center at the following address, and search for *overview and new features for developing and deploying applications*:
>
> http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

► Use the built-in capability in WebSphere Application Server V8 for automated deployments. For more information, see "Monitored directory" on page 248.
► Use other tools such as Rational Build Forge (see "Rational Build Forge" on page 257) with the Rational Automation Framework for WebSphere add on.

Scripts that are already built using Jacl and Jython are publicly available to automate common administrative tasks. Although the scripts were not developed for WebSphere Application Server V8, they can be used as examples and modified to accomplish the desired task.

For more information, see the IBM developerWorks article "Sample Scripts for WebSphere Application Server" at:

http://www.ibm.com/developerworks/websphere/library/samples/SampleScripts.html

8.8 Automated functional tests

A functional test verifies that an application is working as expected. Functional tests are made from the user perspective, ensuring 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 usually sacrificed. However, keep in mind that testing your applications before placing them into production can mean a significant cost reduction and can avoid damaging your image when service is down because of non-working applications.

Depending on the business size, a functional test can be incorporated to test critical and complex applications only. It is important to keep in mind that testing involves certain levels of investment similar to any other process during the application development life cycle. 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/

8.9 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, a proper test environment often consists of multiple test environments.

Figure 8-2 shows an overview of a recommended test environment setup.
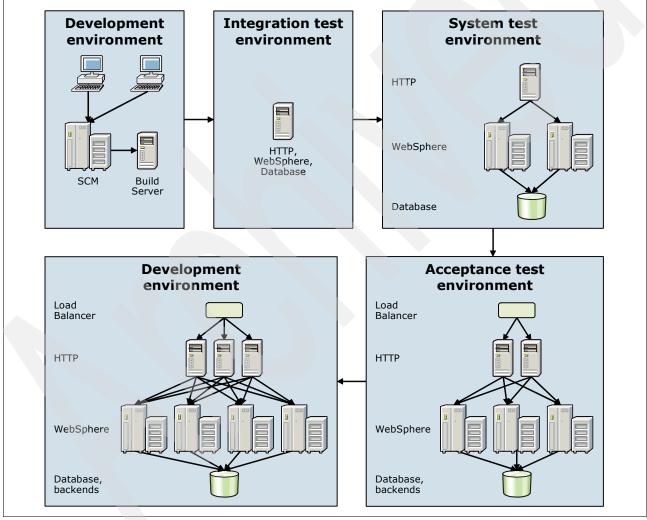
Figure 8-2   Test environments

Test cases must be developed according to system specification and use cases. You must do this task 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 functionality 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 system that is being developed, the project size, the budget constraints, and so on. 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, while larger environments might need a deployment manager for a particular cell environment.

8.9.1 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 execute build scripts; export or otherwise package the application into an EAR file; and deploy that file on a test server for every small change that is made. This capability makes it easy and quick to test applications while developing them and increases developer productivity.

Developers are also responsible for performing unit testing of their own code. Most all tests performed for the system are executed in this environment. The primary goal is to remove obvious code bugs. The developers work against, and share code using, the SCM system. The development environment is most often a powerful desktop machine.

When developers commit their code to the integration stream in the SCM system, a development lead or integration team usually performs a clean build of the whole application, bringing together code developed by different developers. This process is usually done on a special build server and is controlled by automatic build scripts (see 8.6, "Automated build process" on page 256). This server might need to a copy of the IBM Assembly and Deploy Tool or Rational Application Developer for WebSphere Software V8 installed.

The development team should also create a Build Verification Test process (see 8.8, "Automated functional tests" on page 258), where each new build is executed 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, a developer can discover methods that perform 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 cost. For more information about this tool, see the developerWorks article "IBM Monitoring and Diagnostic Tools for Java - Health Center Version 1.3" at:

http://www.ibm.com/developerworks/java/jdk/tools/healthcenter/

8.9.2 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 among all system components 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. These tests should have the application that is free of explicit code issues, such as syntax or compilation errors.

Because the production environment is often not the same platform as the development environment, start testing on the target platform as early as possible in the test phase. 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. Usually no more than 5–10 developers should share a single integration test environment. If a developer needs to perform tests that might damage the environment, use a dedicated environment. If the machine 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.

8.9.3 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 has tested 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.

If the tests in the previous test stages have been less formal, a key aspect of the system test is formality. 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, database, and so on 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.

8.9.4 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, often it is 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, but needs to contain all the same components, same brands, same software patch levels, and the same 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 if 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 proper 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 given requirements. Then, when the application is moved into production, the performance is poor.

Therefore 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.

8.10 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 using tools such as IBM Assembly and Deployment Tool or Rational Application Developer for WebSphere Software V8. Other settings, such as the JVM maximum heap size and database connection pool size, are stored in the WebSphere Application Server configuration repository and modified 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.

8.10.1 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 (for example, if the application has long-running transactions). This category should 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. And during development, the OrderApplication might work with a 256 MB heap, but in the busier production environment, it might need a 1 GB heap size to perform well. These options should not move with the application between environments. They need to be tuned depending on the environment.

► Environment-specific

This category includes configuration options that are specific to a deployment environment but that are common to all applications running 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.

8.10.2 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 it in a clustered environment.

You can manage the settings stored in the WebSphere configuration repository (such as the JVM maximum heap size). In doing so, it is common to develop scripts that are run as part of an automatic deployment to configure the settings correctly after the application has been 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, it is important to develop the applications by taking advantage of the configuration flexibility that the application server offers and avoid using hardcoded connections to back-end systems, such as embedded direct Java Database Connections (JDBC).

However, application-internal configuration settings are often stored in Java property files. These files are plain text files with key-value pairs. Java has provided support for reading and making them available to the application by using the `java.util.Properties` class since Java 1.0. Although you can use databases, Lightweight Directory Access Protocol (LDAP), JNDI, and so on to store settings, 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 have sensitive information, such as passwords or IP addresses, that is stored in property files and you want to prevent casual observation of that information. 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 if it is necessary to store sensitive information in property files. The **PropFilePasswordEncoder** is in the `profile_root/bin` path. Consider the simple property file in Example 8-1.

Example 8-1   Property file

```
userId=myUser
userPassword=myPassword
```

To encode the value of `userPassword`, use the following command:

```
PropFilePasswordEncoder path_to_property_file\myPropFile.props userPassword
```

> **PropFilePasswordEncoder utility:** 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 Version 8 Information Center at the following address, and search for *encoding passwords in files*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

In a clustered environment where the same application runs on multiple servers distributed across different machines, 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 will 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 using the class loader, you can use the code snippet in Example 8-2.

Example 8-2   Loading a property file using the class loader code snippet

```
Properties props = new Properties();
InputStream in =
MyClass.class.getClassLoader().getResourceAsStream("my.properties");
props.load(in);
in.close();
```

Property files packaged in a JAR file in the EAR file are a good solution for property files that should not be modified after the application has been deployed. (The application-specific category is explained in 8.10.1, "Classifying configuration settings" on page 263.)

If you want to make the property files easily accessible after the application has been deployed, store them in a folder outside the EAR file. To load the property files, you make the folder available on the class path for the application (and use the code snippet in Example 8-2). Alternatively, you use an absolute path name and the code snippet in Example 8-3 (considering that the file to load is the `/opt/apps/OrderApp/my.properties` file).

Example 8-3   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, which is not what you want to do. A better solution is to 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, you specify the name of the folder that holds the property files, such as the /opt/apps/OrderApp folder, in the Classpath field for the shared library.

A lesser known, but better, approach to access property files is to use URL resources. Although this approach is not explained in detail here, but the following steps describe it:

1. Create a folder on your system that holds the property file.

2. Using the Integrated Solutions 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 Java EE-compliant because it does not rely on the java.io package for file access, which is prohibited according to the Java EE specification. This method also gives you the opportunity to load the property files by using HTTP and FTP so that you can 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, it is convenient to 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 unless the folders are created under the deployment manager cell configuration folder, which is *dmgr_profile_home*\config\cells*cell_name*.

By creating a folder, such as appconfig, 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 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 point 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. Therefore, instead of using MaxMemory or Timeout, use MaxMemoryMB and TimeoutMS to indicate that the max memory should 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.

If you store property files that need to be changed between different environments inside the EAR file, you might discover that problems are involved with that approach, 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 execute 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 have to 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 only extracts from the EAR file 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 and 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 machine. 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.

8.11 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 right topology to minimize the outage of the applications during upgrades. You need to plan how to make your application server processes highly available in case the application upgrades need a server restart and how to make that process transparent to the user. The topology selection criteria is provided in 5.2, "Topology selection criteria" on page 108.

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, the system administrators might only be able 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 (or multiple applications if they use the same database) to migrate the database to the new layout and update the application. One way is to migrate a copy of the database to the new layout, 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 to users' desktops. Consider a situation where the EJB clients are servlets, but they are not deployed as part of the same EAR file as the EJB, or they are running in a container separate from the EJB. In this case, you might have to set up special EJB bindings so that

the version 1 clients can continue to use the version 1 EJB, while 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 have also enabled 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 might be logged on and have a session on one application server and 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 in-depth information about options for keeping the applications available during an update rollout, see the developerWorks article "Maintain continuous availability while updating WebSphere Application Server enterprise applications" at:

http://www.ibm.com/developerworks/websphere/techjournal/0412_vansickel/
0412_vansickel.html

This article addresses this topic from the development and infrastructure perspective and can provide more detailed information and considerations.

8.12 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 right choice depends on your environment and on the application needs. Table 8-2 compares both options from several perspectives.

Table 8-2   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, while other not so critical applications share the application server or cluster. For application deployment considerations that can help when planning how to accomplish this task, see 5.2.6, "Application deployment" on page 116.

8.13 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).

8.14 Resources

For more information, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *developing and deploying applications*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

For detailed information about application development using Rational Application Developer, see *Rational Application Developer V8 Programming Guide*, SG24-7835.

9

System management

This chapter provides an overview of the planning that is necessary for system management of the 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 proper implementation of the system management processes.

This chapter includes the following sections:

► System management features in WebSphere Application Server V8
► Administrative security
► Administration facilities of WebSphere Application Server
► Automation planning
► Configuration planning
► Change management
► Serviceability
► Planning checklist for system management

271

9.1 System management features in WebSphere Application Server V8

The following list highlights changes in administrative tools and processes in WebSphere Application Server V8:

► Centralized installation manager (CIM)

The CIM, which you can use to install and apply maintenance on remote targets, is now available from the job manager. The CIM for V8 offers the following features:

– Job scheduling
– Removal of cell boundary limitations
– Support for z/OS targets
– Better scaling through the use of the Installation Manager

For more information about these features, see 9.6.3, "Centralized installation manager" on page 288.

► 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 your applications. HPEL does not replace the existing basic log and trace facility. HPEL provides greater flexibility for administrators to manage logging resources while making it easier to work with log and trace content than with 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 with basic logging
– Better administration of resources used to collect and retain logging information
– Enhanced capabilities to work with the logging and trace content

For more information about these topics, see 9.7.1, "Log and traces" on page 291.

► Node management

You can recover or move nodes by using the **-asExistingNode** option with the **addNode** command.

For more information about this option, see "The addNode -asExistingNode command" on page 277.

► Properties file-based configuration

With the portable format of the properties file, available since V7.0.0.9, you can apply property files across multiple environments. Modifying environment-specific variables makes a properties file portable. You can use the **wsadmin** tool to extract a properties file from a cell, modify the environment-specific variables at the bottom of the extracted properties file, and then apply the modified properties file to another cell.

For more information, go to the WebSphere Application Server Version 8 Information Center at the following address, and search for *applying properties files using wsadmin scripting*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

► The `managesdk` command

You can use the `managesdk` command-line tool and associated `wsadmin` commands to manage the software development kits (SDK) that are available to a WebSphere Application Server installation. The `managesdk` command supports the goal of a common API for all WebSphere Application Server platforms.

For more information about this command, see "The managesdk command" on page 278.

► Monitored directory deployment

With monitored directory deployment, you can deploy and update 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 about these topics, see 9.3.8, "Monitored directory deployment" on page 281.

► Job manager

You can complete job manager actions and run jobs from a deployment manager. The administrative console of the deployment manager provides access to the following job manager options:

– Submit a job to the job manager.
– Review the status of a job.
– Identify job manager targets for jobs.
– Identify target resources that are used in jobs.
– Identify target groups for administrative jobs.

For more information about this topic, see 9.3.7, "Job manager" on page 280.

9.2 Administrative security

Consider enabling administrative security to prevent unauthorized access to the administrative tasks. Enabling administrative security secures only administration tasks, not applications.

After administrative security is enabled, a security check is performed when the administrative console or other administrative facilities are accessed. The security check ensures that the accessing user is authenticated and has been mapped to one of the console security roles. Depending on the console role to which the user is mapped, different functions are available.

Proper planning for system management includes identifying the people who will need access and the level of access to the administrative tools. Consider and design a set of groups, users, and roles that fit your needs.

To review the available roles and access levels, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *administrative security*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

WebSphere Application Server offers the option to enable administrative security during profile creation. If you choose this option during profile creation, you are prompted to provide a user ID and password that 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.

9.3 Administration facilities of WebSphere Application Server

WebSphere Application Server V8 provides various administrative tools to configure and manage your runtime environment:

► Integrated Solutions Console

The Integrated Solutions Console (administrative console) is a browser-based client that uses a web application running in the web container to administer WebSphere Application Server.

► WebSphere scripting client (`wsadmin`)

The `wsadmin` client is a non-graphical scripting interface that you can use to administer 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

With Apache Ant, you can create build scripts that compile, package, install, and test your 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 and 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. Where you have few resources, but many tasks, consider using automation and scripts. Where you have multiple administrators that will 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 give you control in handling 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.

9.3.1 Integrated Solutions Console

The Integrated Solutions Console (administrative console) connects to a running stand-alone server or, in a distributed environment, to a deployment manager. In WebSphere Application Server V8, 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://hostname:WC_adminhost/ibm/console/unsecureLogon.jsp
```

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 performing 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/ibm/console/Logon.jsp
```

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.

9.3.2 WebSphere scripting client (wsadmin)

With the WebSphere scripting client (`wsadmin`), you can execute scripts. You can use the `wsadmin` tool to manage a WebSphere Application Server V8 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 and configuration.

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 V8 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 Integrated Solution Console, and if you want, you 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.

> **Jacl support:** 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. Thus, WebSphere Application Server will continue to support 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.

9.3.3 Task automation with Ant

WebSphere Application Server V8 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, that 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 the Apache Ant website at:

http://ant.apache.org

9.3.4 Administrative programming

WebSphere Application Server V8 supports access to the administrative functions through a set of Java classes and methods. You can write a Java application that performs 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, a Java specification part of Java Platform, Enterprise Edition (Java EE), and the specification for the Java EE Management API (JSR-077) are the core of the management architecture for WebSphere Application Server V8. For information about JMX, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *using wsadmin scripting with JMX*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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.

9.3.5 Command-line tools

With command-line tools, you can perform management tasks, including starting, stopping, and checking the status of WebSphere Application Server processes and nodes. These tools only work on local servers and nodes. They cannot operate on a remote server or node. To administer a remote server, you need to use the administrative console or a `wsadmin` script that connects 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 V8 includes the following new 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 9-1. 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.
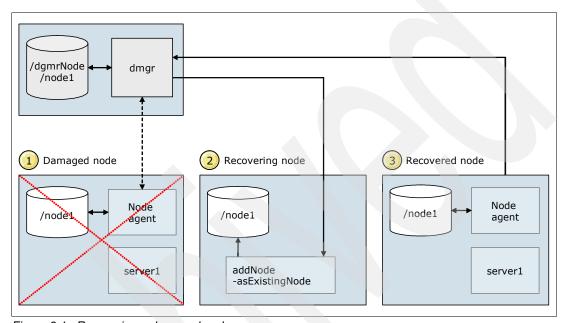
Figure 9-1   Recovering a damaged node

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 9-2.
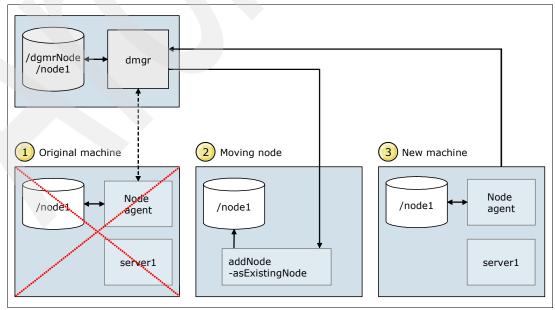
Figure 9-2   Moving a node

You can create a cell from a template cell as shown in Figure 9-3 and configure it as explained in the following steps:

1. Run the **backupConfig** command to create a `template.zip` file of the configurations files.
2. For every new environment, install WebSphere application server.
3. Create deployment manger 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
   ```
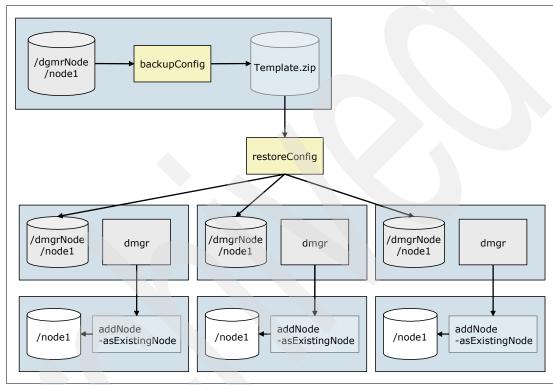

Figure 9-3   Creating cells from a template

For more information about the **addNode -asExistingNode** command, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *addNode -asExistingNode*.

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

The **managesdk** command

WebSphere Application Server V8 introduces the **managesdk** command to manage the SDKs that are available to an installation of WebSphere Application Server. The **managesdk** command supports the goal of 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 that is compatible with the existing WebSphere Application Server infrastructure:

Node default SDK The default SDK for application servers on the node that uses the node level JAVA_HOME variable map.

Server SDK The SDK used by the application server. The default is node default SDK. Each application can override the default by using the server level JAVA_HOME variable map.

> **Multiple SDK support:** In WebSphere Application Server V7 and V8, only IBM i and z/OS platforms support multiple SDKs. The `enablesdk` command used on IBM i platforms is deprecated in V8.

For more information about the `managesdk` command-line tool and associated scripting APIs, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *managesdk*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

9.3.6 Administrative agent

The administrative agent provides a single interface administration 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 that you want the administrative agent to manage by using the `registerNode` command. Also 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 the session if it is lost while performing 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

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 about the administrative agent, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *administrative agent*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

9.3.7 Job manager

In a flexible management environment (Figure 9-4), the job manager allows the management of multiple WebSphere Application Server domains (multiple deployment managers and administrative agents) through a single administration interface.
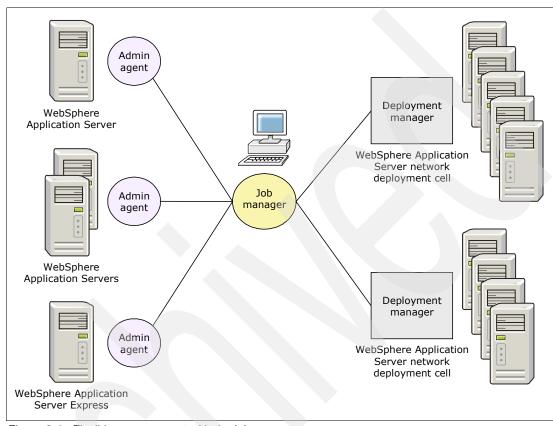
Figure 9-4   Flexible management with the job manager

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.

For WebSphere Application Server V8, you can complete job manager actions and run jobs from a deployment manager. The deployment manager administrative console has a list of job tasks that are available in the navigation tree similar to the jobs manager. The administrative console of the deployment manager provides access to the following job manager options:

► Submit a job to the job manager.
► Review the status of a job.
► Identify job manager targets for jobs.
► Identify target resources used in jobs.
► Identify target groups for administrative jobs.

For more information about the job manager, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *administering nodes remotely using the job manager*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

9.3.8 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` scripting 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 the *profile_root*/*profile_name*/monitoredDeployableApps/servers/*server_name* directory.

► 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 the monitoredDeployableApps/deploymentProperties directory.

The polling interval specifies the number of seconds that the monitored directory is scanned for new applications.

> **Availability of monitored directory:** 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 (SAR) 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.

Adding an application file to a monitored directory: 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 Version 8 Information Center at the following address, and search for *installing enterprise application files by adding them to a monitored directory*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

Adding enterprise application files by adding properties files

You can install, update, or uninstall an EAR, WAR, JAR, or SAR 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.

Properties files differences: 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, you specify `DELETE=true` in the header of the properties section.

For more information, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *installing enterprise application files by adding properties files to a monitored directory*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

9.4 Automation planning

To emphasize the need for automated administration, consider the fact that companies typically have multiple WebSphere Application Server environments to support activities in the different phases of the software development life cycle. Each environment requires the same type of administrative tasks. With automation, a task can be performed manually only one time and then have subsequent requests performed automatically or with less effort.

Automation: Automation is not necessary if the task is not repeated.

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 provides an administrator the opportunity to schedule recurring maintenance and backup procedures and other types of administrative tasks.

You can automate every action that you can perform 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 and use these 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

  – Executes administrative commands interactively or by running a script of commands.

▶ CIM

  – 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.

9.5 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

9.5.1 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.

9.5.2 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 of applications that are configured to start automatically when the server starts. You can access these settings in the administrative console by navigating to **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.

► Launch 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

With this setting, you can specify whether to create Managed Beans for resources such as servlets or JavaServer Pages (JSP) files within an application when the application starts.

The "parallel start" setting for an WebSphere Application Server lets you specify whether to have the server components, services, and applications in an application server start in parallel rather than sequentially. This option can shorten the startup time for a server and can affect how an application server starts. You can access this setting by navigating to **Servers** → **Server Types** → **WebSphere application servers** → *your_server*.

The deployment manager, node agents, and application servers can start in any order they are discovered, with the exception that the node agent 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 later manually. To prevent an application from starting when a server starts, navigate to **Applications** → **Application Types** → **WebSphere enterprise applications** → *application_name* → **Target specific application status**. Then, disable auto start for the application.

9.5.3 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, the server configuration must be exported to a configuration archive, after which it can be imported to another cell.

You might need more than one application server (for example, for a cluster) 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. Subsequent servers in the cluster are also created by using this template, reducing the scope for error and making the task of creating the server cluster much faster.

9.5.4 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, rather than creating the data source multiple times, possibly introducing errors along the way. Also, if the data source definition needs to change (for example, due to changes to an underlying database), you can change the data source definition one time. It is visible to all servers within the node, saving both time and cost.

Consider outlining the resources that you will 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 that are 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 that is 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 and 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, 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, 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 given 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.

Using variables with scopes: 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 proper 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 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. However, 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 the test to work.

For more information, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *test connection service*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

9.6 Change management

Proper 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 operational environment:

► Application update
► Changes in topology
► Centralized installation manager

9.6.1 Application update

WebSphere Application Server V8 permits fine-grained updates to applications. It allows application components to be supplied and the restart of 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. That is, 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 given 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, until the entire cluster has received 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 pause the listeners for an application server, perform the application update, and then resume the listeners. If you use this technique, you do not have to stop and then start the server to perform the application update.

9.6.2 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. Make 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 overridden 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 that 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, consider defining the file synchronization strategy as part of the change management process.

9.6.3 Centralized installation manager

The CIM is used to manage V8 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, as shown in Table 9-1.

Table 9-1   CIM functional differences between product versions

Function	CIM V6 and V7 (all releases)	CIM V8
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	CIM `AdminTask` commands.	The job manager `submitJob` command.

CIM for V8

CIM for V8, which you can use to install and apply maintenance on remote targets, is integrated into the job manager. With CIM for V8, 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 Customization Toolkit
- ► Web server plug-ins

The CIM functions are accessed through the job manager (or deployment manager). Because the CIM functions are implemented as jobs, the CIM supports job scheduling. With CIM jobs, you can perform the following tasks:

- ► Perform an inventory.
- ► Install, update, and uninstall Installation Manager (new for V8).
- ► Manage offerings (install, update, and uninstall WebSphere Application Server).
- ► Manage profiles (new for V8).
- ► Run command (new for V8).
- ► Install SSH public key.
- ► Distribute, collect, and delete files (new for V8).
- ► Test connection.
- ► Add or search Installation Manager agent data locations.

CIM on z/OS environments has some restrictions. For a complete list of available CIM tasks on z/OS targets, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *submitting Installation Manager jobs*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

As shown in Figure 9-5, the CIM 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.
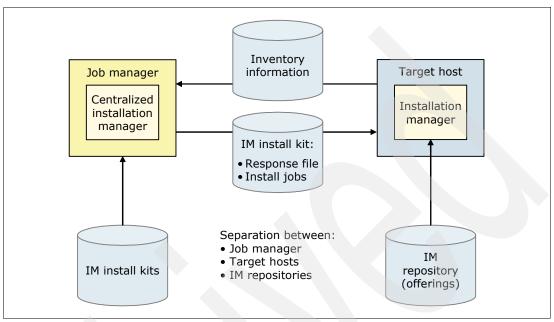
Figure 9-5   Centralized installation manager for WebSphere Application Server V8

The CIM for V8 is supported on 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 CIM for V8, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *installing in group mode*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

CIM for V6.1 (all releases) and V7 (all releases)

The CIM 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 Version 8 Information Center at the following address, and search for *using the centralized installation manager (CIM) to manage Version 6.1.x and 7.x*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

9.7 Serviceability

A major challenge of problem management is dealing with unanticipated issues. Much like detective work, you need to find clues, make educated guesses, verify suspicions, and so on. 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 proper 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, 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 has enough space 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.

9.7.1 Log and traces

WebSphere Application Server V8 includes the following modes of logging:

► High Performance Extensible Logging mode
► Basic mode

High Performance Extensible Logging mode

Beginning in WebSphere Application Server Version 8, you can configure the server to use the HPEL log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. By default, HPEL is not enabled. You can enable it from the administrative console or by using `wsadmin` scripting.

> **Enabling HPEL:** For instructions about enabling HPEL by using the administrative console or `wsadmin` scripting, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *basic mode and HPEL mode*:
>
> http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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 9-6:

► 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 most often useful to administrators.

► 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.
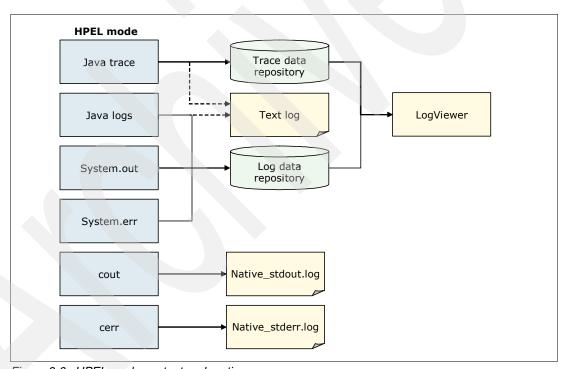
Figure 9-6   HPEL mode content and routing

Log and trace performance is greatly enhanced with HPEL because of the following factors:

► Log and trace events are stored only in one place.

Log and trace events are stored only in one 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 repositories 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.

> **Reading log and trace data:** Log and trace data stored in the repositories cannot be read by using text file editors. To view log and trace data, you can enable the text log file or copy the log and trace data into a plain text format by using 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. To learn more about the configurable parameters, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *using HPEL to troubleshoot applications*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

Administrators can easily configure the resources that are required to store the log and trace data. The administrative console or `wsadmin` scripts is 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 the `LogViewer` command or by using the administrative console as shown in Figure 9-7.
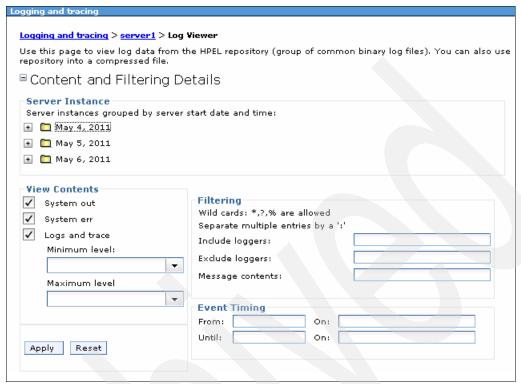
Figure 9-7   Filtering log and trace data using HPEL

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 9-8:

► 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)
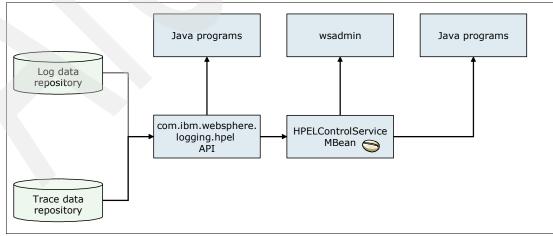
Figure 9-8   HPEL development resources

Basic mode

Basic mode is the log and trace functionality 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 9-9 on page 296:

► JVM logs

The JVM logs are written as plain text files, named `SystemOut.log` and `SystemErr.log`, and are in the *profile_home*/logs/*server_name* directory.

You can view the JVM logs from the administrative console (including logs for remote systems) or by using 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 running 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 (SO), 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 editing it can corrupt the log.

You can view the service log by using the following methods:

– Log Analyzer tool

Use this tool to view the service log. This tool provides interactive viewing and analysis capability that is 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 that you can then write to a file or to the command shell window.

The IBM service log is in the *profile_home*/logs/ directory.
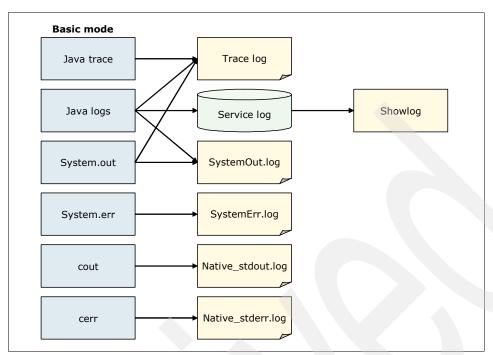
Figure 9-9   Basic mode content and routing

You need to name log files and traces 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.)

> **Log files and traces on z/OS targets:** On z/OS targets, the log files are in the job logs of the application server.

9.7.2 Fix management

Applying regular fixes is a key factor to 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, the best approach is to have a strong fix plan that outlines regular fix updates and reasonable retesting before each fix.

For available fixes, see the WebSphere Application Server support page at:

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

9.7.3 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.

9.7.4 MustGather documents

MustGather documents provide instructions on how to start troubleshooting a problem and the 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 an example of such documents, see the technote at:

`http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=troubleshooting&ui`
`d=swg21201625&loc=en_US&cs=utf-8&lang=en`

On the IBM Support site, many MustGather documents are categorized as troubleshooting and analyzing data. You check the troubleshooting document 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 have just 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.

9.7.5 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 map. 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.

For more information about IBM Support Assistant, see the IBM Support Site website at:

`http://www.ibm.com/software/support/isa`

9.7.6 WebSphere Application Server Information Center

For troubleshooting information, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *troubleshooting and support*:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp`

9.8 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 to 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 good entry point to system management topics, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *administering applications and their environment*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

Messaging and service integration

This chapter provides information about planning for a WebSphere Application Server V8 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 features in WebSphere Application Server V8
- ► Messaging options
- ► Messaging topologies
- ► Security and reliability of messaging features
- ► Planning checklist for messaging
- ► Resources

This chapter briefly describes the concepts that are required to understand messaging. For a good entry point to messaging topics, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *messaging resources*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

10.1 Messaging overview

Generically, the term *messaging* describes communication, or the exchange of information, between two or more interested parties. Messaging can take many shapes and 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 that might reside 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 does 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 recognizes the power of messaging. It implements a powerful and flexible messaging platform within the WebSphere Application Server environment, called the *service integration bus* (SIBus).

10.2 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 application. 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. The section provides information about the main components of service integration.

10.2.1 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 that cooperate to provide asynchronous messaging services. Usually, 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* to which one application sends and from which the other application receives.

► A message-producing application, that is 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, that is a *consumer*, can consume messages from a destination (whenever that destination is available) regardless of which messaging engine the consumer uses to connect to the bus.

To configure a service integration bus, you can use an administrative console. In the navigation pane, expand **Service Integration** → **Buses**.

10.2.2 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, which are the application server components that 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 can use an administrative console. In the navigation pane, expand **Service Integration** → **Buses** → *bus name*, and select **Bus members** as shown in Figure 10-1.
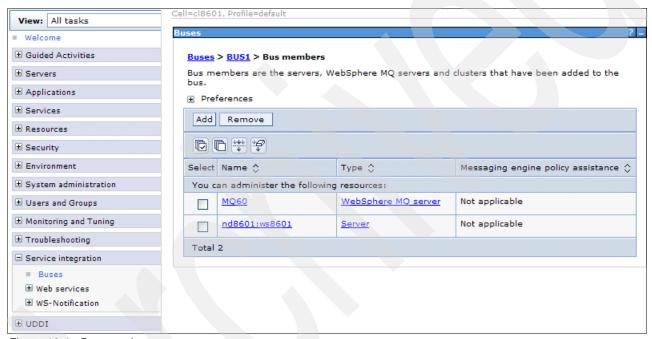
Figure 10-1   Bus members

To use WebSphere MQ as a bus member, you must first define WebSphere MQ as a WebSphere MQ server. Using the administrative console, in the navigation pane, expand **Server** → **Server Types**, and select **WebSphere MQ server**.

10.2.3 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* where, if necessary, it can hold messages until consuming applications are ready to receive them. Alternatively, it can preserve messages if the messaging engine fails. If the bus member is a WebSphere MQ server, it does not have a messaging engine.

Two types of message stores are available:

► A *file store* directly uses files in a file system through the operating system to preserve operating information and to persist the objects that messaging engines need to recover in 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 that the messaging engine needs for recovery 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.

Creation of a messaging engine: A messaging engine is created automatically when you add an application server or server cluster as a bus member.

For information about the type of message store to use, see 10.6.3, "Planning for reliability" on page 325.

Figure 10-2 illustrates the components of the SIBus.
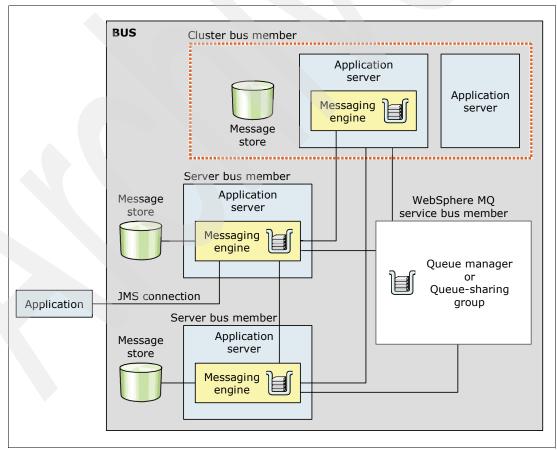
Figure 10-2   Main components of a service integration bus

10.2.4 Messaging provider

WebSphere Application Server applications invoke asynchronous messaging services by using the Java Messaging Service (JMS) application programming interface (API) to interface with a messaging provider. WebSphere Application Server includes 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 10.4, "Messaging options" on page 308.

10.2.5 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* only exists 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 points   The message point for a queue destination. When creating a queue destination on a bus, an administrator specifies the bus member that will hold 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. All of the 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 points 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, depending 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 WebSphere Application Server default messaging provider 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, or components that use messaging. Mediations are used to process in-flight messages. Mediation can undertake 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 two 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, using a 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 10-3 illustrates the mediation process flow.
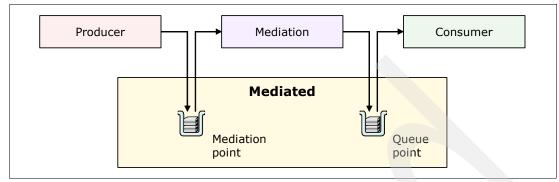
Figure 10-3   Mediation process flow

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 10-4.
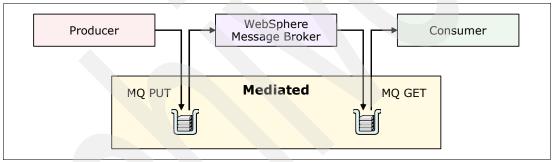
Figure 10-4   Mediation using WebSphere MQ

Transport chains

The term *transport chain* describes the process and mechanism that a messaging engine uses to communicate with another messaging engine, external messaging provider, or messaging application running 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).

10.3 Messaging and service integration features in WebSphere Application Server V8

WebSphere Application Server V8 introduces the following improvements to the service integration bus:

► Support for connecting to multi-instance WebSphere MQ

  WebSphere Application Server Version 8 provides first-class 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, which a connection factory or activation specification uses to connect to multi-instance queue managers.

To define multi-instance WebSphere MQ, follow 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 10-5), select the **Enter host and port information in the form of a connection name list** option. Configure the multi-instance WebSphere MQ in the format host[(port)] [,host(port)].
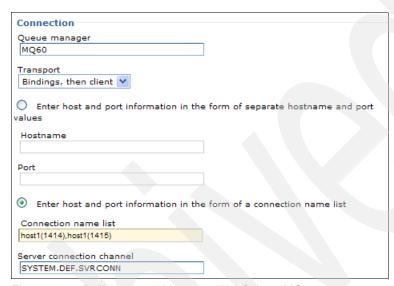
Figure 10-5   Defining a multi-instance WebSphere MQ

► New WebSphere MQ destinations properties

With client reconnection properties for connection factories, you can specify whether a client node connection should reconnect automatically 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 Version 8 Information Center at the following address, and search for *WebSphere MQ messaging provider connection factory advanced properties*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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 that have been 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 Version 8 Information Center (referenced previously), and search for WebSphere MQ messaging provider queue and topic advanced properties settings.

► Disabling WebSphere MQ functionality

When a WebSphere Application Server process or an application process starts, and while it is running, processing is performed to allow it to support WebSphere MQ-related functionality, such as the WebSphere MQ messaging provider. By default, this processing is performed regardless of whether WebSphere MQ-related functionality is used. If you do not need to take advantage of any WebSphere MQ functionality, you can disable all WebSphere MQ functionality in an application server or client process for increased performance. To disable WebSphere MQ functionality, select **Disable WebSphere MQ** (Figure 10-6).
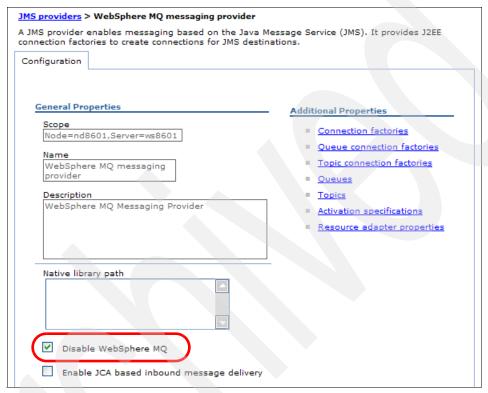
Figure 10-6   Disabling WebSphere MQ functionality

► New 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 10-7 on page 308. The following properties affect the connection pool that is used by 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 again to reconnect to a WebSphere MQ queue manager.
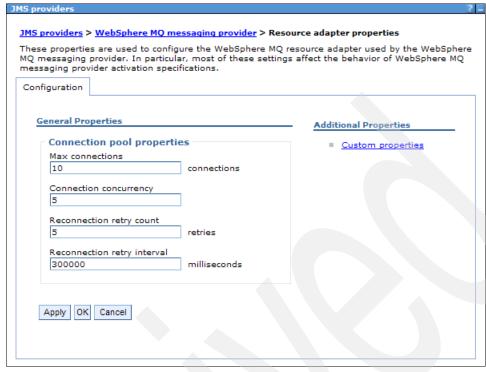
Figure 10-7   New WebSphere MQ connection properties

WebSphere Application Server V8 has several other improvements and new additions to messaging. For a full list, see WebSphere Application Server Version 8 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/v8r0/index.jsp

10.4 Messaging options

This section addresses, at a high level, how messaging is implemented within WebSphere Application Server and the considerations that you need to make. This information helps you in the following instances:

► Messaging provider standards
► Styles of messaging in applications
► Types of messaging providers

   – Default messaging provider
   – WebSphere MQ messaging provider
   – Third-party messaging provider (generic JMS)

Deprecated feature: 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 cell in later versions. For more information, see WebSphere Application Server Version 8 Information Center at the following address, and search for *maintaining (deprecated) Version 5 default messaging resources*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

- ► Example of JMS interfaces: Explicit polling for messages
- ► Example of a message-driven bean: Automatic message retrieval

10.4.1 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 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, while specific to messaging, has been expanded to support many message types and architectures, providing flexibility and versatility in the vast majority of applications.

> **Important:** For the rest of the chapter, JMS is the chosen method to access the messaging middleware provider.

Vendor-specific client libraries

As the name suggests, 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 non-portable across platforms.

- ► Support might not be available for certain languages such as Java, and these libraries have no direct support in WebSphere Application Server.

Consider not 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.

10.4.2 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 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, which 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. Then, it routes the published messages to subscribers. A subscriber receives messages from only those topics to which it has 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 referred to as 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.

10.4.3 Default messaging provider

The fully featured 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 and to 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 that applications interact with and that an administrator configures as a JMS resource of the default messaging provider

Java EE applications (producers and consumers) access the SIBus and the bus members through the JMS API. JMS destinations are associated with SIBus destinations. A SIBus 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 10-8.
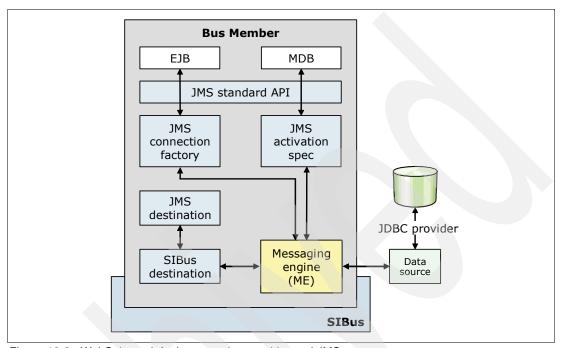
Figure 10-8   WebSphere default messaging provider and JMS

10.4.4 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. However, using another provider can have benefits. If you are unsure about which provider combination is suited to your requirements, see WebSphere Application Server Version 8 Information Center at the following address, and search for *choosing messaging providers for a mixed environment*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

You can use WebSphere Application Server to configure WebSphere MQ resources for applications (for example queue connection factories) and to manage messages and subscriptions associated with JMS destinations. You administer security through WebSphere MQ. You also can use WebSphere Application Server to coordinate global transactions including WebSphere MQ without configuring the extended transaction client.

> **Disabling WebSphere MQ functionality:** If you do not need to take advantage of any WebSphere MQ functionality, you can disable all WebSphere MQ functionality in an application server or client process to give increased performance. For more information, see "Disabling WebSphere MQ functionality" on page 307.

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 show the relative advantages and disadvantages of each approach.

WebSphere MQ as an external messaging provider

The WebSphere MQ messaging provider does not use service integration. It provides JMS messaging access WebSphere MQ from WebSphere Application Server. The WebSphere MQ messaging provider makes point-to-point messaging and publish/subscribe messaging available to WebSphere Application Server applications by using the existing capabilities in the WebSphere MQ environment. Table 10-1 the advantages and disadvantages of using WebSphere MQ as an external messaging provider.

Table 10-1   Advantages and disadvantages of WebSphere MQ as an external messaging provider

Advantages	Disadvantages
► 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 rather than two. ► 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 10-9 shows a JMS application sending messages through WebSphere Application Server to WebSphere MQ using WebSphere MQ as an external messaging provider
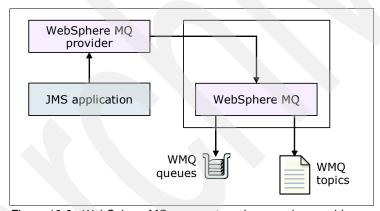
Figure 10-9   WebSphere MQ as an external messaging provider

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), which acts as the gateway to the WebSphere MQ network. When you use a WebSphere MQ link, the WebSphere MQ network views the message bus as a virtual queue manager, and 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 (defined as a destination in the service integration bus). Also, WebSphere MQ applications can send point-to-point messages to destinations in the service integration bus (defined as remote queues in WebSphere MQ).

You can also set up a publish/subscribe bridge. This way, 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 10-2 lists the advantages and disadvantages of using the WebSphere MQ network as a foreign bus.

Table 10-2   WebSphere MQ network as a foreign bus - Advantages and Disadvantages

Advantages	Disadvantages
► A WebSphere MQ client facility is not required on the gateway WebSphere MQ queue manager. ► Each end of the link appears in natural form to the other. WebSphere MQ appears to service integration as a (foreign) bus. Service integration appears to WebSphere MQ as a (virtual) queue manager. ► Better 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. ► Good 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 less easy to achieve because messages have to be "pushed" from either end of the link. ► You cannot use mediations for modifying messages, routing, or logging.

Figure 10-10 shows a JMS application sending messages through WebSphere Application Server to WebSphere MQ using WebSphere MQ network as a foreign bus.
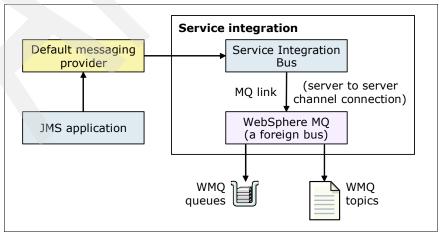
Figure 10-10   WebSphere MQ network as a foreign bus

A 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 only represents queues for point-to-point messaging and ensures that messages are converted between the formats used by WebSphere Application Server and those formats used by WebSphere MQ. Table 10-3 lists the advantages and disadvantages of using a WebSphere MQ server as a bus member.

Table 10-3   Advantages and disadvantages of WebSphere MQ server as a bus member

Advantages	Disadvantages
► WebSphere Application Server and WebSphere MQ can exist on separate hosts. ► Each end of the connection appears in natural form to the other. WebSphere MQ queue manager appears to service integration as a foreign bus. Service integration appears 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, depending on the number of nodes and your version of WebSphere MQ for z/OS, you might require the Client Attachment feature (CAF) on z/OS. ► You must explicitly define all destinations.

Figure 10-11 shows a JMS application sending messages through WebSphere Application Server to WebSphere MQ using WebSphere MQ (a queue manager or queue-sharing group) as a bus member.
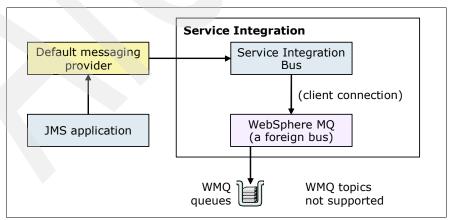
Figure 10-11   WebSphere MQ server as a bus member

10.4.5 Third-party messaging provider (generic JMS)

The third-party messaging provider is the catch-all for any external messaging providers other than WebSphere MQ. Although WebSphere Application Server works with any JMS-compliant messaging provider (after it is defined to WebSphere), there can only be limited administrative support in WebSphere.

Consider using a third-party messaging provider only if you have an existing investment in a third-party messaging provider. Keep in mind that much greater support is available in the WebSphere Application Server default messaging provider and WebSphere MQ messaging provider.

10.4.6 Example of JMS interfaces: Explicit polling for messages

Applications can use JMS to explicitly poll for messages on a destination and then retrieve messages for processing by business logic beans (enterprise beans). Figure 10-12 shows an enterprise application pooling a JMS destination to retrieve an incoming message, which it processes 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.
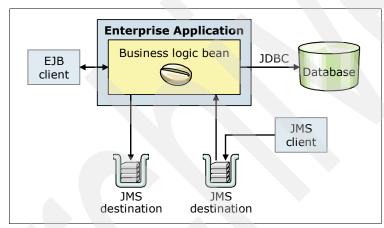
Figure 10-12   Using JMS as asynchronous messaging

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 acting as a sender bean, or it can be handled in the enterprise bean that receives the incoming messages. Optionally, this process can use two-phase commit within the scope of a transaction.

10.4.7 Example of a message-driven bean: Automatic message retrieval

WebSphere Application Server supports the use of MDBs as asynchronous message consumers. Figure 10-13 shows an incoming message being 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.
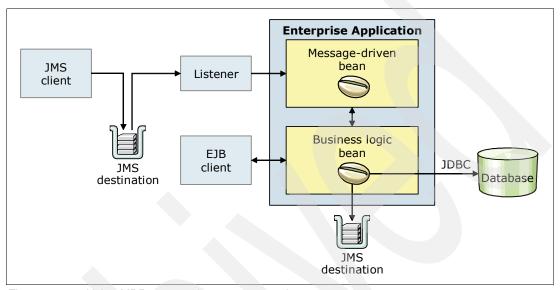
Figure 10-13   Using MDB as asynchronous messaging

MDBs can be configured as listener on Java EE Connector Architecture (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, making MDBs suitable for handling generic requests inbound to WebSphere Application Server from enterprise information system through the resource adapter. In the JCA 1.5 specification, such MDBs are commonly called *message endpoints* or just *endpoints*.

10.5 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 which topology might best fit your needs (depending on the answers to the previous questions). In most cases, the topology might not be clear cut because you can implement a messaging application in many different ways. However, the simpler approach you choose, the better the results will be.

> **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 some of the ones implemented by the default messaging provider of WebSphere Application Server (in increasing complexity) using the previously defined concepts.

10.5.1 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 application server need to communicate among themselves. Additional application servers that are not members of the bus and only need to use bus resources infrequently can connect remotely to the messaging engine (Figure 10-14).
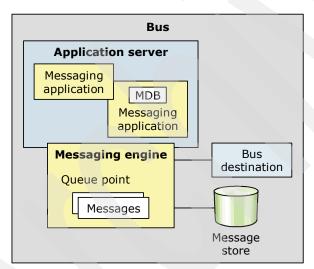
Figure 10-14   Single bus with an application server member

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 the messaging engine and, therefore, can connect to and use the bus, in any of the following situations:

► The application runs in another server in the same cell or in the same server, in a server in a different cell, or in a client container.

► The application uses a client connection to use the bus or in-process call.

Figure 10-15 shows the possible connections:
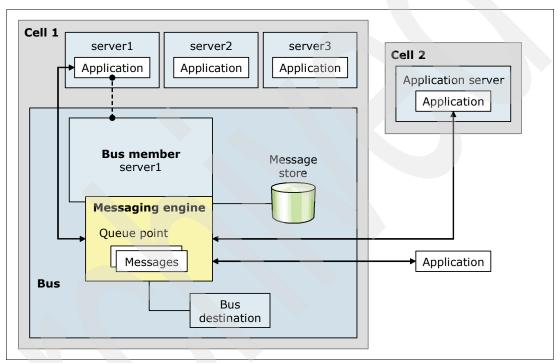
Figure 10-15   Applications connecting to a messaging engine

10.5.2 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, but 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 only consume from a local messaging engine (Figure 10-16). Because everything is tunneled through one messaging engine, performance might still be an issue.
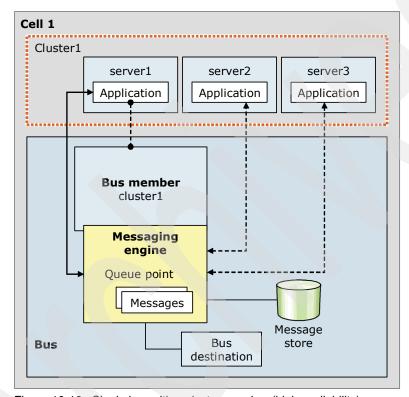
Figure 10-16   Single bus with a cluster member (high availability)

Consider the concept of preferred servers with clustering, for example, a primary server and a backup server in the same cluster. However, this approach must be explicitly configured. It is possible to set up this approach so 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, thus providing workload management and failover (Figure 10-17). Because messaging engines can run on any server, if one server goes down, both messaging engines will run on the remaining server.
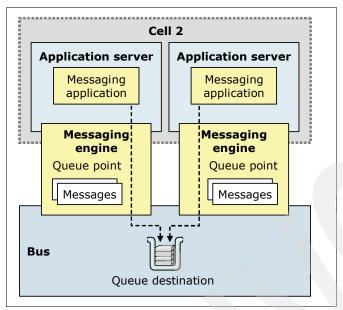
Figure 10-17   Single bus with a cluster member, providing workload management

When a queue destination is assigned to the cluster, the queue is partitioned. Each messaging engine in the cluster owns a partition of the queue. A message sent to the queue is assigned to one partition. 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 running on any of the servers in the cluster.

10.5.3 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 10-18). In this topology, most, if not all servers are bus members. Use care in locating the queue points on the same application server as the messaging application that is the primary user of the queue. This approach maximizes the use of local connections and enhances performance.
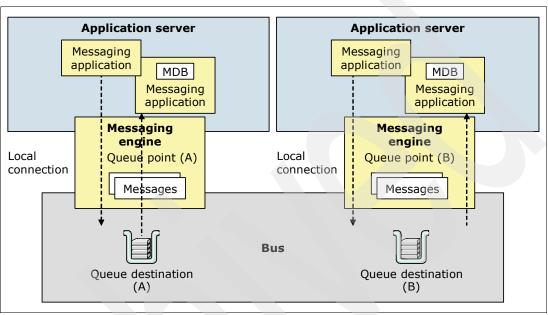
Figure 10-18   Single bus with multiple application server members

10.5.4 Multiple buses

Many scenarios only require relatively simple bus topologies, even just a single server. When integrating applications that have been 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 service integration bus cannot expand beyond the edge of a WebSphere Application Server 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 service integration 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 service integration bus to the WebSphere MQ network, where it appears to be another queue manager. This approach is achieved through the user of an MQ link.

Figure 10-19 illustrates how a service integration bus can be connected to another service integration bus and to a WebSphere MQ network.
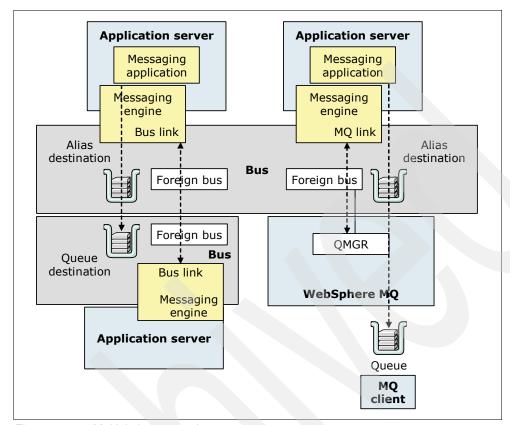
Figure 10-19   Multiple bus scenario

For the connection between the two service integration buses, each messaging engine contains a service integration bus link configuration that defines the location of the messaging engine on the remote bus. For the WebSphere MQ connection, the messaging engine contains an MQ link configuration that defines the queue manager on WebSphere MQ and identifies a queue manager name that it will be known by from the view of 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 that 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.

10.5.5 Connecting to WebSphere MQ on z/OS

A second option for connecting to WebSphere MQ is to create a WebSphere MQ server definition that represents a queue manager or queue sharing group on WebSphere MQ running on z/OS (Figure 10-20). The WebSphere MQ server defines properties for the connection to the queue manager or queue sharing group. With WebSphere Application Server V8, this construct can also be applied to distributed (non z/OS platforms) queue managers.
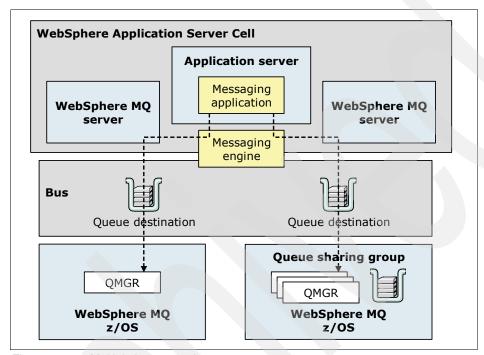
Figure 10-20   Multiple bus scenario

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 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 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 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 same 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 an MQ queue. Similarly, any bus mediations take place immediately upon a message appearing in an MQ queue.

► With WebSphere MQ link, applications have to 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.

10.6 Security and reliability of messaging features

This section provides information about some of the lower-level details and requirements of messaging. It addresses security, high availability, and reliability, which are important points that must be factored into any planning.

This section includes the following topics:

► Planning for security
► Planning for high availability
► Planning for reliability

10.6.1 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 will be denied access even if the credentials are valid.

The following roles also affect permissions for users and groups:

► Sender. User/group can send (produce) messages to the destination.
► Receiver. User/group can read (consume) messages from the destination.
► Browser. 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 message engine uses a particular transport chain to connect to a bus and communicate a message to another messaging engine. The transport chains have attributes such as security encryption (for example, using SSL or HTTPS) and the communication protocol used (for example, 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, you choose the particular transport chains that have the attributes you need.

You need to 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?

10.6.2 Planning for high availability

An application server only has 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 hosting 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, you must 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, or you want the messaging engine to only run on a small group of servers within the cluster.

Each messaging engine on a service integration bus belongs to one high availability group. A policy assigned to the group at runtime 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 restricted.

To obtain workload management across a bus with a cluster, you need to create additional messaging engines and assign the messaging engines to a preferred server. The messaging engines run simultaneously with queues partitioned across them. You might need to consider clustering some application server if you have not already.

10.6.3 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 by using that connection factory, although you can choose for the

reliability settings be set individually at the bus destination. Each reliability setting has different performance characteristics. The settings are as follows:

► Best effort non-persistent
► Express non-persistent
► Reliable non-persistent
► Reliable persistent
► Assured persistent

You must consider the trade-off between reliability and performance. With increasing reliability levels of a given destination, performance or throughput of that destination is decreased. 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 Version 8 Information Center at the following address, and search for the following topics:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

► *JMS delivery mode and service integration quality of service*

  Includes a table that outlines what happens to a message under various circumstances depending 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.

10.7 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).

10.8 Resources

For a good overall reference, see WebSphere Application Server Version 8 Information Center at the following address, and search for *messaging resources*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

11

Web services

This chapter addresses traditional web services and RESTful web services. It highlights the considerations that administrators must make when planning for the use of these services on a WebSphere Application Server V8 architecture.

This chapter includes the following sections:

► Overview of web services
► Web services features for WebSphere Application Server V8
► 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

329

11.1 Overview of web services

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 consumed by existing applications or new ones. As the architecture grows and more applications are added to this portfolio, applications can be orchestrated in flexible business workflows. As a result, businesses can better react to changes, such as the introduction of a new partner or supplier, shifts in the business model, or the streamlining of several application services into one.

Web services provide a standard implementation for SOA, similar to the support that WebSphere Application Server V8 provides. 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

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. More specifically, a web service can be an application or function that can be programmatically invoked over the Internet. For example, buyers and sellers all over the world can discover each other, connect dynamically, and execute 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 invoked 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 perform 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 machines, 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 and loosely coupled. Web services are not tightly coupled and are easily reconfigured into new services. Therefore, web services must be able to be dynamically discovered in an automated fashion, allowing for 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.

11.2 Web services features for WebSphere Application Server V8

This section provides information about the changes to WebSphere Application Server V8 regarding web services. Major changes to web services support were introduced in the previously existing feature pack for web Services for WebSphere Application Server V6.1. The feature pack added support for the following features (also present in WebSphere Application Server V8):

► Web services standards

   – Web Services Reliable Messaging (WS-RM) 1.1
   – Web Services Addressing (WS-Addressing) 1.0
   – Web Services Secure Conversation (WS-SC) 1.0
   – SOAP 1.2
   – SOAP Message Transmission Optimization Mechanism (MTOM) 1.0

► Standards-based programming models

   – Java API for XML Web Services (JAX-WS) 2.0
   – Java Architecture for XML Binding (JAXB) 2.0
   – SOAP with Attachments API for Java (SAAJ) 1.3
   – Streaming API for XML (StAX) 1.0

WebSphere Application Server V7 introduced the following major enhancements (also present in WebSphere Application Server V8):

► Support for World Wide Web Consortium (W3C) SOAP over Java Message Service (JMS) 1.0 for JAX-WS

► Support for WS-Policy 1.5 and related specifications such as WS-SecurityPolicy 1.2

► WS-Addressing now based on JAX-WS

► Adoption of Organization for the Advancement of Structured Information Standards (OASIS) WS-SecureConversation 1.2

► Exposure of Enterprise JavaBeans (EJB) 3.0 components as JAX-WS web services

► Enhanced Configuration Archive support for web services-specific metadata (policy sets, policy types, and bindings)

In addition to the enhancements from the feature pack and WebSphere Application Server V7, WebSphere Application Server V8 includes the following web services enhancements:

► Support for Web Services for Java Platform, Enterprise Edition (Java EE) 1.3 (JSR-109) specification

► Support for JAX-WS 2.2

► Support for Java API for RESTful Web Services (JAX-RS) 1.1

► Support for JAXB 2.2 and JAXB 2.2 Reference Implementation Vendor Extensions

► Improvements to the Integrated Solutions Console with enhanced support for policy sets, application bindings, and endpoints management

11.3 Considerations when using web services

This section addresses the business and technical issues that you need to consider when deciding to use web services. The questions listed here represent the strategic thinking that needs to happen if you want to provide or use web services.

11.3.1 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 the 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 to existing exposed function?

You can use web services as an easier way to expose application programing 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 largely depends 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 transparently to 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?

11.3.2 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 might use the HTTP communications protocol. HTTP is a stateless protocol with no guarantees regarding 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 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?

Although the basic mechanisms underlying web services have been around for some time, other 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 IBM developerWorks article "Standards and web services" at:

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 and 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.

11.4 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.

11.4.1 Components of the architecture

The basic SOA consists of the following primary components:

► The *service provider* creates a service, possibly publishes its interface, and accesses 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 and then binds to the service provider to invoke 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.

Each component can act as one of the two other components. For example, if a service provider needs more information that it can only acquire from some other service, it acts as a service requestor while still serving the original request.

Figure 11-1 shows the operations that each SOA component can perform.
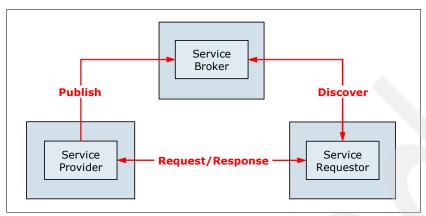
Figure 11-1   SOA components and operations

Before looking at the architecture from a web services-specific view, review the brief explanation of 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 XML-based interface 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 SOAP-based server implementation. It can be used to store and retrieve information about service providers and web services.

▶ *Web Services Inspection Language (WSIL)* is an XML-based specification about how to locate web services without using UDDI. However, WSIL can be also used with UDDI and does not necessarily replace it.

Figure 11-2 shows a lower-level view of an SOA with specific components and technologies. The UDDI and WSIL, separately or together, become the service broker.
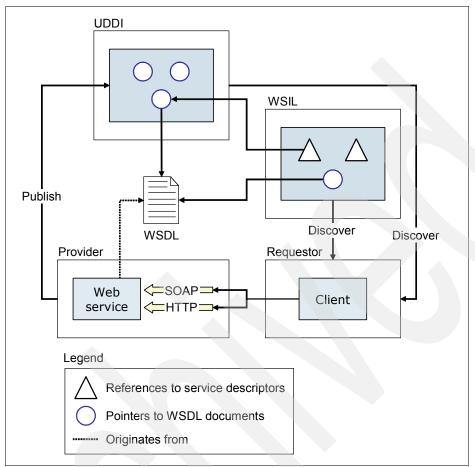
Figure 11-2   Main building blocks in an SOA approach based on web services

11.4.2 How to use this architecture

This section highlights the common message exchange patterns (often referred to as *interaction patterns*) that you might employ. These patterns use the web services architecture that was explained in the previous section. However, some of these patterns might have a bearing on the type of transport that you use and whether you even need to use a web service.

This section addresses other options of which an administrator should be aware, such as 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 (Figure 11-3). Whether the destination accepts the message (with or without error conditions) is not important to the source. The service provider (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 (the source), the new value is sent to a monitoring application (the destination).

Figure 11-3   One-way messaging exchange pattern

Asynchronous two-way message exchange pattern

In an asynchronous two-way message exchange pattern (Figure 11-4), the service requestor expects a response, but the messages are asynchronous in nature (for example, where 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 has to relate a message that it receives to the corresponding message that was sent by the Service 1 Consumer component.

Technically, this message exchange pattern is the same as the one-way pattern, with the additional requirement that there must be a mechanism to associate response messages with their corresponding request message. This mechanism can occur at the application level or by using the SOAP protocol.
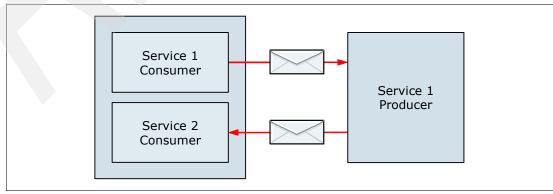
Figure 11-4   Asynchronous two-way messaging pattern

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 11-5). In this message exchange pattern, the underlying transport protocol provides an implicit association between the request message and the response message.

In situations where 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.
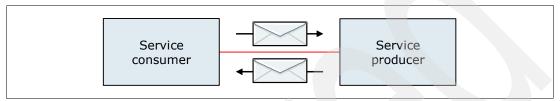
Figure 11-5   Request-response message exchange pattern

Numerous examples of this message exchange pattern are available, such as 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 (Figure 11-6). Each web service plays a specific role in the workflow.
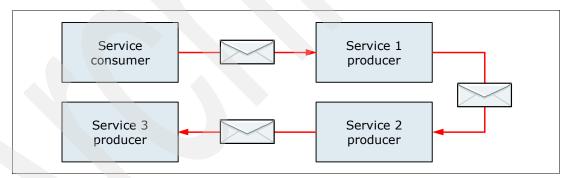
Figure 11-6   Workflow-oriented message exchange pattern

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, also known as the *event-based* or *notification-based* pattern, is generally used in situations where information is pushed out to one or more parties (Figure 11-7 on page 338).

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.
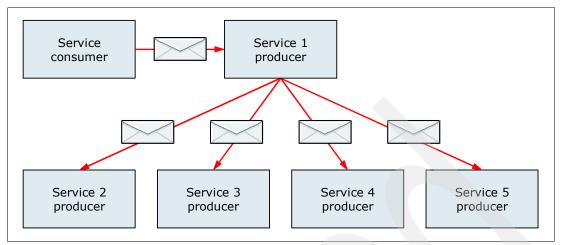
Figure 11-7   Publish-subscribe message exchange pattern

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 11-8).

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.
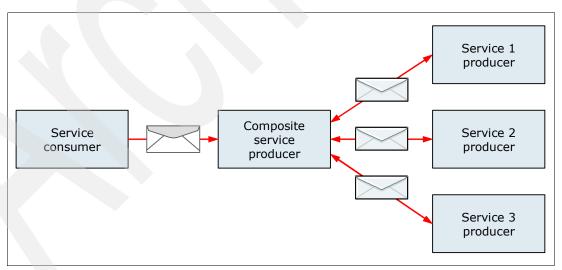
Figure 11-8   Composite message exchange pattern

An example of a composite message exchange pattern is an online ordering system, where the service consumer represents a business partner application placing 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 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 11-9).
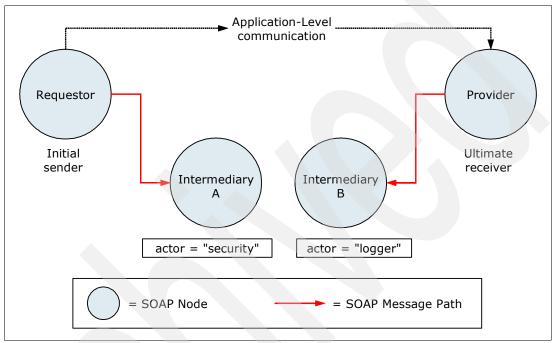
Figure 11-9   SOAP processing model

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 should not alter the meaning of the message body.

A typical situation where you need to use intermediary SOAP nodes is where you have an existing internal web service implementation within your enterprise that you now want to expose externally. New requirements might be associated with requests originating from outside of your organization, such as 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

11.5 Support for web services in WebSphere Application Server

WebSphere Application Server V8 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 J2EE environment, more specifically in the web, EJB, and client application containers.

11.5.1 Supported standards

Web services support in WebSphere Application Server V8 includes the following standards and specifications:

JAX-WS The core programming model and bindings for developing and deploying web services on the Java platform.

WS Transaction support
Defines how web services applications can work within global transactions in enterprise environments by using the following specifications:

WS-Atomic Transaction (WS-AT)
A specific coordination type that defines protocols for atomic transactions.

WS-Business Activity (WS-BA)
A specific coordination type that defines protocols for business activities. A business activity is a group of general tasks that you want to link together so that the tasks have an agreed outcome.

WS-Coordination (WS-Coor)
Specifies a context and a registration service with which participant web services can enlist to take part in the protocols that are offered by specific coordination types.

WS-I Basic Profile A set of non-proprietary web services specifications that promote interoperability.

WS-Notification Publish and subscribe messaging for web services.

WS-Addressing Enables systems to support message transmission and identification through networks that include firewalls or gateways in a transport-neutral manner.

WS-Security Covers a standard set of SOAP extensions that can be used when building secure web services to provide integrity and confidentiality. It is open to other security models including PKI, Kerberos, and SSL. WS-Security provides support for multiple security tokens, multiple signature formats, multiple trust domains, and multiple encryption technologies. It includes security token propagation, message integrity, and message confidentiality.

JAX-RS	Enables the development of services that follow Representational State Transfer (REST) principles.
WS-Policy	Describes and communicates web service policies. It allows service providers to export policy requirements in a standard format. Clients can combine their capabilities with service provider requirements to establish policies that are required for a specific interaction.

For a complete list of supported standards and specifications, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *specifications and API documentation*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

11.5.2 Service integration bus

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

Using an SIBus 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. With this exposure, 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 SIBus 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 and, thus, 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, such as clustering for high availability, is an option for use by web services.

For more information about SIBus, see Chapter 10, "Messaging and service integration" on page 299.

11.5.3 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 that are provided for interacting with a business for such purposes as 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 (B2B) interaction. For the manufacturer who needs to create many relationships with different customers, each with its own set of standards and protocols, 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 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. First, companies often do not want to show all their interfaces to the entire world, which 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. Given that the users of the registry are often automated processes and not humans with the intuitive ability to separate good and bad content, 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 that controls access to that registry to parties who are previously screened.

Private registries provides control over who is allowed to explore the registry, who is allowed to publish to the registry, and standards governing exactly what information is published. Given the cleanliness of the data in a private registry (compared to a public registry), successful hit rates for clients dynamically searching it increase dramatically.

11.5.4 Web services gateway

With web services gateway functionality, users can take an existing web service and expose it as a new service that appears to be provided by the gateway. Gateway functionality is supplied only in WebSphere Application Server Network Deployment. By using the gateway, it is possible for a web services client to access an external web service that is hosted by the gateway.

The gateway can act as a single point of control for incoming web services requests. It can be used to perform protocol transformation between messages (for example, to expose a SOAP/JMS web service over SOAP/HTTP) and 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, making 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.

11.5.5 Security

WebSphere Application Server V8 includes many security enhancements 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, such as 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 and, therefore, can be subject to several forms of attack.

► WS-SecureConversation provides session-based security, allowing secure conversations between applications 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 JAX-WS client application can be also secured by using the Web Services Security API.

11.5.6 Performance

With web services comes a trade-off between performance and interoperability. Specifically, in the use of XML encoding (marshalling and demarshalling) 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. Not all operations should be cacheable (for example, 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 MTOM standard through the optimization of the messages it provides. Avoiding the use of large messages can also help.

11.6 RESTful web services

REST has gained acceptance as a simpler alternative to SOAP and WSDL-based web services. There has been a change in interface design by Web 2.0 service providers such as Yahoo, Google, and Facebook. Those providers have either deprecated or foregone SOAP and WSDL-based interfaces in favor of an easier-to-use, resource-oriented model to expose their services.

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.

WebSphere Application Server V8 implements technologies that support RESTful architectural principles. Many of these technologies support Asynchronous JavaScript and XML (Ajax).

11.6.1 Ajax

Ajax is a set of techniques and technologies that are used to build rich, interactive web applications. It is one of the technologies that 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 only render 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 11-10. Instead of retrieving prerendered web pages (JavaServer Pages (JSP)) from the application server, the JavaScript application acts as a full fledged client application, similar in architecture to the web services model.
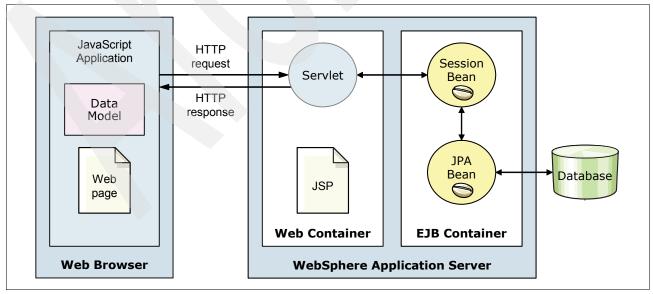
Figure 11-10   Ajax technique moving presentation logic into the browser

Ajax is supported by the feature pack for Web 2.0 and Mobile V1.1.0, which provides server enhancements to support common Web 2.0 application patterns.

11.6.2 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 an easier to use the scripting language.

- *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 machines 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 11.1, "Overview of web services" on page 330.

- *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 simply 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 such as EJBs and plain old Java objects (POJOs). Through the configuration of an RPCAdapter, HTTP GET and POST methods are mapped to Java asset operations, allowing JavaScript to call Java operations with no modifications needed to the 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.

11.6.3 Support for RESTful web services in WebSphere Application Server

Primary support for RESTful web services is provided through the feature pack for Web 2.0 and Mobile V1.1.0.

The WebSphere Application Server Feature Pack for Web 2.0 and Mobile V1.1.0 provides developers ready-to-use components to extend an SOA by connecting web services and Java EE objects into interactive desktop and mobile user interfaces. With this feature pack, WebSphere Application Server applications developed initially for desktop browsers can be adapted and deployed to mobile devices such as smartphones and tablets. The feature pack is built on top of a standard Ajax library.

An advantage of the feature pack 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 feature pack 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 is a solution for Ajax applications to display graphs, or networks, of nodes connected by links. The diagramming component can be executed through a RESTful service or on the client by using JavaScript.

► *RESTful web services* are described in 11.6.2, "Key Ajax technologies" on page 345. For an example of a RESTful web service, see the article "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 is provided with the following enhancements:

   – Atom data access to connect to ATOM services and use ATOM feeds as a data source

   – SOAP connectivity to facilitate invoking public SOAP-based web services 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

The feature pack is supported by the following Java EE application server types:

► WebSphere Application Server V8, V7, V6.1 and V6.0.2
► WebSphere Application Server Community Edition V2.1 and V2.0

For more information, seethe WebSphere Application Server Feature Pack for Web 2.0 and Mobile web page at:

http://www.ibm.com/software/webservers/appserv/was/featurepacks/web20/

11.7 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 (directly, through the service integration bus, or through an 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.

11.8 Resources

For an overall reference for developing and deploying web services in WebSphere Application Server, see *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257. Consider having a copy of this book available as you plan your web services environment. This book is based on V6.1 with the feature pack and does not cover the additions and changes made to WebSphere Application Server V8.

For an entry point to web services topics in the information center, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *web services*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

For examples of using web services in an SOA solution, see *Patterns: SOA Foundation Service Creation Scenario*, SG24-7240.

Security

WebSphere Application Server provides security infrastructure and mechanisms to protect sensitive resources and to 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
- ► Security in WebSphere Application Server
- ► Authentication
- ► Authorization
- ► Internal and external trusted relationships
- ► Security trace
- ► Auditing

12.1 Security features in WebSphere Application Server V8

This section highlights the major security features that have been added to WebSphere Application Server V8.

12.1.1 Audit service provider settings

WebSphere Application Server V7 delivered the ability to audit security events, such as successful and unsuccessful logins. These audit events can be written to a flat file on the file system. WebSphere Application Server V8 adds controls to handle conditions when these flat files become full.

The following additional settings are now 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.

12.1.2 Security hardening and migration considerations

WebSphere Application Server V8 enhances the default security theme that was introduced in WebSphere Application Server V6.1 to ensure that the configuration is set to be secure by default. The following defaults are changed as part of the security hardening features in WebSphere Application Server V8:

▶ Enablement of Secure Sockets Layer (SSL) is required on Common Secure Interoperability Version 2 (CSIv2) transport by default. This level of security ensures that all CSIv2 connections into and out of the server are using the secure SSL connection.

▶ The HttpOnly attribute on Lightweight Third Party Authentication (LTPA) cookies, by default, is set to *true* and is configurable in the Integrated Solutions Console. This attribute is a browser attribute that is created to prevent client-side applications (such as JavaScript) from accessing cookies to prevent cross-site scripting vulnerabilities. In addition, you can now enable or disable this feature from the single sign-on (SSO) page.

▶ Enablement of session security integration by default makes it possible for authenticated users to access sessions that are created in secure pages.

▶ Along with session security integration, credential persistence is enabled so that login information can be available to unprotected web clients, which allows additional access to user information.

12.1.3 Implementing a custom authentication provider using JASPI

WebSphere Application Server V8 supports JSR 196: Java Authentication Service Provider Interface (SPI) for Containers, which 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 the Java Authentication and Authorization Service (JAAS).

12.1.4 Java Servlet 3.0 support for security

WebSphere Application Server V8 supports all security updates as defined in the Java Servlet 3.0 specification (JSR 315). These updates include the servlet security annotations, use of programmatic security APIs, and the dynamic updating of the servlet security configuration.

The `com.ibm.websphere.security.displayRealm` property specifies whether the HTTP basic authentication login window shows the realm name that is not defined in the application `web.xml` file.

12.1.5 Importing and exporting LTPA keys with the AdminTask commands

With the **importLTPAKeys** AdminTask command, you can import an LTPA key from a file and add the key to the security run time and configuration. You can also use the **exportLTAPKeys** AdminTask command to export an LTPA key to a file.

12.1.6 Multiple security domains

In WebSphere Application Server V7, the federated repositories user registry can be configured only at the global level and can have only one instance per cell. In WebSphere Application Server V8, you can configure a unique instance of a federated repository at the domain level in a multiple security domain environment.

When a security domain is copied from the global level, the users and groups that are defined at the global level are also copied to the security domain. This feature is also true when copying from an existing domain. A newly created security domain that uses the file-based virtual memory manager (VMM) repository requires the user to populate the repository with users and groups.

Also the new "Use global schema for model" option is available on the Realm configurations settings panel on the administrative console page. This option sets the *global schema* option for the data model in a multiple security domain environment. Global schema refers to the schema of the administration domain.

12.1.7 Security configuration report

The security configuration report in WebSphere Application Server V8 now includes information and recommendations about cookie protection, session security, web attributes, and the HttpOnly setting. The report is a table that includes the following columns:

► Console name
► Security configuration name
► Value
► Console path name

The security information that is gathered is divided into sections and groups of common security information.

Figure 12-1 shows an example of the security report.

Figure 12-1   Security report

You can run the Security Configuration Report from the administrative console by selecting **Security** → **Global Security** and then by clicking **Security Configuration Report**. A new window opens that shows the report information.

12.1.8 Security custom properties

With WebSphere Application Server V8, the default value for the `com.ibm.CSI.propagateFirstCallerOnly` security custom property is set to `true`. When this custom property is set to `true`, the first caller in the propagation token that stays on the thread is logged when the security attribute propagation is enabled. When this property is set to `false`, all of the caller switches are logged, which can affect performance.

12.2 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 defense being compromised or circumvented. This type of security is becoming more

important today, for example, with the frequent occurrence of cyber attacks in which hackers work to steal credit card numbers or attempt to steal sensitive military secrets.

WebSphere Application Server resides in one of the defensive layers. It takes responsibility and offers the capability to protect and defend itself in mitigating risk. Figure 12-2 illustrates these security layers and how WebSphere Application Servers fits into the layer of defense.
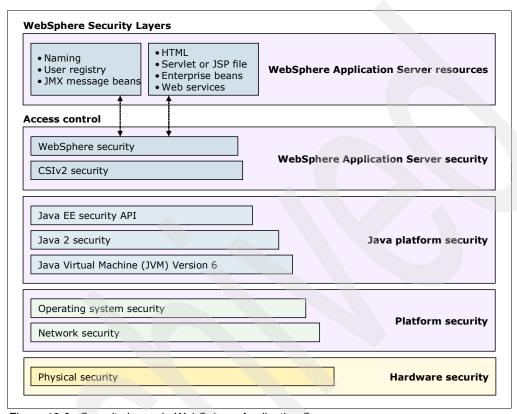
Figure 12-2   Security layers in WebSphere Application Server

WebSphere Application Security includes the following security layers (from bottom to top):

► 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, then no data is secure on that server.

► Network security

The network security layers provide several technologies, such as firewalls, to provide a level of network-based attack and protection, transport level authentication and message integrity, confidentiality, and more.

SystemSSL: WebSphere Application Server for z/OS provides SystemSSL for communication using the Internet. SystemSSL is composed of the SSL and Transport Layer Security (TLS), which enable secure file transfer by providing data privacy and message 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 Access Facility (SAF) works with the user registry to authorize applications to run on the server.

  > **Non-root user in distributed platforms:** If you are interested in protecting your system from applications, run WebSphere Application Server as a non-root user in distributed platforms 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) 6.0

  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 to 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, class loading, and so on. 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 drops the signed code and sandbox model, which 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.

  > **Enabling Java 2 security:** 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, you need to 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.

  > **Java 2 security and performance:** Enabling Java 2 security can cause a significant performance overhead.

- ► Java EE security API

  The security collaborator enforces Java EE-based security policies and supports Java EE security APIs. The Java EE standard API describes a few methods with which the application can obtain the logged in user's name and role membership. WebSphere Application Server never returns the password of any user 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 an IIOP-based, three-tiered, security protocol 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 invoked over the CSIv2 security protocol, which 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 Authentication Service (SAS) is supported only between WebSphere Application Server V6 and previous version servers that have been federated in a V6.1 cell.

► WebSphere security

WebSphere security enforces security policies and services regarding 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.

12.3 Authentication

Authentication is the process of confirming a user or system identity. The authentication mechanism in WebSphere Application Server uses a user registry to perform 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 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 (RSA) token authentication
► Web Services Security Security Assertion Markup Language (SAML) Token Profile

Figure 12-3 shows the authentication mechanism selection list.

Authentication
Authentication mechanisms and expiration

⊙ LTPA

○ Kerberos and LTPA
    Kerberos configuration

○ SWAM (deprecated): No authenticated communication between servers

Authentication cache settings

⊞ Web and SIP security

⊞ RMI/IIOP security

⊞ Java Authentication and Authorization Service

☐ Enable Java Authentication SPI (JASPI)
    Providers

☐ Use realm-qualified user names

Figure 12-3   WebSphere Application Server V8 selectable authentication mechanisms

Panel display on a single server edition: This panel is shown only on a single server edition. WebSphere Application Server Network Deployment edition does not offer the SWAM as a selectable authentication mechanism.

12.3.1 Lightweight Third-Party Authentication

LTPA is intended for single and multiple application server and machine environments as the default user authentication protocol. It supports forwardable credentials and SSO. LTPA can support security in a composite environment through cryptography. The LTPA token contains authentication-related data, which 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 appear to be prematurely expired 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 has 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.

> **LTPA configuration:** When security is enabled during profile creation time, LTPA is configured by default.

LTPA keys are generated automatically during the first server start and are regenerated before they expire. You can disable automatic regeneration by WebSphere Application Server so that you can generate keys on a schedule.

12.3.2 Kerberos

Kerberos is a standard network authentication protocol that is used in providing a proof of identity between a client and server or between a server and a server. 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. By using Kerberos technology, a user can log in one time and then have access to other applications that support Kerberos Authentication without logging in a second time.

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, or 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, which can be reused by the client to request several more 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, then the KRBAuthnToken contains the delegate principal credentials.

WebSphere Application Server V8 supports both LTPA and Kerberos and supports both of them 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 great advantage that the actual password never leaves the user machine. 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 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 users' 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 in to any of the managed systems.

12.3.3 Rivest, Shadler, and Adleman token authentication

The RSA authentication mechanism is used to simplify the security environment for the flexible management topology. It supports the ability 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.

> **Using the RSA token authentication mechanism:** You can use the RSA token authentication mechanism only for administrative requests. As such, the authentication mechanism choices for administrative authentication are part of the Global Security panel of the Integrated Solutions Console, with the "Security Global security Administrative authentication" option.

The RSA token authentication mechanism ensures that, 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 perform the cryptographic operations on the RSA tokens and 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, which is established based on certificates in the target RSA truststore. RSA token authentication does not use the same certificates that are used by 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.

12.3.4 Single sign-on

With SSO support, web users can authenticate once 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 or 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 and can 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 Servers and Domino servers, you must configure SSO for both types of servers.

The following list describes requirements for enabling SSO using LTPA. Other authentication mechanisms might have different requirements.

► All SSO participating servers must use the same user registry (for example, the Lightweight Directory Access Protocol (LDAP) server).

► All SSO participating servers must be in the same domain name system. (Cookies are issued with a domain name and will 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 not to work properly.

► 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 12.3.5, "Simple and Protected GSSAPI Negotiation Mechanism" on page 359. Microsoft Windows users can access WebSphere Application Server resources without requiring an additional authentication process after being authenticated by a Domain Controller.

For detailed information about SPNEGO web authentication, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *creating a single sign-on for HTTP requests using SPNEGO Web authentication*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

12.3.5 Simple and Protected GSSAPI Negotiation Mechanism

SPNEGO is used when a client application wants to authenticate to a remote server, but neither the server nor the client can detect which authentication protocol the other supports. WebSphere offers SPNEGO support, which allows SSO between Microsoft and WebSphere web-based applications. Many customers use SPNEGO as an SSO solution between the Microsoft Windows desktop and WebSphere.

12.3.6 Java Authentication and Authorization Service

The 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.

> **JAAS support:** 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 invoked 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 a specific principal only. If so, the operation is supported if the subject associated with the access control context contains the designated principal only.

12.3.7 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 while the product applies its own authorization policy to the resulting credentials passed by the reverse proxy server.

> **SPNEGO TAI deprecation:** SPNEGO TAI is deprecated in WebSphere Application Server V8.

Demand for such an integrated configuration has become more compelling, especially 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 further use 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.

12.3.8 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 in the `<wsse:Security>` header, as defined by the Web Services Security: SOAP Message Security specification.

> **Support:** The SAML Token Profile is 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, perform protection of the SOAP message, if required, by using a transport level mechanism or another security token, such as an X.509 or Kerberos token, for message-level protection.

▶ Holder-of-key

  When using the *holder-of-key* subject confirmation method, proof of the relationship between the subject and claims 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 so that the receiver can 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 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 Services (STS) or are self-issued by the application server.

12.4 User registries

The information about users and groups resides 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 performed.

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 LDAP

► Custom registry

► Federated repository (a combination of a file-based registry and one or more LDAP servers in a single realm)

12.4.1 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.

> **Local operating system registry:** 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 machine 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 (AIX, Linux, Solaris, and HP-UX), the process ID that runs the WebSphere Application Server process needs root authority to call the local operating system APIs for authentication and to obtain user or group information.

12.4.2 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.

Choosing a registry option: 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 WebSphere Portal V6.1 and later and 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, and IBM has no plans to further enhance this option. IBM strongly recommends that customers 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, you need to migrate to the federated repositories option before these upgrades.

For more information, see the WebSphere Application Server Version 8 Information Center at the following address, and search for the following topics:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

► *Federated repositories*

► *Migrating a stand-alone LDAP repository to a federated repositories LDAP repository configuration*

Using a load balancer: 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 Base and Network Deployment V8.0 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, you must configure the LDAP server by using a custom LDAP feature with an appropriate filter. You must obtain the appropriate filter information from the LDAP vendor.

> **Nested and dynamic groups:** 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 given 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 member of a group.

12.4.3 Custom registry

With the custom user registry, you can connect to any type of user repository. For the custom registry, you can implement the SPI. The SPI is the UserRegistry interface, which is the same interface used by the local 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, you must decide how to map the information that is manipulated by the UserRegistry interface to the information in your registry.

12.4.4 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 sub-tree) repository

For information about the LDAP repository, see 12.4.2, "Stand-alone Lightweight Directory Access Protocol" on page 362.

► Database repository

Database user repositories have been supported since WebSphere Application Server V7. The application server connects to a JDBC resource, which points to a database and a table, which must include the standard VMM entity types PersonAccount, Group, and OrgContainer.

> **Configuring a database repository:** The database repository is configurable only by using the `wsadmin` command-line interface (CLI).

► Custom registry

For a description of this custom registry, see 12.4.3, "Custom registry" on page 364.

12.5 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 in to the host operating system and, if access is granted by the system administrator, the user can issue WebSphere command-line commands, such as `startServer.bat`, `stopServer.sh`, `versionInfo.bat`, and other commands. 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 in to the administrative console but 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 line:

```
stopServer server1 -user wasadmin -password admin
```

For more details about the administrative console user role, see "Fine-grained administrative security" on page 367.

► *Application users* have no access to the operating system or the administrative console. They can only log in 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 369.

The user accounts for the administrative and application users are stored in a user registry, for example in an LDAP server.

12.6 Authorization

Authorization is the process of checking whether a given 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

12.6.1 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 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, as illustrated in Figure 12-4 on page 367:

- ► Monitor

    The monitor role has the fewest permissions and confines 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 and has the permission that is required to access sensitive data, including server password, LTPA password and keys, and so on.

    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 from within the Integrated Solutions Console only.

> **Important:** The ISC Admins role is available only for Integrated Solutions 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 remainder of the security configuration. For more information about the auditor role, see 12.9, "Auditing" on page 375.
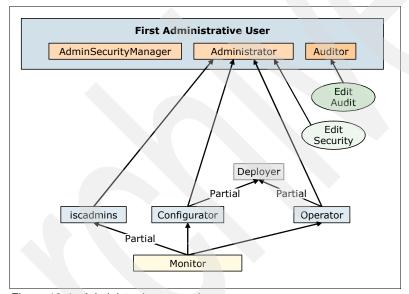
Figure 12-4   Administrative user roles

Fine-grained administrative security

The traditional all-or-none security model was extended in WebSphere Application Server V6.1 with the optional fine-grained administrative security. In WebSphere Application Server V6.1, security was configurable by only using the `wsadmin` CLI. Since WebSphere Application Server V7, configuration from the administrative console is possible.

Fine-grained administrative security can grant access to each user role for 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. One global security configuration must be defined before the security domains can be created. If no security domains are configured, all of the applications use information from the global security configuration.

When a security domain is created and associated with a scope, only the user applications in that scope use the security attributes that are defined in the security 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 have to 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 (such as when the responsibility for user administration of each registry falls on different teams).

12.6.2 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 12-5 shows the components of a Java EE application.
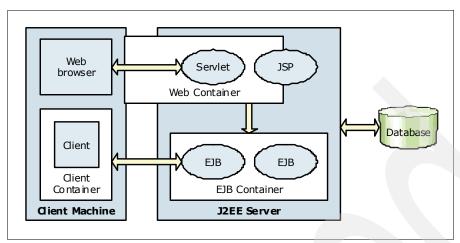
Figure 12-5   Java EE application components

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 if 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, you must consider the security for every module.

Security roles

A *security role* is a logical grouping of users that is defined by the application assembler.
Because 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 create named sets of users (for example
managers, customers, and employees) who have 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 12-6 shows an example of how roles can be mapped to users.

Figure 12-6   User role mapping

This two-phase approach to security gives a great deal of flexibility, because 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 371.

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 gives you 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 transparent to the developer. However, there are cases where 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 Version 8 Information Center at the following address, and search for *developing applications that use programmatic security*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

Java security

Java EE security guards access to web resources (such as servlets, JSPs, and EJBs) and to system resources (such as file I/O, sockets, and properties).

> **New requirements for enabling Java security:** Java security places new 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 Version 8 Information Center (at the previous web address), and search for *Java 2 security*.

Security annotations

Java annotations are powerful programming tools resulting from the JSR-175 recommendation. They are a standard way to include supported security behaviors while allowing the source code and configuration files to be 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 performed 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 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 12-7 shows the communications flow.
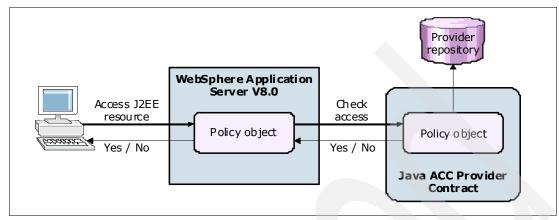
Figure 12-7   Java ACC provider architecture

Authorization decisions: 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.

12.7 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.

12.7.1 Secure communications

To prevent eavesdropping on communications, you must add security. WebSphere Application Server uses Java Secure Sockets 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 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, 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, interchanging 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 Integrated Solutions 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 trust holds store 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.

12.7.2 SSL in cell management

The WebSphere Application Server uses SSL to communicate within the cell among the nodes. 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 1-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 might be overridden:

► Keystore password
► Expiration time in years
► Distinguished name (DN) 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 another one, signed by the cell root

certificate, and it 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 1 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 1 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 keeps track of 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 Version 8 Information Center at the following address, and search for *certificate expiration monitoring in SSL*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

Web server plug-in key ring

When the web server forwards HTTPS requests, it needs to communicate directly with the application servers, for example 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 but 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.

12.7.3 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

12.8 Security trace

WebSphere Application Server V8 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.*`

> **Federated repository:** The `com.ibm.ws.wim.*` class is for tracing with the federated repository.

- ► `SASRas`

The trace facility has different logging levels, among them the 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.

12.9 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 and, 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) with the most common aspect of 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 with an event outcome filter. The captured events, which go through both filters, are logged in to the audit log.

The security auditing subsystem can capture the following types of auditable events:

- ► Audit subsystem-related runtime events (start, stop, and so on)
- ► 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 (start, stop, and so on)

- ► 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. Create a separate auditor user role and user principal. Then 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 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 logged in 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.

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 it. 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 `security.xml` file.

The security audit subsystem is built on the following plug-ins. Default plug-in implementations are shipped with WebSphere Application Server V8 that capture and output the audit records to a binary audit log file:

► Audit Event Factory, which captures data
► Audit Service Provider, which outputs the captured data to a back-end repository

If there is a special need for the 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 on z/OS uses 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 facility 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 information about setting up and starting SAF and SMF, see *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

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, but these logs are not for human interpretation. WebSphere Application Server V8 has an Audit Reader Utility that reads the audit message log file and generates an HTML report. You can invoke this utility by using a `wsadmin` command.

The following Jython administrative script sample generates a basic audit report (Figure 12-8):

```
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 12-8   Audit utility report

12.10 Resources

For an overall reference resource for WebSphere Application Server security, see *WebSphere Application Server V7.0 Security Guide*, SG24-7660. Consider having a copy of this book available as you plan to secure your environment. Keep in mind that this book is written for WebSphere Application Server V7. Therefore, it does not cover the new features and changes found in V8.

For up-to-date information about WebSphere Application Server V8, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *securing applications and their environment*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

13

Feature packs for WebSphere Application Server

A feature pack is an optionally installable product extension for WebSphere Application Server that provides a set of new related standards and innovative features. With feature packs, users can take advantage of these new standards and features without having to wait for a new release of WebSphere Application Server.

This chapter describes the feature packs that are currently available for WebSphere Application Server V8 and highlights the important aspects to consider for a successful implementation.

This chapter includes the following sections:

► Available feature packs
► WebSphere Application Server Feature Pack for Web 2.0 and Mobile
► Resources

379

13.1 Available feature packs

The currently available feature pack for WebSphere Application Server V8 is WebSphere Application Server Feature Pack for Web 2.0 and Mobile version 1.1.0. Other feature packs for WebSphere Application Server V8 will be released later.

The following previously available feature packs for WebSphere Application Server V7 are integrated into WebSphere Application Server V8. You can use their features without additional installation procedures.

► WebSphere Application Server Feature Pack for Communications Enabled Applications
► WebSphere Application Server Feature Pack for Modern Batch
► WebSphere Application Server Feature Pack for OSGi Applications and JPA 2.0
► WebSphere Application Server Feature Pack for SCA
► WebSphere Application Server Feature Pack for XML

In addition, the following feature packs for WebSphere Application Server V6.1 were already integrated into the product and are still available in WebSphere Application Server V8:

► WebSphere Application Server Feature Pack for EJB 3.0
► WebSphere Application Server Feature Pack for Web Services

The WebSphere Application Server Feature Pack for Dynamic Scripting was not integrated, but you can use the previous version with WebSphere Application Server V8.

For a detailed description of the WebSphere Application Server Feature Pack for Dynamic Scripting, see *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708.

13.2 WebSphere Application Server Feature Pack for Web 2.0 and Mobile

This section describes the IBM WebSphere Application Server V8 Feature Pack for Web 2.0 and Mobile.

13.2.1 Highlights and new features in Version 1.1.0

This section highlights the new features of Version 1.1.0 of the Web 2.0 and Mobile feature pack.

> **Installation guidance:** The feature packs for WebSphere Application Server V8 must be installed by using the IBM Installation Manager.

IBM ILOG Dojo Diagrammer

The IBMILOG Dojo Diagrammer is a Dojo widget with advanced diagramming and graph layout capabilities. The purpose of Dojo Diagrammer is to provide a solution for Dojo-based applications to display and interact with graphs, such as networks of nodes connected by links.

IBM ILOG® Diagrammer includes the following features:

► The ability to create graph displays by code or from data in a Dojo data store

► The ability to lay out the nodes and links automatically with graph layout algorithms (Tree, Hierarchical, Force Directed, Circular, Grid Layout, and Link routing)

- The ability to start Graph Layout on the client or on the server through a Graph Layout Representational State Transfer (REST) service
- The ability to customize the appearance of nodes and links through styling and templating
- Graph editing features
- Support Web and Mobile deployment

Showcase gallery

The feature pack for Web 2.0 and Mobile V1.1.0 provides a showcase gallery, featuring the new functionality that is available. The showcase provides samples and demonstrations for the following features:

- Dojo mobile, including widgets, views, charts, gauges, animations, maps, and model-view-controller forms
- Application services that demonstrate a file uploader service with the Dojo FileUploader support, directory listing service, and other services
- IBM ILOG Dojo Diagrammer featuring graphs, node layouts, and other visual diagrams

Mobile Application Services

The feature pack includes new REST-based mobile and rich Internet application (RIA) building blocks, which are provided to simplify the creation of mobile-focused applications.

The building blocks include the following features:

- Directory listing and file upload sample applications
- Graphics conversion service
- Logging, debugging, and analytics capture service
- Device and browser detection service
- Apache Wink WebDAV extension

Dojo

The feature pack for Web 2.0 and Mobile V1.1.0 includes Dojo V1.7, which provides the following benefits:

- Support for HTML 5 compliant attribute names
- New `watch()` method to monitor changes to widget attributes
- New Charting types and Legend options
- New Editor plug-ins
- New EnhancedGrid plug-ins
- Support for the latest browsers including Internet Explorer 9 and Firefox 4.0

Ajax

With Asynchronous JavaScript and XML (Ajax), the interaction model for web applications has become more robust, similar to desktop applications, with continuous interaction and improved usability. Adding Ajax capabilities to your applications includes the following benefits among others:

- A more interactive, differentiated experience which can lead to longer sessions and increased customer loyalty
- Responsive, local actions, which can result in fewer abandoned transactions, higher completion rates, and higher user productivity

Java libraries

The following new technologies exist in Java libraries:

► The Ajax proxy provided with this feature pack is a reverse proxy that is installed near one or more servers.

► Web-remoting is a pattern that provides for JavaScript or a client-side to directly start server-side logic.

► JSON4J library is an implementation of a set of JavaScript Object Notation (JSON) handling classes for use with Java environments.

The web messaging service is a publish or subscribe implementation that connects the browser to the WebSphere Application Server service integration bus (SIBus) for server-side event push to the browser.

13.2.2 Overview of Web 2.0

Web 2.0 describes a set of new technologies that are designed and created to improve existing technologies in the World Wide Web (WWW). Web 2.0 is not a specification and is subject to different definitions and interpretations. In this book, considerations have been made for those technologies included in the feature pack.

Key technologies and concepts

The following main technologies and concepts are being used for building Web 2.0 applications:

► Asynchronous JavaScript and XML

   Ajax is an open technique for creating rich user experiences in web-based applications that do not require web browser plug-ins. It is the most important technology in the Web 2.0 world. Its main characteristic is that, by exchanging small amounts of data with the server, it renders web pages in the web browser without reloading entire pages. As a result, it increases the interactivity, usability, and speed of the user experience. Behind the scenes, Ajax implements a stateful client that interacts asynchronously with the server.

   Dojo is a developer toolkit of Ajax. It is a JavaScript library grouped into three main developer areas. It contains the Ajax standard and extended developer tools and some widgets.

► Representational State Transfer

   REST is a software architecture for building hypermedia systems. It consists of a series of principles that outline how resources are defined and addressed. Any Internet application should follow these principles, but some applications violate them (for example, having large amounts of server-side session data). Ajax, on the contrary, follows the key principles of REST.

► JavaScript Object Notation

   JSON is a human readable data interchange format that is used in Ajax as an alternative format to XML. Although it is based on a subset of the JavaScript language, it is language-independent.

► Atom

Atom was developed as an alternative to RSS feeds to overcome RSS incompatibilities. It is composed of two different standards:

Atom syndication format
An XML language used for *web feeds*, which are a mechanism for publishing information and subscribing to it.

Atom publishing protocol
An HTTP-based protocol for creating and updating web applications.

13.2.3 Overview of the Web 2.0 and Mobile feature pack

The Web 2.0 and Mobile feature pack extends service-oriented architecture (SOA) by connecting external web services, internal SOA services, and Java Platform, Enterprise Edition (Java EE), objects into highly interactive web application interfaces. It provides a supported, best-in-class Ajax development toolkit for WebSphere Application Server. It also provides a rich set of extensions to Ajax.

The following extensions to Ajax are included:

► Client-side extensions:

Ajax client run time Is the main component of the feature pack. It includes extensions to the Dojo toolkit 1.6 for building Ajax and mobile web applications. It also supports the IBM ILOG Diagrammer for displaying advanced diagrams at the client side.

IBM Atom library Consists of utility functions, data models, and widgets for creating, modifying, and viewing Atom feeds.

IBM gauge widgets Are provided as base objects to build other widgets on them. The two gauge widgets are the analog gauge and the bar graph.

IBM SOAP library Implements a simple way to create a SOAP envelope around a request and a widget to connect to external SOAP services.

IBM OpenSearch library
Is used to interface with a server with open search capabilities. Typically, the return type is X/HTML, an Atom, or an RSS feed.

IBM ILOG diagrammer
Provides a set of controls for building custom diagram and dashboard displays.

► Server-side extensions:

Ajax proxy An implementation of a reverse proxy that accepts requests from a client and redirects them to one or more servers while letting the client believe that all the responses from its requests come from the same server.

Web messaging service
A publish and subscribe implementation that connects a web browser to the WebSphere Application Server service integration bus for a server-side event push.

JSON4J libraries An implementation of a set of JSON classes. They are also capable of transforming XML messages to and from JSON. The advantage of this feature is that Ajax-patterned applications can handle JSON formatted data without needing a third-party component.

RPC adapter libraries

> The IBM implementation for web remoting, which is a pattern with which you can invoke Java methods on the server side from JavaScript.

Feed libraries   The Apache Abdera-based feed libraries support RSS feed, while the Apache Wink-based feed libraries have both read and write support for RSS. The Wink library also supports the Atom and Atom Publishing Protocol (APP) protocols in addition to RSS.

Figure 13-1 shows the main components of this feature pack.
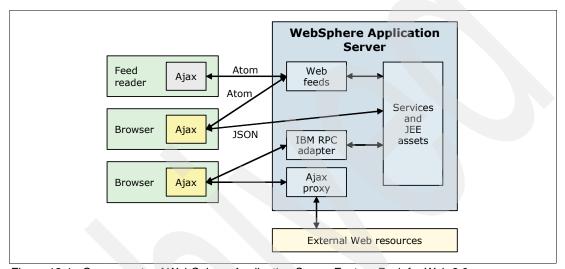
Figure 13-1   Components of WebSphere Application Server Feature Pack for Web 2.0

13.2.4 Security considerations

The security issues on Ajax are well known. Different forms of attacks can take advantage of the usage of client-side scripts that can be easily forged to consume information from untrusted sources or to collect confidential data from the user.

Adopting some measures can lead to more secure applications. Among the measures you must consider are input validation, proper coding, loading scripts only from trusted sources, encryption, and a correct server security configuration on the server side.

13.3 Resources

For additional information, see WebSphere Application Server Feature Pack for Web 2.0 and Mobile at:

http://www.ibm.com/software/webservers/appserv/was/featurepacks/web20-mobile/

For a complete overview and detailed information about the Feature Pack for Web 2.0 and Mobile, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *overview of Feature Pack for Web 2.0 and Mobile*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

For an in-depth description of the feature pack and an essential reference for using it, see *Building Dynamic Ajax Applications Using WebSphere Feature Pack for Web 2.0*, SG24-7635.

14

WebSphere Application Server for z/OS

This chapter concentrates on the features of WebSphere Application Server for z/OS V8. It includes the following sections:

► WebSphere Application Server structure on z/OS
► Functions for WebSphere Application Server for z/OS V8
► Installing WebSphere Application Server for z/OS
► System programmer considerations
► Planning checklist
► Resources

The features and functions described in this chapter are available only with WebSphere Application Server for z/OS V8. The new functions and features of WebSphere Application Server for z/OS V8, and the existing functions and features described in other chapters of this book, are implemented on the z/OS platform, but are not explained in this chapter.

14.1 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 be not familiar with the z/OS operating system, this section includes explanations of z/OS terms or techniques in general IT terminology and how they might add value to your business environment.

14.1.1 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 the back-end server

WebSphere Application Server for z/OS uses the Workload Manager (WLM) component to assign resources to the application server automatically, in order to achieve the performance goals that are set for the environment. You can set these goals on a transaction level, assuring that platinum customers get the best response time.

Local connectors to databases that are running in the same operating system image can enhance the throughput and decrease the used CPU resources.

► 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 by forming 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 to 99.999%. If unplanned downtime occurs, System z and z/OS offer disaster recovery capabilities that bring the system to a productive, industry-leading state.

► Reduced management cost through manageability

The management capabilities of the z/OS platform have evolved, resulting in a platform with the lower management costs and with a high degree of automation and transparency for administrators.

► Reduced cost through 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 cheap to purchase but 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 environment.

► Secure environment to stabilize operations and production

The usage of a central security repository, the Resource Access Control Facility (RACF) can ease the security model, because 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.

14.1.2 Benefits of using WebSphere Application Server for z/OS

This section highlights the benefits of using WebSphere Application Server V8 for z/OS from a security, availability, and performance perspective.

Security

In terms of security, the distinct area for user code offers more protection for other system components running 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, execute scripts, and so on. In WebSphere Application Server for z/OS, these basic functions are performed in the control region. However, the user code is run in the servant region that 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 launches the failed servant again automatically.

► 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 remain available with your application, keep in mind that you can 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, because more requests can 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. The system adjusts resources automatically on a 7x24 year-round basis to ensure that these goals are kept.

  Through the usage of multiple Java virtual machines (JVMs) with smaller heaps, the performance on a single transaction decreases during garbage collection, and general throughput increases.

14.1.3 Common concepts

Both implementations of WebSphere Application Server V8 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, administrative console, and all the other components.

► 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 if the application meets the requirements that are common to the product. 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 (except for the update installer).

> **z/OS as the operating system for WebSphere Application Server:** Using z/OS as the underlying operating system for WebSphere Application Server does not mean rebuilding your processes for administration, operation, and development or training administration staff to learn 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.

14.1.4 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 for each z/OS image (Figure 14-1). 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.
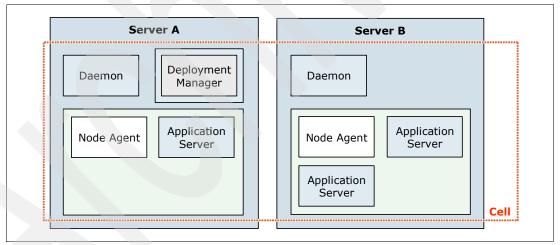
Figure 14-1   WebSphere Application Server for z/OS daemon usage in a cell

Daemon servers are started automatically when the first server for the cell on that z/OS image is started (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.

14.1.5 Structure of an application server

This section provides a conceptual view of an application server inside WebSphere Application Server for z/OS.

Overview

In general, WebSphere Application Server for z/OS uses the same concepts to create and manage an application server. Each application server (or instance of a profile) is built from the following building blocks to represent a single application server:

- ► Control region
- ► Servant region
- ► Control region adjunct

Figure 14-2 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*, because communication with the outside world ends in the control region.

> **A WLM queue:** A WLM queue is used to queue work for further processing. Each queue uses a first-in first-out 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.
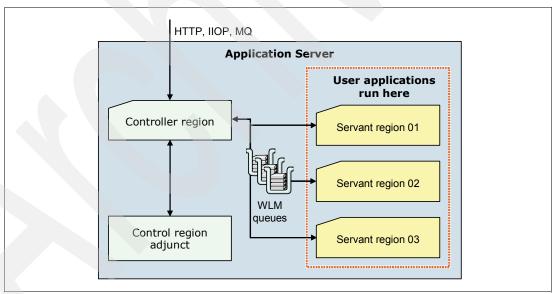
Figure 14-2   Building blocks of the WebSphere Application Server for z/OS V8

Although WebSphere Application Server profiles on z/OS are built by using multiple building blocks, they are still part of a single instance of an application server from an application developer, system administrator, or user perspective. Thus, nearly all WebSphere variables can be defined against a server and are not defined against the servant and control region adjuncts. However, some of the settings, such as heap sizes, must be defined for each component.

All the possible profiles that can be instantiated are built by using the following regions:

- ► 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 executed, 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 only have one control region.
- ► The control region contains a 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 Enterprise JavaBeans (EJB) and web container are included here.

As shown in Figure 14-2 on page 389, you can have multiple servant regions for each application server, but only one servant region for the other profile types. This concept is called a *multi-servant region* or *internal cluster*. Through the usage of this technique, you can take advantage of cluster benefits without the overhead 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 server scope.

- ► An application must be WebSphere Application Server cluster-ready to use the multiservant concept.

- ► The number of servant regions is transparent 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 crashed servant region fail and deliver errors to the user. The other servant regions continue to 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.

14.1.6 Runtime processes

This section describes the runtime behavior of WebSphere Application Server for z/OS V8.

Overview

The non-z/OS platforms are built on a single process model. Thus, the entire application server runs in one single JVM process. WebSphere Application Server for z/OS is built by using a federation of JVMs, each executing in a different *address space*. Together, such a collection represents a single server instance, as illustrated in Figure 14-2 on page 389.

> **Address space?** 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 executing instructions and storing data.

During run time, each building block of an application server or a deployment manager opens an address space, as illustrated in Figure 14-3.
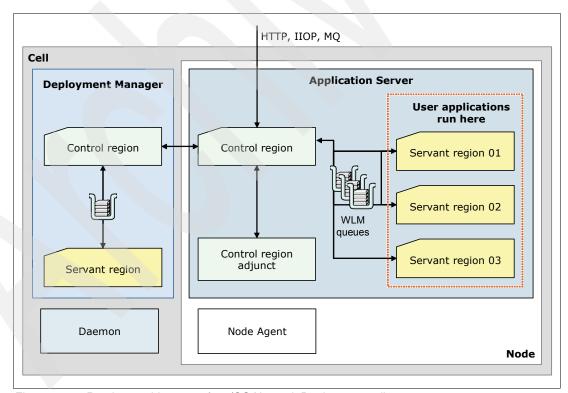
Figure 14-3   Runtime architecture of a z/OS Network Deployment cell

The WebSphere Application Server for z/OS environment shown in Figure 14-3 on page 391 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 each servant region contain a JVM. The installation shown in Figure 14-3 on page 391 has six JVMs, because there are two control regions and four servant regions.

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 executes the user application. Therefore, in terms of specialized JVMs on z/OS, the maximum amount of heap storage defined for the various heaps is reduced, 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 (as described in 14.3.2, "Installation considerations" on page 418):

► Amount of real storage
► Minimum and maximum size for the different heaps
► Shared class cache usage

> **Shared class cache:** 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 into other JVMs. For more information about z/OS settings for the shared class cache, see 14.4.2, "Java settings" on page 426.

14.1.7 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 Version 8 Information Center at the following address, and search for *workload management on z/OS*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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 an 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 this goals by comparing the definitions of the service class to real-time data on how the system is currently 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 and 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 keep track of consumed 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™ (IBM RMF™) is used.

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 execute the work of the thread with the priority of the enclave.

WebSphere Application Server for z/OS uses this technique to pass transactional work, 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 and its previous versions can classify incoming work on a server base. To begin, the control region of an application server determines to which application server the request belongs. 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 14-4, incoming work is assigned a service class, based on information of the user-work request. The granularity is on the application server level.
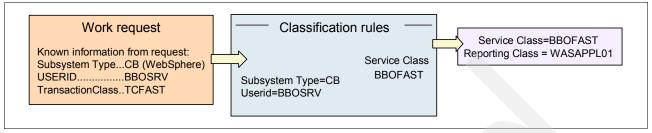
Figure 14-4   Workload classification for WebSphere Application Server for z/OS

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, giving platinum customers a better response time than gold customers, as illustrated in Figure 14-5.
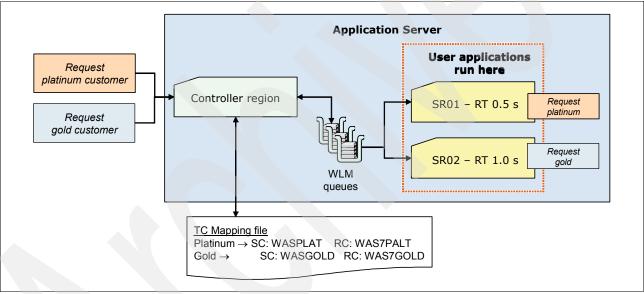
Figure 14-5   Transactional assignment of performance goals

A request that enters the system is assigned a transaction class, 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 using a *workload classification document*. This document is an XML file that classifies inbound HTTP, Internet Inter-ORB Protocol (IIOP), message-driven bean (MDB), Session Initiation Protocol (SIP), optimized local adapter, and mediation work requests and assigns them to a transaction class (TCLASS).

> **Workload classification in a z/OS environment:** 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 to assign a service class or a report service class to each request.

For detailed information about the transaction classification, see the WebSphere Application Server Version 8 Information Center at the following address, search for *controller and servant WLM classifications*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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

Granularity improvement: 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 410.

To implement this document, complete these steps:

1. Talk with the application development and the business functions teams to define the transactions that need performance goals.

2. Create a workload classification document, such as the one shown in Example 14-1 on page 396.

More information: For detailed information about the workload classification document, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *administrator best practices*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

3. Ensure that the file is in ASCII format. If you create the document on a z/OS system in code page IBM-1047 (normal in hierarchical file system (HFS)), convert it to ASCII by using the following command:

```
iconv -f IBM-1047 -t UTF-8 ebcdic.file.name.xml > ascii.filename.xml
```

4. Configure the server to use the classification document. Use the wlm_classification_file variable to specify the XML file that contains the classification information. In the administrative console, click **Environment** → **WebSphere variables** → **New**.

5. Reload the file, and then choose one of the following options:

 – Restart the application server.

 – Reload the workload classification document by issuing the following command:

```
MODIFY|F servername,RECLASSIFY,FILE='/path/file_name.xml'
```

Sample workload classification document

Example 14-1 shows the workload classification document with IIOP and HTTP elements.

Example 14-1   Workload classification file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Classification SYSTEM "Classification.dtd" >
<Classification schema_version="1.0">
<!-- Internal Classification Rules -->
   <InboundClassification type="internal"
                           schema_version="1.0"
                           default_transaction_class="value1"/>
<!-- IIOP Classification Rules -->
   <InboundClassification type="iiop"
                           schema_version="1.0"
                           default_transaction_class="A0">
     <iiop_classification_info transaction_class="A1"
                               application_name="IIOPStatelessSampleApp"
                               module_name="StatelessSample.jar"
                               component_name="Sample20"
                               description="Sample20 EJB Classification">
     </iiop_classification_info>
   </InboundClassification>

<!-- HTTP Classification Rules -->
   <InboundClassification type="http"
                           schema_version="1.0"
                           default_transaction_class="M">
     <http_classification_info transaction_class="N"
                               host="yourhost.yourcompany.com"
                               description="Virtual Host yourhost">
     </http_classification_info>
   </InboundClassification>
</Classification>
```

Example 14-2 shows the document type definition (DTD) file that is related to the workload classification document.

Example 14-2   Workload classification DTD file

```
<?xml version='1.0' encoding="UTF-8"?>
<!ELEMENT Classification (InboundClassification)+>
<!ATTLIST Classification schema_version CDATA #REQUIRED>
<!ELEMENT InboundClassification
((iiop_classification_info*|http_classification_info*))>
<!ATTLIST InboundClassification type (iiop|http) #REQUIRED>
<!ATTLIST InboundClassification default_transaction_class CDATA #REQUIRED>
<!ATTLIST InboundClassification schema_version CDATA #REQUIRED>
<!ELEMENT iiop_classification_info (iiop_classification_info*)>
```

```
<!ATTLIST iiop_classification_info activity_workload_classification CDATA
#IMPLIED>
<!ATTLIST iiop_classification_info application_name CDATA #IMPLIED>
<!ATTLIST iiop_classification_info component_name CDATA #IMPLIED>
<!ATTLIST iiop_classification_info description CDATA #IMPLIED>
<!ATTLIST iiop_classification_info module_name CDATA #IMPLIED>
<!ATTLIST iiop_classification_info transaction_class CDATA #IMPLIED>
<!ELEMENT http_classification_info (http_classification_info*)>
<!ATTLIST http_classification_info host CDATA #IMPLIED>
<!ATTLIST http_classification_info port CDATA #IMPLIED>
<!ATTLIST http_classification_info uri CDATA #IMPLIED>
<!ATTLIST http_classification_info description CDATA #IMPLIED>
<!ATTLIST http_classification_info transaction_class CDATA #IMPLIED>
```

Servant activation

As described in "Servant region" on page 390, an application server can have multiple servant regions defined that all process the user application. 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 the servant regions can be defined by the system programmer.

14.1.8 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. Although it is theoretically possible to address 2 GB of storage with a 31-bit server, 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 it, because this mode is deprecated in V8.

► The migration of a server from V6.1 to V8 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.

- Because 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 14-3). The *server_name* is the name of the server that is running in 31-bit mode.

Example 14-3   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, 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 a 64-bit JVM. Although virtually all versions of purchased software support the usage of 64-bit mode, this point might be of concern for user-built applications that are migrated from a 31-bit environment.

Real and auxiliary storage

The use 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 10% when compared to 31-bit mode when the same JVM heap size is used in the system.

> **Additional memory with 64-bit mode:** 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 will take 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 the JVM heap will not be 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 (for example, an exit, 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 stated in the installation guide, sometimes the tendency occurs to 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.

14.1.9 XCF support for WebSphere HA manager

WebSphere Application Server for z/OS V8 adds the option to use the cross-system coupling facility (XCF) system services to monitor the status of cluster components instead of the default common code base technique. This type of implementation provides the following value:

► Reduced CPU overhead

Using the XCF reduces the overhead 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.

> **XCF usage:** 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 HA 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 reside on z/OS in the same sysplex.

From an architectural side, using XCF adds the components shown in Figure 14-6 to a WebSphere Application Server environment.
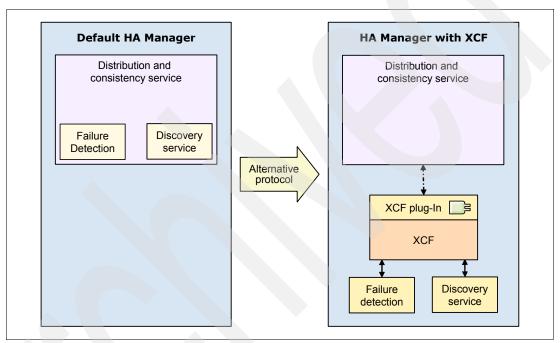
Figure 14-6   Architectural changes when using XCF support

Figure 14-6 illustrates the following main changes:

► The internal failure detection of the HA manager is factored out of the DCS structure. Instead, the XCF failure detection is used to notify the HA manager.

► The Discovery Service, which is used to communicate with the HA manager, now communicates with the XCF component of z/OS.

► XCF plugs into the Distribution and Consistency Service to perform the alive check and to disable the TCP/IP ping-based heartbeat.

14.1.10 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 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.

Figure 14-7 shows the changed flow of a request for a JSP that can be answered from the cache, assuming that IBM HTTP Server also resides on z/OS:

► Without Fast Response Cache Accelerator, a request must be processed by TCP/IP and then by IBM HTTP Server on z/OS until WebSphere Application Server can answer the request from its Dynacache.

► 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 than the same request consumed in a dynamic cache environment. These advantages come from its structure, because the FRCA cache can serve incoming TCP/IP requests directly (Figure 14-7).
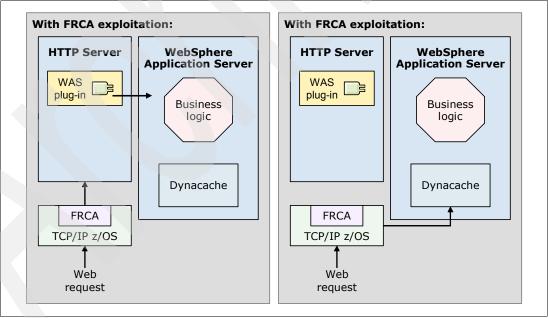
Figure 14-7   Overview of Fast Response Cache Accelerator

For more information about FRCA, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *configuring high-speed caching using FRCA with the WebSphere Application Server on z/OS*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

14.1.11 Thread Hang Recovery

Beginning with WebSphere Application Server for z/OS V7, a technique called *Thread Hang Recovery* is available. A hung thread ends in one of the following situations:

► It hangs around, only blocking threads and application environment resources, such as connections, tables, and so on.

► It ends in a loop state, blocking other resources and consuming central processor or zAAP resources. The type of processor that is used depends on whether a zAAP is available and in the step in which the application 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 perform 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. If such a transaction suddenly consumes a high amount of CPU due to an error, this error is not detected in previous versions unless the normal timeout occurs, which 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, then the servant is terminated, and WLM restarts 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 a long time.

► Respond to the client and continue working.

If protocol_http_timeout_output_recovery=SESSION is set, then 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 to happen on subsequent requests, multiple threads become tied up, and the throughput of the servant is affected, possibly to the point where it has no threads available to process work.

Current technique

If a hung thread is detected, then 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, so that 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 will try to have Java help interrupt the thread if all else fails.

> **Code to unblock a thread:** WebSphere Application Server for z/OS provides the code
> that is used to unblock a thread. To use Thread Hang Recovery for your application serving
> environment, you do not have to implement code for the Interpretable Objects registry.

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 dump (defined by new variables).

If a thread cannot be freed, although the basic options are still the same as in previous
versions of WebSphere Application Server for z/OS, the decision about whether a servant is
abended or kept alive depends on the following factors:

► The amount of processor time that is consumed by the thread (looping or just hanging)
► Whether the servant the last servant
► The number of threads that are already in a hung state, within this servant

If a thread that was reported to the controller as hung completes, the controller is notified so
that 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 controller. By default, this command provides SUMMARY information, but you can also
specify that you want DETAILS. For the REQUEST=`value` parameter, the default is DETAILS.

Use the DISPLAY command as follows:

f `server_name`,display,threads,[ALL | TIMEDOUT | REQUEST=`value` | ASID=`value` |
AGE=`value`]

Example 14-4 shows the output for this command.

Example 14-4   Display Threads command

```
F WS8601,DISPLAY,THREADS,ALL
BBOJ0111I:  REQUEST   ASID JW TO RE DISPATCH TIME
BBOJ0112I: fffff594 0X0041  N  N  N 2011/05/16 18:41:38.633862
BBO00188I END OF OUTPUT FOR COMMAND DISPLAY,THREADS,ALL
```

14.2 Functions for WebSphere Application Server for z/OS V8

This section highlights the new functions for WebSphere Application Server for z/OS V8. See
Chapter 3, "Concepts of WebSphere Application Server" on page 45, for an overview of the
general concepts, functions, and features for WebSphere Application Server.

The following additions are specific to WebSphere Application Server for z/OS V8:

► Distributed identity mapping using System Authorization Facility (SAF)
► High Performance Extensible Logging (HPEL)
► Improved reliability, availability, and serviceability (RAS) granularity for work requests
► Resource Workload Routing
► WebSphere optimized local adapter high availability support

14.2.1 High availability support for WebSphere optimized local adapter

Beginning with WebSphere Application server for z/OS V8, the WebSphere optimized local adapter participates in high availability support for WebSphere Application Server. With this support, you can specify an alternate connection factory Java Naming and Directory Interface (JNDI) name in the connection factory pool custom properties.

The WebSphere optimized local adapter is a cross-memory exchange mechanism. It was introduced with WebSphere Application Server for z/OS V7.0.0.4. It was enhanced with Information Management System (IMS) in the V7.0.0.12 fix pack. WebSphere Application Server for z/OS uses IMS to communicate with external address spaces on the same LPAR. Such spaces 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), instead of a TCP/IP stack. This service is externalized so that programs in external address spaces can access cross-memory service to communicate with Java programs (Figure 14-8).
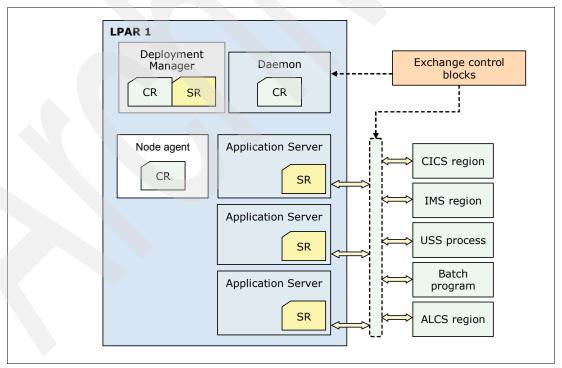

Figure 14-8   WebSphere optimized local adapter communication

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).

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 to optimize performance of calls to applications that are deployed on a locally accessible WebSphere Application Server for z/OS server.

The optimized local adapters also can 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 that extend the life of application assets that might be difficult to replace.

Planning to use WebSphere optimized local adapter for z/OS systems

When using 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 examples 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 Version 8 Information Center at the following address, and search for *optimized local adapters on WebSphere Application Server for z/OS*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

Enabling high availability support for the WebSphere optimized local adapter

To use the optimized local adapters high availability support, follow these steps:

1. Add a custom property, alternateResourceJNDIName, on the connection pool for a particular data source or connection factory:

   a. In the administrative console, navigate to the data source or connection factory. Click the **Connection pool properties** link.

   b. In the Pool Properties panel, click the **Connection pool custom properties** link.

   c. In the Custom properties panel (Figure 14-9), for the resource connection pool, click **New** to create the custom properties. Specify the JNDI name of the alternate resource connection factory.
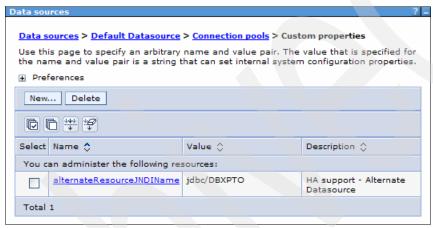
Figure 14-9   Defining an alternate resource

2. Ensure that the external address space that is associated with the alternate connection factory registration name is available for the failover to occur.

3. After adding the alternateResourceJNDIName or any other connection pool properties, restart the server for the properties to take effect.

> **Important**: For the high availability support to function properly, the resource register name must be specified on the managed connection using the connection factory Register Name attribute for both the primary and alternate resource connection factory. For more information, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *connection factory considerations for optimized local adapters*:
>
> http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

The optimized local adapter resource failover process is triggered when an application makes a `getconnection()` request for a resource that fails because the target registration is not available. During this process, the alternate JNDI resource name is used for the `getconnection()` instead as shown in Figure 14-10.
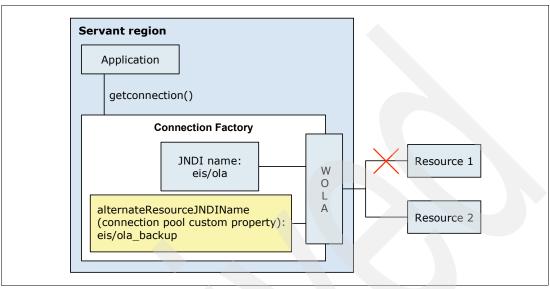
Figure 14-10   WebSphere optimized local adapter failover process

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.

14.2.2 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.

To configure an alternate resource, see "Enabling high availability support for the WebSphere optimized local adapter" on page 406.

WebSphere Application Server for z/OS V8 introduces three new configurable actions, called *action notification*. When an action notification is configured for a particular resource and requests to that particular resource fail past a specified or default threshold value, the WebSphere Application Server for z/OS run time receives notification that a particular action must be performed. The action notification codes are defined in the `failureNotificationActionCode` property.

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 controller region that 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 or PAUSING LISTENERS

When the resource is available again, a BBOJ0131I message is issued to a hardcopy in the controller region, indicating that the resource is again available. BBOJ0131I 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 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 resides for which this action was configured. Server listeners are paused when the resource is deemed unavailable. When combined with a front-end router that supports high availability, such as 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 controller region when the resource is deemed unavailable.

When the resource is available again, a BBOJ0131I message is issued to a hardcopy in the controller region, and the server listeners resume to restore 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 controller 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 remains started 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, or STOPPING APPLICATIONS THAT USE THIS RESOURCE

When the resource is available again, a BBOJ0131I message is issued to a hardcopy in the controller 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, or STARTING APPLICATIONS THAT USE THIS RESOURCE

– The reason the action was taken:

- Normal servant region availability notification
- Unknown resource availability

Figure 14-11 shows a representation of the resource workload routing.
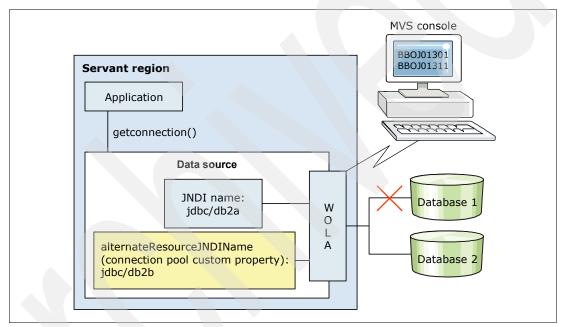
Figure 14-11   Resource workload routing

All properties for this feature must be created as new custom properties on the connection pool for a particular data source or connection factory. To create the properties as new custom properties on the connection pool, follow these steps:

1. In the administrative console, navigate to the data source or connection factory for which the notification is to be enabled. Click the **Connection pool properties** link.

2. In the Pool Properties panel, click the **Connection pool custom properties** link.

3. In the Custom properties panel for the resource connection pool, click **New** to create the following custom properties:

– failureNotificationActionCode

This property is used to enable the notification feature. If this property is not set to one of the valid integer values specified (1, 2, or 3), the notification feature is disabled.

– failureThreshold

This property indicates the number of consecutive resource exceptions that must occur for a particular resource before a notification is sent or failover occurs.

– alternateResourceJNDIName

This property provides the JNDI name of the alternate resource to enable the failover feature.

– resourceAvailabilityTestRetryInterval

This property indicates the time, in seconds, that the test connection thread attempts to test the primary resource.

– enablePartialResourceAdapterFailoverSupport

If value of this property is `true`, failover to the alternate resource occurs, but fail back to the primary is manual.

– disableResourceFailOver

If value of this property is `true`, automatic failover does not occur.

– disableResourceFailBack

If value of this property is `true`, automatic failback does not occur.

– populateAlternateResource

If value of this property is `true`, the alternate resource is populated with connections to maximum connections.

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. For details, see 14.1.7, "Workload management for WebSphere Application Server for z/OS" on page 392.

Before WebSphere Application Server for z/OS V8, 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. This single set of RAS attribute values applies 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 to 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. You can define the dispatch timeout for IIOP requests by using the `control_region_wlm_dispatch_timeout` property and for HTTP requests by using the `protocol_http_timeout_output` property.

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, one for each workload classification element in the workload classification file. The application server classifies requests based on the workload classification elements that are defined in the workload classification file. The application server then applies the set of RAS attribute values that are

defined for a workload classification element to the requests that are classified under that workload classification element.

RAS attributes

You can specify the following RAS attributes on the classification entry element:

dispatch_timeout
Specifies the amount of time that a control region waits after dispatching a request to the WLM queue. The request is a request that the classification element has classified.

queue_timeout_percent
Specifies the WLM queue timeout as a percentage of the dispatch timeout. The range for the percentage is 0-99%.

request_timeout
Specifies the timeout value, in seconds, that is applied to outbound requests that originate under dispatched requests.

stalled_thread_dump_action
Specifies the dump action that the server takes when requests exceed their dispatch timeout, which is specified on the dispatch_timeout attribute.

cputimeused_limit
Specifies the CPU timeout in milliseconds. The CPU timeout is the amount of CPU time that is allowed for the request before the request times out.

cputimeused_dump_action
Specifies the dump action that the server takes when requests exceed their CPU timeout specified on the cputimeused_limit attribute.

dpm_interval
Specifies in seconds the Dispatch Progress Monitor (DPM) interval.

dpm_dump_action
Specifies the DPM dump action that is taken at every DPM interval for requests that the classification element classifies.

SMF_request_activity_enabled
Specifies whether System Management Facilities (SMF) 120 subtype 9 records are collected for requests that the classification element classifies.

SMF_request_activity_timestamps
Specifies whether to format timestamps in human-readable form on the SMF 120 subtype 9 record for requests.

SMF_request_activity_security
Specifies whether the security data section of the SMF 120 subtype 9 record is collected for requests.

SMF_request_activity_CPU_detail
Specifies whether the CPU usage breakdown section of the SMF 120 subtype 9 record is collected for requests.

classification_only_trace
Specifies whether to generate trace records for requests that the classification element classifies.

message_tag
Specifies a string token that prints with all the trace records, log messages, and system messages generated for requests that the classification element classifies.

Each element (SIP, HTTP, IIOP, and on) can contain one or more of these properties as needed to classify the work for a message.

Enabling request-level RAS granularity

You can enable request-level RAS granularity for HTTP, IIOP, optimized local adapter, and certain MDB requests by defining RAS attributes in the workload classification document. With request-level RAS granularity, you can specify RAS attribute values for specific requests, such as a unique dispatch timeout value for all HTTP requests with a URI that ends in the `.jpg` extension.

To implement request-level RAS granularity, choose one of the following options:

► If the workload classification file is defined on your server:

    a. Implement the changes to the file.

    b. Reload the file. Choose one of the following options:

       • Restart the application server.

       • Reload the workload classification document by issuing the following command:

         `MODIFY|F servername,RECLASSIFY,FILE='/path/file_name.xml'`

► If the workload classification file is not defined on your server, follow the instructions in "Workload classification document" on page 395.

Sample workload classification document with RAS attributes

The workload classification document shown in Example 14-5 contains the following elements:

► The IIOP classification element defines attributes for a specific module, component, and method of the EJBApp3 application. The module is `MyEJB3.jar`, the component is `MyEJB3Class`, and the method is `method3`. The `transaction_class`, `dispatch_timeout` `queue_timeout_percent`, `SMF_request_activity_enabled`, and `SMF_request_activity_timestamps` attributes are all defined for this specific method in the EJBApp3 application.

► The HTTP classification element defines a transaction_class, HTC8080, for all HTTP requests that are received on the `my.server.com` host and 8080 port. The classification element also defines the RAS attributes `dispatch_timeout`, `queue_timeout_percent`, `timeout_recovery`, and `stalled_thread_dump_action`.

Example 14-5 Workload classification document classification.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Classification SYSTEM "Classification.dtd" >
<Classification schema_version="1.0">
<!-- IIOP Classification Rules -->
<InboundClassification
  type="iiop"
  schema_version="1.0"
  default_transaction_class="TC">
<iiop_classification_info
    application_name="EJBApp3"
    module_name="MyEJB3.jar"
    component_name="MyEJB3Class"
    method_name="method3"
    transaction_class="TC3"
    dispatch_timeout="40"
    queue_timeout_percent="90"
    SMF_request_activity_enabled="1"
    SMF_request_activity_timestamps="1"
  />
```

```
</InboundClassification>
<!-- Internal Classification Rules -->
<InboundClassification
  type="internal"
  schema_version="1.0"
  default_transaction_class="internal" />
<!-- HTTP Classification Rules -->
<InboundClassification
  type="http"
  schema_version="1.0"
  default_transaction_class="HTC">
<http_classification_info
    host="my.server.com"
    port="8080"
    transaction_class="HTC8080"
    dispatch_timeout="100"
    queue_timeout_percent="98"
    timeout_recovery="session"
    stalled_thread_dump_action="javacore">
  </http_classification_info>
</InboundClassification>
</Classification>
```

The DTD shown in Example 14-6 defines the elements and attributes used in Example 14-5 on page 412.

Example 14-6   Workload classification document classification.dtd file

```
<?xml version='1.0' encoding="UTF-8"?>
<!ELEMENT Classification (InboundClassification)+>
<!ATTLIST Classification schema_version CDATA #REQUIRED>
<!ELEMENT InboundClassification
((iiop_classification_info*|http_classification_info*))>
<!ATTLIST InboundClassification type (iiop|http) #REQUIRED>
<!ATTLIST InboundClassification default_transaction_class CDATA #REQUIRED>
<!ATTLIST InboundClassification schema_version CDATA #REQUIRED>
<!ELEMENT iiop_classification_info (iiop_classification_info*)>
<!-- inputs -->
<!ATTLIST iiop_classification_info activity_workload_classification CDATA
#IMPLIED>
<!ATTLIST iiop_classification_info application_name CDATA #IMPLIED>
<!ATTLIST iiop_classification_info component_name CDATA #IMPLIED>
<!ATTLIST iiop_classification_info description CDATA #IMPLIED>
<!ATTLIST iiop_classification_info method_name CDATA #IMPLIED>
<!ATTLIST iiop_classification_info module_name CDATA #IMPLIED>
<!-- outputs -->
<!ATTLIST iiop_classification_info transaction_class CDATA #IMPLIED>
<!ATTLIST iiop_classification_info dispatch_timeout CDATA #IMPLIED>
<!ATTLIST iiop_classification_info queue_timeout_percent CDATA #IMPLIED>
<!ATTLIST iiop_classification_info request_timeout CDATA #IMPLIED>
<!ATTLIST iiop_classification_info stalled_thread_dump_action
(none|svcdump|javacore|heapdump|traceback|javatdump) #IMPLIED>
<!ATTLIST iiop_classification_info SMF_request_activity_enabled (0|1) #IMPLIED>
<!ATTLIST iiop_classification_info SMF_request_activity_timestamps (0|1) #IMPLIED>
<!ELEMENT http_classification_info (http_classification_info*)>
```

```
<!-- inputs -->
<!ATTLIST http_classification_info host CDATA #IMPLIED>
<!ATTLIST http_classification_info port CDATA #IMPLIED>
<!ATTLIST http_classification_info uri CDATA #IMPLIED>
<!ATTLIST http_classification_info description CDATA #IMPLIED>
<!ATTLIST http_classification_info transaction_class CDATA #IMPLIED>
<!-- outputs -->
<!ATTLIST http_classification_info dispatch_timeout CDATA #IMPLIED>
<!ATTLIST http_classification_info queue_timeout_percent CDATA #IMPLIED>
<!ATTLIST http_classification_info request_timeout CDATA #IMPLIED>
<!ATTLIST http_classification_info stalled_thread_dump_action
(none|svcdump|javacore|heapdump|traceback|javatdump) #IMPLIED>
```

To reload your new workload, stop the server, or use the **modify** command, as shown in Example 14-7.

Example 14-7   Reloading the workload classification document with RAS attributes

```
F WS8601,RECLASSIFY,FILE='/WebSphere/V8R0/wlm_classification.xml'
BBOJ0129I: The /WebSphere/V8R0/wlm_classification.xml workload
classification file was loaded at 2011/05/18 14:14:02.155 (GMT)
BBO00211I MODIFY COMMAND 945
RECLASSIFY,FILE='/WebSphere/V8R0/wlm_classification.xml' COMPLETED
SUCCESSFULLY
```

You can check the settings on the workload configuration document by using the **display** command, as shown in Example 14-8.

Example 14-8   The display command

```
F WS8601,DISPLAY,WORK,CLINFO
BBOJ0129I: The /WebSphere/V8R0/wlm_classification.xml workload
classification file was loaded at 2011/05/18 14:14:02.155 (GMT)
BBO00281I CLASSIFICATION COUNTERS FOR IIOP WORK
BBO00282I CHECKED 0, MATCHED 0, USED 0, COST 2, DESC: IIOP root
BBO00282I CHECKED 0, MATCHED 0, USED 0, COST 2, DESC: IIOP root
BBO00283I FOR IIOP WORK: TOTAL CLASSIFIED 0, WEIGHTED TOTAL COST 0
BBO00281I CLASSIFICATION COUNTERS FOR HTTP WORK
BBO00282I CHECKED 0, MATCHED 0, USED 0, COST 2, DESC: HTTP root
BBO00282I CHECKED 0, MATCHED 0, USED 0, COST 2, DESC: HTTP root
BBO00283I FOR HTTP WORK: TOTAL CLASSIFIED 0, WEIGHTED TOTAL COST 0
BBO00281I CLASSIFICATION COUNTERS FOR INTERNAL WORK
BBO00282I CHECKED 1, MATCHED 1, USED 1, COST 1, DESC: Internal root
BBO00283I FOR INTERNAL WORK: TOTAL CLASSIFIED 1, WEIGHTED TOTAL COST 1
BBO00188I END OF OUTPUT FOR COMMAND DISPLAY,WORK,CLINFO
```

14.2.3 High Performance Extensible Logging

HPEL provides a convenient mechanism for storing and accessing log, trace, System.err, and System.out information that is 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 that use Job Entry Subsystem (JES), LogStreams, Component Trace, 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 trace. 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 9.7.1, "Log and traces" on page 291.

14.2.4 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 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. There is no 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 in 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 14-12.
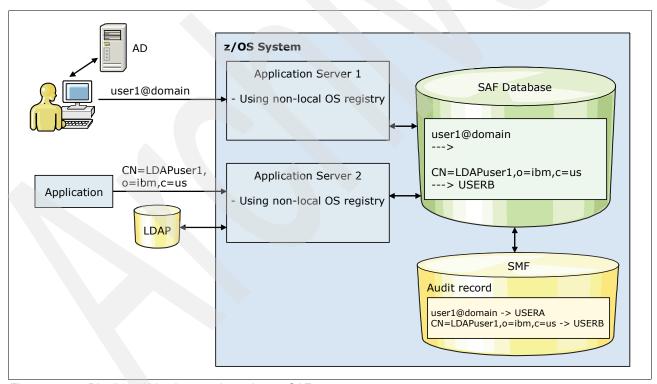

Figure 14-12   Distributed identity mapping using an SAF

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.

Feature support: The SAF distributed identity mapping feature is not supported in a mixed-version cell (nodes before WebSphere Application Server V8).

Before you begin

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 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.

To delete the JAAS login module on the administrative console, complete these steps:

a. Click **Security** → **Global security** → **Java Authentication and Authorization Service** → **System logins**.

b. Click **DEFAULT**.

c. Select the **com.ibm.ws.security.common.auth.module.MapPlatformSubject** login JAAS module, and then click **Delete**.

d. Click **OK**.

e. Repeat these steps for the system logins of WEB_INBOUND, RMI_INBOUND, and SWAM_ZOSMAPPING (instead of DEFAULT).

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 WebSphere console.

► Scenario 2

You have a local OS registry configured on z/OS operating systems with the Kerberos or SPNEGO authentication mechanism. In this case, you might want to map a Kerberos user to an SAF.

To enable distributed identity mapping for this scenario, in the administrative console, click **Security** → **Global security** → **Kerberos configuration**, and select the **Use the RACMAP profiles in the SAF product for distributed identity mapping** option.

► 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.

To enable distributed identity mapping for this scenario:

a. In the administrative console, click **Security** → **Global security**.

b. For Authentication cache setting, expand **RMI/IIOP security**, and select **CSIv2 inbound communications**.

c. For CSIv2 Attribute layer, select the **Map certificate and DN using SAF distributed identity mapping** option.

► 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.

To enable distributed identity mapping for this scenario:

a. In the administrative console, click **Security** → **Global security**.

b. For Authentication cache setting, expand **RMI/IIOP security**, and select **CSIv2 inbound communications**.

c. For CSIv2 Transport layer, select the **Map certificate using SAF distributed identity mapping** option.

The RACMAP command: Use the RACMAP command 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 Version 8 Information Center at the following address, and search for *using distributed identity mapping for SAF*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

---

14.3 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. For detailed installation instructions, including checklists for z/OS, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *install an application serving environment*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

14.3.1 Installation overview

Installing a WebSphere Application Server for z/OS V8.0 on z/OS entails the following steps:

1. Prepare the system.

2. Install the product code with IBM System Modification Program/Extended (SMP/E).

3. Configure the product and create the profile using job control language (JCL).

Data sets: 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. To switch a configuration from using load modules in the product file system to using load modules in a data set, use the switchModules.sh tool. For more information, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *switchModules command*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

---

14.3.2 Installation considerations

This section includes general considerations when installing a WebSphere Application Server for z/OS V8.

General environment considerations

Planning your environment is critical. For more information, see Chapter 4, "Infrastructure" on page 81.

Naming convention

When installing WebSphere Application Server for z/OS V8, 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 components are introduced in WebSphere Application Server for z/OS V8 (such as the administrative agent and job manager). Ensure that your naming convention also reflects this information.

> **Naming convention guidelines from the Washington System Center:** 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 421.

Real memory defined

WebSphere Application Server for z/OS has a different blueprint than WebSphere Application Server for distributed environments. For more information, see 14.1.5, "Structure of an application server" on page 389. Multiple heaps, one for every controller 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 each 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.

For an example, see the environment from Figure 14-3 on page 391, which includes a deployment manager and one application server (with three servants). Assume that each control region heap has a maximum of 256 MB and that each servant region has a maximum of 512 MB defined. Based on this information, the real storage that WebSphere Application Server for z/OS needs is 2.5 GB. This value does not consider that other middleware might be installed on the z/OS image.

Ensure that the LPAR that is used for the installation has enough real storage defined. Also add storage for the operating system and other applications running in this LPAR, such as DB2, CICS, and so on.

> **Health check:** Monitor your system and check for swapping. Swapping can have an 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, as explained in 14.1.5, "Structure of an application server" on page 389. This heap size is of interest when migrating an application from another platform to WebSphere Application Server for z/OS V8.

Often, the memory size from the distributed environment is carried on from the distributed environment and reused for the controller 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 if it runs, it takes up more time, reducing the general throughput.

> **Important:** If you migrate an application to WebSphere Application Server for z/OS V8 from another operating system family, perform a verbose garbage collection analysis. With a verbose garbage collection, you can size the heap to a minimum and maximum value 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 to the monthly z/OS software bill. Because WebSphere Application Server is mainly written in Java, it can use zAAP. Most environments see 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.

> **RMF:** If you currently do not have zAAP physically installed, you can use the RMF to identify the amount of CPU seconds that can be run on zAAP. For detailed 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 regarding the file system, regardless of whether an HFS or IBM System z 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, from a performance perspective, create dedicated file systems for each node.

► Product file system mounted read only

A read only mount improves performance and prevents the change of file system contents.

14.3.3 Function modification identifiers

Table 14-1 lists the function modification identifiers (FMIDs) for WebSphere Application Server for z/OS.

Table 14-1   FMIDs for WebSphere Application Server for z/OS V8

FMID	CompID	Component name
HBBO800	5655I3500	WebSphere Application Server for z/OS V8
HBBO800	5655N0212	DMZ Secure Proxy Server V8
HBBO800	5655I3511	Web server plug-ins V8
HBBO800	5655I3510	IBM HTTP Server V8

Table 14-2 contains the upgrade and subset values for WebSphere Application Server for z/OS V8 and IBM Installation Manager.

Table 14-2   Preventive Service Planning upgrade and subset ID

Upgrade	Subset	Description
WASAS800	HBBO800	WebSphere Application Server for z/OS V8
IIMZOSV1	HGIN140	Installation Manager for z/OS V1.4

14.3.4 SMP/E installation

The product code for WebSphere Application Server V8 is brought to your system by using the SMP/E function of z/OS.

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 detailed system requirements at:

http://www.ibm.com/support/docview.wss?rs=180&uid=swg27006921

14.3.5 Customization

After the SMP/E installation is complete, you configure the product. Figure 14-13 illustrates the configuration process. The configuration includes creating the server profile and executing it on the host by using the following tools:

► The configuration worksheet
► The WebSphere Customization Toolbox (graphical or interactive)
► The `zpmt.sh` z/OS script (command line, batch style, or silent)

This section briefly describes these tools.
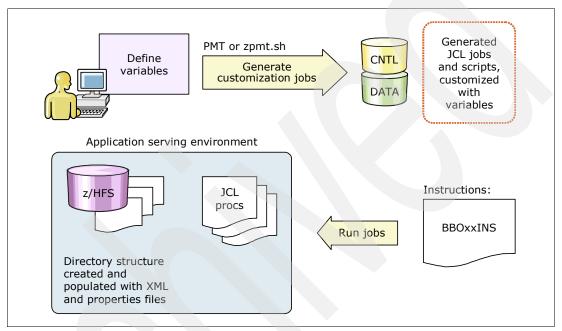
Figure 14-13 Configuration overview of WebSphere Application Server for z/OS

Further resources: For detailed information about the installation of WebSphere Application Server for z/OS V8, see the WebSphere Application Server Version 8 Information Center, and search for *installing your application serving environment*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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 Version 8 Information Center (at the previous web address), and search for *creating customization definitions*:

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

Washington System Center: The Washington System Center created a version of the planning spreadsheet called *WebSphere for z/OS Version 7 - Configuration Planning Spreadsheet* (Reference #PRS3341). You can use this spreadsheet during the customization process. To obtain the spreadsheet, go to WebSphere for z/OS Version 7 - Configuration Planning Spreadsheets at:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS3341

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 can use a response file to generate the actual JCL that creates the profiles (Figure 14-14).

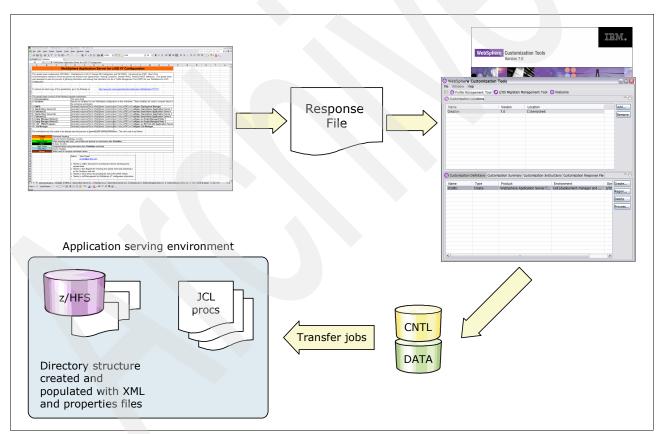*Figure 14-14   Using a planning spreadsheet for WebSphere Application Server for z/OS*

WebSphere Customization Toolbox
After the product code is brought to the system, configure WebSphere Application Server. In WebSphere Application Server V8, you must use the WebSphere customization tools for this configuration. WebSphere Customization Toolbox can be installed through IBM Installation Manager.

For more information about installing the WebSphere Customization Toolbox and using the Installation Manager, see 15.6.4, "z/OS Migration Management Tool" on page 448.

> **ISPF panel:** The Interactive System Productivity Facility (ISPF) panel configuration is no longer available in WebSphere Application Server for z/OS V8.

The following set of tools is available for Windows and Linux based workstations:

► Profile Management Tool (z/OS only), for creating profiles
► z/OS Migration Management Tool, for migrating nodes

For detailed information about how to install and use these tools, see the IBM Education Assistant Information Center at:

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm. iea.was_v8/was.8.0

The zpmt.sh script

The `zpmt.sh` script is the silent implementation of the Profile Manager Tool on the z/OS host. You use a command-line call to the script, including various parameters. It then creates the `.CNTL` and `.DATA` members that correspond to the response file and that 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 14-15 shows an overview of this script.
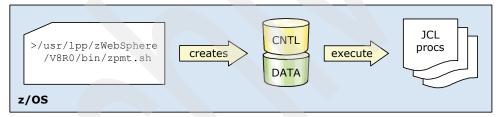
Figure 14-15   Overview of zpmt.sh configuration script

You can find the script in the `/usr/lpp/zWebSphere/V8R0/bin` default WebSphere Application Server for z/OS product directory. By running the script, you open the OSGi command shell.

> **OSGi command shell:** 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.

> **Delete the profilePath directory:** If you have to rerun a script for a profile, delete the `profilePath` directory. Otherwise, you receive the following message:
>
> `The profile path is not valid.`

Running customization jobs

The second step in the customization is running the JCL that was created by using one of the techniques that are described in 14.3.5, "Customization" on page 421.

Table 14-3 shows the jobs that are necessary for the customization of WebSphere Application Server for z/OS V8. The jobs are similar to V6.1, but with a few notable differences.

> **Important:** Read the BBOxxINS module on 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 execute.

Table 14-3   Installation jobs for WebSphere Application Server for z/OS V8

Job name	Description
BBOxxINS	Instruction member that contains the installation steps.
BBOSBRAK	RACF scripts that are created and executed in this JCL. No more "BRAJ" to create a script for "BRAK" to execute.
BBOSBRAM	Creates /home directories for WebSphere users in the OMVS part and set ownership.
BBOxBRAK	RACF scripts that are created and executed in this JCL. No more "BRAJ" to create a script for "BRAK" to execute.
BBOxCFS	Sets up the file system (whether HFS or zFS).
BBOxPROC	Copies the tailored start procedures to the cataloged procedure library.
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.

14.4 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 environment.

14.4.1 WebSphere Application Server settings

This section addresses the following settings:

- ► Intelligent runtime provisioning
- ► Workload profile setting
- ► Addressing mode

Intelligent runtime provisioning

The new intelligent runtime provisioning function is disabled by default. You might want to enable it in the Integrated Solutions Console to reduce startup time and resource consumption. You can see improvements in the startup time of up to 10–15%. For more information about intelligent runtime provisioning, see 3.2.8, "Intelligent runtime provisioning" on page 64.

Workload profile setting

WebSphere Application Server for z/OS V8 introduces a value for the workload profile setting 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.

To change the value through the Integrated Solutions Console, click **Servers → Server Types → WebSphere application servers → *your_server* → Container services → ORB service → z/OS additional settings**.

Table 14-4 lists all possible values.

Table 14-4   Workload profile settings for the z/OS ORB service

Value	# Threads	Description
ISOLATE	1	This setting specifies that the servants are restricted to a single application thread. Use ISOLATE to ensure that concurrently dispatched applications do not run in the same servant. Two requests processed in the same servant can cause one request to corrupt another.
IOBOUND	MIN(30, MAX(5,(Number of CPUs*3)))	This setting specifies more threads in applications that perform I/O-intensive processing on the z/OS operating system. The calculation of the thread number is based on the number of processors. IOBOUND is used by most applications that have a balance of processor-intensive and remote operation calls. A gateway and protocol converter are two examples of applications that use the IOBOUND profile.
CPUBOUND	MAX((Number of CPUs-1),3)	This setting specifies that the application performs processor-intensive operations on the z/OS operating system, and therefore might not benefit from more threads than the number of processors. The calculation of the thread number is based on the number of processors. Use the CPUBOUND profile setting in processor-intensive applications, such as XML parsing and XML document construction, where the most the application response time is spent using the processor.
LONGWAIT	40	This setting specifies more threads than IOBOUND for application processing. LONGWAIT spends most of its time waiting for network or remote operations to complete. Use this setting when the application makes frequent calls to another application system, such as CICS screen-scraper applications, but does not do much of its own processing.
CUSTOM	User defined	This setting specifies that the number of servant application threads is determined by the value that is specified for the `servant_region_custom_thread_count` server custom property. The minimum number of application threads that can be defined for this custom property is 1; the maximum number of application threads that can be specified is 100.

Addressing mode

The addressing mode (AMODE) is a JCL parameter, introduced with V6.1, that 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. Do not modify the default value. In the generated procedures during the installation, the default value is 00. This value means that the value defined in the XML files of the application server is used for the decision 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 (or the other way around), 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.

14.4.2 Java settings

The settings that described in this section are JVM settings that cannot be directly modified by the WebSphere Application Server V8. However, servers will use these underlying techniques.

For more information about the topics in this section, see the Diagnostics Guide for the JVM V6 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 use shared class cache. 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 the 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 destroyed, 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

Memory pages: The JVM uses these memory pages for the shared classes cache. If you request large cache sizes, you might have to increase the amount of shared memory pages that is 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 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 further information about performance implications and using these parameters, see the IBM publications *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 uses the IBM Java Standard Edition V6. This JVM implementation offers the shared class cache that allows multiple JVMs to access the same classes, both application and system classes, without loading them multiple times into memory.

The IBM implementation for distributed platforms (AIX, Linux, and Windows systems) offers the option to write the content to a file system so that it can survive an operating system restart. However, z/OS supports only the use of non-persistent cache.

Compressed references

The use of compressed references improves the performance of many applications because objects are smaller, resulting 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 if it is appropriate.

When using compressed references, the following structures are allocated in the lower area of the address space:

► Classes
► Threads
► Monitors

Because JVM technique is separated from the WebSphere Application Server product, you can activate it only by using a JVM argument on a JVM level. Thus, on the z/OS platform, the activation needs to be performed for all components of an application server that have a heap (adjunct, control, and servant region).

In the Integrated Solutions Console, navigate to **Server → Server Types → WebSphere application servers → *your_server* → Server Infrastructure → Java and Process Management → Process definition → server_component (Adjunct | Control | Servant) → Java Virtual Machine**.

Then add the following statement to the generic JVM arguments:

```
–Xcompressedrefs
```

As always, when changing some JVM settings, restart the server after saving and synchronizing the modifications to activate them.

14.4.3 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 V8.0.

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 controllers do some of the processing that is required to receive work into the system, manage the HTTP transport handler, classify the work, and perform other housekeeping tasks.

► Do not set the servant classification higher in the service class hierarchy than more important work, such as the controller and CICS or IMS transaction servers. Use a high velocity goal.

► Classify enclaves for WebSphere Application Server for z/OS using the Subsystem CB. The performance goals that you set here depend on your application 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 executed during server start. Therefore, if these scripts are not classified in the WLM, the server start time will increase.

> **Health check for WLM classification for OMVS components**: A step in the control region start procedure invokes 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 (as shown in Example 14-9).

Example 14-9   Service class definition for OMVS components

```
Subsystem Type . : OMVS    Fold qualifier names?   Y  (Y or N)
Description  . . . OMVS subsystem rules mod for WAS

Action codes: A=After    C=Copy        M=Move     I=Insert rule
              B=Before  D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                       More ===>
        --------Qualifier--------          -------Class--------
Action   Type   Name    Start           Service    Report
                                DEFAULTS: OMVS____   OMVSREP
____   1  TN    FTPSERVE ___             EBIZ_HI    FTPSERV
____   1  UI    OMVSKERN ___             SYSSTC     _____
____   1  TN    WSSRV*   ___             EBIZ_HI    WAS70
```

For information about how to set WLM service class classifications, see *System Programmer's Guide to: Workload Manager*, SG24-6472.

For additional information, see the WebSphere Application Server Version8 Information Center at the following address, and search for *controller and servant WLM classifications*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

14.4.4 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, including IDs that are associated with cross-process services (such as TCP/IP), which could not be reused in earlier releases of z/OS. 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.

If the operating system runs with the ASID reuse option disabled, you can run the `updateZOSStartArgs` script, in the `profile_root/bin` directory of each profile, to enable the ASID capability for a specific WebSphere Application Server for z/OS profile. Because the script is run on a profile base, you must run it multiple times when multiple profiles or a Network Deployment installation is used. 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 Version 8 Information Center at the following address, and search for *enabling or disabling the reusable ASID function*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

14.4.5 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 Version 8 Information Center (at the web address in the previous section), and search for *deprecated features*:

14.4.6 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. Administrative scripts that use Jacl do not need to be migrated to Jython. However, this stabilized status might change in future versions of WebSphere Application Server.

14.4.7 Application profiling

With application profiling, you can analyze the application during run time. It graphically provides detailed information about which application step uses how much processor. You can identify critical points inside the application. Although it is intended for developers, system programmers should encourage the development team to use such a profiling technique. Application profiling opens the black box application on the host.

A profile tool for z/OS is *JinsightLive for z/OS*. With this tool, you can analyze 31- and 64-bit JVM. 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/

> **Application profiling:** Application profiling usually requires some level of experience with the tooling. After you get used to the technique, application profiling is a powerful way of identifying CPU-intensive points in an application. Many the critical points require only few changes in the application itself.
>
> As a starting point, ask your local IBM representative for assistance.

14.5 Planning checklist

Consider the following items as you plan for WebSphere Application Server for z/OS V8.

► Because 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 administrative agent WebSphere Application Server V8 components. (The z/OS Profile Management Tool uses the recommendations from the z/OS customization worksheet.)

► Test the usage of XCF support for the HA 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 keep track of configuration changes. The use of this no-charge tool is preferred.

Garbage Collection and Memory Visualizer
To analyze verbose garbage collection information and to 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 should 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 performed a verbose garbage collection analysis to identify and verify the heap size.

14.6 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 thoughts are still valid for V8.

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 Version 8 Information Center; search for *installing the WebSphere Customization Toolbox using the GUI*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

► *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

For deeper insight into the Java options and functions used by WebSphere Application Server V8, 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 insight on 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 a straight forward 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/

15

Migration

This chapter addresses migration considerations for moving to WebSphere Application Server V8. This chapter includes the following sections:

► Migration features in WebSphere Application Server V8
► Migration overview
► Migration plan
► Application development migration considerations
► Infrastructure migration considerations
► Migration considerations for WebSphere Application Server for z/OS

15.1 Migration features in WebSphere Application Server V8

WebSphere Application Server V8 provides new features that improve the migration process. This section highlights those new features.

15.1.1 Configuration Migration Management Tool

The new version of the Eclipse-based graphical wizard, called *Configuration Migration Management Tool*, improves migration management. This tool supports the migration of the two additional management profiles of WebSphere Application Server V7: Admin Agent and Job Manager. The tool highlights the potential changes due to the migration. It can also generate the commands that are executed by the graphical wizard to create migration scripts.

15.1.2 Cross platform migrations

With WebSphere Application Server V8, you can also migrate a node from a machine to another machine, even if the other machine has a different operating system (except for IBM i and z/OS). To help you with this migration, WebSphere Application Server V8 provides the `createRemoteMigrJar` tool. This tool creates a compressed (`.zip`) file from the WebSphere V8 binary files that contain the necessary files to run the migration backup command in a machine that does not have WebSphere Application Server V8 installed.

15.1.3 Enhanced z/OS Migration Management Tool

WebSphere Application Server V8 enhances the z/OS Migration Management Tool. This tool now supports the migration of two additional flexible management profiles of WebSphere Application Server V7: Admin Agent and Job Manager. It also supports 64-bit migration on z/OS.

15.2 Migration overview

Migration is the action of moving from an existing release to a newer release. When doing a WebSphere Application Server migration, consider it a complete project because it is more than just applying a new product version.

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, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *removed features*, *deprecated features*, and *new features*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

By reviewing this information, you will have a better understanding of the areas impacted by migration.

15.3 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, beginning 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, tooling, and the development standards in WebSphere Application Server V8.

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.
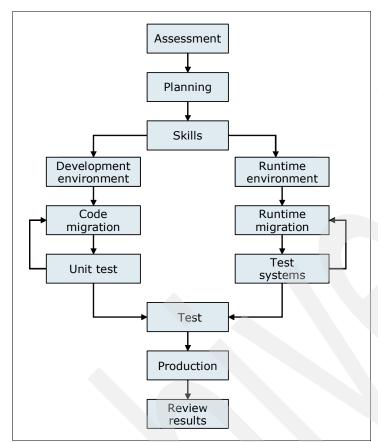
Figure 15-1 illustrates the steps that you might take in performing a migration.

Figure 15-1 Migration path

> **Additional resource:** For more information and updated 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

15.4 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 supports Java Platform, Enterprise Edition 6 (Java EE 6).

Consider the following points:

► Although the newer J2EE version intends to support older versions, some minor exceptions might exist.

► Identify the deprecated APIs and determine whether any of these APIs are used in your existing applications.

► Understand the new WebSphere Application Server V8 features.

For more information about deprecated APIs, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *deprecated features* and *migrating API and specifications*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

See also the deprecated API list for the Java platform at:

http://download.oracle.com/javase/6/docs/api/deprecated-list.html

You can also find information about how to migrate specific application components as web services, EJB, OSGi, or asynchronous beans in the WebSphere Application Server Version 8 Information Center (see previous reference) by searching for *migrating, coexisting, and interoperating*.

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.

15.5 Infrastructure migration considerations

This section addresses topics to consider when migrating from an existing environment to WebSphere Application Server V8.

15.5.1 Coexistence

WebSphere Application Server V8 can be installed and configured to coexist with other WebSphere Application Server V7 and V6.x 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.

15.5.2 Interoperability

WebSphere Application Server V8 is generally interoperable with WebSphere Application Server V7 and V6.x. This interoperability means that different versions of WebSphere Application Server can exchange data and communicate.

Some requirements can exist about the interoperability of some functionality, depending on the WebSphere version. For more information, see the WebSphere Application Server Version 8 Information Center (see previous reference) by searching for *migrating, coexisting, and interoperating*.

15.5.3 Mixed-cell support

To ease the incremental upgrade of your environment, WebSphere Application Server V8 supports mixed-cells with nodes from V7 and at least V6.0.2. A cell can contain nodes from different versions of WebSphere Application Server and from different platforms. The version of your deployment manager has to be at the highest or at the same level as your federated nodes.

However, running in mixed-cell configuration is supported. This situation can be considered to be transitional, due to a migration. In the end, all your nodes must be at the same level.

Mixed-node management is not possible for nodes at V6.0.0.x and V6.0.1. You have to upgrade the nodes to at least V6.0.2 before the migration.

15.5.4 Configuration Migration Tools

WebSphere Application Server V8 provides Configuration Migration Tools to perform a migration.

With Configuration Migration Tools, you can perform the following tasks:

► Migrate configurations including the topology, customizations, and applications, keeping your old environment running. The tools support the migration of V7 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 Job Manager and Administrative Agent profiles.

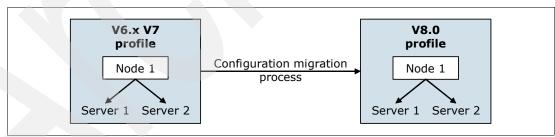Figure 15-2 illustrates the migration process.

Figure 15-2   Migration process

The Configuration Migration Tools are available on the following platforms with the features indicated as follows:

► Distributed

– Configuration Migration Management Tool

This tool provides a graphical interface used to perform all the migration steps (Figure 15-3). The tool is based on the migration commands (after the figure).
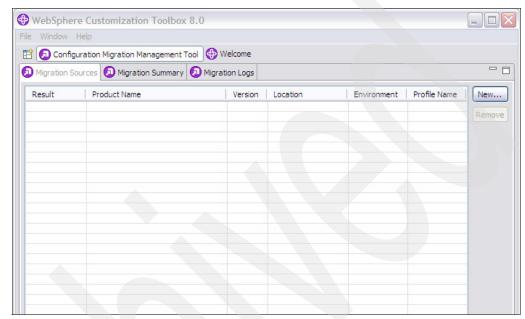

Figure 15-3   Configuration Migration Management Tool

– createRemoteMigrJar tool

This tool creates a compressed (.zip) file from the WebSphere V8 binary files. The compressed file contains the necessary files for running the migration backup command in a remote machine that does not have WebSphere Application Server V8 installed. You use this tool when you want to perform a migration from one machine to another machine. The .zip file that is created contains specific code, making it is 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, or 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 Version 8 Information Center at the following address, and search for the command name:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

► z/OS

z/OS Migration Management Tool

For more information, see the WebSphere Application Server Version 8 Information Center at the previous address, and search for *z/OS Migration Management Tool*.

15.5.5 Properties files

Properties files are a group of scripting commands that are available with `wsadmin`. These commands offer the ability to export and import portions of application server profiles configuration. Almost every WebSphere Application Server settings can be exported.

For more information about the properties file based configuration, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *using properties files to manage system configuration*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

15.5.6 Product configuration migration scenarios

This section describes the different 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:

► WebSphere Application Server Version 8 Information Center (search for *migrating, coexisting, and interoperating*)

   http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

► *WebSphere Application Server V7 Migration Guide*, REDP-4635

> **Important:** The scenarios in this chapter are available only for Distributed platforms and IBM i. For z/OS migration, see 15.6, "Migration considerations for WebSphere Application Server for z/OS" on page 446.
>
> **Tips:** 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 only manage servers at the same release or earlier.
> - The Administrative Agent can only register Base Application Servers at the same release level and on the same machine.
> - The Deployment Manager can only manage nodes at the same release or earlier.

Manual

With the manual migration scenario, you start with a new WebSphere Application Server V8 environment and import all your configuration and applications. The import is ideally done by using scripts. Depending on your configuration, recreating it manually by using the administrative console can be risky, because of human error or default. Remember that you have to 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 much effort and, therefore, be expensive. These scripts must be valid for the new WebSphere version.
  - It requires every change in the environment to 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, and create a profile.
3. Import the profile configuration with the migration tools.
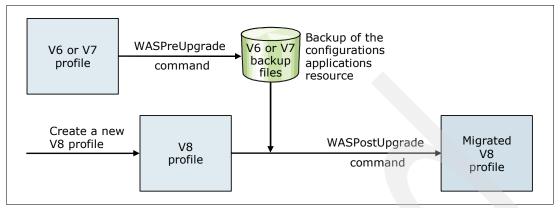
The schema in Figure 15-4 illustrates this migration.

Figure 15-4   Automated migration approach

This approach provides the following advantages and disadvantages:

► 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 of the current configuration is imported to WebSphere Application Server V8.

► Disadvantages

– All applications being 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, you can migrate by using the approach described in "Multinode environment migration with mixed-node and the Configuration Migration Tools" on page 443.

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, 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 with the deployment manager profile in Version 8 and the nodes 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 disadvantages:

- ► 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 of the current configuration is imported to WebSphere Application Server V8.
- ► Disadvantages
  - – All applications being migrated must be ready at the same time.
  - – This approach only works 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, 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 Version 8 and the nodes in the older version.

4. Decide whether to migrate your nodes independently of one another. At the end, you must migrate all your nodes. Running in a mixed-cell configuration is considered a transitional state.

This approach provides the following advantages and disadvantages:

- ► 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.
- ► Disadvantages
  - – 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 15-5 on page 444).

To perform this migration, complete these steps:

1. Install WebSphere Application Server V8, 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 have to install the applications on the temporary profiles. You have to build them. An import command option is available to specify that only applications are to be built.

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 15-5 illustrates the flow of the fine-grained migration approach.
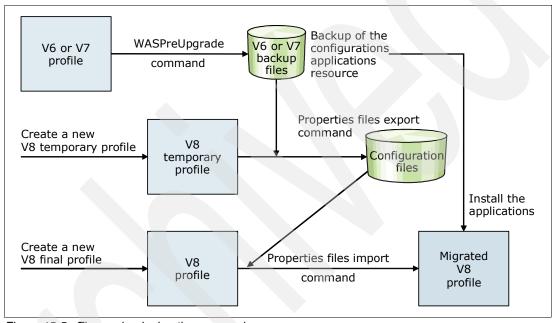

Figure 15-5   Fine-grained migration approach

The fine-grained approach has the following advantages and disadvantages:

► 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.

► Disadvantages

– The migration requires much 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, 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 as explained in "Stand-alone environment with the Configuration Migration Tools" on page 441. You have to 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, 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 442, or in "Job Manager environment with the Configuration Migration Tools" on page 445.

Cross-platform migration

In this approach, you migrate from a stand-alone WebSphere Application Server V7 instance installed on Linux to WebSphere Application Server V8 installed on Windows.

To perform this migration, complete these steps:

1. Install WebSphere Application Server V8 on the Linux machine to generate the `.zip` file that contains the migration tools.

2. Extract the `.zip` file in the Linux machine where WebSphere Application Server V7 is installed. Then back up the previous version of the profile by using the migration tools provided in the `.zip` file.

3. Install WebSphere Application Server V8 on the Windows machine, and create a profile.

4. Import the profile configuration by using the migration tools.

Figure 15-6 illustrates the flow of this migration.

Figure 15-6   Cross-platform migration

15.5.7 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, you do not need to migrate
your existing Jacl scripts to Jython. Create your scripts by using Jython.

15.6 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.

15.6.1 Migration and coexistence

Before attempting a migration, you must meet some coexistence and prerequisite conditions.
The earliest release level of WebSphere Application Server that can be directly migrated to
WebSphere Application Server V8 is V6. 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.

Table 15-1 shows the minimum requirements for the supported releases.

Table 15-1   WebSphere Application Server for z/OS releases for direct migration

Current release	Target release	Minimum level
V5.1	V8.0	V6.1.0
V6.0	V8.0	V6.0.2.12, if security is enabled
V6.1	V8.0	V6.1.0
V7.0	V8.0	V7.0.0

Keep in mind that the deployment manager must always be at the latest version level. For example, when migrating to V8, the deployment manager must be at V8. 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.

> **Job Manager:** You must migrate the Job Manager to V8.0 before migrating Deployment Managers or Administrative Agents that have servers registered to it.

15.6.2 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 Version 8, save your current procedures in case you need to fall back to the previous level.

  – If you choose to use different procedure names, you must 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 hierarchical file system (HFS) for each V8 node, which might require new procedure names if you used a shared HFS in previous versions.

► Review the guidance for *migrating, coexisting, and interoperating* in the WebSphere Application Server Version 8 Information Center at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

15.6.3 Overview of the migration process

After the product code of WebSphere Application Server for z/OS V8 is brought into the system using System Modification Program/Extended (SMP/E), the migration is performed in a three-step approach:

> **Nodes:** 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 highlighted in the following sections.

15.6.4 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 that is used to create the JCL jobs for the migration. It uses *Migration Definitions*, which is a construct that contains all data that is necessary to migrate a WebSphere Application Server for z/OS node from V6.0.x (and later) to V8. It contains the Migration Instructions that are personalized for each Migration Definition. It can optionally be used to transfer the JCL to the z/OS target system, if that system has a File Transfer Protocol (FTP) server up and running.

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 Version 8 nodes run.

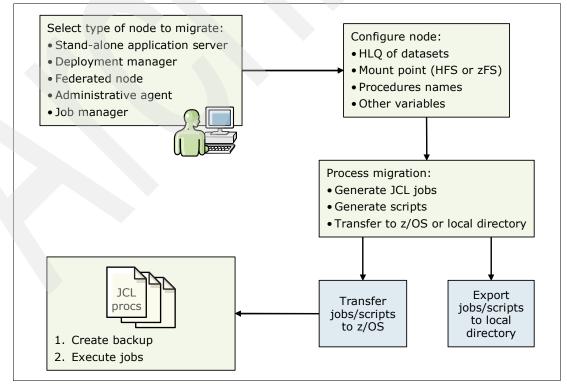Figure 15-7 shows a high-level overview of the migration process using z/OS Migration Management Tool.

Figure 15-7   Migration process with z/OS Migration Management Tool

Installing the z/OS Migration Management Toolbox

z/OS Migration Management Tool is available for Windows and Linux technology-based workstations. It is included in the Optional 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 6.6, "IBM Installation Manager" on page 157.) The WebSphere Customization Toolbox includes the z/OS Migration Management Tool, the z/OS Profile Management Tool, and the Web Server Plug-ins Configuration Tool.

If WebSphere Customization Toolbox is not installed on Installation Manager, follow these steps:

1. Go to the main window of Installation Manager, and click **Install**. For the tool to access the IBM download site, a user ID and password are required.

2. When you see the packages that can be installed, select **WebSphere Customization Toolbox**, and click **Next** (Figure 15-8 on page 449).

3. Read and accept the license agreement.

4. Identify the directory on your local file system in which you want to install the packages.

5. Select the packages that you want to install. After the installation process, you see the packages that were installed.
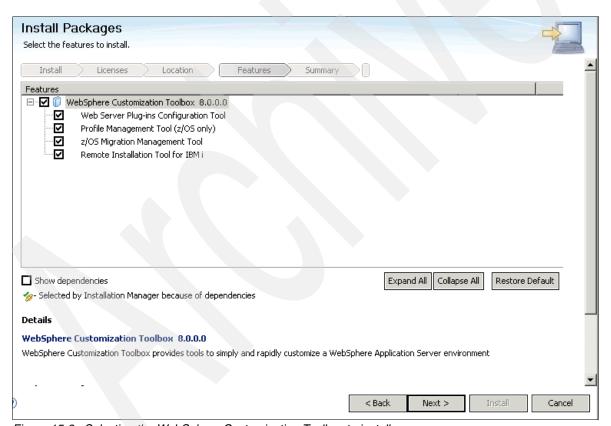

Figure 15-8   Selecting the WebSphere Customization Toolbox to install

To access WebSphere Customization Toolbox, follow these steps:

1. On a Windows operating system, go to **Start Programs** → **IBM WebSphere** → **WebSphere Customization Toolbox V8.0**. Click **WebSphere Customization Toolbox** to start the program.

2. Double-click **z/OS Migration Management Tool** to open the WebSphere Customization Toolbox V8.0 window (Figure 15-9).
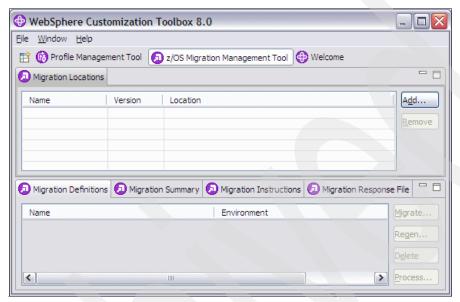

Figure 15-9   z/OS Migration Management Tool

Creating a Migration Definition

To create a Migration Definition, follow the steps provided in this section. For help regarding the z/OS Migration Management Tool, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *using the z/OS Migration Management Tool to create and manage Migration Definitions*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

1. Complete the configuration worksheet in the WebSphere Application Server Version 8 Information Center at the following address (scroll down toward the bottom of the page):

   http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.
   websphere.migration.zseries.doc/info/zseries/ae/tmig_zmmt_usemmt.html

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. In the next window (Figure 15-10), under Migration Locations, select the migration location that you created, and click **Migrate**.

Figure 15-10   Migration process
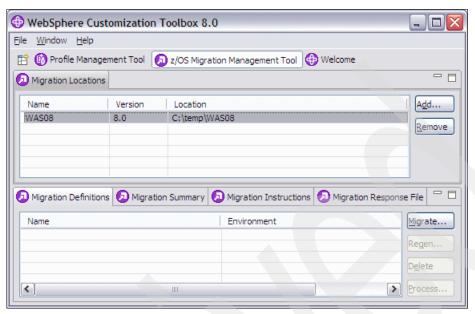
5. In the Migration Node Type Selection window (Figure 15-11), choose the type of node migration, and click **Next**.
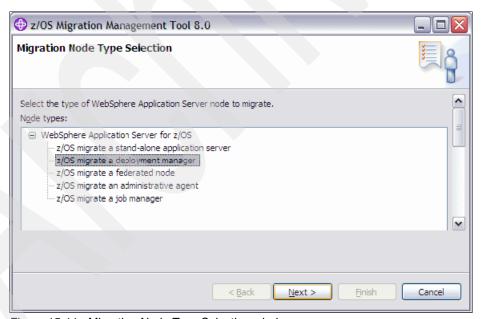

Figure 15-11   Migration Node Type Selection window

6. In the multi-panel window that opens, complete the fields in the panels by using the values that you entered for the variables on the configuration worksheet that you created. Click **Back** and **Next** as necessary.

> **Tips:**
>
> ► The z/OS Migration Management Tool has a good help file that you can access by hovering the mouse over a field for help information.
>
> ► In the Server Customization (Part 2) window, a Migration Definition identifier is shown. 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** to build the Migration Definition on your workstation. This summary window shows the definition type, location, and name.

8. Read the information in the Migration Creation Summary window, and click **Finish**.

As a result, you will find a directory structure populating the path that was specified to store the Migration Definitions. For the next steps, you must upload the migration jobs by using the z/OS Migration Management Tool, as explained in the following section.

Creating migration jobs

To create migration jobs, follow these steps:

1. To create the JCL jobs and the scripts, under Migration Definitions, select the definitions that you created, and click **Process**, as shown in Figure 15-12.
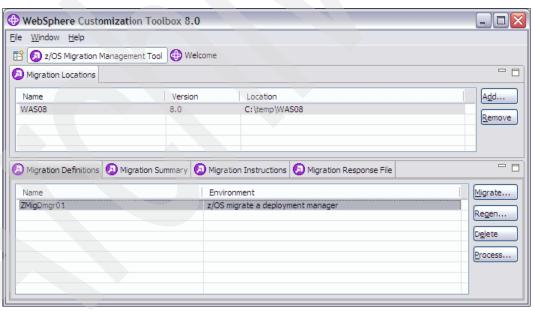
Figure 15-12   Processing the migration definition

2. When you see the types of processing to perform on the migration definition, select one type of processing:

   – If you choose to upload to the target z/OS system, you must provide the host name, IP address, user ID, and password. The JCL and the scripts are then transferred to the z/OS system on the CNTL and DATA data sets that you named in the migration definition.

   – If you choose to export to a local directory, the JCLs and scripts are generated on your local system.

   Click **Next**.

For detailed migration instructions, click a Migration Definition to select it, 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 stated on the **Migration Instruction** tab. The instructions reflect the variables that were entered in the Migration Definition panels.

15.6.5 Migration Management Tool script

This section provides information about the z/OS Migration Management Tool script (`zmmt.sh`) that you can use to create the JCL needed for a node migration.

Overview

You can create the migration jobs completely on z/OS by using a shell script, `zmmt.sh`, which is in the `/usr/lpp/zWebSphere/V8R0/bin` product bin directory. The script also creates the CNTL and DATA data sets and the corresponding JCL that is needed to perform the migration. You need a response file. This response file contains information about the node construction.

> **Using the z/OS Migration Management Tool to create a response file:** 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.

The script

The script is in the `/bin` directory. The default directory is `/usr/lpp/zWebSphere/V8R0`. Example 15-1 shows how to run the script.

Example 15-1   Migration management command

```
zmmt.sh -workspace /xxx -transfer -allocate -responseFile
/xxx/ZCellcmd.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/zWebSphere601/V8R0/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 performs 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 targetHLQ 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 (Example 15-2).

Example 15-2   osgi shell for the zmmt.sh script

```
/usr/lpp/zWebSphere/V8R0/bin#>zmmt.sh -workspace –responseFile
/tmp/zDMgr01.responseFile

osgi>
```

> **osgi command shell:** An **osgi** command shell is an execution environment that allows remote management of Java application 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. However, the script writes messages such as the messages shown in Example 15-3.

Example 15-3   osgi messages when issuing the zmmt.sh script

```
osgi> Customization definition successfully written to /tmp/ZDMgr01 Attempting
to allocate dataset: BOSS.VICOM.BOSS0173.CNTL
Allocation successful.
Attempting to allocate dataset: BOSS.VICOM.BOSS0173.DATA
Allocation successful.
Copying CNTL files to BOSS.VICOM.BOSS0173.CNTL...
Copy successful.
Copying DATA files to BOSS.VICOM.BOSS0173.DATA...
Copy successful.
```

► If you have to rerun the command, delete the profilePath directory. If the directory still exists, the **osgi** shell shows an error message (Example 15-4).

Example 15-4   zmmt.sh script error message

```
osgi> The following validation errors were present with the command line
arguments: profilePath: The profile path is not valid.
```

15.6.6 Migration jobs

Migration jobs are created by using z/OS Migration Management Tool or the zmmt.sh script. This section provides a brief overview of these jobs.

> **Important:** 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. However, you only need to run five of them. Table 15-2 on page 455 shows an overview of the jobs that are used by the migration, depending on 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 15-2   WebSphere Application Server for z/OS V8.0 migration jobs

Job name[a]	Deployment Manager	Federated server	Stand-alone server
BBOMxZFS or BBOMxHFS	Allocates HFS or zFS	Allocates HFS or zFS	Allocates HFS or zFS
BBOMxCP	Copies tailored JCL to PROCLIB	Copies tailored JCL to PROCLIB	Copies tailored JCL to PROCLIB
BBOWMG1x		Clear transaction logs (for XA connectors only)	Clear transaction logs (for XA connectors only)
BBOWMG2x		Disable Peer Restart and Recovery mode (XA only)	Disable Peer Restart and Recovery mode (XA only)
BBOWMG3x	Perform migration	Perform migration	Perform migration

a. The value for x depends on the profile that you are migrating.

BBOWMG3x job

The BBOWMG3x job performs 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 that was generated during the creation of your migration jobs. For normal migration, the space used in the /tmp directory is small. However, if you turn on tracing, the space used can be large. 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 WAS_HOME structure.

7. CRPROF: Create V8 default profile.

8. PREUPGRD: Back up some files in the HFS to be used by WASPostUpgrade.

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
  - /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, turn on traces. The trace states are disabled by default. 'xxxx.DATA(BBOWMxEV)' has to be updated to enable tracing:

- TraceState=enabled
- profileTrace=enabled
- preUpGradeTrace=enabled
- postUpGradeTrace=enabled

► If the job fails in the VERIFY step, it is most 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 that was created during the failed run, before re-running the job. Also check the original configuration for the serverindex.xml file being 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 the **Disable previous deployment manager** check box in the Server Customization (Part 2) window 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, to occur because it runs for a long time. TIME=NOLIMIT on the JCL job card can avoid the problem. Also 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.

15.6.7 Migration considerations for 64-bit mode

WebSphere Application Server V8 runs in 64-bit mode. 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 to run it in a 64-bit application server.

If the application uses the Java Native Interface (JNI) to call a native program, that native program must be a 64-bit program. Typically, these native programs are code written in C or C++, or perhaps an IBM Language Environment® compliant assembler. This point is especially important to check for 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

Bigger heap needed 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, good minimum and maximum heap sizes should be identified by a verbose garbage collection (GC) analysis. With this technique, you can identify these values, reducing the GC overhead and saving processor time. Consider performing a verbose GC analysis on a regular basis, especially if the number of users or the amount of transactions has changed.

For more information about how to perform a verbose GC analysis, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *using the z/OS Migration Management Tool to create and manage migration definitions*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

> **Maximum heap size:** In general, the structure of WebSphere Application Server for z/OS significantly reduces the maximum heap size compared to the maximum heap sizes used on the distributed platforms. For more information, see 14.1.6, "Runtime processes" on page 391.

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

459

Topology review

Figure A-1 illustrates one of the most common (and complex) topologies implemented in real scenarios according to 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.
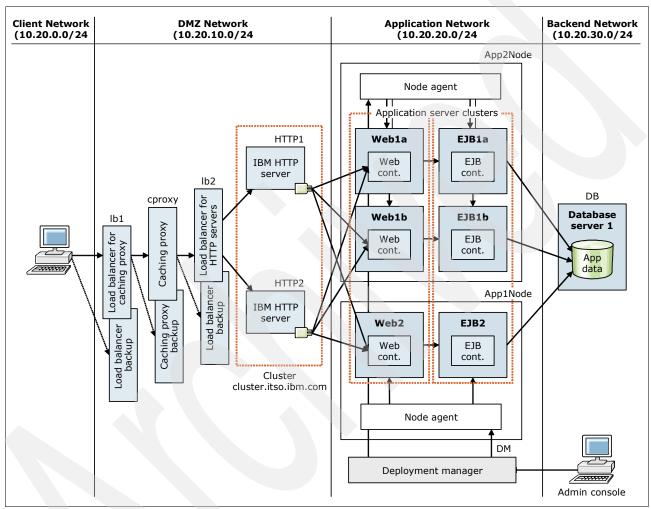

Figure A-1   Complex topology

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 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 (JSPs) 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, which 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 consisting of three application servers

Each cluster spans two machines. In this topology, one cluster contains application servers that provide the web container functionality of the applications (servlets and JSPs), and the second cluster contains the EJB container functionality. 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 running 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 (SPOF) 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 machine processor and memory. More JVMs can be created to take more advantage of the available resources.

► Workload management

In this topology, workload management is done by the load balancer to distribute work among the web servers and by the WebSphere Plug-ins to 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 JVM if necessary. As more nodes or more web servers are added, the loader balancer can continue distributing a load across all new members added to the original topology.

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, 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 460. The objective of this sample topology is to demonstrate the necessary steps to implement the main elements of the complex topology explained in "Topology review" on page 460. Figure A-2 shows the topology that is implemented in this section.
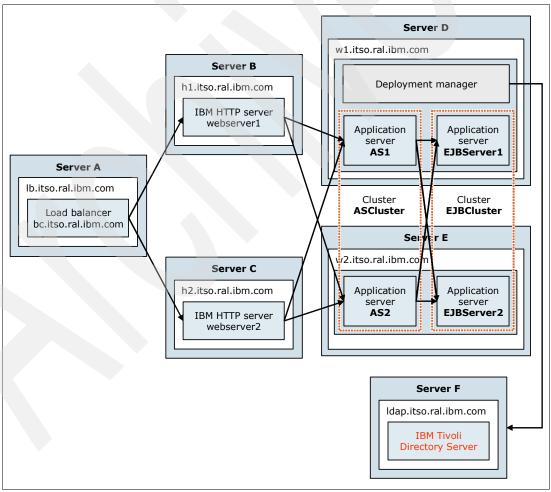

Figure A-2   Sample topology

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 IBM Redbooks publication. For more information, see Appendix B, "Additional material" on page 483.

The BeenThere sample application demonstrates the workload management and clustering capabilities of WebSphere Application Server. Because its responses contain the application servers where requests have been 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 has been 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, all elements are installed with the Installation Manager.

> **No Internet connection:** 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. Launch `lbadmin`.

   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 have to use to access the BeenThere application. It is a different IP address from the machine 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 from **Start Programs** → **IBM HTTP Server for WebSphere Software V8.0** → **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 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 installation.

e. Launch the WebSphere Customization Toolbox after installation.

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.

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 launch the Profile Management Tool selected, and click **Finish**.

2. Create a deployment manager profile:

    a. Click the **Create** button.

    b. Create a management profile of type *deployment manager* with the default options.

3. Start the deployment manager from **Start Programs** → **IBM WebSphere** → **WebSphere Application Server Network Deployment V8.0** → **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 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. Launch Profile Management Tool, and select **Custom Profile**.

    b. Follow the prompts taking the defaults, including the federation of the node to the cell as part of the process. (The deployment manager must be installed and running.)

3. Create the application server clusters:

    a. Log in to the Integrated Solutions 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 in to the Integrated Solutions Console.

2. For both application servers from the ASCluster, complete the following steps:

   a. Navigate to **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 in to the Integrated Solutions 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 in to the Integrated Solutions Console. Then select **Applications** → **Application Types** → **WebSphere Enterprise Applications**. The BeenThere application should be listed and 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 **BeenThereEJB** module and the **MyEJBCluster** row under Cluster and Servers. Then click **Apply**.

   > **Using a property file:** It is also possible to 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 Integrated Solutions Console was used in this example. To create the property file and use it to deploy the modules, see the WebSphere Application Server Version 8 Information Center at the following address, and search for *installing enterprise application files by adding properties files to a monitored directory*:
   >
   > http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

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 in to the Integrated Solutions Console.

   b. Navigate to **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.

> **Validation failed error message:** 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. Enable application security. In the Global security panel, set **Enable application security**.

3. Assign administrative roles.

    The user account that is being used to log in 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 monitor 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 it 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

After you build the environment according to the sample topology, you must test and verify it. In this scenario, the following aspects were selected for testing:

► Service
► Administration

Service

Because the purpose of the tests in this appendix is 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 test was the first one, the window shown in Figure A-3 opened, prompting for user name and password. In this scenario, we entered the credentials of the user stored in the LDAP repository and clicked **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 a consequence of the weights that were configured when those servers were created (Table A-1 on page 466).

Workload Management - BeenThere

Servlet run summary

Summary of the node name, application server name, and process ID of the environment where the BeenThere servlet request is dispatched.

Node	w2Node01
Server	AS2
Process ID	4660

Bean run summary

Summary of the node name, application server name, and process ID of the environment where each BeenThere EJB request is dispatched.

Iteration	Node	Server	Process ID
1	w2Node01	EJBServer2	6108
2	w1Node01	EJBServer1	4548
3	w2Node01	EJBServer2	6108
4	w2Node01	EJBServer2	6108

Figure A-4   Response from the BeenThere application

> **Prefer local feature:** 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.
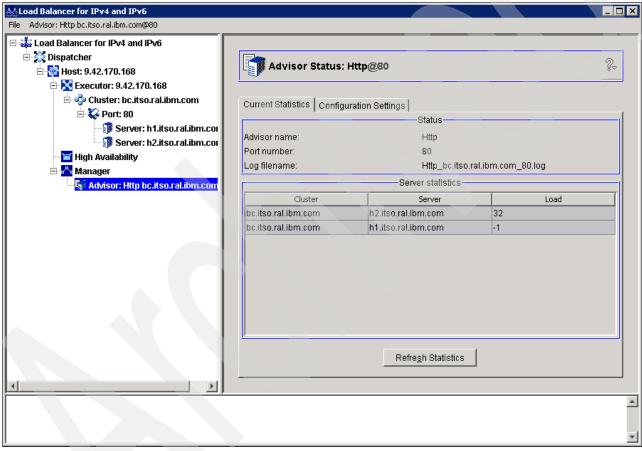
Figure A-5   Advisor Status window

Due to the load balancing mechanism, this environment was still working, and new requests to the system received 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).
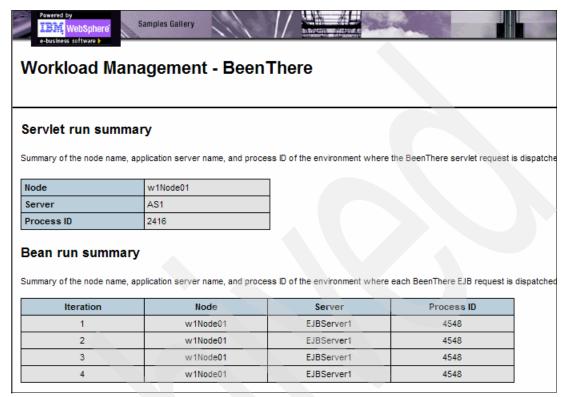

Workload Management - BeenThere

Servlet run summary

Summary of the node name, application server name, and process ID of the environment where the BeenThere servlet request is dispatche

Node	w1Node01
Server	AS1
Process ID	2416

Bean run summary

Summary of the node name, application server name, and process ID of the environment where each BeenThere EJB request is dispatched

Iteration	Node	Server	Process ID
1	w1Node01	EJBServer1	4548
2	w1Node01	EJBServer1	4548
3	w1Node01	EJBServer1	4548
4	w1Node01	EJBServer1	4548

Figure A-6   All responses from EJBServer1

Administration

In WebSphere Application Server V8, you can submit jobs from the Integrated Solutions Console from a deployment manager. Two useful jobs are how to collect files and how to create a profile on the w2.itso.ral.ibm.com server.

First you must register the target host that will receive the jobs. To register the target, complete these steps:

1. Log in to the Integrated Solutions Console, and click **Jobs** → **Targets**.

2. In the Targets window (Figure A-7), click **New Host**.

Figure A-7   Targets window

3. In the Targets window (Figure A-8), enter the required information:

   a. Enter a host name.
   b. Select an operating system. In this example, we select **any**.
   c. Enter an administrative user.
   d. Select Password authentication. Then enter the password and confirm it.
   e. Click **OK**.

Figure A-8   Target properties

4. After adding the target, click the target name to obtain information about the host (Figure A-9).
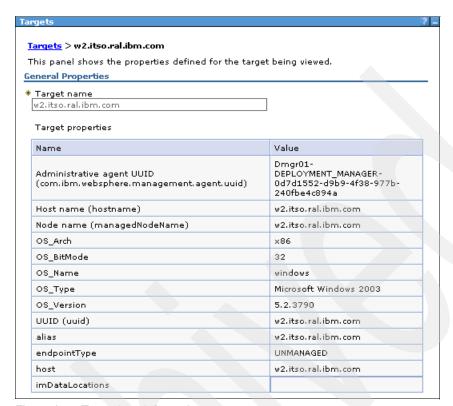
Figure A-9   Target host information

Collecting files

Common files used during the WebSphere Application Server environment management are log files or application property files. When troubleshooting an application problem, files such as javacores or heap dumps, are also collected to help diagnose the issue. All those files can be collected by using the collection job.

In this example, a javacore was generated first on the w2.itso.ral.ibm.com system by using the new troubleshooting features in WebSphere Application Server V8 and was then collect by using a collection job.

To accomplish this task, complete these steps:

1. Log in to the Integrated Solutions Console, and navigate to **Troubleshooting** → **Java dumps and cores**.

2. Select **server A2**, and click the **Java Core** button, which generates a file starting with `javacore` in the profile root.

3. Navigate to **Jobs** → **Submit**.

4. In the Choose a job type panel (Figure A-10), if the **Collect file** job type is not selected, select it. Click **Next**.
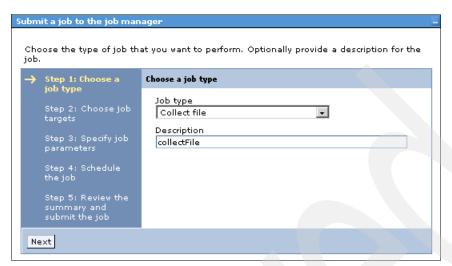
Figure A-10   Collecting file job

5. In the Choose job targets panel (Figure A-11), add the `w2.itso.ral.ibm.com` node as the Target name, and enter its authentication credentials. Then click **Next**.
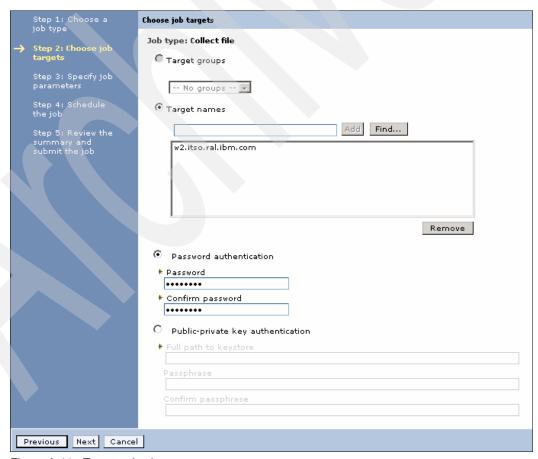
Figure A-11   Target selection

6. In the Specify job parameters panel (Figure A-12), specify the path for the javacore file, which, by default, is the profile root. You can use wildcards such as an asterisk (*) for multiple unknown characters in the file name. By default, the destination path is `dmgr_profile/config/temp/JobManager/jobToken/targetName`. Click **Next**.
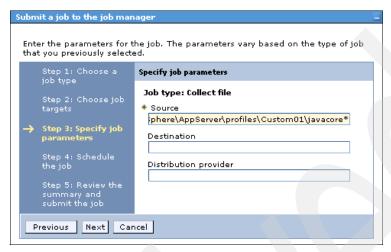
Figure A-12   Job parameters

7. In the Schedule the job panel (Figure A-13), to run the task immediately, accept the default values for the job schedule, and click **Next**.
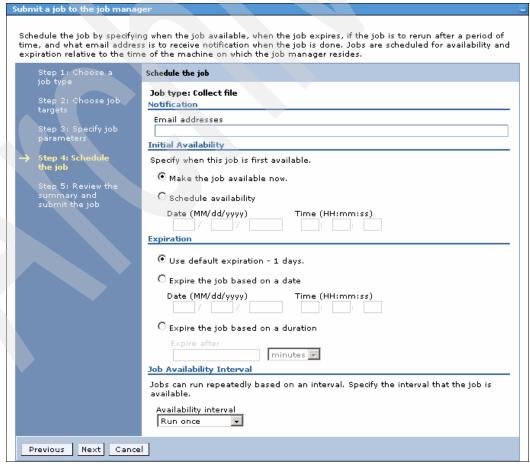
Figure A-13   Job schedule

8. In the next panel (Figure A-14), review the job summary, and click **Finish**.
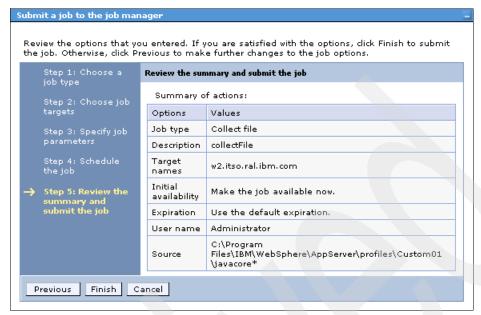
Figure A-14   Job summary

After a while the Job status refreshed from pending to *Succeeded* (Figure A-15).
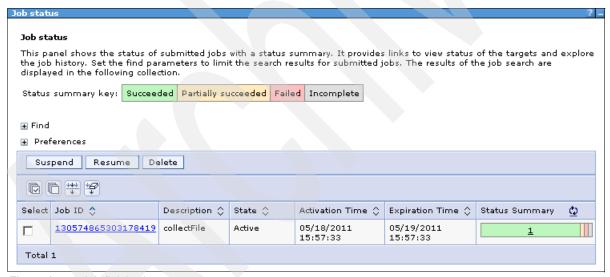
Figure A-15   Job finished

9. To confirm that the file was copied, navigate to the destination default path. In this case, the file was copied successfully (Figure A-16).
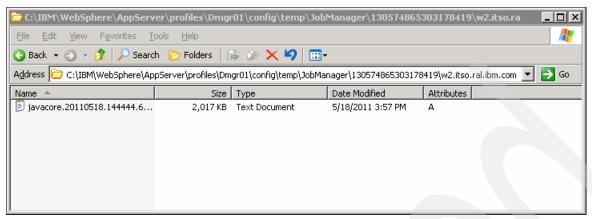
Figure A-16   File copied in destination path

Creating a profile

You can create profiles remotely by submitting a job from the deployment manager console. In this example, a new profile was added to the w2.itso.ral.ibm.com server. To accomplish this task a response file was created with the necessary parameters for the profile creation. Example A-1 shows the response file used for the profile creation.

Example A-1   Response file for profile creation

```
create
profileName=myProfile
profilePath=C:\\Program Files\\IBM\\WebSphere\\AppServer\\profiles\\myProfile
templatePath=C:\\Program
Files\\IBM\\WebSphere\\AppServer\\profileTemplates\\default
```

To submit the job for the profile creation, complete these steps:

1. Log in to the Integrated Solutions Console, and then expand **Jobs** → **Submit**.

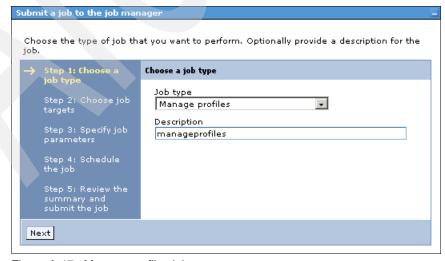2. In the Choose a job type panel (Figure A-17), select the **Manage profiles** job type.

Figure A-17   Manage profiles job

3. In the Choose job targets panel (Figure A-18), add the `w2.itso.ral.ibm.com` host as the job target. Then click **Next**.
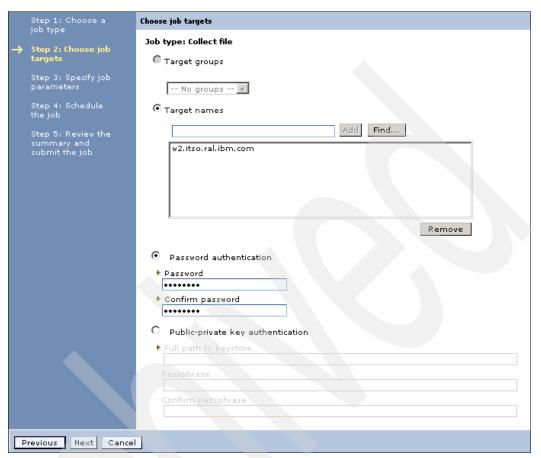

Figure A-18   Job target

4. In the Specify job parameters panel (Figure A-19), specify the WebSphere Application Server home path and response file location path. Then click **Next**.
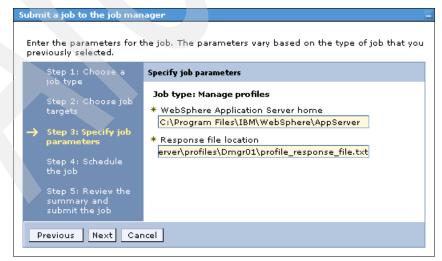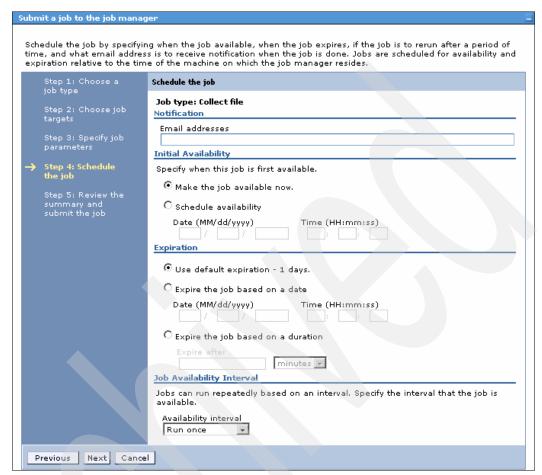

Figure A-19   Job parameters

5. In the Schedule the job panel (Figure A-20), to run the task immediately, accept the default values for the job schedule. Then click **Next**.

Figure A-20   Job schedule

6. In the next panel (Figure A-21), review the job summary, and then click **Finish** to submit the job.
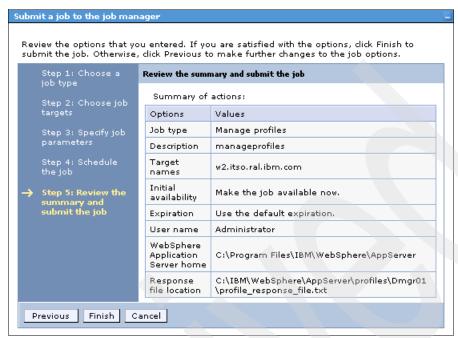

Figure A-21 Job summary

After a few minutes, the job shows a status of *Succeeded* (Figure A-22).
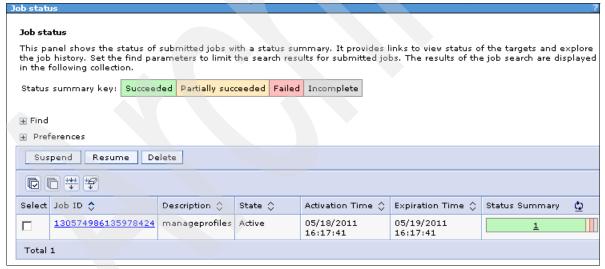

Figure A-22 Job status

7. To confirm the profile creation, navigate to **Targets**. Select the host, click **Display Resources,** and then select **Profile** (Figure A-23).

Figure A-23   Target resources selection

The new profile is listed under the host resources. You can select it to view its general properties (Figure A-24).
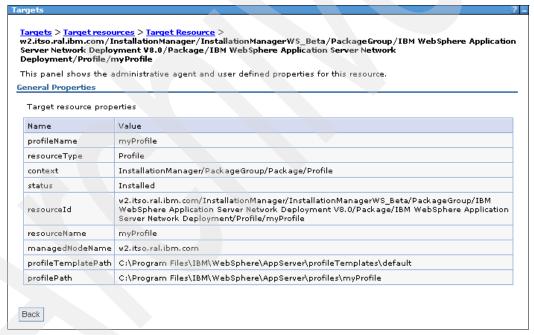

Figure A-24   New profile properties

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, such as collecting files, creating profiles, and discovering installed resources on the job targets.

Additional material

This book refers to additional material that you can download from the Internet as we describe in the following sections.

Locating the web material

The web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your web browser at:

ftp://www.redbooks.ibm.com/redbooks/SG247957

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select **Additional materials** and then open the directory that corresponds with the IBM Redbooks form number, SG247957.

Using the web material

The additional web material that accompanies this book includes the following files:

File name   *Description*
SG247957.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

We consider the publications that we list in this section particularly suitable for a more detailed discussion of the topics that we cover 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.

- *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
- *System Programmer's Guide to: Workload Manager*, SG24-6472
- *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257
- *WebSphere Application Server V5: Separating Static and Dynamic Web Content*, TIPS0223
- *WebSphere Application Server V6.1: Planning and Design*, SG24-7305
- *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708
- *WebSphere Application Server V7 Migration Guide*, REDP-4635
- *WebSphere Application Server V7.0 Security Guide*, SG24-7660

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:

- ► Java EE 6 specifications

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

- ► The WebSphere Application Server Version 8 Information Center

  http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

- ► Project Zero

  http://www.projectzero.org/

- ► WebSphere sMash

  http://www.ibm.com/software/webservers/smash/

- ► Tivoli Access Manager for e-business

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

- ► 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/

- ► WebSphere Portal Server - Enterprise portal software

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

- ► Java Community Process: Information about SIP applications

  http://www.jcp.org/

- ► Developing enterprise OSGi applications for WebSphere Application Server,

  http://www.ibm.com/developerworks/websphere/techjournal/1007_robinson/1007_robinson.html?S_TACT=105AGY82&S_CMP=MAVE

- ► IBM WebSphere Developer Technical Journal: Maintain continuous availability while updating WebSphere Application Server enterprise applications

  http://www.ibm.com/developerworks/websphere

- ► IBM Systems Workload Estimator

  http://www.ibm.com/systems/support/tools/estimator/index.html

- ► SPEC benchmark

  http://www.spec.org/benchmarks.html

- ► TPC benchmark

  http://www.tpc.org/information/benchmarks.asp

- ► IBM SmartCloud Enterprise

  http://www.ibm.com/services/us/igs/cloud-development/

- ► IBM Support Assistant Tool Add-Ons List

  http://www.ibm.com/support/docview.wss?rs=3455&uid=swg27013116
- ► IBM Support Assistant

  http://www.ibm.com/software/support/isa/
- ► Application Response Measurement from the Open Group

  http://www.opengroup.org/tech/management/arm/
- ► Understanding WebSphere Application Server for z/OS

  http://websphere.sys-con.com/read/98083.htm
- ► Apache Subversion

  http://subversion.apache.org/
- ► JUnit

  http://www.junit.org/
- ► Apache Ant

  http://ant.apache.org
- ► Java Messaging Service on Oracle

  http://java.sun.com/products/jms
- ► RESTful web service

  http://www.peej.co.uk/articles/restfully-delicious.html
- ► JinsightLive for IBM System z

  http://www.alphaworks.ibm.com/tech/jinsightlive
- ► Eclipse Test and Performance Tools Platform Project

  http://www.eclipse.org/tptp/

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

IBM

Redbooks

IBM WebSphere Application Server V8 Concepts, Planning, and Design Guide

IBM WebSphere Application Server V8 Concepts, Planning, and Design Guide

IBM ®

Redbooks ®

Includes details about end-to-end planning for WebSphere implementations

Defines WebSphere concepts and preferred 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 environments. The target audience of this book is IT architects and consultants who want more information about the planning and designing of application-serving environments, from small to large, and complex implementations.

This book addresses the packaging and features in WebSphere Application Server V8 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 and WebSphere Application Server Network Deployment V8 on distributed platforms and for WebSphere Application Server for z/OS V8. This book contains information about migration considerations when moving from previous releases.