

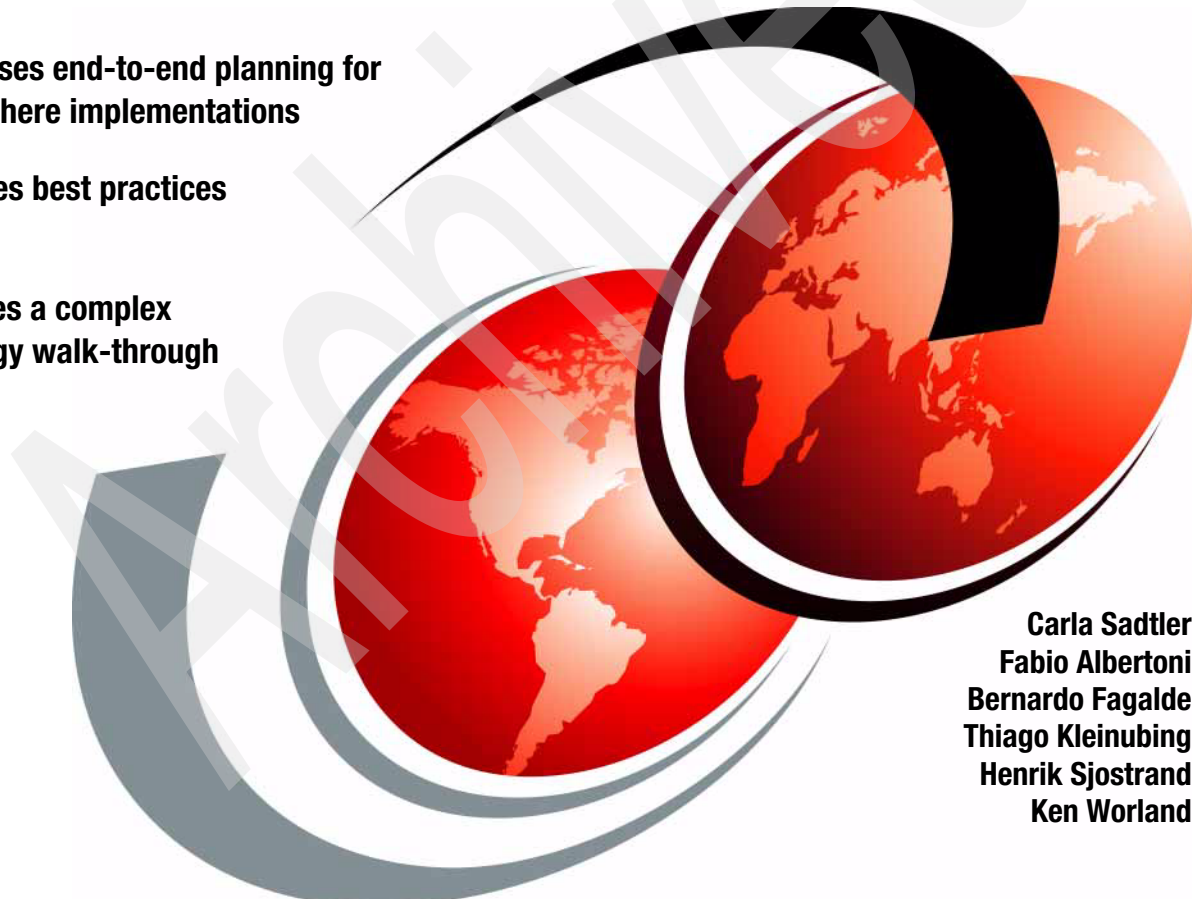
WebSphere Application Server V6.1

Planning and Design

Discusses end-to-end planning for
WebSphere implementations

Provides best practices

Includes a complex
topology walk-through



Carla Sadtler
Fabio Albertoni
Bernardo Fagalde
Thiago Kleinubing
Henrik Sjostrand
Ken Worland



International Technical Support Organization

WebSphere Application Server V6.1: Planning and Design

October 2006

Archived

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

First Edition (October 2006)

This edition applies to IBM WebSphere Application Server Version 6.1 and IBM WebSphere Application Server for z/OS Version 6.1.

© Copyright International Business Machines Corporation 2006. All rights reserved.

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

Contents

Notices	ix
Trademarks	x
Preface	xi
The team that wrote this redbook	xi
Become a published author	xiv
Comments welcome	xv
Chapter 1. Introduction to WebSphere Application Server V6.1	1
1.1 Product overview	2
1.2 WebSphere Application Server	3
1.3 Packaging	5
1.4 Supported platforms and software	9
1.4.1 Operating systems	9
1.4.2 Web servers	10
1.4.3 Database servers	10
1.4.4 Directory servers	11
Chapter 2. Integration with other products	13
2.1 Tivoli Access Manager	14
2.1.1 WebSphere Application Server security	14
2.1.2 Tivoli Access Manager and WebSphere Application Server	15
2.2 Tivoli Directory Server	18
2.2.1 The Lightweight Directory Access Protocol (LDAP)	18
2.2.2 Tivoli Directory Server and WebSphere Application Server	19
2.3 WebSphere MQ integration	20
Chapter 3. Planning for infrastructure	25
3.1 Infrastructure deployment planning	26
3.2 Design for scalability	27
3.3 Sizing	29
3.4 Benchmarking	30
3.5 Performance tuning	32
3.5.1 Application design problems	32
3.5.2 Understand your requirements	32
3.5.3 Test environment setup	33
3.5.4 Load factors	33
3.5.5 Production system tuning	34
3.5.6 Conclusions	35

3.6	Planning for backup and recovery	36
3.6.1	Risk analysis	36
3.6.2	Recovery strategy	37
3.6.3	Backup plan	37
3.6.4	Recovery plan	37
3.6.5	Update and test process	38
Chapter 4. WebSphere Application Server concepts		39
4.1	WebSphere Application Server concepts	40
4.1.1	Stand-alone application servers	40
4.1.2	Distributed application servers	41
4.1.3	Nodes, node groups, and node agents	42
4.1.4	Cells	42
4.1.5	Application server clusters	43
4.1.6	Web servers	43
4.2	Distributed server environments	45
4.2.1	Single cell configurations	45
4.2.2	Multiple cells	47
4.2.3	Mixed node versions in a cell	47
4.3	Application server clusters	48
4.4	Runtime processes	51
4.4.1	Distributed platforms	51
4.4.2	WebSphere Application Server for z/OS	52
4.5	Using Web servers	54
4.5.1	Managed Web servers	55
4.5.2	Unmanaged Web servers	56
4.5.3	IBM HTTP Server as an unmanaged Web server (special case)	57
Chapter 5. Topologies		59
5.1	Topology selection criteria	60
5.1.1	Security	60
5.1.2	Performance and throughput	60
5.1.3	Availability	62
5.1.4	Maintainability	64
5.1.5	Topology selection summary	64
5.2	Terminology	65
5.3	Stand-alone server topology	67
5.4	Reverse proxy topology	71
5.5	Vertical scaling topology	73
5.6	Horizontal scaling topology	75
5.7	Horizontal scaling with IP sprayer topology	77
5.8	Topology with redundancy of several components	79
Chapter 6. Planning for installation		85

6.1	What is new in V6.1	86
6.2	Selecting a topology	87
6.3	Selecting hardware and operating systems	88
6.4	Naming conventions	88
6.5	Planning for WebSphere Application Server	88
6.5.1	Determine whether to perform a single install or multiple	90
6.5.2	Select an installation method	92
6.5.3	Plan for profiles	94
6.5.4	Plan for names	102
6.5.5	Plan for TCP/IP port assignments	105
6.5.6	Security considerations for the installation	106
6.6	Planning for migration	109
6.7	Planning for the Web server and plug-ins	111
6.7.1	Stand-alone server environment	114
6.7.2	Distributed server environment	117
6.8	Planning checklist for the installation	120
Chapter 7. Planning for application development and deployment		123
7.1	What is new in V6.1	124
7.2	End-to-end life cycle	125
7.3	Development and deployment tools	127
7.3.1	Application Server Toolkit V6.1	128
7.3.2	Rational Web Developer V6.0	129
7.3.3	Rational Application Developer V6.0	130
7.3.4	WebSphere rapid deployment	131
7.3.5	Which tool to use	132
7.4	Naming conventions	133
7.4.1	Naming for applications	133
7.4.2	Naming for resources	133
7.5	Source code management	134
7.5.1	Rational ClearCase	135
7.5.2	Concurrent Versions System (CVS)	136
7.5.3	Which SCM to use	136
7.6	Automated build process	137
7.7	Automated functional tests	139
7.8	Test environments	139
7.9	Managing application configuration settings	144
7.9.1	Classifying configuration settings	144
7.9.2	Managing configuration settings	145
7.10	Planning for application upgrades in production	148
7.11	Mapping applications to application servers	150
7.12	Planning checklist for applications	151

Chapter 8. Planning for system management	153
8.1 What is new in V6.1	154
8.2 Administrative security	154
8.3 WebSphere administration facilities	155
8.3.1 Administrative console	155
8.3.2 WebSphere scripting client (wsadmin)	156
8.3.3 Task automation with Ant	157
8.3.4 Administrative programs	157
8.3.5 Command line tools	157
8.4 Configuration planning	158
8.4.1 Configuration repository location and synchronization	158
8.4.2 Configuring application and server startup behavior	158
8.4.3 Custom application server configuration templates	159
8.4.4 Planning for resource scope use	160
8.5 Change management topics	162
8.5.1 Application updates	163
8.5.2 Changes in topology	164
8.6 Problem management	165
8.6.1 Logs and traces	165
8.6.2 Fix management	167
8.6.3 Backing up and restoring the configuration	167
8.7 Planning checklist for system management	167
Chapter 9. Planning for performance, scalability, and high availability	169
9.1 What is new in V6.1	170
9.2 Scalability	170
9.2.1 Workload categorization	171
9.2.2 System tuning	172
9.2.3 Application environment tuning	173
9.2.4 Scaling the system	174
9.2.5 Default messaging provider scalability	175
9.3 Workload management	176
9.3.1 Clustering application servers	177
9.3.2 Scheduling tasks	179
9.4 High availability	179
9.4.1 Hardware availability	180
9.4.2 Process availability	180
9.4.3 Data availability	181
9.4.4 Clustering and failover	182
9.4.5 Maintainability	183
9.4.6 WebSphere Application Server high availability features	183
9.5 Caching	187
9.5.1 Dynamic caching	188

9.5.2	Edge caching	188
9.5.3	Data caching	190
9.6	Session management	191
9.6.1	Session support	192
9.7	Data replication service	197
9.8	WebSphere Application Server performance tools	198
9.8.1	Performance Monitoring Infrastructure	199
9.8.2	Tivoli Performance Viewer	200
9.8.3	WebSphere performance advisors	201
9.8.4	WebSphere request metrics	202
9.9	Planning checklist for performance	205
Chapter 10. Planning for messaging		209
10.1	Messaging overview: What is messaging?	210
10.2	What is new in messaging for V6.1	210
10.3	Messaging considerations: Is messaging for me?	211
10.4	Messaging options: What things do I need?	212
10.4.1	Selecting a messaging service type	212
10.4.2	Choosing a messaging service provider	214
10.5	Messaging topologies: How can I use messaging?	215
10.5.1	Default messaging provider concepts	216
10.5.2	Choosing a messaging topology	217
10.6	Messaging features: How secure and reliable is it?	224
10.6.1	More messaging concepts	224
10.6.2	Planning for security	225
10.6.3	Planning for high availability	226
10.6.4	Planning for reliability	227
10.7	Planning checklist for messaging	229
Chapter 11. Planning for Web services		231
11.1	What are Web services?	232
11.2	What is new in V6.1	233
11.3	Are Web services something you should use?	234
11.4	What do you need to implement Web services?	236
11.4.1	What is the basic Web services architecture?	237
11.4.2	How can this architecture be used?	239
11.4.3	How does WebSphere implement this architecture?	245
11.5	What other Web service considerations are there?	249
11.5.1	What are the options for Web service security?	250
11.5.2	How can Web service performance be improved?	250
11.6	Planning checklist for Web services	251
Chapter 12. Planning for security		253
12.1	What is new in V6.1	254

12.2 Why you need security and how it works in WebSphere	259
12.3 Security fundamentals on WebSphere	264
12.3.1 Authentication	264
12.3.2 Authentication process	269
12.3.3 Authorization	271
12.4 J2EE security	272
12.4.1 Security roles	272
12.4.2 Security for J2EE resources	273
12.5 Planning for security	278
12.6 Planning checklist for security	281
Appendix A. Sample topology walk-through	283
Topology review	284
Advantages	286
Disadvantages	286
Component installation	287
Deployment manager node (server E)	288
Application server nodes (server D)	289
IBM HTTP Server V6.1 (server B and server C)	290
Creating the application server clusters	291
Load Balancer (server A)	291
Deploying applications	293
Testing the topology	293
Related publications	295
IBM Redbooks	295
Online resources	295
How to get IBM Redbooks	298
Help from IBM	298
Index	299

Notices

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

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

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

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

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

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

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

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


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

COPYRIGHT LICENSE:

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

Trademarks

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

AIX 5L™	ibm.com®	Rational®
AIX®	IBM®	Redbooks (logo)  ™
CICS®	IMS™	Redbooks™
ClearCase MultiSite®	Informix®	RequisitePro®
ClearCase®	iSeries™	RUP®
ClearQuest®	Lotus®	S/390®
Cloudscape™	MVS™	SecureWay®
DB2 Universal Database™	OS/400®	System z™
DB2®	Power PC®	Tivoli®
developerWorks®	POWER™	WebSphere®
Domino®	RACF®	Workplace™
HACMP™	Rational Rose®	z/OS®
i5/OS®	Rational Unified Process®	zSeries®

The following terms are trademarks of other companies:

iPlanet, Enterprise JavaBeans, EJB, Java, Java Naming and Directory Interface, Javadoc, JavaBeans, JavaScript, JavaServer, JavaServer Pages, JDBC, JDK, JMX, JSP, JVM, J2EE, J2SE, Solaris, Sun, Sun Java, Sun ONE, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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

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

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

Preface

This IBM® Redbook discusses the planning and design of IBM WebSphere® Application Server Version 6.1 environments. The content of this redbook is oriented to IT architects and consultants who require assistance when planning and designing small implementations to large and complex implementations.

This redbook addresses the packaging and features incorporated in WebSphere Application Server, covers the most common implementation topologies, and addresses planning for specific tasks and components that conform to the WebSphere Application Server environment.

The book includes planning information for WebSphere Application Server V6.1 and WebSphere Application Server Network Deployment V6.1 on distributed platforms and WebSphere Application Server for z/OS®. It does not cover WebSphere Application Server for i5/OS®.

Note the following companion pieces to this book:

- ▶ *WebSphere Application Server V6.1: Technical Overview*, REDP-4191, at:
<http://www.redbooks.ibm.com/abstracts/redp4191.html>
- ▶ *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304, at:
<http://www.redbooks.ibm.com/redpieces/abstracts/sg247304.html>

The team that wrote this redbook

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

Carla Sadtler is a certified IT Specialist at the ITSO, Raleigh Center. She writes extensively about the WebSphere and IBM Patterns for e-business areas. Before joining the ITSO in 1985, Carla worked in the Raleigh branch office as a Program Support Representative. She holds a degree in mathematics from the University of North Carolina at Greensboro.

Fabio Albertoni is an IBM Senior IT Specialist working at Integrated Technology Delivery SSO, in Hortolandia, Brazil. He has nine years of experience work in the IT industry and banks, and he has spent last five years developing and implementing integrated solutions using the WebSphere family, including WebSphere Application Server and WebSphere MQ. He hold a degree in data process from FATEC University of Ourinhos and a master degree in computer engineer from Instituto de Pesquisas Tecnologicas of Sao Paulo, Brazil.

Bernardo Fagalde is an IT Architect at IBM Uruguay and has worked for IBM since 2000. During his time at IBM, he has had many positions, including database administrator, system administrator, developer, designer, application server administrator, and finally as a technical lead for e-business projects. He has worked with WebSphere Application Server since V3.5 and mainly designs e-business solutions focused on using the WebSphere product family. He is currently the lead IT Architect on a large J2EE™ project. Bernardo holds a computing engineer degree from the Uruguayan main University (Universidad de la República Oriental del Uruguay).

Thiago Kleinubing is an IT Specialist in Brazil and has more than nine years of experience in the IT field. He has worked at IBM for the last six years and is currently a Team Leader for the IBM Global Business Services Organization - Total Workplace™ Experience Center of Excellence. His areas of expertise include the architecture, design, and development of J2EE applications. He is also an expert on IBM WebSphere Application Server, performance tuning, and problem determination. Thiago holds a degree in computer science and is certified in IBM Rational® Application Developer and WebSphere Studio v5.

Henrik Sjostrand is a Senior IT Specialist and has worked for IBM Sweden for 12 years. He is currently working as a technical consultant for the Nordic IBM Software Services for WebSphere team. The last six years, he has focused on J2EE application development, and WebSphere Application Server architecture, deployment, performance tuning, and troubleshooting. He is certified in WebSphere Application Server V4, V5, and V6, WebSphere Studio V5, and Rational Application Developer V6. Henrik holds a master of science in electrical engineering from Chalmers University of Technology in Gothenburg, Sweden, where he lives.

Ken Worland is a senior IT Specialist based in Melbourne, Australia. He specializes in Web services and messaging solutions and has more than 15 years experience in the IT field. His areas of expertise include IBM WebSphere Application Server, WebSphere MQ, DB2®, Oracle, and much more, having worked as a UNIX® system administrator and database administer on occasion. Ken holds a bachelor's degree in computer science from LaTrobe University in Melbourne.

Special thanks to the authors and contributors to the previous version of this book, *WebSphere Application Server V6 Planning and Design WebSphere Handbook Series*, SG24-6446:

Hernan Cunico
Leandro Petit
Michael Asbridge
Derek Botti
Venkata Gadepalli
William Patrey
Noelle Jakusz

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

Margaret Ticknor
International Technical Support Organization, Raleigh Center

Rich Conway
International Technical Support Organization, Raleigh Center

Mollie Tucker
IBM Intern from North Carolina State University

Daniel Tishman
IBM Intern from Penn State University

Peter Kovari
ITSO, Raleigh Center

Nicolai Nielsen
IBM Denmark

Klemens Haegele
IBM Germany

Partha Sarathy Momidi
IBM India

Sandhya Kapoor
IBM U.S.

Keys Botzum
IBM U.S.



Figure 1 Authors: (from left to right) Fabio Albertoni, Carla Sadtler, Thiago Kleinubing, Bernardo Fagalde, Ken Worland, Henrik Sjostrand

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

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

Archived



Introduction to WebSphere Application Server V6.1

IBM WebSphere is the leading software platform for On Demand Business. Providing comprehensive leadership, WebSphere is evolving to meet the demands of companies faced with challenging business requirements, such as the need for increasing operational efficiencies, strengthening client loyalty, and integrating disparate systems. WebSphere provides answers in today's challenging business environments.

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

In this chapter, we introduce WebSphere Application Server V6.1 for distributed platforms and WebSphere Application Server for z/OS V6.1.

1.1 Product overview

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

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

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

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

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

1.2 WebSphere Application Server

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

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

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

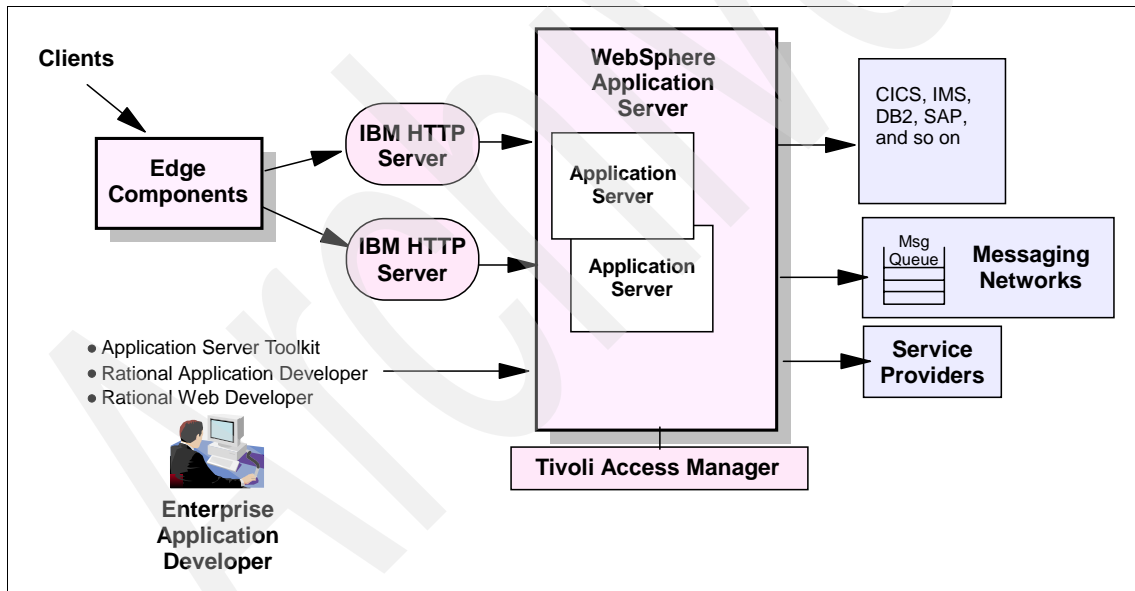


Figure 1-1 WebSphere Application Server product overview

The application server is the key component of WebSphere Application Server, providing the runtime environment for applications that conform to the J2EE 1.2, 1.3, and 1.4 specifications. Clients access these applications through standard interfaces and APIs. The applications, in turn, have access to a wide variety of external sources such as back-end systems, databases, Web services, and

messaging resources that can be used to process the client requests. Version 6.1 extends the application server to allow it to run JSR 168 compliant portlets and Session Initiation Protocol (SIP) applications written to the JSR 116 specification.

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

WebSphere Application Server supports asynchronous messaging through the use of a JMS provider and its related messaging system. WebSphere Application Server includes a fully integrated JMS 1.1 provider called the default messaging provider. This messaging provider complements and extends WebSphere MQ and 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 provides authentication and authorization capabilities to secure administrative functions and applications. Your choice of user registries include the operating system user registry, an LDAP registry (for example, IBM Tivoli® Directory Server), custom registries, file-based registries, or federated repositories. In addition to the default authentication and authorization capabilities, you have the option of using an external Java Authorization Contract for Containers (JACC)-compliant authorization provider for application security. The IBM Tivoli Access Manager client embedded in WebSphere Application Server is JACC-compliant and can be used to secure your WebSphere Application Server-managed resources. This client technology is designed to be used with the Tivoli Access Manager server (shipped with Network Deployment).

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

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

1.3 Packaging

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

WebSphere Application Server - Express V6

Note: WebSphere Application Server - Express V6.1 is anticipated to be announced later this year. This book specifically deals with V6.1 of Base and Network Deployment. When you see references to Express, they are specifically referring the V6.0 of Express.

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

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

WebSphere Application Server V6.1

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

This package includes two tools for application development and assembly:

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

In addition, the toolkit provides tools for the development, assembly, and deployment of JSR 116 SIP and JSR 168 portlet applications.

- ▶ This package also includes a trial version of Rational Application Developer, which supports the development, assembly, and deployment of J2EE 1.4 applications.

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

WebSphere Application Server Network Deployment V6.1

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

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

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

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

WebSphere Application Server V6.1 for z/OS

IBM WebSphere Application Server for z/OS is a full-function version of the Network Deployment product. WebSphere Application Server for z/OS can support On Demand Business on any scale.

Packaging summary

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

Table 1-1 WebSphere Application Server packaging

Features included	Express V6.0 ^a	Base V6.1	Network Deployment V6.1	V6.1 for z/OS
WebSphere Application Server	Yes	Yes	Yes	Yes
Deployment manager	No	No	Yes	Yes
Web server plug-ins	Yes	Yes	Yes	Yes
IBM HTTP Server	Yes	Yes	Yes	Yes
Application Client (not available on Linux® for zSeries®)	Yes	Yes	Yes	Yes
Application Server Toolkit	Yes	Yes	Yes	Yes
DataDirect Technologies JDBC™ Drivers for WebSphere Application Server	Yes	Yes	Yes	Yes (for Microsoft® Windows® only)
Rational Development tools	Rational Web Developer (single use license)	Rational Application Developer Trial	Rational Application Developer Trial	Rational Application Developer Trial (non-z/OS platforms)

Features included	Express V6.0 ^a	Base V6.1	Network Deployment V6.1	V6.1 for z/OS
Database	IBM DB2 Universal Database™ Express V8.2	IBM DB2 Universal Database Express V8.2 (development use only)	IBM DB2 UDB Enterprise Server Edition V8.2 for WebSphere Application Server Network Deployment	No
Production ready applications	IBM Business Solutions	No	No	No
Tivoli Directory Server for WebSphere Application Server (LDAP server)	No	No	Yes	No
Tivoli Access Manager Servers for WebSphere Application Server	No	No	Yes	Yes (non-z/OS platforms)
Caching Proxy and Load Balancer Edge components	No	No	Yes	Yes (non-z/OS platforms)

a. Express is limited to a maximum of two CPUs.

WebSphere Application Server includes a new tool called Installation Factory for creating customized install packages (CIPs). Consider using Installation Factory to create one or more CIPs and use those CIPs to deploy or update WebSphere throughout your organization.

WebSphere Application Server also now ships the Update Installer (UPDI) for installing maintenance (fix packs, interim fixes, and so on. In previous versions, these tools were only available as separate Web downloads.

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

1.4 Supported platforms and software

The following tables illustrate the platforms, software, and versions that WebSphere Application Server V6.1 supports at the time of the writing of this document.

For the most up-to-date operating system levels and requirements, refer to the WebSphere Application Server system requirements Web page, at:

<http://www.ibm.com/software/websevers/appserv/doc/latest/prereq.html>

1.4.1 Operating systems

Table 1-2 shows the supported operating systems and versions for WebSphere Application Server V6.1.

Table 1-2 Supported operating systems and versions

Operating systems	Versions
Microsoft Windows	<ul style="list-style-type: none">▶ Microsoft Windows 2000 Advanced Server with SP4▶ Microsoft Windows 2000 Server with SP4▶ Microsoft Windows 2000 Professional Server with SP4▶ Microsoft Windows Server® 2003 (Datacenter with SP1)▶ Microsoft Windows Server 2003 (Enterprise with SP1)▶ Microsoft Windows Server 2003 (Standard with SP1)▶ Microsoft Windows XP Professional with SP2▶ Microsoft Windows Server 2003 x64 Editions
IBM AIX® 5L™	<ul style="list-style-type: none">▶ AIX 5L Version 5.2 Maintenance Level 5200-07▶ AIX 5L Version 5.3 with Service Pack 5300-04-01
Sun™ Solaris™	<ul style="list-style-type: none">▶ Solaris 9 with the latest patch Cluster▶ Solaris 10 with the latest patch Cluster
HP-UX	<ul style="list-style-type: none">▶ HP-UX 11iv2 (11.23) with the latest Quality Pack
Linux (Intel®)	<ul style="list-style-type: none">▶ Red Hat Linux Enterprise AS, ES, WS V3 with Update 5 or 6▶ Red Hat Linux Enterprise AS, ES, WS V4 with Update 2▶ SUSE Linux Enterprise Server V9 with SP2 or 3
Linux (Power PC®)	<ul style="list-style-type: none">▶ Red Hat Enterprise Linux AS V3 with Update 5 or 6▶ Red Hat Enterprise Linux AS V4 with Update 2▶ SUSE Linux Enterprise Server V9 with SP2 or 3
Linux on IBM System z™ (Supported for WebSphere Application Server Network Deployment only)	<ul style="list-style-type: none">▶ Red Hat Enterprise Linux AS V3 with Update 5 or 6▶ Red Hat Enterprise Linux AS V4 with Update 2▶ SUSE Linux Enterprise Server V9 with SP2 or 3

Operating systems	Versions
IBM i5/OS and OS/400®	<ul style="list-style-type: none"> ▶ i5/OS and OS/400, V5R3 ▶ i5/OS V5R4
z/OS (Supported for WebSphere Application Server Network Deployment only)	<ul style="list-style-type: none"> ▶ z/OS 1.6 or later ▶ z/OS.e 1.6 or later

1.4.2 Web servers

All available platforms of WebSphere Application Server V6.1 support the following Web servers:

- ▶ Apache HTTP Server 2.0.54
- ▶ IBM HTTP Server for WebSphere Application Server 6.0.2
- ▶ IBM HTTP Server for WebSphere Application Server 6.1
- ▶ Internet Information Services 5.0
- ▶ Internet Information Services 6.0
- ▶ IBM Lotus® Domino® Enterprise Server 6.5.4 or 7.0
- ▶ Sun Java™ System Web Server 6.0 SP9
- ▶ Sun Java System Web Server 6.1 SP3

1.4.3 Database servers

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

Table 1-3 Supported database servers and versions

Databases	Versions
IBM DB2	DB2 for iSeries™ 5.2, 5.3, or 5.4 DB2 for z/OS v7 or v8 DB2 Enterprise Server Edition 8.2 FP4 DB2 Express 8.2 FP4 DB2 Workgroup Server Edition 8.2 FP4
Cloudscape™	Cloudscape 10.1 (Derby)
Oracle	Oracle 9i Standard/Enterprise Release 2 - 9.2.0.7 Oracle 10g Standard/Enterprise Release 1 - 10.1.0.4 Oracle 10g Standard/Enterprise Release 2 - 10.2.0.1 or 10.2.0.2

Databases	Versions
Sybase	Sybase Adaptive Server Enterprise 12.5.2 or 15.0
Microsoft SQL Server	Microsoft SQL Server Enterprise 2000 SP4 Microsoft SQL Server Enterprise 2005
Informix®	Informix Dynamic Server 9.4C7W1 or 10.00C4
IMS™	IMS V8 or V9
WebSphere Information Integrator	WebSphere Information Integrator V8.2 FP4

1.4.4 Directory servers

Table 1-4 shows the LDAP servers that WebSphere Application Server V6.1 supports.

Table 1-4 Supported directory servers and versions

Directory server	Versions
IBM Tivoli Directory Server	5.2 and 6.0
z/OS Security Server	1.6 and 1.7
z/OS.e Security Server	1.6 and 1.7
Lotus Domino Enterprise Server	6.5.4 and 7.0
Sun ONE™ Directory Server	5.1 SP4 and 5.2
Windows Active Directory®	2003 and 2000
Novell eDirectory	8.7.3 and 8.8

Archived

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 that provide enhanced security and messaging options.

This chapter includes the following sections:

- ▶ Tivoli Access Manager
- ▶ Tivoli Directory Server
- ▶ WebSphere MQ integration
- ▶ Information Integration

2.1 Tivoli Access Manager

IBM Tivoli Access Manager provides a more holistic security solution at the enterprise level than the standard security providing mechanisms found in WebSphere Application Server. The following sections give an overview of built-in WebSphere Application Server security, how WebSphere Application Server integrates with Tivoli Access Manager, and when and why the two products might be used together.

For more information about Tivoli Access Manager, see:

- ▶ Tivoli Access Manager for e-business home page
<http://www.ibm.com/software/tivoli/products/access-mgr-e-bus/>
- ▶ Tivoli Access Manager Information Center
<http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.itame.doc/toc.xml>

2.1.1 WebSphere Application Server security

WebSphere Application Server V6.1 provides its own security infrastructure. This infrastructure is composed of some mechanisms that are specific to WebSphere Application Server but also many that use open standards security technologies. This security technology is widely proven, and the software can integrate with other enterprise technologies easily.

A brief overview of WebSphere security

The rich, standards-based architecture for WebSphere Application Server security offers various configuration options:

- ▶ At the most basic level, a single server can use the Simple WebSphere Authentication Mechanism (SWAM). However, the SWAM mechanism has been deprecated and will be removed in future versions, so we strongly recommend that you to plan your security to use a different authentication mechanism.
- ▶ If more than one server is required to share a security mechanism (that is if containers on more than one server need to be able to track user credentials across the servers), you can use Lightweight Third Party Authentication (LTPA). (It can also be used for single servers.) To make this possible, LTPA generates a security token that is passed between servers for authenticated users.

- ▶ Reverse proxy servers (servers that mediate between Web clients and multiple servers behind a firewall) can be integrated with LTPA in WebSphere Application Server to allow for client authentication. A trust association is implemented, which is a contract between the application server and the reverse proxy server. IBM WebSEAL Reverse Proxy is such a reverse proxy product.
- ▶ An operating system, LDAP or customer *user registry* is configured to be the user registry for the environment. Only one registry can be configured at any one time. V6.1 introduces a file-based user registry and the ability to federate this registry with other registries. The file-based registry is the default for administrative security enabled out of the box.
- ▶ WebSphere uses the Java Authentication and Authorization Service (JAAS) API, which enables services to authenticate and to enforce access controls on users.

WebSphere uses a specialized JAAS module to implement the user credentials mapping module in its J2EE Connector Architecture (J2C) implementation that enables WebSphere Application Server to integrate with enterprise information systems.

- ▶ Java 2 Security can also be enabled. This security mechanism, which is part of the Java runtime, allows or disallows access for Java code to specific system resources based on permissions, which can be specified in a fine-grained manner. For example, a Java application can be granted permission to access the operating system's file system for file input and output.
- ▶ J2EE 1.4 prescribes the use of the JACC API specification. (The relevant Java Community Process Java Specification Request is JSR-115.) JACC allows application servers to interact with third-party authorization providers (such as Tivoli Access Manager) through standard interfaces to make authorization decisions. Previously, proprietary interfaces for third-party authorization providers had to be used.

2.1.2 Tivoli Access Manager and WebSphere Application Server

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

The advantages at the enterprise level of using this approach are:

- ▶ 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

Repositories

As with WebSphere Application Server security, Tivoli Access Manager requires a user repository. It supports many different repositories such as Microsoft Active Directory, iPlanet™, and IBM Tivoli Directory Server. Tivoli Access Manager can be configured to use the same user repository as WebSphere Application Server, enabling you to 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, which contains the security policy information for all resources and all credentials information of all participants in the secure domain, both users and servers. The authorization database is then replicated across all local authorization servers. IBM WebSEAL Reverse Proxy Server, for example, has its own local authorization server.

Tivoli Access Manager for WebSphere component

The Tivoli Access Manager clients are embedded in WebSphere Application Server. The Tivoli Access Manager client can be configured using the scripting and GUI management facilities of WebSphere Application Server.

The Tivoli Access Manager server is bundled with WebSphere Application Server Network Deployment. Tivoli Access Manager further integrates with WebSphere Application Server in that it supports the special subjects AllAuthenticated and Everyone.

Note: AllAuthenticated and Everyone are subjects that are specific to WebSphere Application Server. These special categories allow access to a resource to be granted to all those users who have been authenticated regardless of what repository user groups they might belong to and allow access to be granted to all users whether or not they are authenticated.

All communication between the Tivoli Access Manager clients and the Tivoli Access Manager server is done through the JACC API.

Figure 2-1 shows the integration interfaces between WebSphere Application Server and Tivoli Access Manager.

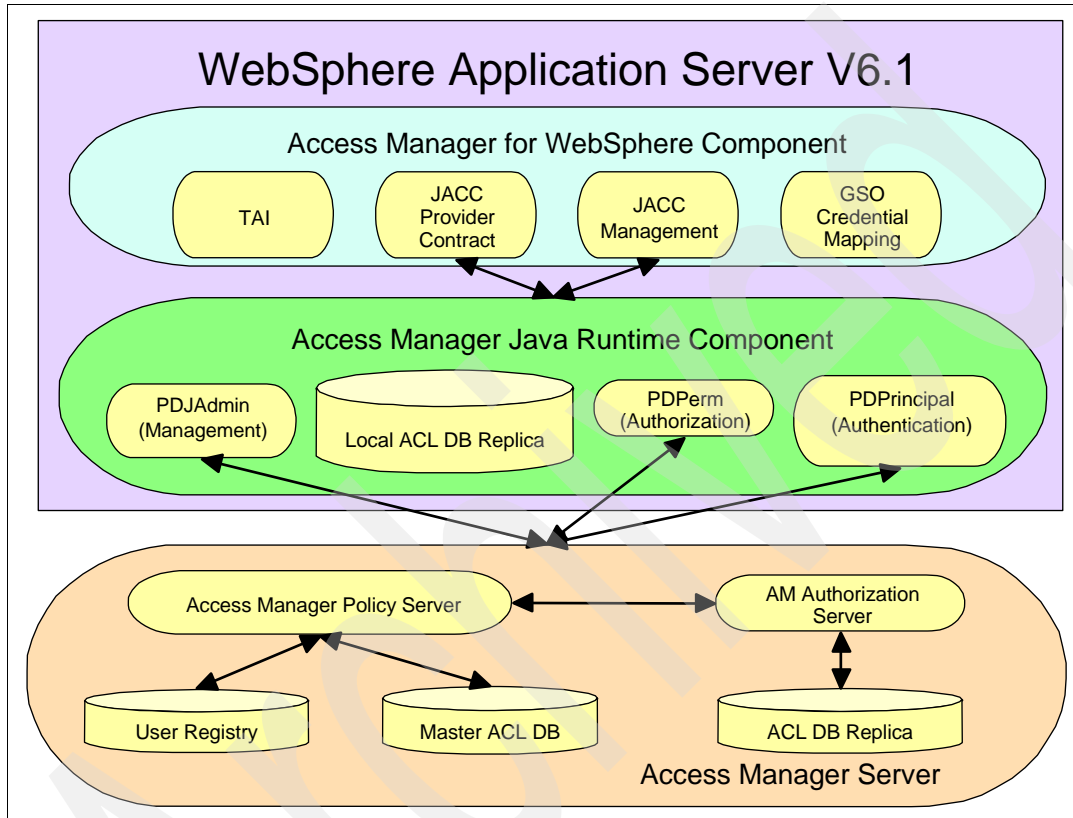


Figure 2-1 Integration of WebSphere Application Server with Tivoli Access Manager

Further advantages of using Tivoli Access Manager

We already reviewed the enterprise level advantages of using Tivoli Access Manager. Using Tivoli Access Manager at the application server level has the following further advantages:

- ▶ Supports accounts and password policies.
- ▶ Supports dynamic changes to the authorization table without having to restart the applications.
- ▶ Provides tight integration with WebSphere Application Server.

Security, networking, and topology considerations

Clearly, because the LDAP server contains and the Access Manager server manages sensitive data in terms of authentication, authorization and privacy, the servers belong in the data layer of the network. It is best practice to enable Secure Sockets Layer (SSL) configuration options between the databases so that the data is encrypted.

Legal considerations (privacy and data protection)

You should be aware that there might be some legal and or regulatory issues that surround storing of certain data types, such as personally identifiable data in the European Union, on IT systems. Ensure that you have consulted your legal department before deploying such information about your systems. These considerations vary by geography and industry, and it is beyond the scope of this book to discuss specific issues.

2.2 Tivoli Directory Server

This section describes IBM Tivoli Directory Server and its integration with WebSphere Application Server.

Note: IBM SecureWay® Directory Server has been renamed to IBM Tivoli Directory Server in WebSphere Application Server Version 6.1.

For more information about IBM Tivoli Directory Server, see the following Web site:

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

2.2.1 The Lightweight Directory Access Protocol (LDAP)

A directory is a data structure that enables the look up of names and associated attributes arranged in a hierarchical tree structure. In the context of enterprise application servers, this enables applications to look up a user principal and determine what attributes the user has and of which groups the user is a member. Decisions about authentication and authorization can then be made using this information.

LDAP is a fast and simple way of looking up user entities in a hierarchical data structure. It has advantages over simply 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, common name, and so forth. The schema can also have custom objects added to it, which means 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 IBM Tivoli Directory Server) use database engines under the covers, but these engines are optimized for passive lookup performance (through indexing techniques). This 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.

Today, there are many LDAP server implementations. For example, IBM Tivoli Directory Server, iPlanet Directory Server, Open LDAP's SLAPD server, and Microsoft Active Directory all support the LDAP protocol.

For a list of supported directory servers, see 1.4.4, "Directory servers" on page 11.

2.2.2 Tivoli Directory Server and WebSphere Application Server

When you enable application security in WebSphere Application Server, you must select the user registry to be used (in this case, an LDAP registry). This can be done through the WebSphere administrative console or through the `wsadmin` command line tool.

Because the LDAP server contains sensitive data in terms of authentication, authorization, and privacy, the LDAP server belongs in the data layer of the network. It is a best practice to enable SSL options in the WebSphere Application Server security configuration so that the data is encrypted between the application server layer and the data layer.

There might be some legal and or regulatory issues that surround storing of certain data types, such as personally identifiable data in the European Union, on IT systems. Ensure that you have consulted your legal department before deploying such information about your systems. These considerations vary by geography and industry, and it is beyond the scope of this book to discuss specific issues. Legal considerations might become even more of an issue when you create custom objects and attributes in the LDAP directory schema that can store further information relating to individuals.

2.3 WebSphere MQ integration

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

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

With the advent of 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:

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

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 Java Message Service (JMS) programming interface and the use of a JMS provider and its related messaging system. JMS providers must conform to the JMS Specification version 1.1.

In WebSphere Application Server V6, you can use the following as JMS providers:

- ▶ The default messaging provider
- ▶ WebSphere MQ
- ▶ Generic JMS providers
- ▶ V5 default messaging provider (for migration purposes)

The default messaging provider is the JMS API implementation for messaging (connection factories, JMS destinations, and so on). 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 will have one (or possibly more in the case of clusters) messaging engine 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 V6 messaging engine.

Whether to use the default messaging provider, the direct WebSphere MQ messaging provider, or a combination depends on a number of factors. There is no set of questions that can lead you directly to the decision; however, consider the following guidelines.

In general, the default messaging provider is a good choice if:

- ▶ 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.
- ▶ WebSphere Application Server can support the topology required for scalability.

The WebSphere MQ messaging provider is good choice if:

- ▶ You are currently using a WebSphere MQ messaging provider and simply want to continue using it.
- ▶ You require access to heterogeneous, non-JMS EIS systems.
- ▶ You require access to WebSphere MQ clustering.

Using a topology that combines WebSphere MQ and the default messaging provider gives you the benefit of the tight integration between WebSphere and the default messaging provider (clustering), and the flexibility of WebSphere MQ.

Connecting WebSphere Application Server to WebSphere MQ

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

- ▶ Extend the WebSphere MQ and service integration bus networks 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 sees the connected service integration bus as a queue manager. The service integration bus sees the WebSphere MQ network as another service integration bus.

WebSphere MQ applications can send messages to queue destinations on the service integration bus and default messaging applications can send messages to WebSphere MQ queues without being aware of the mixed topology. As with WebSphere MQ queue manager networks, this mechanism 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.

Note that:

- WebSphere MQ to service integration bus connections are only supported 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 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.

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. 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.

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 as long as another is available in the bus,

increasing robustness and availability. This mechanism 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 in the case 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.

Archived

Archived

Planning for infrastructure

Wondering about how to plan and design an infrastructure deployment that is based on WebSphere middleware? This chapter describes the WebSphere-specific components that you have to understand in order to run a successful WebSphere infrastructure project. This chapter contains the following sections:

- ▶ Infrastructure deployment planning
- ▶ Design for scalability
- ▶ Sizing
- ▶ Benchmarking
- ▶ Performance tuning
- ▶ Planning for backup and recovery

3.1 Infrastructure deployment planning

This section gives a general overview of the typical phases you have to go through during a project, how to gather requirements, and how to apply those requirements to a WebSphere project.

Typically, a new project starts with only a concept. Very little is known about specific implementation details, especially as they relate to the infrastructure. Hopefully, your development team and infrastructure team work closely together to bring scope to the needs of the overall application environment.

Bringing together a large team of people can create an environment that helps hone the environment requirements. If unfocused, however, a large team can be prone to wander aimlessly and to create more confusion than resolving issues. For this reason, carefully consider the size of the requirements team and try to keep the meetings as focused as possible. Provide template documents to be completed by the developers, the application business owners, and the user experience team.

Try to gather information that falls into the following categories:

- ▶ Functional requirements, which are usually determined by the business use of the application and are related to function
- ▶ Non-functional requirements that describe the properties of the underlying architecture and infrastructure such as reliability, availability, or security
- ▶ Capacity requirements, including traffic estimates, traffic patterns, and expected audience size

Requirements gathering is an iterative process. There is no way, especially in the case of Web-based applications, to have absolutes for every item. The best you can do is create an environment that serves your best estimates, and then monitor the environment closely to adjust as necessary after launch. Make sure that all your plans are flexible enough to deal with future changes in requirements, and always keep in mind that the plans can impact other parts of the project.

With this list of requirements, you can start to create the first drafts of your designs. Target developing at least 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.

This book does not attempt to cover the specifics of a software development cycle. There are multiple methodologies for application design and volumes dedicated to best practices.

► Implementation design

This design defines the target deployment infrastructure on which your application will be deployed.

The final version of this implementation design will contain details about the hardware, processors, and software that will be installed. However, you do not begin with all these details. Initially, your implementation design simply lists component requirements, such as a database, a set of application servers, a set of Web servers, and whatever other components are defined in the requirements phase.

This design might need to be extended during your project, whenever a requirement for change occurs or when you get new sizing information. Too often, however, the reality is that a project can require new hardware and, therefore, might be constrained by capital acquisition requirements. A good initial implementation design can reduce the chance that you will need to constantly ask for additional resources after the project has been accepted.

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. We describe this exercise in sizing in 3.3, “Sizing” on page 29.

In some cases, it might be appropriate to do benchmark tests. There are many ways to perform benchmarking tests, and in 3.4, “Benchmarking” on page 30, we describe some of these methods.

The last step in every deployment is to tune your system and measure whether it can handle the projected load that your non-functional requirements specify. For more details about how to plan for load tests, see 3.5, “Performance tuning” on page 32.

3.2 Design for scalability

Understanding the scalability of the components in your e-business infrastructure and applying appropriate scaling techniques can greatly improve availability and performance. Scaling techniques are especially useful in multi-tier architectures, when you want to evaluate components that are associated with IP load balancers, such as dispatchers or edge servers, Web presentation servers, Web application servers, data servers, transaction servers, and LPARs in a z/OS environment.

You can use the following steps to classify your Web site and to identify scaling techniques that are applicable to your environment:

1. Understand the application environment.

Applications are key to the scalability of the infrastructure. It is important to understand the component flow and traffic volumes associated with existing applications and to evaluate the nature of new applications. Different types of applications represent different workload patterns. For example, online banking applications might experience the greatest delay at the database server, while other application applications might experience the greatest delays at the application server.

2. Categorize your workload.

Knowing the workload pattern for a site determines where you should 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 would 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.

Web sites with similar workload patterns can be classified into site types, for example:

- Publish/subscribe
- Online shopping
- Customer self-service
- Online trading
- Business to business

3. Determine the components most affected.

This step involves mapping the most important site characteristics to each component. Once again, 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.

4. Select the scaling techniques to apply.

When the information gathering is as complete as it can be, it is time to consider matching scaling techniques to components. Manageability, security, and availability are critical factors in all design decisions. Do not use techniques that provide scalability but that compromise any of these critical factors.

The scaling techniques are:

- Using a faster machine
- Creating a cluster of machines
- Using appliance servers

- Segmenting the workload
 - Using batch requests
 - Aggregating user data
 - Managing connections
 - Using caching techniques
5. Apply the techniques.
- Testing is key to successful application of these techniques. It is crucial that you determine not only if the scaling techniques are effective but that they do not adversely affect other areas. Only when you are satisfied that you have achieved the desired results should you move into production.
6. Reevaluate.
- Recognize that any system is dynamic. The initial infrastructure will at some point need to be reviewed and possibly expanded. Changes in the nature of the workload can create a need to reevaluate the current environment. Large increases in traffic will require examination of the machine configurations. As long as you understand that scalability is not a one time design consideration, that instead it is part of the growth of the environment, you will be able to keep a system resilient to changes and avoid possibly negative experiences due to a poorly planned infrastructure.

The article *Design for Scalability - an Update* provides a detailed discussion about these steps:

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

We expand on some of the scaling techniques in Chapter 9, “Planning for performance, scalability, and high availability” on page 169. The following IBM Redbooks cover scalability and high availability techniques for WebSphere Application Server V6:

- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- ▶ *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688

3.3 Sizing

After determining the application design and scalability techniques, you need to determine the number of machines required for the project. It is a given that the application design will evolve over time and sizing is usually done in the early stages of design. However, when sizing, it is important that you have a static

version of the application design with which to work. The better view you have of the application design, the better your sizing estimate will be.

You should also consider which hardware platforms you want to use. This decision is primarily dependent on your platform preference, which platforms have sizing information available, and which platforms WebSphere Application Server supports. Hardware decisions might also be driven by the availability of hardware that forces a limitation in the operating systems that can be deployed.

Next, determine whether you want to scale up or out. Scaling up means to do vertical scaling on a small number of machines with many processors. This can present fairly significant single points of failure. Scaling out, however, means using a larger number of smaller machines. This can generally be thought of as significantly more resilient, because it is unlikely that the failure of one small server is adequate to create a complete application outage. However, you would have to support and maintain many small machines. This is usually a decision of preference and cost for your environment. However, the reality is that application resiliency issues can change your view.

What you need to understand, though, is that the sizing estimates are solely based on your input, which means the better the input, the better the results. Sizing work assumes an average standard of application performance behavior and an average response time is assumed for each transaction. Calculations based on this are performed to determine the estimated number of machines and processors your application will require. If your enterprise has a user experience team, they might have documented standards for a typical response time that your new project will be required to meet.

If you need a more accurate estimation of your hardware requirements and you already have your application, consider using one of the benchmarking services discussed in 3.4, “Benchmarking” on page 30.

Based on your estimate, you might have to update your production implementation design and the designs for the integration and development environments accordingly. Changes to the production environment should be incorporated into the development and testing environments if at all possible.

3.4 Benchmarking

Benchmarking is the process used to take an application environment and determine the capacity of that environment through load testing. This determination enables you to make reasonable judgements as your environment begins to change. Using benchmarks, you can determine the current work

environment capacity and set expectations as new applications and components are introduced.

Benchmarking is primarily interesting to two kinds of clients:

- ▶ Clients who already have an application and want to migrate to a new version of WebSphere or who want to evaluate the exact number of machines for their target deployment platform.
- ▶ Clients who sell products that are based on WebSphere and who want to provide sizing estimations for their products.

Many sophisticated enterprises maintain a benchmark of their application stack and change it after each launch or upgrade of a component. These customers usually have well-developed application testing environments and teams dedicated to the cause. For those that do not, alternatives are available such as the IBM Test Center. There are also third-party benchmark companies that provide this service. When choosing, make sure that the team that performs the benchmark tests has adequate knowledge of the environment and a clear set of goals. This helps to reduce the costs of the benchmark tests and creates results that are much easier to quantify.

IBM Test Center

IBM Global Services offers you the ability to retain IBM for Performance Management, Testing, and Scalability services. This team will come to a customer site and assess the overall site performance. This investigation is platform neutral, with no sales team poised to sell additional hardware as a result of the analysis. Offerings include, but are not limited to:

- ▶ Testing and Scalability Services for TCP/IP networks
- ▶ Testing and Scalability Services for Web site stress analysis
- ▶ Performance Engineering and Test Process Consulting
- ▶ Performance Testing and Validation
- ▶ Performance Tuning and Capacity Optimization

When using a service such as those provided by the IBM Test Center, you are presented with large amounts of supporting documentation and evidence to support the results of the tests. You can then use this data to revise your current architecture or possibly just change the overall infrastructure footprint to add additional machines or to correct single points of failure.

These offerings can be purchased through your local IBM account representative.

3.5 Performance tuning

Performance is one of the most important non-functional requirements for any WebSphere environment. Application performance should be tracked continuously during your project.

Imagine your project is finished, you switch your environment to production, and your environment is unable to handle the user load. This is 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.

3.5.1 Application design problems

Many performance problems cannot be fixed by using more hardware or changing WebSphere parameters. Therefore, you really want to make performance testing (and tuning) part of your development and release cycles. Otherwise, it takes much more effort and money to correct issues after they occurred in production than to fix them up front. If performance testing is part of your development cycle, you are able to correct issues with your application design much earlier, resulting in fewer issues when using your application on the production environment.

3.5.2 Understand your requirements

Without a clear understanding of your requirements, you have no target that you can tune against. It is important when doing performance tuning to know your objectives. Do not waste time trying to do performance tuning on a system that was improperly sized and cannot withstand the load, no matter how long you tune it. Also, do not continue tuning your system when you are already beyond your performance targets.

Understanding your requirements demands knowledge of two things:

- ▶ Non-functional requirements
- ▶ Sizing

If you do not have this information and you are asked to tune a system, you will either fail or will not know when to stop tuning.

3.5.3 Test environment setup

When executing performance tests, follow these general tips throughout all your tests:

- ▶ Execute your tests in an environment that mirrors the production server state. Using an environment that is as close in nature as possible to the production environment enables you to extrapolate the test results to the production environment.
- ▶ Make sure that nobody is using the test machines 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 the backup is running. It is acceptable to run monitoring software in the background that will also run in production.
- ▶ Check for processor, memory, and disk utilization before and after each test run to see if there are any unusual patterns. If the target environment will be using shared infrastructure (messaging servers or authentication providers, for example) try to make sure the shared component is performing under the projected shared load.
- ▶ Isolate network traffic as much as possible. Using switches, there is rarely a circumstance where traffic from one server overruns the port of another. It is possible, however, to flood ports used for routing off the network to other networks or even the switch backbone for very heavy traffic. Make sure that your network is designed in a manner that isolates the testing environment as much as possible prior to starting, because performance degradation of the network can create unexpected results.

We describe test environments in more detail in Chapter 7, “Planning for application development and deployment” on page 123.

3.5.4 Load factors

The most important factors that determine how you conduct your load tests are:

- ▶ Request rate
- ▶ Concurrent users
- ▶ Usage patterns

This is not a complete list and other factors can become more important depending on the kind of site that is being developed.

Usage patterns

At this point in the project, it is very important that you think about how your users will use the site. You might want to use the use cases that your developers

defined for their application design as an input to build your usage patterns. This makes it easier to build the scenarios later that the load test would use.

Usage patterns consist of:

- ▶ Use cases modeled as click streams through your pages
- ▶ Weights applied to your use cases

Combining weights with click streams is very important because it shows you how many users you expect in which of your application components and where they generate load. After all, it is a different kind of load if you expect 70% of your users to search your site instead of browsing through the catalog than the other way around. These assumptions also have an impact on your caching strategy.

Make sure that you 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, we recommend that you 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.

3.5.5 Production system tuning

This is the only environment that impacts the user experience for your customers. This is where you apply all the performance, scalability, and high availability considerations in production. Tuning this system is an iterative process that involves optimizing WebSphere parameters to suit your runtime environment.

Important: To perform tuning in the production environment, you should have the final version of code running. This version should have passed performance tests on the integration environment prior to changing any WebSphere parameters on the production system.

When changing a production environment, use some standard practices:

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

Changes between test runs should not differ by more than a small percentage to preclude introducing new problems that you might need to sort out before you continue tuning.

As soon as you finish tuning your production systems, apply the settings to your test environments to make sure that they are similar to production. Plan to rerun your tests there to establish new baselines on these systems and to see how these changes affect the performance.

Keep in mind that you often have only one chance to get this right. Normally, as soon as you are in production with your system, you cannot run performance tests on this environment any more, simply 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, and you have already lost half the battle.

Note: Because it is rare to use a production system for load tests, it is usually a bad idea to migrate these environments to new WebSphere versions without doing a proper test on an equivalent test system or new hardware.

After completing your first performance tests on your production systems and tuning the WebSphere parameters, evaluate your results and compare them to your objectives to see how all of this worked out for you.

3.5.6 Conclusions

There are various possible outcomes from your performance tests that you should clearly understand and act upon:

- ▶ Performance meets your objectives.

Congratulations! However, do not stop here. Make sure that you have planned for future growth and that you are meeting all your performance goals. After that, we recommend documenting your findings in a performance tuning report and archiving it. Include all the settings that you changed to reach your objectives.

This report is useful when you set up a new environment or when you have to duplicate your results somewhere else on a similar environment with the same application. This data also is essential when adding additional replicas of some component in the system, because they need to be tuned to the same settings the current resources use.

- ▶ Performance is slower than required.

Your application performance is somewhat slower than expected, and you have already done all possible application and WebSphere parameter tuning. You might need to add more hardware (for example, increase memory, upgrade processors, and so forth) to those components in your environment that showed to be bottlenecks during your performance tests. Then, run the tests again. Verify with the appropriate teams that there were no missed bottlenecks in the overall system flow.

- ▶ Performance is significantly slower than required.

In this case, you should start over with your sizing and ask the following questions:

- Did you underestimate any of the application characteristics during your initial sizing? If so, why?
- Did you underestimate the traffic and number of users/hits on the site?
- Is it still possible to change parts of the application to improve performance?
- Is it possible to obtain additional resources?

After answering these questions, you should have a better understanding about the problem that you face right now. Your best bet is to analyze your application and try to find the bottlenecks that cause your performance problems. Tools such as the Profiler that is part of Rational Application Developer can help you with this. For further information about profiling tools, refer to *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

3.6 Planning for backup and recovery

In general, computer hardware and software is very reliable, but sometimes failures can occur and damage a machine, network device, software product, configuration, or more importantly, business data. It is important to plan for such occurrences. There are a number of stages to creating a backup and recovery plan, which the following sections discuss.

3.6.1 Risk analysis

The first step to creating a backup and recovery plan is to complete a comprehensive risk analysis. The goal is to discover which areas are the most critical and which hold the greatest risk. It is also important to identify which

business processes are the most important and how they will be affected by application outages.

3.6.2 Recovery strategy

When the critical areas have been identified, you need to develop a strategy for recovering those areas. There are numerous backup and recovery strategies available which vary in recovery time and cost. In most cases, the cost increases as the recovery time decreases. The key to developing 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.

3.6.3 Backup plan

With your recovery strategy, a backup plan needs to be created to handle the daily backup operations. There are numerous backup methods varying in cost and effectiveness. A *hot backup site* provides real-time recovery by automatically switching to a whole new environment quickly and efficiently. Because of the cost, only the largest sites use hot backup. For less critical applications, *warm* and *cold backup sites* can be used. These are similar to hot backup sites, but are less costly and effective. More commonly, sites use a combination of backup sites, load-balancing, and high availability.

More common backup strategies combine replication, shadowing, and remote backup, as well as the more mundane methods such as tape backup or Redundant Array of Independent Disks (RAID) technologies. All of which are just as viable as a hot backup site but require longer restore times.

Any physical backup must be stored at a remote location in order to be able to recover from a disaster, such as a fire. New technologies make remote electronic vaulting a viable alternative to physical backups and many third-party vendors offer this service.

3.6.4 Recovery plan

If a disaster does occur, a plan for restoring operations as efficiently and quickly as possible must be in place. The recovery plan must be coordinated with the backup plan to ensure that the recovery happens smoothly. The appropriate response must be readily available so that no matter what situation occurs, the site will be operational at the agreed upon disaster recovery time. A common practice is to rate outages from minor to critical and have a corresponding response. For example, a hard disk failure could be rated as a Class 2 outage

and have a Class 2 response where the disk gets replaced and replicated with a 24-hour recovery time. This makes recovering easier because resources are not wasted and disaster recovery time is optimized.

Another key point of the recovery plan must address what happens after the recovery. Minor disasters, such as disk failure, have little impact afterward but critical disasters, such as the loss of the site, have a significant impact. For example, if a hot backup site is used, the recovery plan must account for the return to normal operation. New hardware or possibly a whole new data center might need to be created. Post-disaster activities need to be completed quickly to minimize costs.

3.6.5 Update and test process

You must revise the backup and recovery plan on a regular basis to ensure that the recovery plan meets your current needs. You also need to test the plan on a regular basis to ensure that the technologies are functional and that the personnel involved know their responsibilities. In addition to the regular scheduled reviews, review the backup and recovery plan when adding new hardware, technologies, or personnel.

WebSphere Application Server concepts

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

This chapter contains the following sections:

- ▶ WebSphere Application Server concepts
- ▶ Distributed server environments
- ▶ Application server clusters
- ▶ Runtime processes
- ▶ Using Web servers

4.1 WebSphere Application Server concepts

WebSphere Application Server is organized based on the concept of cells, nodes, and servers. While all of these elements are present in each configuration, cells and nodes do not play an important role until you take advantage of the features provided with Network Deployment.

The *application server* is the primary runtime component in all configurations and is where an application executes. All WebSphere Application Server configurations can have one or more application servers. In the Express and Base configurations, each application server functions as a separate entity. There is no workload distribution or central administration among application servers. With Network Deployment, you can build a distributed server environment consisting of multiple application servers maintained from a central administration point. In a distributed server environment, you can cluster application servers for workload distribution.

Runtime environments are built by creating *profiles*. Each profile contains files specific to that runtime such as logs and configuration files. Profiles can be created during or after installation, or both. After creating the profiles, use the WebSphere administrative tools for further configuration and administration. Each profile is stored in a unique directory path selected at profile creation time. The default is for the profiles to be stored in a subdirectory of the installation directory, but they can be located anywhere. All profiles share the product binaries.

4.1.1 Stand-alone application servers

All 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 J2EE applications and provides the services required to run those applications. Each stand-alone server is created by defining an *application server profile*.

Multiple stand-alone application servers can exist on a machine, either through independent installations of the WebSphere Application Server code or by creating multiple application server profiles within one installation. However, WebSphere Application Server does not provide centralized management or administration for multiple stand-alone application servers. Stand-alone application servers do not provide workload management or failover capabilities.

4.1.2 Distributed application servers

With Network Deployment, you can build a *distributed server* configuration to enable central administration, workload management, and failover. In this environment, you integrate one or more application servers into a cell that is managed by a *deployment manager*. The application servers can reside on the same machine as the deployment manager or on multiple separate machines. Administration and management is handled centrally from the administration interfaces by the deployment manager.

With a distributed server configuration, you can create multiple application servers to run unique sets of applications and then manage those applications from a central location. However, more importantly, you can *cluster* application servers to allow for workload management and failover capabilities. Applications that you install in the cluster are replicated across the application servers. When one server fails, another server in the cluster continues processing. Workload is distributed among Web and EJB containers in a cluster using a weighted round-robin scheme.

A distributed server configuration can be created in one of three ways:

- ▶ Create a *deployment manager profile* to define the deployment manager. Then, create one or more *custom node profiles*. The nodes defined by each custom profile can be federated into the cell managed by the deployment manager during profile creation or later manually. The custom nodes can exist on the deployment manager machine or on multiple separate machines. Application servers can then be created using the administrative tools, for example, the administrative console.

The method is useful when you will be creating multiple nodes, multiple application servers on a node, or clusters. The process for creating and federating each node is fairly quick and streamlined. Because servers created using the application server profile are always named “server1”, this method gives you more flexibility in server names.

- ▶ 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 machine or on multiple separate machines.

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

This is useful in a development or test environment. Creating a single profile gives you a simple distributed system on a single server.

4.1.3 Nodes, node groups, and node agents

A *node* is a grouping of application servers for configuration and operational management on one machine. Nodes are generally associated with a physical machine. It is possible to have multiple nodes on a single machine but nodes cannot span machines. In a stand-alone application server configuration, there is only one node. With Network Deployment, you can configure a distributed server environment consisting of multiple nodes that are managed from one central administration server.

In distributed server configurations, each node has a *node agent* that works with the deployment manager to manage administration processes. The node agent is created under the covers when you add (federate) a stand-alone node to a cell.

A *node group* is a grouping of nodes within a cell that have similar capabilities. A node group validates that the node is capable of performing certain functions before allowing those functions. For example, a cluster cannot contain both z/OS nodes and nodes that are not z/OS. 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 automatically created based on the deployment manager platform. This node group 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 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.

4.1.4 Cells

A *cell* is a grouping of nodes into a single administrative domain. In the Base and Express configurations, a cell contains one node and that node contains one server.

In a distributed server configuration, a cell can consist of multiple nodes, which are all administered from a single point (the 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.

It is possible to have a cell made up of nodes on mixed platforms. This is referred to as a heterogeneous cell.

4.1.5 Application server clusters

With Network Deployment, you can use application server clustering to enhance workload distribution. A cluster is a logical collection of application server processes that provides workload balancing and high availability.

Application servers that belong to a cluster are members of that cluster and must all have identical application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any other configuration data.

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

The members of a cluster can be located on a single node (vertical cluster), across multiple nodes (horizontal cluster), or on a combination of the two. A cluster can span machine or LPAR boundaries and can span across operating systems with one exception. A cluster cannot span z/OS and non-z/OS platforms.

When you install, update, or delete an application, the updates are automatically distributed to all members in the cluster. A rollout update option enables you to update and restart the application servers on each node, one node at a time, providing continuous availability of the application.

4.1.6 Web servers

Although Web servers are an independent product, they can be defined to the WebSphere Application Server administration process. The primary purpose for this is to enable the administrator to associate applications with one or more defined Web servers in order to generate the proper routing information for Web server plug-ins. In addition, Web server management capabilities are available in some circumstances.

As with application servers, Web servers are associated with nodes. These nodes can be managed or unmanaged. 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 it to the Web server. You normally have managed Web server nodes behind the firewall with the WebSphere Application Server installations.

Unmanaged nodes, as the name implies, are not managed by WebSphere. You normally find these outside the firewall or in the demilitarized zone. You must manually copy or FTP the Web server plug-in configuration file to the Web server on an unmanaged node. In a z/OS environment, you must use unmanaged nodes if the Web server is a non-z/OS product.

Note: As a special case, if the unmanaged Web server is IBM HTTP Server, you can administer the Web server from the WebSphere administrative console. Then, you can automatically push the plug-in configuration file to the Web server with the deployment manager using HTTP commands to the IBM HTTP Server administration process. This configuration does not require a node agent.

IBM HTTP Server is shipped with all WebSphere Application Server packages.

Web server plug-ins

A Web server can serve requests that do not require any dynamic content, for example, 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.

To forward a request, you use a Web server plug-in that is included with the WebSphere Application Server packages for installation on a Web server. You copy an Extensible Markup Language (XML) configuration file, configured on the WebSphere Application Server, to the Web server plug-in directory. The plug-in uses the configuration file to determine whether a request should be handled by the Web server or an application server. When WebSphere Application Server receives a request for an application server, it forwards the request to the appropriate Web container in the application server. The plug-in can use HTTP or HTTPS to transmit the request.

4.2 Distributed server environments

Network Deployment enables you to build distributed server environments consisting of multiple application servers maintained from a central administrative point. Application servers can be clustered for workload management and high availability. This section provides information about how you can structure distributed server environments.

4.2.1 Single cell configurations

A cell in a distributed server environment is a network 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 can reside on the same machine or MVS™ image as the nodes or on a separate machine.

Figure 4-1 illustrates the cell concept, showing all processes on one system.

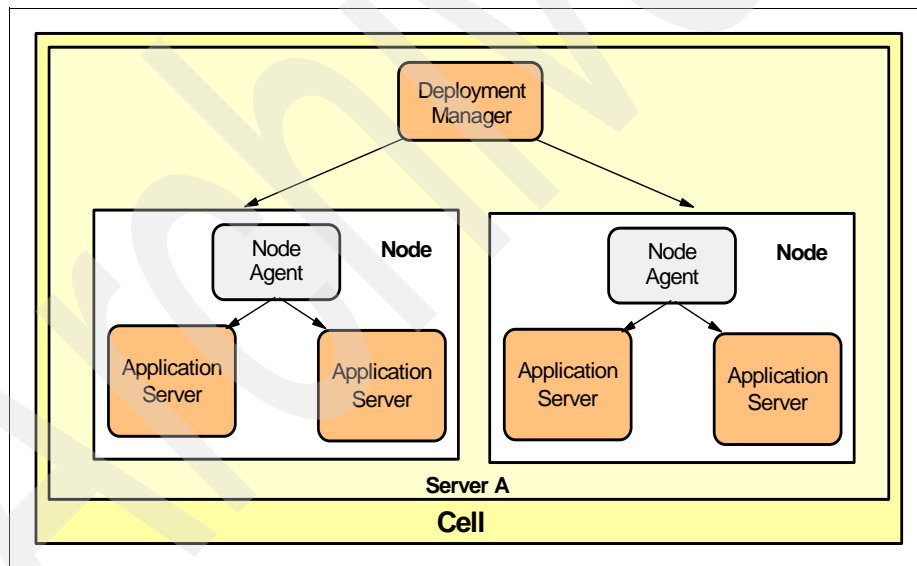


Figure 4-1 Cell topology option: Single system

Alternately, the deployment manager can be installed on one machine (Server A) and each node on a different machine (Server B and Server C), as shown in Figure 4-2. The servers do not have to be the same platform. For example, Server A can be a Microsoft Windows machine while Server B and Server C can be AIX 5L systems. This would require a high-performing system.

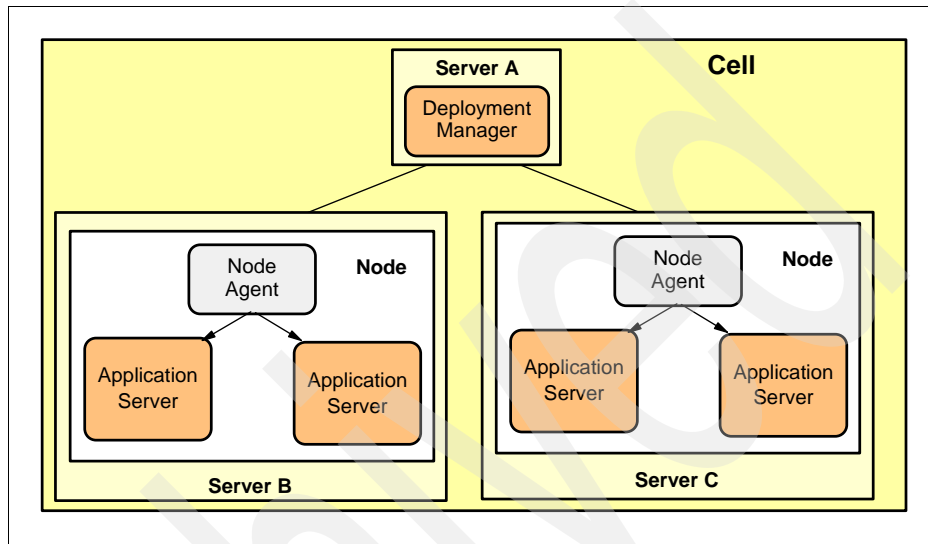


Figure 4-2 One cell multiple systems

By the same logic, you can install other combinations, such as the deployment manager on a single server, with multiple nodes on another. Or, the deployment manager and a node on one server, with additional nodes installed on separate servers.

The point is that a distributed server environment gives you the flexibility to install the WebSphere components on servers and in locations that suit your requirements.

4.2.2 Multiple cells

Several nodes can be created on a machine or MVS image. Those nodes can belong to the same cell or spread across multiple cells. Additionally, these cells can reside entirely within one server or be sprayed among two or more servers, as shown in Figure 4-3.

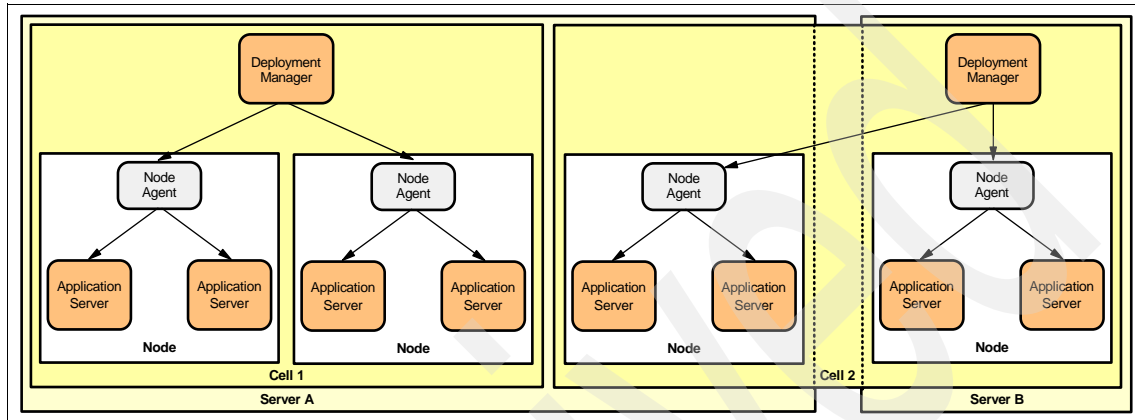


Figure 4-3 Multiple independent cells sharing a physical machine or LPAR

4.2.3 Mixed node versions in a cell

WebSphere Application Server V6.1, V6, and V5 nodes can be part of the same cell. This requires that the deployment manager be at V6.1. You can upgrade a portion of the nodes in a cell, while leaving others at the earlier release level. Therefore, you might be managing servers that are running multiple release levels in the same cell.

Figure 4-4 shows a configuration where mixed node versions reside within a cell.

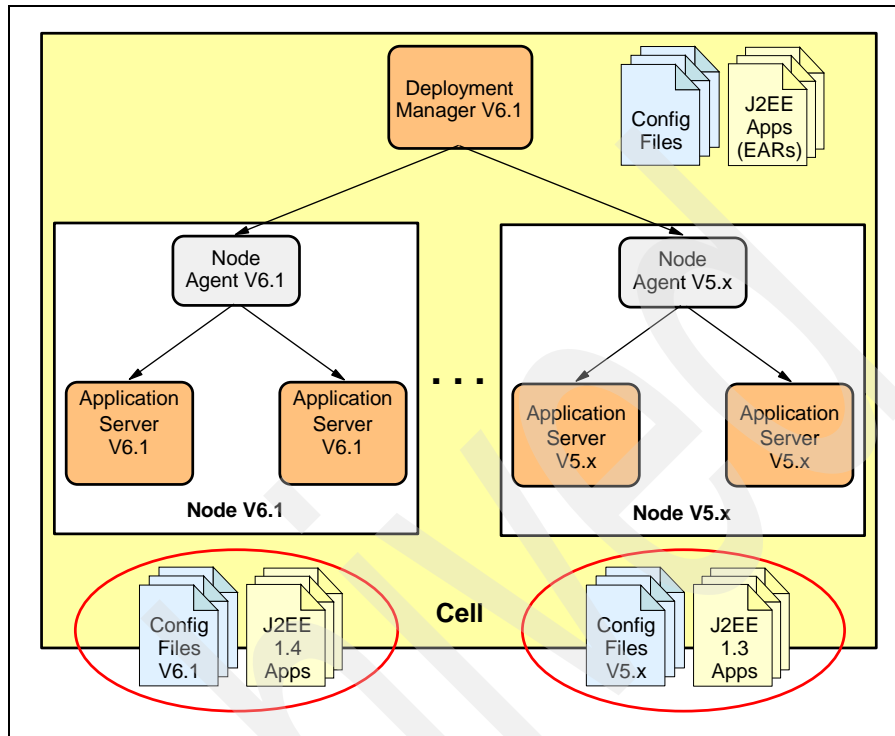


Figure 4-4 Mixed V6.1 and V5 nodes

4.3 Application server clusters

A cluster is a grouping of application servers that run a set of applications managed in such way that they behave as a single application server (parallel processing). A cell can have zero or more clusters and all cluster members that are part of the cluster must belong to the same cell. Network Deployment is required for clustering.

All cluster members can reside on the same machine or MVS image. This topology is known as *vertical scaling*.

Cluster members can also be spread across different machines or MVS images and can span multiple operating system types, with one exception. A cluster cannot span from distributed to z/OS. In this topology, each machine has a node in the cell holding a cluster member. This topology is known as *horizontal scaling*.

The combination of vertical and horizontal scaling is possible.

Figure 4-5 shows a cluster that has four cluster members. Those cluster members belong to two different nodes (but they still serve the same J2EE application). On each node, there is also an application server that is in the same cell but is not part of the cluster (not a cluster member). Therefore, not all the application servers in the node have to be part of the cluster.

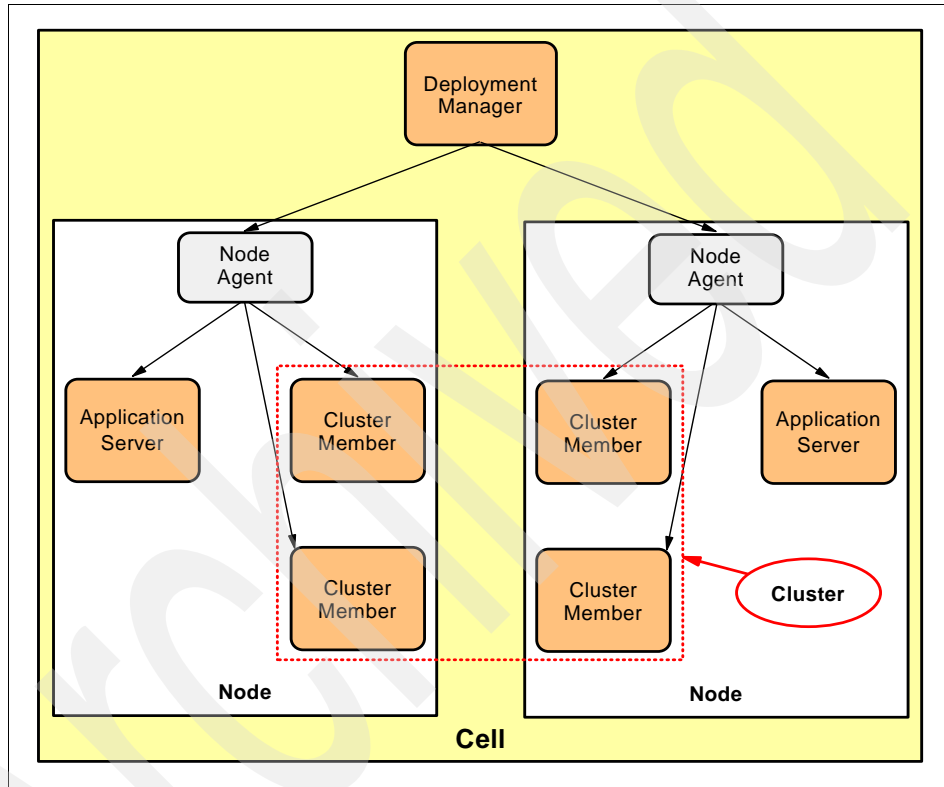


Figure 4-5 Cluster and cluster member

Use of a HTTP traffic-handling device such as IBM HTTP Server is highly recommended. This is a simple and efficient way to front-end the WebSphere HTTP transport. Weighted averages can be used to select the most efficient server in the cluster. The use of a Web server plug-in to manage the workload across the cluster will ensure that the loss of an application server will not disrupt the flow of requests. In the case of horizontal scaling where each node resides on a separate server, the loss of one server will not disrupt the flow of requests.

The loss of the deployment manager would have minimal impact on operations and will primarily affect configuration activities.

Mixed node versions in a cluster

The topology shown in Figure 4-6 contains mixed version nodes within a cluster. The deployment manager has to be at V6.1.

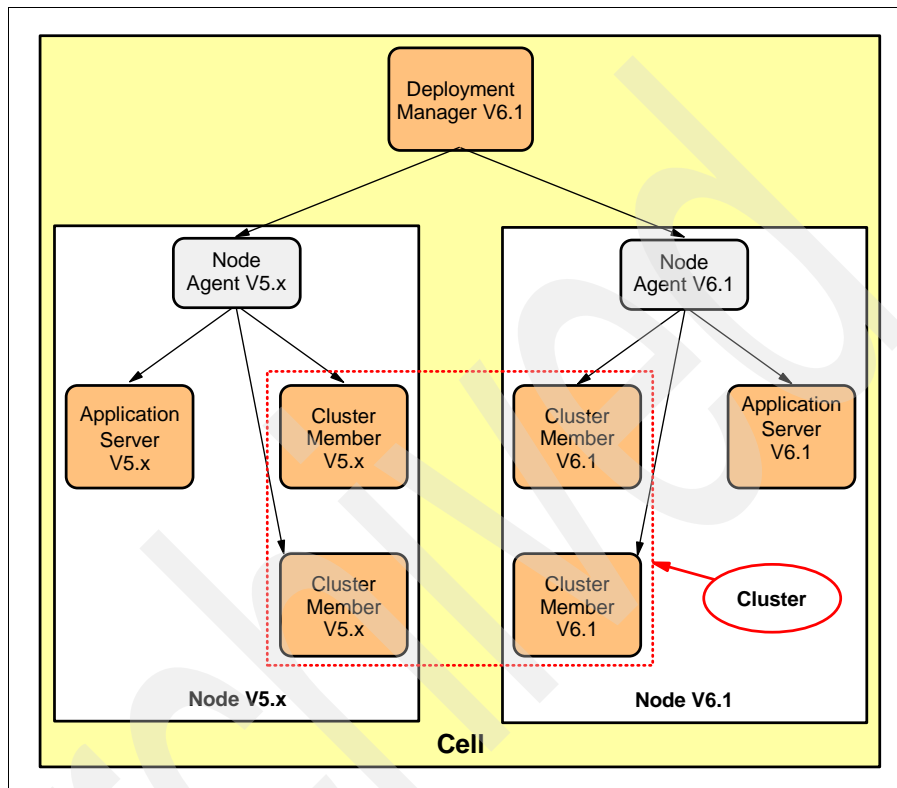


Figure 4-6 Mixed version cluster nodes in a cell

You can upgrade a portion of the nodes in a cell, while leaving others at a previous release level. This is not just a migration scenario, but is supported for stable cells and clusters.

4.4 Runtime processes

In this section, we discuss how WebSphere processes execute in runtime. The executable processes include deployment managers, node agents, and the application server. Note that cells, nodes, and clusters are administrative concepts and not executable components.

4.4.1 Distributed platforms

On distributed platforms, WebSphere Application Server is built using a single process model where the entire server runs on a single Java virtual machine (JVM™) process.

Each process appears as a Java process. For example, when you start a deployment manager on Windows, a `java.exe` process will be visible in the Windows Task Manager. Starting a node agent starts a second `java.exe` process, and each application server started will be seen as a `java.exe` process (Figure 4-7).

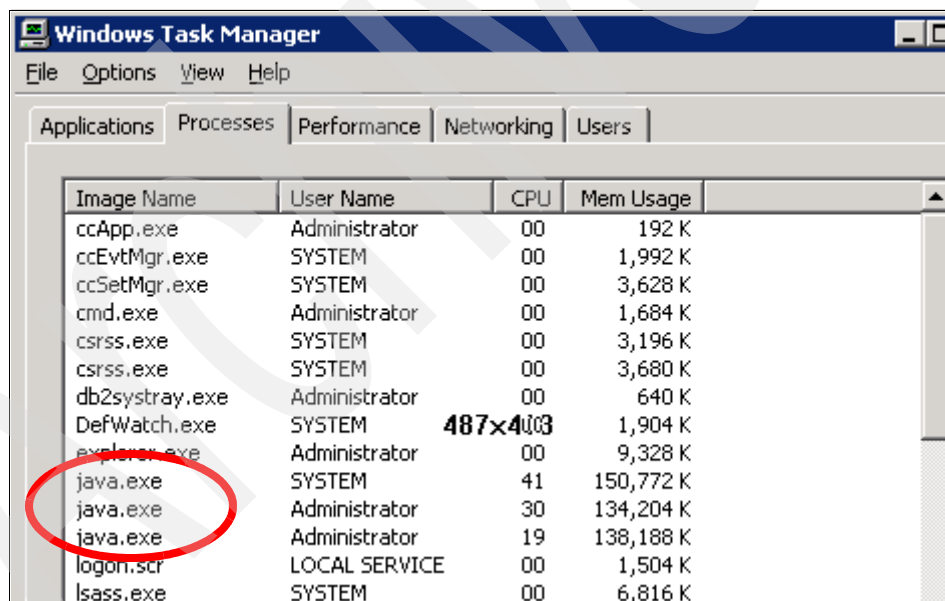


Image Name	User Name	CPU	Mem Usage
ccApp.exe	Administrator	00	192 K
ccEvtMgr.exe	SYSTEM	00	1,992 K
ccSetMgr.exe	SYSTEM	00	3,628 K
cmd.exe	Administrator	00	1,684 K
csrss.exe	SYSTEM	00	3,196 K
csrss.exe	SYSTEM	00	3,680 K
db2sysstray.exe	Administrator	00	640 K
DefWatch.exe	SYSTEM	487x4003	1,904 K
explorer.exe	Administrator	00	9,328 K
java.exe	SYSTEM	41	150,772 K
java.exe	Administrator	30	134,204 K
java.exe	Administrator	19	138,188 K
logon.scr	LOCAL SERVICE	00	1,504 K
lsass.exe	SYSTEM	00	6,816 K

Figure 4-7 WebSphere Application Server processes on a Windows system

To find the processes on an AIX 5L or Linux system, use the following command:

```
ps -ef | grep java
```

4.4.2 WebSphere Application Server for z/OS

WebSphere Application Server for z/OS contains a unique process model that enables the product to manage many z/OS unique services and provides Quality of Service (QoS). On z/OS, an application server is built using a federation of JVMs, each in a different process that together represents a single server instance. A server is composed of address spaces that actually run the code.

Figure 4-8 illustrates how WebSphere processes are structured in a z/OS environment.

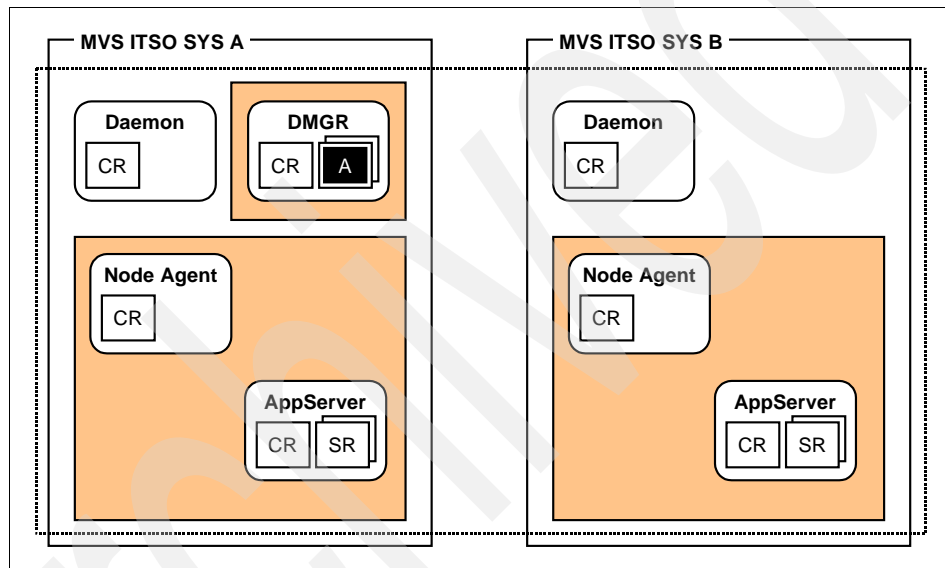


Figure 4-8 One cell, deployment manager, node on same LPAR, node separated

To understand how WebSphere on z/OS is unique from the distributed platforms, we briefly discuss some of the z/OS-specific concepts.

Address space

An address space is the area of successive virtual addresses that z/OS assigns to a user (or separately running program) for executing instructions and storing data. It is equivalent to a process on distributed platforms.

Control region

The control region (CR) is basically the only public interface to this collection of JVMs that, all together, represent a single application server. All requests go through the CR and the CR forwards them to one of the potentially many servant

controllers for processing. In short, a CR is like a router or even an address space that binds the TCP ports used by the server. A CR does have an embedded JVM, which is the only JVM allowed to receive connections from the outside world. Each server has only one CR that is started through a JCL start procedure.

The requests arrive in the CR process, which then works with the z/OS workload manager (WLM) to dispatch the work to the servant regions (Figure 4-9).

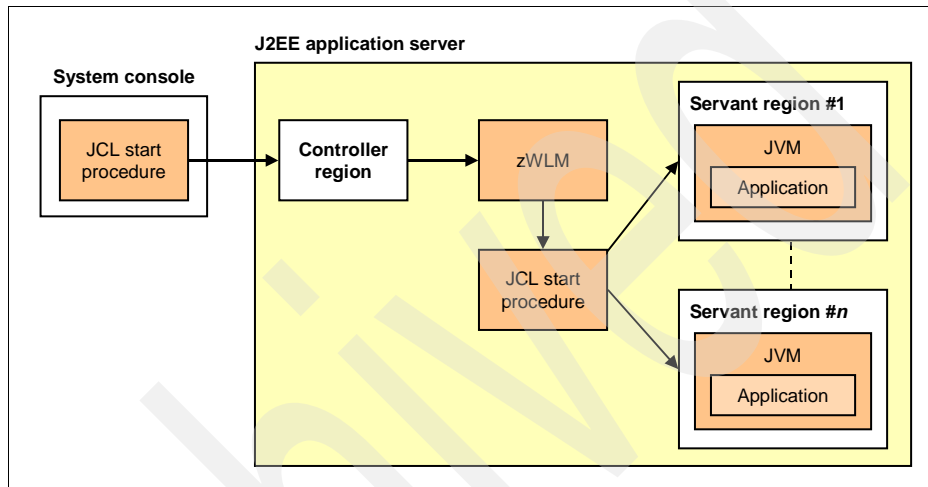


Figure 4-9 Inside the application server

Control region adjunct

The control region adjunct is a specialized servant that interfaces with new service integration buses to provide messaging services.

Servant region

The servant region (SR) is where the requests will be actually processed and is equivalent to the application server on a distributed environment platform that has a J2EE Web container and EJB container. All the SRs are identical and have the same J2EE level. The SR depends on the CR for many services such as communication, security and transaction control.

When multiple SRs are created a copy of each application is found in each SR and the CR will forward the requests to the appropriate SR.

zWLM

The z/OS workload manager manages resources to ensure that performance goals are met. It is a part of z/OS. To differentiate this from the workload

management of WebSphere, we refer to this as zWLM. As the CR receives incoming requests, it works with zWLM to ensure that these requests are classified according to organization-defined rules and dispatched appropriately to servant regions that can handle the load. zWLM can alter factors to ensure that performance goals are met, for example, by updating importance levels of services classes and starting additional servant regions.

Daemon

A daemon server provides the location name service for external clients. There is one daemon per cell per MVS image. If your cell consists of multiple MVS images, a daemon will be created for each MVS image where your cell exists. If there are two cells on the same MVS image, two daemons will be created. Each daemon server consists of a single CR.

Daemon servers are started automatically when the first server for the cell on that MVS image is started. If you kill a daemon, all the servers for that cell on that MVS image come down.

4.5 Using Web servers

In WebSphere Application Server, a Web server can be administratively defined to the cell. This allows the association of applications to one or more Web servers and custom plug-in configuration files to be generated for each Web server. This section discusses the options you have for managing Web servers in a WebSphere Application Server environment.

When you define a Web server to WebSphere, it is associated with a node. The node will either be a *managed* or an *unmanaged* node. When we refer to managed Web servers, we are referring to a Web server defined on a managed node. Similarly, an unmanaged Web server resides on an unmanaged node. In a stand-alone server environment, you can define one unmanaged Web server. In a distributed environment, you define multiple managed or unmanaged Web servers.

4.5.1 Managed Web servers

Defining a managed Web server gives you the ability to start and stop the Web server from the WebSphere Application Server console and automatically push the plug-in configuration file to the Web server. It requires a node agent to be installed on the Web server machine. An exception to this is the case where the Web server is IBM HTTP Server (see 4.5.3, “IBM HTTP Server as an unmanaged Web server (special case)” on page 57). Figure 4-10 illustrates a Web server managed node.

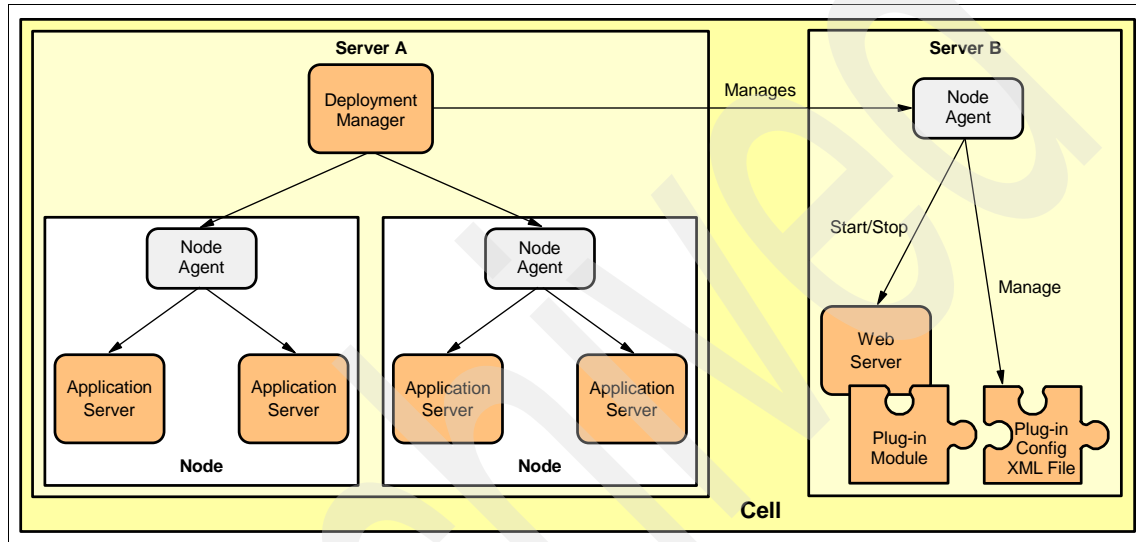


Figure 4-10 Web server managed node

4.5.2 Unmanaged Web servers

Unmanaged Web servers reside on a system without a node agent. This is the only option in a stand-alone server environment and is a common option for Web servers installed outside a firewall. The use of this topology requires that each time the plug-in configuration file is regenerated, it is copied from the machine where WebSphere Application Server is installed to the machine where the Web server is running. Figure 4-11 illustrates a Web server unmanaged node.

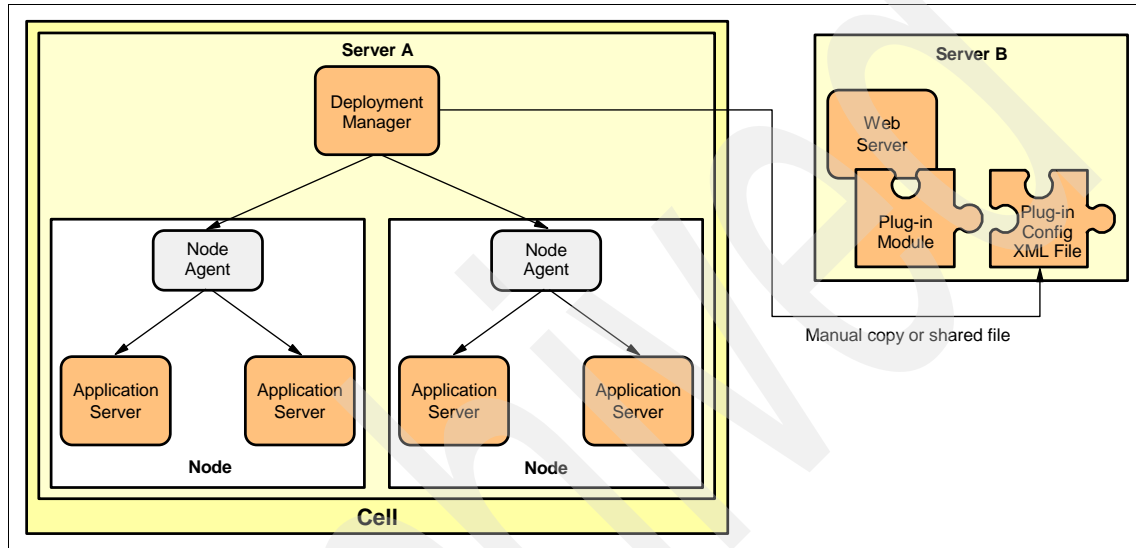


Figure 4-11 Web server unmanaged node

4.5.3 IBM HTTP Server as an unmanaged Web server (special case)

If the Web server is IBM HTTP Server, it can be installed on a remote machine without installing a node agent. You can administer IBM HTTP Server through the deployment manager using the IBM HTTP Server Admin Process for tasks such as starting, stopping, or automatically pushing the plug-in configuration file. Figure 4-12 illustrates an IBM HTTP Server unmanaged node.

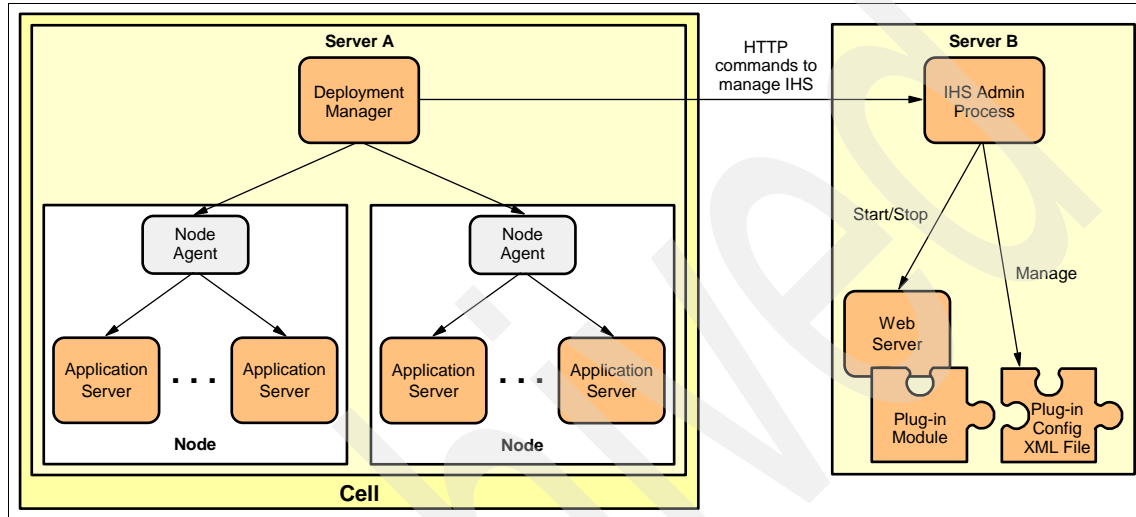


Figure 4-12 IBM HTTP Server unmanaged node

Archived

Topologies

Topology refers to what devices and computers are going to be used to set up a Web application—the physical layout of each one and the relationship between them. When selecting a topology, there are a number of considerations that impact the decision of which one to use.

This chapter describes some common topologies that are used in WebSphere Application Server implementations. It addresses topologies that are relatively simple to those that are more complex by describing the relationship between WebSphere Application Server components and their role in the solution. This chapter contains the following sections:

- ▶ Topology selection criteria
- ▶ Terminology
- ▶ Stand-alone server topology
- ▶ Reverse proxy topology
- ▶ Vertical scaling topology
- ▶ Horizontal scaling topology
- ▶ Horizontal scaling with IP sprayer topology
- ▶ Topology with redundancy of several components
- ▶ Heterogeneous cell

5.1 Topology selection criteria

There are many ways to architect a system to provide the results you need. This section provides a quick overview of the primary considerations in selecting a topology, including:

- ▶ Security
- ▶ Performance and throughput
- ▶ Availability
- ▶ Maintainability

5.1.1 Security

Security is one of the most critical considerations when designing a new system. A secure system requires sophisticated controls in place to protect resources from threats. Security is a vast topic but can be thought of in two basic categories: physical security and logical security. Physical security means protection against physical actions such as the room where the machines are installed. Logical security is connected to a specific IT solution, architecture, and application design.

When selecting a topology, be aware that security usually requires a physical separation of the Web server from the application server processes, typically across one or more firewalls. A common configuration is the use of two firewalls to create a DMZ between them. Information in the DMZ has some protection that is based on protocol and port filtering. A Web server intercepts the requests and forwards them to the corresponding application servers through the next firewall. The sensitive portions of the business logic and data resides behind the second firewall, which filters based on protocols, ports, IP addresses, and domains.

Before selecting a topology, review the information in Chapter 12, “Planning for security” on page 253.

For more information regarding security fundamentals, refer to *WebSphere Security Fundamentals*, REDP-3944, available at:

<http://www.redbooks.ibm.com/abstracts/redp3944.html>

5.1.2 Performance and throughput

Performance involves minimizing the response time for a given transaction. Proper hardware selection, sizing, and application design and test are critical in achieving a well-performing system.

Throughput is related to performance but in the sense of the volume being served. This volume can be measured in the number of bytes that are transferred or the number of transactions that are completed in a given period of time.

WebSphere Application Server Network Deployment enables you to cluster application servers so that you have multiple servers running the same application available to handle incoming requests. Clustering provides improvements for both performance (response time) and throughput (how many).

Scaling

Scaling represents the ability to create more application server processes to serve the user requests. Multiple machines can be configured to add processing power, improve security, maximize availability, and balance workloads. There are two options for scaling: vertical or horizontal. Which you use depends on where and how the scaling is taking place:

- ▶ Vertical scaling

Involves creating additional application server processes on a single physical machine or MVS image, providing application server failover as well as load balancing across multiple application servers. This topology does not provide a very efficient fault tolerance because a failure of the operational system or the hardware on the physical machine itself might cause problems to all servers in the cluster.

- ▶ Horizontal scaling

Involves creating application servers on multiple machines to take advantage of the additional processing capacity available on each machine. This also provides hardware failover support.

The components that provide functions for configuring scalability include:

- ▶ WebSphere Application Server cluster support

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

The workload management capabilities of WebSphere can be used to distribute requests among Web containers and EJB containers in clustered application servers. This enables 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 a one specific Web server, the IP sprayer is actually intercepting all those requests and distributing them among all the available Web servers in the group. IP sprayers (such as the Load Balancer Edge component or Cisco Local Director) can provide scalability, load balancing, and failover for Web servers.

5.1.3 Availability

To avoid a single point of failure and to maximize system availability, the topology should have some degree of process redundancy. High-availability topologies typically involve horizontal scaling across multiple machines. Vertical scaling can improve availability by creating multiple processes, but the machine itself becomes a point of failure.

In addition, an IP sprayer can direct Web client requests to the next available Web server, bypassing any server that is offline. The IP sprayer server can be configured with a cascade backup (standby) that comes online in the event that the primary server fails. This configuration eliminates single point of failure on the IP sprayer.

Improved availability is one of the key benefits of scaling to multiple machines. Applications that are hosted on multiple machines generally have less down time and are able to service client requests more consistently.

This section discusses the commonly used techniques that you can combine to take advantage of the best features of each topology and create a highly available system.

Hardware and process redundancy

Eliminate the single points of failure in a system by including hardware and process redundancy using the following techniques:

- ▶ Use horizontal scaling to distribute application servers (and applications) across multiple physical machines or MVS images. If a hardware or process failure occurs, clustered application servers are available to 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 locations to prevent site-related problems.

Process isolation

Provide process isolation so that a failing servers does not impact the remaining healthy servers. The following configurations provide some degree of process isolation:

- ▶ Deploy the Web server on a different machine from the application servers. This configuration ensures that problems with the application servers do not affect the Web server and vice versa.
- ▶ Use horizontal scaling, placing application servers on different machines.

Load balancing

Use load balancing techniques to make sure that individual servers are not overwhelmed with client requests while other servers are idle. These techniques include:

- ▶ Use an IP sprayer to distribute requests across the Web servers in the configuration.
- ▶ Direct requests from high-traffic URLs to more powerful servers.

The Edge components included with the Network Deployment package provides these features.

Failover support

The site must be able to continue processing client requests, even if one or more servers are offline. Some ways to provide failover support include:

- ▶ Use horizontal scaling with workload management to take advantage of its failover support.
- ▶ Use an IP sprayer to distribute requests across the Web servers in the configuration.
- ▶ Use HTTP transport to distribute client requests among application servers.

Hardware-based high availability

The WebSphere Application Server high availability framework provides integration into an environment that might be using other high availability frameworks, such as HACMP™, to manage resources that do not use WebSphere.

5.1.4 Maintainability

The topology affects the ease with which system hardware and software can be updated. For instance, using horizontal scaling can make a system easier to maintain because individual machines can be taken offline without interrupting other machines running the application, thus providing business continuity. When deciding how much vertical and horizontal scaling to use in a topology, consider needs for hardware upgrades (for example, adding or upgrading processors and memory).

5.1.5 Topology selection summary

The following tables list the requirements for topology selection and the possible solutions.

Table 5-1 Topology selection based on availability requirements

Requirement = availability	Solution/topology
Web server	<ul style="list-style-type: none">▶ Load Balancer (with backup) or a high availability solution, based on additional requirements
Application server	<ul style="list-style-type: none">▶ Horizontal scaling (process and hardware redundancy)▶ Vertical scaling (process redundancy)▶ A combination of both
Database server	<ul style="list-style-type: none">▶ HA software solution

Table 5-2 Topology selection based on performance requirements

Requirement = performance/throughput	Solution/topology
Web server	<ul style="list-style-type: none">▶ Multiple Web servers in conjunction with Load Balancer▶ Caching Proxy▶ Dynamic caching with Adaptive Fast Path Architecture (AFPA) or ESI external caching
Application server	<ul style="list-style-type: none">▶ Clustering (in most cases horizontal)▶ Dynamic caching▶ Serving static content from Web server to offload application server
Database server	<ul style="list-style-type: none">▶ Separate database server

Table 5-3 Topology selection based on security requirements

Requirement = security	Solution/topology
Web server	▶ Separate the Web server into a DMZ, either on LPAR or separate system.
Application server	▶ Separate components into a DMZ, implement security (for example, LDAP).

Table 5-4 Topology selection based on maintainability requirements

Requirement = maintainability	Solution/topology
Web server	▶ Single machine environment is the easiest option to maintain. It can be combined with horizontal scaling.
Application server	
Database server	

5.2 Terminology

Before examining the topologies, take a minute to review the following information. These are elements that you will see in the diagrams representing each topology.

Web server and plug-in

Most topologies feature a stand-alone Web server that resides in a DMZ between two firewalls. Web clients send requests to the Web server, which serves static content such as HTTP pages. Requests for content that requires processing by servlets, JSPs, enterprise beans, and back-end applications are forwarded (or re-directed) to the application server.

WebSphere Application Server V6.1 ships with IBM HTTP Server and supports several other Web server products (see 1.4.2, “Web servers” on page 10). It also ships a set of Web server plug-ins that perform the redirection to the application server.

Application server

This refers to the application server in WebSphere that runs user applications. The application server collaborates with the Web server to return a dynamic, customized response to a client request. The application server is capable of running both presentation and business logic but generally does not serve HTTP requests. Each application server node has its own HTTP/HTTPS listener, and it

is possible to point a browser directly to the application server's HTTP/HTTPS port; however, we do not recommend this.

Domain and protocol firewalls

A firewall is a hardware and software system that manages the flow of information between the Internet and an organization's private network. Firewalls can prevent unauthorized Internet users from accessing private networks that are connected to the Internet, especially intranets, and can block some virus attacks, as long as those viruses are coming from the Internet. Firewalls also can prevent denial of service attacks.

A firewall can separate two or more parts of a local network to control data exchange between departments. Components of firewalls include filters or screens, each of which controls transmission of certain classes of traffic. Firewalls provide the first line of defense for protecting private information, but 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 implementations of firewall services include:

- ▶ Screening routers (the protocol firewall)
 - Prevents unauthorized access from the Internet to the DMZ. The role of this node is to provide the Internet traffic access only on certain ports and to block other IP ports.
- ▶ Application gateways (the domain firewall)
 - Prevents unauthorized access from the DMZ to an internal network. The role of firewall allows the network traffic originating 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.

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 Web application system. This node can supply information for various security services (authentication and authorization) and can also perform the actual security processing, for example, to verify certificates. The authentication in most current designs validates the access to the Web application server part of the Web server, but this node also authenticates for access to the database server.

An example of a product that provides directory services is IBM Tivoli Directory Server, included in the Network Deployment package. For a list of directory servers supported by WebSphere Application Server, see 1.4.4, “Directory servers” on page 11.

Existing applications and data

Existing applications and data represent the core business logic and data for the organization. The application server accesses this data to build responses for the user. For example, this can represent an enterprise information system (EIS), such as CICS®, a Web service, or a business database.

Load balancer

A load balancer, also referred to as an IP sprayer, provides horizontal scalability by dispatching HTTP requests among several identically configured Web servers. In these topologies, the load balancer is implemented using the Load Balancer Edge component provided with the Network Deployment package.

Note: On the z/OS platform, the function that provides intelligent load balancing is the *Sysplex Distributor*.

5.3 Stand-alone server topology

A stand-alone server topology refers to the installation of WebSphere Application Server on one single (physical) machine or logical partition (LPAR) with one application server. This topology does not provide failover or workload management capabilities. Although this topology is possible with all packages, Base and Express installations only support stand-alone servers. The focus of this topology is to illustrate using a Web server and Web server plug-in in the DMZ to direct requests to the application server.

Although you can direct HTTP requests directly to the application server, you would typically have a Web server as a front-end to receive requests. The Web server is located in a DMZ to provide security, performance, throughput, availability, and maintainability, while the application server containing business logic is located securely in the internal network. Figure 5-1 illustrates this topology.

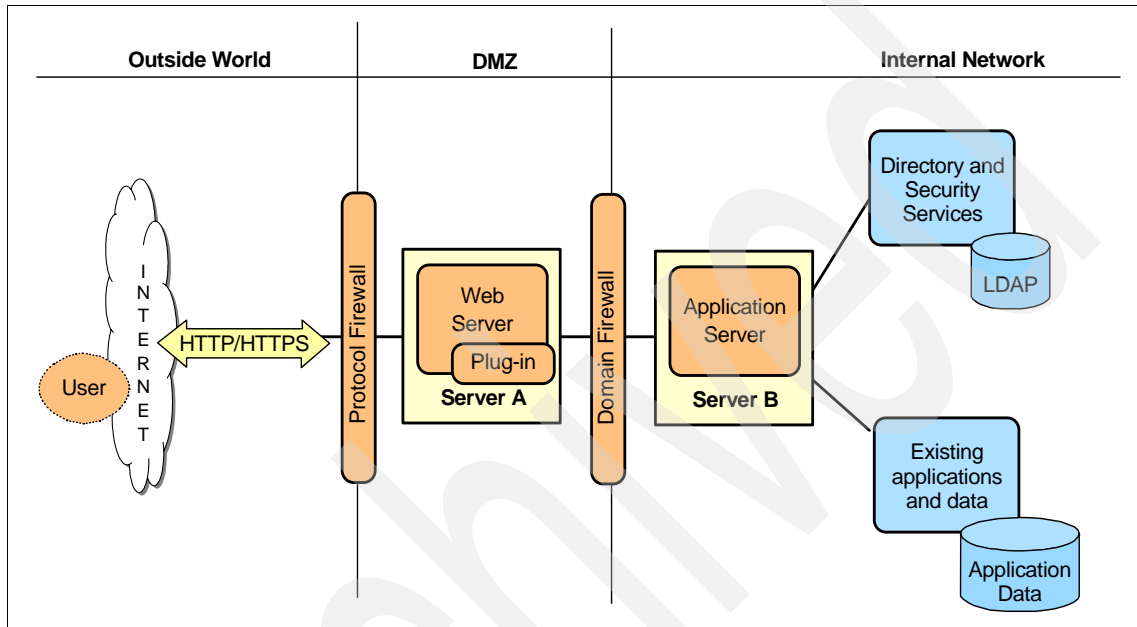


Figure 5-1 Web server separation

WebSphere Application Server provides a set of Web server plug-ins for supported Web servers. The plug-in uses a configuration file that contains settings describing how to pass requests to the application server. The configuration file is generated on the application server and must be regenerated and propagated to the Web server machine each time a change on the application server would affect application request routing (for example, a new application is installed). This move must be done manually.

Note: In a stand-alone topology, only unmanaged Web servers are possible. This means the plug-in must be manually pushed out to the Web server system. The exception to this 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.

Advantages

Some reasons to separate the Web server from the application server are:

- ▶ Performance
 - Size and configure servers appropriately to each task.

By installing components (Web server and application server) on separate machines or MVS images, each machine can be sized and configured to optimize the performance of each component.
 - Remove resource contention.

By physically separating the Web server from the application server, a high load of static requests will not affect the resources (processor, memory, and disk) available to WebSphere, and therefore does not affect its ability to service dynamic requests. The same applies when the Web server serves dynamic content using other technologies, such as CGI.
- ▶ Maintainability

Web server separation provides component independence. Server components can be reconfigured, or even replaced, without affecting the installation of other components on separate machines.
- ▶ Security

Isolating the Web server in a DMZ protects the business applications and data in the internal network by restricting access from the public Web site to the servers and databases where this valuable information is stored. Desirable topologies should not have servers that directly access databases in the DMZ.

Disadvantages

Consider the following disadvantages when you separate the Web server from the application server:

- ▶ Maintainability

The plug-in configuration file is generated on the WebSphere Application Server machine and must be moved to the Web server machine each time a configuration change occurs that affects requests for applications.
- ▶ Performance

Depending on the network capacity and remoteness 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 this, you must ensure that you have an adequate network bandwidth between the Web server and the application server.

- ▶ Security considerations

This configuration uses encrypted transport. The plug-in allows encryption of the link between the Web server and the application server using SSL. This reduces the risk that attackers are able to obtain secure information by “sniffing” packets sent between the Web server and application server. A performance penalty usually accompanies such encryption. We recommend configuring this connection so that the plug-in and Web container must mutually authenticate each other using public-key infrastructure. This prevents unauthorized access to the Web container.

Software requirements

The following list indicates the minimum software configuration that you need for the topology shown in Figure 5-1 on page 68:

- ▶ Server A

- A supported Web server (see 1.4.2, “Web servers” on page 10).
- A Web server plug-in.

- ▶ Server B

WebSphere Application Server (Express, Base, or Network Deployment):

- Profile: Application server profile.
- Administrative configuration tasks: Create a 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 unmanaged.

5.4 Reverse proxy topology

Reverse proxy servers, such as the one in Edge components, are typically used in DMZ configurations to provide additional security between the public Internet and the Web servers (and application servers) servicing requests. The topology shown in Figure 5-2 illustrates the use of a reverse proxy server.

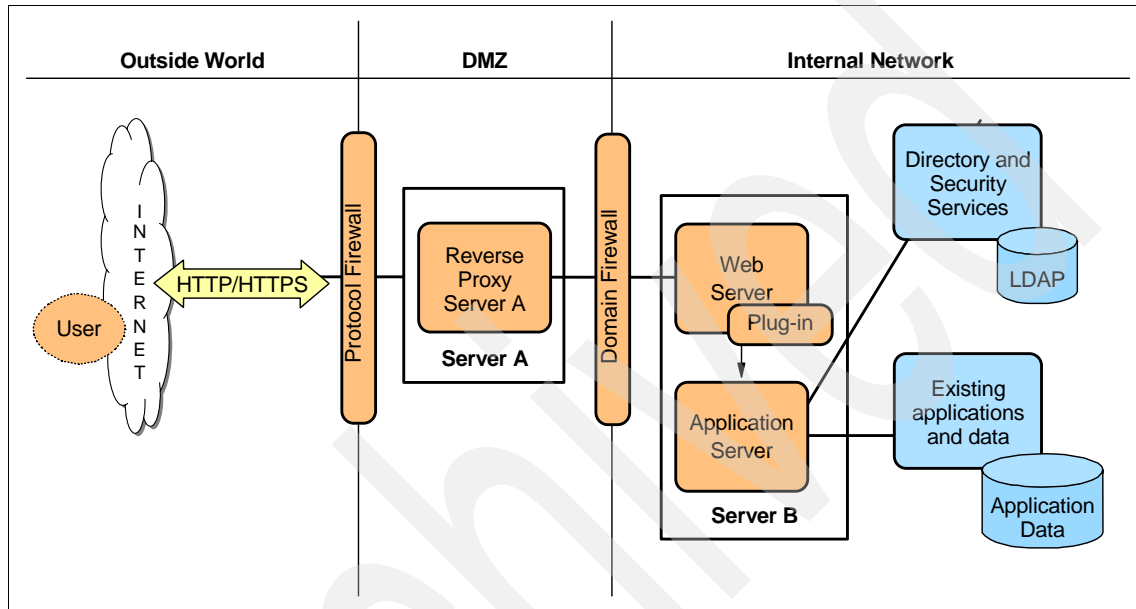


Figure 5-2 Topology using a reverse proxy

In this example, a reverse proxy in the DMZ listens on the HTTP port (typically port 80) for requests. It then 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 server.

Reverse proxy configurations support high-performance DMZ solutions that require as few open ports in the firewall as possible. For the reverse proxy in the DMZ to access the Web server behind the domain firewall, it requires as few as one port open in the firewall (two ports if using SSL).

Note: In WebSphere Application Server V6.0.2, you had to enhance the deployment manager profile to manage the proxy server. For WebSphere Application Server V6.1 and later versions, the proxy server is managed from the administrative console without initial enhancement.

Advantages

Advantages of using a reverse proxy server in a DMZ configuration include:

- ▶ This is a well-known and tested configuration. It is, therefore, easy to implement.
- ▶ It is a reliable and fast-performing solution.
- ▶ It eliminates protocol switching by using the HTTP protocol for all forwarded requests.
- ▶ It has no effect on the configuration and maintenance of a WebSphere application.

Disadvantages

Disadvantages of using a reverse proxy server in a DMZ configuration include:

- ▶ It requires more hardware and software than similar topologies that do not include a reverse proxy server, making it more complicated to configure and maintain.
- ▶ It cannot be used in environments where security policies prohibit the same port or protocol being used for inbound and outbound traffic across a firewall.

Software requirements

The following list indicates the minimum software configuration that you need for the topology shown in Figure 5-2 on page 71:

- ▶ Server A
 - WebSphere Edge components
- ▶ Server B
 - A supported Web server
 - A Web server plug-in
 - WebSphere Application Server (Express, Base, or Network Deployment)
 - Profile: Application server profile.
 - Administrative configuration tasks: Create a managed Web server definition.

5.5 Vertical scaling topology

Vertical scaling (depicted in Figure 5-3) refers to configuring multiple application servers on a single machine and creating a cluster of associated application servers all hosting the same J2EE applications.

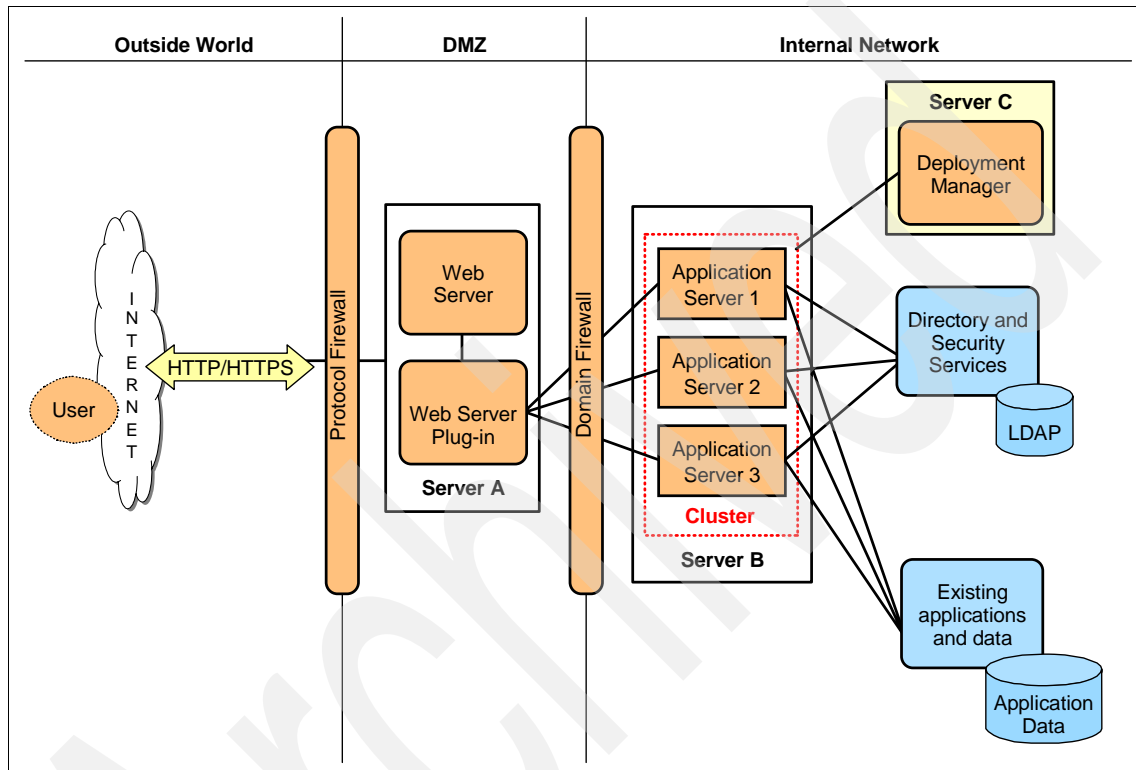


Figure 5-3 Vertical scaling

This vertical scaling example includes a cluster and three cluster members. The Web server plug-in routes the requests according to the application server's availability. Some basic load balancing is performed at the Web server plug-in level based on a weighted round-robin algorithm.

Vertical scaling can be combined with other topologies to boost performance, throughput, and availability.

Advantages

Vertical scaling has the following advantages:

- ▶ Efficient use of machine processing power
With vertical scaling, each application server runs in its own JVM, each using a portion of the machine's processor and memory. The number of application servers can be increased or decreased to balance the resource utilization on the machine.
- ▶ Load balancing
Vertical scaling topologies can make use of WebSphere workload management.
- ▶ Process failover
Vertical scaling can provide failover support among application servers of a cluster. If one application server process goes offline, the other one continues processing client requests.

Disadvantages

Single machine vertical scaling topologies have the drawback of introducing the host machine as a single point of failure in the system.

Software requirements

The following list indicates the minimum software configuration that you need for the topology shown in Figure 5-3 on page 73:

- ▶ Server A
 - A supported Web server
 - A Web server plug-in
- ▶ Server B
 - WebSphere Application Server Network Deployment
 - Custom profile (federated to the cell)
- ▶ Server C
 - WebSphere Application Server Network Deployment
 - Deployment manager profile
 - Administration:
 - Create a cluster with three application servers on the Server B node.
 - Define a Web server.

5.6 Horizontal scaling topology

Horizontal scaling exists when cluster members are located across multiple machines. The topology shown in Figure 5-4 lets a single application span multiple machines, yet still presents as a single logical image. In this example, the cluster spans Server B and Server C, each with one application server. The deployment manager is installed on a fourth server, Server D.

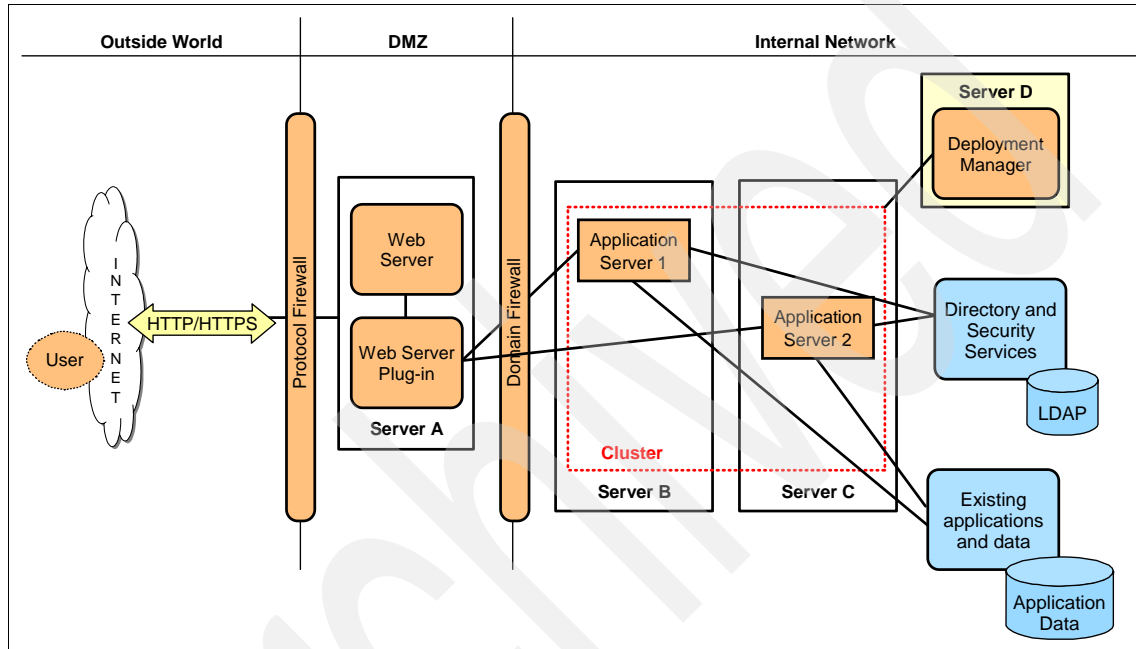


Figure 5-4 Horizontal scaling with cluster

The Web server plug-in distributes requests to the cluster members on each server performing load balancing and offering an initial failover. If any component (hardware or software) on Server B fails, the application server on Server C can serve requests and vice versa.

Figure 5-5 shows a horizontal cluster on a z/OS platform.

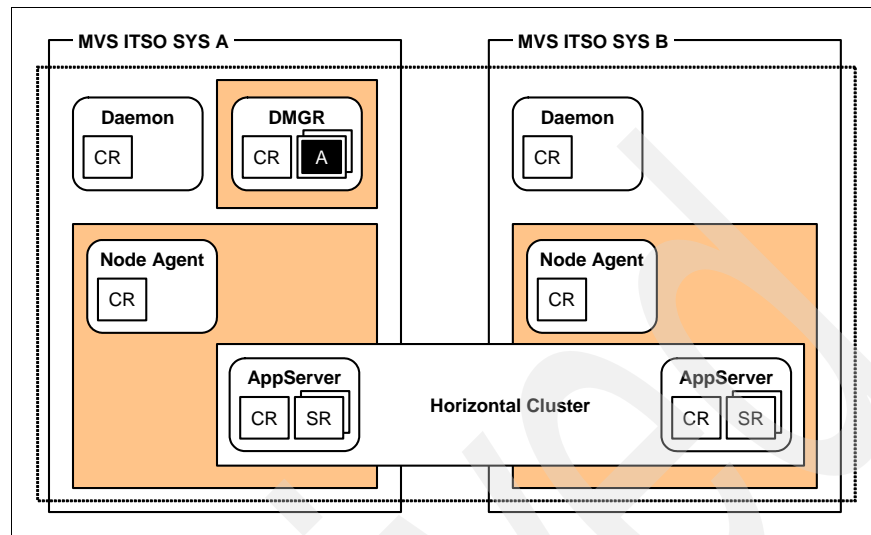


Figure 5-5 Horizontal scaling with cluster on z/OS

The Load Balancer Network Dispatcher can be configured to create a cluster of Web servers and add it to a cluster of application servers. See 5.7, “Horizontal scaling with IP sprayer topology” on page 77 for more information.

Advantages

Horizontal scaling using clusters has the following advantages:

- ▶ Improved throughput
The use of clusters enables the handling of more client requests simultaneously.
- ▶ Improved performance
Hosting cluster members on multiple machines enables each cluster member to make use of the machine's processing resources, avoiding bottlenecks and improving response time.
- ▶ Hardware failover
Hosting cluster members on multiple machines isolates hardware failures and provides failover support. Client requests can be redirected to cluster members on other machines if a machine goes offline.

- ▶ Application software failover

Hosting cluster members on multiple nodes isolates application software failures and provides failover support if an application server stops running. Client requests can be redirected to cluster members on other nodes.

Disadvantages

Horizontal scaling using clusters has the following disadvantages:

- ▶ Higher hardware costs.
- ▶ More complex maintenance.
- ▶ Application servers must be maintained on multiple machines.

Software requirements

The following list indicates the minimum software configuration that you need for the topology shown in Figure 5-4 on page 75:

- ▶ Server A
 - A supported Web server
 - A Web server plug-in
- ▶ Server B
 - WebSphere Application Server Network Deployment
 - Custom profile (federated to the cell)
- ▶ Server C
 - WebSphere Application Server Network Deployment
 - Custom profile (federated to the cell)
- ▶ Server D
 - WebSphere Application Server Network Deployment
 - Deployment manager profile
 - Administration:
 - Create a cluster with two application servers, one on the Server B node and one on the Server C node.
 - Define a Web server.

5.7 Horizontal scaling with IP sprayer topology

Load balancing products can be used to distribute HTTP requests among Web servers running on multiple physical machines. The Load Balancer Network Dispatcher is an IP sprayer that performs intelligent load balancing among Web servers based on server availability and workload.

Figure 5-6 illustrates a horizontal scaling configuration that uses an IP sprayer to redistribute requests between Web servers on multiple machines.

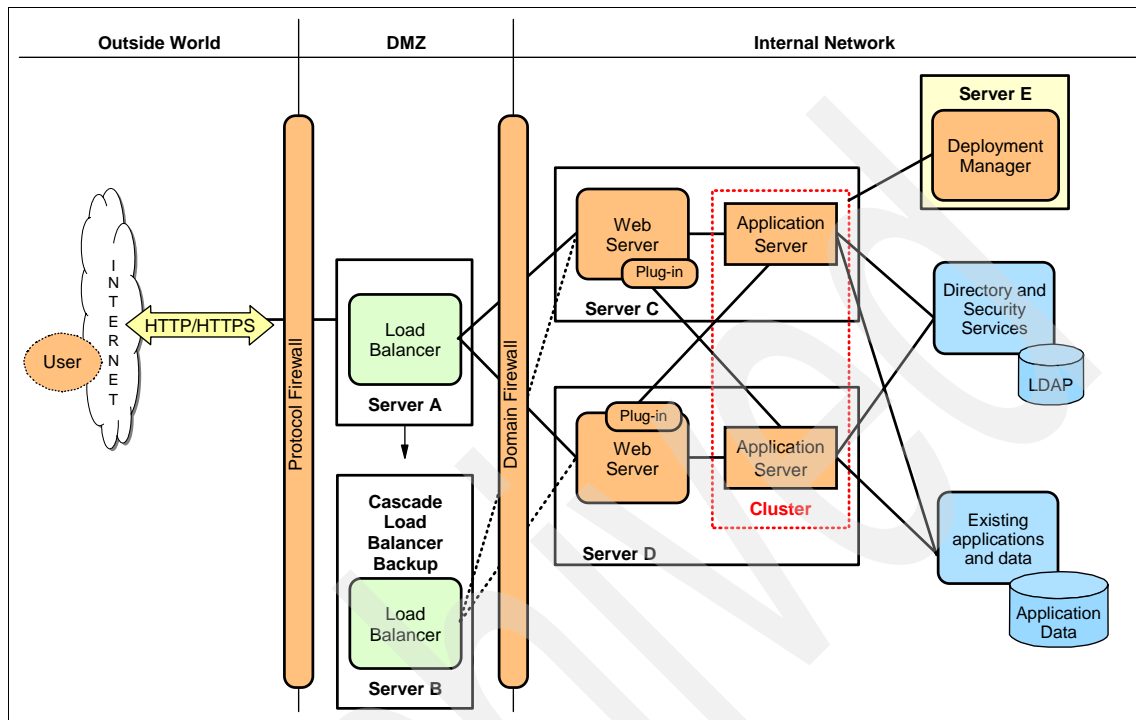


Figure 5-6 Simple IP sprayer horizontally scaled topology

The Load Balancer sprays requests to the Web servers. The Load Balancer is configured in cascade. The primary Load Balancer communicates to its backup through a heartbeat to perform failover, if needed, eliminating a single point of failure.

Both Web servers perform load balancing and failover between the application servers in the cluster through the Web server plug-in. If any component on Server C or Server D fails, the other can continue to receive requests.

Advantages

Using an IP sprayer to distribute HTTP requests has the following advantages:

- ▶ Improved server performance by distributing incoming TCP/IP requests (in this case, HTTP requests) among a group of Web servers.
- ▶ The use 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 single point of failure. Used in combination with WebSphere workload management, it eliminates the application servers as a single point of failure.
- ▶ Improved throughput and performance by maximizing processor and memory use.

Disadvantages

Using an IP sprayer to distribute HTTP requests means that hardware and software are required for the IP sprayer servers.

Software requirements

The following list indicates the minimum software configuration that you need for the topology shown in Figure 5-6 on page 78:

- ▶ Server A and Server B
 - WebSphere Edge components
- ▶ Server C and Server D
 - A supported Web server
 - A Web server plug-in
 - WebSphere Application Server Network Deployment
 - Custom profile (federated to the cell)
- ▶ Server E
 - WebSphere Application Server Network Deployment
 - Deployment manager profile
 - Administration:
 - Create a cluster with two application servers, one on the Server C node and one on the Server D node.
 - Define the Web servers.

5.8 Topology with redundancy of several components

Having as much redundancy of components as possible eliminates or minimizes the single point of failure. Most components have some kind of redundancy, such as a Load Balancer backup server in cascade with the primary Load Balancer server, clustered Web servers, clustered application servers, and so forth.

Figure 5-7 illustrates a topology with redundancy of several components.

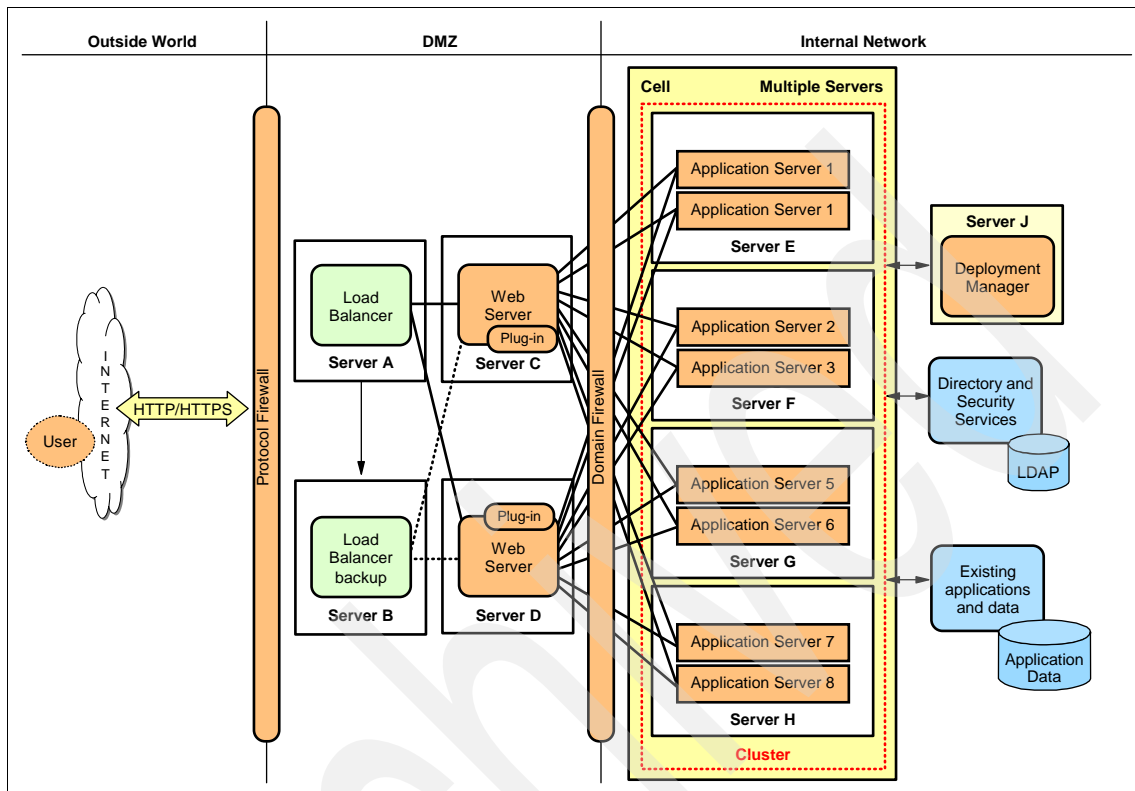


Figure 5-7 Topology with redundancy of several components

The redundant components in this example include:

- ▶ Two Load Balancers

The one on Server A is the primary Load Balancer. It is synchronized, through a heart beat, with a backup Load Balancer in cascade that is in standby on another machine, Server B.

- ▶ Two Web servers

Both Web servers receive requests from the Load Balancer and share the requests that come from the Internet. Each one is installed on a different machine.

- ▶ An application server cluster

The cluster is spread across four server machines and implements vertical and horizontal scaling. The cluster consists of eight cluster members, two on each server.

The application servers on Server E are independent installations. The application servers on Server F are profiles of a single installation.

- ▶ Two database servers

The database servers use a high availability software product. Therefore, one copy of the database is being used and the other one is a replica that will replace the first one if it fails.

- ▶ Two LDAP servers

The LDAP servers use a high availability software product. Therefore, one copy of the database is being used and the other one is a replica that will replace the first one if it fails.

Advantages

This topology is designed to maximize performance, throughput, and availability. It incorporates the benefits of the other topologies that have been discussed already in this chapter. The advantages include:

- ▶ Single point of failure is eliminated from the Load Balancer node, Web server, application server, database server, and LDAP server due to the redundancy of those components.
- ▶ It provides both hardware and software failure isolation. Hardware and software upgrades can be easily handled during off-peak hours.
- ▶ Horizontal scaling is done by using both the IP sprayer (for the Web server nodes) and the application server cluster to maximize availability.
- ▶ Application performance is improved using the following techniques:
 - Hosting application servers on multiple physical machines, MVS images, or both to boost the available processing power.
 - Using clusters to scale application servers vertically, which makes more efficient use of the resources of each machine.
- ▶ Applications with this topology can make use of workload management techniques. In this example, workload management is performed through:
 - Load Balancer Network Dispatcher to distribute client HTTP requests to each Web server.
 - WebSphere Application Server Network Deployment workload management feature to distribute work among clustered application servers.

Disadvantages

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

Software requirements

The following list indicates the minimum software configuration that you need for the topology shown in Figure 5-6 on page 78:

- ▶ Server A and Server B
 - WebSphere Edge components
- ▶ Server C and Server D
 - A supported Web server
 - A Web server plug-in
- ▶ Servers E, F, G, and H
 - WebSphere Application Server Network Deployment
 - Custom profile (federated to the cell)
- ▶ Server J
 - WebSphere Application Server Network Deployment
 - Deployment manager profile
 - Administration:
 - Create a cluster with two application servers on each node.
 - Define the Web servers.

Heterogeneous cell

Cells can span z/OS sysplex environments and other operating systems as well. For example, z/OS nodes, Linux nodes, UNIX nodes, and Microsoft Windows nodes can exist in the same application server cell. This kind of configuration is referred to as a heterogeneous cell. With WebSphere V6.1, there are many different topologies that are possible to compose a heterogeneous cell, as shown in Figure 5-8.

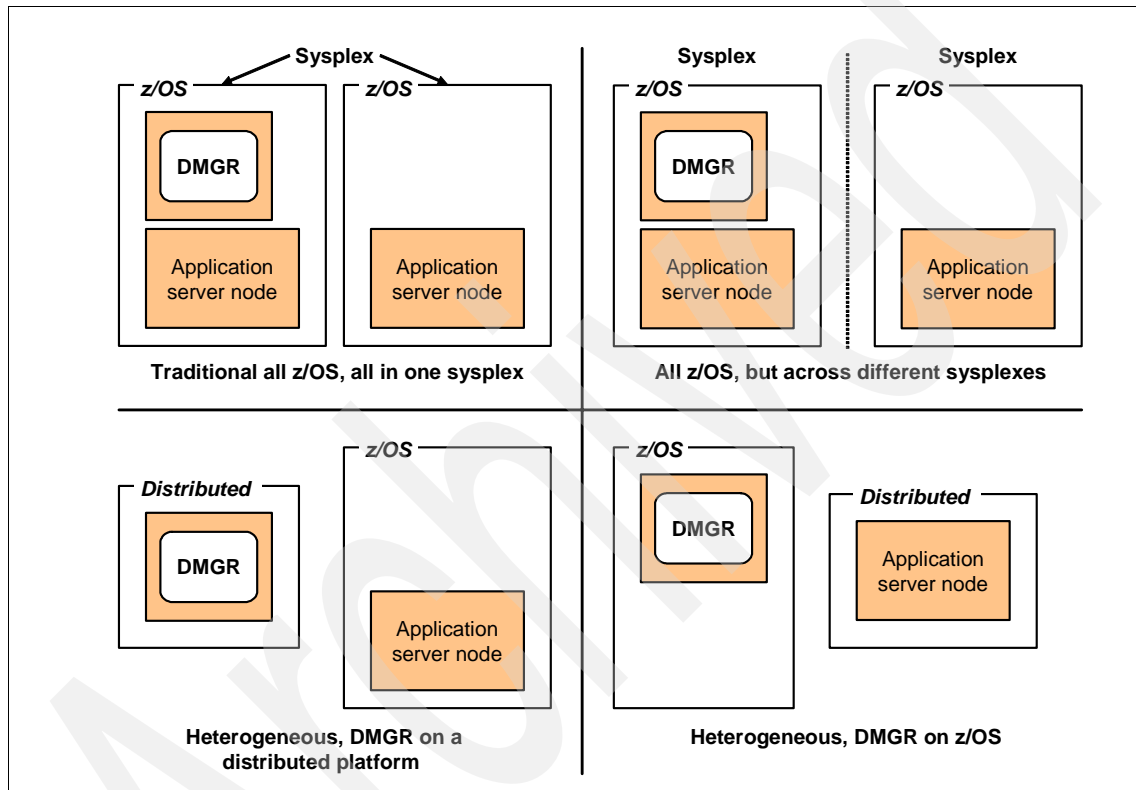


Figure 5-8 Different possibilities with a heterogeneous cell

WebSphere Application Server Version 6.1 products can coexist with the following supported versions:

- ▶ IBM WebSphere Application Server Version 5.0.x
- ▶ IBM WebSphere Application Server Network Deployment Version 5.0.x
- ▶ IBM WebSphere Application Server Version 5.1.x
- ▶ IBM WebSphere Application Server Network Deployment Version 5.1.x
- ▶ IBM WebSphere Application Server Version 6.0.x
- ▶ IBM WebSphere Application Server Network Deployment Version 6.0.x

All combinations of Version 5.x products, Version 6.0.x products, and Version 6.1 products can coexist. WebSphere Application Server Version 5.x and Version 6.0.x clients can coexist with Version 6.1 clients.

Advantages

This topology is designed to maximize performance, throughput, and availability. It incorporates the benefits of the other distributed server topologies and adds the possibility to mix different operating systems. Advantages include:

- ▶ Single point of failure is eliminated due to the redundancy of components.
- ▶ It provides both hardware and software failure isolation. Hardware and software upgrades can be easily handled during off-peak hours.
- ▶ Horizontal and vertical scaling can be implemented to maximize availability.

Disadvantages

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

For information about planning and system considerations required to build a heterogeneous cell, see *WebSphere for z/OS -- Heterogeneous Cells*, at:

<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100644>

Planning for installation

This chapter provides a general guide to planning the installation of WebSphere Application Server and any of its components. To effectively plan an installation, you need to select a topology and the hardware and operating systems you plan to use.

This chapter contains the following sections:

- ▶ What is new in V6.1
- ▶ Selecting a topology
- ▶ Selecting hardware and operating systems
- ▶ Naming conventions
- ▶ Planning for WebSphere Application Server
- ▶ Planning for the Web server and plug-ins
- ▶ Planning checklist for the installation

This document does not describe the installation process. Refer to the product documentation during the installation.

6.1 What is new in V6.1

This section outlines some of the major changes from WebSphere Application Server V6.0 to WebSphere Application Server V6.1. For a complete list of new and improved items for installers, see the Information Center under the “What is new in this release” topic for each WebSphere Application Server package.

The following installation-related items are new for the Base and Network Deployment packages for WebSphere Application Server V6.0.2 and V6.1 for *distributed systems*:

- ▶ Installation:
 - Support added for non-root installations.
 - A new Installation Factory feature that enables you to create custom installation packages.
- ▶ Silent installations:
 - A single response file can be used for installation and profile creation. In V6.0, two response files are required.
 - Option names have changed to names that are more user-friendly and are less likely to change between releases.
- ▶ Profiles:
 - Profiles can be created directly from the installer.
 - A new profile called the Cell profile enables the creation of a deployment manager, a federated node, and an application server on a single server.
 - Ports used by WebSphere Application Server installations are automatically detected. The wizard defaults to unused ports but provides a button to revert to the standard ports.
 - Administrative security can be enabled during profile creation.
- ▶ Enhanced integration for the Linux operating system:
 - Menu entries and startup scripts created during the installation.
 - New support for Linux services.
- ▶ The installation verification utility detects inconsistency and corruption in WebSphere Application Server, application client, IBM HTTP Server, Web server plug-ins, and Update Installer installations.

The following installation-related items are specific to *WebSphere Application Server for z/OS V6.1 systems*:

- ▶ A workstation-based Profile Management Tool is now available as an alternative to the host-based ISPF Customization Dialog. This tool uses the same worksheets and overall customization flow as the Customization Dialog, and also provides new functions, such as the ability to create a complete Network Deployment cell in one pass, including an application server.
- ▶ Customization tools enable the customization file system to be created as either an HFS or zFS file system. The zFS file system has significant performance advantages.
- ▶ WebSphere Application Server requires a minimum z/OS level of Version 1 Release 6. Because Version 1 Release 6 always supports WLM application environments, the customization steps and documentation for static application environments have been removed.

The Web server plug-in installation includes the following changes:

- ▶ The IBM HTTP Server installation now includes the Web server plug-in for IBM HTTP Server, eliminating an extra step. You can enter information in just one place when setting up the Web server environment.
- ▶ The separate plug-ins installer installs the Web server plug-ins on a machine on which WebSphere Application Server is not installed.
- ▶ New convenience scripts added to the installation package, enabling you to configure another instance of the Web server post-installation. Version 6.0 requires running the Plug-ins installer again to accomplish this.

6.2 Selecting a topology

Chapter 5, “Topologies” on page 59 describes some common configurations that illustrate single-tier and multi-tier, sometimes complex, environments. Each topology description contains information about the software products required and the information needed to create the WebSphere Application Server runtime environment.

After identifying the topology that best fits your needs, you 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

The next logical step is to decide what platforms you will use to map the selected topology to a specific hardware. These selections are driven by several factors including existing conditions, future growth, cost, and skills within your company.

When you choose the platform or platforms, you can then determine the specific configuration of each server by selecting the processor features, the amount of memory, and the number of direct-access storage device (DASD) arms and storage space that are required.

Along with selecting the hardware comes the operating system selection. The operating system must be at a supported version with a proper maintenance level installed for WebSphere Application Server to work properly.

For an updated list on the hardware and software requirements and supported platforms for WebSphere Application Server V6.1, see the system requirements for WebSphere Application Server V6.1, available at:

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

6.4 Naming conventions

Naming conventions make the runtime environment more comprehensible. Using a consistent naming convention helps standardize the structure of the environment and allow for easy expansion.

Naming conventions should be developed, established, and maintained for the hardware and networking infrastructure, as well as the WebSphere infrastructure, applications, and resources. When it comes to naming, most companies have already developed a working naming convention for existing infrastructure, and it is usually best to adhere to the existing convention instead of trying to invent new one specific to WebSphere.

Because naming conventions are also related to many different aspects of a company, they will vary depending on the characteristics of the environment. However, with a proper naming convention, you should be able to understand the purpose of an artifact by just looking at its name.

6.5 Planning for WebSphere Application Server

WebSphere Application Server V6.1 is a full installation, not an upgrade installation.

Before installing WebSphere Application Server, you need to determine on which systems you need to install and how you plan to create the necessary profiles. In addition, 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 each WebSphere Application Server package that is tailored to each platform. This section gives you a high-level look at the planning tasks you need to perform.

If you are planning a WebSphere Application Server for z/OS environment, we strongly suggest that you review the following documents:

- ▶ For more information and examples of defining a naming convention for WebSphere for z/OS, see *WebSphere z/OS V6 -- WSC Sample ND Configuration*, available at:

<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100653>

A spreadsheet goes with this document that will help you create and document your names. Download the spreadsheet from the same URL.

- ▶ For information about differences you in V6.1, see *WebSphere for z/OS V6.1 - New Things Encountered During Configuration*, available at:

<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100781>

Address the following planning items before starting the installation of WebSphere Application Server:

- ▶ Determine whether to perform a single install or multiple

The norm is to install WebSphere Application Server once on a machine and create multiple runtime environments using profiles. Each profile has its own configuration data but shares the product binaries. In some instances (test environments, for example), you might want to install multiple instances.

- ▶ Select an installation method

You have multiple options for the installation. Your choice will be influenced by several factors, including the size of the installation (how many systems), the operating systems involved, how many times you anticipate performing the same installation (should you use the Installation Factory feature?), and if you will be performing remote installations with unskilled personnel.

- ▶ Plan for profiles

The environment is defined by creating profiles. You need to determine the types of profiles you will need and on which systems you will need to install them. Each topology discussed in Chapter 5, “Topologies” on page 59 includes profile information.
- ▶ Plan for names

Naming conventions can be an important management tool in large environments.
- ▶ Plan for TCP/IP port assignments

Each application server, node agent, and deployment manager uses a series of TCP/IP ports. These ports must be unique on a system.
- ▶ Security considerations for the installation

Security for WebSphere falls into two basic categories: administrative security and application security. During the installation, you will have the option to enable administrative security. Plan a scheme for identifying administrative users, their roles, and the user registry you will use to hold this information.

6.5.1 Determine whether to perform a single install or multiple

You can install WebSphere Application Server V6.1 several times on the same machine in different directories. Those installations are independent from 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.

When you have a single installation of WebSphere Application Server V6.1, you can create multiple application server profiles. In this case, all profiles share the same product binaries. When fixes are installed, they affect all profiles. Each profile has its own user data.

Figure 6-1 shows the difference between multiple installations and multiple WebSphere profiles in a stand-alone server environment (Base and Express).

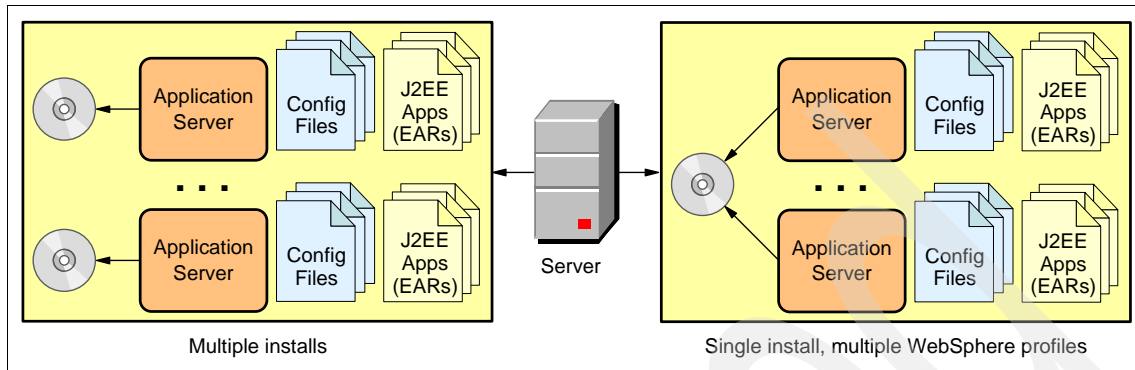


Figure 6-1 Stand-alone server installation options

Note: 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 machine (multiple installs), each one for administering different cells. Or, you can install Network Deployment once and then create multiple profiles so that each profile is used to administer a different cell.

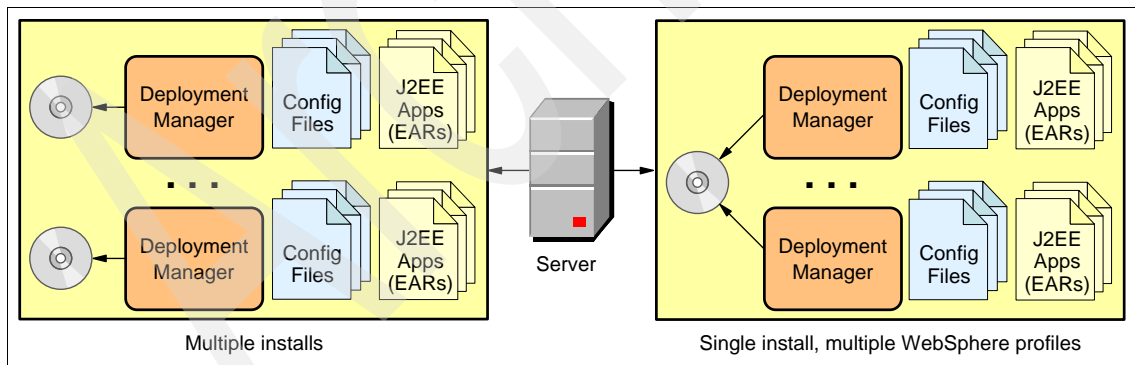


Figure 6-2 Deployment manager installation options

Using multiple installations and multiple profiles

Another possibility is the combination of multiple installation instances and multiple profiles. Figure 6-3 illustrates a Network Deployment environment where multiple runtime environments have been created using profiles.

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.

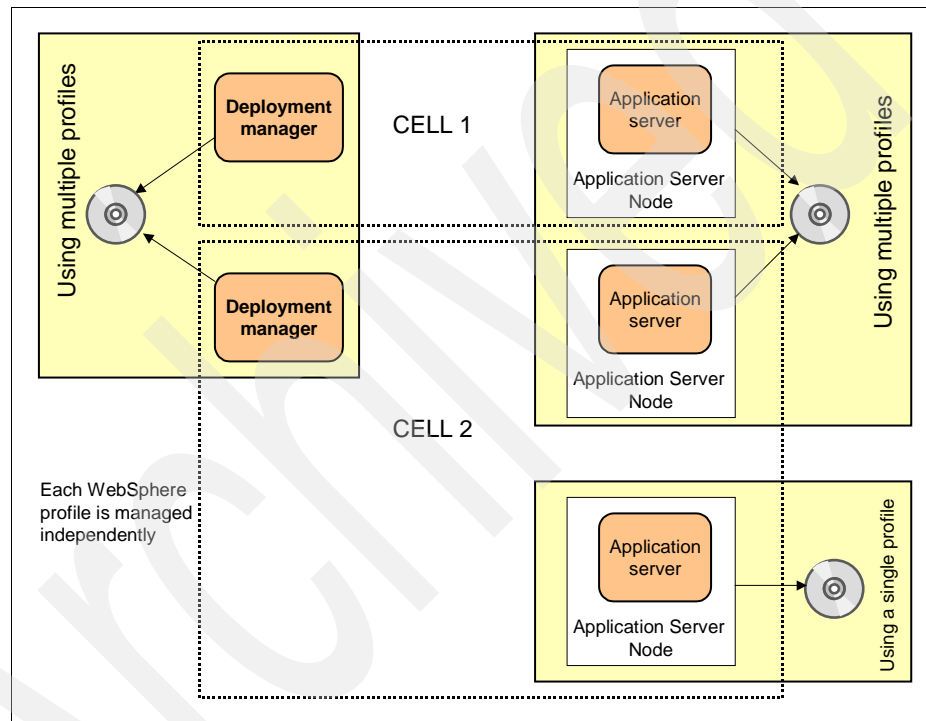


Figure 6-3 Cell configuration flexibility

6.5.2 Select an installation method

Before starting installation activities, review the options you have for installing the code and select the option that best fits your needs.

WebSphere Application Server for z/OS is installed using SMP/E.

On distributed systems, you have several choices for installation:

▶ Graphical install interface

The installation wizard is suitable for installing WebSphere Application Server on a small number of systems. Executing the installation wizard will install one system. You can start with the Launchpad, which contains a list of installation activities to select, or you can execute the installation program directly.

The installer checks for the required operating system level, sufficient disk space, and user permissions. If you fail any of these, you can choose to ignore the warnings; however, ultimately, the installation might fail.

▶ Silent installation

To install on multiple systems or remote systems, use the silent installation. This option enables you to store installation and profile creation options in a single response file, and then issue a command to perform the installation and (optionally) profile creation.

As with the wizard, the installer checks for the required prerequisite conditions. If any of these checks fail, the installation will stop. You can override the check in the response file by specifying:

```
disableOSPrereqChecking="true"
```

▶ Installation Factory (new with V6.0.2)

The Installation Factory provides an automated method for creating custom installation packages (CIPs). These packages include product maintenance, enterprise applications, and configuration actions in order to improve installation repeatability.

This solution provides an easy-to-use and reliable installation that can be used to install or update a product by selecting which fix packs should be included. However, there are some things to consider when taking this approach:

- The combination of multiple products and maintenance can make packages very large.
- Because there are multiple steps involved, installations can take a long time and the reason might not be visible to the installer.
- If any steps fail, the entire installation will fail.
- Creating a CIP might not be a trivial matter.

6.5.3 Plan for profiles

The installation of WebSphere Application Server essentially gives you the product files required to create a runtime environment. However, the actual runtime is defined through the use of profiles. The product binaries remain unchanged after installation until you install maintenance. All server processes that share the binaries use the updated level of the system files after installing the service.

During the installation process, you have the option to create profiles that build the runtime configuration files.

Note the following planning considerations for profiles:

- ▶ What profile types you need.
- ▶ How to create the profiles.
- ▶ Where to store the profile configuration files. Profiles can be stored under the installation root for WebSphere or in any other location you choose. In z/OS, the configuration is stored on an HFS or zFS.

Although an HFS can be shared across multiple MVS images in a parallel sysplex, experience has shown that there is a performance cost associated with doing this in a WebSphere environment. In most cases, it can be better to create a file system for each node, but of course it all depends on your system requirements.

Note: This section is intended to help you plan for the profiles you will need and the method you want to use to create them. *WebSphere Application Server V6.1: System Management and Configuration, SG24-7304*, describes profile creation options in more detail.

Profile types

The types of profiles available to you depend on the WebSphere Application Server package that you have installed. The profiles types that you need are determined by your topology.

The profile types are:

- ▶ Application server profiles

An application server profile includes default applications and the server1 application server. The application server in the Network Deployment product can run as a managed node or as a stand-alone application server.

The stand-alone application server is the same as the one in the Express product and in WebSphere Application Server V6.1 with one important

exception: You can add Network Deployment stand-alone application server nodes to a cell under the centralized management of the deployment manager.

► Deployment manager profiles

The deployment manager profile creates the deployment manager process (dmgr). The deployment manager provides centralized administration of multiple application server nodes and custom nodes as a single cell. The deployment manager provides administration for basic clustering and caching support, including failover support and workload balancing.

► Custom profiles

A custom profile (referred to as a managed node on z/OS) is an empty node that you must federate. Use the deployment manager to customize the node by creating servers and clusters. The node does not include a default application server or default applications.

► Cell profiles

A cell profile can be used to create a deployment manager, a federated node, and an application server on that node on a single system. It creates two profiles, one for the deployment manager and one for the node and application server.

Creating profiles during the installation

On distributed platforms, profiles are created during the installation, or after using the Profile Management Tool (for Express V6.0 this is done using the profile creation wizard). During the installation of Express and Base, an application server profile is created automatically during the installation. During a Network Deployment installation, you have the option to create a profile but do not have to.

Table 6-1 shows the application server environments that are created automatically during the product installation.

Table 6-1 Application server environments created during product installation

Product	Default environment	WebSphere profiles available
WebSphere Application Server - Express V6	Stand-alone application server (application server profile).	Application server profile.
WebSphere Application Server V6.1	Stand-alone application server (application server profile).	Application server profile.

Product	Default environment	WebSphere profiles available
WebSphere Application Server Network Deployment V6.1	Optional: The last installation panel lets you launch the Profile Management Tool if you want to create a profile.	All types.
WebSphere for z/OS	A default environment is not created during the installation of the core product files.	All types. Cell profiles must be built from the zPMT tool in the Application Server Toolkit.

Note: Profiles created during the installation process are created using the “typical” settings. See “Creating additional profiles” on page 97 for more information about what these will be.

Express installations

The installation procedure for WebSphere Application Server - Express V6 installs the core product files and creates a stand-alone application server in a profile named *default*. After the installation, you can create additional application server profiles using the profile creation wizard. The profile created during installation is placed under the `<was_home>/profiles` directory. Additional profiles that you create can be located anywhere on the file system.

WebSphere Application Server (base) installations

The installation procedure for WebSphere Application Server V6.1 installs the core product files and creates a stand-alone application server in a profile named *AppSrv01*. After install, you can create additional application server profiles using the Profile Management Tool. The profile created during installation is placed under the `<was_home>/profiles` directory. Additional profiles that you create can be located anywhere on the file system.

Network Deployment installations

The installation procedure for WebSphere Application Server Network Deployment V6.1 installs the core product files (product binaries) and gives you the option of creating a profile. For test environments, the Cell profile is a handy option to create a deployment manager and node. To create the environment for your chosen topology, refer to the specific topology section to get a list of the profiles required and on which the systems to create them.

After the installation, you can create additional profiles using the Profile Management Tool.

WebSphere for z/OS installations

The installation on WebSphere Application Server for z/OS uses SMP/E and only installs the product binaries. After the installation, you create profiles using the WebSphere Application Server for z/OS ISPF panels or using the Profile Management Tool (zPMT) available in the Application Server Toolkit.

Creating additional profiles

Creating profiles after the installation enables you to create additional runtime environments and to expand distributed server environments. Using the Profile Management Tool to create the profiles also enables you to take an “advanced” path through the profile creation, giving you more flexibility in the options you take.

Deployment manager profile options

Table 6-2 shows a summary of the options available when creating a profile for a deployment manager. The options depend on whether you take the typical or advanced path through the Profile Management Tool.

Table 6-2 Deployment manager profile options

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

Application server profile options (Base and Network Deployment)

Table 6-3 shows a summary of the options available when creating a profile for an application server in a Base or Network Deployment installation. The options depend on whether you take the typical or advanced path through the Profile Management Tool.

Table 6-3 Application server profile options: V6.1

Typical settings	Advanced options
The administrative console and default application are deployed by default. The sample applications are not deployed.	You have the option to deploy the administrative console (recommended), the default application, and the sample applications (if installed).
The profile name is AppSrvxx by default, where xx is 01 for the first application server profile and increments for each one created. The profile is stored in <was_home>/profiles/AppSrvxx.	You can specify profile name and its location.
The profile is not the default profile.	You can choose whether to make this the default profile. (Commands run without specifying a profile will be run against the default profile.)
The application server is built using the default application server template.	You can choose the default template, or a development template that is optimized for development purposes.
The node name is <host>Nodexx. The host name defaults to your system's DNS host name.	You can specify the node name and host name.
You can enable security (yes or no). If you select yes, you will be asked to specify a user name and password that will be given administrative authority.	
TCP/IP ports will default to a set of ports not used by any profiles in this WebSphere installation instance.	You can use the recommended ports (unique to the installation), use the basic defaults, or select port numbers manually.
(Windows) The deployment manager will be run as a service.	(Windows) You can choose whether the deployment manager will run as a service.
Does not create a Web server definition.	Enables you to define an external Web server to the configuration.

Application server profile options (Express V6.0)

Table 6-4 on page 99 shows a summary of the options available when creating a profile for a an application server in Express V6.0. The profile creation wizard does not contain a quick (typical) path as the new Profile Management Tool does.

Table 6-4 Application server profile options: Express V6.0

Advanced options
You can specify profile name and its location.
You can choose whether to make this the default profile. (Commands run without specifying a profile will be run against the default profile.)
The application server is built using the default application server template.
You can specify the node name and host name.
You can use the recommended ports (unique to the installation) or select port numbers manually.
(Windows) You can choose whether the deployment manager will run as a service.
The default application and the sample applications are deployed.

Cell profile options

Table 6-5 shows a summary of the options available when creating a cell profile. Using this option actually creates two distinct profiles, a deployment manager profile and an application server profile. The application server profile is federated to the cell. The options you see are a reflection of the options you would see if you were creating the individual profiles versus a cell.

Table 6-5 Cell profile options

Typical settings	Advanced options
The administrative console and default application are deployed by default. The sample applications are not deployed.	You have the option to deploy the administrative console (recommended), the default application, and the sample applications (if installed).
The profile name for the deployment manager is Dmgrxx by default, where xx is 01 for the first deployment manager profile and increments for each one created. The profile is stored in <was_home>/profiles/Dmgrxx.	You can specify the profile name and its location
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 one created. The profile is stored in <was_home>/profiles/AppSrvxx.	You can specify the profile name and its location.
Neither profile is made the default profile.	You can choose to make the deployment manager profile the default profile.

Typical settings	Advanced options
<p>The cell name is <i><host>Cellxx</i>. The node name for the deployment manager is <i><host>CellManagerxx</i>. The node name for the application server is <i><host>Nodexx</i> . The host name defaults to your system's DNS host name.</p>	<p>You can specify the cell name, the host name, and the profile names for both profiles.</p>
<p>You can enable security (yes or no). If you select yes, you will be asked to specify a user name and password that will be given administrative authority.</p>	
<p>TCP/IP ports will default to a set of ports not used by any profiles in this WebSphere installation instance.</p>	<p>You can use the recommended ports for each profile (unique to the installation), use the basic defaults, or select port numbers manually.</p>
<p>(Windows) The deployment manager will be run as a service.</p>	<p>(Windows) You can choose whether the deployment manager will run as a service.</p>
<p>Does not create a Web server definition.</p>	<p>Enables you to define an external Web server to the configuration.</p>

Custom profile options

Table 6-6 shows a summary of the options available when creating a custom profile.

Table 6-6 Custom profile options

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

Starting the Profile Management Tool

After the installation of Base or Network Deployment V6.1 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.
- ▶ By executing the **pmt.bat (sh)** command to start the Profile Management Tool. For operating systems such as AIX 5L or Linux, the command is in the `<was_home>/bin/ProfileManagement` directory. For the Windows platform, the command is in the `<was_home>\bin\ProfileManagement` directory.

The Profile Management Tool provides a graphical interface to the `<was_home>/manageprofiles.bat (sh)` command. You can use this command directly to manage (create, delete, and so on) profiles without a graphical interface.

For Express V6.0, start the profile creation wizard in one of the following ways:

- ▶ From the First Steps window. The First Steps window opens after a profile has been created.
- ▶ On Windows systems, from the Start menu.
- ▶ The command is in the `<was_home>/bin/ProfileCreator` directory. The name of the command varies per platform:
 - **pctAIX.bin**
 - **pctHPUX.bin**
64-bit platforms: **pctHPUXIA64.bin**
 - **pctLinux.bin**
64-bit platforms: **pct.bin**
S/390® platforms: **pctLinux390.bin**
POWER™ platforms: **pctLinuxPPC.bin**
 - **pctSolaris.bin**
 - **pctWindows.exe**
64-bit platforms: **pctWindowsIA64.exe**

In WebSphere for z/OS, profiles are created using a series of jobs. The JCL for the jobs is created using ISPF panels or through the Profile Management Tool for z/OS, referred to as the zPMT tool, in the Application Server Toolkit.

To start the zPMT:

1. Open the Application Server Toolkit.
2. Select **Windows** → **Preferences**.
3. Expand **Server** and click **WebSphere for z/OS**.
4. Click **Create**.

Profile location

Profiles created as a part of the installation are automatically placed in the `<was_home>/profiles` directory.

When you create profiles after the installation, you can designate the location where the profiles are stored. When deciding where to keep the profiles, consider performance, backup capabilities, and availability.

6.5.4 Plan for names

The purpose of developing systematic naming concepts and rules for a WebSphere site is two-fold:

- ▶ To provide guidance during setup and configuration
- ▶ In case of issues, to quickly narrow down the source of the issue

Naming the WebSphere infrastructure artifacts, such as cells, nodes, application servers, and so on should follow the company's normal naming conventions as far as possible. Discuss all parts of the installation planning, particularly the naming concepts, with the subject matter experts that are involved, and document decisions and results in writing. Failure to do so can affect operational processes for the future site and has proved in the past to be a root source for labor-intensive issues within client installations.

Naming conventions

Some considerations to be taken into account when developing the key concepts for the site during the installation planning are:

- ▶ Naming profiles

The profile name can be any unique name, but it is a good idea to have a standard for naming profiles. This will help administrators easily determine a logical name for a profile when creating it and will help them 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 so forth)
- Slashes (/) or (\)

► Naming cells

A cell represents an administrative domain.

In a stand-alone environment, the cell name is not usually visible to administrators and a naming convention is not required. The name is automatically generated during profile creation, and will be in the following format:

`<system_name><node_name><number>Cell`

The `<number>` will increment, starting with “01”, with every new node, for example, `server1Node01Cell`, `server1Node02Cell`, and so on.

In a distributed server environment, there are considerations for naming a cell. A cell name must be unique in any circumstance in which the product is running on the same physical machine or cluster of machines, such as a sysplex. Additionally, a cell name must be unique in any circumstance in which network connectivity between entities is required either between the cells or from a client that must communicate with each of the cells. Cell names also must be unique if their name spaces are going to be federated.

Often a naming convention for cell names will include the name of the stage (such as integration test, acceptance test, production) and the name of the department or project owning it, if appropriate. But a proper naming convention also depends on the size of the infrastructure. In a small company, there might be only be just a few WebSphere cells in total, but in a large company, there can be a lot more cells to accommodate test environments for several projects as well as a highly available production environment.

► Naming nodes

In a stand-alone environment, you will 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 take the default, the node name will be in the format:

`<system_name>NODE<number>`

The `<number>` will increment, starting with “01”, with every new node, for example, `server1Node01`, `server1Node02`, and so on.

In a distributed server environment, the node must be unique within a cell. Nodes generally represent a system and will often include the host name of the system, but this is not necessary. You can have multiple nodes on a system.

Naming conventions for nodes often include the physical machine name where they are running, such as `NodeA123` if the server name is `ServerA123`.

► Naming application servers

In stand-alone environments, the server name will always be “server1”. Note that when you federate a stand-alone application server to a cell, you will have a unique combination of node name and server1, thus allowing multiple “server1”s to exist in the cell.

In distributed server environments, the same applies. If you create an application server profile, you get a new node and stand-alone server. The server name will be server1. However, it is more likely that new application servers in a distributed server environment will be created on a federated node using the administrative console or other administrative tool. In this case, the server can be named and a meaningful name should be assigned. Whether you choose to name servers based on their location, function, membership in a cluster, or some other scheme will largely depend on how you anticipate your servers being used and administered.

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, which makes it easier to find them in the WebSphere administrative console.

If an application server hosts multiple applications, develop some other kind of 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:

- Cells
- Nodes
- Servers
- Clusters
- Applications
- Deployments

When you create a new object 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 first character. The name string also cannot contain leading and trailing spaces.

WebSphere Application Server for z/OS considerations

Naming conventions take on a special importance when running WebSphere for z/OS due to MVS length restrictions. The concept of long and short names is used in WebSphere Application Server for z/OS to accommodate these restrictions. The long name of a node, cell, or server, for example, is the equivalent of the names you assign on other platforms. There will be a corresponding short name that is limited to eight characters and must be uppercase. Long names can be much longer and can be mixed case.

During the installation process, you will be asked to provide long and short names when defining the cell, node, and each application server. Bear in mind that procedures and jobs are also created to support these components. For example, the deployment manager will consist of multiple address spaces, including at a minimum, a control region, a servant region, and a daemon. Naming for these elements also becomes important to avoid confusion.

Your approach should ensure that you will avoid name conflicts among other applications running on the same environment and make sure that all names are unique within the MVS image at least.

For more information and examples of defining a naming convention for WebSphere for z/OS, see *WebSphere z/OS V6 -- WSC Sample ND Configuration*, available at:

<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100653>

To go with this document, a spreadsheet has been developed that will help you create and document your names, *WebSphere for z/OS Version 6 - Configuration Planning Spreadsheet*, available at:

<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/PRS1331>

6.5.5 Plan for TCP/IP port assignments

The port assignment scheme for a WebSphere site should be developed in close cooperation with the network administrators. From a security point-of-view, it is highly desirable to know each and every port usage ahead of time, including the process names and the owners using them.

Depending on the chosen WebSphere Application Server configuration and hardware topology, the setup even 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 utilization, the following approach is reliable and reduces the complexity of such a scenario considerably:

- ▶ Discuss and decide on with the network administration a fixed range of continuous ports for exclusive use for the WebSphere Application Server installation.
- ▶ Draw the overall WebSphere Application Server topology and map your application life cycle stages onto WebSphere profiles.
- ▶ List the number of required ports per WebSphere profile and come up with an enumeration scheme, using appropriate increments at the cell, node, and application server level, starting with your first cell. You can use a spreadsheet and develop this spreadsheet as part of your installation documentation. The same spreadsheet also can serve for the server names, process names, user IDs, and so forth.

Note that the PMT can identify the ports used in the same installation on that system and those ports that are currently in use and will suggest unique ports to use. The `updatePorts.ant` script is also included in `<was_home>/profileTemplates/<template_name>/actions` to help you change port numbers quickly after profile creation.

For a list of the ports used by WebSphere Application Server and their default settings, see the “Port number settings in WebSphere Application Server versions” topic in the WebSphere Application Server Information Center. The URL for the Network Deployment version of this article is:

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

For z/OS installations, use the *WebSphere for z/OS Version 6 - Configuration Planning Spreadsheet* to help you prepare for port assignments:

<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1331>

6.5.6 Security considerations for the installation

To plan a secure WebSphere Application Server environment, it is imperative that you have highly skilled security specialists that can evaluate your business and network security needs. You need to have a fairly clear idea of your plans for security before installation of any production systems. We give an overview of the aspects of security specific to WebSphere Application Server in Chapter 12, “Planning for security” on page 253. For more details, refer to *WebSphere Application Server V6 Security Handbook*, SG24-6316.

There are some specific considerations for installers that we address here. 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 will be 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.
- ▶ Network and physical security
 - Usually a firewall is part of the topology. After determining what ports 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, root on a Linux or UNIX operating system and a member of the administrator group on a Windows operating system. Non-root installation is also supported.
 - If there is a policy on password expiration, it should be well known to avoid disruption on the service (password expiration of root, QSECOFR, Administrator, or the password of the user to access some database).

Root versus non-root installation

The term non-root implies a Linux or UNIX installer, but also means a non-administrator group installer on a Windows system. Non-root installers can install V6.1 in both silent and interactive mode for full product installations and removals, incremental feature installations, and silent profile creation.

Installing as a non-root user in Version 6.1 works the same as installing as a root user does in previous versions. However, there are limitations to be aware of. For a list of these limitations, see the “Limitations of non-root installers” topic in the WebSphere Application Server Information Center, available at:

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

Secure administration tasks

WebSphere Application Server provides a mechanism to secure the administrative functions.

Fine-grained administrative security (new):

In releases prior to WebSphere Application Server Version 6.1, users granted administrative roles could administer all of the resource instances under the cell. With V6.1, administrative roles are now assigned per resource instance rather than to the entire cell. Resources that require the same privileges are placed in a group called the authorization group. Users can be granted access to the authorization group by assigning to them the required administrative role within the group.

A cell-wide authorization group exists for backward compatibility. Users assigned to administrative roles in the cell-wide authorization group can still access all of the resources within the cell.

In order for a user ID to have administrative authority, it must be assigned to one of the following roles:

- ▶ **Monitor**
The Monitor role has the least permissions. This role primarily confines the user to viewing the configuration and current state.
- ▶ **Configurator**
The Configurator role has the same permissions as the Monitor, and in addition, can change the configuration.
- ▶ **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 Configurator and the permission required to access sensitive data, including server password, Lightweight Third Party Authentication (LTPA) password and keys, and so on.
- ▶ **Deployer (for wsadmin users only)**
Users granted this role can perform both configuration actions and runtime operations on applications.
- ▶ **iscadmins (for administrative console users only)**
An individual or group that uses the iscadmins role has Administrator privileges for managing users and groups in the federated repositories from within the administrative console only.

- ▶ `adminsecuritymanager` (for `wsadmin` users only)

Only users who are assigned to this role can assign users to administrative roles. Also, when fine-grained administrative security is used, only users who are assigned to this role can manage authorization groups.

With V6.1, you have the option to enable security for administrative tasks during profile creation for an application server or deployment manager (including those created with Cell profiles). Note that this option does not enable application security.

If you intend to create a profile during installation and want to secure your administrative environment at the same time, you need to identify one user ID to be used for administration. The user ID and password specified during profile creation will be 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, LTPA is used as the authentication mechanism.

On distributed systems, a file-based user repository is created and populated with the administrator ID. This 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. In WebSphere for z/OS, you can choose to use the file-based repository or use the z/OS system SAF-compliant security database.

Whether you choose to enable administration security during profile creation or after, it is important that you do it before going into production.

6.6 Planning for migration

The IBM Redbook *WebSphere Application Server V6 Migration Guide*, SG24-6369, thoroughly discusses migration of both the runtime configuration and applications. This section gives a very brief summary of the migration paths and options available for the WebSphere runtime configuration.

There are three ways to migrate an existing WebSphere Application Server cell to V6.1:

- ▶ Fully automated migration
The fully automated migration uses the **WASPreUpgrade** and **WASPostUpgrade** tools to perform the migration. Strictly speaking, the migration is not fully automatic. It is more accurate to say that the configuration of the migrated elements is automated. We refer to this approach as “fully automated” because we use the migration tools on every node in the cell.
- ▶ Partially automated migration
Like the fully automated migration, this method uses the migration tools, but only to migrate the deployment manager. You might use this method if you are not 100% confident in the abilities of the migration tools and you need a mixed version cell (automated migration of the deployment manager is the only way to include Version 5.x nodes in a Version 6 cell).
- ▶ Manual migration
In a manual migration, a new cell is created to first augment and then replace the existing Version 5.x cell. You might use this method if you need to keep Version 5.x running for an extended period (that is, you require an extended period of interoperability) and you have adequate hardware to support a second deployment manager.

WebSphere Application Server V6.1 includes automatic migration utilities that help you transform an operational WebSphere configuration on a prior release to V6.1. The automatic migration utilities save you a lot of time by eliminating the need to create the new configuration manually.

Table 6-7 shows the products that the automatic migration utilities support as a starting point.

Table 6-7 Supported WebSphere releases for migration starting points

Version 5.0	Version 5.1	Version 6.0
WebSphere Application Server	WebSphere Application Server	WebSphere Application Server
WebSphere Application Server - Express	WebSphere Application Server - Express	WebSphere Application Server - Express
WebSphere Application Server Network Deployment	WebSphere Application Server Network Deployment	WebSphere Application Server Network Deployment
WebSphere Application Server Enterprise	WebSphere Business Integration Server Foundation	WebSphere Business Integration Server Foundation
WebSphere Application Server for z/OS	WebSphere Application Server for z/OS	WebSphere Application Server for z/OS

If your existing version is not listed in Table 6-7 on page 110, you must perform a manual migration.

6.7 Planning for the Web server and plug-ins

The options for defining and managing Web servers depend on your chosen topology and your WebSphere Application Server package. Decisions to make include 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 installing a supported Web server and the appropriate Web server plug-in and defining the Web server to WebSphere Application Server.

The following examples outline the process required to create each sample topology. Note that each example assumes that only the WebSphere processes shown in the diagrams are installed on each system and that the profile for the process is the default profile.

This is not a substitute for using the product documentation, rather it is intended to help you understand the process. For detailed information about how the Plug-ins installation wizard works and the logic it follows to determine how to create the configuration scripts, see the *Getting Started with Web server plug-ins* guide that comes with the plug-in.

During the plug-in installation, you will be asked if the installation is local or remote. Depending on the response, a certain path through the installation will occur. Figure 6-4 on page 112 illustrates the plug-in installer behavior in more detail. You will see this in the examples.

Note: When installing the IBM HTTP Server shipped with WebSphere Application Server, you now have the option to install the Web server plug-in at the same time. If you choose this option, you will automatically get a *remote* installation.

The location for the plug-in configuration file is `<rhs_install>/Plugins/config/`.

The location for the plug-in configuration script is `<rhs_install>/Plugins/bin`.

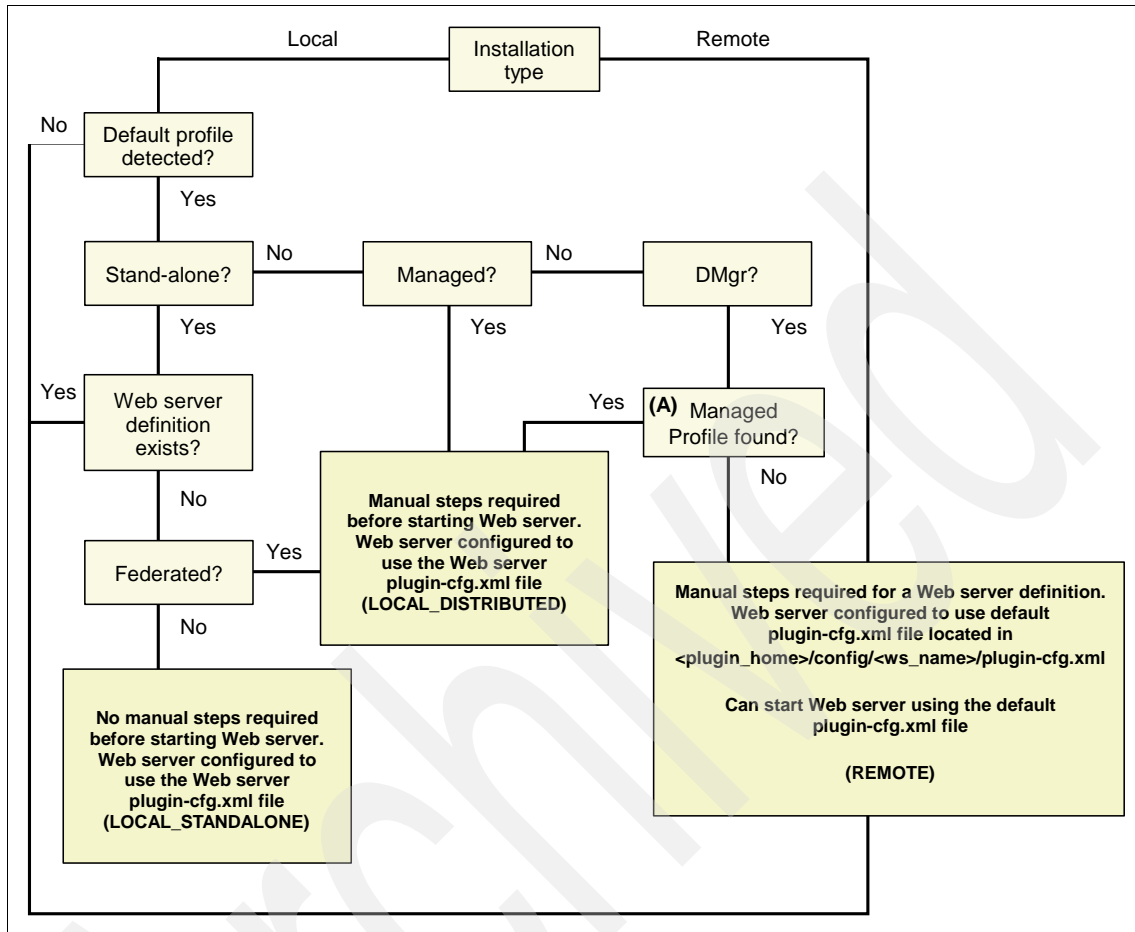


Figure 6-4 Web server plug-in installer behavior

The plug-in installer maps all possible configurations to three scenarios, LOCAL_STANDALONE, LOCAL_DISTRIBUTED, and REMOTE, as depicted in Figure 6-4:

► LOCAL_STANDALONE

What classifies as a LOCAL_STANDALONE plug-in configuration?

A default unfederated stand-alone profile that has no existing Web server definition is a LOCAL_STANDALONE plug-in configuration.

What tasks does the plug-ins Installation Wizard perform in this case?

- It creates a Web server definition for the default stand-alone profile.
- It configures the Web server to use the plugin-cfg.xml file.

Note: If the stand-alone profile is federated, you need to re-create 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

What classifies as a LOCAL_DISTRIBUTED plug-in configuration?

- A stand-alone profile that has been 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 detected. See (A) in Figure 6-4 on page 112.

What tasks does the Plug-ins installation wizard perform in this case?

- It does not create a Web server definition for the default distributed profile.
- It 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.

What is next?

- If the managed node is still not federated, federate the node first. This will avoid the Web server definition being lost after the federation has occurred.
- Run the manual Web server definition creation script.
- Start the Web server and WebSphere Application Server and hit the snoop servlet to verify that everything is working.

► REMOTE

What classifies as a REMOTE plug-in configuration?

- A remote install type selected by user at install time.
- A default deployment manager profile.
- No default profiles detected in the WebSphere Application Server directory given by user.
- A default, unfederated, stand-alone profile with an existing Web server definition.

What tasks does the plug-ins installation wizard perform in this case?

- It does not create a Web server definition for the default distributed profile.
- It configures the Web server to use the plugin-cfg.xml file in `<plugin_home>/config/<webserver_name>/plugin-cfg.xml`.

What is next?

If the user uses the default plugin-cfg.xml file in the `<plugin_home>` directory, start the Web server and WebSphere Application Server and select the snoop servlet to verify that everything is working.

To benefit from the Web server definition:

- 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_home>` 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 to verify that everything is working.

6.7.1 Stand-alone server environment

In a stand-alone server environment, a Web server can be remote to the application server machine or local, but there can only be one defined to WebSphere Application Server. The Web server always resides on an unmanaged node.

Remote Web server

In this scenario, 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. See Figure 6-5.

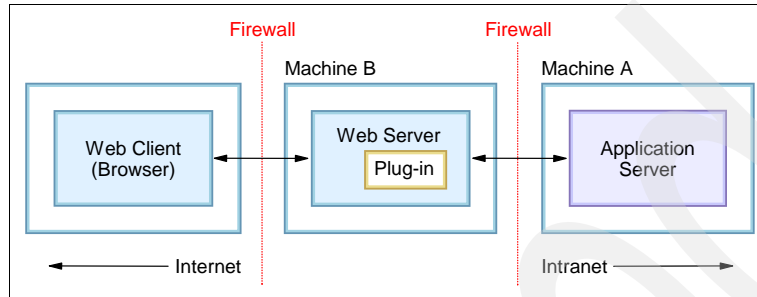


Figure 6-5 Remote Web server in a stand-alone server environment

Assume that the application server is already installed and configured on machine A. Perform the following tasks:

1. Install the Web server on machine B.
2. Install the Web server plug-in on machine B by performing the following steps:
 - a. Select **Remote** installation.
 - b. Enter a name for the Web server definition. The default is `webserver1`.
 - c. Select the location for the plug-in configuration file. By default, the location is under the config directory in the plug-in install directory. For example, when the name specified for the Web server definition in the previous step is `webserver1`, the default location for the plug-in file is:

```
<plugin_home>/config/webserver1/plugin-cfg.xml
```

During the installation, the following tasks are performed:

- a. A temporary plug-in configuration file is created and placed in the location specified.
- b. The Web server configuration file is updated with the plug-in configuration, including the location of the plug-in configuration file.
- c. A script is generated to define the Web server to WebSphere Application Server. The script is located in:

```
<plug-in_home>/bin/configure<web_server_name>
```

3. At the end of the plug-in installation, copy the script to the `<was_home>/bin` directory of the application server machine, machine A. Start the application server, and then execute the script.
4. 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, a stand-alone application server exists on machine A. The Web server and Web server plug-in are also installed on machine A. This topology is suited to a development environment or for internal applications. See Figure 6-6.

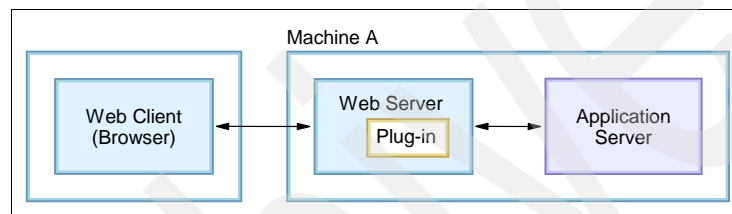


Figure 6-6 Local Web server in a stand-alone server environment

Assume that the application server is already installed and configured. Perform the following tasks:

1. Install the Web server on machine A.
2. Install the Web server plug-in on machine A by performing the following steps:
 - a. Select **Local** installation.
 - b. Enter a name for the Web server definition. The default is `webserver1`.
 - c. Select the location for the plug-in configuration file. By default, the location under the config directory in the profile for the stand-alone application server will be selected. For example, when the name specified for the Web server definition in the previous step is `webserver1`, the default location for the plug-in file is:

```
<profile_home>/config/cells/<cell_name>/nodes/webserver1_node/servers/webserver1/plugin-cfg.xml
```

Be aware that in a local scenario, the plug-in configuration file does not need to be propagated to the server when it is regenerated. The file is generated directly in the location from which the Web server reads it.

During the installation, the following tasks are performed:

- a. The plug-in configuration file is created and placed in the location specified.
- b. The Web server configuration file is updated with the plug-in configuration, including the location of the plug-in configuration file.
- c. The WebSphere Application Server configuration is updated to define the new Web server.

The plug-in configuration file is automatically generated. Because this is a local installation, you do not have to propagate the new plug-in configuration to the Web server.

6.7.2 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, and 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. Note that the process for this scenario is almost identical to that outlined for a remote Web server in a stand-alone server environment. The primary difference is that the script that defines the Web server is run against the deployment manager and you will see an unmanaged node created for the Web server node. In Figure 6-7, the node is unmanaged because there is no node agent on the Web server system.

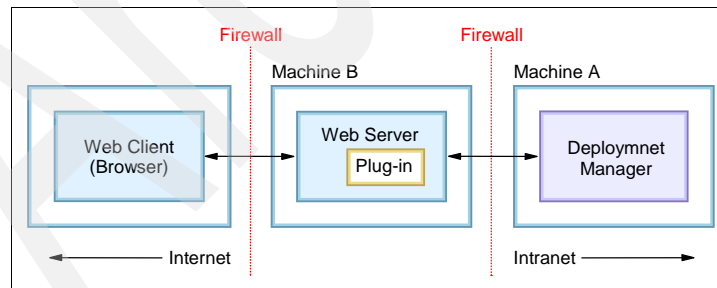


Figure 6-7 Remote Web server in a stand-alone server environment

Assume that the deployment manager is already installed and configured on machine A. Perform the following tasks:

1. Install the Web server on machine B.
2. Install the Web server plug-in on machine B by performing the following steps:
 - a. Select **Remote** installation.
 - b. Enter a name for the Web server definition. The default is `webserver1`.
 - c. Select the location for the plug-in configuration file. By default, the file will be placed in the directory that contains the server's configuration. For example, when the name specified for the Web server definition in the previous step is `webserver1`, the default location for the plug-in file is:

```
<plugin_home>/config/webserver1/plugin-cfg.xml
```

During the installation, the following tasks are performed:

- a. A temporary plug-in configuration file is created and placed in the location specified.
 - b. The Web server configuration file is updated with the plug-in configuration, including the location of the plug-in configuration file.
 - c. A script is generated to define the Web server and an unmanaged node to WebSphere Application Server. The script is located in:

```
<plugin_home>/bin/configure<web_server_name>
```
3. At the end of the plug-in installation, you need to copy the script to the `<was_home>/bin` directory of the deployment manager machine (machine A), start the deployment manager, and 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 to a federated application server (managed node)

In this scenario, the Web server is installed on a machine that also has a managed node. Note that this scenario would also be the same if the deployment manager was also installed on machine A. See Figure 6-8.

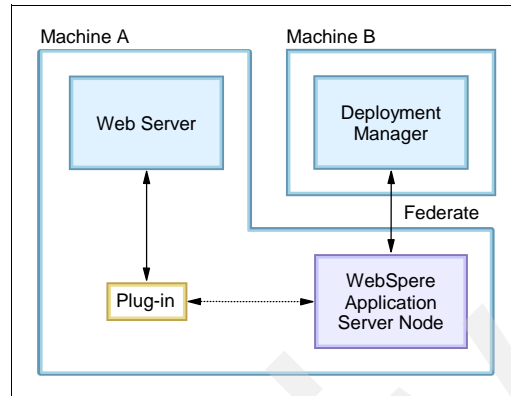


Figure 6-8 Web server installed locally on an application server system

Assume that the application server is already installed, configured, and federated to the deployment manager cell. Perform the following tasks:

1. Install the Web server on machine A.
2. Install the Web server plug-in on machine A by performing the following steps:
 - a. Select **Local** installation.
 - b. Enter a name for the Web server definition. The default is `webserver1`.
 - c. Select the location for the plug-in configuration file. By default, the file will be placed in the directory that contains the server's configuration. For example, when the name specified for the Web server definition in the previous step is `webserver1`, the default location for the plug-in file is:

```
<profile_home>/config/cells/<cell_name>/nodes/<AppSrv_node>/servers/webserver1/plugin-cfg.xml
```

During the installation, the following tasks are performed:

- a. The plug-in configuration file is created and placed in the location specified.
- b. The Web server configuration file is updated with the plug-in configuration, including the location of the plug-in configuration file.

- c. A script is generated to define the Web server and an unmanaged node to WebSphere Application Server. The script is located in:

```
<plug-in_home>/Plugins/bin/configure<web_server_name>
```

- 3. At the end of the plug-in installation, you need to execute the script to define the Web server from the location the wizard stored it in on machine A. Make sure that the deployment manager is running on Machine B. The deployment manager configuration will be updated and propagated back to machine A at node synchronization.

The plug-in configuration file is generated automatically and propagated at the next node synchronization.

6.8 Planning checklist for the installation

Table 6-8 provides a summary of items to consider as you plan and additional resources that can help you.

Table 6-8 Planning checklist for installation planning

Planning item
Examine your selected topology to determine hardware needs and software licenses. Create a list of what software should be installed on each system.
Determine what WebSphere Application Server profiles need to be created and whether you will create them during installation or after. Decide on a location for the files (see 8.4.1, "Configuration repository location and synchronization" on page 158).
Develop a naming convention that includes system naming and WebSphere Application Server component naming.
Develop a strategy for assigning TCP/IP ports to WebSphere processes.
Select an installation method (wizard, silent, Installation Factory).
Plan an administrative security strategy including user repository and role assignment.
Determine the user ID to be used for installation and whether you will perform a root or non-root installation. If non-root, review the limitations.
If migrating, review <i>WebSphere Application Server V6 Migration Guide</i> , SG24-6369.
Plan for the Web server and Web server plug-in installation. Determine if the Web server will be a managed or unmanaged server and note the implications. Create a strategy for generating and propagating the Web server plug-in configuration file.

Resources

WebSphere Application Server ships with an installation guide that can be accessed through the Launchpad.

The WebSphere Application Server Information Center also contains information that helps you through the installation process:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/welc6topinstalling.html>

Depending on the topology you selected, you might need additional information. Refer to following list of articles that you might find of interest. Note that the WebSphere Application Server Information Center articles point to the Network Deployment version.

For information about WebSphere Application Server for z/OS:

- ▶ *WebSphere for z/OS V6 Sample ND Configuration*
<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100653>
- ▶ *New Things You'll Encounter When Building a V6.1 Cell*
<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100781>
- ▶ *WebSphere for z/OS Version 6 - Configuration Planning Spreadsheet*
<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/PRS1331>

For information about the WebSphere application client:

- ▶ “Application Client for WebSphere Application Server”
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/ccli_appclients.html
- ▶ “Planning to install WebSphere Application Client”
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tins_scenario6.html

For information about installing and configuring Edge components, visit the IBM WebSphere Application Server Edge Component Information Center:

<http://www.ibm.com/software/webservers/appserv/ecinfocenter.html>

For the most current information about hardware and software requirements for Edge components, refer to the following Web page:

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

Archived

Planning for application development and deployment

This chapter highlights important topics you need to be aware of when planning for the development and deployment of WebSphere applications. This chapter contains items of interest for application developers, WebSphere infrastructure architects, and system administrators. It includes the following topics:

- ▶ What is new in V6.1
- ▶ End-to-end life cycle
- ▶ Development and deployment tools
- ▶ Naming conventions
- ▶ Source code management
- ▶ Test environments
- ▶ Managing application configuration settings
- ▶ Planning for application upgrades in production
- ▶ Mapping applications to application servers
- ▶ Planning checklist for applications

7.1 What is new in V6.1

The following list highlights the features added since V6.0:

- ▶ Application Server Toolkit enhancements

The Application Server Toolkit has shipped with WebSphere Application Server since Version 5.1, but with Version 6.1, it has been significantly improved and is now a full-blown integrated development environment (IDE). It can be used to build, test, and deploy J2EE applications on a WebSphere Application Server V6.1 environment (but not on any previous release). It has support for all J2EE artifacts supported by WebSphere Application Server V6.1, such as servlets, JSPs, EJBs, XML, and Web services, and also supports developing Java 5.0 applications.

- ▶ Portlet application support

The portlet container in WebSphere Application Server V6.1 provides the runtime environment for JSR 168 compliant portlets. Portlet applications are intended to be combined with other portlets to collectively create a single page of output. The portlet container takes the output of one or more portlets and generates a complete page that can be displayed.

The primary development tool for portlets on WebSphere Application Server portlet applications is the Application Server Toolkit. You can also use Rational Application Developer, but you should review the following item in the WebSphere Information Center:

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

Portlets are packaged in WAR files.

Note that the portlet runtime 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 168, see:

<http://jcp.org/en/jsr/detail?id=168>

- ▶ Session Initiation Protocol (SIP) support

SIP applications are Java programs that use at least one Session Initiation Protocol (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, click-to-call, and instant messaging.

The Application Server Toolkit provides special tools for developing SIP applications. SIP applications are packaged as SIP archive (SAR) files and are deployed to the application server using the standard WebSphere Application Server administrative tools. SAR files can also be bundled within a J2EE application archive (EAR file), just like other J2EE components.

For more information, see:

- JSR 116 SIP Servlet API 1.0 Specification
<http://www.jcp.org/aboutJava/communityprocess/final/jsr116/>
- RFC 3261
<http://www.ietf.org/rfc/rfc3261.txt>

7.2 End-to-end life cycle

The WebSphere Application Server V6.1 environment and its integration with Rational tools offers the developer support at every stage of the application development life cycle. Key stages in this life cycle are:

- ▶ Requirements gathering and analysis
- ▶ Prototyping
- ▶ High level design
- ▶ Low level design
- ▶ Implementation/coding/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 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.

RUP is an iterative process, which means that the cycle can feed back 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 level. 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 indeed back 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 basically 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 re-coding can take place a number of times. Figure 7-1 gives an overview of the Rational Unified Process.

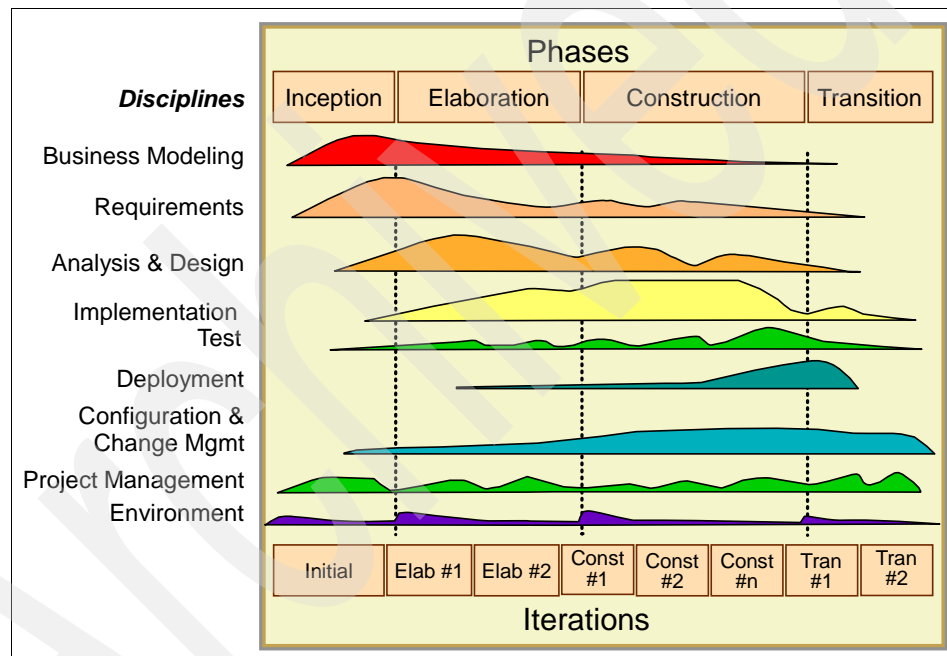


Figure 7-1 Rational Unified Process overview

RUP identifies a number of disciplines that are practiced during the various phases. These disciplines are practiced during all phases but the amount of activity in each phase varies. Clearly the requirements discipline will be more active during the earlier inception and elaboration phases, for example.

RUP maps disciplines to roles. There are many roles, but the roles break down into the following four 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; it is just a process specification after all. However, RUP provides specific guidance (called Tool Mentors) on how to use Rational Software when following the process. 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 “bought into” as much or as little as is required.

For more information about the Rational Unified Process, see:

<http://www.ibm.com/software/awdtools/rup>

7.3 Development and deployment tools

The WebSphere Application Server V6.1 environment comes with a rich set of development tools. All editions of WebSphere Application Server V6.1 include the Application Server Toolkit V6.1, which has been much improved since previous WebSphere releases and is now a full-blown J2EE development tool.

WebSphere Application Server - Express V6.0 comes with the Rational Web Developer V6.0. WebSphere Application Server and WebSphere Application Server Network Deployment come with a trial version of Rational Application Developer V6.0. For a full version of Rational Application Developer V6.0, additional licensing is required.

The Application Server Toolkit is targeted to support only the version of the WebSphere Application Server with which it ships. This means that Application Server Toolkit V6.1 supports all new features of WebSphere Application Server V6.1 and supports it as an integrated test environment. It does not, however, support any of the previous versions of WebSphere Application Server as integrated test environments.

At the time of writing, Rational Web Developer V6.0 and Rational Application Developer V6.0 have not yet been updated to support the new features of WebSphere Application Server V6.1. This means that, for example, they do not include support for developing SIP applications and they do not include support for WebSphere Application Server V6.1 as an integrated test environment. However, they do support previous versions of WebSphere Application Server (V5.1 and V6.0) as integrated test environments, which the Application Server Toolkit V6.1 does not.

To develop applications in Rational Web Developer V6.0 or Rational Application Developer V6.0 and test them on a WebSphere Application Server V6.1 test environment, you need to export them from the development environment as an EAR or WAR file and deploy that onto your server.

This limitation in the Rational product set will be fixed as new versions of the Rational products are released.

7.3.1 Application Server Toolkit V6.1

Application Server Toolkit was first shipped with WebSphere Application Server V5.1 and was originally only for the assembly, deployment, and debugging of J2EE applications on WebSphere. In WebSphere Application Server V6.1, it has been significantly enhanced and is now a full-blown development tool that can be used also for developing J2EE applications.

Application Server Toolkit V6.1 is based on the Eclipse 3.1.2 platform and inherits much of its functionality from the Eclipse Web Tools Platform, which is a relatively new Eclipse project to which IBM has been a major contributor. The Web Tools Platform is what provides the Web and J2EE concepts to Eclipse and, thus, the Application Server Toolkit.

The Application Server Toolkit V6.1 provides the following features:

- ▶ Java 5.0 support.
- ▶ Development of standard J2EE artifacts, such as servlets, JSPs, and EJBs complying with J2EE 1.2, 1.,3 and 1.4 specifications.
- ▶ Web services tools including wizards to generate Web services from Java beans, EJBs, and WSDL files and to consume Web services. Also includes UDDI test registry integration.
- ▶ Development of static Web projects (HTML, CSS style sheets, JavaScript™).
- ▶ SIP development, including support for JSR 116 SIP servlets.
- ▶ Portlet development (JSR 168).
- ▶ XML tools to build and validate XML artifacts, including schemas, DTDs, and XML files.
- ▶ Data tools for connecting to and interacting with various database vendors.
- ▶ WebSphere Enhanced EAR support.
- ▶ Support for annotation-based development (part of WebSphere rapid deployment).

- ▶ Support for WebSphere Application Server V6.1 test environments in either a local or remote configuration, but no support for any previous versions of WebSphere Application Server (such as 6.0 or 5.1).
- ▶ Jython script development, including script debugging capabilities.
- ▶ Jacl to Jython script conversion tools (jacl2jython).
- ▶ Integration with Concurrent Versions System (CVS), which is a popular Source Code Management (SCM) repository. (No integration with Rational ClearCase® is provided.)

To summarize, Application Server Toolkit V6.1 is a full-blown development environment that provides you with the tooling necessary to create, test, and deploy the various artifacts supported by WebSphere Application Server V6.1.

It does not, however, include the productivity-enhancing features and visual editors found in Rational Web Developer and Rational Application Developer. It also does not include Rational ClearCase, Crystal Reports, UML modeling, Struts, or JSF support, and it does not support any of the previous releases of WebSphere Application Server (such as 5.1 or 6.0) as test environments.

As of writing, the Application Server Toolkit V6.1 is also the only WebSphere or Rational development environment that fully supports development of Java 5.0 applications. Although you can plug a Java 5.0 JDK™ into Rational Web Developer V6.0 and Rational Application Developer V6.0 and use it to compile Java 5.0 code, the Java editors are still not updated for Java 5.0 and therefore not aware of the new features in the Java 5.0 language, so you might not get for example proper syntax highlighting or code completion.

7.3.2 Rational Web Developer V6.0

Rational Web Developer is a subset of the functionality in Rational Application Developer. This set of functions focuses on Web development tooling. JavaServer™ Pages™, servlets, Struts, JavaServer Faces, static HTML, XML, and Web services development are all supported. However, if you require development of full J2EE applications, including EJB development, you need Rational Application Developer.

Note: See 7.3, “Development and deployment tools” on page 127 for important information about supported WebSphere Application Server versions.

Rational Web Developer V6.0 includes the following features, among others:

- ▶ Web services tools including wizards to generate Web services from Java beans and Web Services Description Language (WSDL) files and to consume Web services. It also includes a WSDL editor.
- ▶ Rapid Web development with page templates to give a consistent look and feel to Web sites.
- ▶ Apache Struts support with visual editors for most Struts artifacts.
- ▶ JavaServer Faces tools that support JavaServer Faces (JSF) 1.0. Includes additional IBM JSF components and s visual diagrams and editors to enable the clear layout of actions and JSF page navigation.
- ▶ Service Data Objects.
- ▶ Ant scripting and JUnit testing framework.
- ▶ Integration with CVS, a popular SCM repository.
- ▶ Integration with Rational ClearCase, the Rational SCM repository of choice.
- ▶ Unit testing using JUnit and Hyades frameworks. For further information, see the JUnit and Hyades sites at:
<http://www.junit.org>
<http://www.eclipse.org/hyades>
- ▶ IBM Rational Enterprise Generation Language (EGL) support, a high-level implementation-independent language that is used for developing business logic. For more information about EGL, see:
<http://www.ibm.com/software/awdtools/eglcobol/index.html>

7.3.3 Rational Application Developer V6.0

Rational Application Developer includes all the features of Rational Web Developer and adds more productivity-enhancing features, which makes it an even more appealing development environment for advanced J2EE development.

Note: See 7.3, “Development and deployment tools” on page 127 for important information about supported WebSphere Application Server versions.

In addition to the features in Rational Web Developer V6.0, Rational Application Developer V6.0 brings features such as:

- ▶ Full support for J2EE 1.4, including EJB 2.1 support.

- ▶ Portal application development with support for WebSphere Portal 5.0.2.2 test environment.
- ▶ EJB test client for easy testing of EJBs.
- ▶ Support for annotation-based development (part of WebSphere rapid deployment).
- ▶ UML modeling functionality.
- ▶ Integration with the Rational Unified Process and Rational tool set, which provides the end-to-end application development life cycle.
- ▶ Application analysis tools that check code for coding practices. Examples are even 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.
- ▶ Includes Rational ClearCase LT Server. ClearCase LT is the junior member of the Rational ClearCase family of Source Code Management (SCM) repositories.
- ▶ Crystal Reports for developing visual data reports.
- ▶ Component test automation tools to automate creating tests and building and managing test cases.

7.3.4 WebSphere rapid deployment

WebSphere rapid deployment is a set of tools and capabilities included in the WebSphere Application Server packaging and also used by the development tools. These features allow for the deployment of applications with minimum effort on the part of the developer or administrator. The rapid deployment model has three basic features:

- ▶ Annotation-based programming
- ▶ Deployment automation
- ▶ Enhanced EAR file

Annotation-based programming enables you to annotate the EJB, servlet, or Web service module code with special Javadoc™ syntax annotations. When the source of the module is saved, the directives in these annotations are parsed, and the rapid deployment tools use the directives to update deployment descriptors. Therefore, the developer can concentrate on the Java source code rather than metadata files.

Deployment automation is where applications that installation packages are dropped into a hot directory under an application server and the application is installed automatically. Any installation parameters that have not been specified

by the installation package's deployment descriptors have default values that are applied by the automated deployment process.

Rapid deployment allows for a free-form deployment operation. In this mode, it is possible to drop source code or compiled classes for servlets, EJBs, JSPs, images, and so on into the hot directory without strict J2EE packaging. Rapid deployment then compiles the classes, adds deployment descriptors, and generates an EAR file that is automatically deployed on a running server.

An enhanced EAR file enables the enterprise archive package to include information about resources and properties, such as data sources, that is required by an application. When deployed on a server, the resources are automatically created on the server.

The WebSphere rapid deployment set of tools can be useful for quickly testing an application. For example, if you know you are going to test several versions of the same application, you can use the automatic deployment feature to have the rapid deployment tools automatically deploy the versions for you. However, there are limitations and rules you need to obey when working with these utilities, and most of the time you are significantly more productive using a full-blown development environment, such as the Application Server Toolkit.

7.3.5 Which tool to use

Which tool you choose depends on your requirements. As of writing, if you need to develop and test applications on WebSphere Application Server V6.1 and you want an integrated WebSphere test environment for fast turnaround times, if you require a Java 5.0 capable development environment, or if you need to develop SIP applications, choose the Application Server Toolkit.

However, if you are developing applications that do not require the new features only available in WebSphere Application Server V6.1 (and thus the Application Server Toolkit V6.1), you can also use Rational Web Developer or Rational Application Developer. These tools are feature-rich and have lots of productivity-enhancing features not found in the Application Server Toolkit.

If you are using Rational Web Developer or Rational Application Developer today, your best option is probably to stay with your current tool and then upgrade when new versions become available.

7.4 Naming conventions

Spending some extra time on application-related naming concepts quickly pays off in practice, because it can reduce the time spent on analyzing the source of issues during standard operations of future J2EE applications.

7.4.1 Naming for applications

Generally, some 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 of identifying, creating, and re-creating 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 enterprise archive (EAR) file name, such as in `OrderApplication-1.4.0.1.ear`.

Sometimes, the version number of included components, such as utility JAR files packaged in the EAR, can also have version numbers in their file names, but this can often cause problems. Consider a utility JAR with a version number in the file name, such as `log4j-1.2.4.jar`. If that is updated and the name is changed to `log4j-1.2.5.jar`, each developer has to update the class path settings in their workspace, which will cost them time. It is then better to use an SCM system and label the new JAR file as being version 1.2.5, but keep the file name constant, such as just `log4j.jar`.

To keep track of all the versions of included components, it is a good idea to include a bill of materials file inside the EAR file itself. The file can be a simple text file in the root of the EAR file that includes versions of all included components, information about the tools used to build it, and the machine on which application was built. The bill of materials file can also include information about dependencies to other components or applications, as well as a list of fixes and modifications made to the release.

7.4.2 Naming for resources

When naming resources, preferably associate the resource to both the application using it and the physical resource to which it refers. As an example for our discussion, we use a data source, but the concept holds also for other types of resources such as messaging queue. Remember, if your company already has a naming convention for other environments (non-WebSphere) in place, it is probably a good idea to use the same naming convention in WebSphere.

Assume that you have a database called ORDER that holds orders placed by your customers. The obvious name of the data source would be Order and its JNDI name 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 would then be called Order_OrderApplication and its JNDI name jdbc/Order_OrderApplication.

Because the WebSphere administrative console sorts resources by name, you might want to include the name of the application first in the resource, such as in OrderApplication_Order. This gives you the possibility to sort your resources according to the application using them.

To group and sort resources in the WebSphere administrative console, you can also use the Category field, which is available for all resources in the administrative console. In this text field, you can enter, for example, a keyword and then sort your resource on the Category column. So 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.

7.5 Source code management

In development, it is important to manage generations of code. Carefully organize and track application builds and the source code 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 integrated development environment (IDE) such as the Application Server Toolkit or Rational Application Developer to do that. Code in an IDE is stored in a workspace on the file system, usually locally on each developer's machine. 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. This allows for:

- ▶ Development team collaboration (work on common code)
- ▶ Code versioning (managing which versions are in which releases)
- ▶ Wider team collaboration (access for project managers, testers)

SCM systems are used for these purposes.

Rational Web Developer and Rational Application Developer support Rational ClearCase and CVS as SCM systems, while the Application Server Toolkit supports only CVS.

7.5.1 Rational ClearCase

Rational ClearCase organizes its code repositories as versioned object bases or VOBs. VOBs contain versioned file and directory elements. Users of Rational ClearCase are organized according to their role. Each user has their own view of the data that is in the VOB on which they are working. Rational ClearCase tracks VOBs and 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 not just an SCM system but also a Software Asset Management (SAM) system. This means that it not only manages code but other assets. These further assets might be produced by the other Rational products with which Rational ClearCase integrates.

The Rational products with which ClearCase integrates are Rational Enterprise Suite Tools, the Rational Unified Process, and, of course, Rational IDEs. Artifacts such as use cases generated by Rational RequisitePro® can be stored in Rational ClearCase. These can then be fed into a Rational Rose design model and used to design Java components and generate Unified Modeling Language (UML) diagrams and documentation.

ClearCase can also be used to implement the Unified Change Management (UCM) process. This change management process can be enhanced by using Rational ClearCase in conjunction with Rational ClearQuest, a change and defect tracking software.

The software is scalable. Rational ClearCase LT is a cut down version of Rational ClearCase for small-to medium-sized teams. It can be upgraded seamlessly to Rational ClearCase as a user's needs change. Additionally, use a ClearCase MultiSite® add-on to support use of the software in geographically dispersed development teams.

In short, although ClearCase is an SCM system, it is also an integral part of the Rational toolset and RUP.

For more information about Rational software, see:

<http://www.ibm.com/software/rational>

7.5.2 Concurrent Versions System (CVS)

CVS uses a branch model to support multiple courses of work that are somewhat isolated from each other but still highly interdependent. Branches are where 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. This model enables individuals to 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:

- ▶ Free to use under the GNU license.
- ▶ Open source.
- ▶ Widely used in the development community.
- ▶ Other SCM repositories can be converted to CVS.
- ▶ Many free client applications are available, for example, WinCVS.
- ▶ Can store text and binary files.
- ▶ Handles versioning and branching.
- ▶ Is a centralized repository.

For more information about Concurrent Versions System, see:

<http://ximbiot.com/cvs/wiki>

7.5.3 Which SCM to use

The obvious question arises: Which SCM should the team use? There is no simple answer to this question, because the answer depends on a number of factors.

Current software and processes

To some extent, the choice depends on what the existing situation is (if any) and what the SCM and development process requirements are now and in the future. If a team uses CVS and an existing, successful, development process, ClearCase might not be necessary, especially if the size and complexity of requirements is not likely to grow in the future. If this is not the case, Rational ClearCase LT or Rational ClearCase are a good choice so that the full integration of Rational and WebSphere products can be exploited now and in the future.

Team size

Rational ClearCase LT gives a sound starting place for smaller teams. Rational ClearCase LT can be upgraded to Rational ClearCase later if necessary. On very large development projects, Rational ClearCase and Rational ClearQuest have a MultiSite option that allows for easier development by geographically dispersed development teams.

Complexity of requirements

RUP provides a holistic approach to the end-to-end development life cycle. The use of the UCM process, which is part of the RUP, can shield the user from complex tagging and branching of code. CVS does not shield the user from this.

Cost

CVS is a possibly a cheaper option because it is free and has a large user base, which means cheaper skills. In terms of hardware, it is likely that hardware costs for hosting CVS itself are cheaper because of its smaller footprint. However, these might be false economies. The limitations of CVS can cause a team to migrate to Rational ClearCase later.

Change management process

If the development team uses CVS rather than Rational ClearCase, the team does not get a prescribed change management process for CVS such as the UCM. If their organization does not have its own change management process, such a process should be created and put into place.

Summary

In summary, the smaller the development team and the less complex the requirements, the more likely that CVS or Rational ClearCase LT are good choices. As team size and complexity grows, Rational ClearCase and then Rational ClearCase MultiSite become more attractive. Existing processes and software as well as the budget for new software, hardware, and training are likely to inform the decision further. In matters of cost, there might be false economies.

7.6 Automated build process

The major driver for implementing and maintaining an automated build process is to provide a simple and convenient method for developers to perform builds for development, test, and production environments.

The main problems you might run into when you do not have an automated process are:

- ▶ Things fail 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 even customers have to wait to get the code out to a test, staging, or production environment because the only person who has control over these is unavailable.
- ▶ You cannot reproduce a problem on production because you do not know what version of files are in production at the moment.

The time spent developing an automated build script will pay for itself over time. After you have an automatic build process in place, you can virtually eliminate failures due to improper deployment and packaging, considerably reduce the turnaround time for a build, allow you to easily recreate what is in each of your environments, and ensure that the code base is under configuration management.

There are several tools on the market to help you develop a build script, including Apache Ant. Apache Ant is a Java-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 very popular tool in the Java world.

WebSphere Application Server provides a copy of the Ant tool and a set of Ant tasks that extend its capabilities to include product-specific functions. These Apache Ant tasks reside 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.

The tasks included with WebSphere Application Server enable you to:

- ▶ Install and uninstall applications.
- ▶ Run EJB deployment and JSP pre-compilation tools.
- ▶ Start and stop servers in a base configuration.
- ▶ Run administrative scripts or commands.

By combining these tasks with those provided by Ant, you can create build scripts that pull the code from the SCM repository, and then compile, package, and deploy the enterprise application on WebSphere Application Server. To run Ant and have it automatically see the WebSphere classes, use the `ws_ant` command.

For more detailed information about Ant, refer to the Apache organization Web site at:

<http://ant.apache.org/index.html>

7.7 Automated functional tests

Automating your functional tests might be a good idea depending on your project size and how complex the requirements of the project are. Scripts execute much faster than people, but they are not automatically generated, so someone has to create them at least one time. It is possible to create a script to cover all function in your application but it would be very complicated and costly. A good idea is to create scripts for the main features of the system and those that will not change that much over the time, so every time a new build is published by an automated build tool or a human, you can be sure that the application still works properly.

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/>

7.8 Test environments

Before moving an application into production, it is very important to test it thoroughly. Because there are many kinds of tests that need to be run by different teams, a proper test environment often consists of multiple test environments.

Tests cases must be developed according to system specification and use cases. Do this before the application is developed. System specification and use cases need to 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 non-functional requirements (such as performance or capacity requirements). After developing the test cases and enough functionality has been developed in the application, start testing.

Figure 7-2 shows an overview of a recommended test environment setup.

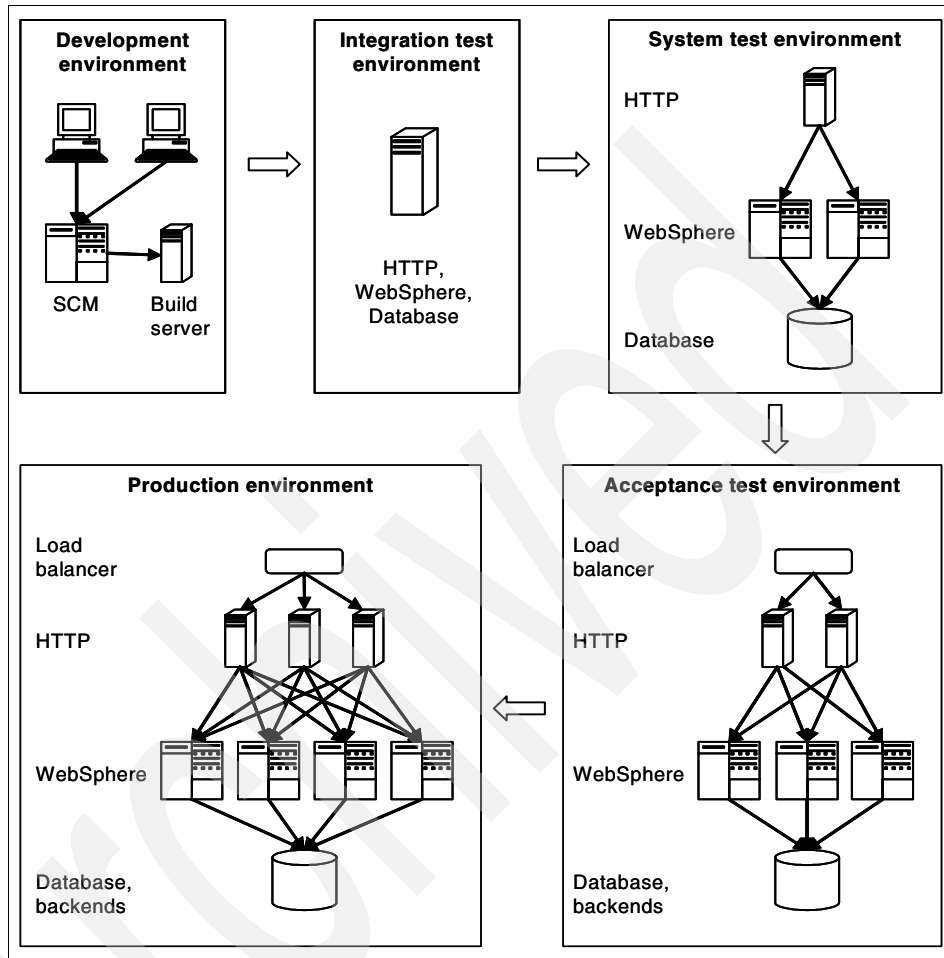


Figure 7-2 Test environments

Whether you choose to use some of these test environments, all of them, or even additional test environments depends on the system being developed, project size, budget constraints, and so on.

Each environment is maintained as a separate cell in order to completely isolate the environments from each other. For smaller environments, a single application server profile is usually sufficient, while larger ones might need a deployment manager for that particular cell environment.

Development environment

Usually each developer has their own WebSphere test environment integrated in the development tool. This test environment is used for the developer's daily work and it 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 using the resources in the developer's workspace. This eliminates the need for developers to execute build scripts, export, or otherwise package the application into an EAR file and deploy that on a test server for every small change made. This capability makes it very easy and quick to test applications while developing them and increases developer productivity.

Each developer is also responsible for performing unit testing of their own code. The majority of all tests performed for the system are executed in this environment, and the primary goal is to wash out obvious code bugs. The developers work against and share code using the SCM system. The development environment is most often a powerful Windows desktop machine.

When each developer has committed their code on 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 is usually done on a special build server and is controlled by automatic build scripts (see 7.6, "Automated build process" on page 137). This server might need to have a copy of the Application Server Toolkit or Rational Web Developer installed.

The development team should also create a Build Verification Test process (see 7.7, "Automated functional tests" on page 139), one where each new build is executed before making this 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.

Another activity that is every developer's responsibility is to perform basic code profiling. By using the profiling tools in Rational Application Developer, a developer can discover methods that perform poorly, find memory leaks, or excessive creation of objects.

Integration test environment

After a successful build and regression test, the application is deployed to the integration test environment. This is the environment where the developers perform integration tests among all system components on a hardware and software platform that mirrors the production environment, although in a very small size.

Because the production environment is often not the same platform as the development environment, a guideline is to start testing on the target platform as early as possible in the test phase. This testing will help discover problems with incompatibilities between platforms, for example, hard coded folder paths (such as C:\ versus /usr). The integration test environment is usually the first environment suitable for that.

For small projects, the integration test environment can often be shared between different projects. But if the number of projects or developers is too large, it becomes difficult to manage. Usually no more than 5 to 10 developers should share a single integration test environment. If a developer needs to perform tests that might damage the environment, a dedicated environment should be used.

As long as the machine has enough resources in terms of CPU and memory, using multiple WebSphere profiles can also be a good method to isolate different teams from each other. Using VMWare is another option.

The development team manages and controls the integration test environment.

System test environment

The purpose of the system test is to verify that the system meets both functional and non-function requirements. After the development team has tested the application in their own controlled environment, it is delivered to the system test team. When the application is delivered, the system test team deploys it 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. Functional requirements include things such as does the system execute the business rules defined, does the user interface show the right information, and so on. Non-functional requirements include capacity, performance, installation, backup, and failover requirements.

The system test team completely controls the system test environment. The environment is usually a cut-down version of the real production environment, but with all the important components in place. If the production environment is a highly available environment with WebSphere clusters, the system test should also be set up with clusters to verify both application functionality and deployment routines.

The system test environment can also be used by other teams. Perhaps the system administrators need to test new patch levels for the operating system, WebSphere, database, and so on before rolling them out in production. The system test environment is a good place to do that. If a patch is committed, it

should also be applied to the other test environments to keep all environments in sync.

Acceptance test environment

The acceptance test environment is the last stage where testing takes place before moving the application into production. The acceptance test environment is the one that most closely resembles the actual production environment. Hardware and software must be identical to the production environment.

Because of cost constraints, it is often not possible to have an acceptance test environment with identical capacity as the production environment. The acceptance test environment is, therefore, usually smaller than the production environment, but needs to contain all the same components, same brands, same software patch levels, and 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 (such as installation, backup, failover, and so on). It also provides an opportunity to test unrelated applications together. The previous environments all focused on testing the applications independently of each other.

Often the acceptance test environment is where performance tests are run, because the acceptance test environment is the one most similar to the real production environment.

When doing performance tests, it is extremely important to have a representative configuration as well as representative test data. It is not unusual that projects perform successful performance tests where the results meet the given requirements, and then when the application is moved into production, the performance is bad. Often this can be because the production database is much larger than the databases used in the acceptance test environment. Therefore, it is very important that the test databases have been populated with representative data. Ultimately, a copy of the production database should be used, but sometimes this is not possible because tests might involve placing orders or sending confirmation e-mails. Other causes for differences in performance between the successful performance tests and the production environment is, for example, that the performance tests ran without HTTP session persistence, while the production environment uses session persistence. To get realistic results, the performance test environment and setup must be realistic, too.

7.9 Managing application configuration settings

Almost all non-trivial applications require at least some amount of configuration to their environment in order to run optimally. Part of this configuration (such as references to EJBs, data sources, and so on) is stored in the application deployment descriptors and is modified by developers using tools such as the Application Server Toolkit or Rational Web Developer. Other settings, such as the JVM maximum heap size and database connection pool size, are stored in the WebSphere configuration repository and modified using the WebSphere administrative tools. Finally, there are settings that are application-internal, usually created by the developers and stored in Java property files. These files are then modified, usually using a plain text editor, by the system administrators as necessary after deploying the application.

7.9.1 Classifying configuration settings

Configuration data can often be categorized into three different categories.

Application-specific

This category includes configuration options that are specific for an application regardless of its deployment environment. Examples include how many 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 both 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 along with the application between environments, but need to be tuned differently for the application in each environment.

Environment-specific

This category includes configuration options that are specific to a deployment environment but 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

might be C:\temp, but in the UNIX production environment, it might be /tmp. This category of options must not move between environments.

7.9.2 Managing configuration settings

Dealing with configuration settings is usually a major challenge for both developers and system administrators. Not only may configuration settings have to be changed when the application is moved from one deployment environment to another, but the settings must also be kept in sync among all application instances if running in a clustered environment.

To manage the settings stored in the WebSphere configuration repository (such as the JVM maximum heap size), 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 as the application is moved between different environments. Instead, the J2EE specification does a good job here and separates the developers' work from 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).

Application-internal configuration settings, however, 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 using the `java.util.Properties` class since Java 1.0. Although you can use databases, LDAP, JNDI, and so on to store settings, plain Java property files are still by far the most common way of configuring internal settings for Java applications, mainly because it is an easy and straightforward method to accomplish the task. However, in a clustered environment where the same application runs on multiple servers distributed across different machines, care must be taken as to how to package, distribute, and access the property files.

For packaging the property files, you have two approaches. Either you include the property files within the EAR file itself or you distribute them separately. To include them within an EAR file, the easiest approach is to create a utility JAR project, 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. Best practice, however, is to centralize access to the property files using a custom property manager class, so 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 following 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 described earlier).

However, if you want to make the property files easily accessible after the application has been deployed, you might want to store them in a folder outside the EAR file. To load the property files, you then either make the folder available on the class path for the application (and use the previous code snippet) or you use an absolute path name and the following code snippet instead (assuming the file to load is `/opt/apps/OrderApp/my.properties`):

```
Properties props = new Properties();
InputStream in = new
FileInputStream("/opt/apps/OrderApp/my.properties");
props.load(in);
in.close();
```

Using absolute path names is usually a bad idea because it tends to hard code strings into your code, which is not what you want to do.

A better approach is to make the folder with the property files available on the class path for the application. You can do this 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 `/opt/apps/OrderApp`, in the Classpath field for the shared library.

Another less well-known but even better approach to access property files is to use URL resources. We do not go into the details of exactly how to do that here, but the following steps describe the approach:

1. Create a folder on your system that holds the property file.
2. Use the WebSphere administrative console and 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 pointing to the JNDI name chosen.

4. In Java, use JNDI to look up the URL resource reference. Create an `InputStream` from the URL, and then 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 J2EE compliant because it does not rely on the `java.io` package for file access, which is prohibited according to the J2EE specification.

The method also gives you additional opportunities that comes with the `URL` class, and that is to load the property files using HTTP and FTP. So if you want, you can set up an HTTP server serving properties files from a central location.

To learn more about this method, see the IBM developerWorks® article *Using URL resources to manage J2EE property files in IBM WebSphere Application Server V5*, available at:

http://www.ibm.com/developerworks/websphere/library/techarticles/0502_botzum/0502_botzum.html

Unless 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 stored in folders outside the EAR files are not propagated to the WebSphere nodes 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. But because this folder is not known to the WebSphere infrastructure, it will not happen automatically when the contents are changed. Instead, 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, and you can include that folder on the class path using a shared library or point your URL resources to it.

Tip: When deciding on names for settings in property files, it is a good idea to include the unit the setting refers to in the name. Therefore, instead of using `MaxMemory` or `Timeout`, it is better to use `MaxMemoryMB` and `TimeoutMS` to indicate that the max memory should be given as megabytes and the timeout as milliseconds. This can help reduce confusion for the system administrator who does not know the internals of the application.

If you store property files that need to be changed between different environments inside the EAR file, you might discover that there are problems involved with that approach, especially in a clustered environment.

In a clustered environment when an enterprise application is deployed to WebSphere, it is distributed to each node in the cluster using the WebSphere file transfer mechanism. At each node, the EAR file is expanded and laid out on the file system so that WebSphere can execute it. This means that a property file included in the EAR file is automatically replicated to each member of the cluster.

If you then need to make a change to the property file, you either have to do that manually on each cluster member, which can be error prone, or you do it on the deployment manager itself and then distribute the updated file to each node again. However, WebSphere 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 needed to configure the application in the WebSphere cell repository), so the property file is not readily accessible on the deployment manager. Because of this, you must manually unpack the EAR file, extract the property file, modify it, and then re-create the EAR file again and redeploy the application. This is not a recommended approach.

Another option is to distribute the property files within the EAR, but after deployment, extract them from the EAR file and place them in a folder separate from the EAR. An example of a folder name suitable for that is `<dmgr_profile_home>\config\cells\<cell_name>\configData` on the deployment manager machine. Anything in that folder is replicated to each node in the cell when WebSphere synchronizes with the nodes. For the application to find the file, it must then refer to it on its local file system. But because that folder name then includes both the name of the profile and the name of the cell, it can quickly become messy and is usually not a good solution.

7.10 Planning for application upgrades in production

To have a production environment that enables you to roll out new versions of applications while maintaining continuous availability is not only the responsibility of the WebSphere infrastructure architects and system administrators. 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, there might not be much the system administrators can do but to bring down the whole system for an application upgrade. And unfortunately, application developers are too often not aware of the impact their decisions have on the production environment.

Developers need to consider at least the following areas when planning for new versions:

- ▶ Database schema compatibility

If a change in the database layout is introduced, it might be necessary to shut down all instances of an application (or even multiple applications if they use the same database) in order to migrate the database to the new layout and update the application.

One possibility is to migrate a copy of the database to the new layout and 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 will need to be re-applied to the copy, which is now the hot database.

- ▶ EJB version compatibility

If EJB interfaces do not maintain backward compatibility and the application has stand-alone Java clients, it might be necessary to distribute new versions of the Java clients to users' desktops. Or if the EJB clients are servlets but they are not deployed as part of the same EAR file as the EJBs or they are running in a container separate from the EJB, it might be necessary to set up special EJB bindings so that the version 1 clients can continue to use the version 1 EJBs while version 2 clients use new version 2 EJBs.

- ▶ Compatibility of objects in HTTP session

If you take a simple, straightforward approach and use the WebSphere rollout update feature and you have enabled HTTP session persistence, you must make sure that objects stored in the HTTP session are compatible between the two application releases. If a user is logged on and has a session on one application server and that server is shut down for its application to be upgraded, the user will be moved to another server in the cluster and his session will be restored from the in-memory replica or from a database. When the first server is upgraded, the second server will be shut down and the user will then be moved back to the first server again. Now, 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. And, it might even require training for users to learn a new user interface or navigation system.

We do not go into depth on this subject but instead point you to an excellent article about the subject on developerWorks, *Maintain continuous availability while updating WebSphere Application Server enterprise applications*. It describes what to consider both from a developer point of view and a

WebSphere infrastructure and system administration point of view in order to have a highly available environment that can handle a rollout of new application versions.

http://www.ibm.com/developerworks/websphere/techjournal/0412_vansickel/0412_vansickel.html

7.11 Mapping applications to application servers

A question that often arises when deploying multiple WebSphere applications is whether the applications need to be deployed to the same application server instance (JVM) or if you need to create a separate instance for each application. There is no easy answer to this question because it depends on several factors. Normally, you deploy multiple applications to one application server, mainly because it uses a lot less resources. However, it is not always possible to do so.

This section attempts to give you some points to think about so that you can make the right decision for your particular environment.

The advantages of deploying each application to its own application server are:

- ▶ If an application server process crashes, it will bring down only the application running in that server. However, if using a cluster, you would still have other instances running.
- ▶ Many WebSphere settings, such as JVM heap size and EJB transaction timeout, are configured at the application server level. If two applications require different settings, they cannot be deployed to the same application server. Note, however, that multiple applications can be deployed to the same application server and use the same heap, as long as it is large enough to accommodate all applications.
- ▶ Java environment variables (specified using `-D` on the Java command line, or using JVM custom properties) are specified per JVM instance. This means that if you need to specify, for example, `-Dlog4j.configuration` with different settings for each application, you cannot do that if they are all in the same JVM.
- ▶ You can use the WebSphere rollout update feature, which requires the application server to be stopped. Although it can be used even when there are multiple applications in each application server, all applications are stopped as the application server is stopped. You need to have all applications clustered to have other instances always available.

- ▶ Each application will receive its own SystemOut and SystemErr log file. If multiple applications are deployed on the same application server, system log output from all applications would be interleaved in the SystemOut and SystemErr logs. Usually, however, this is not a problem because applications often use, for example, log4j (which is very configurable) to perform logging instead of plain system out print statements.

The advantages of deploying multiple applications to the same application server are:

- ▶ By deploying multiple applications to the same application server, you can reduce the memory used. Each application server instance requires about 130 MB of RAM for WebSphere to run. If you have 10 such application server processes running on your system, you have consumed more than 1 GB worth of RAM for the WebSphere runtime alone.
- ▶ You can use EJB local interfaces to make local calls from one application to another, because they are in the same JVM.
- ▶ Fewer application servers means fewer ports open in the WebSphere Application Server tier, which means fewer ports need to be opened in the firewall between the HTTP tier and the WebSphere Application Server tier.

Sometimes, a mixed approach is the best way to go. By grouping related or similar applications and deploying them to the same application server, while deploying other applications that need to run in their own environment, you can achieve a good compromise.

7.12 Planning checklist for applications

Table 7-1 lists a summary of items to consider as you plan and additional resources that can help you.

Table 7-1 Planning checklist for applications

Planning item
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.
Design an end-to-end test environment.
Create a strategy for maintaining and distributing application configuration data.
Create a strategy for application maintenance.

Planning item
Determine where applications will be deployed. (All on one server?)

Resources

For a good overall reference for packaging and deploying J2EE applications in WebSphere Application Server, refer to *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304. We suggest that you have a copy of this book available as you plan your Web services environment.

For detailed information about application development using Rational Application Developer, refer to *Rational Application Developer V6 Programming Guide*, SG24-6449.

The WebSphere Application Server Information Center also contains a lot of useful information. For a good entry point to information about application development and deployment, go to:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6topdeveloping.html>

Planning for system management

This chapter provides an overview of the planning necessary for the system management of the WebSphere Application Server runtime environment. It focuses on developing a strategy to best 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 contains the following sections:

- ▶ What is new in V6.1
- ▶ Administrative security
- ▶ WebSphere administration facilities
- ▶ Configuration planning
- ▶ Change management topics
- ▶ Problem management
- ▶ Planning checklist for system management

8.1 What is new in V6.1

The following list highlights the changes in administrative tools and processes since V6.0:

- ▶ WebSphere Application Server V6.1 introduces thin administrative client libraries that enable you to create and run administrative clients. These clients rely on a JAR file, rather than requiring a full installation of WebSphere Application Server or a Websphere Application Server client installation.
- ▶ Commands and arguments for the `wsadmin` tool are not case-sensitive anymore.
- ▶ The `wsadmin` language used (either Jacl or Jython) is inferred by the suffix on the script file.
- ▶ Command assistance in the administrative console maps your administrative activities to `wsadmin` scripting commands. Using it, you can view the `wsadmin` script command in Jython for your last action run in the administrative console.

8.2 Administrative security

We recommend enabling administrative security to prevent unauthorized access to the administrative tasks. Enabling administrative security only secures administration tasks, not applications.

Proper planning for system management includes identifying the people who will need access to the administrative tools and designing a system of groups, users, and roles that fit your needs.

WebSphere Application Server V6.1 gives you the option to enable administrative security during profile creation. If you choose this option during profile creation, you will be asked to provide a user ID and password that will be stored in a file-based user repository and be mapped to the Administrator role. Additional users can be added after profile creation using the administrative tools.

Administration users are assigned to roles that determine their level of authority (see “Secure administration tasks” on page 107).

8.3 WebSphere administration facilities

WebSphere Application Server provides a variety of administrative tools for configuring and managing your runtime environment, including:

- ▶ Administrative console
Use the administrative console to perform the deployment and system administration tasks through a Web interface.
- ▶ WebSphere scripting client (wsadmin)
The `wsadmin` tool is intended for production environments and unattended operations.
- ▶ Task automation with Ant
With Ant, create build scripts that compile, package, install, and test your application on the application server.
- ▶ Administrative programs
Develop Java classes that perform administrative functions on WebSphere Application Server.
- ▶ Command line tools
Using the command line tools, start and stop application servers, check server status, add or remove nodes, and complete similar tasks.

The choice of which combination of administrative tools you will employ ultimately depends on the size and complexity of your runtime environment. Where you have few resources, but many tasks, we recommend the use of automation and scripts. Where you have multiple administrators that will perform different tasks, you might want to carefully consider defining different access control roles. This is especially important where you want non-administrators to be able to perform limited roles such as application deployment.

8.3.1 Administrative console

The WebSphere Application Server administrative console connects to a running stand-alone server or, in a distributed environment, to a deployment manager.

Non-secure administration access

If administrative security is not enabled, the console is accessed through a Web browser through the following URL:

```
http://<hostname>:<WC_adminhost>/ibm/console
```

For example, `http://localhost:9060/ibm/console`.

You can gain access to the console without entering a user name. If you do enter a name, it is not validated and 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 console is accessed through a Web browser through the following URL (note the use of https:// versus http://):

```
https://<hostname>:<WC_adminhost>/ibm/console
```

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.

8.3.2 WebSphere scripting client (wsadmin)

The WebSphere administrative (**wsadmin**) scripting program is a powerful, text console-based command interpreter environment that enables you to run scripted administrative operations. The **wsadmin** administrative scripting program supports two scripting languages, Jacl and Jython.

Note: WebSphere Application Server V6.1 represents the start of the deprecation process for the Jacl syntax associated with the **wsadmin** tool. The Jacl syntax for the **wsadmin** tool continues to remain in the product and is supported for at least two major product releases. After that time, the Jacl language support might be removed from the **wsadmin** tool. For this reason, you should use Jython when creating new scripts.

A conversion tool called **jacl2jython** is supplied with WebSphere Application Server V6.1 to help you convert existing scripts.

You can run the **wsadmin** tool in interactive and unattended mode. Use the **wsadmin** tool to perform the same tasks that you can perform using the administrative console.

WebSphere Application Server V6.1 adds command assistance in the administrative console that maps your administrative activities to **wsadmin** scripting commands written in Jython. These commands can be viewed from the console, and if you want, you can log the command assistance data to a file. You can also allow command assistance to emit Java Management Extensions (JMX™) notifications to the Application Server Toolkit. The Application Server Toolkit has Jython development tools that help you develop and test Jython scripts.

8.3.3 Task automation with Ant

WebSphere Application Server 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 very popular tool among Java programmers.

Apache Ant is a platform-independent, Java-based build automation tool, 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 a number of tasks that are specific to managing and building applications in WebSphere Application Server.

The Ant environment enables you to create platform-independent scripts that compile, package, install, and test your application on the application server. Integrate with `wsadmin` scripts and use Ant as their invocation mechanism.

8.3.4 Administrative programs

WebSphere Application Server supports access to the administrative functions through a set of Java classes and methods. You can write a Java program 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.

You can prepare, install, uninstall, edit, and update applications through programming. Preparing an application for installation involves collecting various types of WebSphere Application Server-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, or 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.

8.3.5 Command line tools

Command line tools enable you to 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 directory `<was_home>/WebSphere/AppServer/bin`, the command will run within the default profile.

8.4 Configuration planning

This section describes global configuration planning topics. Configuring and managing the WebSphere 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 server startup behavior
- ▶ Custom application server configuration templates
- ▶ Planning for resource scope use

8.4.1 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 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 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. This can have an effect on the performance and availability of the configuration file. The location is chosen during profile creation. If you run WebSphere Application Server for z/OS, we recommend that you use a separate HFS for each node.

In addition, consider whether to use automatic synchronization to push out changes to the nodes or to synchronize changes manually. In an environment where there are a lot of administration changes going on, automatic synchronization might have a performance impact on the network. We discuss this more in 8.5.2, “Changes in topology” on page 164.

8.4.2 Configuring application and server startup behavior

One feature of WebSphere Application Server is the ability to manage the startup of applications and servers.

By default, applications start when their server starts. The following settings enable you to fine-tune the startup speed and order of applications that are configured to start automatically when the server starts. Access these settings in the administrative console by navigating to **Applications** → **Enterprise applications** → **<your_application>** → **Startup behavior**.

- ▶ The *Startup order* setting for an application lets you specify the order in which to start applications when the server starts. The application with the lowest startup order starts first. Applications with the same startup order start in parallel. This can be very important for applications that have been split into sub-applications that need to start in a certain order due to dependencies between them.
- ▶ The *Launch application before server completes startup* setting lets you specify whether an application must initialize fully before its server is considered started. Background applications can be initialized on an independent thread, thus allowing the server startup to complete without waiting for the application.
- ▶ The *Create MBeans for resources* setting specifies whether to create MBeans for resources such as servlets or JavaServer Pages (JSP) files within an application when the application starts.

The following setting can affect how an application server starts. Access this setting by navigating to **Servers** → **Application servers** → **<your_server>**.

- ▶ The *Parallel start* setting for an 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 can shorten the startup time for a 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 startup and each has its own configuration and application data to start.

You can prevent an application from starting automatically at application server startup, enabling you to start it later manually. To prevent an application from starting when a server starts, navigate to **Applications** → **Enterprise Applications** → **<application_name>** → **Target specific application status** and disable auto start for the application.

8.4.3 Custom application server configuration templates

WebSphere Application Server provides the ability to create a customized server template that is based on an existing server configuration. Server templates can

then be used to create new servers. This provides a powerful mechanism to propagate the server configuration both within the same cell and across cell boundaries. In order to propagate the server configuration across cell a boundary template, the server configuration must be exported to a configuration archive, after which it can be imported to another cell.

If you are going to need more than one application server (say, for a cluster), and the characteristics of the server are different from the default server template, it is much more efficient to create a custom template and then use that template to create your application servers. When creating a cluster, be sure to use this template when you add the first member to the cluster. Subsequent servers in the cluster will also be created using this template. This will reduce the scope for error and make the task of creating the server cluster much faster.

8.4.4 Planning for resource scope use

Resource scope is a very 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, it makes sense to define that data source once at the node level, rather than create the data source multiple times, possibly introducing errors along the way. Also, if the data source definition needs to change (maybe due to changes to an underlying database), the data source definition can be changed once and is visible to all servers within the node. The savings in time and cost should be self-evident.

Some thought needs to be put toward outlining what resources you will need for all the applications to be deployed and at what scope to define each. You select the scope of a resource when you create it.

The following list describes the scope levels, listed 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. We recommend that cell scope resource definitions should be further granularized 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 occur 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 and 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 Java Naming and Directory Interface™ (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. All cluster members must be running at least at V6 to use cluster scope for the cluster. The resource factories that are defined within the cluster scope are available for all the members of this cluster to use and override any resource factories that have the same JNDI name that is defined within the cell scope. The resource factories that are defined within the cell scope are available for this cluster to use, in addition to the resource factories that are defined within this cluster scope.

▶ Node scope (default)

The node scope value limits the visibility to all the servers on the named node. This is the default scope for most resource types. The resource factories that are 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 that are defined within the cell scope are available for servers on this node to use, in addition to the resource factories that are defined within this node scope.

▶ Server scope

The server scope value limits the visibility to the named server. This is the most specific scope for defining resources. The resource factories that are defined within the server scope are available for applications that are deployed on this server and override any resource factories that have the same JNDI name defined within the node and cell scopes. The resource factories that are defined within the node and cell scopes are available for this server to use, in addition to the resource factories that are 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 the Application Server Toolkit or the `wsadmin` tool to view or modify the application scope resource configuration. The resource factories that are 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 lower scope. For example, a data source created at a node level might be visible to servers within the node.

Note: A common source of confusion is the use of variables at one scope and the resources that use them at a different in scope. Assuming that the proper definitions are available at a scope the server can see, they do not have to be the same scope during runtime.

However, consider the case of testing a data source. A data source is associated with a JDBC provider. JDBC providers are commonly defined 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 necessarily have to be the same to be successful during runtime. However, when using the test connection service to test a data source 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 test connection service topic in the Information Center at:

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

8.5 Change management topics

Proper change management is important to the longevity of any application environment. WebSphere Application Server contains a number of technologies to aid with the change management process.

This section highlights some topics to think about when planning for changes to the WebSphere Application Server V6 operational environment. Topics include:

- ▶ Application updates
- ▶ Changes in topology

8.5.1 Application updates

WebSphere Application Server V6 permits fine-grained updates to applications. It allows application components to be supplied and the restart of only parts of the application. This preserves application configuration during the update process.

There are several options to update the applications' files deployed on a server or cluster:

- ▶ Administrative console update wizard
Use this option to update enterprise applications, modules, or files already installed on a server. The update can be whole EAR files, single/multiple modules (such as WAR or JAR files), or single/multiple file updates.
- ▶ *wsadmin* scripts
Use the **wsadmin** script to perform the same updates as the console wizard.
- ▶ WebSphere rapid deployment
WebSphere rapid deployment allows placement of application artifacts (EAR, WAR, JAR, or smaller files) into a monitored directory, from where they are detected and automatically deployed to the server by whatever steps are deemed necessary.
- ▶ Hot deployment and dynamic reloading
Hot deployment and dynamic reloading requires that you directly manipulate the application or module file on the server where the application is deployed; that is, the new files are copied directly into the EAR directory on the relevant server or servers.

When an application is deployed in a cluster, there is the option to perform an automatic application rollout. This is a mechanism where each member in the cluster is brought down and is updated with the application changes one at a time. When a given server has been updated, the next server is updated. Where clusters span multiple nodes, only one node at a time is updated. This allows the cluster to operate uninterrupted as work is diverted from the node being updated to the other nodes, until the entire cluster has received the update. If there is only a single node involved, it is brought down and updated.

In WebSphere Application Server for z/OS, you can use the MVS 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.

WebSphere Application Server supports J2EE 1.4 enterprise applications and modules. You can deploy J2EE 1.4 modules to WebSphere Application Server

V6.0 or V6.1 servers or to clusters that contain such cluster members only. WebSphere Application Server V6.1 ships with a JVM that supports JDK 5. If you compile applications using JDK 5 and they are not backward compatible, you must deploy your application to a V6.1 application server. The Application Server Toolkit shipped with V6.1 provides the capability of compiling applications with JDK 5. The default is to compile in backward compatibility mode.

Refer to Chapter 7, “Planning for application development and deployment” on page 123 for further details about what application deployment options are available in WebSphere Application Server V6.1.

8.5.2 Changes in topology

In a distributed server environment, the deployment manager node contains the master configuration files. Each node has its required configuration files available locally. Configuration updates should 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**
Turn on automatic synchronization using administrative clients. The default file synchronization interval is 60 seconds and starts when the application server starts.
- ▶ **Manual**
Perform manual synchronization using the administrative console, the **wsadmin** tool, or using the **syncNode** command located in the `<install_root>/bin` directory.

The file synchronization process should coincide with the whole change management process. In general, we recommend that you define the file synchronization strategy as part of the change management process.

8.6 Problem management

A proactive approach to problem management is always the best. This section outlines general practices to follow. For more information, including an approach to problem determination and detailed information about resolving specific types of problems, refer to *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798.

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®, Windows Event Viewer, and so forth
 - WebSphere Application Server SystemOut and SystemErr logs
 - SYSPRINT of the 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 have been installed; a quick check for a fix can save hours of debugging.
- ▶ Become familiar with the problem determination tools available in WebSphere Application Server and what they provide.

8.6.1 Logs and traces

Log files and traces need to be properly named. We recommend that you name log files according to the application that they belong to and group them in different directories. Clean log files periodically (saved to a media and then

deleted). WebSphere Application Server can write system messages to several general purpose logs. These include:

▶ JVM logs

The JVM logs are written as plain text files. They are named SystemOut.log and SystemErr.log and are in the following location:

`<profile_home>/logs/<server_name>`

You can view the JVM logs from the administrative console (including logs for remote systems) or by using a text editor on the machine where the logs 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 the 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 as:

- `<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 written in a binary format. You cannot view the log directly using a text editor. You should never directly edit the service log, because doing so will corrupt the log.

You can view the service log in two ways:

- We recommend that you use the Log Analyzer tool to view the service log. This tool provides interactive viewing and analysis capability that is helpful in identifying problems.
- If you are unable to use the Log Analyzer tool, you can 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 dump to the command shell window.

The IBM service log is in the following directory:

`<profile_home>/logs/`

8.6.2 Fix management

You should have a fix management policy in place, and you should test fixes before you apply them to the production system. For available fixes, see the WebSphere Application Server support page:

<http://www.ibm.com/software/webservers/appserv/was/support/>

8.6.3 Backing up and restoring the configuration

Back up the WebSphere Application Server configuration to a zipped file by using the **backupConfig** command.

For a stand-alone node, run the **backupConfig** utility at the node level. For a network deployment cell, run the **backupConfig** utility at the deployment manager level, because it contains the master repository. Do not perform **backupConfig** at the node level of a cell.

The **restoreConfig** command restores the configuration of your stand-alone node or cell from the zipped file that was created using the **backupConfig** command.

8.7 Planning checklist for system management

Table 8-1 lists a summary of items to consider as you plan and additional resources that can help you.

Table 8-1 Planning checklist for system management

Planning item
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 admin security until after installation.
Review the administration facilities available (scripting, administrative console, and so on) and create an overall strategy for configuration and management of WebSphere resources.
Determine where the profile directories (including the configuration repositories) will be located.
Consider whether to use automatic or manual synchronization to nodes.

Planning item
Plan for application server startup: <ul style="list-style-type: none"> ▶ Starting order ▶ Allow applications to start before server completes startup ▶ Create MBeans for resources ▶ 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. This includes maintaining and updating applications (see also 7.9, “Managing application configuration settings” on page 144 and 7.10, “Planning for application upgrades in production” on page 148). It also includes strategies for changes in cell topology and updates to WebSphere code.
Create a strategy for problem management. Identify a location and naming convention for storing WebSphere logs. Configure the processes to use those locations.
Create a strategy for backup and recovery of the installation and configuration files.


Resources

For a good overall reference for developing and deploying Web services in WebSphere Application Server, refer to *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

We suggest that you have a copy of this book available as you plan your Web services environment.

The WebSphere Application Server Information Center also contains a lot of useful information. For a good entry point to system management topics, see:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6topmanaging.html>



Planning for performance, scalability, and high availability

This chapter discusses the items to consider when implementing WebSphere Application Server V6.1 so that the environment performs well and is highly scalable. It contains the following sections:

- ▶ What is new in V6.1
- ▶ Scalability
- ▶ Workload management
- ▶ High availability
- ▶ Caching
- ▶ WebSphere Application Server performance tools
- ▶ Session management
- ▶ Planning checklist for performance

For detailed information as you plan for performance and high availability, we suggest you start with the following publications:

- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- ▶ *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688

9.1 What is new in V6.1

WebSphere Application Server V6.1 has a new JVM for all platforms that ship with an IBM JDK (versus Sun HotSpot-based JVMs, such as the Sun HotSpot JVM on Solaris and the HP JVM for HP-UX). This new JVM includes a new garbage collection scheme and a new Just-In-Time (JIT) compiler. The IBM Java 5.0 JVM provides major improvements in virtual machine technology to provide significant performance and serviceability enhancements over the earlier IBM Java execution technology.

9.2 Scalability

Scalability, in general terms, means adding hardware to improve performance. However, adding more hardware might not necessarily improve the performance if the software is not tuned correctly. Before investing in additional resources, you should understand the workload characteristics of your systems and ensure that your systems are properly tuned for this workload.

After optimizing the software, consider additional hardware resources as the next step for improving performance. There are two ways to improve performance when adding hardware:

- ▶ *Vertical scaling* means increasing the throughput of the site by handling more requests in parallel. From a hardware perspective, we can increase the number of the processors for the server. For example, by upgrading the system hardware with two or more processors, the server can potentially gain twice the processing capacity. Therefore, more requests should be able to be handled at the same time. This concept is relatively easy to implement and can apply to any server in the system. Examine vertical scaling at every potential bottleneck tier.
- ▶ *Horizontal scaling* means adding duplicate servers to handle additional load. This applies to multi-tier scenarios and can require the use of a load balancer that sits in front of the duplicated servers so that the physical number of servers in your site are hidden from the Web client.

Adding resources usually affects the dynamics of the system and can potentially shift the bottleneck from one resource to another. In a single tier scenario, the Web application and database servers are all running in the same system. Creating a cluster and spreading application servers across systems should improve the throughput. But at the same time, additional servers introduce new communication traffic to the database server. How much network bandwidth will this server configuration consume? What will be the performance improvement by adding more servers?

Because of these unknowns, we recommend that you always perform a scalability test to identify how well the site is performing after a hardware change. Throughput can be measured to ensure that the result meets your expectations.

In addition, be aware that the law of diminishing returns does play a role when increasing scalability either vertically or horizontally. The law of diminishing returns is an economics principle that states that if one factor of production is increased while all others 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 you 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.

9.2.1 Workload categorization

The workload defines how the performance of a system is evaluated. Two key concepts that help identify workload performance are throughput and response time:

- ▶ *Throughput* means the number of requests relative to some unit of time 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. Let us say that each customer on average submits 60 requests per visit; then we can also represent throughput by estimating six visits per minute or 360 visits per hour.
- ▶ *Response time* is the time it takes when the user initiates a request at the browser until the result of the HTML page returns to the browser. At one time, there was an unwritten rule of the Internet known as the *eight second rule*. This rule stated that any page that did not respond within eight seconds would be abandoned. Many enterprises still use this as the response time benchmark threshold for Web applications.

A workload needs the following characteristics:

- ▶ **Measurable:** 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.
- ▶ **Static:** The same results can be achieved no matter for how long you execute the run.

- ▶ **Representative:** The workload realistically represents the stress to the system under normal operating considerations.

Workload should be discerned based on the specifications of the system. If you are developing a data driven system, where 90% of the requests require a database connection, this is a significantly different workload compared to a Web application that makes 90% JSP and servlet calls, with the remaining 10% being messages sent to a back-end store. This requires close work between the application architect and the system architect, which are rarely the same person.

Determine the projected load for a new system as early as possible. In the case of Web applications, it is often difficult to predict traffic flows or to anticipate user demand for the new application. In those cases, estimate using realistic models, and then review the data when the system is launched to adjust expectations.

9.2.2 System tuning

The first step in scaling considerations is to verify that the application and application environment has been adequately tuned. Tuning needs to be verified at the application code layer, the network layer, the server hardware layer, and the WebSphere Application Server layer.

Application code layer

Review the application code itself as part of the regular application life cycle to ensure that it is using the most efficient algorithms and the most current APIs that are available for external dependencies. Take care to use optimized database queries or prepared statements instead of dynamic SQL statements.

Network layer

Tuning at the network layer is usually very cursory. 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. Depending on the operating system, this can cause performance problems due to the negotiation done by the system. Also, reviewing these settings establishes a baseline expectation as you move forward with scaling attempts.

Server hardware layer

Take the time to verify that the servers used in the application environment have been tuned adequately. This includes network options, TCP/IP settings, and even possibly kernel parameters. Verify disk configurations and layout as well. The tuning at this layer can be quite difficult without a specialist in the server

hardware or the operating system, but the adjustments can lead to significant improvements in throughput.

9.2.3 Application environment tuning

Within the WebSphere Application Server environment, there are many settings that can increase application capacity and reduce performance issues. The purpose of this section is not to directly discuss those tuning parameters, which are thoroughly covered in *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392. This section, however, should generate some thoughts as to settings to consider when designing a WebSphere Application Server application.

Java virtual machine (JVM)

When configuring a JVM, look at the minimum and maximum heap sizes closely. The system default for the starting heap size is 2 MB and the default maximum is 64 MB. Neither of these settings is likely desirable.

Starting a JVM with that little memory means that the application must immediately switch context to allocate memory resources. This will slow down the execution of the application until it reaches the heap size it needs to run. Conversely, a JVM that can grow too large does not perform garbage collection often enough, which can leave the machine littered with unused objects.

The levels should be adjusted during testing to ascertain reasonable levels for both settings. In addition, be aware that the prepared statement cache and dynamic fragment caching also consume portions of the heap, and as such, you might be required to make additional adjustments to the heap when those values are adjusted.

Web container thread pool

The thread pool for the Web container should be closely monitored during initial load testing and implementation. This is the most common bottleneck in the application environment. Adjust the number of threads in the pool too high, and the system will spend too much time swapping between threads, and requests will not complete in a timely manner. Adjust the number of threads too low, and the Web server threads can back up and cause a spike in response at the Web server tier.

There are no hard rules in this space, because things such as the type and version of Web server and the percentage of requests that require the application server can impact the overall optimum tuning level. The best guideline is to tune in small increments and measure with identical loads over that time to make sure the adjustment has not crossed a threshold where the performance diminishes.

EJB container

The EJB container can also 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. Set it too low, and the application will spend more time instantiating new EJBs when an existing instance could have been reused. Set it too high, and the application will have a larger memory heap footprint with unused objects remaining in memory. EJB container cache settings can also create performance issues if not properly tuned for the system.

The *Performance Tuning Guide* within the WebSphere Application Server Information Center and other publications, such as *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392, can provide some general rules for adjusting these values to increase the throughput or decrease response time of the system.

Database connection pool

The database connection pool is another common location for bottlenecks, especially in data-driven applications. The default pool size is usually 10, and depending on the nature of the application and the number of requests, this pool can become full quite rapidly. During implementation testing, pay special attention to the pool usage and adjust it higher as needed. This pool consumes additional Java heap as it gets used, so you might be required to go back and adjust the heap size after tuning the pool size. Also, be aware that in a clustered environment, the connection pool is a cell resource. This means that all nodes in the cluster share one pool. Adjust the pool in such way that it is large enough to handle the requests of all the servers participating in the cluster and cell.

9.2.4 Scaling the system

After you verifying and tuning the application, the next step is to examine the entire application environment to identify potential bottlenecks. Simply throwing additional resources into the environment without fully understanding what those resources are addressing can potentially worsen performance or scalability. It is necessary to know the application environment, end-to-end, to quickly identify the bottlenecks and ensure the appropriate response is timely.

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 memory or processors to a network device will increase the throughput of that device, which positively impacts the scaling of that device. Moving from 10 MB to 100 MB or even 1 GB connections can significantly increase performance at the network layer.

When scaling at the Web server layer, the most often employed solution is horizontal scaling. This means adding additional Web servers to the environment. To do so, load balancers must be used. Be careful when adding servers to make sure that the load balancer has adequate capacity, or adding the new Web servers will simply shift the bottleneck from the Web tier to the load balancing tier. Vertical scaling can be performed as well by adjusting HTTP tuning parameters, or by increasing memory or processors.

Scaling at the application server layer can be done with vertical or horizontal scaling, or both. Vertical scaling at the application server layer can be done physically or logically. WebSphere Application Server is a Java application itself and can take advantage of additional processors or memory in the machine. WebSphere Application Server applications can be clustered vertically, providing multiple copies of the same application on the same physical machine. WebSphere Application Server also supports horizontal cloning of applications across multiple machines, which do not necessarily need to be identical in terms of physical resources. A workload manager is required to do a smart load balancing across heterogeneous hardware resources.

At the data services layer, it is most common to implement vertical scaling. Adding multiple copies of the database can introduce complexities that can be unmanageable, and providing these resources to the application server might introduce higher costs than the value provided. A database server, however, can definitely make use of higher numbers of processors or more memory. Most database management systems also can be tuned for performance with respect to I/O, pinned memory, and numbers of processors. Database tuning should be asserted prior to adding additional resources, because these new resources might not improve the problem, and can even increase the problem.

9.2.5 Default messaging provider scalability

With the introduction of the messaging engine and the service integration bus, WebSphere now offers the ability to scale messaging resources more efficiently. Using a cluster as the service integration bus member, you can create a partitioned destination. This enables a single logical queue to be spread across multiple messaging engines. In this scenario, all messaging engines are active all the time. For n cluster members, the theory is that each receives an n th of the messages. This allows for greater scalability of messaging resources across the cell. One key factor to consider in this design is that message order is not preserved. That might or might not be significant, depending on the nature of the application.

Multiple messaging engines for a cluster bus member is not the default. For workload management, you must take manual steps to add additional messaging engines.

9.3 Workload management

Workload management is the concept of sharing requests across multiple instances of a resource. 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 allows for better use of resources by distributing load more evenly. Components that are overworked, and therefore, perhaps 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.

In WebSphere Application Server, workload management is achieved by sharing requests across one or more application servers, each running a copy of the Web application. In more complex topologies, workload management is embedded in load balancing technologies that can be used in front of Web servers.

Workload management (WLM) is a WebSphere facility to provide load balancing and affinity between nodes in a WebSphere clustered environment. WLM can be an important facet of performance. WebSphere uses WLM to send requests to alternate members of the cluster if the current member is too busy to process the request in a timely fashion. WebSphere will route concurrent requests from a user to the same application server to maintain session state.

WLM for WebSphere for z/OS (we refer to this as zWLM to distinguish it) works differently from the WLM for distributed platforms. The workload management structure for incoming requests is handled by the WLM features in z/OS. Organizations can define business-oriented rules that are used to classify incoming requests. As a request arrives at the control region, the CR works with zWLM to classify the request. It then creates a unit of work and assigns the work to a service class suited to its performance requirements. The CR then places the request on a zWLM queue for that service class, where it can be selected for processing by a servant region. zWLM manages the number of servant regions according to workload in the queue. As you can imagine, this explanation is an over-simplification of how workload management works in z/OS. For a good summary of how zWLM works, see *Understanding WAS for z/OS* at:

<http://websphere.sys-con.com/read/98083.htm>

9.3.1 Clustering application servers

Clustering application servers that host Web containers automatically enables plug-in workload management for the application servers and the servlets they host. Routing of servlet requests occurs between the Web server plug-in and the clustered application servers using HTTP or HTTPS, as shown in Figure 9-1.

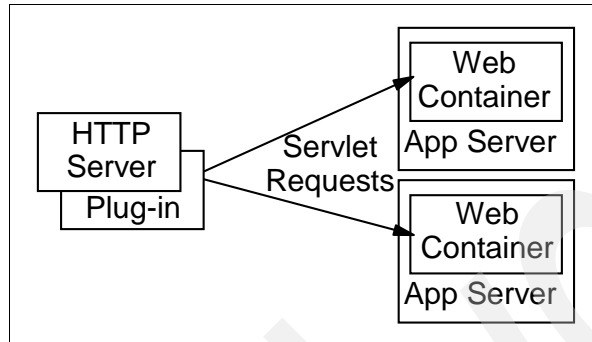


Figure 9-1 Plug-in (Web container) workload management

This routing is based on weights associated with the 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 from 0 to 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. Weights can be changed dynamically during runtime by the administrator.

A guideline formula for determining routing preference is:

$$\% \text{ routed to Server1} = \text{weight1} / (\text{weight1} + \text{weight2} + \dots + \text{weightn})$$

Where there are n cluster members in the cluster.

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

Workload management for EJB containers can be performed by configuring the Web container and EJB containers on separate application servers. Multiple application servers with the EJB containers can be clustered, enabling the distribution of EJB requests between the EJB containers, as shown in Figure 9-2.

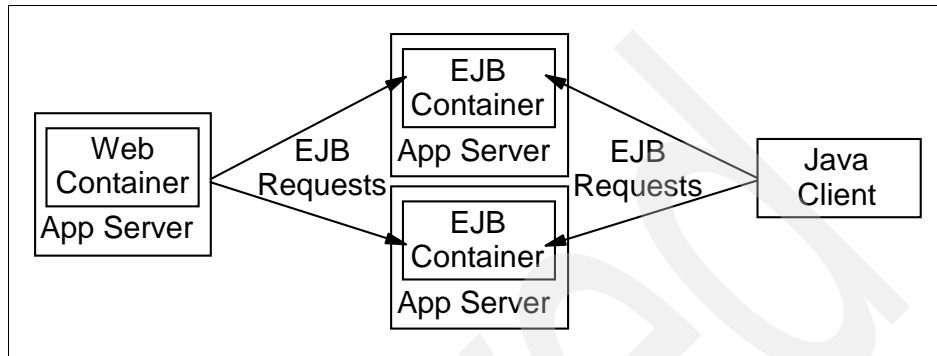


Figure 9-2 EJB workload management

In this configuration, EJB client requests are routed to available EJB containers 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 EJBs.

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. The policy ensures the desired distribution based on the weights assigned to the cluster members.

You can also choose to have requests sent to the node on which the client resides as the preferred routing. In this case, only cluster members on that node are chosen (using the round-robin weight method). Cluster members on remote nodes are chosen only if a local server is not available.

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 will attempt to select the EJB on the same system as the client, eliminating network communication.

9.3.2 Scheduling tasks

WebSphere Application Server provides a Scheduler service that can be used to schedule actions to happen only once, some time in the future, or on a recurring basis or at regular intervals. It can also receive notifications about task activity. Scheduler tasks can be stored in a relational database and can be executed for indefinite repetitions and long time periods. Scheduler tasks can be EJB-based tasks or they can be triggered 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 task is run in the work manager 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 governed by the `Number of alarm threads` parameter on the work manager.

9.4 High availability

Also known as resiliency, availability is the description of the system's ability to respond to requests no matter the circumstances.

Both performance and scalability can affect availability. The key is to determine the threshold for availability. The most common method of describing availability is by the "nines" or the percentage availability for the system. Therefore, 99.9% availability represents 8.5 hours of outage in a single year. To add an additional 9, or 99.99% represents approximately 1 hour of outage in a single year. The cornerstone measurement of "five nines" or 99.999% availability represents an outage of less than five minutes in a year. Calculating availability is a simple process using the following formula, where MTBF is the mean time between failure and MTTR is the mean time to recovery:

$$\text{Availability} = (\text{MTBF}/(\text{MTBF} + \text{MTTR})) \times 100$$

When planning for performance and scalability, consider availability. Make sure, however, that the business case justifies the costs. In many real world examples, moving a system availability from 99.9% to 99.99% can be extremely expensive. It can also be true that the system will only be used during regular business hours on regular working days. This implies that an availability of 99.9% would be more than adequate to meet the operational window.

Because it is likely that the complete environment is made up of multiple systems, the goal is to make the system as available as possible. This can be

done by minimizing the number of single points of failure (SPOF) throughout the system. If an SPOF cannot be eliminated, perform planning to mitigate the impact of that potential failure.

9.4.1 Hardware availability

Hardware can and potentially will fail. Any mechanical component has an expected failure rate and a projected useful life until failure.

To mitigate power failures, you can configure the equipment to 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.

For servers, using multiple network interface cards in an adapter teaming configuration allows a server to bind one IP address to more than one adapter, and then provide failover facilities for the adapter. This, of course, should be extended by plugging each adapter into separate switches as well to mitigate the failure of a switch within the network infrastructure.

Hardware availability for servers at the disk level is also an important consideration. External disk drive arrays and hard disk drive racks can be deployed to provide redundant paths to the data, as well as make the disks available independent of server failure. When working with disks, consider the appropriate RAID levels for your disks.

Network hardware availability can be addressed by most major vendors. There is now built-in support for stateful failover of firewalls, trunking of switches, and failover for routers. These devices also support duplicate power supplies, multiple controllers, and management devices.

Duplicating hardware can have an impact on the cost of the solution. You have to evaluate this increment in the implementation cost against the cost of not having the application available.

9.4.2 Process availability

In WebSphere Application Server, the concept of a singleton process is used. Although not a new concept, it is important to understand what this represents in the environment. A singleton process is an executing function that can exist in only one location at any given instance, or multiple instances of this function operate independently of one another. In any system, there are likely to be singleton processes that are key components of the system functionality. WebSphere Application Server has recognized this as a problem, and has taken steps to address these issues for singleton processes running in the cell.

WebSphere Application Server uses a high availability manager to provide availability for singleton processes. We discuss this further in 9.4.6, “WebSphere Application Server high availability features” on page 183.

9.4.3 Data availability

In a WebSphere Application Server environment, there are many places where data availability is important. For database-driven applications, the database availability is instrumental to the successful execution of the user request. For stateful applications, the user’s session state is important to maintaining the user experience within the application. Stateful session EJB availability is similar in nature to the session state of a servlet request. Lastly, EJB persistence is critical in EJB-driven applications to maintain a variety of user data. The majority of these requirements can be satisfied using facilities available in WebSphere Application Server.

Database server availability

A database server is for many systems the largest and most critical single point of failure in the environment. Depending on the nature of this data, there are many techniques that you can employ to provide availability for this data.

For read-only data, multiple copies of the database can be placed behind a load balancing device, and the client then connects to the virtual IP the load balancer advertises. This enables the application to connect to one of many copies of the data and transparently fail over to the working copy of the resource.

If the data is mostly read-only, consider the possibility of 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.

If the data is read/write and there is no prevalence of read-only access, consider a hardware clustering solution for the database node. This requires external shared disk through Storage Area Network (SAN), network attached storage (NAS), or some other facility that can help to mitigate failure of a single node.

Session data

WebSphere Application Server can persist session data by storing it in application server memory and using memory-to-memory replication to create a copy in one or more additional servers, or by storing the data in an external database. The choice of which to use is really left to the user. External database persistence will survive node failures and application server restarts, but introduces a new single point of failure that must be mitigated using an external hardware clustering or high availability solution. Memory-to-memory replication

can reduce the impact of failure, but if a node fails, the data held on that node is lost.

Stateful session EJB availability is handled using memory-to-memory replication. Using the EJB container properties, you can specify a replication domain for the EJB container and enable the stateful session bean failover using memory-to-memory replication. When enabled, all stateful session beans in the container are able to 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, the bean persistence is available for all members of the cluster. Using access intent policies, you can govern the data access for the persisted bean.

9.4.4 Clustering and failover

Hardware clustering is the concept of creating highly available system processes across multiple servers. It is usually deployed in a manner such that only one of the servers is actively running the system resource.

Hardware clustering is achieved by using an external clustering software, such as IBM HACMP, to create a cluster of servers. Each server is generally attached to a shared disk array through NAS, a SAN, or simply by chaining SCSI connections to an external disk array. Each system has the base software image installed. Depending on the configuration and licensing constraints, the software might be configured to run on each server in the cluster or configured to run from the shared disk so that only the active server has the software. The servers stay in constant communication with each other over several connections through the use of heartbeats. Multiple paths are configured for these heartbeats so that the loss of a switch or network interface does not necessarily cause a failover. Too few paths can create problems with both servers believing they should be the active node. Too many paths can create unnecessary load associated with heartbeat management.

With software clustering, the idea is to create multiple copies of an application component and then have all of these copies available at once, both for availability and scalability. In WebSphere Application Server Network Deployment, application servers can be clustered. This provides both workload management and high availability.

Web containers or EJB containers can be clustered. If an application has both EJB and Web containers, you can create two separate clusters to increase the

overall availability of the system. Whether this helps or hurts performance depends on the nature of the applications and the load.

In a clustered environment, the Web server plug-in module knows the location of all cluster members and routes requests to all of those members. In the case of a failure, the plug-in marks the cluster member as unavailable and does not send requests to that member for a fixed interval. The plug-in periodically retries the path to that cluster member and marks the member available again after the path has been asserted. This retry interval is tunable through the administrative console.

9.4.5 Maintainability

Maintainability is the ability to keep the system running before, during, and after scheduled maintenance. When considering maintainability in performance and scalability, remember that maintenance needs to be periodically performed 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. It is always a distinct possibility that the goal of maintainability will clash with performance. The ideal is that a high performing system is easily maintained, and that is still what you would strive for.

9.4.6 WebSphere Application Server high availability features

This section discusses the WebSphere Application Server features that facilitate high availability. It will help you understand how the high availability features work. Before creating a plan for high availability within WebSphere Application Server, you read *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688.

High availability manager

WebSphere Application Server uses a high availability manager to eliminate single points of failure. A high availability manager is responsible for running key services on available application servers rather than on a dedicated one (such as the deployment manager). It takes advantage of fault tolerant storage technologies such as network attached storage (NAS), which significantly lowers the cost and complexity of high availability configurations. The high availability manager also provides peer-to-peer failover for critical services by always maintaining a backup for these services. WebSphere Application Server also supports other high availability managers such as HACMP.

A high availability manager continually monitors the application server environment. If an application server component fails, the high availability manager takes over the in-flight and in-doubt work for the failed server. This action significantly improves application server availability.

A high availability manager focuses on recovery support and scalability in the following areas:

- ▶ Messaging
- ▶ Transaction managers
- ▶ Workload management controllers
- ▶ Application servers
- ▶ WebSphere partitioning facility instances

To provide this focused failover service, the high availability manager supervises the JVMs of the application servers that are core group members. The high availability 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 very tolerant of extreme application server loading, which might occur if the application server is swapping or thrashing heavily. This method is recommended 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, every second. If the JVM is unresponsive for more than 20 seconds, it is considered down. You can use this method with multicast emulation. However, this method must be used for true multicast addressing.

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 restarted 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.

All of the application servers in a cell are defined as members of a core group. Each core group has only one logical high availability manager that services all of the members of that core group. The high availability manager is responsible for making the services within a core group highly available and scalable. It

continually polls all of the core group members to verify that they are active and healthy.

A policy matching program is used to localize certain policy-driven components and to place these components into high availability groups. When a core group member fails, the high availability manager assigns the failing member's work to the same type of component from the same high availability group. Using NAS devices in the position of common logging facilities helps to recover in-doubt and in-flight work if a component fails.

WebSphere Application Server provides a default core group that is created during installation. New server instances are added to the default core group as they are created. The WebSphere Application Server environment can support multiple core groups, but one core group is usually sufficient for most environments.

Core group

A core group is a set of application servers that can be divided into various high availability groups. It is a statically defined component of the WebSphere Application Server high availability manager function that monitors the application server environment and provides peer-to-peer failover of application server components.

A core group can contain one or more high availability groups. However, a high availability group must be contained totally within a single core group. Any component, such as the service integration bus or the transaction manager, can create a high availability group for that component's use. For example, the service integration bus might need a high availability group to support its messaging engines, and the transaction manager component might need a high availability group to support its transaction logs.

A cell must have at least one core group. One is defined for each cell automatically and is called `DefaultCoreGroup`. All the server processes and JVMs are initially members of this core group.

When properly configured, the default core group is sufficient for establishing a high availability environment. However, certain topologies require the use of multiple core groups. For example, if a firewall is used to separate the proxy environment from the server environment, an additional core group is required in order to provide proper failover support. For this particular type of environment, application servers outside of the firewall are members of a separate core group from the application servers that are inside of the firewall.

The core group contains a bridge service, which 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 to IP addresses and ports. It is through this set of bridge interfaces that the core group bridge provides access to a core group.

If you create additional core groups when you move core group members to the new core groups, remember that:

- ▶ Each server process within a cell can only be a member of one core group. Core groups cannot overlap each other.
- ▶ If a cluster is defined for the cell, all of the cluster members must belong to the same core group.

Network communication between all the members of a core group is essential. The network environment must consist of a fast local area network 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 dynamically created components of a core group. They cannot be configured directly but are directly affected by static data, such as policy configurations, which are specified at the core group level.

A high availability group cannot extend beyond the boundaries of a core group. However, members of a high availability group can also be members of other high availability groups, as long as all of these high availability groups are defined within the same core group.

Any WebSphere Application Server component can create a high availability group for that component to use. The component code must specify the attributes that are used to create the name of the high availability group for that component. For example, to establish a high availability group for the transaction manager:

- ▶ 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.

Whenever transaction manager code joins a high availability group, the high availability manager matches the match criteria of the Clustered TM Policy to the high availability group member name. In this example, the string `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 quorum, fail back, and preferred servers. However, you cannot change the policy type. If you need to change the policy type, you must create a new policy and then use the match criteria to associate it with the appropriate group.

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.

Default messaging provider availability

A messaging engine is considered a singleton process. For the most part, the service integration bus is used to provide high availability to the messaging system process. If the service integration bus member is a cluster, and the cluster member running the messaging engine fails, and the high availability manager is configured for the messaging engine (the default), the service integration bus starts the messaging engine on another member in the cluster.

To accomplish the failover seamlessly, the queue information and messages must be stored in some shared location, either using an external database or a shared disk environment. For those using embedded Cloudscape as a messaging data store, concurrent access can be a concern. The embedded Cloudscape does not support multiple servers running the Cloudscape engine, so there would be no ability to have multiple servers communicating with the same shared file system. Most database engines have minimal support for distributed row locking. This means that it is generally difficult for more than one application to update a data element at the same time. Because the messaging engine is a singleton process, there are no concerns about concurrent access or issues with distributed row locking requirements. Only one cluster member at any given time is executing the process, although any cluster member has the ability to spawn the process in case of the failure of the active cluster member.

9.5 Caching

Caching is a facility to offload some or all of the work to an external device or devices so that the application server is not required to do all of the work associated with a user request. There are caching options at many different layers in a complete system solution. 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 highly varied.

9.5.1 Dynamic caching

Dynamic caching refers to the methods employed by WebSphere Application Server to provide fragment caching or the reuse of components within the application server engine. Fragment caching means only the static portions of a servlet or JSP are cached, and is really a subset of dynamic caching as a whole.

Dynamic caching is enabled at the application server container services level. Cacheable objects are defined inside the `cachespec.xml` file, located inside the Web module `WEB-INF` or enterprise bean `META-INF` directory. The `cachespec.xml` file enables you to configure caching at a per servlet level. The caching options include parameters the request might specify and a timeout value to indicate the cache is no longer valid and should be reloaded at the next request. Pay special attention to the servlet caching configuration, because you can create unexpected results by returning a cached servlet fragment that is stale.

Another facet of caching available is the Edge Side Include (ESI) caching. ESI caching is an in-memory caching at the Web server plug-in. If dynamic caching is enabled at the servlet Web container level, the plug-in uses the ESI caching. An additional header is added to the request from the plug-in, called the Surrogate-Capabilities header, and the application server returns a Surrogate-Control header in the response. Then, depending on the rules specified for servlet caching, you can cache responses for JSP and servlets in the plug-in itself.

9.5.2 Edge caching

Edge caching embraces a variety of methods. Software components at the Web server, such as the Fast Response Cache Accelerator, can be deployed to provide caching of static content at the Web server itself. Reverse proxy engines can be used to cache content even closer to the client. Lastly, there are numerous external caching providers that can provide content offloading at points significantly closer to the client. WebSphere Application Server can be used to configure and manage how these resources are accessed.

Fast Response Cache Accelerator

The first point to mention is that although WebSphere Application Server supports static file serving, the Web server is ultimately more efficient at this type of serving. By separating the two, the application will perform more efficiently. When the static content has been separated from the application content, IBM

HTTP Server provides the Fast Response Cache Accelerator (FRCA) to speed up serving static content. The FRCA can be used to improve the performance of IBM HTTP Server when serving static content, such as images, HTML, and other text files that are not dynamic content. It is configured in the `httpd.conf` file.

This cache can even be configured to enable high speed caching of servlets and JSP files by using the Afa adapter bean through the external caching configuration section of the administrative console.

Caching Proxy

By using the Edge components Caching Proxy, you can intercept requests and then cache the results away from the Web servers or the application servers. This enables you to offload additional work from the primary processing environment. Implementing this caching adds servers and cost to the solution, but can reflect a significant performance improvement and a reduction in response time. This cache can be configured to offload some or all of the site. Using the WebSphere Application Server administrative console, you can also control through WebSphere how the content is loaded and invalidated.

Hardware caching

There are many network equipment providers that sell hardware cache devices. These are very similar to a software cache in the way they can be used to offload content. The main difference is that these appliances are usually not running full versions of an operating system, opting instead for a thinner operating system that is dedicated to the caching function. This can include custom file systems that are higher performance than some operating system file systems and a significantly reduced instruction set. By placing dedicated appliances instead of software caching, you might be able to reduce total cost of ownership, because these appliances do not have to be managed as strictly as machines with full operating systems.

Caching services

There are a variety of providers that now 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. This means the user is not required to travel all the way through the Internet to get to the application serving network to return content. In other words, these providers bring the cached files physically as close as possible to the client.

Caching providers also can lower project costs in the long run in terms of less dedicated staff, equipment, and floor space to internalize the same function. The number of players in this market has exploded in the recent years. Making no

preference or recommendation statements about any of the providers, some of the names involved are:

- ▶ Accelion
- ▶ Activate
- ▶ Akamai
- ▶ AT&T
- ▶ Digital Island
- ▶ EdgeStream
- ▶ Fireclick
- ▶ Limelight Networks
- ▶ Speedera

When selecting an external caching service, make sure that they reach your target market. If your client base is in Europe, and the service only has equipment in the United States, you might not see as much performance enhancement as you would like.

9.5.3 Data caching

Data caching can be a tricky proposition. The key is to minimize the back-end database calls while at the same time assuring the currency of the data. In most cases, the decision for data currency is a business decision. This is a decision that should not be taken lightly, especially if the data is provided by external sources. When configuring data caching, there are multiple methods. The first is to keep a local copy in a database within the same network realm as the application. The second is to cache data from a localized database in memory to minimize database reads. The next is to use EJB persistence to keep the data in the memory space of the running application.

In the case of some data, an external provider maintains the content. Making live calls to this data can prove to be a single point of failure and a slower performer. However, if there are no strict dependencies 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.

Caching data from a local database is a normal operation for a database administrator. There are facilities to minimize the number of times a query must perform a read from disk. Facilities such as fetch ahead constructs attempt to anticipate that additional pages from that table are required and pre-loads those pages into memory pools. Buffer pools offer the ability to keep the data loaded into memory, assuming that it is likely the same portion of the data will be requested again. Both of these constructs limit disk access, opting instead for reading the data from the memory, which is always much faster to read. These

facilities necessarily 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. These can increase performance by minimizing the number of disk reads that are required. Also, memory buffers can be used to house data pages when read into memory, reducing disk access. The key is to make sure that the system has memory to provide to the database.

Another possibility is to cache some of the database or Web page data on an application code level 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 make sure there is some kind of synchronous or asynchronous mechanism, or both, to update this cache on a timely basis according to the system requirements. This approach, however, can create additional memory requirements, especially when a dynamic cache that might grow over the time is implemented.

EJB persistence implies loading the data into an EJB after a call to the data provider. This is similar in nature to database caching, except that the caching takes place in the application space, not 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.

9.6 Session management

Multi-machine 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 completely 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 independent. A client often makes a request, waits for the result, and then makes one or more subsequent requests that depend on the results received from the earlier requests. This sequence of operations on behalf of a client falls into two categories:

- ▶ **Stateless**

A server processes requests based solely on information provided with each request and does not rely on information from earlier requests. In other words, 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. In other words, the server needs to access and maintain state information that is generated during the processing of an earlier request.

For stateless interactions, it does not matter whether different requests are processed by different servers. However, for stateful interactions, the server that processes a request needs access to the state information necessary to service that request. Either the same server can process all requests that are associated with the same state information or the state information can be shared by all servers that require it. In the latter case, accessing the shared state information from the same server minimizes the processing associated with accessing the shared state information from multiple servers.

The load distribution facilities in WebSphere Application Server use several different techniques for maintaining state information between client requests:

- ▶ **Session affinity**, where 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**, where 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**, where 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 WebSphere Application Server session manager, which is part of each application server, stores client session information and 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 the cluster members of an application server.

9.6.1 Session support

Information entered by a user in a Web application is often needed throughout the application. For example, a choice made by a user might be used by the application to determine the path through future menus or the options to show the user. This information is kept in a session.

A session is a series of requests to a servlet that originate from the same user at the same browser. Each request arriving at the servlet contains a session ID,

allowing 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, allocating session IDs that identify a specific session, and tracking the session ID associated with each client request through the use of cookies or URL rewriting techniques.

When planning for session data, there are three basic considerations: application design, session tracking mechanism, and session storage options. The following sections outline the planning considerations for each.

Note: Before designing your session management topology, review Chapter 10, “Session management,” in *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

Application design

Although using session information is a simple and convenient method for the developer, it should be minimized. If sessions are persisted during runtime, there can be a cost in performance.

Session tracking mechanism

You can choose to use cookies, URL rewriting, SSL session IDs, or a combination of these as the mechanism for managing 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 browser to be stored as a cookie. This option can be a problem if you anticipate having users that have disabled the use of cookies on their browsers.

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 limits the flow of site pages exclusively to dynamically generated pages, such as pages generated by servlets or JSPs. Therefore, after the application creates the user's session data, the user must visit dynamically generated pages exclusively until they finish with the portion of the site requiring sessions. 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 HTTP server, it cannot survive an HTTP server failure. However, the failure of an application server does not affect the SSL session ID, and if a distributed session is not configured, the session itself is lost. In environments that use WebSphere components with multiple HTTP servers, you must use an affinity mechanism when SSL session ID is used as the session tracking mechanism.

SSL tracking is supported only for IBM HTTP Server and Sun ONE Web Server.

The lifetime of an SSL session ID can be controlled by configuration options in the Web server. For example, in IBM HTTP Server, the configuration variable `SSLV3TIMEOUT` must be set to allow for an adequate lifetime for the SSL session ID. Too short an interval can result in a premature termination of a session. Also, some Web browsers might have their own timers that affect the lifetime of the SSL session ID. These Web browsers might not leave the SSL session ID active long enough to be useful as a mechanism for session tracking.

When the SSL session ID is to be 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 rewritten URL but is derived from the SSL information.

The main disadvantage of using SSL ID tracking is the performance hit of using SSL. If you have a business requirement to use SSL, this is a good choice. If you do not have such a requirement, it is probably a good idea to consider using cookies instead.

Selecting multiple tracking mechanisms

It is possible to select all three options for a Web application. If you do this:

- ▶ SSL session identifiers are used in preference to cookie and URL rewriting.
- ▶ Cookies are used in preference to URL rewriting.

If selecting SSL session ID tracking, we recommended that you also select cookies or URL rewriting so that session affinity can be maintained. The cookie or rewritten URL contains session affinity information enabling 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 in the application server memory (local), in a database, or using memory-to-memory data replication. The

last two options allow session data to be accessed by multiple servers and should be considered when planning for failover.

Storing session data external to the server can have drawbacks in performance. The amount of impact depends on 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 store, especially when consecutive requests are routed to the same application server.

Local sessions (non-persistent)

If you choose to use application server memory, the session data is not available to any other servers. Although the fastest option and the simplest to set up, an application server failure destroys the session.

The following settings can help you manage the local session storage:

- ▶ *Maximum in-memory session count*: Enables you to define a limit to the number of sessions in memory. This prevents the sessions from acquiring too much of the JVM memory.
- ▶ *Allow overflow*: Permits an unlimited number of sessions. If you choose this option, monitor the session cache size closely.
- ▶ *Session timeout*: 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 define it to WebSphere.

The *Use multi-row schema* setting gives you the option to use multi-row sessions to support large session objects. With multi-row support, the WebSphere session manager breaks the session data across multiple rows if the size of the session object exceeds the size for a row. This also provides a more efficient mechanism for storing and retrieving session contents under certain circumstances.

Memory-to-memory replication for persistent sessions

Memory-to-memory replication copies session data across application servers in a cluster, storing the data in the memory of an application server and providing session persistence. Using memory-to-memory replication eliminates the cost of maintaining a production database. It also eliminates the single point of failure that can occur with a database.

The administrator sets up memory-to-memory replication by creating a replication domain and adding application servers to it.

Manage replication domains from the administrative console by navigating to **Environment** → **Replication domain**.

When defining a replication domain, you must specify whether each session is replicated to one server (single replica), to every server (entire domain), or to a defined number of servers. The number of replicas can affect performance. Smaller numbers of replicas result in better performance because the data does not have to be copied as many times. However, if you create more replicas, you have more redundancy in your system. By configuring more replicas, your system becomes more tolerant to possible failures of application servers in the system because the data is backed up in several locations.

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 sessions. It does not send copies of sessions created in that particular server.
- ▶ **Client mode**
In this mode, a server only broadcasts or sends copies of the sessions it owns. It does not receive backup copies of sessions from other servers.
- ▶ **Both mode**
In this mode, the server simultaneously sends copies of the sessions it owns and acts as a backup table for sessions owned by other application servers. Because each server has a copy of all sessions, this mode uses the most storage on each server. Replication of sessions can impact performance.

Session manager settings

Session management in WebSphere Application Server can be defined at the following levels:

- ▶ **Application server**
This is the default level. Configuration at this level is applied to all Web modules within the server.
Navigate to **Servers** → **Application servers** → **<server_name>** → **Session management** → **Distributed environment settings** → **Memory-to-memory replication**.
- ▶ **Application**
Configuration at this level is applied to all Web modules within the application.
Navigate to **Applications** → **Enterprise applications** → **<app_name>** → **Session management** → **Distributed environment settings** → **Memory-to-memory replication**.

- ▶ Web module

Configuration at this level is applied only to that Web module.

Navigate to **Applications** → **Enterprise applications** → **<app_name>** → **Manage modules** → **<Web_module>** → **Session management** → **Distributed environment settings** → **Memory-to-memory replication**.

With one exception, the session management properties you can set are the same at each configuration level:

- ▶ *Session tracking mechanism* lets you select from cookies, URL rewriting, and SSL ID tracking. Selecting cookies will lead you to a second configuration page containing further configuration options.
- ▶ Select *Maximum in-memory session count* and whether to allow this number to be exceeded, or overflow.
- ▶ *Session timeout* specifies the amount of time to allow a session to remain idle before invalidation.
- ▶ *Security integration* specifies that the user ID be associated with the HTTP session.
- ▶ *Serialize session access* determines if concurrent session access in a given server is allowed.
- ▶ *Overwrite session management*, for enterprise application and Web module level only, determines whether these session management settings are used for the current module, or if the settings are used from the parent object.
- ▶ *Distributed environment settings* select how to persist sessions (memory-to-memory replication or a database) and set tuning properties. Memory-to-memory persistence is only available in a Network Deployment distributed server environment.

9.7 Data replication service

The data replication service (DRS) is the WebSphere Application Server component that replicates data. Session manager, dynamic cache, and stateful session beans are the three consumers (users) of the replication service.

To use data replication for these services, you must first create the replication domains:

- ▶ Create one replication domain for dynamic cache. The replication domain must be configured for full group replication (both mode).
- ▶ Create one replication domain to handle sessions for both HTTP sessions and stateful session beans.

We discuss replication domains and session management memory-to-memory replication in 9.6.1, “Session support” on page 192.

Configure dynamic cache replication through **Servers** → **Application servers** → **<server_name>** → **Container services** → **Dynamic cache service**.

Session management for stateful session beans in WebSphere Application Server can be defined at the following levels:

- ▶ Application server EJB container
Navigate to **Servers** → **Application servers** → **<server_name>** → **EJB Container**.
- ▶ Application
Navigate to **Applications** → **Enterprise applications** → **<app_name>** → **Stateful session bean failover settings**.
- ▶ EJB module
Navigate to **Applications** → **Enterprise applications** → **<app_name>** → **Manage modules** → **<EJB_module>** → **Stateful session bean failover settings**.

Note for migration: Any replication domains that were created in a previous version of WebSphere might be multi-brokered domains, as opposed to the new data replication domains in WebSphere Application Server V6. If existing multi-broker domains remain functional, however, after you upgrade your deployment manager, you can create only data replication domains in the administrative console. For improved performance and easier configuration, migrate any existing multi-broker domains to the new data replication domains.

9.8 WebSphere Application Server performance tools

When reviewing the application environment, you often need to delve deeper into the behavior of the application than what is presented at the operating system layer. This requires the use of specialized tools to capture this information. WebSphere Application Server provides tools that allow the administrator to gather information related to the performance of various components in the J2EE environment. In this section, we discuss the enhanced Tivoli Performance Viewer and the request metrics available for WebSphere transactions.

9.8.1 Performance Monitoring Infrastructure

The Tivoli Performance Viewer is a monitoring tool that is used to capture data presented through the WebSphere Application Server Performance Monitoring Infrastructure (PMI). The PMI is a server-side monitoring component. WebSphere PMI complies with J2EE 1.4 standards by implementing the J2EE 1.4 Performance Data Framework. This means that the old API has been deprecated, although it will continue to exist to allow for migration to the new standard.

Configured through the administrative console, the PMI allows monitoring tools to peer inside the WebSphere environment to capture specific details about the application behavior. Using this interface, you are able to capture information about the following resources and more:

- ▶ Customer's application resources
 - Custom PMI
 - EJBs
 - Servlets/JSPs
 - Web services
 - Any component written by the client that runs in the environment
- ▶ WebSphere runtime resources
 - JVM memory
 - Thread pools
 - Database connection pools
 - Session persistence data
- ▶ System resources
 - Processor usage
 - Total free memory
 - Components that are controlled outside the WebSphere environment but that are vital in healthy application state

PMI now offers the Custom PMI API. This enables you to insert your own 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 categories:

- ▶ Basic
 - J2EE components
 - CPU usage
 - HTTP session information

- ▶ Extended (basic +)
 - WLM
 - Dynamic cache
- ▶ All
- ▶ Custom (select your own mix of metrics)

The Java Virtual Machine Tool Interface (JVMTI) is a native programming interface that provides tools the ability to inspect the state of the JVM. This interface is new for the JVM V1.5. JVMTI replaces the Java Virtual Machine Profiling Interface (JVMPPI), which is supported in WebSphere Application Server, Version 6.0.2 and earlier. The JVMPPI interface is deprecated as of WebSphere Application Server Version 6.1. Both interfaces (JVMTI and JVMPPI) provide the ability to collect information about the JVM that runs the application server. The statistics that are gathered through the JVM Tool Interface (JVMTI) is different between the JVM provided by IBM and the Sun HotSpot-based JVM, including Sun HotSpot JVM on Solaris and the HP 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.

Monitoring a system naturally changes the nature of the system. Introducing performance metrics consumes some resources for the application. In WebSphere Application Server V6, this impact has been minimized, though obviously the more statistics you capture, the more processing power is required.

9.8.2 Tivoli Performance Viewer

Tivoli Performance Viewer is the tool included with WebSphere Application Server V6 for measuring performance. Unlike its predecessor, however, this version is now integrated into the administrative console.

Using Tivoli Performance Viewer, you can:

- ▶ Display PMI data collected from local and remote application servers:
 - Summary reports show key areas of contention.
 - Graphical/tabular views of raw PMI data.
 - Optionally save collected PMI data to logs.
- ▶ Provide configuration advice through performance advisor section:
 - Tuning advice formulated from gathered PMI and configuration data.

You use Tivoli Performance Viewer to create summary reports. These reports let you monitor the server's real-time performance and health. Tivoli Performance Viewer enables you to work with the performance modules. With these modules,

you drill down on specific areas of interest, even including old logs. Use the log analysis tools to detect trends over time. Tivoli Performance Viewer can also save performance data for later analysis or problem determination.

9.8.3 WebSphere performance advisors

Gathering information made available through the PMI, the WebSphere performance advisors have the ability to make suggestions about the environment. The advisors are able to determine the current configuration for an application server, and trending the PMI data over time, make informed decisions about potential environmental changes that can enhance the performance of the system. Advice is hard coded into the system and is based on IBM best practices for tuning and performance. The advisors do not implement any changes to the environment. Instead, they identify the problem and allow the system administrator to make the decision whether or not to implement. You should test after any change is implemented from the advisor suggestions. There are two types of advisor: the Runtime Advisor and the Performance Advisor in Tivoli Performance Viewer.

Runtime Advisor

The Runtime Advisor is configured through the administrative console. The Runtime Advisor writes to the SystemOut.log and to the console while in monitor mode. The interface is configurable to determine how often data is gathered and advice is written. The Runtime Advisor offers advice about the following components:

- ▶ ORB service thread pools
- ▶ Web container thread pools
- ▶ Connection pool size
- ▶ Persisted session size and time
- ▶ Prepared statement cache size
- ▶ Session cache size

This is not the complete set of items monitored through the PMI. If you need to gather advice about items outside this list, you need to use the Tivoli Performance Viewer Advisor. The Runtime Advisor, by its nature, does not have the ability to play back data captured offline.

Performance Advisor in Tivoli Performance Viewer

This advisor is slightly different from the Runtime Advisor. The Performance Advisor in Tivoli Performance Viewer is invoked only through the Tivoli Performance Viewer interface. It still runs on the application server you are monitoring, but the refresh intervals are based on selecting refresh through the console instead of fixed intervals. Also, the output is to the user interface instead

of to an application server output log. This advisor also captures data and gives advice about more components than the Runtime Viewer. Specifically, this advisor can capture:

- ▶ 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

This advisor is used more for calculation-intensive operations. The metrics it monitors can become quite large and can potentially impact performance if analyzed through the Runtime Advisor. It is an on demand tool, much more suited to problem analysis and avoidance.

9.8.4 WebSphere request metrics

Request metrics gather information about single transactions within an application. The metric tracks each step in a transaction and determines the process time for each of the major application components. Several components support this transaction metric, including:

- ▶ Web server plug-ins
- ▶ Web container
- ▶ EJB container
- ▶ JDBC calls
- ▶ Web services engine
- ▶ Default messaging provider

The amount of time that is spent in each component is measured and aggregated to define the complete transaction time for that transaction. Both the individual component times and the overall transaction time can be very useful metrics when trying to gauge user experience on the site. The data allows for a hierarchical view for each individual transaction by response time. When debugging resource constraints, these metrics provide critical data at each component. The request metric can provide filtering mechanisms to monitor synthetic transactions or to track the performance of a specific transaction. By using artificial transactions, you can measure performance of the site end-to-end.

From a performance perspective, using transaction request metrics can aid in determining if an application is meeting service level agreements (SLAs) for the client. The metrics can be used to alert when an SLA target is not met.

Request metrics help administrators answer the following questions:

- ▶ What performance area should be focused on?
- ▶ Is there too much time being spent on any given area?
- ▶ How do I determine if response times for transactions are meeting their goals?
- ▶ How can I validate that SLA agreements are being met?

Those familiar with the Application Response Measurement (ARM) standard know that beginning in WebSphere Application Server V5.1, the environment was ARM 4.0 compliant. WebSphere Application Server V6 extended the attributes measured to include Web services, JMS, and asynchronous beans.

Implementing request metrics

There are several methods for implementing request metrics. This section briefly discusses 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 you care to monitor. As an example, you can use an IP address filter to monitor synthetic transactions that always come from the same server. Some of the available filters are:

- ▶ HTTP requests: Filtered by IP address, URI, or both
- ▶ Enterprise bean requests: Filtered by method name
- ▶ JMS requests: Filtered by parameters
- ▶ Web services requests: Filtered by parameters

The performance impact is less than about 5% when all incoming transactions are being instrumented. This is not a significant amount, but factor this in when implementing the metrics. The console path for implementation is through **Monitoring and Tuning → Request Metrics**.

Tracing

By setting the trace depth, you control not only the depth of information gathered through the metric, but also the overall performance hit on the system. The higher a tracing level, the greater the performance hit the system takes. There are several available trace levels:

- ▶ 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)

Set the tracing levels in the console through the same path, **Monitoring and Tuning** → **Request Metrics**.

Output for request metrics

The data captured by request metrics is placed in several levels, depending on the nature of the metric selected. For Web requests, the HTTP request is logged to the output file 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 SystemOut.log for that application server. The data can also be output to an ARM agent and visualized 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 direct data to either the logs or the agent, or both at the same time. The best practice, however, is to not use metric logging while implementing the ARM agent monitoring, because the disk I/O can negatively impact performance.

Application Response Measurement (ARM)

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. In doing so, the request metrics use call across the ARM API to gather the data.

For more information about ARM, review:

<http://www.opengroup.org/tech/management/arm/>

WebSphere request metrics support the Open Group ARM 4.0 Standard, as well as the Tivoli ARM 2.0. The 4.0 standard is supported in all components. For the Tivoli standard, WebSphere Application Server V6 supports all components except Web server plug-ins. A correlator can be extracted from request metrics transactions. This correlator can be passed to sub-transactions taking place in non-WebSphere containers. This facility allows the complete transaction to be accurately timed in complex environments.

Additional resources

For additional information about ARM and the WebSphere request metrics implementation, refer to the following links:

- ▶ ARM 4.0 specification
<http://www.opengroup.org/management/arm.htm/>
- ▶ Information about the ARM standard
<http://www.opengroup.org/pubs/catalog/c807.htm>

- ▶ IBM Tivoli Monitoring for Transaction Performance
<http://www.ibm.com/software/tivoli/products/monitor-transaction/>
- ▶ IBM Enterprise Workload Management
<http://www.ibm.com/servers/eserver/about/virtualization/enterprise/ewlm.html>

9.9 Planning checklist for performance

Table 9-1 provides a summary of items to consider as you plan and additional resources that can help you.

Table 9-1 Planning checklist for Web services

Planning item
Establish performance goals and identify workload characteristics (throughput, response time, availability).
Design your topology to meet the performance goals. Determine if clustering will be used. Determine if the appropriate mechanisms are in place for workload management and failover. As part of this, you need to consider where applications will be deployed (see 7.11, “Mapping applications to application servers” on page 150).
Implement a monitoring system to watch for performance problems and to assist in determining if adjustments are necessary.
Monitor the following as potential physical bottleneck areas: <ul style="list-style-type: none"> ▶ Network load balancers ▶ Firewalls ▶ Application servers ▶ Database servers ▶ LTPA providers

Planning item
<p>Examine initial settings for performance tuning parameters and adjust if necessary. Reevaluate these periodically:</p> <ul style="list-style-type: none"> ▶ JVM heap maximum and minimum sizes ▶ Web container <ul style="list-style-type: none"> – Thread pool – Maximum persistent requests – Timeout values ▶ EJB container <ul style="list-style-type: none"> – Inactive pool cleanup interval – Cache size ▶ Database connection pool <ul style="list-style-type: none"> – Maximum connections – Unused timeout – Purge policy ▶ Database servers <ul style="list-style-type: none"> – 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 <ul style="list-style-type: none"> – Database tuning – Authentication cache intervals
<p>Plan for clustering:</p> <ul style="list-style-type: none"> ▶ Number of application servers ▶ Physical location ▶ Server weights ▶ Prefer local setting
<p>Consider the Scheduler service to run intensive tasks in off-peak hours.</p>
<p>Evaluate session management needs:</p> <ul style="list-style-type: none"> ▶ Session ID mechanism (cookies, URL rewriting, SSL) ▶ Session timeout values ▶ Session, transaction, and server affinity ▶ Distributed session data store (memory-to-memory or database store)
<p>For messaging applications using the default messaging provider, consider:</p> <ul style="list-style-type: none"> ▶ Quality of service settings ▶ Bus topology

Resources

For a good overall reference for performance and scalability planning in WebSphere Application Server, refer to the following IBM Redbooks:

- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- ▶ *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688

We suggest that you have a copy of these books available as you plan your environment.

For information about session management, see *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

The WebSphere Application Server Information Center also contains a lot of useful information.

For a good entry point to monitoring topics, see:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6topmonitoring.html>

For a good entry point to performance topics, see:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6toptuning.html>

Archived

Planning for messaging

In this chapter, we discuss planning for a WebSphere Application Server V6.1 environment that uses messaging facilities. WebSphere Application Server V6.0 introduced significant changes to the messaging environment. A number of further improvements have been added with V6.1. Depending on your current messaging needs, carefully read the information that is provided in this chapter. This chapter asks the following questions:

- ▶ Messaging overview: What is messaging?
- ▶ What is new in messaging for V6.1
- ▶ Messaging considerations: Is messaging for me?
- ▶ Messaging options: What things do I need?
- ▶ Messaging topologies: How can I use messaging?
- ▶ Messaging features: How secure and reliable is it?
- ▶ Planning checklist for messaging

This chapter briefly describes the concepts required to understand messaging. For detailed information, refer to *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

10.1 Messaging overview: What is messaging?

Generically, the term *messaging* describes communication, or exchange of information, between two or more interested parties. Messaging can take many shapes and forms, such as a sending a fax message from one point to another, which is an example of point-to-point messaging. An e-mail sent to a mailing list is an example of the publish/subscribe messaging concept, where a single message is sent to many destinations.

However, for the purposes of this chapter, we define messaging 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 even 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, which then does additional processing. The end result can be that the producer receives some data back from the consumer or that the consumer does some processing task for the producer.

Messaging is a very popular facility for exchanging data between applications and clients of very different types. It is also an excellent tool for communication between heterogeneous platforms. WebSphere Application Server recognizes the power of messaging and implements a powerful and flexible messaging platform within the WebSphere environment that is continually being improved.

10.2 What is new in messaging for V6.1

WebSphere Application Server V6.1 introduced some improvements to the service integration bus from Version 6.0. We briefly describe these here. For those new to WebSphere Application Server messaging, we explain the concepts later in this chapter.

- ▶ Introduction of file stores for messaging engines

Messaging engines can now have their message stores, used for storing persistent data, implemented as flat files (file stores) in addition to the traditional database implementation (data stores). These files can be administered through the host operating system and can potentially increase messaging engine performance and throughput.

- ▶ Improved service integration bus security

Security surrounding the service integration bus has been enhanced to provide much more flexibility and robustness to your security configuration.

- ▶ Introduction of WebSphere MQ server

For those who want to access WebSphere MQ on a z/OS platform, there is a new mechanism called WebSphere MQ server that enables applications to take advantage of the high availability and load balancing features of the MQ queue sharing groups that the z/OS implementation of MQ provides.

- ▶ Strict message ordering enhancement for bus destinations

Defined destinations inside a bus can now be configured to be much more strict in the delivery of messages in the order that they were produced. When enabled, certain automatic restrictions are placed on the use of the destination, such as disallowing concurrent consumption of messages by multiple applications, which can disrupt message ordering.

10.3 Messaging considerations: Is messaging for me?

Messaging can be a powerful technology for applications. It is important to remember, however, that not all environments need to use messaging. Just because WebSphere Application Server provides messaging does not mean the technology is a panacea for applications. Note the following high-level considerations when choosing whether messaging is appropriate for your environment:

- ▶ Is your application complex or simple?

If your application just uses servlets to provide Web presentation services for dynamic content, it is unlikely that messaging would add any degree of value to the application. If, however, the application is data driven or has many tiers of access, messaging can provide a good solution.

- ▶ Does your application environment have multiple platforms?

When dealing with heterogeneous systems, messaging can provide a common input and output mechanism that crosses platforms easily. If the environment is predominately homogenous, messaging might not do any more than add a layer of complexity that is generally undesirable.

- ▶ Does your application need to cross security realms?

Some application environments require access to data that might not be in the same security realm as the application servers. This does not just mean a demilitarized zone (DMZ) environment, because some large enterprises use multiple security realms within an environment that would not be considered appropriate for a DMZ. Messaging provides Secure Sockets Layer (SSL) communication and cryptography to enhance the security of the system overall.

- ▶ Does your current environment use messaging?
If your environment currently is a robust and stable WebSphere MQ environment, the possibility is high that your applications will be more robust and stable if they implement messaging.
- ▶ What are your application resiliency requirements?
Not all applications require high availability or high degrees of reliability. Introduce messaging into an environment where some back-end resource is a potential single point of failure and you need to minimize the impact of that component being unavailable. Messaging itself can introduce a single point of failure, especially when integrating into existing messaging environments, so be careful as to how the choice is made.
- ▶ Are your developers comfortable with messaging technologies?
Many developers have worked without using messaging for quite some time. They have found ways using session state, EJBs, and other tricks to implement some of the same functionality without using messaging constructs directly. If the developers are not comfortable using messaging, you can be turning on an environment that might consume resources that could be put to use elsewhere in the application environment.

There are many more questions that you can ask when considering messaging. The point really is that just because a guide discusses it or a software product offers it as a technology does not mean the application will make use of it. The best advice is to use the keep it simple and straightforward (KISS) method.

10.4 Messaging options: What things do I need?

In this section, we discuss at a high level how messaging is implemented within WebSphere Application Server and what questions need consideration. This information helps you when:

- ▶ Selecting a messaging service type
- ▶ Choosing a messaging service provider

10.4.1 Selecting a messaging service type

To implement messaging within your application, either as a message producer, consumer, or both, your application needs to communicate with a messaging service provider. Examples of messaging provider middleware include the default messaging provider in WebSphere Application Server, WebSphere MQ, Oracle Enterprise Messaging Service, SonicMQ, and many others.

Your application code can interact with these providers in one of three ways, through the JMS API, through the J2EE Connector Architecture API (JCA), or directly through provider-specific client libraries. In the following sections, we briefly explain these terms and discuss their use.

Java Message Service (JMS)

JMS is the standard API for accessing enterprise messaging systems from Java-based applications. It provides methods and functions that are directly implemented by the underlying messaging provider. WebSphere Application Server V6.1 supports version 1.1 of the specification, which forms part of the overall J2EE 1.4 specification. For more information about the JMS V1.1 specification, see the Sun Developer Network Java Message Service Web page:

<http://java.sun.com/products/jms>

We recommend using JMS in preference to anything else 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.

For the rest of the chapter, we assume that JMS is the chosen method to access the messaging provider middleware.

J2EE Connector Architecture (JCA)

The J2EE Connector Architecture is a standard way for a Java component to connect to any enterprise system, not just messaging provider software. WebSphere Application Server V6.1 supports version 1.5 of the JCA specification, which states that an application communicates with an enterprise system through a *resource adaptor*. A resource adaptor is supplied by the vendor of the enterprise system that is compliant with the JCA API (a bit like a software driver). This provides a level of abstraction between your application and the enterprise system, allowing your application to still be portable between enterprise systems. For more information about the JCA V1.5 specification, see the Sun Developer Network J2EE Connector Architecture Web page:

<http://java.sun.com/j2ee/connector/>

We recommend using JMS in preference to JCA whenever you are accessing a messaging provider, because the JCA API is much less messaging specific, and therefore harder and more prone to error. However, it is a viable alternative when your chosen provider does not support JMS.

Vendor-specific client libraries

As the name suggests, these are libraries supplied by a software vendor so that applications can interact with their software. These libraries are similar to resource adaptors except in a few important ways:

- ▶ They are proprietary and do not usually conform to any open standard.
- ▶ Their use renders your application non-portable across enterprise systems, and probably across platforms as well.
- ▶ There might not be support for certain languages such as Java, and these libraries have no direct support in WebSphere Application Server.

We recommend that you do not use 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 Choosing a messaging service provider

WebSphere Application Server supports several JMS message providers (for the rest of this chapter, we assume that the external messaging provider is WebSphere MQ for the purposes of explanation):

- ▶ WebSphere Application Server *default messaging provider*

This full featured messaging provider comes free with WebSphere Application Server. It is a robust and stable messaging platform that can handle any number of point-to-point queues, topics in a publish-subscribe environment, or Web service endpoints.

- ▶ WebSphere MQ messaging provider

WebSphere MQ is the premier messaging middleware provided by IBM. We recommend WebSphere MQ when you require advanced messaging facilities and options. WebSphere MQ has been around for a lot longer than the WebSphere Application Server default messaging provider and is available on many platforms, supporting many programming languages. It is fully JMS compliant and has a large client base.

- ▶ Generic JMS provider

This 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.

This approach is only recommended if you have an existing investment in a third-party messaging provider, because much greater support is available in WebSphere Application Server for the default and WebSphere MQ messaging providers. For the rest of this chapter, we assume that the external messaging provider is WebSphere MQ for the purposes of explanation.

- ▶ V5 default messaging provider

The V5 default messaging provider is supported for migration purposes only.

Note: The remainder of this chapter focuses on the default messaging provider. Planning for external providers is out of the scope for this document.

10.5 Messaging topologies: How can I use messaging?

Choosing a topology depends largely on the answers to many questions about the topology of the application and your own messaging requirements. Some of the more important questions include:

- ▶ What is the topology of my application?
- ▶ Can I break it up into logical parts that can be separately deployed?
- ▶ Which parts need to communicate with which others?
- ▶ Are there natural divisions within the application that are autonomous, needing separate communication channels?
- ▶ Does my application need to communicate with external systems?
- ▶ Do I need to balance the messaging workload for each part?
- ▶ Are there any critical parts that need to have high availability?
- ▶ Will I need application server clustering, or do I have it already?

The following sections try to generally outline what topology will best fit your needs depending on the answers to the previous questions. In most cases, the topology will not be clear cut because there will be many different ways to implement a messaging application. However, the simpler, the better.

Note: This section provides a high-level look at messaging topologies, focusing on the default messaging provider. Before designing anything but the simplest topology for messaging, it is important that you understand how the default messaging provider handles messages. Review the chapters about asynchronous messaging and the default messaging provider in *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

10.5.1 Default messaging provider concepts

Some concepts need described very briefly in order to understand the basic intent of the topologies.

Service integration bus

The service integration bus (or just the bus) provides the transport mechanism for the default messaging provider. It is a group of interconnected application servers and server clusters that have been added as part of the bus. Each member of the bus has a *messaging engine* so that the applications can connect to the bus.

Messaging engine

The messaging engine is the core part of the application server bus member that accomplishes the communication of the message to its *destination*, in cooperation with the other messaging engines of other bus members.

Destinations

A destination is defined within a bus and represents a logical address to which applications can attach as message producers, consumers, or both. There are different types of destinations, which are used for different message models, such as point-to-point or publish/subscribe. Destinations are associated with a messaging engine using a message point.

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 (this is a specialized message point):

- ▶ Queue points

A queue point is 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 associated with the specified bus member.

If the bus member is an application server, a single queue point will be created and associated with the messaging engine on that application server. All of the messages that are sent to the queue destination will be 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

A publication point is 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.

Foreign bus and link

In reality, a foreign bus is just another service integration bus with its own set of members and destinations. You can set up a link to it so that messages traverse from one bus to another. The WebSphere MQ network can be seen as a foreign bus by the default messaging provider using a WebSphere MQ link.

10.5.2 Choosing a messaging topology

The following topologies are some of the typical ones implemented by the default messaging provider (in increasing complexity) using the previously defined concepts.

One bus, one bus member (application server)

This is the simplest and most common topology. It is chiefly 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. See Figure 10-1.

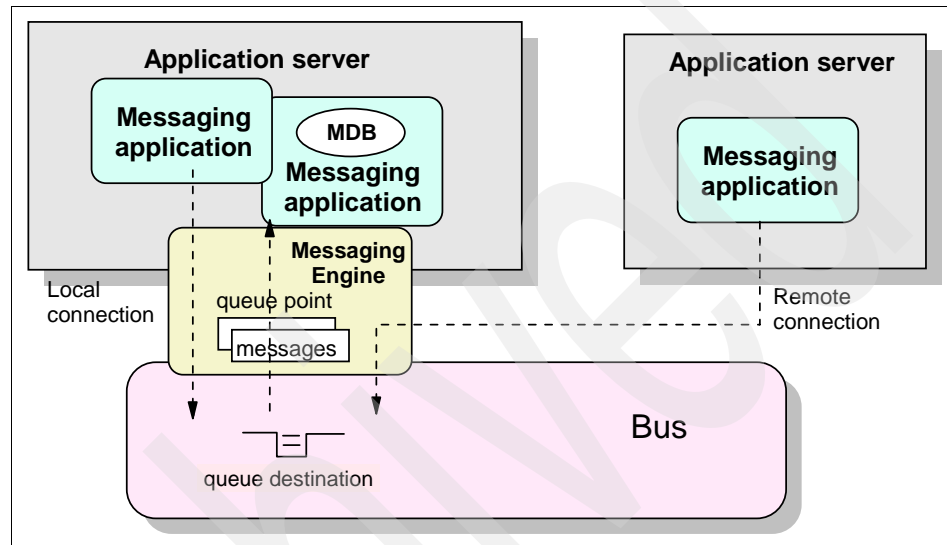


Figure 10-1 Single bus with an application server member

Although this is simple to set up, there might be a performance impact for message producers and consumers that connect to the messaging engine remotely.

Because the single messaging engine is running on a non-clustered application server, no high availability or workload management is supported.

One bus, one bus member (server cluster)

With this variation, the bus member is a cluster. By default, only one 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. This 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 accessing the bus remotely can consume asynchronous messages from remote messaging engine. However, an instance

of a message-driven bean (MDB) deployed to the cluster can only consume from a local messaging engine. See Figure 10-2.

Because everything is funneled through one messaging engine, performance might still be an issue.

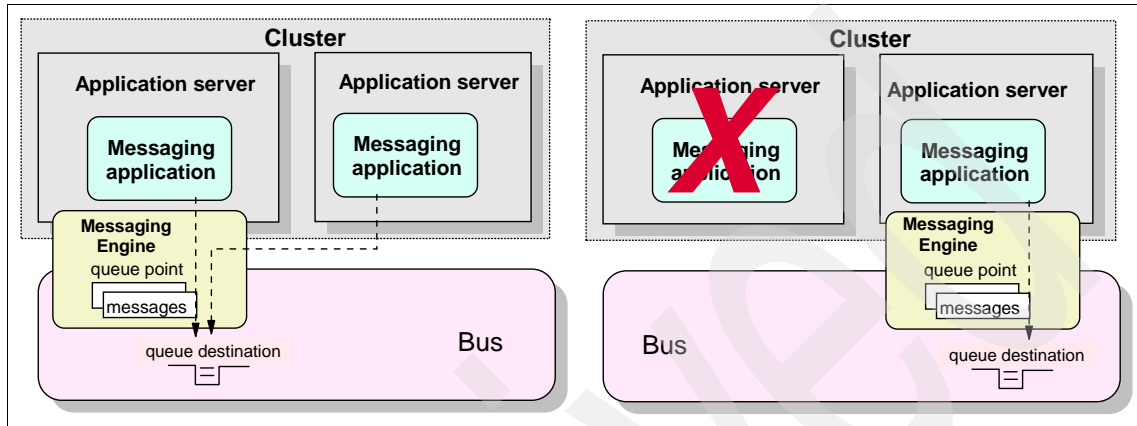


Figure 10-2 Single bus with a cluster member: High availability

There is the concept of preferred servers with clustering, for example, a primary server and a backup server in the same cluster. However, this must be explicitly configured. It is possible to set this up such that *only* preferred servers are used. This might circumvent the high availability advantages of the cluster if there are no more preferred servers available.

With some additional configuration, you can create a topology where each server in the cluster will be configured to have an active messaging engine, thus providing workload management as well as failover (Figure 10-3). Note that because messaging engines can run on any server, if one server goes down, both messaging engines will run on the remaining server.

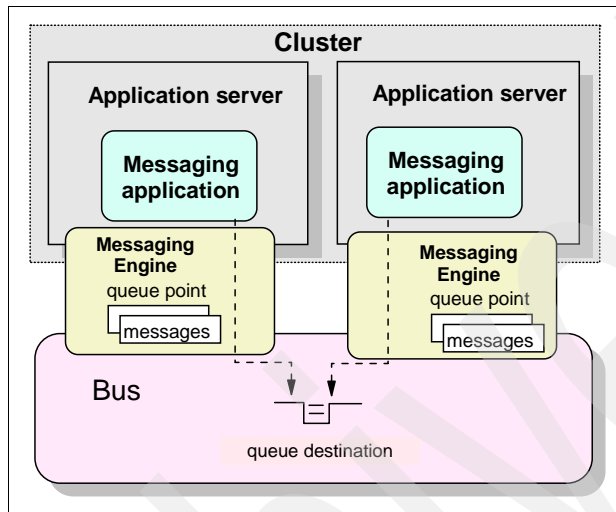


Figure 10-3 Single bus with a cluster member: Workload management

When a queue destination is assigned to the cluster, the queue is partitioned with each messaging engine in the cluster owning a partition of the queue. A message sent to the queue will be assigned to one partition. The messaging engine that owns the partition is responsible for managing the message. This means that requests sent to a destination can be served on any of the messaging engines running on any of the servers in the cluster.

Each of these topologies require special consideration before using them. If you are considering using a cluster topology, refer to *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392, for more information.

One bus, multiple bus members

In this topology, there are multiple non-clustered application servers connected as members of the bus (Figure 10-4). In this topology, most, if not all servers are bus members. Take care to locate the queue points on the same application server as the messaging application that is the primary user of the queue. This will maximize the use of local connections and enhance performance.

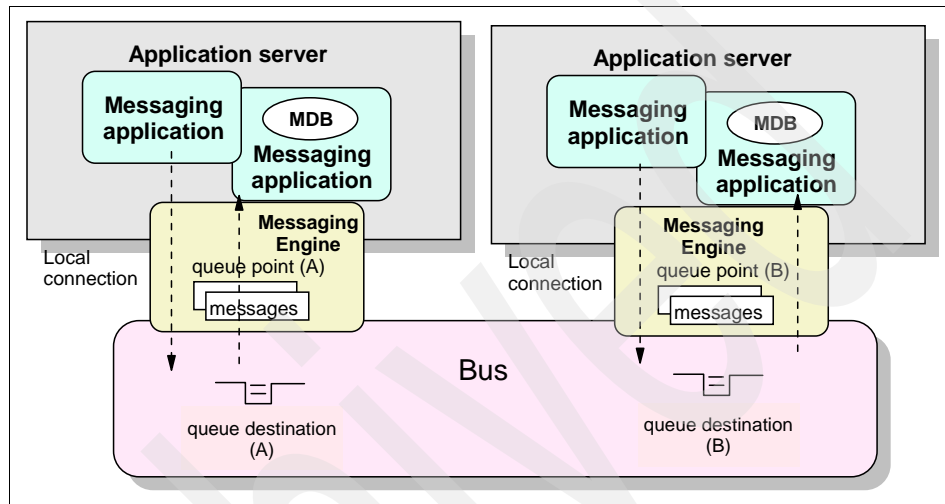


Figure 10-4 Single bus with multiple application server members

Multiple buses

Many scenarios only require relatively simple bus topologies, perhaps 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 span 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 to have separately administered buses in each location. Or perhaps 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.

Figure 10-5 illustrates how a service integration bus can be connected to another service integration bus and to a WebSphere MQ network.

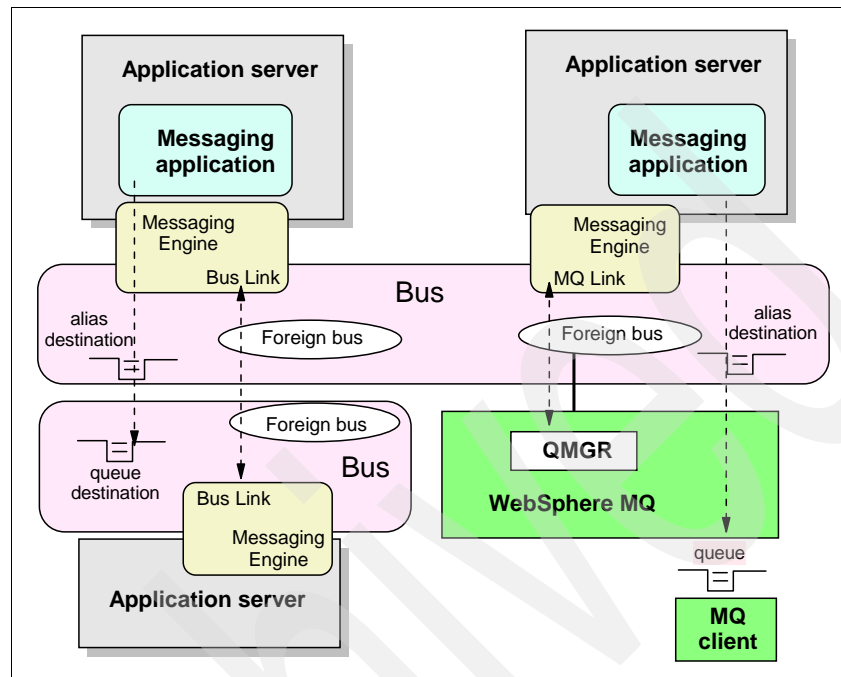


Figure 10-5 Multiple bus scenario

In the case of 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 sends it to an alias destination 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 also important to note that an application cannot consume messages from a destination in a foreign bus.

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 a WebSphere MQ running on z/OS (Figure 10-6). The WebSphere MQ server defines properties for the connection to the queue manager or queue sharing group.

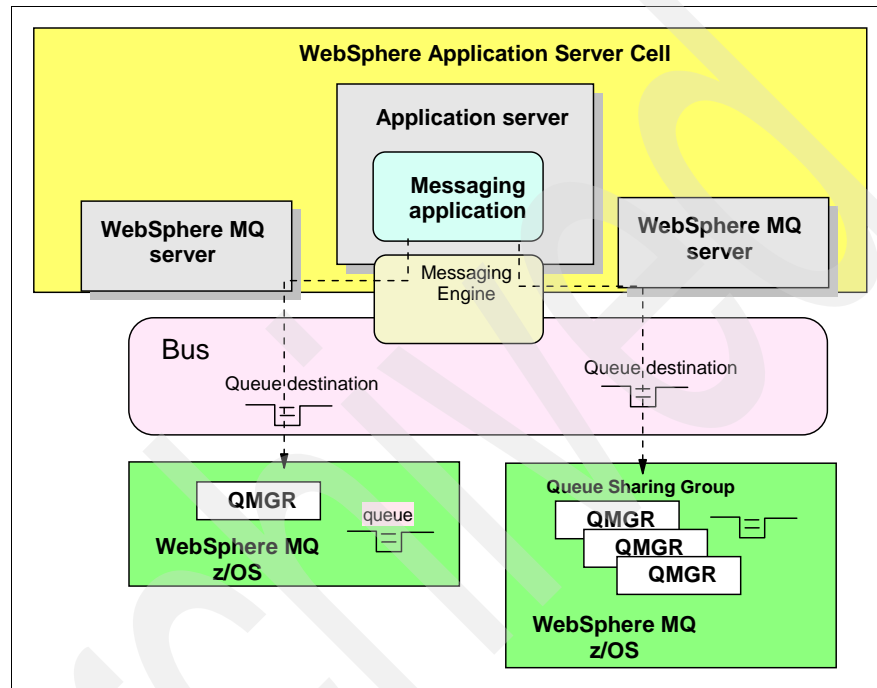


Figure 10-6 Multiple bus scenario

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 shared group is regarded as a mechanism to queue messages for the bus. The WebSphere MQ server is regarded by the WebSphere MQ network as just another MQ client attaching to the queue manager or shared group.

WebSphere MQ server provides the following advantages over a WebSphere MQ link:

- ▶ WebSphere MQ server allows applications to exploit 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 is 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. This allows message beans to be configured to immediately process messages as they arrive on an MQ queue. Similarly, any bus mediations take place immediately upon a message appearing on 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 requiring optimum load balancing.

WebSphere MQ server supports J2EE, JMS, WebSphere Message Queuing Interface (MQI), and the service integration mediations API.

10.6 Messaging features: How secure and reliable is it?

This final chapter describes some of the lower-level details and requirements of messaging. We cover three categories: security, high availability, and reliability. These are important points that must be factored in to any planning. This section contains the following topics:

- ▶ More messaging concepts
- ▶ Planning for security
- ▶ Planning for high availability
- ▶ Planning for reliability

10.6.1 More messaging concepts

We must briefly discuss the following concepts before discussing messaging in more detail.

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 applications running outside of a server with a messaging engine. They are divided into inbound and outbound, and encompass things such as encryption and communication protocols, for example, TCP/IP.

Message stores

At the center of a message engine is a *message store*. This is a repository that allows data (messages, operational data, or both) to be stored in both a permanent and temporary fashion. Permanent means that the data will survive a shutdown of the message engine.

10.6.2 Planning for security

There are two main areas in messaging security. The first is authorization and authentication of users and groups that want to connect to a bus. The second is securing the transportation of the message from source to destination. This is not covered in depth here. For more information, refer to *WebSphere Application Server V6 Security Handbook*, SG24-6316.

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 some sort of external access registry, such as an 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. There is a role called the *bus connector role* to which the user or group must be assigned; otherwise, they might be denied access even if the credentials are valid. Other roles that affect permissions for users and groups include:

- ▶ 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.

Address the following questions:

- ▶ What users or groups, or both, do I need to define or have already been defined?
- ▶ What are the minimum permissions I 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 message engine. The transport chains have attributes such as security encryption (using SSL or HTTPS, for example) and the communication protocol used (TCP/IP, for example).

Encryption is obviously more secure, but can have performance impacts. This is also true for the 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.

The relevant questions here include:

- ▶ What types of messages do I need secured?
- ▶ Where do I need to use encryption, and to what extent?
- ▶ What are the connection requirements (in terms of security) of the party I am trying to communicate with?

10.6.3 Planning for high availability

We describe this to some extent when discussing the choice of messaging topologies (10.5, “Messaging topologies: How can I use messaging?” on page 215), but we briefly recap here.

Application server clustering

An application server only has one messaging engine for each bus of which it is a member. There is no option 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 activates on another server in the cluster.

To ensure that the messaging engine runs on one particular server in the cluster, for example, if you have one primary server and one backup server, or if you want the messaging engine to only run on a small group of servers within the cluster, you must specifically configure this by defining the preferred server for the messaging engine. 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 servers if you have not already.

10.6.4 Planning for reliability

The JMS specification supports two modes of delivery for JMS messages: persistent and non-persistent. The WebSphere administrator can select the mode of delivery on the JMS destination (queue/topic) configuration:

- ▶ Application (persistence is determined by the JMS client)
- ▶ Persistent
- ▶ Nonpersistent

Messages can also have a quality of service 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 using that connection factory, though you can opt to let the reliability settings be set individually at the bus destination.

Each reliability setting has different performance characteristics. The settings are:

- ▶ Best effort nonpersistent
- ▶ Express nonpersistent
- ▶ Reliable nonpersistent
- ▶ Reliable persistent
- ▶ Assured persistent

There is a trade-off between reliability and performance to consider. With increasing reliability levels of a given destination, performance or throughput of that destination is decreased. There is a default setting that is configured when the destination is created, but this can be overridden by message producers and consumers under certain circumstances.

The “Message reliability settings” topic in the WebSphere Application Server Information Center contains a table that outlines what happens to a message under various circumstances depending on delivery mode and reliability setting:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/concepts/cjj9000_.html

For information about how reliability levels are affected when messages flow over an MQ link, see the “Mapping of message delivery options flowing through the WebSphere MQ link” topic in the WebSphere Application Server Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.pmc.nd.doc/ref/rjc0014_.html

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 expectation?

Select a message store type

Another consideration is the message store that each messaging engine employs. This is where the messages are persisted according to the reliability levels of the messages. This, as well as the reliability levels, will directly affect the performance of the messaging engine.

Message stores can be implemented as either flat files (called *file stores*) or as tables inside a database (called *data stores*).

File stores are flat files that can be administered by the local operating system. This is the default type of message store. File stores will generally be faster and cheaper than data stores because of the absence of the database. File stores have no extra licensing fees and much less administration costs, as well as no database administrator.

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. A limited license version of Cloudscape is provided with WebSphere Application Server V6 for this purpose. Data stores might be preferable for larger organizations with an existing database infrastructure and skills.

Both types of message store can be subject to security, such as file system/database encryption and physical security access.

10.7 Planning checklist for messaging

Table 10-1 provides a summary of items to consider as you plan and additional resources that can help you.

Table 10-1 *Planning checklist for Web services*

Planning item
Determine if messaging will be used and how.
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 reliability levels for those destinations.
Determine the type of message data store to use.
Design a security strategy for messaging: <ul style="list-style-type: none">▶ Bus security▶ Transport security
Plan for high availability. If clustering application servers, decide whether to use one messaging engine (high availability) or multiple (workload management).

Resources

For a good overall reference for messaging applications in WebSphere Application Server, refer to *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

We suggest that you have a copy of this book available as you plan your Web services environment.

The WebSphere Application Server Information Center also contains a lot of useful information. For a good entry point to messaging topics, see:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6tech_msg.html

For examples of using messaging in a SOA solution, see *Enabling SOA Using WebSphere Messaging*, SG24-7163.

Archived

Planning for Web services

This chapter describes Web services and what considerations administrators should make when planning for their usage on a WebSphere Application Server V6.1 architecture. It contains the following sections:

- ▶ What are Web services?
- ▶ What is new in V6.1
- ▶ Are Web services something you should use?
- ▶ What do you need to implement Web services?
- ▶ What other Web service considerations are there?
- ▶ Planning checklist for Web services

For detailed information about Web services and WebSphere Application Server, refer to *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257.

11.1 What are Web services?

Web services, as a concept, are an implementation of another concept called *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. SOA is an important trend in the IT community. 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, they can be orchestrated together in flexible business workflows, enabling businesses to 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 and that is the support that WebSphere Application Server V6.1 provides. Any implementation of an SOA, including Web services, must have the following characteristics:

- ▶ Interoperability between different platforms, systems, and programming languages
- ▶ Clear and unambiguous service interface description language
- ▶ Dynamic search and retrieval capabilities of a service at runtime
- ▶ Security

Web services are described as self-contained, modular applications that can be described, published, located, and invoked over a network. More specifically, a Web service can be said to 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 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 leverage existing network infrastructure and Internet standards such as HTTP.
- ▶ Web services are modular. Simple Web services can be chained together, or otherwise grouped into more complex ones, to perform higher-level business functions with little effort.

- ▶ Web services are interoperable across platforms and language-independent. The client and the server can be on different platforms, on different machines, in different countries. There are no restrictions on the language used so long as 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 have long been widely used and understood.
- ▶ Web services are dynamic and loosely coupled. Web services are not tightly coupled, and are easily reconfigured into new ones. Therefore, Web services must be able to be dynamically discovered in an automated fashion. This allows 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. Potentially, this provides a large gain with little effort.

11.2 What is new in V6.1

For those of you already familiar with Web services in WebSphere, WebSphere Application Server V6.1 offers some improvements to the Web services implementation from V6.0:

- ▶ Performance increases in the form of a new JAXP SAX parser, improvements to the SAAJ 1.2 implementation, and support for SOAP/JMS connection caching for clients.
- ▶ Support for the WS Interoperability Basic Security Profile (WS-I BSP) 1.0, addressing common problems that include improved support for enabling interoperability in multivendor Web services solutions.
- ▶ Performance improvement in WS security by supporting the offloading of complex cryptographic operations to specialized hardware or devices. This is not supported for iSeries hardware. The support for the newest and fastest cryptographic devices has also been improved and updated.
- ▶ Support for the WS Transaction Business Agreement (WS-BA) specification, enabling the registration and flow of business agreement protocols between Web service participants.
- ▶ Improved support for the WS Atomic Transactions (WS-AT) specification, removing some limitations present in the previous release. Atomic transaction contexts can now span firewalls and use virtual host names.

- ▶ Updated, but backward compatible support for compliancy with the latest WS-Addressing specification from World Wide Web Consortium (W3C). This provides transport-neutral mechanisms to address Web services and facilitate end-to-end addressing. A limited API is available to WS applications to use the WS-Resource Framework Resource Access Pattern (WSRF-RAP).
- ▶ Support added for the WS-Notification 1.3 specification. This enables Web services to use the publish/subscribe messaging pattern, enabling the typical one-to-many message distribution scenario. Part of this support is the introduction of the Notification Broker Service as a point of separation between producing and consuming notification applications.

11.3 Are Web services something you should use?

There are a number of both business and technical aspects to this question, and it depends on what you want to do with Web services. The following questions represent the types of strategic thinking that needs to happen if you want to provide or use Web services:

- ▶ Do you have business functionality that is common and can be shared?

The typical reason to use a Web service is to save time and effort by reusing existing infrastructure. Over time, this enables the entire IT infrastructure of an enterprise to reduce redundancy and consist of mature, well-tested components. Does your application have this sort of functionality? Can you reduce the complexity of your application by using other Web services?

- ▶ What business functionality do you want to expose to external parties?

The thing to keep in mind here is that you have the option to expose as much or as little of your application as you want. This can range from single business functions exposed as services, to the entire application wrapped up as a single Web service. It 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 functionality 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, generating goodwill and business opportunities, while allowing you the flexibility to alter or replace the back-end logic transparently to the client. The importance of this varies with the type of clients targeted. What do you know about your potential clients? Are your clients internal or external to your enterprise? Are there a limited set of clients?

There are a number of technical issues that might affect your decision on the use of Web services, including:

- ▶ Will your Web services be stateful or stateless?

If you intend to expose your application over the Internet, you will probably be using 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. Multirequest transactions that require a state to be maintained (say for a “shopping cart” or similar functionality) will need to address this shortcoming. This can be done by using messaging middleware based on JMS or other protocols that provide for the maintenance of state. The bottom line is that stateful Web services are something to be wary of. It is best to 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, some of the other newly adopted standards, such as security and transaction workflows, are still in flux with varying levels of maturity. Take care to ensure that only industry adopted standards are used. This might influence your decisions on candidate business functions for Web service enablement. Figure 11-1 on page 236 shows the various Web services standards and their levels of maturity.

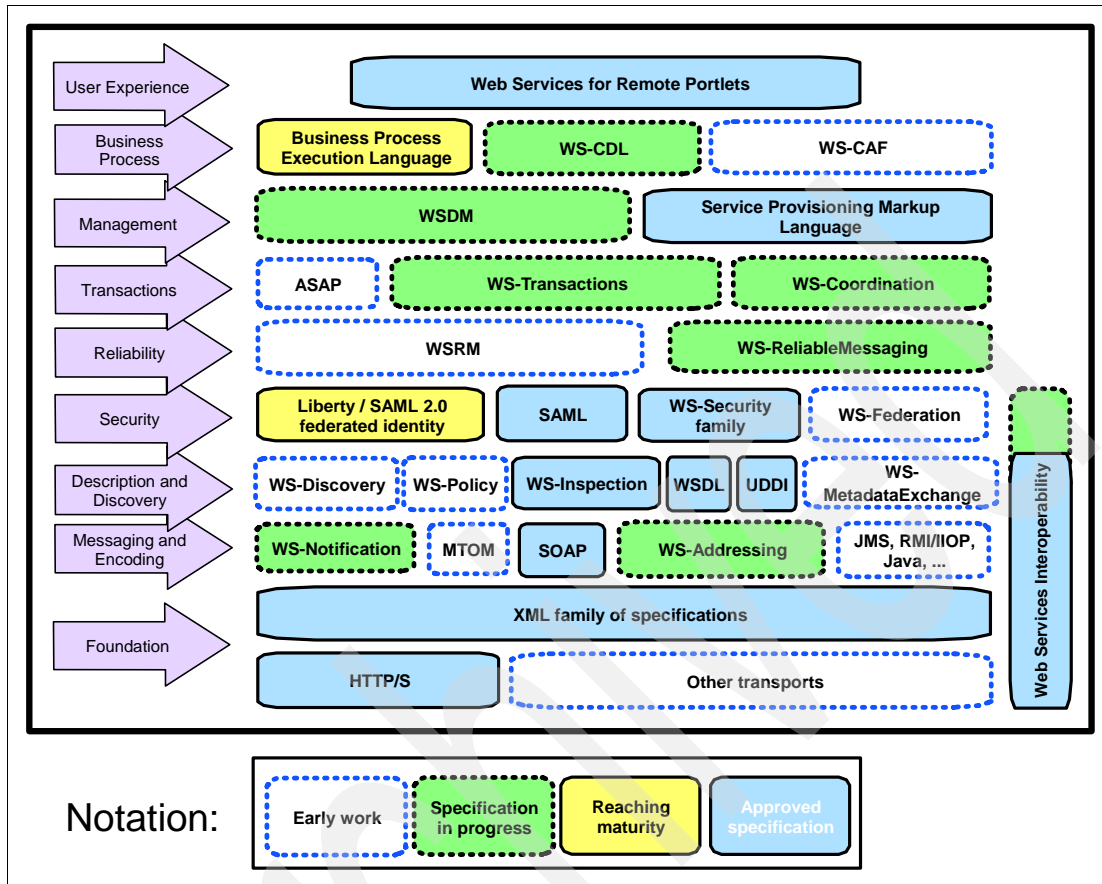


Figure 11-1 Web services protocol stack

- What are you using Web services for?

Web services are designed for 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, it is quite inefficient, especially where it requires the use of parsers and other post-processing.

11.4 What do you need to implement Web services?

Once again, the answer to this question depends on your intended use. The best way to approach this is to outline the basic architecture of an SOA approach

using Web services and how it is implemented with WebSphere Application Server. From this, you can decide which parts will be useful to you. This section describes the following points:

- ▶ What is the basic Web services architecture?
- ▶ How can this architecture be used?
- ▶ How does WebSphere implement this architecture?

11.4.1 What is the basic Web services architecture?

Your basic service-oriented architecture consists of the following three primary components, as shown in Figure 11-2:

- ▶ Service provider (or service producer)
- ▶ Service requestor (or service consumer)
- ▶ Service broker

Each component can also act as one of the two other components. For example, if a service provider needs some more information that it can only acquire from some other service, it acts as a service requestor while still serving the original request. Figure 11-2 shows the operations each SOA component can perform.

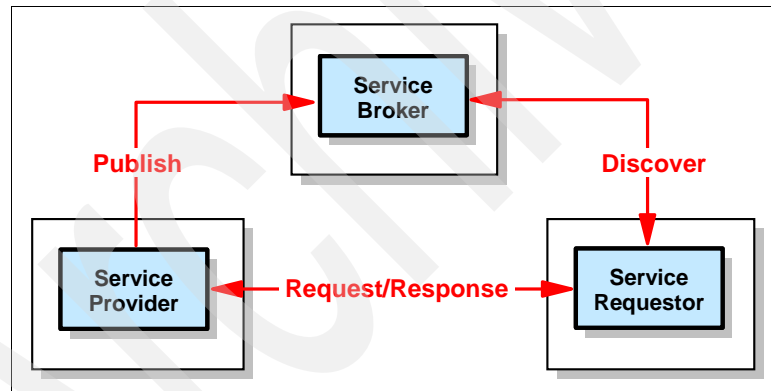


Figure 11-2 SOA components and operations

The components perform the following actions:

- ▶ The *service provider* creates a service and 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 using various find operations and then binds to the service provider in order to invoke one of its services. Another name for the service requestor is the *service consumer*. The terms are interchangeable.

- ▶ The *service broker* (also known as *service registry*) 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.

Before we go on to a Web services-specific view of the architecture, the following terms need a brief explanation:

- XML** Extensible Markup Language is a generic language that can be used to describe any kind of content in a structured way, separated from its presentation to a specific device.
- SOAP** SOAP is a network, transport, and programming language and platform-neutral protocol that enables a client to call a remote service. The message format is XML.
- WSDL** Web Services Description Language is an XML-based interface and implementation description language. The service provider uses a WSDL document in order to specify the operations a Web service provides and the parameters and data types of these operations. A WSDL document also contains the service access information.
- WSIL** Web Services Inspection Language is an XML-based specification about how to locate Web services without the necessity of using UDDI. However, WSIL can be also used together with UDDI, and does not necessarily replace it.
- UDDI** Universal Description, Discovery, and Integration is both a client-side API and a SOAP-based server implementation that can be used to store and retrieve information about service providers and Web services.

Figure 11-3 is a lower-level view of an SOA architecture, but now showing some specific components and technologies. The UDDI and WSIL, separately or together, become the service broker.

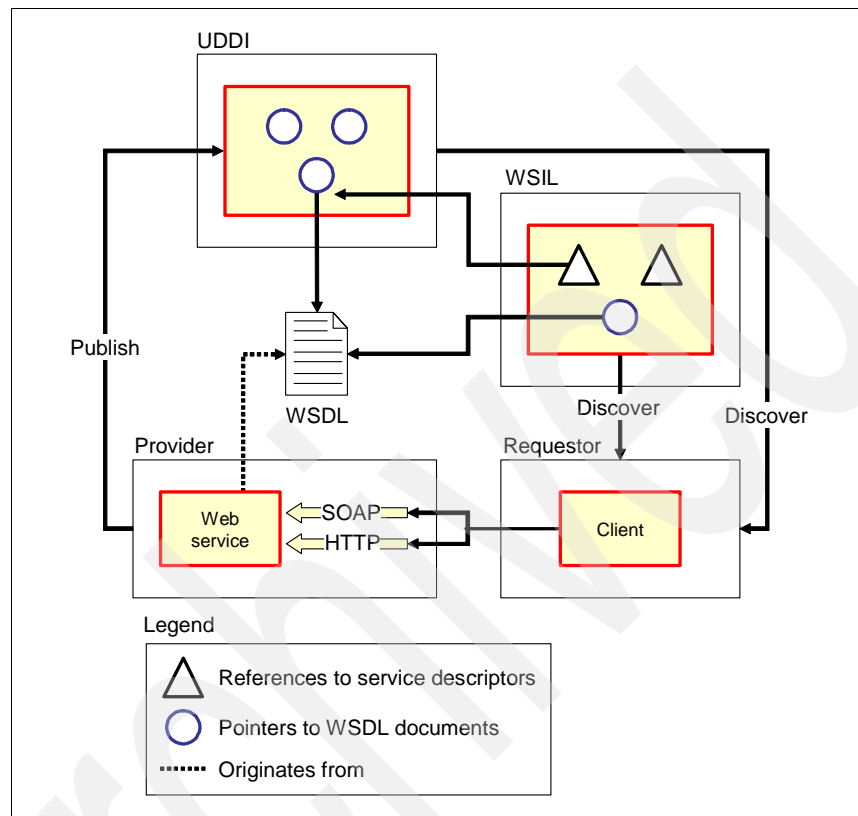


Figure 11-3 Main building blocks in an SOA approach based on Web services

11.4.2 How can this architecture be used?

Here we consider the common message exchange patterns (sometimes referred to as interaction patterns) that might be employed. These all use the previously discussed Web services architecture; however, some of these might have a bearing on the type of transport used and whether or not a Web service should be used at all.

We also look at 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 discusses some of the common message exchange patterns in the context of Web services and considerations for their use. The message exchange patterns are:

- ▶ One-way
- ▶ Asynchronous two-way
- ▶ Request-response
- ▶ Workflow-oriented
- ▶ Publish-subscribe
- ▶ Composite

For more information about selecting and applying IBM Patterns for e-business to a Web services or SOA project, refer to *Patterns: Service-Oriented Architecture and Web Services*, SG24-6303.

One-way

In this very simple message exchange pattern, messages are pushed in one direction only. The source does not care whether the destination accepts the message (with or without error conditions). The service provider (service producer) implements a Web service to which the requestor (or consumer) can send messages (Figure 11-4). This 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).

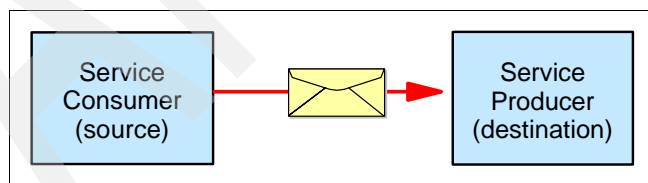


Figure 11-4 One-way message exchange pattern

Asynchronous two-way

In this message exchange pattern (Figure 11-5 on page 241), 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 provider component has to relate a message it receives to the corresponding message that was sent by the service 1 requestor component.

Technically, this message exchange pattern is the same as one-way with the additional requirement that there has to be a mechanism to associate response messages with their corresponding request message. This can be done at the application level or using SOAP protocol.

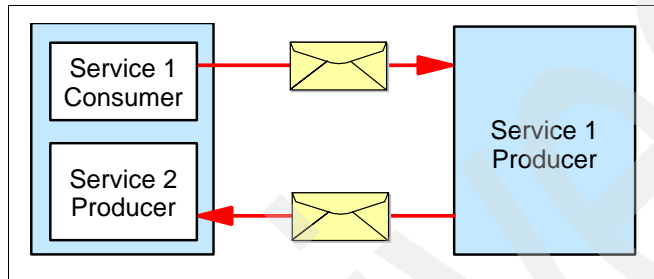


Figure 11-5 Asynchronous two-way message exchange pattern

Request-response

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 11-6). 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), alternative solutions might have to be sought.

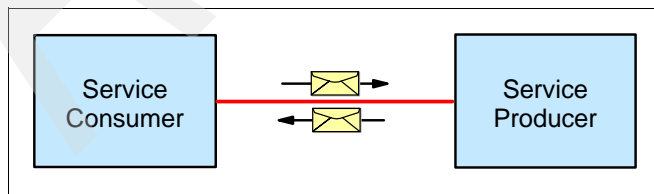


Figure 11-6 Request-response message exchange pattern

There are numerous examples of this message exchange pattern, for example, requesting an account balance on a bank account.

Workflow-oriented

A workflow message exchange pattern can be used 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. Each Web service plays a specific role in the workflow (Figure 11-7).

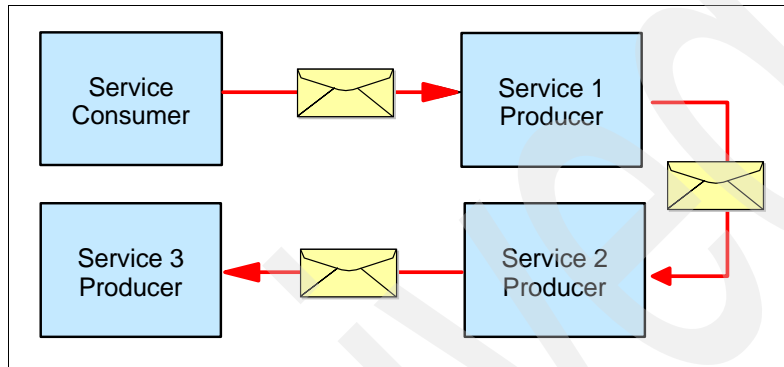


Figure 11-7 Workflow-oriented message exchange pattern

This message exchange pattern is inflexible and does not facilitate reuse—the workflow or *choreography* has been built into each of the Web services, and the individual Web services can no longer be self-contained.

Publish-subscribe

The publish-subscribe message exchange pattern, also known as the event-based or notification-based pattern, is generally used in situations where information is being *pushed* out to one or more parties (Figure 11-8 on page 243).

Implementation of this pattern at the application level is one possible architecture. Alternatively, the service 1 provider component can publish SOAP messages to a messaging infrastructure that supports the publish-subscribe paradigm.

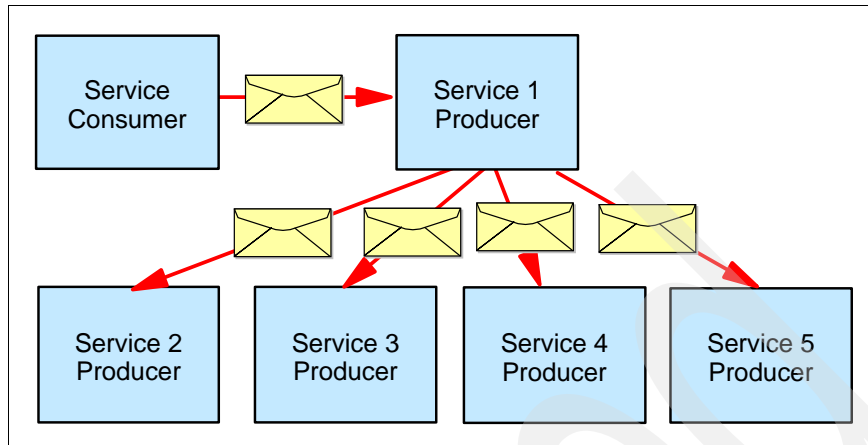


Figure 11-8 Publish-subscribe message exchange pattern

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

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 will generally also include business logic (Figure 11-9 on page 244).

This 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 using a business process choreography engine.

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 has been 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, service 2 to verify the credit standing of the customer, and service 3 to request delivery of the parts to the customer. Some of these services might be internal to the company and others might be external.

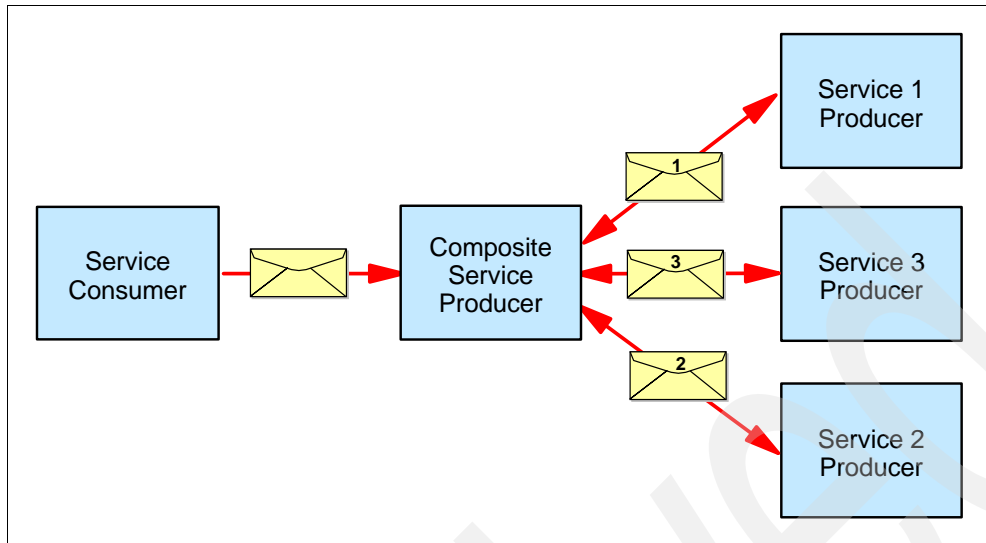


Figure 11-9 Composite message exchange pattern

SOAP processing model

At an application level, a typical Web service interaction occurs between a service consumer and a service provider, optionally with a lookup to a service registry. However, at the infrastructure level, additional intermediary SOAP nodes might be involved in the interaction (Figure 11-10).

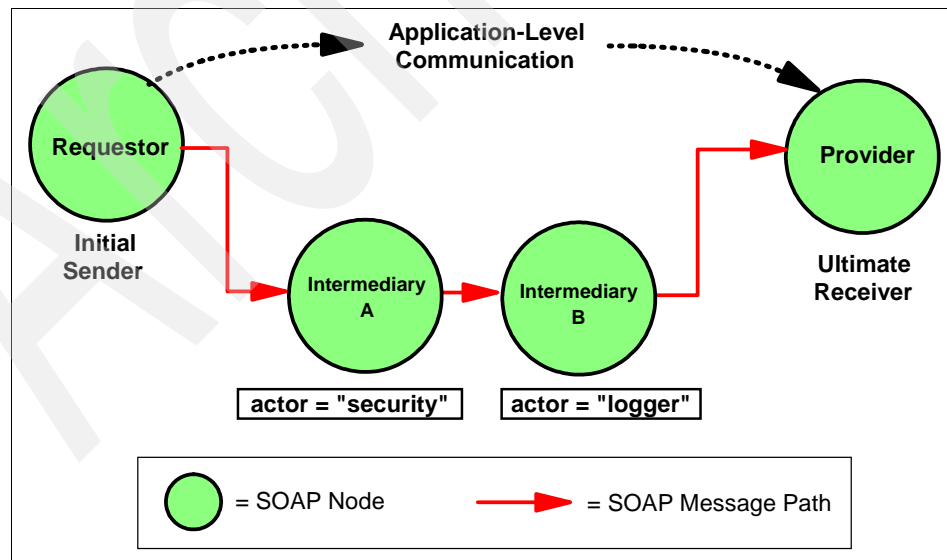


Figure 11-10 SOAP processing model

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. There might be new requirements 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 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, but regardless of where it is placed, it can provide some or all of the following functions:

- ▶ Provides automatic publishing of WSDL files to an external UDDI or WSIL registry
- ▶ Provides automatic protocol/transport mappings
- ▶ Provides security functions
- ▶ Provides mediation of message structure
- ▶ Implements a proxy server for Web service communications through a firewall
- ▶ Provides auditing of SOAP messages
- ▶ Provides operational management and reporting of published interfaces
- ▶ Provides Web service threat detection and defense

11.4.3 How does WebSphere implement this architecture?

As stated previously, you do not need very much to implement a Web service. There is support for SOAP (in the form of several types of SOAP engines) and for XML built into the J2EE 1.4 standard, which WebSphere Application Server V6.1 supports. However, this does not represent a true SOA, rather it will be simply a collection of point-to-point services. To set up an infrastructure that properly implements an SOA with all its characteristics of loose coupling, component reuse, and service composition, certain components must exist at an organizational or enterprise level. Figure 11-11 on page 246 shows these components.

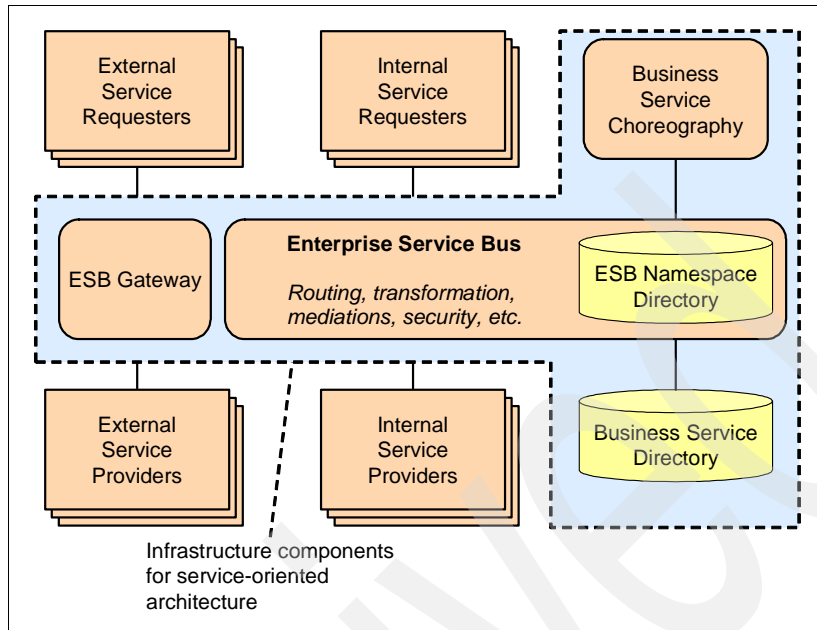


Figure 11-11 Components of a service-oriented architecture

The most important component is the enterprise service bus (ESB). IBM has two ESB products: WebSphere Enterprise Service Bus and WebSphere Message Broker. However, the service integration technologies in WebSphere Application Server V6.1 can also provide some basic ESB function.

The primary features describing an ESB provided by the service integration bus in WebSphere Application Server V6.1 are:

- ▶ Communication middleware supporting a variety of communication paradigms, platforms, and protocols
- ▶ Support for *quality of service* (QoS) characteristics such as security, guaranteed delivery, performance, and transactions
- ▶ Message format transformation and transport protocol conversion

Other components include:

- ▶ The service directory (such as a UDDI, or WSIL repository) that is used for brokering of services and an organization-level WSDL repository (stored as Service Data Objects or SDOs). These are part of the J2EE 1.4 standard, and as such, come with WebSphere Application Server V6.1.
- ▶ The ESB gateway (or Web service gateway), which is an optional component, the major function of which is the enablement of internal services to be exposed to the Internet.

Note: The Web services gateway only comes standard with WebSphere Application Server V6.1 Network Deployment.

- ▶ Optional business service choreography tools to compose workflows from individual services. These are external products to WebSphere Application Server V6.1 and are not described here.

Service integration bus

You know what an SOA is and how it is implemented in WebSphere Application Server V6.1 by a service integration bus, but why use a bus at all? There are a number of advantages that apply both to the application and to the enterprise at large. The advantages include:

- ▶ Securely externalizing existing applications
The bus can be used to expose existing applications as Web services regardless of the implementation details of the application. This enables the applications to 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
After the bus, and optionally a UDDI registry and a Web service gateway, is set up for use by Web services, any application that is Web service-enabled can reuse the infrastructure.
- ▶ Messaging support
The bus is built around support for JMS. This allows 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.
- ▶ Protocol transformation
If your application has existing services implemented in one protocol, the bus can transform that into another, entirely different protocol. For example, a SOAP/JMS internal service can only be exposed to the Internet through a SOAP/HTTP protocol.
- ▶ Support for standards
The bus is part of the J2EE 1.4 implementation and thus supports the major Web services standards that are also part of J2EE 1.4. These include WS-I Basic Profile 1.1, JAX-RPC (JSR-101) 1.1, UDDI V3, WS-I Security, and WS-Transaction. This enables businesses to build flexible and interoperable solutions.

- ▶ 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.

UDDI registries

Universal Description, Discovery, and Integration (UDDI) is a specification that defines a way to store and retrieve information about a business and its technical interfaces, in our case, Web services.

A UDDI registry makes it possible to discover what technical programming interfaces 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 addresses a number of business problems. First, it 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.

Public or private?

One type of implementation is the Business Registry. This is a group of Web-based UDDI nodes, which together form a *public* UDDI registry. These nodes are run on separate sites by several companies (including IBM and Microsoft) and can be used by anyone who wants to make information available about a business or entity, as well as anyone who wants to find that information.

However, there are a couple of problems with public registries. First, companies often do not want to show all of their interfaces to the whole world, which invites the whole world to try to communicate with their service with unknown and possibly malicious intent. Secondly, because the registry is accessible by anyone, there is often inaccurate, obsolete, wrong, or misleading information in there. There are no expiration dates for published information, nor any quality review mechanisms. Given that the users of the registry are often automated processes and not humans with the intuitive ability to separate good and bad content, this can be a severe problem.

In this type of situation, companies can opt for their own private or protected registries. A totally private UDDI registry can be placed behind the firewall for the internal use of the organization only. A protected registry can be a public registry

that is managed by the organization that controls access to that registry to parties that have been previously screened.

Private registries allow 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 one), successful hit rates for clients dynamically searching it increase dramatically.

Web services gateway

Web services gateway functionality enables users to 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 the Network Deployment release of WebSphere Application Server. By using the gateway, it is possible for a Web services client to access an external Web service 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 has the ability to create proxy services and administer handlers for services it manages, providing infrastructure-level facilities for security and logging among others.

Some of the benefits of using the gateway are:

- ▶ A gateway service is located 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 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 What other Web service considerations are there?

There are a couple of minor Web service considerations worth mentioning in closing. These will probably not influence your decision to use Web services or not, but need to be planned for.

- ▶ What are the options for Web service security?
- ▶ How can Web service performance be improved?

11.5.1 What are the options for Web service security?

The Web service security standard WS-Security is still evolving, and some of the responsibility for implementing security is in the hands of the application developer. However, there are a number of things that can be configured with the bus to enforce security for a Web service:

- ▶ WS-Security configuration and binding information specifies the level of security 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 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.

Note that, as always, the more security you have, the more performance is likely to suffer.

11.5.2 How can Web service performance be improved?

Unfortunately, performance of Web services is still poor as compared to other distributed computing technologies. The main problem is the trade-off between performance and interoperability. Specifically, this means the use of XML encoding (marshalling and demarshalling) for SOAP/HTTP bound Web services.

However, for HTTP and HTTPS-bound Web services, there is the concept of Web service dynamic caching. This 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 which service operations are cachable. Not all of them should be, for example, dynamic or sensitive data. This can be a very complex task depending on the size of the application and the number of operations exposed.

Over a slow network, client-side caching can be especially beneficial. For further information about the dynamic cache service concept and functions, refer to the WebSphere Application Server Information Center.

11.6 Planning checklist for Web services

Table 11-1 provides a summary of items to consider as you plan and additional resources that can help you.

Table 11-1 *Planning checklist for Web services*

Planning item
Determine if Web services will be used and how.
Determine if a Web services gateway will be required.
Determine if a UDDI service will be used. If so, decide whether you will subscribe to a public UDDI service or set up a private UDDI.
Determine how Web service clients will call providers (directly, through the service integration bus, or through an ESB).
Design a security strategy for Web services: <ul style="list-style-type: none">▶ WS-Security for applications▶ Transport-level security▶ HTTP basic authentication
Determine if you will use Web service dynamic caching.

Resources

For a good overall reference for developing and deploying Web services in WebSphere Application Server, refer to *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257.

We suggest that you have a copy of this book available as you plan your Web services environment.

The WebSphere Application Server Information Center also contains a lot of useful information. For a good entry point to Web services topics, see:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6tech_wbs.html

For examples of using Web services in a SOA solution, refer to *Patterns: SOA Foundation Service Creation Scenario*, SG24-7240.

For examples of using Web services through an ESB product, such as WebSphere Enterprise Service Bus, see *Patterns: SOA Foundation Service Connectivity Scenario*, SG24-7228.

Archived

Planning for security

WebSphere Application Server provides security infrastructure and mechanisms to protect sensitive resources and to address enterprise end-to-end security requirements.

This chapter cannot possibly cover the complex aspects that are inherent in planning security for a WebSphere Application Server installation, but it describes the concepts and gives you a good look at what you need to consider.

This chapter contains the following sections:

- ▶ What is new in V6.1
- ▶ Why you need security and how it works in WebSphere
- ▶ Security fundamentals on WebSphere
- ▶ J2EE security
- ▶ Planning for security
- ▶ Planning checklist for security

For detailed information about WebSphere Application Server security, see *WebSphere Application Server V6 Security Handbook*, SG24-6316.

12.1 What is new in V6.1

This section outlines some of the major changes from WebSphere Application Server V6.0 to WebSphere Application Server V6.1. A complete list of new and improved items for installers is in the Information Center under the “What is new in this release topic” for each WebSphere Application Server package.

The following security-related items are new for WebSphere Application Server V6.1:

- ▶ Administrative security can be enabled out of the box. Access to the administrative system and its data is now protected by default.
- ▶ Simplified security configuration and administration. The administrative console security panels are simplified, and new wizards and a configuration reporting tool are provided.
- ▶ Automatically generated server IDs. You no longer need to specify a server user ID and password during security configuration, unless using a mixed cell environment.
- ▶ Federate various repositories, so you can manage them as one. Inclusion of virtual member manager in this release provides a single model for managing organizational entities. You can configure a realm that consists of identities in the file-based repository that is built into the system in one or more external repositories or in both the built-in, file-based repository and in one or more external repositories.
- ▶ WebSphere key and certificate management has been simplified.
- ▶ Interoperability with other vendors of WS-Security. The product now supports the WS-I Basic Security Profile 1.0, which promotes interoperability by addressing the most common problems encountered from implementation experience to date.
- ▶ SPNEGO support for single sign-on authentication through a Microsoft Windows desktop.
- ▶ Separate Web authentication and authorization. Now, Web authentication can be performed with or without Web authorization. A Web client's authenticated identity is available whether or not Web authorization is required.
- ▶ Enhanced control over Web authentication behavior.
- ▶ Portlet URL security, enabling direct access to portlet URLs just like servlets.
- ▶ Larger variety of administrative roles.

- ▶ Fine-grained administrative role authorization. In prior releases, users granted administrative roles could administer all of the resource instances under the cell. Now, the product is more fine-grained, meaning that access can be granted to each user per resource instance.
- ▶ Hardware cryptographic device support for Web services security.

The following security-related items are new for WebSphere Application Server for z/OS V6.1 systems:

- ▶ WebSphere Application Server exploits the new mixed-case password option for RACF.
- ▶ Sync to OS Thread control enhancements. In addition to the application and the configuration specifying the desire to use Sync to OS Thread, the RACF administrator must also define a resource rule in order for Sync to OS Thread to operate. A new FACILITY class profile must be defined to allow or disallow the use of Sync to OS Thread. Also, an optional SURROGAT class profile can be used to further refine the use of Sync to OS Thread to particular authenticated users.
- ▶ Enabling trusted applications. A new FACILITY class profile must be defined to enable trust applications. WebSphere Applications Server checks this FACILITY class profile during initialization to ensure that only authorized trusted applications are enabled. This new FACILITY class profile expands the RACF administrator role in ensuring that only authorized trusted applications are enabled.

Table 12-1 on page 256 describes some of the common security issues reported in previous WebSphere Application Server versions that are addressed in Version 6.1. It is a good starting point to understand what is new in V6.1 and how you can improve your WebSphere configuration with the new security features.

The source of this information is *WebSphere Application Server V6.1: What's new in security?* by Keys Botzum at:

http://www.ibm.com/developerworks/websphere/library/techarticles/0606_botzum/0606_botzum.html

Table 12-1 V6.1 security enhancements

Problem	Solution on V6.1
<p>Too hard to secure WebSphere Application Server.</p>	<p>Many basic security setup steps are now made by default. For example:</p> <ul style="list-style-type: none"> ▶ Administrative security is enabled automatically during installation. ▶ By default, all internal transports are authenticated and most of them are encrypted by default, too. ▶ A cell-specific set of keys is created automatically and the default encryption keys are eliminated. ▶ JNDI is ready-only by default to all instead of read/write. ▶ Messaging limits connections to only authenticated users granted the bus connect role by default. All Authenticated no longer has that role by default.
<p>Too hard to manage certificates, private keys, and encryption keys.</p>	<p>The management of trust stores and key stores is now handled as a first class construct. There are several enhancements to improve usability. For example:</p> <ul style="list-style-type: none"> ▶ Clients support prompting (such as SSH) for adding certificates to the client trust store when contacting a server not previously accessed (this can be disabled if desired). ▶ When configuring an SSL endpoint, the admin client can query the server and automatically import the server's signing certificate (with administrative approval, of course). ▶ The Web server plug-in configuration will automatically generate a plugin-key.kdb file containing a self-signed certificate that is trusted by the WebSphere Application Servers. This greatly simplifies one of the most difficult SSL configurations. ▶ iKeyman usage is largely eliminated. Using the admin tools, you can generate certificates, generate certificate requests, import keys and certificates, manage certificates and keys, and even share them across the cell. ▶ Key management applies to more than SSL certificates. The same infrastructure manages the keys used to encrypt LTPA tokens. You can even manage your own custom keys using the console, and applications can access them using IBM-defined APIs. Applications can now be freed from that burden.

Problem	Solution on V6.1
<p>Too hard to manage certificates, private keys, and encryption keys (continuation).</p>	<ul style="list-style-type: none"> ▶ There are now programmatic APIs for applications to obtain URLStreamHandlers, SSLContext instances, and SSLSocketFactories, based on the WebSphere Application Server-managed SSL configuration. You can also set SSL properties on the thread to be used for SSL connections that occur on that thread. ▶ Support for custom JSSE trust and key managers enables more control of the SSL handshake. ▶ The ability to associate an SSL configuration with specific target hosts and ports. Previously, a single static SSL configuration had to handle all outbound connections for any given protocol. This enables special configurations for targets that have unique handshake requirements. ▶ Certificate expiration is managed. Where possible, new keys are generated automatically prior to expiration. When this is impossible, notifications are sent through the serious event stream and, optionally, by e-mail. Expiration monitoring is on by default. ▶ LTPA encryption keys are automatically changed at regular intervals. To avoid outages, multiple key versions are simultaneously supported. ▶ Better support is now provided for hardware encryption and hardware key storage.
<p>My registry infrastructure is not supported by WebSphere.</p>	<p>WebSphere V6.1 supports a much more complicated registry configuration out of the box. Of course, it will not solve all problems and you might still need a custom registry but it helps in many cases. For example:</p> <ul style="list-style-type: none"> ▶ File registry is supported, which can be useful for small applications with just few users or during early prototypes and development. You can now manage all of your users and groups in a file. There are admin tools for managing these users and the files are automatically replicated throughout the cell. ▶ More than one LDAP directory can be combined together into one logical registry. Now, if your users are spread across multiple directories, you can combine them into one registry for WebSphere Application Server. Just be aware that user IDs must be <i>unique</i>. ▶ LDAP failover is supported directly. Now, you can list multiple LDAP server replicas by IP address or host name. The application servers will automatically fail over to backup servers if the primary fail.

Problem	Solution on V6.1
<p>I cannot isolate the administrators from each other.</p>	<p>With V6.1, administrative authority can now be separated at a finer-grained level. It is referred as fine-grained administrative security.</p> <p>For more information, refer to: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/ae/ae/csec_fineg_admsec.html</p>
<p>Too hard to set up the messaging bus to use security.</p>	<p>Configuring a messaging bus to use secure transport has become much easier. You do not have to disable the unsecure transports and manually ensure that every client used the secure transports. It is now all configured automatically and is easily controllable through the admin console.</p> <p>This is done through a new concept known as permitted chains, which controls what type of transport chains can be used (for example, only those using SSL).</p> <p>For more information, refer to: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.iseries.doc/concepts/cjr0490_.html</p>
<p>I cannot use single sign-on from my Windows to my intranet application.</p>	<p>WebSphere Application Server V6.1 supports a SPNEGO trust association interceptor (TAI) that allows the Kerberos credential from a Windows desktop to be sent from the browser to the WebSphere Application Server and then used as the identity for access to WebSphere resources. A slightly different version of the SPNEGO TAI is available from IBM Software Services for WebSphere for previous releases, but with V6.1, we now have a fully supported product solution.</p> <p>For more information, refer to: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.iseries.doc/info/series/ae/csec_SPNEGO_overview.html</p>

Problem	Solution on V6.1
My Web applications mix protected and unprotected URIs, but I still need to know the user's identity from unprotected URIs.	<p>With WebSphere Application Server V6.1, this behavior is now supported. You can optionally configure security to persist identity information across requests. Now, if a user accesses a protected URI and then an unprotected URI, their identity will be available. Anonymous users can, of course, directly access unprotected URIs as usual but you can turn that off as well through another option.</p> <p>For more information, refer to: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/ae/ae/usec_webauth.html</p>

12.2 Why you need security and how it works in WebSphere

The fundamental reason for having security is to keep intruders out and to prevent unauthorized users from gaining access to your systems. There are many reasons for intruders to gain access to your systems. They might be hackers, competitors, or even your own employees. Intruders might want to gain access to information they should not have or to alter the behavior of your systems.

When planning security for WebSphere Application Server, it is important to have a comprehensive security policy that coordinates neatly with the overall environment security. WebSphere Application Server adheres to standard J2EE specifications as closely as possible and integrates easily with existing security systems. There is no single solution for security concerns. However, proper planning and diligence can keep systems functional and minimize the impact on business.

Security can be divided into the following areas:

- ▶ Physical security

Physical security encompasses the area where the environment is located and the major concern is access to the site. Commonly, such areas are physically secured and access is limited to a small number of individuals.

- ▶ Logical security

Logical security is the software used to protect the various systems. Password protection is the most common logical security but can also include firewalls and protective devices.

- ▶ **Application security**

Application security is the use WebSphere Application Server technologies to secure the application from intrusion. WebSphere Application Server provides many plug-in points to integrate with enterprise software components to provide end-to-end security.

Security policy

There are a number of key principles of a security policy:

- ▶ **Identify key assets**

Identify critical areas of business and those assets that host them. By identifying those key assets, you can adopt the methods that are best for the environment and create an effective security policy.

- ▶ **Identify vulnerabilities**

Complete a comprehensive assessment of the environment to identify all the threats and vulnerabilities. Examine each and every area of the environment to find any way the system can be compromised. It is very important to be thorough. Remember to examine the three types of security mentioned earlier: physical, logical, and application. This can be a very resource-intensive activity but it is crucial to the security of the environment.

- ▶ **Identify acceptable risk**

After completing a vulnerability assessment, the acceptable risk must be determined. In most instances, this will be a cost issue. To completely secure an environment would be extremely expensive, so compromises have to be made to complete the security policy. In most cases, the most cost effective method to meet the required security level will be used. For example, in a system that contains mission-critical data for a company, the most advanced firewall available is necessary. However, on a test system with no external access, the appropriate security level can be met with minimal or no firewalls.

- ▶ **Use layered security model**

In complex systems, it is important to have multiple layers of security to ensure the overall safety of the environment. A layered security model plans for expected loss and minimizes the impact. It also ensures that all components are protected, from the user to the back-end data systems, and that a failure in any one component does not impact the whole environment.

Security configuration

After creating the security policy, you must implement it. Implement steps to configure the physical, logical, and application security as recommended in the security policy. In many cases, recommended configurations will not be possible

and, once again, compromises must be made. Note any discrepancies so that they can be addressed and possibly remedied at a future date.

Security reviews

A timely and regular review of the security policy is essential to its effectiveness. As the environment evolves over time, the security policy must also evolve. Regular appraisals of the security policy, including key assets, vulnerability assessment, and acceptable risk, are needed to maintain the expected level of security.

WebSphere Application Server security

When you access information on the Internet, you connect through Web servers and product servers to WebSphere Application Server, and in the end, to the enterprise data at the back end. WebSphere Application Server provides you a set of features to help you to secure your systems and manage all resources.

Figure 12-1 illustrates the building blocks that make up the operating environment for security in WebSphere Application Server.

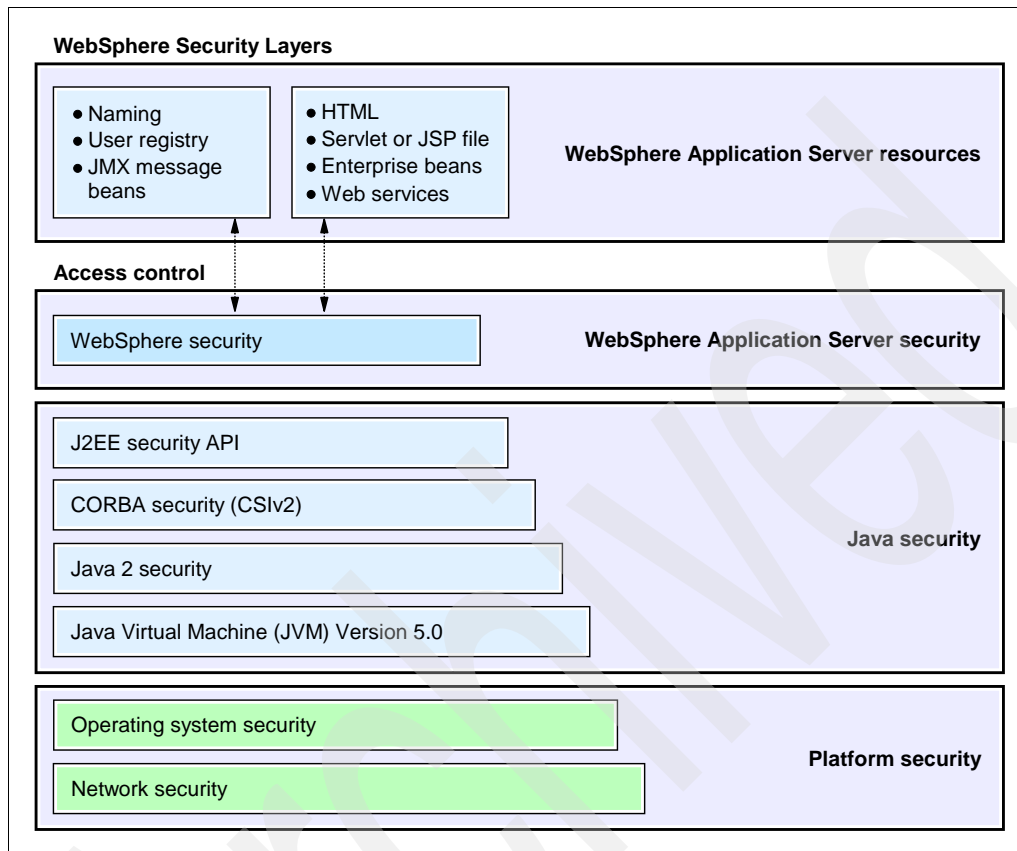


Figure 12-1 WebSphere Application Server security layers

These building blocks consist of:

- ▶ WebSphere Application Server security

WebSphere Application Server security enforces security policies and services in a unified manner on access to Web resources, enterprise beans, and JMX administrative resources. It consists of WebSphere Application Server security technologies and features to support the needs of a secure enterprise environment.

- ▶ Java security

- J2EE security API

The security collaborator enforces J2EE-based security policies and supports J2EE security APIs.

- CSiv2 (CORBA security)

Any calls made among secure Object Request Brokers (ORBs) are invoked over the Common Secure Interoperability Version 2 (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. CSiv2 is an IOP-based, three-tiered, security protocol that is developed by the Object Management Group (OMG). This protocol provides message protection, interoperable authentication, and delegation. The three layers include a base transport security layer, a supplemental client authentication layer, and a security attribute layer.

Note: For backward compatibility, WebSphere Application Server supports the Secure Authentication Service (SAS) security protocol, which was used in prior releases of WebSphere Application Server and other IBM products.

- Java 2 security

The Java 2 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.

- Java virtual machine (JVM) 5.0

The JVM security model provides a layer of security above the operating system layer. For example, JVM security protects the memory from unrestricted access, creates exceptions when errors occur within a thread, and defines array types.

- ▶ Platform security

- Operating system security

The security infrastructure of the underlying operating system provides certain security services for WebSphere Application Server. These services include the file system security support that secures sensitive files in the product installation for WebSphere Application Server.

The administrator can configure WebSphere Application Server to obtain authentication information directly from the operating system user registry. When you select the local operating system as a registry on z/OS, SAF works in conjunction with the user registry to authorize applications to run on the server.

- Network security

The network security layers provide transport level authentication and message integrity and confidentiality. You can configure the communication between separate application servers to use SSL. Additionally, you can use IP security and virtual private network (VPN) for added message protection.

Note: WebSphere Application Server for z/OS provides SystemSSL for communication using the Internet. SystemSSL is composed of the Secure Sockets Layer (SSL) and Transport Layer Security (TLS), which enable secure file transfer by providing data privacy and message integrity.

12.3 Security fundamentals on WebSphere

This section discusses some fundamental security concepts that WebSphere Application Server supports.

12.3.1 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 keys, encryption keys), or biometric (fingerprints, retina scan, DNA, and so forth).

The authentication mechanism in WebSphere Application Server typically collaborates closely with a user registry. 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.

Although this product 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:

- ▶ Simple WebSphere Authentication Mechanism (SWAM)
- ▶ Lightweight Third Party Authentication (LTPA)

Note: LTPA is the only authentication mechanism available in a Network Deployment environment.

SWAM is deprecated in WebSphere Application Server Version 6.1 and will be removed in a future release. If you need more information about SWAM, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/rsec_swam.html

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. This support permits LTPA to encrypt, digitally sign, and securely transmit authentication-related data, and later decrypt and verify the signature.

Application servers distributed in multiple nodes and cells can securely communicate using this protocol. It also 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 getting prompted.

When using LTPA, a token is created with the user information and an expiration time and is 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 not, LTPA tokens appear 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 receiving servers share the same keys as the originating server, the token can be decrypted to obtain the user information, which then is validated to make sure that it has not expired and that the user information in the token is valid in its registry. On successful validation, the resources in the receiving servers are accessible after the authorization check. All of the WebSphere Application Server processes in a cell (deployment manager, node agents, application servers) share the same 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, the exported keys are encrypted with a user-defined password. This same password is needed when importing the keys into another cell.

Note: When security is enabled during profile creation time, LTPA is configured by default.

When security is enabled for the first time with LTPA, configuring LTPA is normally the initial step performed. LTPA requires that the configured user registry be a centrally shared repository, such as an LDAP or a Windows domain type registry, so that users and groups are the same regardless of the machine.

Lightweight Directory Access Protocol (LDAP)

LDAP is a directory service, not a database. The information in the LDAP directory is descriptive and attribute-based. LDAP users generally read the information more often than they change it. The LDAP model is based on entries that are referred to as objects. Each entry consists of one or more attributes such as a name or address and a type. The types typically consist of mnemonic strings, such as *cn* for common name or *mail* for e-mail address. Each directory entry also has a special attribute called *objectClass*. This attribute controls which attributes are required and allowed in each entry.

Examples of LDAP servers include Tivoli Directory Server, Lotus Domino Enterprise Server, Sun ONE Directory Server, and Microsoft Active Directory Server.

User registry

The information about users and groups reside in a *user registry*. In WebSphere Application Server, a user registry authenticates a user and retrieves information about users and groups to perform security-related functions, including authentication and authorization. Before configuring the user registry or repository, decide which user registry or repository to use.

Although WebSphere Application Server supports different types of user registries, only one can be active. This active registry is shared by all of the product server processes.

WebSphere Application Server supports the following types of user registries:

- ▶ Federated repository
- ▶ Local operating system
- ▶ Stand-alone Lightweight Directory Access Protocol (LDAP) registry

In the event that none of these options are feasible for you, you can implement a custom registry, for example, a database. WebSphere provides a service provider interface (SPI) that you can implement to interact with your custom user registry. The SPI is the *UserRegistry* interface, which is the same interface used by the local OS and LDAP registry implementations.

The *UserRegistry* interface is a collection of methods that are required to authenticate individual users using either a password or certificates and to collect

information about the user authorization purposes. This interface also 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.

Delegation

Delegation occurs when a client requests a method on server A and the method request results in a new invocation to another method of an object in server B. Server A performs the authentication of the identity of the client and passes the request to server B. Server B assumes that the client identity has been verified by server A and responds to that request.

Secure Sockets Layer (SSL)

SSL is the industry standard for data interchange encryption between clients and servers. SSL provides secure connections through:

- ▶ 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 encrypted, password-protected file that includes:

- ▶ The name of the certificate holder
- ▶ The private key for encryption/decryption
- ▶ The verification of sender's public key
- ▶ The name of the certificate authority
- ▶ Validity period for the certificate

A certificate authority is an organization that issues certificates after verifying the requester's identity.

Single sign-on (SSO)

SSO is the process where users provide their credentials (user ID, password, and token) just once within a session. These credentials are available to all enterprise applications for which SSO was enabled without prompting the user to re-enter a user ID and password when switching from one application to another.

The following list describes the requirements for enabling SSO using LTPA:

- ▶ All SSO participating servers must use the same user registry (for example, the LDAP server).
- ▶ All SSO participating servers must be in the same Domain Name System. (Cookies are issued with a domain name and 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 these cause the cookie not to work properly.
- ▶ The browser must be configured to accept cookies.
- ▶ Server time and time zone must be correct. The SSO token expiration time is absolute.
- ▶ All servers participating in the SSO scenario must be configured to share LTPA keys.

Public key infrastructure

A public key infrastructure (PKI) represents a system of digital certificates, certificate authorities, registration authorities, a certificate management service, and X.500 directories. 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. Examples of these transactions include confirming the origin of proposal bids or the author of e-mail messages.

A PKI supports the use of certificate revocation lists (CRLs). A CRL is a list of revoked certificates. CRLs provide a more global method for authenticating client identity by certificate and can verify the validity of trusted CA certificates. An X.500 directory server stores and retrieves CRLs and trusted CA certificates. The protocols used for storing and retrieving information from an X.500 directory server include Directory Access Protocol (DAP) and LDAP. IBM HTTP Server supports LDAP.

You can distribute information about multiple directory servers over the Internet and intranets, enabling an organization to manage certificates, trust policy, and CRLs from either a central location or in a distributed manner. This capability makes the trust policy more dynamic because you can add or delete trusted CAs from a network of secure servers, without having to reconfigure each of the servers.

12.3.2 Authentication process

Figure 12-2 shows the authentication process in WebSphere Application Server. As the figure shows, authentication is required for enterprise bean clients and Web clients when they access protected resources.

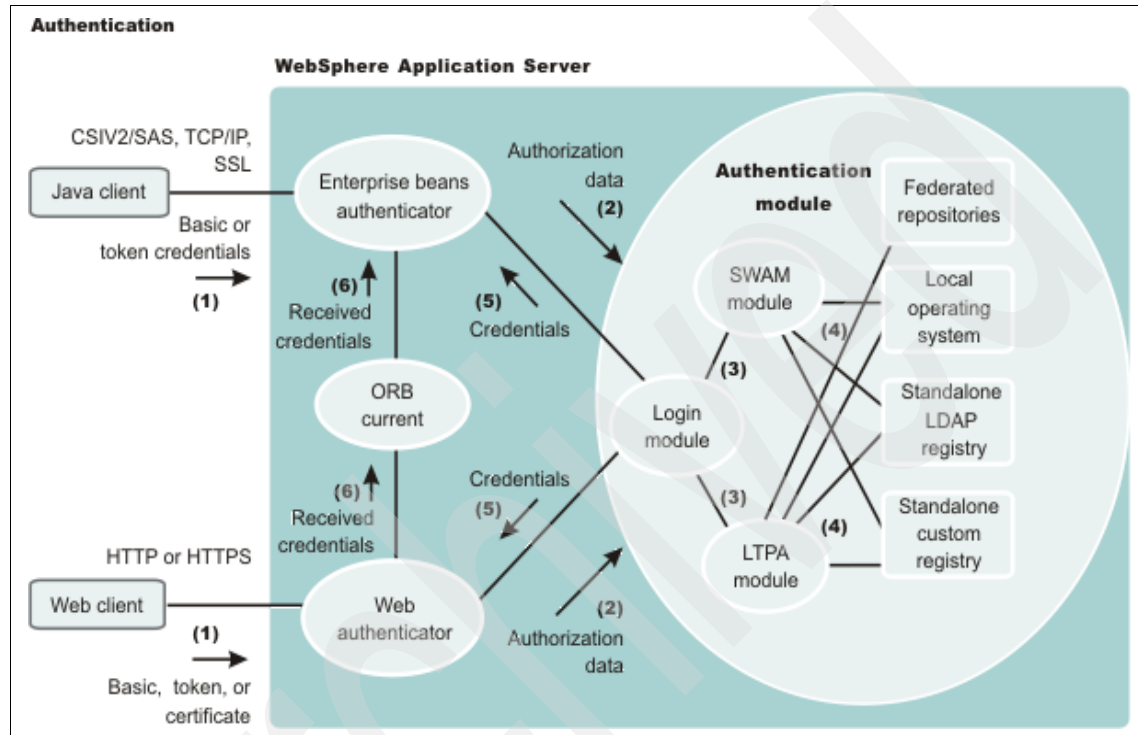


Figure 12-2 Authentication process

As illustrated in the figure, authentication involves two primary cases:

- ▶ Web client authentication
- ▶ EJB client authentication

Web client authentication

When a security policy is specified for a Web resource and WebSphere Application Server security is enforced, the Web container performs access control when the resource is requested by a Web client. The Web container challenges the Web client for authentication data (if none is present) according to the specified authentication method, ensures that the data constraints are met, and determines whether the authenticated user has the required security role.

WebSphere Application Server takes the authentication information, looks up the user's unique ID in the registry, and then verifies the password against the registry.

Securing Web components

Web components such as static HTML files, JSPs, and servlets can be secured using either the HTTP server or WebSphere Application Server. When discussing securing Web components, there are two elements to consider:

- ▶ Static HTML components
- ▶ Servlets and JSPs

Static HTML components

Most Web servers can secure the files they serve. WebSphere Application Server also has the ability to serve static content using the built-in Web server. WebSphere cannot manage the static content stored in the Web server. It can only serve the content that is packaged as part of the Web module (WAR file). The Application Server Toolkit can be used to set up security constraints to protect static content for the Web application module.

Servlets and JSPs

WebSphere Application Server can secure dynamic resources such as servlets using role-based declarative security mechanisms. This means that the logical application security structure is defined independently from the application itself. The logical security structure is stored in deployment descriptors of the application. For servlets, WebSphere Application Server enables you to protect the resources on the method level. For example, the POST method of a servlet can be part of a different security constraint than the GET method. The full list of predefined methods that can be secured is:

- ▶ GET
- ▶ POST
- ▶ PUT
- ▶ DELETE
- ▶ HEAD
- ▶ OPTION
- ▶ TRACE

Using method-level security constraints for servlets, you might want to separate dynamic content that all the users can view from the administrative functions that only privileged users are allowed to access. In WebSphere Application Server, this is done using different security constraints for the different servlet methods.

EJB client authentication

Enterprise bean clients, such as a servlet or other enterprise beans or a pure client, send the authentication information to a Web application server using one of the following protocols:

- ▶ Common Secure Interoperability Version 2 (CSIv2)
- ▶ Secure Authentication Service (SAS)
- ▶ z/OS Secure Authentication Service (z/SAS)

Note: SAS or z/SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

12.3.3 Authorization

Authorization is the process of checking whether a given user has the privileges necessary to get access to the requested resource. WebSphere Application Server supports many authorization technologies including:

- ▶ Authorization involving the Web container and J2EE technology
- ▶ Authorization involving an enterprise bean application and J2EE technology
- ▶ Authorization involving Web services and J2EE technology
- ▶ Java Message Service (JMS)
- ▶ Java Authorization Contract for Containers (JACC)

WebSphere Application Server V6 supports both a default authorization provider, which was supported in previous releases, and an authorization provider that is based on the JACC specification. The JACC-based authorization provider enables third-party security providers to handle the J2EE authorization. WebSphere Application Server also supports an authorization infrastructure that enables you to plug in an external authorization provider.

When security is enabled, the default authorization is used unless a JACC provider is specified. The default authorization does not require special setup, and the default authorization engine makes all of the authorization decisions.

When a JACC provider is used for authorization, the J2EE application-based authorization decisions are delegated to the provider per the JACC specification.

All administrative security authorization decisions are made by the WebSphere Application Server default authorization engine. The JACC provider is not called to make the authorization decisions for administrative security.

When a protected J2EE resource is accessed, the authorization decision to give access to the principal is the same whether using the default authorization engine or a JACC provider. Both of the authorization models satisfy the J2EE specification, so there should be no differences in function. Choose a JACC provider only when you want to work with an external security provider such as Tivoli Access Manager. The provider must support the JACC specification and be set up to work with the WebSphere Application Server.

12.4 J2EE security

The J2EE specification defines the building blocks and elements of a J2EE application that build an enterprise application. The specification also provides details about security related to the different elements. A typical J2EE application consists of an application client tier, a Web tier, and EJB tier. When designing a security solution, you need to be aware of the connections between each of the modules. Figure 12-3 shows the components of a J2EE application.

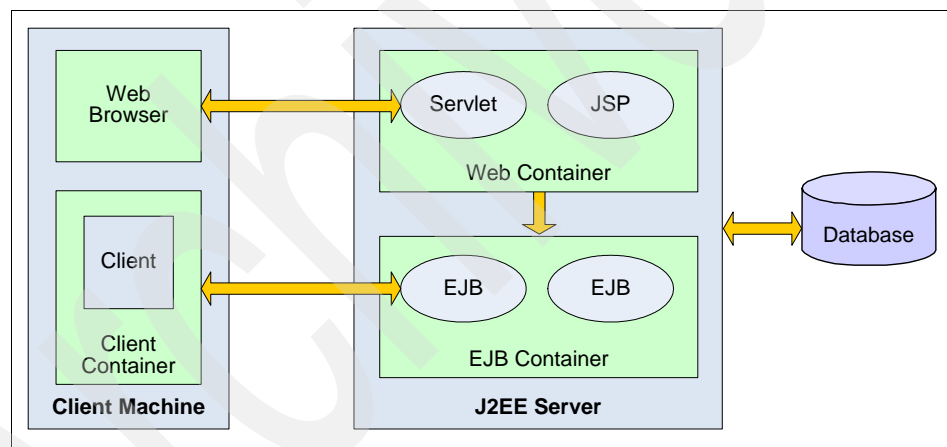


Figure 12-3 J2EE application components

For example, a user 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 need to consider the security for every module.

12.4.1 Security roles

A security role is a logical grouping of users that are defined by the application assembler. Because at development time it is not possible to know all the users

that are going to be using the application, security roles provide the developers a mechanism through which the security policies for an application can be defined. This is done by creating named sets of users (for example, managers, customers, and employees) that have access to secure resources and methods. At application assembly time, these users are just place holders. At deployment time, they are mapped to real users or groups. Figure 12-4 shows an example of how roles can be mapped to users.

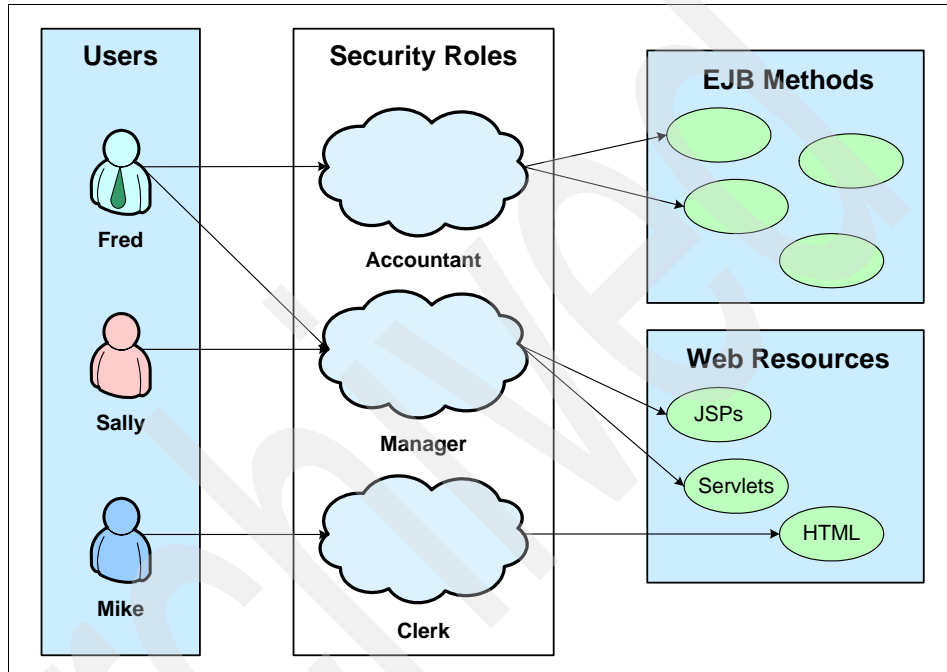


Figure 12-4 User role mapping

This two-phase approach to security gives a great deal of flexibility because deployers and administrators have a great control over how their users are mapped to the various security roles.

12.4.2 Security for J2EE resources

J2EE containers enforce security in two ways:

- ▶ Declarative security
- ▶ Programmatic security

Declarative security

Declarative security is the means by which an application's security policies can be expressed externally to the application code. At application assembly time, security policies are defined in an application's *deployment descriptor*. A deployment descriptor is an XML file that includes a representation of an application's security requirements, including the application's security roles, access control, and authentication requirements. When using declarative security, application developers can write component methods that are completely unaware of security. By making changes to the deployment descriptor, an application's security environment can be radically changed without requiring any changes in application code. There are several descriptor files that are used for security role mappings. These files can be created using:

- ▶ IBM Rational Application Developer
- ▶ Application Server Toolkit

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 at all. 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 very robust security infrastructure, which is transparent to the developer. However, there are cases where the security provided is not sufficient and the developer wants greater control over to what the user has access. For such cases, there are a few security APIs that the developers can implement.

Java Authentication and Authorization Service

The Java Authentication and Authorization Service (JAAS) extends the Java 2 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 and extends the access control architecture of the Java 2 platform in a compatible fashion to support user-based authorization or principal-based authorization. WebSphere Application Server fully supports the JAAS architecture and extends the access control architecture to support role-based authorization for J2EE resources including servlets, JSP files, and EJB components.

JAAS authentication is performed in a pluggable fashion. This permits Java applications to remain independent from underlying authentication technologies. Therefore, new or updated authentication technologies can be plugged under an application without requiring modifications to the application itself. Applications

enable the authentication process by instantiating a `LoginContext` object, which in turn references a `Configuration` to determine the authentication technology, or `LoginModule`, to be used in performing the authentication. A typical `LoginModule` can prompt for and verify a user name and password.

A typical JAAS-secured application has two 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.

Trust association interceptors

Web clients can also authenticate by using a trust association interceptor (TAI). 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 onto the resulting credentials passed by the reverse proxy server.

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 exploit 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.

Java Authorization Contract for Containers (JACC)

JACC is a specification introduced in J2EE 1.4. This specification defines a contract between J2EE containers and authorization providers. The contract enables third-party authorization providers to plug into J2EE 1.4 application servers (such as WebSphere Application Server) to make the authorization decisions when a J2EE resource is accessed. The access decisions are made through the standard `java.security.Policy` object.

Note: For WebSphere Application Server for z/OS, if SAF- based authorization is implemented, the implementation at this point does not use or implement the JACC policy provider interface.

WebSphere Application Server supports two authorization contracts using both a native and a third-party JACC provider implementation. The default (out-of-box) solution is the WebSphere Application Server default J2EE role-based authorization implementation, which does not implement the JACC policy provider interface. The JACC provider in WebSphere Application Server is implemented by both the client and the server pieces of the Tivoli Access Manager server. The client piece of Tivoli Access Manager is embedded in WebSphere Application Server. The server piece is located on a separate installable CD that is shipped as part of the WebSphere Application Server Network Deployment V6.1 package.

The JACC specification does not specify how to handle the authorization table (user or group to role) information between the container and the provider. It is the responsibility of the provider to provide some management facilities to handle this information. It does not require the container to provide the authorization table information in the binding file to the provider.

WebSphere Application Server provides two role configuration interfaces (RoleConfigurationFactory and RoleConfiguration) to help the provider obtain information from the binding file, as well as an initialization interface (InitializeJACCProvider). The implementation of these interfaces is optional.

Note: The JACC provider is not an out-of-box solution. You must configure WebSphere Application Server to use the JACC provider.

JACC access decisions

When security is enabled and an enterprise bean or Web resource is accessed, the EJB container or Web container calls the security runtime to make an authorization decision on whether to permit access. When using an external provider, the access decision is delegated to that provider. According to the JACC specification, the appropriate permission object is created, the appropriate policy context handlers are registered, and the appropriate policy context identifier (contextID) is set. A call is made to the java.security.Policy object method implemented by the provider to make the access decision. See Figure 12-5 on page 277.

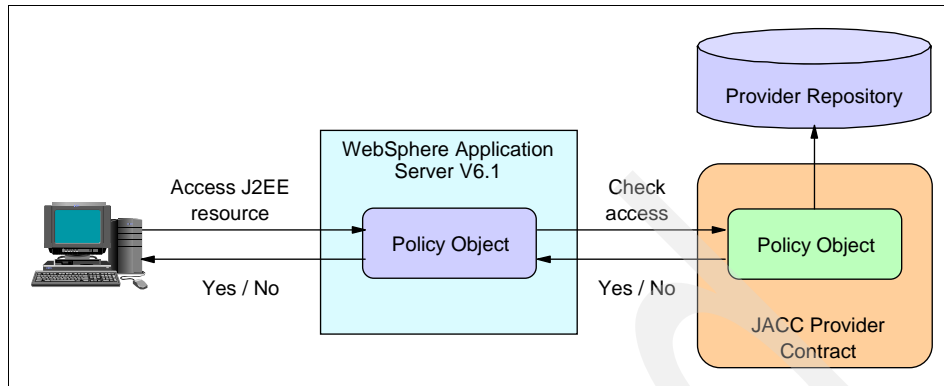


Figure 12-5 JACC provider architecture

Dynamic module updates in JACC

WebSphere Application Server supports dynamic updates to Web modules under certain conditions. If a Web module is updated, deleted, or added to an application, only that module is stopped or started as appropriate. The other existing modules in the application are not impacted, and the application itself is not stopped and then restarted.

When any security policies are modified in the Web modules, the application is stopped and then restarted when using the default authorization engine. When using the JACC-based authorization, the behavior depends on the functionality that a provider supports. If a provider can handle dynamic changes to the Web modules, only the Web modules are impacted. Otherwise, the entire application is stopped and restarted for the new changes in the Web modules to take effect.

A provider can indicate if they support the dynamic updates by configuring the "Supports dynamic module updates" option in the JACC configuration model. Enable or disable this option by using the administrative console or by scripting. It is expected that most providers store the policy information in their external repository, which makes it possible for them to support these dynamic updates. This option is enabled by default for most providers.

When the "Supports dynamic module updates" option is enabled, if a Web module that contains security roles is dynamically added, modified, or deleted, only the specific Web modules are impacted and restarted. If the option is disabled, the entire application is restarted. When dynamic updates are performed, the security policy information of the modules impacted are propagated to the provider.

Mixed node environment and JACC

Authorization using JACC is a new feature in WebSphere Application Server V6. JACC configuration is set up at the cell level and is applicable for all the nodes and servers in that cell. If you are planning to use the JACC-based authorization, the cell should only contain Version 6.0.x and later nodes. This means that a mixed node environment containing a set of V5 or later nodes in a Version 6.0.x cell or later is not supported.

Java 2 security

Where J2EE security guards access to Web resources such as servlets, JSPs, and EJBs, Java 2 security guards access to system resources such as file I/O, sockets, and properties.

Note: You need to be careful before enabling Java 2 security. Java 2 security places new requirements on application developers and administrators and your applications might not be prepared for the very fine-grain access control programming model that Java 2 security is capable of enforcing.

For more information about Java 2 security, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/csec_rsecmgr2.html

12.5 Planning for security

When planning for security, it is very important that you understand the difference between administrative security and application security from the WebSphere perspective:

- ▶ **Administrative security:** Administrative security protects the cell from unauthorized modification.
- ▶ **Application security:** Application security enables security for the applications in your environment. This type of security provides application isolation and requirements for authenticating application users.

In previous releases of WebSphere Application Server, when a user enabled global security, both administrative and application security were enabled. Now in WebSphere Application Server V6.1, these security functions can be enabled separately. Administrative security is enabled or disabled during profile creation. The default is for it to be enabled. Application security is disabled, by default, and must be enabled after profile creation using the administrative tools. To enable application security, you must also enable administrative security.

When a new application server profile or deployment manager profile is created, you have the following options for administrative security:

- ▶ Use WebSphere Application Server to manage user identities and the authorization policy (file-based repository).
- ▶ Do not enable security.
- ▶ Use a z/OS security product to manage user identities and authorization policy (z/OS only).

LTPA is used as the authentication mechanism.

After profile creation, you can activate application security, selecting from the following user registry options:

- ▶ Federated repository (including the file-based registry created for administrative security)
- ▶ Local operating system
- ▶ Stand-alone LDAP registry
- ▶ Stand-alone custom registry

For more information about administrative security and the roles for administrators, see “Secure administration tasks” on page 107.

Server IDs

WebSphere Application Server V6.1 distinguishes between the user identities for administrators who manage the environment and server identities for authenticating server-to-server communications.

When you configure the user registry from the console, you can choose to have a server ID automatically generated. Automatically generated IDs are not stored in the registry.

Alternatively, you can specify a user ID and password. On distributed systems, this is a user ID that exists in the user registry. On z/OS, you can choose to use the ID for that is associated with each started task.

If you are adding a Version 5.0.x or 6.0.x node to a Version 6.1 cell, you must ensure that the Version 5.x or Version 6.0.x server identity and password are defined in the repository for this cell.

Scenarios

In order to give a better explanation of what the implications are if you select one of the previous options, we describe two scenarios with different configurations to illustrate common setups.

Scenario 1: Enable administrative security at profile creation

In this scenario, let us say that you want to enable administrative security during the installation process. With this option, the profile creation tools create a file-based registry in the configuration file system, and a user ID /password combination of your choice is registered with administrator authority. Self-signed digital certificates for servers are created in the configuration file system automatically and LTPA is enabled.

Additional users can be added and assigned administrative roles through the administrative tools (for example, through the “Console users and groups” settings in the administrative console).

Note that so far, only administrative security has been enabled. After the profile is complete and the application server or deployment manager is running, you can enable application security through the administrative console.

You can federate the file-based registry holding the administrative security information with another user registry of your choice.

Scenario 2: Enable security after profile creation

In this scenario, let us say that you do not enable administrative security during the profile creation process. Anyone with access to the administrative console port can make changes to the server or cell configuration.

After profile creation, you can enable both administrative and application security using a user registry of your choice.

Scenario 3: Using a z/OS security product

In this scenario, let us say that you want to enable administrative security during the profile creation process using a z/OS security product to manage security. With this option, each user and group identity corresponds to a user ID or group in the z/OS system's SAF-compliant security system (IBM RACF or an equivalent product).

Access to WebSphere Application Server roles is controlled using the SAF EJBROLE profile, and digital certificates for SSL communication are stored in the z/OS security product.

Summary of options to enable security at profile creation

Table 12-2 summarizes these options.

Table 12-2 Options to enable security at profile creation

Option chosen	Implications
Use WebSphere Application Server to manage user identities and the authorization policy.	<ul style="list-style-type: none"> ▶ Each WebSphere Application Server user and group identity corresponds to an entry in a WebSphere Application Server user registry. The initial registry is a file-based user registry, created during customization, and residing in the configuration file system. ▶ Access to roles is controlled using WebSphere Application Server role bindings. In particular, administrative roles are controlled using the "Console users and groups" settings in the administrative console. ▶ Digital certificates for SSL communication are stored in the configuration file system.
Do not enable security.	<ul style="list-style-type: none"> ▶ No administrative security is configured. Anyone with access to the administrative console port can make changes to the server or cell configuration.
Use a z/OS security product to manage user identities and authorization policy (z/OS only).	<ul style="list-style-type: none"> ▶ Each WebSphere Application Server user and group identity corresponds to a user ID or group in the z/OS system's SAF-compliant security system (RACF or an equivalent product). ▶ Access to WebSphere Application Server roles is controlled using the SAF EJBROLE profile. ▶ Digital certificates for SSL communication are stored in the z/OS security product.

12.6 Planning checklist for security

Table 12-3 provides a summary of items to consider as you plan and additional resources that can help you.

Table 12-3 Planning checklist for Web services

Planning item
Determine when and how you will enable WebSphere Application Security (bus, administrative, application, Java 2).

Planning item
Create a strategy for administrative security (see also 6.5.6, “Security considerations for the installation” on page 106).
Determine the type of user registry you will use and procure the appropriate products and licenses. If you do not want to use a federated repository, delay turning on admin security until after installation. Populate the user registry with the appropriate user IDs and groups for initial security.
Determine the authentication mechanism (LTPA strongly suggested).
Determine the authorization method (default or JACC). If JACC, plan for the implementation of the JACC provider.
Plan where you will implement SSL in your network.
Plan for certificate management.
Plan for single sign-on. See the single sign-on topics in the WebSphere Application Server Information Center at: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/csec_sso.html
Create a strategy for securing applications using J2EE security (declarative or programmatic?). Application security requires close cooperation between application developers, security specialists, and administrators. Plan for coordinating role definitions with development and assigning users to roles during the application installation. Determine individual application components that have special security requirements.
Review and incorporate security strategies for Web services (see 11.5.1, “What are the options for Web service security?” on page 250).
Review and incorporate security strategies for the service integration bus (see 10.6, “Messaging features: How secure and reliable is it?” on page 224).

Resources

For a good overall reference for WebSphere Application Server security, refer to *WebSphere Application Server V6 Security Handbook*, SG24-6316.

We suggest that you have a copy of this book available as you plan to secure your environment.

The WebSphere Application Server Information Center also contains a lot of useful information. For a good entry point to security topics, see:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/welc6topsecuring.html>



Sample topology walk-through

This appendix explores a complex topology and provides general guidance on setting it up. It contains the following sections:

- ▶ Topology review
- ▶ Component installation
- ▶ Testing the topology
- ▶ Deploying applications

Topology review

The topology shown in Figure A-1 takes elements from several of the topologies found in Chapter 5, “Topologies” on page 59 and combines them. The motivating factor for this combination is scalability, workload management, and high availability. This topology is a common implementation according to discussions with IBM clients, as well as IBM Global Services teams who are responsible for the implementation of WebSphere environments.

Note: This appendix addresses a subset of this topology. For the full details about implementing this topology, see *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

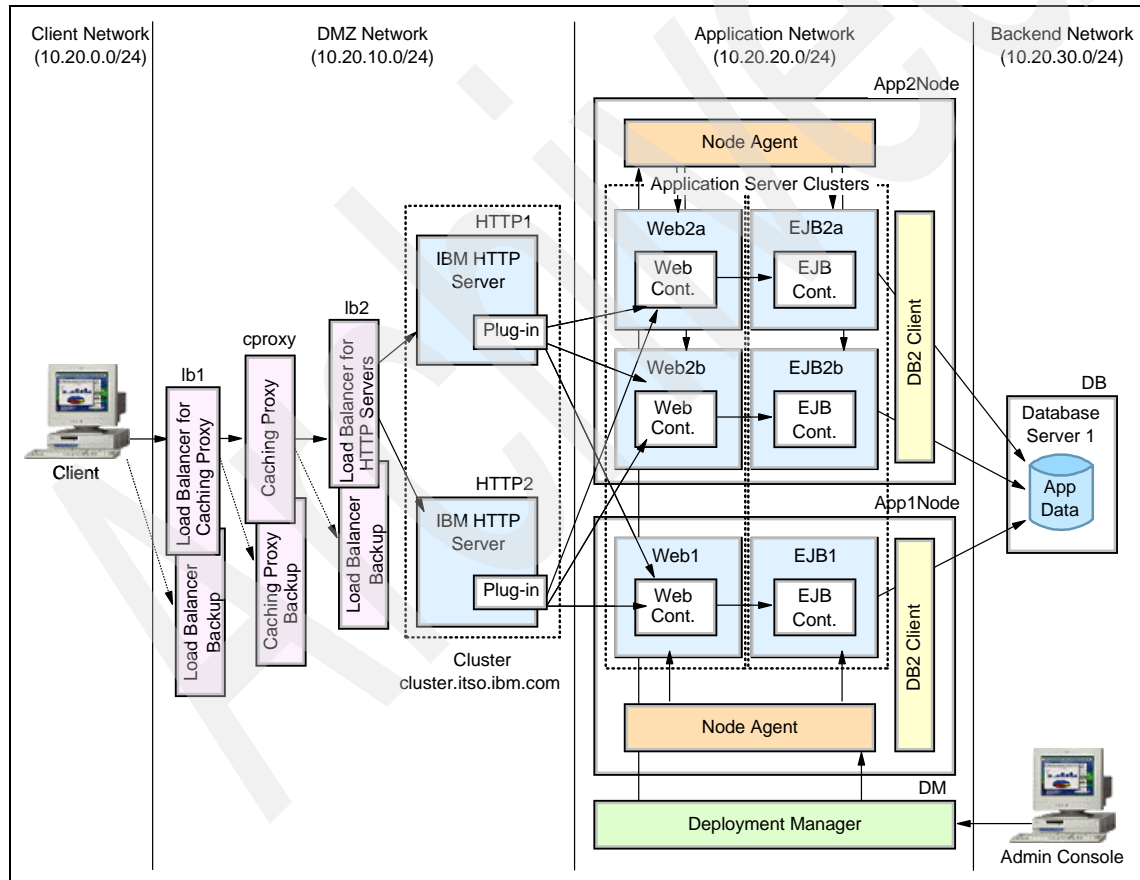


Figure A-1 Complex topology

This configuration provides both the greatest resiliency of a site, and the greatest administrative complexity. The topology includes the following elements:

- ▶ 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 is forwarded to the appropriate application server by the Web server plug-in.
- ▶ A Caching Proxy server that keeps a local cache of recently accessed pages
Cachable content includes static Web pages and JSPs with dynamically generated but infrequently changing 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.
A backup server is configured for high availability.
- ▶ A Load Balancer server to direct incoming requests to the Caching Proxy and a second Load Balancer server to manage the workload across the HTTP servers
The Load Balancers distribute 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 for high availability.
- ▶ 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. It has a master copy of the configuration that should be backed up on a regular basis.
- ▶ Two clusters consisting of three application servers
Each cluster spans two machines. Note that in this topology, one cluster contains application servers that provide the Web container functionality of the applications (servlets, JSPs), and the second cluster contains the EJB container functionality. Whether you choose to do this or not is a matter of careful consideration. Although it provides failover and workload management capabilities for both Web and EJB containers, it can also affect performance.
- ▶ A dedicated database server running IBM DB2 UDB V8.2

Advantages

This topology is designed to maximize scalability, workload management, and high availability and has the following advantages:

- ▶ Single points of failure are eliminated for many components (Web server, application server, and so on) through the redundancy of those components.
- ▶ It provides both hardware and software failure isolation. Hardware and software upgrades can be easily handled during off-peak hours.
- ▶ Horizontal scaling is done by using both the Load Balancer IP sprayer (for the Web server nodes) and the application server cluster to maximize availability.
- ▶ Application performance is improved by using several techniques:
 - Hosting application servers on multiple physical machines to boost the available processing power
 - Using clusters to vertically scale application servers, which makes more efficient use of the resources of each machine
- ▶ Applications with this topology can make use of workload management techniques. In this example, workload management is performed through:
 - The Network Dispatcher component of the Load Balancer to distribute client HTTP requests to each Web server
 - WebSphere Application Server workload management to distribute work among clustered application servers

Disadvantages

This combined topology has the following disadvantages:

- ▶ Isolating multiple components and providing hardware backups increases the overall cost of this implementation.
- ▶ Licenses must be obtained for all the machines to be used, and this can add significant costs to provide resiliency.
- ▶ This is an example of a standard implementation, but it should also be noted that it is one of the most complex. Consider the costs of administration and configuration work in relation to the benefits of increased performance, higher throughput, and greater reliability.

Component installation

The example topology involves isolating as many of the components as possible. For the purposes of this illustration, we did not implement all of the components in the topology due to the availability of equipment and resources in our lab. We did, however, create a representative subset, shown in Figure A-2. We take you through the general process required to build this environment.

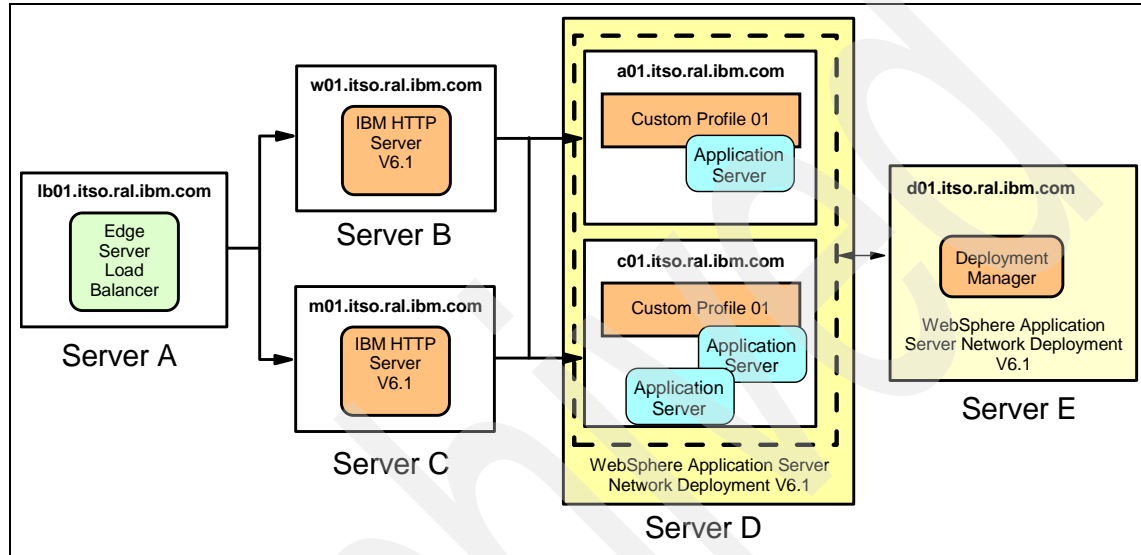


Figure A-2 Completed topology

The remainder of this appendix works through the process of building out the components shown in Figure A-2.

The installation process for all the components is fairly similar. With the exception of the Load Balancer, all the installations are started from a single point, the Launchpad for the Network Deployment package, shown in Figure A-3.

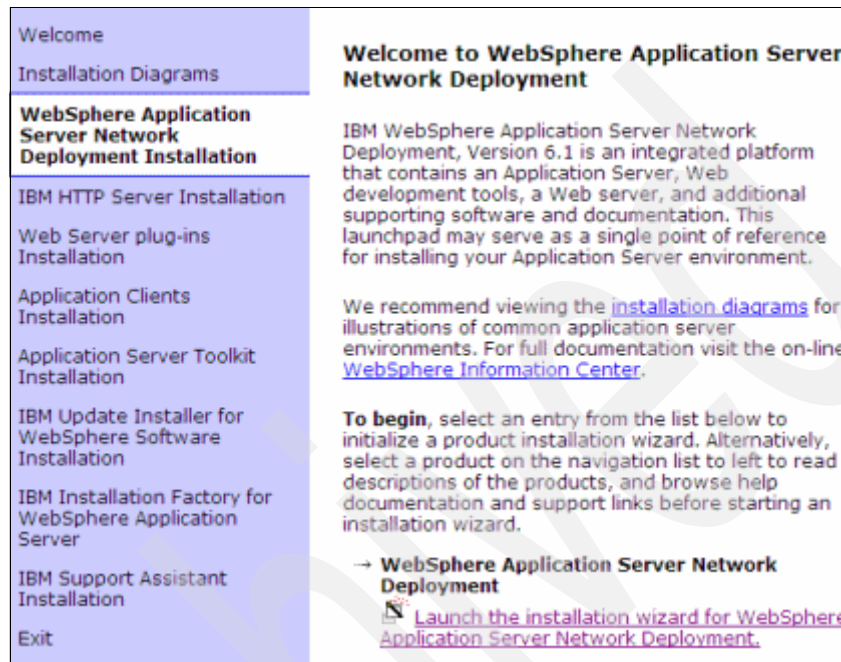


Figure A-3 Network Deployment V6.1 Installation Launchpad

Deployment manager node (server E)

Server E hosts the deployment manager used to administer the servers (including the Web servers and application servers), applications, and resources in the WebSphere Application Server cell. To build this node:

1. Install WebSphere Application Server Network Deployment.
From the Launchpad, we selected **Launch the Installation wizard for WebSphere Application Server Network Deployment** and navigated through the panels in the installation wizard.
2. Launch the Profile Management Tool to create a deployment manager profile for the node. You can launch it after installation or as a last step in the installation.

We launched the PMT after the installation and selected **Deployment manager**. We followed the prompts, taking the defaults.

3. After creating the profile, we used the First Steps menu to start the deployment manager, test the installation, and finally to log in to the administrative console.

This resulted in a running deployment manager on d01.itso.ral.ibm.com, shown in Figure A-2 on page 287.

Application server nodes (server D)

Server D hosts the application server clusters. This configuration is one in which WebSphere Application Server is installed on a node and a custom profile is built to be federated to a cell.

The process for installing and building the WebSphere Application Server components is the same for each application server node. In the sample topology, there are two application server nodes on two separate machines. In our walk-through, we only have one machine but still create two nodes for illustration purposes.

We performed the following actions:

1. On each physical machine, install WebSphere Application Server Network Deployment.
2. For each node, launch the Profile Management Tool (PMT) to create a custom profile for the node.

We launched the PMT after the installation for each node. We selected **Custom Profile** and followed the prompts taking the defaults, including allowing the node to be federated to the cell as part of the process (the deployment manager must be installed and running).

The result is the two nodes c01.itso.ral.ibm.com and a01.itso.ral.ibm.com in Figure A-2 on page 287. Because these are custom nodes, no application servers or applications exist yet. We create these using the administrative console on the deployment manager.

Tip: Nodes that are federated need to have their system clocks very closely in sync. This does not mean in the same time zone, but that they must be within five minutes of each other with respect to time zone variance. If they are not, the node will not federate. If the node is federated and then the gap grows to greater than five minutes, the node will not be able to synchronize correctly with the deployment manager repository.

IBM HTTP Server V6.1 (server B and server C)

Server B (w01.itso.ral.ibm.com) and server C (m01.itso.ral.ibm.com) host the Web servers. Communication between the Web servers and the application servers is facilitated through the Web server plug-in module. The Web servers are defined to the deployment manager as unmanaged nodes. Because they are IBM HTTP Servers, we can propagate changes to the plugin-cfg.xml file through the console.

We performed the following actions:

- ▶ On each physical machine:

We installed IBM HTTP Server and the Web server plug-in.

From the Launchpad, we selected **Launch the installation wizard for IBM HTTP Server**. We accepted the defaults, which included the default ports of 80 for the HTTP Server and 8008 for the HTTP Administration Server. We also took the default of creating a Windows service for the HTTP server and administration server.

The installation panels gave us the option of entering a user ID and password to be used to log in to the HTTP Administration Server, which we did. Additional user IDs can be added after the installation using the `htpasswd` command in the HTTP server bin directory.

Because we installed the IBM HTTP Server packaged with WebSphere Application Server, we were able to launch the plug-in installation as part of the IBM HTTP Server installation.

- ▶ On the deployment manager machine:

After correctly installing the IBM HTTP Server and Web server plug-in, we copied the `<plugin_home>/bin/configure<web_server_name>` script from each plug-in installation to the deployment manager's `<profile_home>/bin` directory on server E and executed it. This created the Web server definitions on the deployment manager and generated the plug-ins. We then propagated the new plug-in configuration files back to the IBM HTTP Server installations to the location stated in the `httpd.conf` file.

After completing the installation, we ran functional tests to verify the operation of each Web server. First, we started the HTTP server and administration server on each machine. Our server installation was Windows, so we were able to start both using the Windows services defined for them.

We verified that the HTTP servers were working by accessing the default splash page (`http://localhost`) for each. Next, we verified that the deployment manager could access the configuration data, log files, and start and stop the Web servers through the administrative console.

We then tested loading the snoop servlet and the Plants by WebSphere sample application using each remote Web server instead of the embedded HTTP server for WebSphere.

Creating the application server clusters

Our test topology required a cluster of application servers spread across the two application server nodes. To do so, we created a cluster named test from the administrative console and added three servers to the cluster (one from the a01 node and two from the c01 node). We configured memory-to-memory session replication as our distributed session management mechanism.

We wanted to test a sample application, so we took the simple step of creating an application server profile and federating it to the cell. This gave us the sample and default applications. We then changed the module mapping for the Default Application to use the cluster instead of the original stand-alone server. We assigned the Web modules to the newly defined Web servers.

After we had the module mapping set correctly, we were then able to regenerate the plugin-cfg.xml configuration file and propagate it to remote Web servers. We then tested the default application (http://<web_server>/snoop) to verify that the applications are still served correctly.

Note that in the complex topology shown in Figure A-1 on page 284, there are two clusters. The second cluster is created in the same manner as the first.

Load Balancer (server A)

We chose to use the Load Balancer component of the WebSphere Edge components so that we could distribute traffic between the two Web servers. The Caching Proxy would typically be used for segmenting traffic, but was not implemented for this testing. A single cluster was created that supported port 80 load balancing. We chose not to implement a port 443 cluster in the test environment. Referring to Figure A-2 on page 287, the machine that was our load balancing machine was named lb.itso.ral.ibm.com and is labeled server A.

Configuring the Load Balancer can be done in one of two ways: using the graphical console, or using the command line. For this test, we chose to use the graphical console. To invoke the console, execute the **lbadmin** command from a prompt, or click the icon to run this script. If you prefer the command line, you can issue the **dscontro1** command from a prompt and enter the commands there.

We performed the following steps:

1. First, create a cluster. This cluster is the host name and IP combination that are used by clients to access the Web site and guarantee the load balancing.

When creating the cluster, the console prompts you for the interface to which to bind this cluster. Select the primary interface as defined for that machine. It is customary for the name of the cluster (which is simply a label) to be the DNS name for the cluster for ease of administration. After creating the cluster, you add a port to the cluster. For our purposes, we chose **HTTP**, which defaults to port 80.

The next step is to add the servers that will be serving for this cluster. We added `m01.itso.ral.ibm.com` and `w01.itso.ral.ibm.com`. We used this name for both the label and the host name so that we could verify that the Load Balancer could resolve the names correctly to an IP address.

Note: The dialog box in the console has an option for the network router address and a check box to indicate it must be used. This should only be the case if your Web server is not on the same network as the Load Balancer. For the purposes of our testing, all the machines were on the same logical network.

2. Now that the Web servers are defined to the cluster, we must go to the Web servers and verify the configuration to assure they can serve that cluster address. When using Load Balancer, you are required to bind the cluster IP as an alias to the loopback adapter for each Web server, and the load balancer itself. Refer to specifics for your operating system for adding a loopback alias.

Important: Windows operating systems require a special network adapter be installed to handle loopback addressing. Also, Windows adds a bad route for the loopback alias, which then must also be removed prior to verifying that the server is functioning correctly. Refer to *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392, for details about the adapter and the bad route.

3. At this point, we are now ready to enable the advisor. The Load Balancer uses a port advisor to verify the health of servers participating in the cluster. In the case of HTTP requests, the advisor sends a HEAD request to each of the servers participating in the cluster. If a server does not respond correctly to the advisor, the advisor marks the node down in the Load Balancer cluster so that traffic is not sprayed to the malfunctioning server.
4. To do unit testing of the Load Balancer cluster, we simply marked one of the two Web servers “down” in sequence, and then verified that the cluster was still serving from the remaining Web server.

Deploying applications

So far, we have built a WebSphere Application Server environment with a cluster of application servers. To deploy applications in this topology, you must remember to map the modules to the clustered application servers. In the complex topology (Figure A-1 on page 284), this involves mapping Web modules to servers in one cluster and the EJB modules to servers in the second cluster. In our simplified version, we simply map all the modules to the single cluster we created. Remember to also map the Web modules to both Web servers so that the plug-ins are generated correctly. Then, you need to regenerate and propagate the plug-ins.

Testing the topology

After building and unit testing the entire environment, we verified the expected functionality. To do this, we created a simple frameset HTML page that was placed on each one of the Web servers. The top portion of the frame indicated which Web server was serving the request. The lower frame executed the snoop servlet, showing which application server responded to the plug-in request. Table A-1 shows the various page loads and the results of those loads. The table lists, in order, the page request and which Web server and application server responded to the two frames. We continued the tests until we verified that each Web server processed a request from each of the four application server instances.

Table A-1 *Testing the entire environment*

Web server host	Application server host	Server instance
w01.itso.ral.ibm.com	c01.itso.ral.ibm.com	AppSrv01
m01.itso.ral.ibm.com	c01.itso.ral.ibm.com	AppSrv01
w01.itso.ral.ibm.com	c01.itso.ral.ibm.com	AppSrv02
m01.itso.ral.ibm.com	c01.itso.ral.ibm.com	AppSrv02
w01.itso.ral.ibm.com	a01.itso.ral.ibm.com	AppSrv01
m01.itso.ral.ibm.com	a01.itso.ral.ibm.com	AppSrv01

One item of interest is that the operation of the Load Balancer is such that it uses a statistical round-robin when the advisor is running, which means that the Web servers responded based on the metrics the Load Balancer used to weight the likelihood a server would be able to respond to the request. Even after turning off the HTTP port advisor and the manager, the weights assigned to the servers

based on response times continued to keep the clustered Web servers from serving in a true round-robin fashion.

After testing at this level, we were confident that the environment was behaving as expected. Each Web server received some of the traffic from the Load Balancer, and each application server served some of the requests to the snoop servlet.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 298. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Enabling SOA Using WebSphere Messaging*, SG24-7163
- ▶ *Patterns: Service-Oriented Architecture and Web Services*, SG24-6303
- ▶ *Patterns: SOA Foundation Service Connectivity Scenario*, SG24-7228
- ▶ *Patterns: SOA Foundation Service Creation Scenario*, SG24-7240
- ▶ *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257
- ▶ *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688
- ▶ *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798
- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- ▶ *WebSphere Application Server V6 Security Handbook*, SG24-6316
- ▶ *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304
- ▶ *WebSphere Application Server V6.1: Technical Overview*, REDP-4191
- ▶ *WebSphere Security Fundamentals*, REDP-3944

Online resources

These Web sites are also relevant as further information sources:

- ▶ WebSphere Application Server Information Center
<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>

- ▶ WebSphere Application Server V6.1 supported software
<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27007642>
- ▶ WebSphere Application Server V6.1 system requirements
<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27007651>
- ▶ WebSphere Application Server support page
<http://www.ibm.com/software/webservers/appserv/was/support/>
- ▶ WebSphere Application Server Edge Component Information Center
<http://www.ibm.com/software/webservers/appserv/ecinfocenter.html>
- ▶ WebSphere MQ home page
<http://www.ibm.com/software/integration/wmq/>
- ▶ IBM Enterprise Workload Management
<http://www.ibm.com/servers/eserver/about/virtualization/enterprise/ewlm.html>
- ▶ Application Client for WebSphere Application Server topic
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/cli_appclients.html
- ▶ Planning to install WebSphere Application Client topic
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tins_scenario6.html
- ▶ Rational software
<http://www.ibm.com/software/rational>
- ▶ Rational Enterprise Generation Language
<http://www.ibm.com/software/awdtools/eglcobol/index.html>
- ▶ Rational Unified Process
<http://www.ibm.com/software/awdtools/rup>
- ▶ Rational Functional Tester
<http://www.ibm.com/software/awdtools/tester/functional/>
- ▶ IBM Tivoli Directory Server home page
<http://www.ibm.com/software/tivoli/products/directory-server/>
- ▶ IBM Tivoli Access Manager for e-business home page
<http://www.ibm.com/software/tivoli/products/access-mgr-e-bus/>

- ▶ IBM Tivoli Access Manager Information Center
<http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.itame.doc/toc.xml>
- ▶ IBM Tivoli Monitoring for Transaction Performance
<http://www.ibm.com/software/tivoli/products/monitor-transaction/>
- ▶ *Design for Scalability - an Update*
<http://www.ibm.com/developerworks/websphere/library/techarticles/hipods/scalability.html>
- ▶ *WebSphere for z/OS -- Heterogeneous Cells*
<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100644>
- ▶ *WebSphere for z/OS V6 -- WSC Sample ND Configuration*
<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100653>
- ▶ *WebSphere for z/OS V6.1 - New Things Encountered During Configuration*
<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100781>
- ▶ *WebSphere for z/OS Version 6 - Configuration Planning Spreadsheet*
<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/PRS1331>
- ▶ *Understanding WAS for z/OS*
<http://websphere.sys-con.com/read/98083.htm>
- ▶ *WebSphere Application Server V6.1: What's new in security?*
http://www.ibm.com/developerworks/websphere/library/techarticles/0606_botzum/0606_botzum.html
- ▶ *Using URL resources to manage J2EE property files in IBM WebSphere Application Server V5*
http://www.ibm.com/developerworks/websphere/library/techarticles/0502_botzum/0502_botzum.html
- ▶ *Maintain continuous availability while updating WebSphere Application Server enterprise applications*
http://www.ibm.com/developerworks/websphere/techjournal/0412_vansickel/0412_vansickel.html
- ▶ Sun Developer Network Java Message Service Web page
<http://java.sun.com/products/jms>
- ▶ Sun Developer Network J2EE Connector Architecture Web page
<http://java.sun.com/j2ee/connector/>

- ▶ ARM 4.0 specification
<http://www.opengroup.org/management/arm.htm/>
- ▶ Information about the ARM standard
<http://www.opengroup.org/pubs/catalog/c807.htm>
- ▶ The Apache Ant project
<http://ant.apache.org/index.html>
- ▶ Concurrent Versions System
<http://ximbiot.com/cvs/wiki>
- ▶ The Java Community Process Program - JSRs Java Specification Requests - detail JSR# 168
<http://jcp.org/en/jsr/detail?id=168>
- ▶ JSR 116 SIP Servlet API 1.0 Specification
<http://www.jcp.org/aboutJava/communityprocess/final/jsr116/>
- ▶ RFC 3261
<http://www.ietf.org/rfc/rfc3261.txt>
- ▶ JUnit, Testing Resources for Extreme Programming
<http://www.junit.org>
- ▶ Eclipse Test and Performance Tools Platform
<http://www.eclipse.org/hyades>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

access intent policies 182
Access Manager for WebSphere 16
activity.log 166
adapter teaming configuration 180
Adaptive Fast Path Architecture (AFPA) 64
address space 52
administrative console 155
administrative security 86, 90, 154–156, 254, 256, 258, 271, 278–281
administrative security roles 108, 154
AFPA 189
allow overflow 195
annotation-based programming 128, 131
Ant 130, 155, 157
Apache Ant 138
Apache HTTP Server 10
Apache Struts 130
Application Client 7, 86
application design 26
Application Response Measurement (ARM) 203–204
application rollout 163
application scope 161
application security 90, 260, 270, 278–280, 282
application server 27, 40, 65, 104, 119
application server profile 40–42, 70, 72, 94, 98
Application Server Toolkit 6–7, 97, 101, 124, 127–128, 161, 270, 274
application startup 159
application upgrades 148
ARM agent 204
atomic transaction contexts 233
authentication 225, 254, 263–264, 266–267, 269–270, 274–275, 282
authentication mechanism 264–265, 279, 282
authorization 225, 254, 265–267, 271, 277
authorization policy 281
authorization provider 275
automated build process 137
automated functional tests 139
availability 62

B

B2B 248
backup and recovery 36
backupConfig command 167
bean persistence 182
benchmark 27
benchmarking 30
bill of materials file 133
biometric 264
buffer pool 190
bus connector role 225
Business Registry 248
business-to-business 248

C

cachespec.xml 188
caching 187
Caching Proxy 4, 6, 189, 285, 291
caching service 190
caching strategy 34
capacity 142
cell 42, 103
cell profile 42, 95, 99
cell scope 160
certificate 107, 254, 256–257, 266–268, 280–281
certificate authority 268
certificate management service 268
certificate revocation lists (CRLs) 268
certification authority 267
change management 162
choreography 242–243
CIP 93
Cisco Local Director 62
Cloudscape 10, 187
cluster 40, 43, 45, 48, 61, 63, 73, 76, 80, 160, 163, 175, 177–178, 182, 187, 218, 226, 286, 289, 291
cluster scope 161
Clustered TM Policy 186
code versioning 134
cold backup 37
collaboration 134
com.ibm.websphere.ant.tasks 138
command assistance 154

- command line tools 155, 157
- Common Secure Interoperability Version 2 (CSlv2) 263
- composite message 243
- concurrent users 33
- Concurrent Versions System (CVS) 129, 136
- configuration backup and restore 167
- configuration repository 158
- configuration settings 144–145
- connection pool 174
- connection pool size 201–202
- contextID 276
- control region (CR) 52
- control region adjunct 53
- cookies 193–194, 197, 268
- CORBA security 263
- core group 184–185
- cryptography 211
- Crystal Reports 129, 131
- CSlv2 263, 271
- custom node profile 41
- Custom PMI API 199
- custom profile 74, 77, 79, 82, 95, 100, 289
- custom registry 266, 279
- customized install packages (CIPs) 8
- CVS 129–130, 135

D

- daemon server 54
- data availability 181
- data caching 190
- data protection 18
- data server 27
- data store 228
- database connection pool 174, 199
- database connection pool size 144
- database schema 149
- database server 175, 181
- database servers 10
- database tuning 175
- DataDirect Technologies JDBC Drivers for WebSphere Application Server 7
- DB2 8, 10
- DB2 performance configuration 202
- declarative security 274
- default authorization provider 271
- default messaging provider 20, 187, 202, 214
- default profile 111

- DefaultCoreGroup 185
- DefaultNodeGroup 42
- delegation 267
- denial of service 66
- deployment automation 131
- deployment descriptor 274
- deployment manager 7, 41–43, 45–46, 117, 184, 265, 285, 289–290
- deployment manager profile 41–42, 74, 77, 79, 82, 95, 97, 288
- destination 216
- development tools 127
- Directory Access Protocol (DAP) 268
- directory servers 11
- disable auto start 159
- disk utilization 33
- Distributed Process Discovery 158
- distributed server 41
- distributed server configuration 41
- distributed server environment 40, 42
- DMZ 60, 65, 68, 115, 211
- DNA 264
- DRS 197
- dscontrol 291
- dump files 165
- dynamic cache 197, 200
- dynamic cache size 202
- dynamic caching 64, 188, 250
- dynamic reloading 163
- dynamic SQL 172

E

- edge caching 188
- Edge Component 4, 6, 8, 62–63, 67, 71–72, 79, 82, 121, 291, 296
- Edge server 27
- Edge Side Include (ESI) 188
- EJB client 178
- EJB client authentication 269, 271
- EJB container 174, 178, 182, 202
- EJB module 293
- EJB persistence 181–182, 190–191
- encryption 257
- encryption key 256, 264
- enhanced EAR 128
- enhanced EAR file 132
- enterprise bean request 203
- Enterprise Generation Language (EGL) 130

enterprise service bus (ESB) 246
ESB gateway 246
ESI caching 188
ESI external caching 64
Extensible Markup Language
 see XML
Extensible Markup Language (XML) 44

F

FACILITY class profile 255
failover 40, 61, 63, 74, 76, 142, 180, 182–183, 185,
187, 218, 220, 257
Fast Response Cache Accelerator (FRCA) 188
fault tolerant storage 183
federate 42
federated repository 254, 266
file registry 257
file store 228
file synchronization 164
file-based registry 15
file-based repository 279, 281
fingerprints 264
firewall 66, 107, 180, 185, 248
First Steps 101
fix management 167
flat file message store 210
foreign bus 217, 222
fragment caching 188
FRCA 188–189

G

generic JMS provider 215
guaranteed delivery 246

H

HACMP 63, 182–183
hardware availability 180
hardware caching 189
hardware cluster 182
hardware cryptographic device support 255
heap size 165, 173–174, 202
heartbeat 182, 184
heterogeneous cell 43, 83
HFS 87, 94, 158
high availability 179, 182–183, 219, 226, 284
high availability group 186
high availability manager 181, 183–184, 186

horizontal cluster 43
horizontal scaling 48–49, 61, 63–64, 75, 77–78,
170, 175, 286
hot backup 37
hot deployment 163
HP JVM for HP-UX 170
HTML 129
htpasswd 290
HTTP administration server 290
HTTP request 203
Hyades 130

I

IBM Enterprise Workload Management 204
IBM Global Services 31
IBM HTTP Server 7, 10, 44, 49, 57, 65, 86–87, 290
IBM JDK 170
IBM service log 166
IBM Test Center 31
IBM Tivoli Access Manager client 4
IBM Tivoli Monitoring for Transaction Performance
204
IBM WebSeal Reverse Proxy 15
identity 267
implementation design 27
IMS 11
inactive pool cleanup interval 174
Informix 11
InitializeJACCProvider 276
install method 92
Installation Factory 8, 86, 89, 93, 120
installation verification utility 86
integrity 267
intermediary 245
Internet Information Services 10
intruder 259
IP sprayer 62–63, 67, 77–79, 81, 286
IP sprayers 62
iPlanet 16
iPlanet Directory Server 19
ISPF Customization Dialog 87

J

J2EE Connector Architecture (JCA) 213
J2EE security 272
J2EE security API 262
JAAS 15, 274–275
JACC 15, 17, 271, 275–278, 282

- JACC Policy provider interface 276
- JACC provider 271–272, 276, 282
- JACC-based authorization 278
- JACC-based authorization provider 271
- Jacl 129, 154
- jacl2jython 129
- Java 2 Platform, Enterprise Edition (J2EE) 2
- Java 2 security 15, 263, 278
- Java 2, Standard Edition (J2SE) 2
- Java 5 6, 128–129, 132, 170
- Java Authentication and Authorization Service 274
- Java Authentication and Authorization Service (JAAS) 15
- Java Authorization Contract for Containers (JACC) 4, 271
- Java environment variables 150
- Java Message Service (JMS) 271
- Java property files 144–145
- Java virtual machine (JVM) 51, 173
- Java Virtual Machine Profiling Interface (JVMPi) 200
- Java Virtual Machine Tool Interface (JVMTI) 200
- java.exe 51
- java.security 276
- java.security.Policy 275
- Javadoc 131
- JavaServer Faces 129–130
- JavaServer Pages 129
- JAXP SAX parser 233
- JAX-RPC 247
- JCA V1.5 specification 213
- JDBC call 202
- JDK 5 164
- JMS message delivery mode 227
- JMS provider 20
- JMS request 203
- JMS V1.1 specification 213
- JNDI 160–161
- JSF 129–130
- JSR 116 4, 6
- JSR 116 specification 124
- JSR 168 4, 6, 124, 128
- JSR-101 247
- JSR-115 15
- JSSE 257
- JUnit 130
- Just-In-Time (JIT) compiler 170
- JVM 74, 184
- JVM 5.0 263

- JVM for HP-UX 200
- JVM logs 166
- JVM security 263
- Jython 129, 154

K

- KEEP_ALIVE 184
- key management 256
- key sharing 265

L

- lbadm 291
- LDAP 4, 15, 18–19, 65, 81, 257, 266, 268
- LDAP registry 266, 279
- legal 18
- Lightweight Directory Access Protocol (LDAP) 18, 266
- Lightweight Third Party Authentication (LTPA) 14, 108, 264
- Load Balancer 4, 6, 64, 67, 78, 80, 285–286, 288, 291–293
- load balancer 67, 181
- load balancers 175
- load balancing 63, 73–74
- load factors 33
- load tests 33
- local operating system user registry 266
- Log Analyzer tool 166
- logical security 259
- LoginContext 275
- logs 165
- long and short names 105
- Lotus Domino Enterprise Server 10–11, 266
- LTPA 14, 257, 265–266, 268, 279–280, 282
- LTPA token 256, 265

M

- maintainability 183
- managed node 44, 54, 95, 117, 119
- managed Web server 54–55
- manageprofiles.bat(sh) 101
- Managers 126
- mapping applications 150
- master repository 158
- match criteria 186–187
- maximum heap size 144–145
- maximum in-memory session count 195, 197

- MBeans 159
- mediation 224, 245
- memory utilization 33
- memory-to-memory replication 181–182, 195, 197
- memory-to-memory session replication 291
- message bean 224
- message exchange
 - composite 243
 - one-way 240
 - publish-subscribe 242
 - request-response 241
 - two-way 240
 - workflow 242
- message exchange patterns 240
- message ordering 211
- message point 216
- message store 210, 225, 228
- message-driven bean (MDB) 219
- messaging 210
- messaging engine 175, 185, 187, 216, 218, 220, 222, 224, 227
- messaging security 225
- messaging service provider 214
- messaging service type 212
- method level security 270
- Microsoft Active Directory 16, 19
- Microsoft Active Directory Server 266
- Microsoft SQL Server 11
- migration 109
- mixed node 48
- MQ client 223
- MQ link 224
- MTBF 179
- MTTR 179

N

- naming conventions 88, 90, 102, 105, 133
- NAS 181–183, 185
- native code 166
- Network Dispatcher 76–77, 81, 286
- network interface card 180
- node 42–43, 103
- node agent 42, 44, 55, 184, 265
- node group 42
- node scope 161
- non-functional requirements 32
- non-root installation 107
- Notification Broker Service 234

- Novell eDirectory 11
- Number of alarm threads 179

O

- Object Request Broker (ORB) 263
- one-way message 240
- operating systems 9
- Oracle 10
- ORB service thread pool 201–202
- overwrite session management 197

P

- packaging 5
- PAM 274
- parallel processing 48
- parallel start 159
- partitioned queue 220
- pattern
 - observer 242
- patterns 33
- performance 60, 76, 142
- Performance Advisor 201
- performance management 31
- Performance Monitoring Infrastructure (PMI) 199
- performance requirements 32
- performance tuning 32
- persisted session size 201–202
- persisted session time 201–202
- physical security 259
- planning requirements 26
 - capacity 26
 - functional 26
 - non-functional 26
- Plants by WebSphere 291
- Pluggable Authentication Module 274
- pmt.bat(sh) 101
- policy 185, 187
- port assignment 105
- portal application development 131
- portlet 6
- portlet container 124
- portlet URL security 254
- power failure 180
- prepared statement cache size 201–202
- prepared statements 172
- privacy 18, 267
- private key 256
- private UDDI registry 248

- problem management 165
- process availability 180
- process choreography 232
- process isolation 63
- process logs 166
- processing model 244
- processor utilization 33
- profile 40, 86, 90–91, 94, 102
- profile creation 280
- profile creation wizard 95, 98, 101
- Profile Management Tool 87, 95–98, 101, 288–289
- Profile Management Tool for z/OS (zPMT) 101
- programmatic security 274
- protocol transformation 247
- proxy 185
- proxy server 71
- public key infrastructure (PKI) 268
- public UDDI registry 248
- publication point 217
- publish-subscribe message 242

Q

- QSECOFR 107
- quality of service 246
- quality of service attribute 227
- queue destination 216, 220
- queue manager 223–224
- queue point 216–217
- queue sharing group 211, 223–224

R

- RACF 255, 280–281
- RAID 180
- rapid deployment 132
- Rational Application Developer 6–7, 127, 129–130, 274
- Rational ClearCase 129–130, 135
- Rational ClearCase LT 135
- Rational ClearCase LT Server 131
- Rational ClearQuest 135
- Rational Enterprise Suite Tools 135
- Rational RequisitePro 135
- Rational Rose 135
- Rational Unified Process 131, 135
- Rational Unified Process (RUP) 125
- Rational Web Developer 7, 127, 129–130
- recovery plan 37
- recovery strategy 37

- Redbooks Web site 298
 - Contact us xv
- redundancy 62, 64, 79
- Redundant Array of Independent Disks (RAID) 37
- registration authority 268
- registry 238
- reliability 61, 227
- replication domain 195–197
- request filtering 203
- request metric 202–204
- request metrics 202
- request rate 33
- request-response 241
- resource adaptor 213
- resource scope 160
- response time 61, 171
- restoreConfig command 167
- retina scan 264
- reverse proxy 15, 275
- reverse proxy server 71–72
- risk analysis 36
- RMI/IIOP 178
- RoleConfiguration 276
- RoleConfigurationFactory 276
- rollout update 150
- round-robin routing policy 178
- Runtime Advisor 201
- RUP 126, 135, 137

S

- SAAJ 1.2 233
- SAF 263, 280–281
- SAF EJBROLE 281
- SAN 182
- SAS 271
- scalability 27, 170–171, 179, 284
- scaling 61, 174
- scaling techniques 27–28
- scaling-out 30
- Scheduler service 179
- scheduling tasks 179
- SCM 131, 133, 138, 141
- scope 160, 162
- scripting client 156
- scripting program 156
- SCSI connections 182
- Secure Authentication Service (SAS) 271
- secure message transportation 226

- Secure Sockets Layer (SSL) 211, 267
- security 60, 69–70, 90, 107, 210
 - single sign-on (SSO)
- security integration 197
- security policy 269
- security roles 272
- self-signed certificate 256
- serialize session access 197
- servant region (SR) 53
- server affinity 192
- server configuration template 159
- server ID 254, 279
- server scope 161
- server startup 158, 166
- server time 268
- server weighted routing policy 178
- servers
 - database 10
 - directory 11
- service broker 237–238
- service consumer 237
- Service Data Objects 130
- service integration bus 175, 185, 187, 210, 216, 222, 225, 247, 249
- service level agreement (SLA) 202
- service producer 237
- service provider 237
- service provider interface (SPI) 266
- service registry 238
- service requestor 237
- service-oriented architecture 246
- service-oriented architecture (SOA) 232
- servlets 129
- session affinity 192, 194
- session cache size 201–202
- session data 181
- session ID 192–193
- Session Initiation Protocol (SIP) 4, 124
- session management 191, 193, 195, 197
- session manager 192, 197
- session persistence 194, 199
- session timeout 195, 197
- session tracking 193
- shopping cart 235
- Showlog 166
- silent installation 86, 93
- Simple WebSphere Authentication Mechanism (SWAM) 14, 264
- single point of failure 74, 183, 195
 - see SPOF
- single points of failure 62
- single sign-on
 - see SSO
- single sign-on (SSO) 254, 265
- single-sign on 258
- singleton process 180–181
- SIP 125, 127
- SIP archive (SAR) files 125
- sizing 29–30, 32, 60
- SLAPD server 19
- snoop servlet 291
- SOAP 238, 244
- SOAP/HTTP 240
- SOAP/JMS 240
- Software Asset Management 135
- software cluster 182
- Source Code Management 131
- Source Code Management (SCM) 135
- SPNEGO 254, 258
- SPOF 180
- SSL 226, 256–258, 264, 267, 280–281
- SSL ID tracking 194, 197
- SSL session ID 193–194
- SSLContext instances 257
- SSLSocketFactories 257
- SSLV3TIMEOUT 194
- SSO 265, 267–268
- stand-alone application server 40, 42
- stand-alone server environment 114
- startup order 159
- stateful 192
- stateful session bean 197–198
- stateful session EJB 181–182
- stateless 191
- stderr 166
- stdout 166
- Storage Area Network (SAN) 181
- Struts 129
- Sun HotSpot JVM 170, 200
- Sun Java System Web Server 10
- Sun ONE Directory Server 11, 266
- support 167
- SURROGAT class profile 255
- Surrogate-Capabilities header 188
- SWAM 265
- Sybase 11
- Sync to OS Thread 255
- synchronization 120

- synchronous 241
- syncNode command 164
- sysplex node group 42
- SystemErr 151, 165–166
- SystemOut 151, 165–166, 204
- SystemSSL 264

T

- TAI 275
- team collaboration 134
- template 159
- test connection service 162
- test environment 33, 139, 141–142
- test results 33
- thin administrative client 154
- thread pool 173, 199, 201–202
- throughput 61, 76, 171
- time zone 268
- Tivoli Access Manager 14–17, 276
- Tivoli Access Manager policy server 16
- Tivoli Access Manager Server 4
- Tivoli Access Manager Servers for WebSphere Application Server 8
- Tivoli Directory Server 4, 11, 16, 18–19, 67, 266
- Tivoli Directory Server for WebSphere Application Server 8
- Tivoli Performance Viewer 200
- Tool Mentors 127
- topic space 217
- topology 59
- traces 165
- tracing 203
- transaction affinity 192
- transaction manager 184–185
- transaction server 27
- transport chain 225
- Transport Layer Security (TLS) 264
- transport level authentication 264
- trust association 15
- trust association interceptor 275
- trust association model 275
- trust policy 268
- tuning 172–173
- two-way message 240

U

- UBR 248
- UCM 135, 137

- UDDI 238, 246–247
- UDDI registry 247–248
- UML 135
- UML modeling 129, 131
- Unified Change Management (UCM) 135
- Unified Modeling Language 135
- Universal Description, Discovery, and Integration
 - see UDDI
- unmanaged node 54, 117
- unmanaged Web server 44, 54, 56–57, 68
- Update Installer 86
- Update Installer (UPDI) 8
- update wizard 163
- updatePorts.ant 106
- URL rewriting 193–194, 197
- URLStreamHandlers 257
- usage patterns 33
- Use multi-row schema 195
- user registry 15, 90, 266, 268
- user repository 266
- UserRegistry 266–267
- utility JAR 133

V

- V5 default messaging provider 215
- variables 162
- Versioned Object Bases (VOB) 135
- vertical cluster 43
- vertical scaling 48, 61, 73–74, 170, 175
- virtual IP 181
- virtual private network (VPN) 264
- VOB 135

W

- warm backup 37
- WASPostUpgrade 110
- WASPreUpgrade 110
- Web client authentication 269
- Web container 173, 177–178, 182, 202
- Web container thread pool 173, 201–202
- Web module 270, 293
- Web server 27, 43, 54
- Web server node 44
- Web server plug-in 7, 44, 65, 68, 72, 75, 82, 111, 177, 183, 202, 256, 290
- Web server plug-in remote versus local install 111
- Web server plug-ins 86
- Web servers 10

Web service gateway 245–247
Web services 130, 231–232
Web Services Description Language
 see WSDL
Web services engine 202
Web services gateway 245, 249
Web Services Inspection Language
 see WSIL
Web services request 203
WebSEAL Reverse Proxy Server 16
WebSphere Application Server - Express 5, 40
WebSphere Application Server Network Deployment 6, 41
WebSphere Application Server V6.1 for z/OS 7
WebSphere Enterprise Service Bus 3, 246
WebSphere Information Integrator 11
WebSphere Message Broker 246
WebSphere Message Queuing Interface (MQI) 224
WebSphere MQ 4, 20, 214, 222
WebSphere MQ link 22, 217, 223–224
WebSphere MQ messaging provider 214
WebSphere MQ on z/OS 223
WebSphere MQ server 22, 211, 223–224
WebSphere partitioning facility 184
WebSphere Performance Advisors 201
WebSphere Process Server 3
WebSphere rapid deployment 128, 131, 163
WebSphere request metrics 204
WebSphere workload management 61
weighted round-robin algorithm 73
weights 177
Windows Active Directory 11
WLM 61, 200
WLM for WebSphere z/OS 176
workflow 242
workload 28, 171
workload distribution 40, 43
workload management 40, 61, 63, 176, 178, 182, 192, 220, 227, 284
workload management (WLM) 176
workload manager 175
workload patterns 28
wsadmin 154, 156, 163
WS-AT 233
WS-BA 233
WSDL 238
WS-I Basic Profile 247
WS-I Basic Security Profile 254
WS-I BSP 233

WS-I Security 247
WSIL 238, 246
WS-Notification 234
WSRF-RAP 234
WS-Security 250, 254
WS-Transaction 247

X

X.500 directories 268
XML 129, 238

Z

z/OS Secure Authentication Service (z/SAS) 271
z/OS security 280
z/OS Security Server 11
z/OS workload manager 53
z/OS.e Security Server 11
z/SAS 271
zFS 87, 94
zPMT 97
zWLM 54, 176

Archived



WebSphere Application Server V6.1 : Planning and Design

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



WebSphere Application Server V6.1 Planning and Design



Discusses end-to-end planning for WebSphere implementations

Provides best practices

Includes a complex topology walk-through

This IBM Redbook discusses the planning and design of IBM WebSphere Application Server Version 6.1 environments. The content of this redbook is oriented to IT architects and consultants who require assistance when planning and designing small implementations to large and complex implementations.

This redbook addresses the packaging and features incorporated in WebSphere Application Server, covers the most common implementation topologies, and addresses planning for specific tasks and components that conform to the WebSphere Application Server environment.

The book includes planning information for WebSphere Application Server V6.1 and WebSphere Application Server Network Deployment V6.1 on distributed platforms and WebSphere Application Server for z/OS. It does not cover WebSphere Application Server for i5/OS.

Note the following companion pieces to this book:

- ▶ *WebSphere Application Server V6.1: Technical Overview*, REDP-4191
- ▶ *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks