

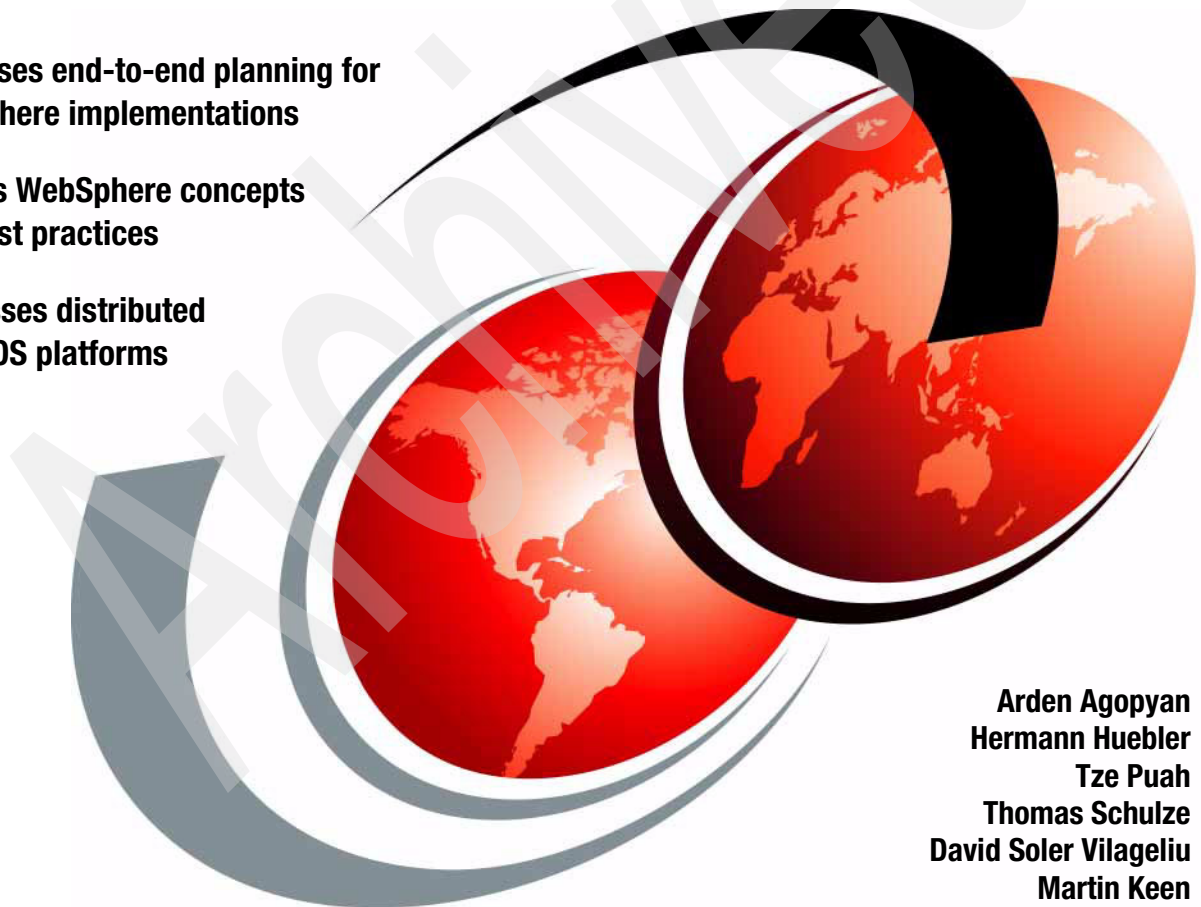
WebSphere Application Server V7.0:

Concepts, Planning, and Design

Discusses end-to-end planning for
WebSphere implementations

Defines WebSphere concepts
and best practices

Addresses distributed
and z/OS platforms



Arden Agopyan
Hermann Huebler
Tze Puah
Thomas Schulze
David Soler Vilageliu
Martin Keen



International Technical Support Organization

**WebSphere Application Server V7.0: Concepts,
Planning, and Design**

February 2009

Archived

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

First Edition (February 2009)

This edition applies to Version 7.0 of WebSphere Application Server.

© Copyright International Business Machines Corporation 2009. 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	xv
Trademarks	xvi
Preface	xix
The team that wrote this book	xix
Become a published author	xxii
Comments welcome	xxii
Chapter 1. Introduction to WebSphere Application Server V7.0	1
1.1 Java Platform Enterprise Edition (Java EE)	2
1.2 WebSphere Application Server overview	4
1.2.1 Application server purpose	4
1.2.2 Evolving Java application development standards	6
1.2.3 Enhanced management	7
1.2.4 Broader integration	8
1.2.5 Advanced tooling and extensions	11
1.3 Packaging	13
1.3.1 WebSphere Application Server - Express V7.0	14
1.3.2 WebSphere Application Server V7.0	15
1.3.3 WebSphere Application Server for Developers V7.0	15
1.3.4 WebSphere Application Server Network Deployment V7.0	15
1.3.5 WebSphere Application Server for z/OS V7.0	16
1.3.6 Packaging summary	17
1.4 Supported hardware, platforms, and software	18
1.4.1 Hardware	18
1.4.2 Operating systems	19
1.4.3 Web servers	21
1.4.4 Database servers	21
1.4.5 Directory servers	22
1.5 Related products	23
1.5.1 WebSphere Application Server Community Edition	23
1.5.2 WebSphere Extended Deployment	24
1.5.3 Rational Application Developer Assembly and Deploy V7.5	25
1.5.4 Rational Application Developer for WebSphere Software V7.5	26
1.5.5 Project Zero and WebSphere sMash	27
Chapter 2. Integration with other products	29
2.1 Tivoli Access Manager	30
2.1.1 Integration with WebSphere Application Server	30

2.2	Tivoli Directory Server	33
2.2.1	Lightweight Directory Access Protocol	33
2.2.2	Integration with WebSphere Application Server	34
2.3	WebSphere MQ	35
2.3.1	Integration with WebSphere Application Server	35
2.4	WebSphere Adapters	39
2.4.1	Integration with WebSphere Application Server	41
2.5	WebSphere DataPower	41
2.5.1	DataPower appliance models	43
2.5.2	Integration with WebSphere Application Server	45
2.6	DB2	46
2.6.1	Integration with WebSphere Application Server	46
2.7	Tivoli Composite Application Manager for WebSphere	47
2.7.1	Integration with WebSphere Application Server	48
2.7.2	ITCAM for WebSphere architecture	48
Chapter 3. WebSphere Application Server concepts		51
3.1	WebSphere Application Server concepts	52
3.1.1	Profiles	53
3.1.2	Stand-alone application servers	55
3.1.3	Distributed application servers	57
3.1.4	Nodes, node groups, and node agents	59
3.1.5	Cells	61
3.1.6	Deployment manager	62
3.1.7	Administrative agent	63
3.1.8	Job manager	64
3.1.9	Web servers	65
3.1.10	Proxy servers	67
3.1.11	Generic servers	70
3.1.12	Business level applications	70
3.1.13	Centralized installation manager	72
3.1.14	Intelligent runtime provisioning	72
3.2	Server environments	73
3.2.1	Single cell configurations	74
3.2.2	Multiple cell configurations	77
3.2.3	Mixed node versions in a cell	77
3.2.4	Flexible management	78
3.3	Clusters	81
3.3.1	Application server clusters	81
3.3.2	Proxy server clusters	87
3.3.3	Generic server clusters	87
3.4	Runtime processes	88
3.4.1	Distributed platforms	88

3.4.2	WebSphere Application Server for z/OS	89
3.5	Using Web servers	89
3.5.1	Managed Web servers	89
3.5.2	Unmanaged Web servers	90
3.5.3	IBM HTTP Server as an unmanaged Web server (special case) . . .	92
Chapter 4.	Infrastructure	93
4.1	Infrastructure planning	94
4.2	Design considerations	95
4.2.1	Scalability	96
4.2.2	Caching	98
4.2.3	High availability	99
4.2.4	Load-balancing and fail-over	100
4.2.5	Disaster recovery	101
4.2.6	Security	102
4.2.7	Application deployment	104
4.2.8	Servicability	105
4.3	Sizing the infrastructure	106
4.4	Benchmarking	107
4.5	Performance tuning	108
4.5.1	Application design issues	108
4.5.2	Understand your requirements	109
4.5.3	Test environment setup	109
4.5.4	Load factors	110
4.5.5	Production system tuning	112
4.5.6	Conclusions	113
4.6	Planning for monitoring	114
4.6.1	Environment analysis for monitoring	114
4.6.2	Performance and fault tolerance	116
4.6.3	Alerting and problem resolution	116
4.6.4	Testing	117
4.7	Planning for backup and recovery	117
4.7.1	Risk analysis	117
4.7.2	Recovery strategy	117
4.7.3	Backup plan	118
4.7.4	Recovery plan	118
4.7.5	Update and test process	119
4.8	Planning for centralized installation	119
Chapter 5.	Topologies	121
5.1	Topology selection criteria	122
5.1.1	High availability	122
5.1.2	Disaster recovery	125

5.1.3 Security	126
5.1.4 Maintainability	126
5.1.5 Performance	127
5.1.6 Application deployment	129
5.1.7 Summary: Topology selection criteria	133
5.2 Terminology	134
5.2.1 Load balancers	134
5.2.2 Reverse proxies	135
5.2.3 Domain and protocol firewall	136
5.2.4 Web servers and WebSphere Application Server Plug-in	137
5.2.5 Application servers	138
5.2.6 Directory and security services	138
5.2.7 Messaging infrastructure	138
5.2.8 Data layer	139
5.3 Topologies in detail	139
5.3.1 Stand-alone server topology	140
5.3.2 Vertical scaling topology	143
5.3.3 Horizontal scaling topology	147
5.3.4 Reverse proxy topology	154
5.3.5 Topology with redundancy of multiple components	160
5.3.6 Heterogeneous cell	166
5.3.7 Multi-cell topology	168
5.3.8 Advanced topology using an administrative agent	171
5.3.9 Advanced topology using a job manager	174
Chapter 6. Installation	179
6.1 What is new in V7.0	180
6.2 Selecting a topology	181
6.3 Selecting hardware and operating systems	181
6.4 Planning for disk space and directories	182
6.5 Naming conventions	184
6.6 Planning for the load balancer	184
6.6.1 Installation	185
6.6.2 Configuration	185
6.7 Planning for the DMZ secure proxy	186
6.8 Planning for the HTTP server and plug-in	187
6.8.1 Stand-alone server environment	191
6.8.2 Distributed server environment	194
6.9 Planning for WebSphere Application Server	197
6.9.1 File systems and directories	198
6.9.2 Single install or multiple installations	199
6.9.3 Installation method	201
6.9.4 Installing updates	203

6.9.5 Profile creation	205
6.9.6 Naming convention	220
6.9.7 TCP/IP port assignments	222
6.9.8 Security considerations	223
6.10 IBM Support Assistant	226
6.11 Summary: Installation checklist	227
Chapter 7. Performance, scalability, and high availability	229
7.1 What is new in V7.0	230
7.1.1 Runtime provisioning	230
7.1.2 Java SE 6	230
7.1.3 DMZ secure proxy	231
7.1.4 Flexible management	231
7.2 Scalability	231
7.2.1 Scaling overview	232
7.2.2 Scaling the system	233
7.3 Performance	235
7.3.1 Performance evaluation	235
7.3.2 System tuning	236
7.3.3 Application environment tuning	237
7.3.4 Application tuning	240
7.4 Workload management	241
7.4.1 HTTP servers	241
7.4.2 DMZ proxy servers	241
7.4.3 Application servers	241
7.4.4 Clustering application servers	243
7.4.5 Scheduling tasks	245
7.5 High availability	246
7.5.1 Overview	246
7.5.2 Hardware high availability	247
7.5.3 Process high availability	247
7.5.4 Data availability	248
7.5.5 Clustering and failover technique	249
7.5.6 Maintainability	250
7.5.7 WebSphere Application Server high availability features	250
7.6 Caching	255
7.6.1 Edge caching	256
7.6.2 Dynamic caching	257
7.6.3 Data caching	258
7.7 Session management	260
7.7.1 Overview	260
7.7.2 Session support	261
7.8 Data replication service	267

7.9	WebSphere performance tools	268
7.9.1	Performance monitoring considerations	268
7.9.2	Tivoli performance viewer	271
7.9.3	WebSphere performance advisors	271
7.9.4	WebSphere request metrics	273
7.10	Summary: Checklist for performance	276
Chapter 8. Application development and deployment		279
8.1	What is new in V7.0.	280
8.2	End-to-end life cycle	283
8.3	Development and deployment tools	285
8.3.1	Rational Application Developer for Assembly and Deploy V7.5	285
8.3.2	Rational Application Developer for WebSphere Software V7.5	286
8.3.3	WebSphere rapid deployment.	287
8.3.4	Which tools to use.	288
8.4	Naming conventions	288
8.4.1	Naming for applications.	288
8.4.2	Naming for resources	289
8.5	Source code management	290
8.5.1	Rational ClearCase	290
8.5.2	Concurrent Versions System (CVS)	291
8.5.3	Subversion	292
8.5.4	Choosing an SCM to use	292
8.6	Automated build process.	294
8.7	Automated deployment process	295
8.8	Automated functional tests	296
8.9	Test environments.	296
8.10	Managing application configuration settings	301
8.10.1	Classifying configuration settings	301
8.10.2	Managing configuration setting	302
8.11	Planning for application upgrades in production	305
8.12	Mapping application to application servers	307
8.13	Planning checklist for applications	309
Chapter 9. System management		311
9.1	What is new in V7.0.	312
9.2	Administrative security	314
9.3	WebSphere administration facilities	314
9.3.1	Integrated Solutions Console	316
9.3.2	WebSphere scripting client (wsadmin)	316
9.3.3	Task automation with Ant	317
9.3.4	Administrative programming	317
9.3.5	Command line tools	318

9.3.6	Administrative agent	318
9.3.7	Job manager	319
9.4	Automation planning	320
9.5	Configuration planning	321
9.5.1	Configuration repository location and synchronization	321
9.5.2	Configuring application and application server startup behaviors	322
9.5.3	Custom application configuration templates	323
9.5.4	Planning for resource scope use	323
9.6	Change management topics	326
9.6.1	Application update	326
9.6.2	Changes in topology	327
9.6.3	Centralized installation manager (CIM)	328
9.7	Serviceability	329
9.7.1	Log and traces	330
9.7.2	Fix management	331
9.7.3	Backing up and restoring the configuration	332
9.7.4	MustGather documents	332
9.7.5	IBM Support Assistant	333
9.7.6	Information Center	333
9.8	Planning checklist for system management	334
Chapter 10. Messaging		335
10.1	Messaging overview: What is messaging?	336
10.2	What is new in V7.0	336
10.3	Messaging options: What things do I need?	337
10.3.1	Messaging provider standards	338
10.3.2	Choosing a messaging provider	339
10.4	Messaging topologies: How can I use messaging?	340
10.4.1	Default messaging provider concepts	341
10.4.2	Choosing a messaging topology	344
10.5	Messaging features: How secure and reliable is it?	350
10.5.1	More messaging concepts	351
10.5.2	Planning for security	351
10.5.3	Planning for high availability	352
10.5.4	Planning for reliability	353
10.6	Planning checklist for messaging	355
Chapter 11. Web services		357
11.1	Introduction to Web services	358
11.2	What is new in V7.0	359
11.2.1	What was in Feature Pack for V6.1	360
11.2.2	Features added to WebSphere Application Server V7.0	360
11.3	Important aspects in using Web services	361

11.4	Web services architecture	363
11.4.1	How can this architecture be used?	365
11.5	Support for Web services in WebSphere Application Server	371
11.5.1	Supported standards	372
11.5.2	Service integration bus	372
11.5.3	Universal Description, Discovery, and Integration registries	373
11.5.4	Web services gateway	374
11.5.5	Security	375
11.5.6	Performance	375
11.6	Planning checklist for Web services	376
Chapter 12. Security		379
12.1	What is new in V7.0.	380
12.2	Security in WebSphere Application Server	381
12.2.1	Authentication	385
12.2.2	Authorization	392
12.2.3	Secure communications	395
12.2.4	Application security	396
12.2.5	Security domains	399
12.2.6	Auditing	401
12.3	Security configuration considerations	402
12.4	Planning checklist for security	405
Chapter 13. WebSphere Application Server Feature Packs		407
13.1	Available feature packs	408
13.2	WebSphere Application Server Feature Pack for Web 2.0	408
13.2.1	Introduction to Web 2.0	408
13.2.2	Overview of the Web 2.0 feature pack	409
13.2.3	Security considerations	411
13.2.4	Resources	411
13.3	WebSphere Application Server Feature Pack for Service Component Architecture	412
13.3.1	Introduction to SCA	412
13.3.2	Overview of the SCA feature pack	415
13.3.3	Other considerations	417
13.3.4	Resources	417
Chapter 14. WebSphere Application Server for z/OS		419
14.1	WebSphere Application Server structure on z/OS	420
14.1.1	Value of WebSphere Application Server on z/OS	420
14.1.2	Common concepts	421
14.1.3	Cell component—daemon	422
14.1.4	Structure of an application server	422
14.1.5	Runtime processes	425

14.1.6	Workload management for WebSphere Application Server for z/OS .	428
14.1.7	Benefits of z/OS	431
14.2	What is new in V7.0.	432
14.3	WebSphere Application Server 64-bit on z/OS	433
14.3.1	Overview	433
14.3.2	Planning considerations	434
14.3.3	Administration considerations	435
14.4	Load modules in the HFS	436
14.4.1	Overview	436
14.4.2	Installation considerations.	437
14.4.3	HFS structure	438
14.5	XCF support for WebSphere HA manager	438
14.5.1	XCF support overview and benefits	438
14.5.2	WebSphere HA manager	439
14.5.3	Default core group discovery and failure detection protocol	441
14.5.4	XCF—alternative protocol on z/OS.	442
14.5.5	Activating XCF support for HA manager	444
14.6	z/OS Fast Response Cache Accelerator.	446
14.6.1	Overview and benefits.	446
14.6.2	Configuring FRCA.	447
14.6.3	Monitoring FRCA.	451
14.6.4	Resource Access Control Facility (RACF) integration	452
14.7	Thread Hang Recovery	452
14.7.1	Overview	452
14.7.2	Pre-WebSphere Application Server V7.0 technique	453
14.7.3	WebSphere Application Server V7.0 technique	453
14.7.4	New properties	454
14.7.5	Display command	456
14.8	Installing WebSphere Application Server for z/OS	456
14.8.1	Installation overview	457
14.8.2	Installation considerations.	457
14.8.3	Function Modification Identifiers	460
14.8.4	SMP/E installation.	460
14.8.5	Customization	461
14.8.6	Execute customization jobs.	465
14.9	System programmer considerations	466
14.9.1	Systems Management Facility enhancements	466
14.9.2	WebSphere Application Server settings	467
14.9.3	Java settings	469
14.9.4	Basic WLM classifications	472
14.9.5	Address Space ID reuse	473
14.9.6	Deprecated features WebSphere Application Server for z/OS	474

14.9.7 Java Command Language (JACL) stabilized	474
14.9.8 Application profiling	475
14.10 Planning checklist	475
14.10.1 Planning considerations	476
14.10.2 Resources	477
Chapter 15. Migration	479
15.1 Infrastructure migration considerations	480
15.1.1 Scripting migration	482
15.1.2 HTTP server plug-in support	482
15.1.3 Coexisting versions on one system	482
15.1.4 Runtime inter operability	483
15.1.5 Runtime migration tools	483
15.1.6 Mixed cell support	484
15.1.7 Network Deployment migration strategies	484
15.2 Application development migration considerations	486
15.3 System management migration considerations	487
15.4 Messaging migration considerations	488
15.5 Web services migration considerations	488
15.6 Security migration considerations	489
15.7 WebSphere Application Server for z/OS migration considerations	490
15.7.1 Migration and coexistence	490
15.7.2 General considerations	490
15.7.3 Migration process overview	491
15.7.4 z/OS Migration Management Tool	491
15.7.5 Migration Management Tool script	495
15.7.6 Migration jobs	497
15.7.7 Migration considerations for 64-bit mode	499
Appendix A. Sample topology walkthrough	501
Topology review	502
Advantages	504
Disadvantages	504
Sample topology	504
Characteristics	505
Installation	507
Installing the Load Balancer (server A)	507
Installing HTTP Servers (servers B and C)	508
Creating Deployment manager (server D)	508
Creating Application servers (servers D and E)	508
Creating Job manager (server E)	510
Deploying applications	510
Configuring security	511

Testing the topology	513
Service	513
Administration	516
Summary	520
Related publications	521
IBM Redbooks	521
Online resources	522
How to get Redbooks	522
Help from IBM	522

Archived

Archived

Notices

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

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

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

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

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

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

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

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

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


COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX 5L™	i5/OS®	Rational Rose®
AIX®	IBM®	Rational®
Build Forge®	Informix®	Redbooks®
CICS®	iSeries®	Redbooks (logo)  ®
ClearCase MultiSite®	Language Environment®	RequisitePro®
ClearCase®	Lotus®	System i®
ClearQuest®	OMEGAMON®	System z®
DataPower®	Parallel Sysplex®	Tivoli®
DB2 Connect™	POWER®	VTAM®
DB2®	PR/SM™	WebSphere®
developerWorks®	Processor Resource/Systems	z/OS®
Domino®	Manager™	zSeries®
HACMP™	RACF®	

The following terms are trademarks of other companies:

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

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

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

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

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

EJB, Enterprise JavaBeans, J2EE, J2SE, Java, JavaBeans, Javadoc, JavaScript, JavaServer, JDBC, JDK, JMX, JNI, JRE, JSP, JVM, Solaris, Sun, Sun Java, ZFS, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

Archived

Archived

Preface

This IBM® Redbooks® publication discusses the concepts, planning, and design of WebSphere® Application Server V7.0 environments. This book is aimed at IT architects and consultants who want more information for the planning and designing of application-serving environments, ranging from small to large, and complex implementations.

This IBM Redbooks publication addresses the packaging and the features incorporated into 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 for WebSphere Application Server V7.0 and WebSphere Application Server Network Deployment V7.0 on distributed platforms, and WebSphere Application Server for z/OS V7.0. It also covers migration considerations for migrating from previous releases.

The team that wrote this book

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

Arden Agopyan is an IT Specialist for WebSphere Technical Sales working at Software Group, IBM Turkey since 2006. He has worked with WebSphere Application Server since V5.1 and is an expert on planning, design, implementation, and problem determination of WebSphere Application Server solutions. Before joining IBM he worked as a senior Java™ and .NET solutions developer. He holds a Computer Engineer degree from Galatasaray University in Istanbul (Turkey).

Hermann Huebler is an IT Specialist working for IBM Global Services Strategic Outsourcing Service Delivery in Austria. He has 21 years of experience in IT and is working for IBM since 1994. After working as a System i® specialist for several years, he started focusing on WebSphere products on distributed platforms in 2001. His main areas of expertise are implementation, problem determination, high availability, and performance tuning of the WebSphere Application Server product family. These products include WebSphere Application Server, WebSphere Portal, WebSphere MQ, Edge Components.

Tze Puah is an Application Infrastructure Services Principal working for Coles Group Ltd in Melbourne, Australia since 2005. His areas of expertise include infrastructure architecture design, implementation, problem determination and performance tuning on WebSphere and Tivoli® products. These products include WebSphere Application Server, WebSphere Portal Server, WebSphere Process Server, WebSphere Commerce Server, WebSphere MQ, WebSphere Message Broker and Tivoli Access Manager. He has 14 years of experience in IT and holds a master degree in Computer Science.

Thomas Schulze is a Senior IT Specialist working since 2003 in System z® pre-sales support in Germany. His areas of expertise include WebSphere Application Server running on the System z platform with a speciality on performing Healthchecks and Performance Tuning. He has written extensively on the z/OS® parts of this book. He has 10 years of experience in IBM and holds a degree in Computer Science from the University of Cooperative Education in Mannheim (Germany).

David Soler Vilageliu is an IT Architect working for IBM Global Services in Barcelona, Spain. He has more than 19 years of experience playing different technical roles in the IT field and he is working for IBM since 2000. His areas of expertise include WebSphere Application Server, ITCAM for WebSphere, ITCAM for Response Time, and Unix systems. He holds a degree in Computer Science from the Universitat Politècnica de Catalunya (Spain).

Martin Keen is a Senior IT Specialist at the ITSO, Raleigh Center. He writes extensively about WebSphere products, and SOA. He also teaches IBM classes worldwide about WebSphere, SOA, and ESB. Before joining the ITSO, Martin worked in the EMEA WebSphere Lab Services team in Hursley, UK. Martin holds a bachelor's degree in Computer Studies from Southampton Institute of Higher Education.



Figure 0-1 Team (left-to-right): Tze, Arden, David, Martin, Thomas, and Hermann

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

Tom Alcott
IBM WorldWide BI Technical Sales, USA

Jeff Mierzejewski
WebSphere for z/OS Configuration Test, USA

Dustin Amrhein
WebSphere Web Services Development
IBM Software Group, Application and Integration Middleware Software, USA.

Aaron Shook
IBM WebSphere Application Server Proxy Development, USA.

Keys Botzum
IBM Software Services for WebSphere (ISSW), USA

Tiaoyu Wang
IBM Application and Integration Middleware Software Development, Canada

David Follis
IBM WebSphere Application Server for z/OS Runtime Architect, USA

Holger Wunderlich
IBM Field Technical Sales Support, z/OS WebSphere Solutions, Germany

Mark T. Schleusner
IBM WebSphere Serviceability Development, USA

Matthew Leming
IBM WebSphere Messaging Development, UK

Graham Wallis
IBM Senior Technical Staff Member, WebSphere Messaging, UK

Gerhard Poul
IBM Software Services for WebSphere, Austria

Brian Stelzer
IBM AIM Early Programs - WebSphere Application Server, USA

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

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review Redbooks 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



Introduction to WebSphere Application Server V7.0

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

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

IBM WebSphere is designed to enable you to build business-critical enterprise applications and solutions as well as combining them with innovative new functions. WebSphere includes and supports a wide range of products that help you develop and serve your business applications. They are designed to make it easier for clients to build, deploy, and manage dynamic Web sites and other more complex solutions productively and effectively.

This chapter introduces WebSphere Application Server V7.0 for distributed platforms and z/OS, and highlights other IBM software products related with WebSphere Application Server.

1.1 Java Platform Enterprise Edition (Java EE)

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, applications, and services with internal and external resources. WebSphere is infrastructure software, or middleware, designed for dynamic On Demand Business and for enabling SOA for your enterprise. It delivers a proven, secure, robust, and reliable software portfolio that provides an excellent return on investment.

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 Platform, Enterprise Edition (Java EE) platform. This contrasts with the Java Platform, Standard Edition (Java SE) platform, with which most clients are familiar. Java SE supported the development of client-side applications with rich graphical user interfaces (GUIs). The Java EE platform, built on top of the Java SE platform, provides specifications for developing multi-tier enterprise applications with Java. It consists of application technologies for defining business logic and accessing enterprise resources such as databases, enterprise resource planning (ERP) systems, messaging systems, internal and external business services, e-mail servers, and so forth.

Java Platform name changes: Java EE and Java SE were formerly named Java 2 Platform, Enterprise Edition (J2EE™) and Java 2 Platform, Standard Edition (J2SE™) respectively.

After this name change, J2EE 1.5 is renamed as Java EE 5. J2SE 6 is renamed as Java SE 6. This book will cover these standards with their new names.

The potential value of Java EE to clients is tremendous. This includes the following benefits:

- ▶ An architecture-driven approach allowing application development helps reduce maintenance costs and allows for construction of an information technology (IT) infrastructure that can grow to accommodate new services.
- ▶ Application development standards, tools and predefined rules improve productivity and accelerates and shortens development cycles.
- ▶ Packaging, deployment, and management standards for your enterprise applications facilitate systems and operations management.

- ▶ Industry standard technologies enable clients to choose among platforms, development tools, and middleware to power their applications.
- ▶ Platform independence gives flexibility to create an application once and to run it on multiple platforms. This provides true portability to enterprise applications.
- ▶ Embedded support for Internet and Web technologies allows for applications that can bring services and content to a wider range of customers, suppliers, and others, without creating the need for proprietary integration.

Java EE 5, the latest release of the Java EE platform, represents a significant evolution in the Java enterprise programming model. It improves the application developer experience and productivity with following new features:

- ▶ Latest specifications
 - Enterprise JavaBeans™ (EJB™) 3.0
 - JAX-WS 2.0
 - JSP™ 2.1
 - Servlet 2.5
 - JSF 1.2
- ▶ Java Development Kit (JDK™) 6.0
- ▶ Usage of progressive disclosure
- ▶ Annotations and injection support to reduce complexity
- ▶ EJB development as Plain Old Java Objects (POJOs)
- ▶ Java Persistence API (JPA) to allow creation of simpler entities using annotated POJO model

Another exciting opportunity for IT is Web services. Web services allow for the definition of functions or services within an enterprise that can be accessed using industry standard protocols (such as HTTP and XML) already in use today. 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. Web services are also the key elements of SOA, which provides reuse of existing service components and more flexibility to enable businesses to address changing opportunities.

1.2 WebSphere Application Server overview

WebSphere Application Server is the IBM runtime environment for Java-based applications. This section gives an overview of the options and functionalities that WebSphere Application Server V7.0 offers:

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

1.2.1 Application server purpose

An application server provides the infrastructure for executing applications that run your business. It insulates the infrastructure from hardware, operating system, and the network (Figure 1-1). An application server also serves as a platform to develop and deploy your Web services and Enterprise JavaBeans (EJBs), and as a transaction and messaging engine while delivering business logic to end-users on a variety of client devices.

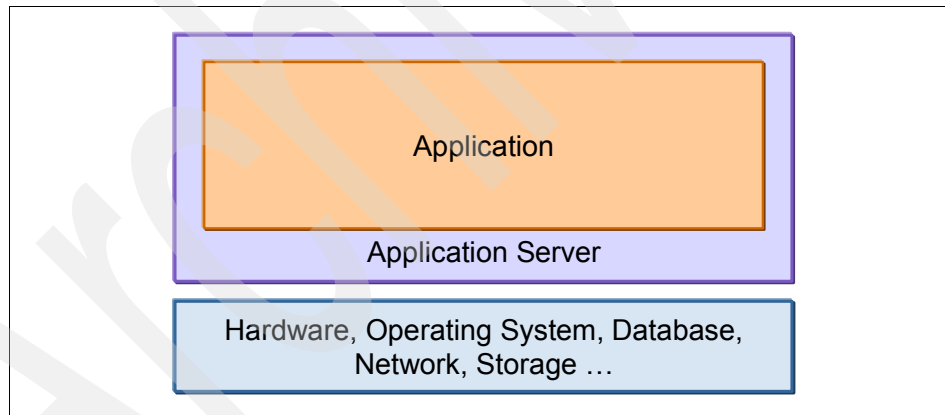


Figure 1-1 Basic presentation of an application server and its environment

The application server acts as middleware between back-end systems and clients. It provides a programming model, an infrastructure framework, and a set of standards for a consistent designed link between them.

WebSphere Application Server provides the environment to run your solutions and to integrate them with every platform and system as business application services conforming to the SOA reference architecture (Figure 1-2 on page 5).

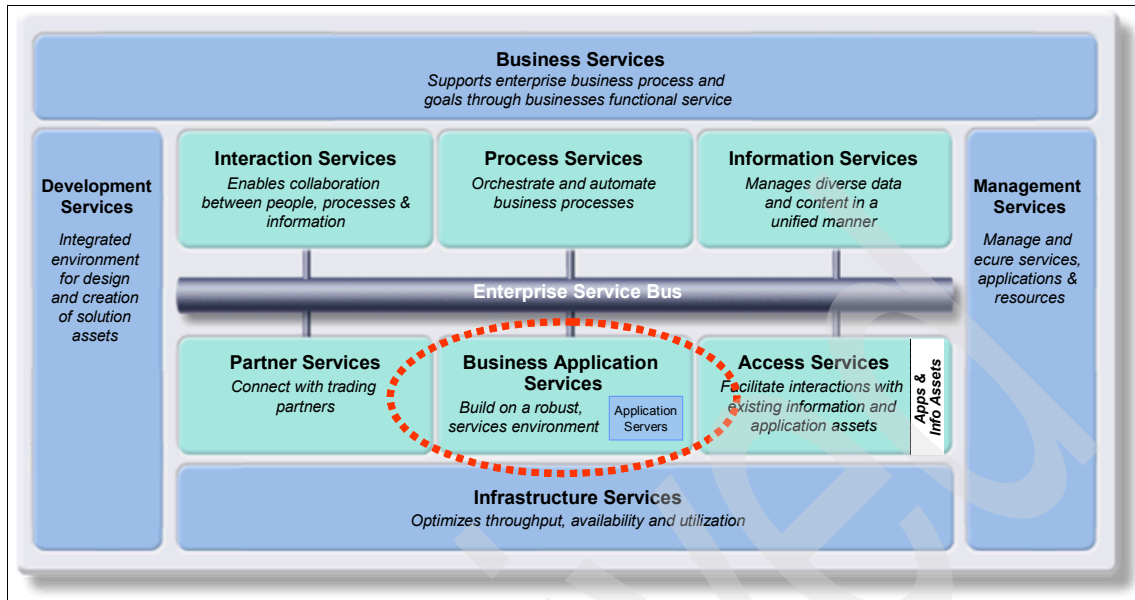


Figure 1-2 Position of Business Application Services in SOA reference architecture

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

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

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

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

1.2.2 Evolving Java application development standards

WebSphere Application Server V7.0 provides the runtime environment for applications that conform to the J2EE 1.2, 1.3, 1.4, and Java EE 5 (formerly J2EE 1.5) specifications.

Java SE V6 support adds the ability to invoke the Java compiler from within the Java Virtual Machine (JVM™) and includes scripts with the ability to access APIs within the JVM.

Feature Pack for EJB 3.0: The Feature Pack for EJB 3.0 available for WebSphere Application Server V6.1 is embedded into WebSphere Application Server V7.0.

If you have used EJB 3.0 Feature Pack functionality in your application in a WebSphere Application Server V6.1 environment before, there is no need to install additional WebSphere Application Server packages to use these applications with V7.0.

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

- ▶ **Java DataBase Connectivity (JDBC™) API 4.0**
Using the JDBC API, you can connect to any type of data sources.
- ▶ **Java Persistence API (JPA)**
JPA delivers simplified programming models and a standard persistence API for building reusable persistent objects.
- ▶ **Service Data Objects (SDO)**
With SDO, application programmers can uniformly access and manipulate data from heterogeneous data sources as collections of tree-structured or graph-structured objects. WebSphere Application Server V7.0 extends the application server to support the following tasks:
 - ▶ **Java Specification Requests (JSR) 286 (Portlet 2.0) compliant portlets**
 - ▶ **Session Initiation Protocol (SIP) applications conforming to the JSR 116 specification**
 - ▶ **Java Servlet 2.5 (JSR 154) and JavaServer™ Pages (JSR 245) specifications for Web applications**

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

1.2.3 Enhanced management

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

New management features

WebSphere Application Server V7.0 adds the following new management features to its packages:

- ▶ Flexible management components

These components allow you to build advanced and large-scale topologies while reducing management and maintenance complexity.

- Administrative agent

Manage multiple stand-alone servers from a central point.

- Job manager

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

- ▶ Centralized Installation Manager

This component provides the capability to perform centralized installations from the deployment manager to remote endpoints.

- ▶ Business Level Application definition

The new notion of defining applications, this allows you to group and manage Java EE and other related artifacts under a single application definition.

- ▶ Improved Console Command Assistant

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

- ▶ Enhanced administrative scripting for wsadmin

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

- ▶ Consolidated administration feature for WebSphere DataPower®

This component allows you to manage and integrate your WebSphere DataPower appliances into your environment.

Runtime provisioning

WebSphere Application Server's runtime provisioning mechanism allows the application server runtime to select only the needed functions for memory and space dynamically while running applications. Starting only the necessary components for an application reduces the server footprint and startup time.

Security management and auditing

WebSphere Application Server V7.0 adds value to your installations by providing the following security management and auditing improvements:

- ▶ A broader implementation of Kerberos and Single Sign-On features delivering improved interoperability with other applications and environments.
- ▶ Ability to create multiple security domains within a single WebSphere Application Server cell. Each security domain can have its own user population (and underlying repository). Additionally, the application domain can be separated from the administrative domain.
- ▶ Security auditing records the generation of WebSphere Application Server administrative actions. These actions can be security configuration changes, key and certificate management, access control policy changes, bus and other system resources management, and so on. This feature enables you to hold administrative users accountable for configuration and run time changes.
- ▶ The DMZ Secure Proxy, a proxy server hardened for DeMilitarized Zone (DMZ) topologies, allows you to have a more secure out-of-box proxy implementation outside the firewall.
- ▶ Fine-grained administration security can now be enforced through the administration console. You can restrict access based on the administrators' role at the cell, node, cluster, or application level, offering fine-grained control over administrator scope. This capability is valuable in large-cell implementations where multiple administrators are responsible for subsets of the application portfolio running on the cell.

1.2.4 Broader integration

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

Web services

WebSphere Application Server V7.0 includes support for the following Web services and Web services security standards:

- ▶ Web Services Interoperability Organization (WS-I) Basic Profile 1.2 and 2.0
- ▶ WS-I Reliable Secure Profile

- ▶ Java API for XML Web Services (JAX-WS)
- ▶ SOAP 1.2
- ▶ SOAP Message Transmission Optimization Mechanism (MTOM)
- ▶ XML-binary Optimized Packaging (XOP)
- ▶ WS-ReliableMessaging
- ▶ WS-Trust
- ▶ WS-SecureConversation
- ▶ WS-Policy

For centrally defining various quality-of-service policies you might want to apply to Web services that are already deployed, WebSphere Application Server V7.0 provides Web service policy sets. There are two types of policy sets:

- ▶ Application policy sets
These sets are used for business-related assertions, such as business operations that are defined in the Web Service Description Language (WSDL) file.
- ▶ System policy sets
These sets are used for non-business-related system messages, such as messages that are defined in specifications that apply quality of service policies. Examples include security token (RST) messages that are defined in WS-Trust, or the creation of sequence messages that are defined in WS-Reliable Messaging metadata.

Rather than defining individual policies and applying them on a Web service base, policy sets can be applied once to all Web service applications to which they are applicable, insuring a uniform quality of service for a given type of Web service. Related to this, WebSphere Service Registry and Repository, an additional WebSphere Software product, can discover WebSphere Application Server V7.0 JAX-WS policy sets and existing associations, and can represent those as policy attachments.

Web Services Feature Pack: The Feature Pack for Web Services available for WebSphere Application Server V6.1 is embedded into WebSphere Application Server V7.0.

If you have used Web Services Feature Pack functionality in your application in a WebSphere Application Server V6.1 environment before, there is no need to install additional WebSphere Application Server packages to use these applications with V7.0.

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

Messaging, connectivity, and transaction management

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

WebSphere Application Server also supports Java EE Connector Architecture (JCA) 1.5 resource adapters, which provides connectivity between application servers and Enterprise Information Systems (EIS). WebSphere Application Server V7.0 comes with Java Transaction API (JTA) 1.1 specification support, which provides standard Java interfaces for transaction management.

Authentication and authorization

WebSphere Application Server provides authentication and authorization capabilities to secure administrative functions and applications. Your choice of user registries include an operating system user registry, like the Resource Access Control Facility (RACF®) on z/OS, 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, WebSphere Application Server has support for Java Authorization Contract for Containers (JACC) 1.1. This gives you the option of using an external 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.

Application client

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

The Application Client for WebSphere Application Server Version 7.0 consists of the following client components:

- ▶ Java EE application client application
This component uses services provided by the Java EE Client Container.
- ▶ Thin application client application
This component does not use services provided by the Java EE Client Container and includes a JVM API.
- ▶ Applet application client application
This component allows users to access enterprise beans in the WebSphere Application Server through a Java applet in a HTML document.
- ▶ ActiveX® to EJB Bridge application client application
This component uses the Java Native Interface (JNI™) architecture to programmatically access the JVM API (Windows® only).

Web server support

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

1.2.5 Advanced tooling and extensions

This section discusses WebSphere Application Server tooling and extension enhancements.

Application development and deployment tools

WebSphere Application Server includes the Rational® Application Developer Assembly and Deploy V7.5 software platform, which is a set of tools based on Eclipse that allows you to develop and deploy Java EE 5 applications. It includes scripting tools to be used with WebSphere Application Server. WebSphere Application Server is also compatible with Rational Application Developer for WebSphere Software V7.5, which is available separately and fully exploits capabilities within WebSphere Application Server V7.0 as a Java EE 5 application development and deployment tool.

Note: Rational Application Developer Assembly and Deploy V7.5, Rational Application Developer V7.5 for WebSphere Software, and their differences are explained in 1.5, “Related products” on page 23.

Installation Factory

WebSphere Application Server includes the Installation Factory tool for creating customized install packages (CIPs). CIPs are used for packaging installations, updates, fixes, and applications to create custom and all-in-one installation solutions. Installation Factory is convenient for ISVs that want to create customized and bundled application solutions with WebSphere Application Server, as well as for organizations who prefer to have ready-to-deploy installation packages containing all the pre-installed artifacts. Consider using Installation Factory to create one or more CIPs and use those CIPs to deploy or update WebSphere throughout your organization.

Update Installer

WebSphere Application Server ships the Update Installer (UPDI) for installing maintenance packages, fix packs, interim fixes, and so on.

WebSphere Application Server Feature Packs

WebSphere Application Server Feature Packs simplify the adoption of new technology standards and make them available before the release of a new version of WebSphere Application Server. This strategy started with WebSphere Application Server V6.1 and continued with WebSphere Application Server V7.0. Unlike in version 6.1, Feature Pack for EJB 3.0 and Feature Pack for Web Services are embedded to WebSphere Application Server V7.0. At the time of writing, there are two feature packs available for WebSphere Application Server V7.0, as shown in Figure 1-3 on page 13:

- ▶ Feature Pack for Web 2.0

This feature pack enables you to build and run SOA services in a Web 2.0 application by offering Ajax and other Web 2.0-specific technical supports on the server, rather than the front-end support only.

- ▶ Feature Pack for Service Component Architecture (SCA)

This feature pack provides an implementation of the Open SCA specifications and a powerful programming model for constructing applications based on SOA.

About Open SCA specifications: The Open SCA specifications were designed by key technology vendors, including IBM, to address the service composition and assembly development needs of organizations adopting SOA.

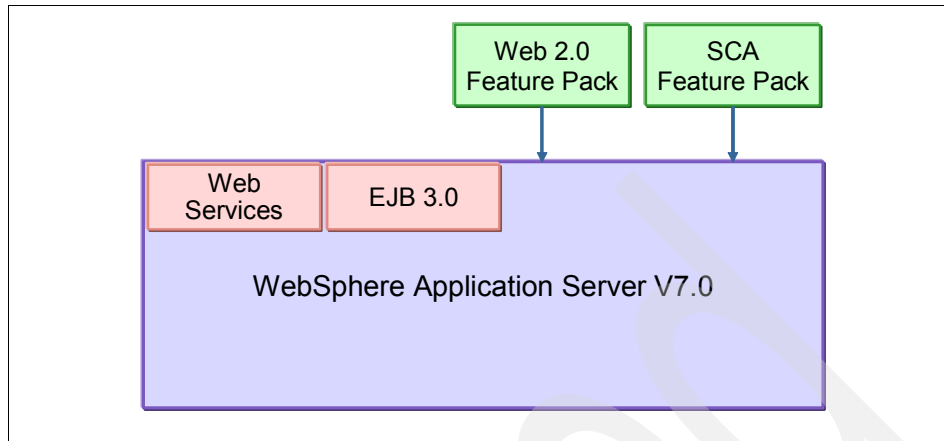


Figure 1-3 Feature Pack structure for WebSphere Application Server V7.0

1.3 Packaging

Because different 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. The following packages are available:

- ▶ WebSphere Application Server - Express V7.0
- ▶ WebSphere Application Server V7.0
- ▶ WebSphere Application Server for Developers V7.0
- ▶ WebSphere Application Server Network Deployment V7.0
- ▶ WebSphere Application Server for z/OS V7.0

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

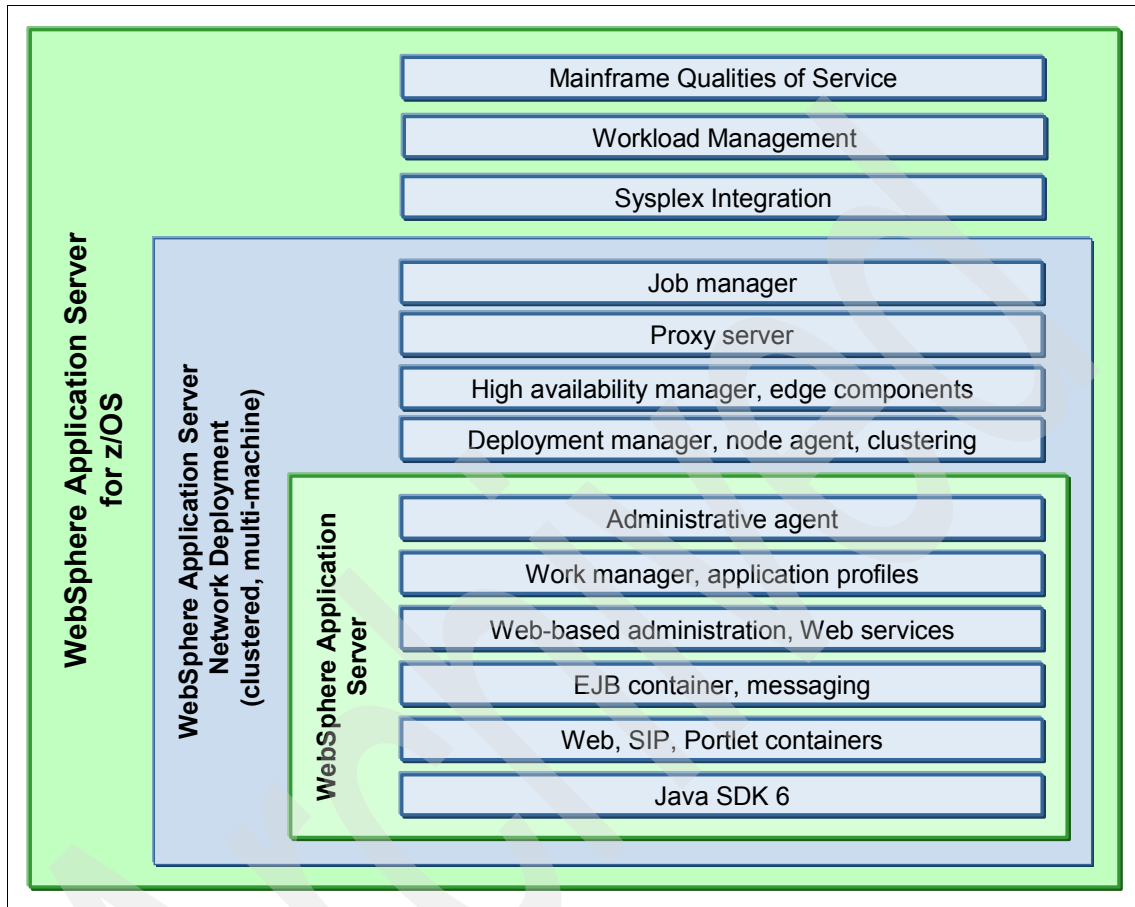


Figure 1-4 Packaging structure of WebSphere Application Server V7.0

1.3.1 WebSphere Application Server - Express V7.0

The Express package is geared to those who need to get started quickly with a strong and affordable application server based on standards. 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 Java EE 5, EJB 3.0, and Web services support but is limited to a 32-bit single-server environment and to a maximum of 240 Processor Value Units (PVUs) per server for licensing purposes.

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

For more information about WebSphere Application Server - Express V7.0, see the following Web page:

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

1.3.2 WebSphere Application Server V7.0

The WebSphere Application Server package is the next level of server infrastructure in the WebSphere Application Server family. Although the WebSphere Application Server package is functionally equivalent to Express (single-server environment), this package differs in packaging and licensing. It is available for both 32-bit and 64-bit platforms. This package is ideal for lightweight application solutions where cost and simplicity are key. This package is also referred to as the WebSphere Application Server V7.0 Base package.

For more information about WebSphere Application Server V7.0, see the following Web page:

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

1.3.3 WebSphere Application Server for Developers V7.0

The WebSphere Application Server for Developers package is functionally equivalent to the WebSphere Application Server package but it is licensed for development use only.

WebSphere Application Server for Developers is an easy-to-use development environment to build and test applications for your SOA.

1.3.4 WebSphere Application Server Network Deployment V7.0

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

- ▶ Clustering capabilities
- ▶ Edge components
- ▶ Dynamic scalability
- ▶ High availability
- ▶ Advanced management features 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 high availability requirements are in place.

WebSphere Application Server Network Deployment Edge Components provide high performance and high availability features. For example, the Load Balancer (a software load balancer) provides horizontal scalability by dispatching HTTP requests among several Web server or application server nodes supporting various dispatching options and algorithms to assure high availability in high volume environments. The usage of Edge Component Load Balancer can reduce Web server congestion, increase content availability, and provide scaling ability for the Web server.

WebSphere Application Server Network Deployment also includes a dynamic cache service, which improves performance by caching the output of servlets, commands, Web services, and JSP files. This cache can be replicated to the other servers. The state of dynamic cache can be monitored with the cache monitor application. For more information about WebSphere Application Server Network Deployment V7.0, see the following Web page:

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

1.3.5 WebSphere Application Server for z/OS V7.0

IBM WebSphere Application Server for z/OS is a full-function version of WebSphere Application Server Network Deployment. While it offers all the options and functions common to WebSphere Application Server V7.0 on distributed platforms, it enhances the product in a variety of ways:

- ▶ Defines Service Level Agreements (SLA) on a transaction base (response time per transaction).
- ▶ Protects your production with Workload Management, in times of unpretendable peaks.
- ▶ Uses z/OS functionality for billing based on used resources or transactions.
- ▶ Uses one central security repository, including Java role-based security.
- ▶ Builds a cluster inside of a single application server (multi-servant).
- ▶ Profits from near linear hardware and software scalability.
- ▶ Profits from System z cluster (Parallel Sysplex®) and up to 99.999% availability.

For more information about WebSphere Application Server for z/OS V7.0, see Chapter 14, “WebSphere Application Server for z/OS” on page 419 or visit the following Web page:

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

1.3.6 Packaging summary

Table 1-1 details the WebSphere Application Server features.

Table 1-1 WebSphere Application Server V7.0 Packaging

Features	Express	Base	Network Deployment	z/OS
Workload management within a server: integrated with z/OS Workload Manager (for SLA's on transactional level and reporting for chargeback)	No	No	No	Yes
Enterprise Java Beans 3.0	Yes	Yes	Yes	Yes
Full Java EE 5 support	Yes	Yes	Yes	Yes
Advanced security	Yes	Yes	Yes	Yes
Broad operating-system support and database connectivity	Yes	Yes	Yes	Yes
Advanced clustering	No	No	Yes	Yes
Integration with IBM Rational Application Developer Assembly and Deploy	Yes	Yes	Yes	Yes
Job manager and deployment manager	No	No	Yes	Yes
Rapid Java Development and Deployment Kit (JDK) 6.0	Yes	Yes	Yes	Yes
Runtime provisioning	Yes	Yes	Yes	Yes
Large-scale transaction support	No	No	Yes	Yes
Dynamic caching	No	No	Yes	Yes
Reporting and charge back: Granular reporting on resource consumption	No	No	No	Yes
WebSphere Application Server Feature Packs	Yes	Yes	Yes	Yes
Administrative agent	Yes	Yes	Yes	Yes
Edge components	No	No	Yes	Yes

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 hardware, platforms, and software

This section details the hardware, platforms, and software versions that WebSphere Application Server V7.0 supports at the time this book was written.

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

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

1.4.1 Hardware

Supported hardware for WebSphere Application Server V7.0 include the following:

- ▶ IBM POWER® family of processors
- ▶ PA-RISC processor
- ▶ Intel® Itanium® 2 processor
- ▶ Intel Pentium® processor at 500 MHz or faster
- ▶ Intel EM64T or AMD™ Opteron™
- ▶ IBM System z processors
- ▶ IBM System i (iSeries®)
- ▶ SPARC workstations

1.4.2 Operating systems

Table 1-2 shows supported operating systems and their versions for WebSphere Application Server V7.0

Table 1-2 Supported operating systems and versions

Operating systems	Versions
Microsoft® Windows	Supported with 32-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ Microsoft Vista Business ▶ Microsoft Vista Enterprise ▶ Microsoft Windows Server® 2003 with SP2 ▶ Microsoft Windows Server 2003 R2 ▶ Microsoft Windows Server 2008, Datacenter with SP1 ▶ Microsoft Windows Server 2008, Enterprise with SP1 ▶ Microsoft Windows Server 2008, Standard with SP1 ▶ Microsoft Windows XP Professional with SP2 Supported with 64-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ Microsoft Windows Server 2003 x64 R2 ▶ Microsoft Windows Server 2008 Enterprise x64 with SP1 ▶ Microsoft Windows Server 2008 Datacenter x64 with SP1 ▶ Microsoft Windows Server 2008 Standard x64 with SP1
IBM AIX®	Supported with 32-bit and 64-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ AIX 6.1 with Maintenance Package 6100-00-04 ▶ AIX 5L™ 5.3 with Service Pack 5300-07-01
Sun™ Solaris™ on Sparc	Supported with 32-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ Sun Solaris Version 9 with Patch Cluster dated 1/03/08 or later ▶ Sun Solaris Version 10 with Patch Cluster dated 1/07/08 or later Supported with 64-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ Sun Solaris Version 10 with Patch Cluster dated 1/07/08 or later
Sun Solaris on x64	Supported with 64-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ Sun Solaris Version 10 with Patch Cluster dated 1/07/08 or later
HP-UX on PA-RISC	Supported with 32-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ HP-UX 11iv2 with Patch Bundle dated Dec 2007 ▶ HP-UX 11iv3 with Patch Bundle dated Sep 2007
HP-UX on Itanium	Supported with 64-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ HP-UX 11iv2 with Patch Bundle dated Dec 2007 ▶ HP-UX 11iv3 with Patch Bundle dated Sep 2007

Operating systems	Versions
Linux® on x86	Supported with 32-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ Red Hat Enterprise Linux AS, Version 4 with Update 6 ▶ Red Hat Enterprise Linux ES, Version 4 with Update 6 ▶ Red Hat Enterprise Linux WS, Version 4 with Update 6 ▶ Red Hat Enterprise Linux, Version 5 with Update 1 ▶ SUSE® Linux Enterprise Server, Version 9 with SP4 ▶ SUSE Linux Enterprise Server, Version 10 with Update 1
Linux on x86-64 and POWER	Supported with 32-bit and 64-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ Red Hat Enterprise Linux, Version 4 with Update 6 ▶ Red Hat Enterprise Linux, Version 5 with Update 1 ▶ SUSE Linux Enterprise Server, Version 9 with SP4 ▶ SUSE Linux Enterprise Server, Version 10 with Update 1
Linux on System z and zSeries®	Supported with 31-bit and 64-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ Red Hat Enterprise Linux AS, Version 4 with Update 6 ▶ Red Hat Enterprise Linux, Version 5 with Update 1 ▶ SUSE Linux Enterprise Server, Version 9 with SP4 ▶ SUSE Linux Enterprise Server, Version 10 with Update 1
IBM z/OS (Supported for WebSphere Application Server V7.0 for z/OS)	Supported with 31-bit and 64-bit WebSphere Application Server: <ul style="list-style-type: none"> ▶ z/OS V1.7 ▶ z/OS V1.8 ▶ z/OS V1.9 ▶ z/OS.e ▶ z/OS.e V1.7 ▶ z/OS.e V1.8
IBM i	<ul style="list-style-type: none"> ▶ IBM i V5R4 ▶ IBM i V6R1

About support limitations: To see if there is a 32-bit or 64-bit support limitation for your operating system, refer to the WebSphere Application Server system requirements Web page:

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

1.4.3 Web servers

Supported Web servers for WebSphere Application Server V7.0 are as follows:

- ▶ Apache HTTP Server 2.0.58
- ▶ Apache HTTP Server 2.2
- ▶ IBM HTTP Server for WebSphere Application Server V6.1
- ▶ IBM HTTP Server for WebSphere Application Server V7.0
- ▶ Internet Information Services (IIS) 6.0
- ▶ Internet Information Services (IIS) 7.0
- ▶ Lotus® Domino® Enterprise Server 7.0.2
- ▶ Lotus Domino Enterprise Server 8.0
- ▶ Sun Java™ System Web Server 6.1 SP8
- ▶ Sun Java System Web Server 7.0 Update 1

Note: Not every Web server is supported on all platforms. For more information refer to the WebSphere Application Server system requirements Web page:

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

1.4.4 Database servers

Table 1-3 shows the database servers that WebSphere Application Server V7.0 supports.

Table 1-3 Supported database servers and versions

Databases	Versions
IBM DB2® ^a	DB2 for iSeries 5.3, 5.4, or 6.1 DB2 for z/OS V8, V9 or V9.1 DB2 Connect™ DB2 Enterprise Server Edition V8.1, V8.2, V9.0 and V9.5 DB2 Express V8.1, V8.2, V9.0 and V9.5 DB2 Workgroup Server Edition V8.1, V8.2, V9.0 and V9.5
Cloudscape	Cloudscape 10.3 (Derby)
Oracle®	Oracle 10g Standard/Enterprise Release 1 - 10.1.0.5 Oracle 10g Standard/Enterprise Release 2 - 10.2.0.3 Oracle 11g Standard/Enterprise Release 1 - 11.1.0.6
Sybase	Sybase Adaptive Server Enterprise 12.5.4 Sybase Adaptive Server Enterprise 15.0.2
Microsoft SQL Server®	Microsoft SQL Server Enterprise 2005 SP2

Databases	Versions
Informix®	Dynamic Server 10.00xC6 Dynamic Server 11.10xC1 Dynamic Server 11.5xC1
IMS	IMS V9 IMS V10
WebSphere Information Integrator	WebSphere Information Integrator 8.2 FP8 WebSphere Information Integrator 9.1 FP3 WebSphere Information Integrator 9.5 FP2

a. For a detailed list of supported fixpack levels and editions for DB2, see the support document #7013265 at the following Web page:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27013265>

1.4.5 Directory servers

Supported directory servers for WebSphere Application Server V7.0 are as follows:

- ▶ LDAP Servers using Stand Alone LDAP User Registry Configuration and Federated Repository Configuration:
 - IBM Tivoli Directory Server 6.0
 - IBM Tivoli Directory Server 6.1
 - IBM z/OS Integrated Security Services 1.8
 - IBM z/OS Integrated Security Services 1.9
 - IBM z/OS.e Security Server 1.8
 - IBM z/OS.e Security Server 1.9
 - Lotus Domino Enterprise Server 7.0
 - Lotus Domino Enterprise Server 8.0
 - Novell® eDirectory 8.7.3 SP9
 - Novell eDirectory 8.8.1
 - Sun Java Directory Server 6.0
 - Sun Java Directory Server 6.1
 - Windows Active Directory® 2003
 - Windows Active Directory 2008
- ▶ LDAP Servers using only Federated Repository Configuration:
 - Windows Active Directory Applications Mode 1.0 SP1
 - OpenLDAP 2.4.7

1.5 Related products

IBM offers complementary software products surrounding WebSphere Application Server that provide simplification for developers, enhanced management features, and a high performance runtime environment. Some of these are introduced in this chapter:

- ▶ WebSphere Application Server Community Edition
- ▶ WebSphere Extended Deployment
- ▶ Rational Application Developer Assembly and Deploy V7.5
- ▶ Rational Application Developer for WebSphere Software V7.5
- ▶ Project Zero and WebSphere sMash

1.5.1 WebSphere Application Server Community Edition

WebSphere Application Server Community Edition is a lightweight single-server Java EE application server built on Apache Geronimo, the open source application server project of the Apache Software Foundation. This edition of WebSphere Application Server is completely based on open source code and available to download free of charge.

Note: WebSphere Application Server Community Edition's codebase is different from the WebSphere Application Server. It is not a different packaging option for WebSphere Application Server. It is a separate product.

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

- ▶ Brings together best-of-breed technologies across the broader open source community to support Java EE 5 specifications such as the following:
 - Apache MyFaces
 - Apache OpenEJB
 - Apache Open JPA
 - Apache ActiveMQ
 - TranQL
- ▶ Supports the Java Developer Kit (JDK) from IBM and Sun.
- ▶ Can be used as a runtime for Eclipse with its plug-in.
- ▶ Includes an open source Apache Derby database which is a small-footprint database server with full transactional capability.
- ▶ Contains an easy-to-use administrative console application.

- ▶ Product binaries and source code are available as no-charge downloads from the IBM Web site.
- ▶ Optional fee-based support for WebSphere Application Server Community Edition from IBM Technical support teams.
- ▶ Can be included in advanced topologies and managed with WebSphere Extended Deployment.

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

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

1.5.2 WebSphere Extended Deployment

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

- ▶ **WebSphere Virtual Enterprise**
This product, formerly Operations Optimization, provides application infrastructure virtualization capabilities. Virtualization can lower costs required to create, manage, and run enterprise applications by using existing hardware resources. This package also contains additional management features like application edition and dynamic application management, management of heterogeneous application servers, and service policy management, which provide an enhanced management environment for existing infrastructure.
- ▶ **WebSphere eXtreme Scale**
This product, formerly WebSphere Extended Deployment Data Grid, allows business applications to process large volumes of transactions with efficiency and linear scalability. WebSphere eXtreme Scale operates as an in-memory data grid that dynamically caches, partitions, replicates, and manages application data and business logic across multiple servers. It provides transactional integrity and transparent failover to ensure high availability, high reliability, and constant response times. WebSphere eXtreme Scale is a software technology for conducting extreme transaction processing.
- ▶ **Compute Grid**
This product enables the scheduling, execution, and monitoring of batch type jobs and compute-intensive tasks with service policy and workload management.

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

Some of the benefits of WebSphere Extended Deployment are as follows:

- ▶ Provides virtualization capabilities that can dynamically match available resources to changing workload demands. It enables more efficient resource use as well as improved application performance and scalability.
- ▶ Features virtualization, and workload and health management for the following application environments:
 - PHP
 - BEA WebLogic
 - JBoss
 - Apache Tomcat
 - WebSphere Application Server Community Edition
- ▶ Offers enhanced capabilities in job scheduling, monitoring, and management for batch-type workloads.
- ▶ Delivers customizable health policies and actions to help enhance manageability.
- ▶ Supports innovative and highly scalable data fabrics to accelerate data-intensive application performance.

For more information about WebSphere Extended Deployment, see the following Web page:

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

1.5.3 Rational Application Developer Assembly and Deploy V7.5

In WebSphere Application Server V7.0, Rational Application Developer Assembly and Deploy V7.5 replaces the Application Server Toolkit (AST) function provided with WebSphere Application Server V6.1. This Eclipse-based Integrated Development Environment (IDE) expands on the functions provided in the Application Server Toolkit with Java EE 5 and is a subset of Rational Application Developer for WebSphere Software V7.5. Rational Application Developer Assembly and Deploy is fully licensed and supported with the WebSphere Application Server license.

Some of the features of Rational Application Developer Assembly and Deploy are as follows:

- ▶ Profile management tools for WebSphere Application Server
- ▶ Jython tools for administration script development and Jacl to Jython conversion tools
- ▶ J2EE 1.4 (same level as Application Server Toolkit V6.1)
- ▶ Java EE 5 XML-form based DD editors
- ▶ Java EE 5 application support
- ▶ Supports only WebSphere Application Server V7.0 as runtime
- ▶ Contains WebSphere Application Server debug extensions
- ▶ Enhanced EAR support
- ▶ Application deployment support
- ▶ SIP development tools
- ▶ Visual editing tools
- ▶ Adapters support for simplified and enhanced integration

1.5.4 Rational Application Developer for WebSphere Software V7.5

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

Rational Application Developer for WebSphere Software adds functions to Rational Application Developer Assembly and Deploy (Figure 1-5).



Figure 1-5 Rational development tools

Rational Application Developer for WebSphere Software includes the following functions:

- ▶ Concurrent support for J2EE 1.2, 1.3, 1.4, and Java EE 5 specifications and support for building applications with JDK 5 and JRE™ 1.6.
- ▶ EJB 3.0 productivity features.

- ▶ Visual editors, as follows:
 - Domain modeling
 - UML modeling
 - Web development
- ▶ Web services and XML productivity features.
- ▶ Portlet development tools.
- ▶ Relational data tools.
- ▶ WebSphere Application Server V6.0, V6.1 and V7.0 test servers.
- ▶ Web 2.0 development features for visual development of responsive Rich Internet Applications with Ajax and Dojo¹.
- ▶ Integration with the Rational Unified Process and the Rational tool set, which provides the end-to-end application development life cycle.
- ▶ Application analysis tools to check code for coding practices. Examples are provided for best practices and issue resolution.
- ▶ Enhanced runtime analysis tools, such as memory leak detection, thread lock detection, user-defined probes, and code coverage.
- ▶ Component test automation tools to automate test creation and manage test cases.
- ▶ WebSphere adapters support, including CICS®, IMS, SAP®, Siebel®, JD Edwards®, Oracle and PeopleSoft®.
- ▶ Support for Linux and Microsoft Windows operating systems.

Note: Rational Application Developer for WebSphere Software is optionally installable as a 60-day trial with WebSphere Application Server V7.0 and purchased as a separate product.

For more information about Rational Application Developer for WebSphere Software V7.5, see the following Web page:

<http://www-01.ibm.com/software/awdtools/developer/application/>

1.5.5 Project Zero and WebSphere sMash

Project Zero is an incubator project started by IBM, centered around agile development and the next generation of dynamic Web applications. The project brings a simple environment to create, assemble, and run applications based on popular Web technologies. This environment includes a scripting runtime for

¹ The Dojo Toolkit is a modular open source JavaScript™ toolkit (or library), designed for the rapid development of cross platform, JavaScript/Ajax based applications and Web sites.

Groovy and PHP. It also includes application programming interfaces optimized for producing REST-style services, integration mash-ups, and rich Web interfaces corresponding to Web 2.0 concepts. It gives access to the latest features by defaulting to experimental modules. Project Zero is free for development and limited deployment.

Project Zero creates a new way to build commercial software, an approach that is called Community-Driven Commercial Development. Community-Driven Commercial Development is powered by community feedback.

The Community-Driven Commercial Development process served as a base for two IBM Software products:

▶ WebSphere sMash Developer Edition

This product is a stable build of Project Zero, offered free of charge for development and limited deployment usage. It includes useful tools for developing applications in WebSphere sMash.

▶ WebSphere sMash

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

Some of the features of WebSphere sMash are as follows:

- It is an application-centric runtime. You can just create an application and run it. Everything needed to run the application is built in, including the HTTP stack,
- It is designed around Dynamic Scripting, REST, Rich Web Interfaces, AJAX, and Feeds corresponding to Web 2.0 concepts.
- Application logic is created in one of two scripting languages:
 - Groovy, for people that prefer Java.
 - PHP for existing PHP programmers. The PHP runtime is provided directly in the WebSphere sMash.
- It empowers developers to build applications directly on the Web with a Web interface, and compose applications by wiring together REST services.

For more information about Project Zero, see the following Web page:

<http://www.projectzero.org/>

For more information about WebSphere sMash, see the following Web page:

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

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, and broad integration features:

- ▶ “Tivoli Access Manager” on page 30
- ▶ “Tivoli Directory Server” on page 33
- ▶ “WebSphere MQ” on page 35
- ▶ “WebSphere Adapters” on page 39
- ▶ “WebSphere DataPower” on page 41
- ▶ “DB2” on page 46
- ▶ “Tivoli Composite Application Manager for WebSphere” on page 47

2.1 Tivoli Access Manager

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

Tivoli Access Manager provides the following features:

- ▶ Defines and manages centralized authentication, access, and audit policy for a broad range of business initiatives.
- ▶ Establishes a new audit and reporting service that collects audit data from multiple enforcement points, as well as from other platforms and security applications.
- ▶ Enables flexible Single Sign-on (SSO) to Web-based applications that span multiple sites or domains with a range of SSO options, to eliminate help-desk calls and other security problems associated with multiple passwords.
- ▶ Uses a common security policy model with the Tivoli Access Manager family of products to extend support to other resources.
- ▶ Manages and secures your business environments from your existing hardware (mainframe, PCs, servers) and operating system platforms including Windows, Linux, AIX, Solaris, and HP-UX.
- ▶ Provides a modular authorization architecture that separates security code from application code.

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

For more information about Tivoli Access Manager, see the following Web page:

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

2.1.1 Integration with WebSphere Application Server

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

For more information about WebSphere Application Server's security infrastructure, refer to 12.2, "Security in WebSphere Application Server" on page 381.

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

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

- ▶ 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 IBM Tivoli Directory Server and Microsoft Active Directory. 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 the credentials information of all participants in the secure domain, both users and servers. The authorization database is then replicated across all local authorization servers.

Tivoli Access Manager for WebSphere component

The Tivoli Access Manager client is embedded in WebSphere Application Server. 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 by supporting 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 performed through the Java Authorization Contract for Containers (JACC) API.

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

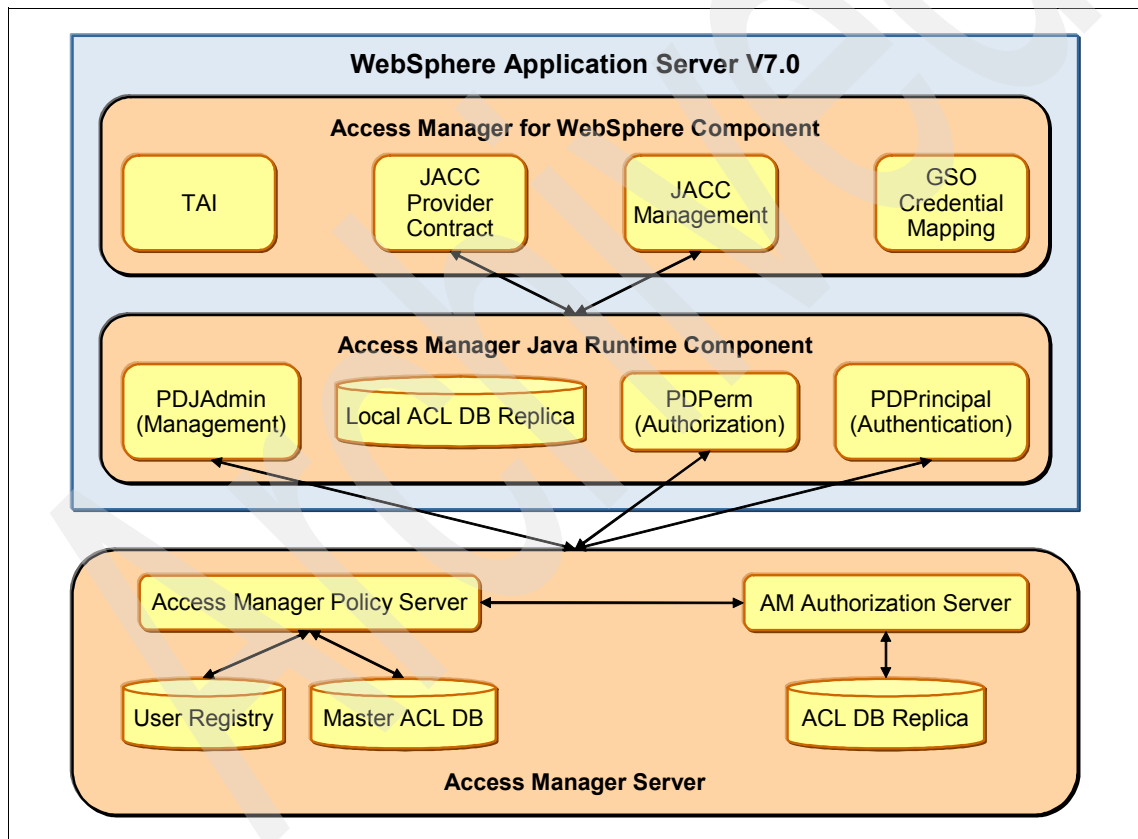


Figure 2-1 Integration of WebSphere Application Server with Tivoli Access Manager

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 applications
- ▶ Provides tight integration with WebSphere Application Server

Security, networking, and topology considerations

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

Legal considerations (privacy and data protection): Be aware that there might be some legal 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 on your systems. These considerations vary by geography and industry.

2.2 Tivoli Directory Server

In today's highly connected world, directory servers are the foundation of authentication systems for internal, and more commonly, external user populations in the corporate infrastructure.

IBM Tivoli Directory Server provides a high-performance Lightweight Directory Access Protocol (LDAP) identity infrastructure capable of handling millions of entries. It is built to serve as the identity data foundation for your Web applications and identity management initiatives.

2.2.1 Lightweight Directory Access Protocol

A directory is a data structure that enables the 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 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 have custom objects added to it, meaning that your directory is extensible and customizable.

Generally, LDAP is chosen over a custom database repository of users for these reasons. LDAP implementations (such as 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. For more information about IBM Tivoli Directory Server, see the following Web page:

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

2.2.2 Integration with WebSphere Application Server

You can enable security in WebSphere Application Server to manage users, and assign specific roles to them. To have a user account repository you must select the type of user registry to be used (in this case, an LDAP registry). Tivoli Directory Server can be used as a standalone LDAP registry for user account repository of WebSphere Application Server. You can configure your user account repository through the Integrated Solutions Console or through the `wsadmin` command line tool.

Security, networking, and topology considerations

Because the LDAP server contains sensitive data in terms of authentication, authorization, and privacy, the LDAP server belongs to the data layer of the network. It is suggested 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.

Legal considerations (privacy and data protection): There might be some legal 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 on your systems. These considerations vary by geography and industry.

For a list of supported directory servers for WebSphere Application Server, see 1.4.5, “Directory servers” on page 22.

2.3 WebSphere MQ

IBM WebSphere MQ is an asynchronous messaging technology that is available from IBM. WebSphere MQ is a middleware technology 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 other languages, which allow applications to construct, send, and receive messages.

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

For more information about WebSphere MQ, see the following Web page:

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

2.3.1 Integration with WebSphere Application Server

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

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

For WebSphere Application Server Version 7.0, the WebSphere MQ messaging provider has enhanced administrative options supporting the following functions:

- ▶ WebSphere MQ channel compression
Data sent between WebSphere Application Server and WebSphere MQ can be compressed, reducing the amount of data that is transferred.
- ▶ WebSphere MQ client channel definition table
The client channel definition table reduces the effort required to configure a connection to a queue manager.
- ▶ Client channel exits
Client channel exits are pieces of Java code that you develop, and that are executed in the application server at key points during the life cycle of a WebSphere MQ channel. Your code can change the runtime characteristics of the communications link between the WebSphere MQ messaging provider and the WebSphere MQ queue manager.
- ▶ Transport-level encryption using SSL
Transport-level encryption using SSL is the supported way to configure SSL for JMS resources associated with the WebSphere MQ messaging provider. The SSL configuration is associated with the communication link for the connection factory or activation specification.
- ▶ Automatic selection of the WebSphere MQ transport type
Servers in a cluster can be configured automatically to select their transport.

In WebSphere Application Server V7.0, you can use the following JMS providers:

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

The default messaging provider is the JMS API implementation for messaging (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 messaging engine (more, in the case of clusters) that manages connections to the bus and messages. A service integration bus can connect to other service integration buses and to WebSphere MQ. Similarly, the WebSphere MQ JMS provider is the JMS API implementation with WebSphere MQ (with queue managers, for example) implementing the real destinations for the JMS interface. WebSphere MQ can coexist on the same host as a WebSphere Application Server messaging engine.

Whether to use the default messaging provider, the direct WebSphere MQ messaging provider, or a combination depends on 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 for the following circumstances:

- ▶ You are currently using the WebSphere Application Server V5 embedded messaging provider for intra-WebSphere Application Server messaging.
- ▶ You require messaging between WebSphere Application Server and an existing WebSphere MQ backbone and its applications.

The WebSphere MQ messaging provider is good choice for the following circumstances:

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

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

For more information about messaging with WebSphere Application Server and about new features for WebSphere MQ connectivity see the following IBM Information Center page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.iseries.doc/info/iseriesnd/ae/cmj_jmsp_wmq.html

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.

Keep in mind:

- ▶ 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 group¹ 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.

Only WebSphere MQ queue managers and queue sharing groups running on z/OS can be accessed from a service integration bus in this way.

The WebSphere MQ server does not depend on any one designated messaging engine. This type of connectivity to MQ can tolerate the failure of any given message engine if another is available in the bus, increasing robustness and availability. This 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 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.

¹ 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.

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

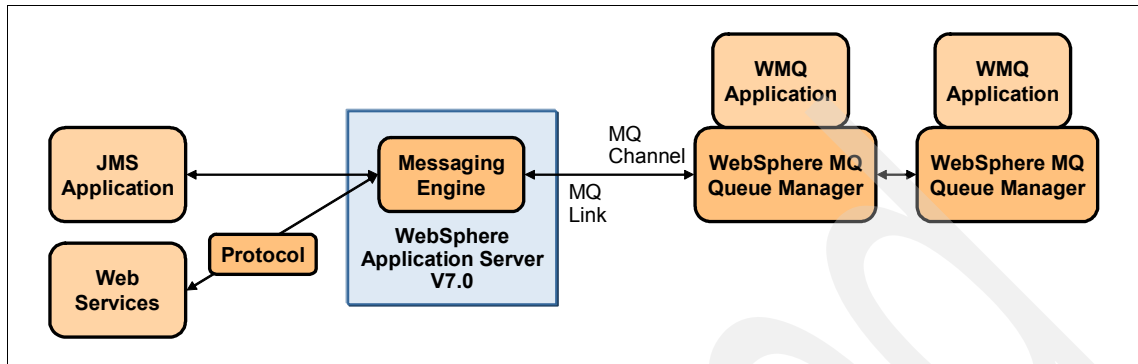


Figure 2-2 WebSphere Application Server integration with WebSphere MQ

2.4 WebSphere Adapters

IBM WebSphere Adapters provide a set of generic technology and business application adapters with wizards that quickly and easily service enable Enterprise Information Systems (EISs) such as legacy applications, Enterprise Resource Planning (ERP), Human Resources (HR), Customer Relationship Management (CRM), and supply chain systems, and integrate them to IBM Business Process Management (BPM), Enterprise Service Bus (ESB), and application server solutions in a service-oriented architecture (SOA).

WebSphere Adapters implement the Java Connector Architecture (JCA) and Enterprise MetaData Discovery specifications to provide a simple and quick integration experience with graphical discovery tools without resorting to writing code.

WebSphere Adapters includes three types of adapters:

- ▶ Application Adapters

These adapters integrate enterprise business application suites:

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

▶ Technology Adapters

These adapters deliver file and database connectivity solutions:

- Email
- Flat Files
- File Transfer Protocol (FTP)
- Java Database Connectivity (JDBC)

▶ WBI Adapters

These adapters use an asynchronous stand alone runtime architecture (the WBI Framework) with WebSphere MQ or JMS as the underlying transport protocol. Originally designed for WebSphere InterChange Server (WICS), they remain available for WebSphere Message Broker, WebSphere ESB, and WebSphere Process Server users transitioning from WebSphere InterChange Server or WebSphere Business Integration Server.

WebSphere Adapters are used with the WebSphere Adapter Toolkit, an Eclipse-based toolkit, to enable customers and business partners to develop custom JCA adapters to meet unique business requirements. The toolkit helps to create either a basic JCA 1.5 adapter, or an adapter that uses the additional capabilities of the Adapter Foundation Classes used by WebSphere Adapters. The WebSphere Adapter Toolkit is provided as a no-fee download from IBM developerWorks® to customers and business partners who secure licenses to WebSphere Integration Developer and Rational Application Developer for WebSphere Software.

For more information about IBM WebSphere Adapters, see the following Web page:

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

2.4.1 Integration with WebSphere Application Server

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

Figure 2-3 shows the relation between WebSphere Application Server and a WebSphere Adapter.

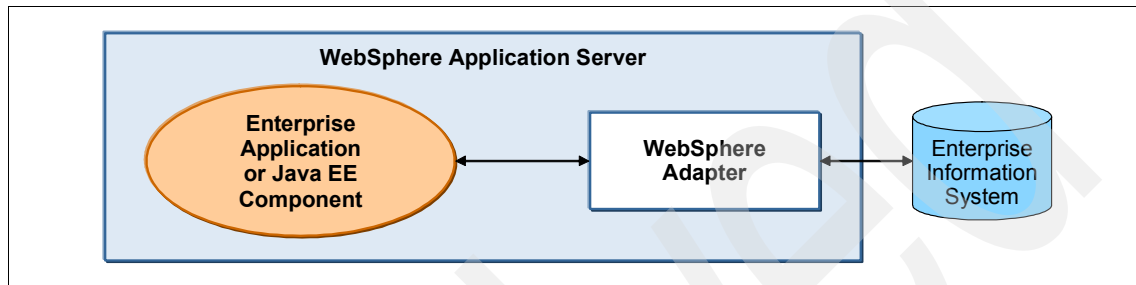


Figure 2-3 WebSphere Adapter integration with WebSphere Application Server

Note: Other available WebSphere Adapters are supported only with WebSphere Integration Developer, WebSphere Process Server and WebSphere Enterprise Service Bus.

2.5 WebSphere DataPower

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

The IBM WebSphere DataPower SOA Appliances family contains rack-mountable network devices that offer the following features:

- ▶ 1U (1.75-inch thick) rack-mountable, purpose-built network appliances.
- ▶ XML/SOAP firewall, field-level XML security, data validation, XML Web services access control, and service virtualization.
- ▶ Lightweight and protocol-independent message brokering, integrated message-level security, fine-grained access control, and the ability to bridge important transaction networks to SOAs and ESBs.
- ▶ High performance, multi-step, wire-speed message processing, including XML, XML Stylesheet Language Transformation (XSLT), XPath, and XML Schema Definition (XSD).
- ▶ Centralized Web services policy and service-level management.
- ▶ Web services (WS) standard support:
 - WS-Security
 - Security Assertion Markup Language (SAML) 1.0/1.1/2.0
 - portions of the Liberty Alliance protocol
 - WS-Federation
 - WS-Trust
 - XML Key Management Specification (XKMS)
 - Radius, XML Digital Signature
 - XML-Encryption
 - Web Services Distributed Management (WSDM)
 - WS-SecureConversation
 - WS-Policy
 - WS-SecurityPolicy
 - WS-ReliableMessaging
 - SOAP
 - Web Services Description Language (WSDL)
 - Universal Description
 - Discovery, and Integration (UDDI)
- ▶ Transport layer flexibility, which supports HTTP/HTTPS, MQ, Secure Sockets Layer (SSL), File Transfer Protocol (FTP), and others.
- ▶ Scalable, wire-speed, any-to-any message transformation, such as arbitrary binary, flat text and XML messages, which include COBOL copybook, CORBA, CICS, ISO 8583, ASN.1, EDI, and others.

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

- ▶ Consumable simplicity

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

- ▶ Enhanced security

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

- ▶ Acceleration

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

2.5.1 DataPower appliance models

The WebSphere DataPower appliance family contains three models at the time this book was written (Figure 2-4 on page 44). Each appliance has its own characteristics. Each is designed to fit various business needs. These appliance models are as follows:

- ▶ IBM WebSphere DataPower XML Accelerator XA35

This model can help speed common types of XML processing by offloading the processing from servers and networks. It can perform XML parsing, XML schema validation, XPath routing, XSLT, XML compression, and other essential XML processing with wire-speed XML performance.

- ▶ IBM WebSphere DataPower XML Security Gateway XS40

XS40 provides a security-enforcement point for XML and Web service transactions. It offers encryption, firewall filtering, digital signatures, schema validation, WS-Security, XML access control, XPath, and so on.

- ▶ IBM WebSphere DataPower Integration Appliance XI50

This appliance provides transport-independent transformations between binary, flat text files, and XML message formats. Visual tools are used to describe data formats, create mappings between different formats, and define message choreography. The XI50 appliance can transform binary, flat text, and other non-XML messages to offer an innovative solution for security-rich XML enablement, ESBs, and mainframe connectivity.

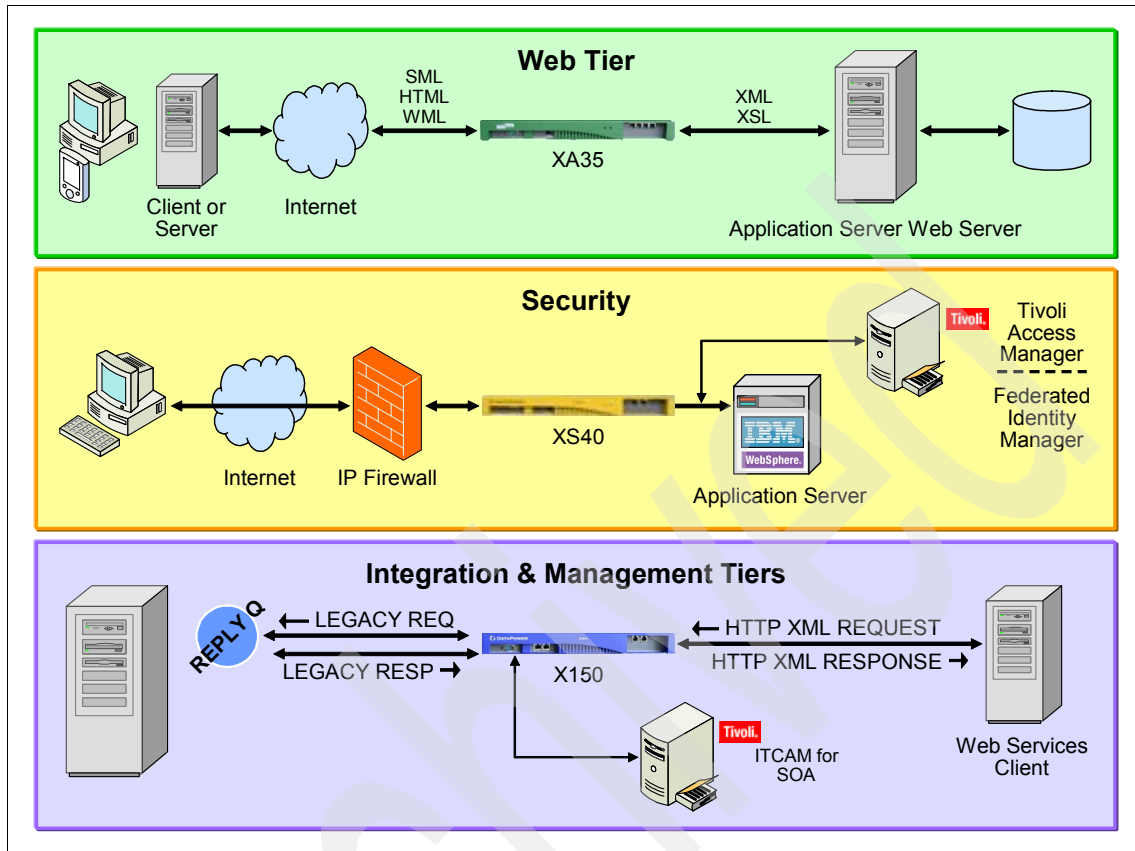


Figure 2-4 DataPower SOA Appliance models

For more information about IBM WebSphere DataPower SOA Appliances, see the following Web page:

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

For more information about devices and usage scenarios, refer to IBM Redpaper *IBM WebSphere DataPower SOA Appliances Part I: Overview and Getting Started*, REDP-4327.

2.5.2 Integration with WebSphere Application Server

In WebSphere Application Server V7.0, the new consolidated administration feature for WebSphere DataPower allows you to manage and integrate appliances into your environment.

The Integrated Solutions Console contains an administration interface (DataPower appliance manager) to manage multiple WebSphere DataPower boxes (Figure 2-5). The Integrated Solutions Console is the single point of administration to manage both WebSphere Application Server and WebSphere DataPower, and for solutions that combines them.

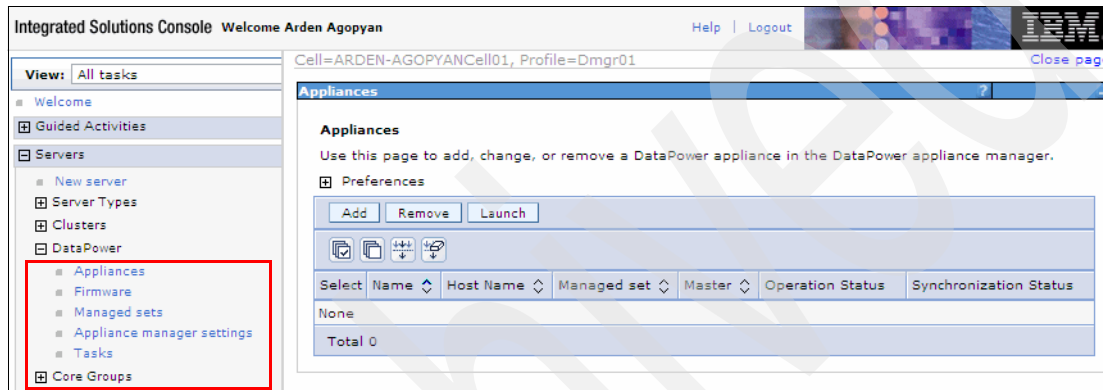


Figure 2-5 DataPower appliance manager interface of the Integrated Solutions Console

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

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

Note: DataPower appliance manager of WebSphere Application Server V7.0 can be used to manage DataPower appliances with a V3.6.0.4 or higher level of firmware.

2.6 DB2

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

DB2 extends its innovative abilities as a hybrid data server. It enables rapid use and deep compression of data, and extracts the full value of XML data in operational processes.

DB2 delivers the following advantages:

- ▶ Improved performance for high priority workloads
- ▶ Shortened developer timelines with enhanced XML features
- ▶ Shortened time to recover
- ▶ Enhancements to server and compliance to safeguard your data server
- ▶ Reduced administration with advances in performance, manageability, and installation

DB2 has editions to work on Linux, UNIX®, Windows, and z/OS. These editions are designed to best fit your business needs. It also brings extra features (like storage optimization, geodetic data management, and so on) to your environment with its priced extensions.

For more information about IBM DB2 and its editions, see the following Web page:

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

2.6.1 Integration with WebSphere Application Server

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

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

- ▶ DB2 can be the hybrid data store for your applications. It can enhance your data processing with its powerful XML capabilities. You can configure your datasources to use DB2 using Java Database Connectivity (JDBC) drivers.
- ▶ With its pureQuery runtime environment, a high performance Java data access platform that helps manage applications that access data, DB2 provides an alternate set of APIs that can be used instead of JDBC to access the DB2 database. PureQuery support is based on Java Persistence API (JPA) of the Java EE and Java SE environments.

- ▶ You can configure a service integration bus (messaging) member to use DB2 as a data store.
- ▶ The session management facility of WebSphere Application Server can be configured for database session persistence, using DB2 as the data store. You can collect and store session data in a DB2 database.
- ▶ DB2 can be used as the data store for your UDDI registry data.
- ▶ The scheduler database for storing and running tasks of the scheduler service of WebSphere Application Server can be a DB2 database. The scheduler service is a WebSphere programming extension responsible for starting actions at specific times or intervals,

For more information about data access resources for WebSphere Application Server, see the IBM Information Center Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.base.doc/info/aes/ae/welc6tech_dat.html

2.7 Tivoli Composite Application Manager for WebSphere

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

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

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

2.7.1 Integration with WebSphere Application Server

ITCAM for WebSphere enables you to analyze the health of the WebSphere Application Server and the transactions that are invoked in it. It is able to trace the transaction execution to the detailed method-level information. It connects transactions that spawn from one application server. It also invokes services from other application servers, including mainframe applications in IMS or CICS. ITCAM for WebSphere provides a flexible level of monitoring, from a non-intrusive production ready monitor, to a detailed deep-dive tracing for problems of locking or even memory leaks. ITCAM for WebSphere provides a separate interactive Web console and allows monitoring data to be displayed on the Tivoli Enterprise Portal.

ITCAM for WebSphere is an evolution from WebSphere Studio Application Monitor and OMEGAMON® XE for WebSphere Application Server and provides the following additional functions:

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

2.7.2 ITCAM for WebSphere architecture

ITCAM for WebSphere is a distributed performance monitoring application for application servers. Its components are connected through TCP/IP communication. The central component of ITCAM for WebSphere, the managing server, is its heart and brain. It collects and displays various performance information from application servers. The application servers run a component of ITCAM for WebSphere called the data collector (DC), which is a collecting agent. It helps you pinpoint, in real time, the source of bottlenecks in application code, server resources, and external system dependencies. The Tivoli Enterprise Monitoring Agent component collects information that shows the status of the WebSphere server and sends this information to the Tivoli Enterprise Monitoring Agent. This agent is installed on the individual machines where the data collector resides.

Figure 2-6 shows the overall architecture of ITCAM for WebSphere.

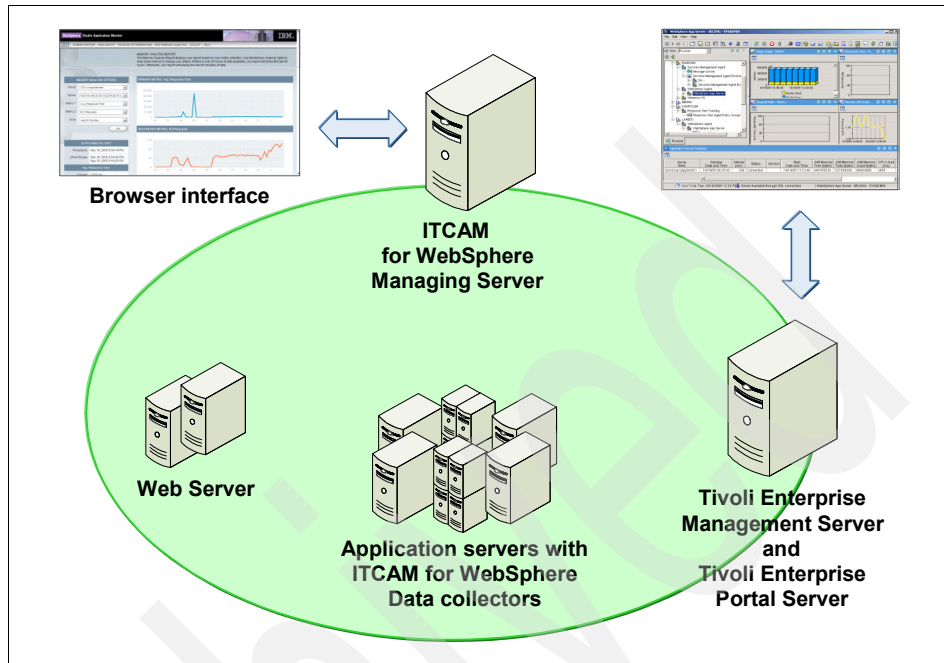



Figure 2-6 ITCAM for WebSphere architecture

For more information about IBM Tivoli Composite Application Manager for WebSphere usage scenarios, refer to the following Redbooks publications:

IBM Tivoli Composite Application Manager Family Installation, Configuration, and Basic Usage, SG24-7151

Solution Deployment Guide for IBM Tivoli Composite Application Manager for WebSphere, SG24-7293

Archived



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 and elements that make up a WebSphere Application Server runtime environment.

This chapter will introduce common WebSphere Application Server concepts in the following sections:

- ▶ “WebSphere Application Server concepts” on page 52
- ▶ “Server environments” on page 73
- ▶ “Clusters” on page 81
- ▶ “Runtime processes” on page 88
- ▶ “Using Web servers” on page 89

3.1 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 are concepts primarily related to Network Deployment packaging. WebSphere Application Server also contains various agents (explained later in this chapter) to provide service to management components.

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 and there is no workload distribution or fail-over capabilities among application servers. With Network Deployment, you can build a distributed server environment consisting of multiple application servers maintained from a central administration point as well as cluster application servers for workload distribution. With Express and Base configurations, you can also use the provided agents and manager components to build a topology for central administration.

This section addresses the following concepts:

- ▶ “Profiles” on page 53
- ▶ “Stand-alone application servers” on page 55
- ▶ “Distributed application servers” on page 57
- ▶ “Nodes, node groups, and node agents” on page 59
- ▶ “Cells” on page 61
- ▶ “Deployment manager” on page 62
- ▶ “Administrative agent” on page 63
- ▶ “Job manager” on page 64
- ▶ “Web servers” on page 65
- ▶ “Proxy servers” on page 67
- ▶ “Generic servers” on page 70
- ▶ “Business level applications” on page 70
- ▶ “Centralized installation manager” on page 72
- ▶ “Intelligent runtime provisioning” on page 72

3.1.1 Profiles

This section focuses on profiles as a basic building block of WebSphere Application Server.

Overview

WebSphere Application Server runtime environments are built by creating profiles. Each profile is an instance of a WebSphere Application Server configuration.

WebSphere Application Server Network Deployment allows you to create the following profile types using provided profile templates:

▶ Cell

This environment creates two profiles:

- A management profile with a deployment manager
- An application server profile added (federated) to the management profile

▶ Management

A management profile provides components for managing multiple application server environments. Possible profiles are as follows:

- Deployment manager
- Administrative agent
- Job manager

▶ Application server

An application server profile runs your enterprise applications and makes them available to the internet or to an intranet. It contains a stand-alone application server.

▶ Custom

A custom profile contains an empty node with no servers. However, a server can be added after the profile is created.

▶ Secure proxy (configuration-only)

A secure proxy (configuration-only) profile is for use with a DeMilitarized Zone (DMZ) secure proxy server. This configuration-only profile is intended only to be used to configure the profile using the Integrated Solutions Console. After you configure the profile, you can export the profile configuration and then import it into the secure proxy profile in your DMZ. Secure proxy (configuration-only) profile is only an administrative component.

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

Profile concept

In WebSphere Application Server files are divided into two categories:

- ▶ Core product files
Application binaries for WebSphere Application Server
- ▶ User files
Customizations, including configuration files, installed applications, resource adapters, properties, log files

Profiles are collections of user files (Figure 3-1). They share core product files. A profile contains its own set of scripts, its own environment, and its own repository.

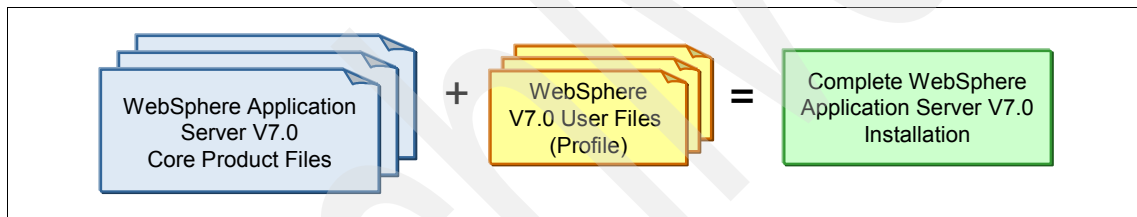


Figure 3-1 WebSphere Application Server installation structure

Each profile is stored in a unique directory path selected by the user at profile creation time. Profiles are stored in a subdirectory of the installation directory by default, but they can be located anywhere (Example 3-1).

Example 3-1 Directory structure of WebSphere Application Server V7.0 on Windows

```
C:\Program Files\IBM\WebSphere\AppServer\profiles
  \AdminAgent01
  \AppSrv01
  \AppSrv02
  \AppSrv03
  \Dmgr01
  \JobMgr01
```

Profile creation

Profiles can be created at any point of time, during or after installation, by using graphical or command line tools. After creating the profiles, for further configuration and administration, you can use these profile management tools provided with WebSphere Application Server:

- ▶ Manageprofiles
Command line interface for profile management functions.
- ▶ Profile Management Tool (PMT)
Eclipse Rich Client Platform (RCP)-based GUI that gathers user input and invokes the manageprofiles command line tool to manage the profiles. For z/OS, zPMT is used as the profile management tool. The zPMT is part of the WebSphere Configuration Tools, that can be downloaded at the following Web page:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg24020368>

3.1.2 Stand-alone application servers

All WebSphere Application Server packages support a single stand-alone server environment. With a stand-alone configuration, each application server acts as an unique entity. An application server runs one or more applications and provides the services required to run those applications. Each stand-alone server is created by defining an application server profile (Figure 3-2).

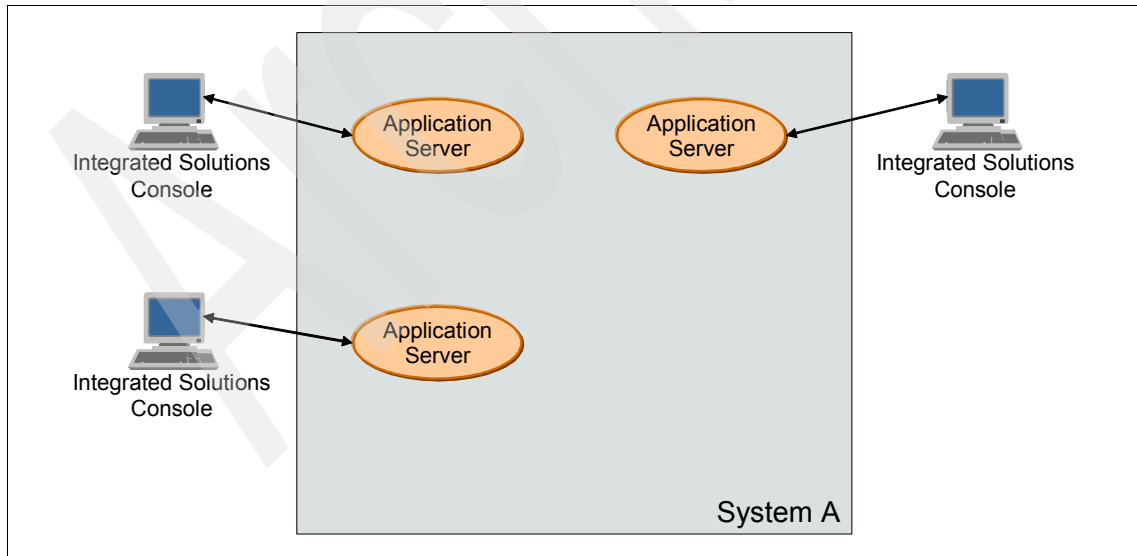


Figure 3-2 Stand-alone application server configuration

About Figure 3-2 : This figure only shows portions of the actual WebSphere Application Server components, The figure will change throughout this chapter to reflect the concepts introduced in each section.

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

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

With WebSphere Application Server for z/OS, it is possible to use workload load balancing and response time goals on a transactional base as well as a special clustering mechanism, the multi-servant region, with a stand-alone application server. For more information about this refer to 14.1.4, "Structure of an application server" on page 422.

Manage stand-alone servers from a central point: With WebSphere Application Server V7.0 it is possible to manage stand-alone servers from a central point by using administrative agents and a job manager. This feature is explained later in 3.2.4, "Flexible management" on page 78 on page 78.

3.1.3 Distributed application servers

With the Network Deployment packaging, you can build a distributed server configuration to enable central administration, workload management, and failover. In this environment, you integrate one or more application servers into a cell that is managed by a central administration instance, a deployment manager (explained in 3.1.6, “Deployment manager” on page 62). 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, as shown in Figure 3-3.

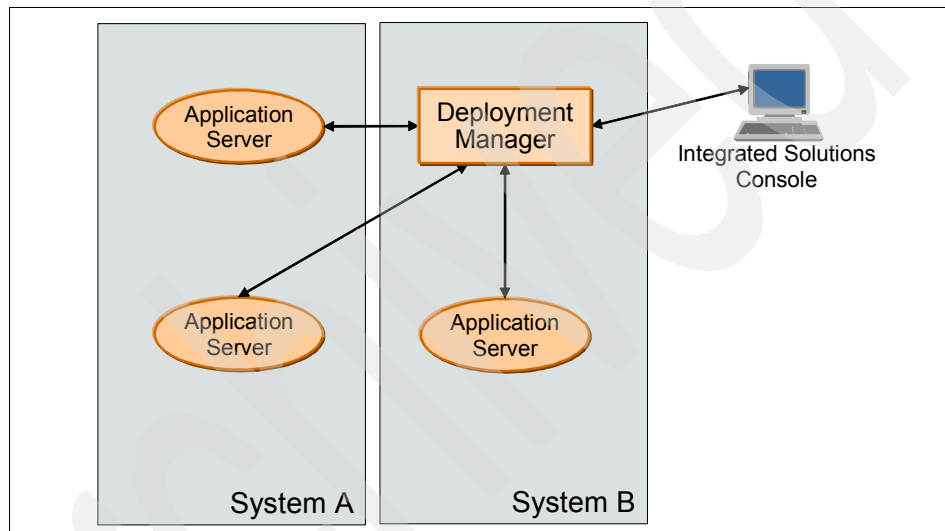


Figure 3-3 Distributed application servers with WebSphere Application Server V7.0

With a distributed server configuration, you can create multiple application servers to run unique sets of applications and manage those applications from a central location. However, more importantly, you can cluster application servers to allow for workload management and failover capabilities. Applications that are installed 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 Enterprise JavaBeans (EJB) containers in a cluster using a weighted round-robin scheme. In z/OS, the weighted round robin mechanism is replaced by the integration of WebSphere Application Server for z/OS in the Workload Manager (WLM). The WLM is an integral part of the operating system. This allows requests to be dispatched to a cluster member according to real time load and whether or not the member reaches its defined response time goals.

It is also possible to replicate sessions saved in an application server of the cluster to other cluster members with the session replication feature of WebSphere Application Server.

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 manually later. The custom nodes can exist inside the same operating system image as the deployment manager or in another operating system instance. Application servers can then be created using the Integrated Solutions Console or **wsadmin** scripts.

The method is useful when you want to create multiple nodes, multiple application servers on a node, or clusters. The process for creating and federating each node is quick.

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

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

- ▶ Create a cell profile. This 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 or z/OS images.

3.1.4 Nodes, node groups, and node agents

This section defines node-related concepts.

Nodes

A *node* is an administrative grouping of application servers for configuration and operational management within one operating system instance (virtualization allows multiple operating systems on one machine). It is possible to create multiple nodes inside one operating system instance, but a node cannot leave the operating system boundaries. 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, which are managed from one central administration server (Figure 3-4).

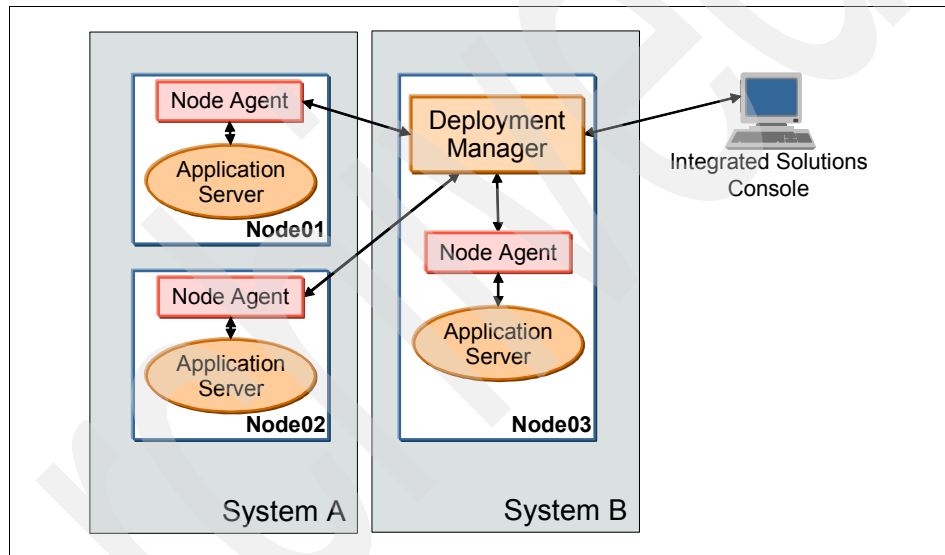


Figure 3-4 Node concept in a WebSphere Application Server network deployment configuration

Node agents

In distributed server configurations, each node has a node agent that works with the deployment manager to manage administration processes (Figure 3-4). A node agent is created automatically when you add (federate) a stand-alone node to a cell. It is not included in the Base and Express configurations.

Node groups

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 them. For example, a cluster cannot contain both z/OS nodes and nodes that are not z/OS-based. In this case, you can define multiple node groups, one for the z/OS nodes and one for nodes other than z/OS. A DefaultNodeGroup is automatically created. 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 system complex (sysplex) node group. Nodes in the same sysplex must be in the same sysplex node group. A node can be in one sysplex node group only.

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

Figure 3-5 shows node and node groups concepts in an example.

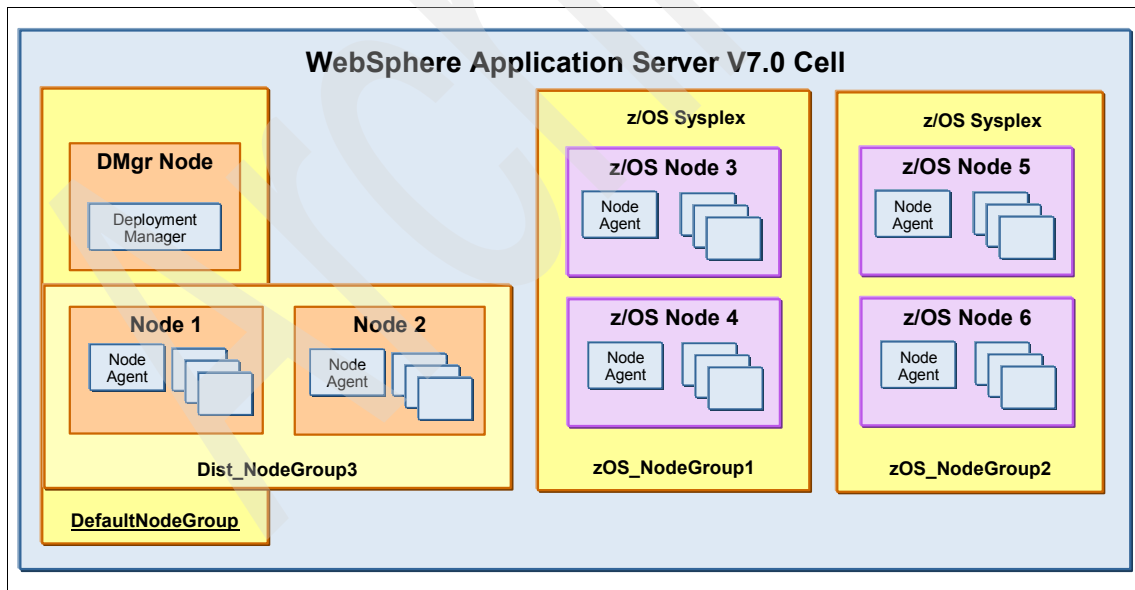


Figure 3-5 Cell, deployment manager, node, and node group concepts

3.1.5 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. See Figure 3-6.

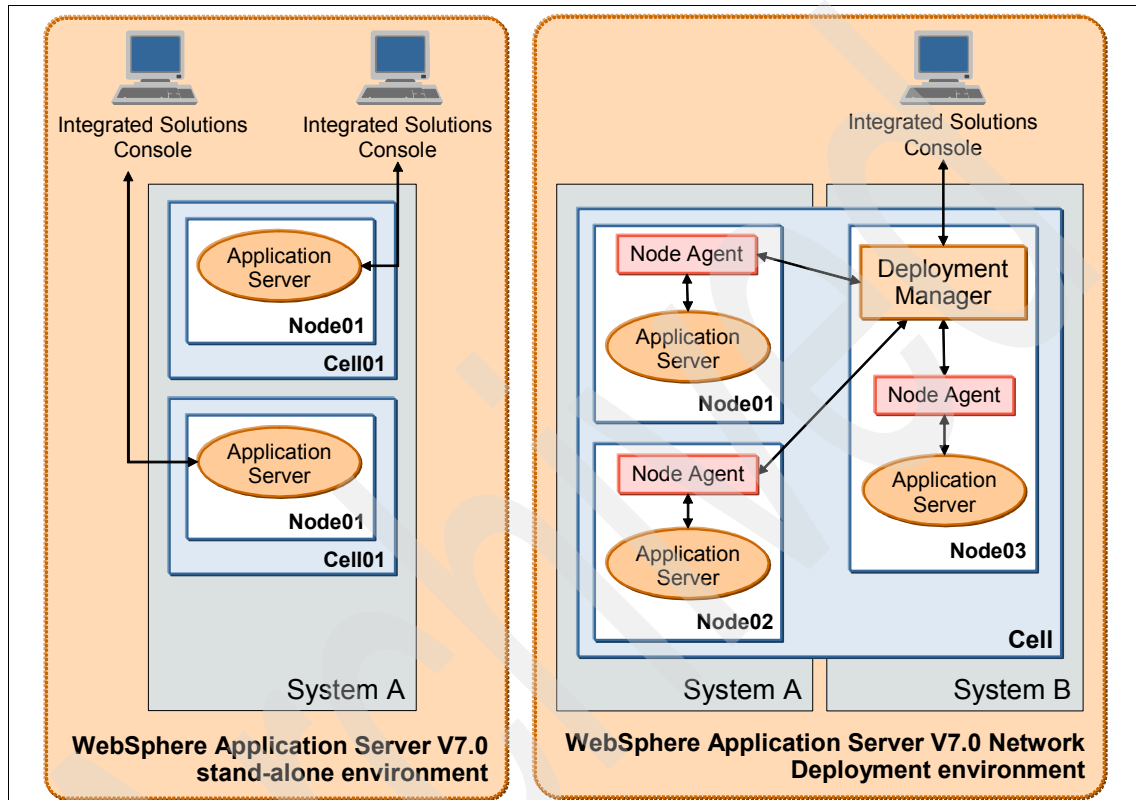


Figure 3-6 Cell component in a WebSphere Application Server topology

In a Network Deployment environment, a cell can consist of multiple nodes (and node groups), which are all administered from a single point, the deployment manager (Figure 3-6). If your cell configuration contains nodes running on the same platform, it is called a *homogeneous cell*. It is also possible to have a cell made up of nodes on mixed platforms. This is referred to as a *heterogeneous cell*.

Note: See 3.2, “Server environments” on page 73 for detailed information about the various cell configurations.

Figure 3-5 on page 60 shows a cell containing multiple nodes and node groups.

3.1.6 Deployment manager

The deployment manager is the central administration point of a cell that consists of multiple nodes and node groups in a distributed server configuration. See Figure 3-5 on page 60. The deployment manager uses the node agent to manage the applications servers within one node.

A deployment manager provides management capability for multiple federated nodes and can manage nodes that span multiple systems and platforms. A node can only be managed by a single deployment manager and must be federated to the cell of that deployment manager.

The configuration and application files for all nodes in the cell are centralized into a master configuration repository. This centralized repository is managed by the deployment manager and synchronized with local copies that are held on each of the nodes. See Figure 3-7.

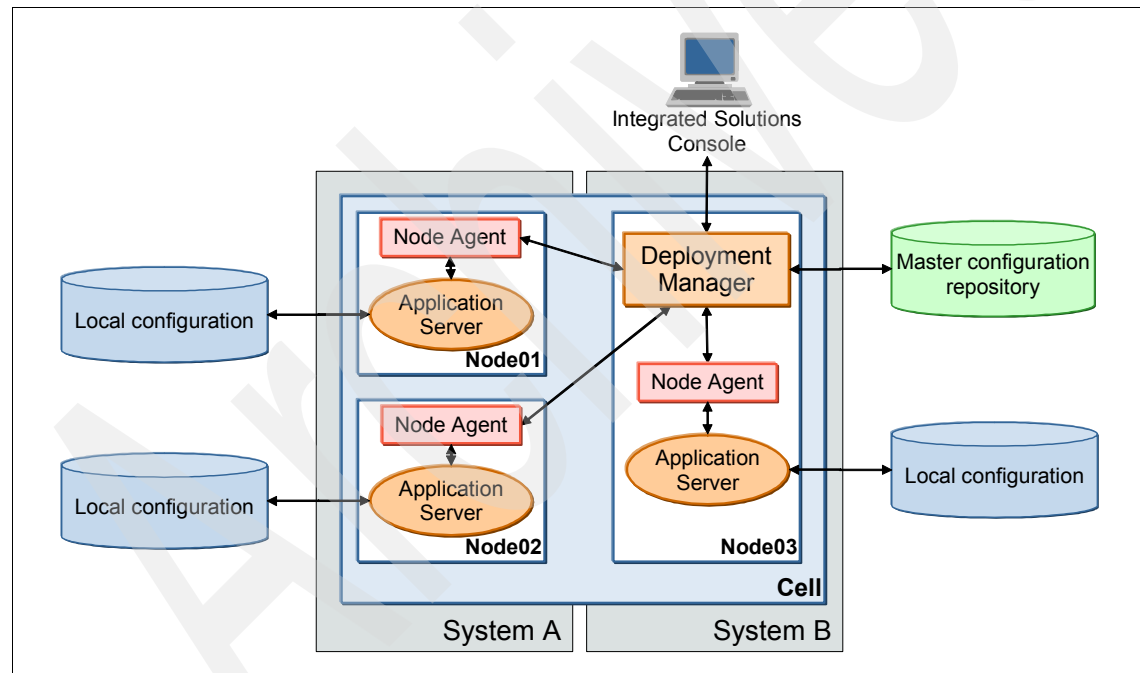


Figure 3-7 Configuration repositories in a network deployment installation

3.1.7 Administrative agent

An administrative agent is a component that provides enhanced management capabilities for stand-alone (Express and Base) application servers. This is a new concept introduced with WebSphere Application Server V7.0.

In previous versions of WebSphere Application Server, each stand-alone server was a single point of management containing its own administrative console. With V7.0, most of the administrative components are separated from the application server runtime. See Figure 3-8.

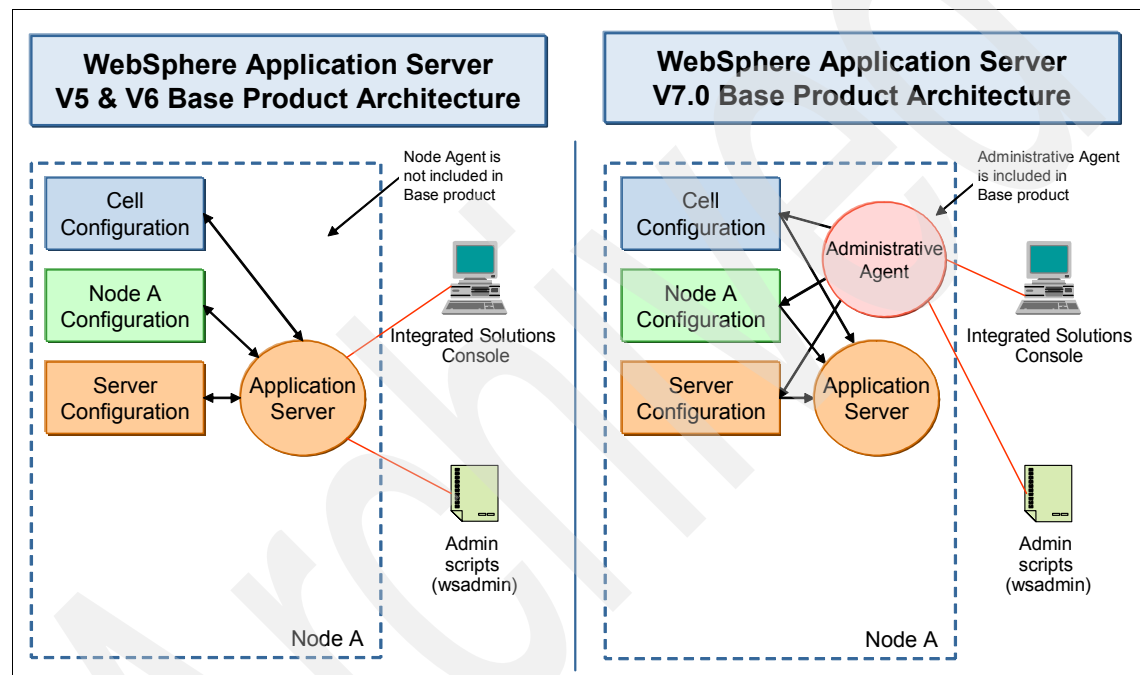


Figure 3-8 Comparison of V5, V6 and V7.0 base product architectures

All configurations related to the application server are directly connected to the administrative agent that provides services to administrative tools.

An administrative agent can manage multiple stand-alone server instances on a single system or z/OS image. When using an administrative agent, as the number of application server instances increases, the redundancy of the administration footprint (for each application server) is therefore eliminated.

The administrative agent acts as the main component for the expanded multiple node remote management environment provided with the job manager, as explained in 3.2.4, “Flexible management” on page 78.

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

- ▶ The administrative agent can only manage application servers that are installed in the same operating system image as the administrative agent.
- ▶ The administrative agent only provides management of these application servers and their applications. It does not provide clustering and failover capabilities. For clustering, failover, and centralized application management, you still need WebSphere Application Server Network Deployment.

3.1.8 Job manager

A job manager (Figure 3-9) is a component that provides management capabilities for multiple stand-alone application servers, administrative agents, and deployment managers. It brings enhanced multiple node installation options for your environment.

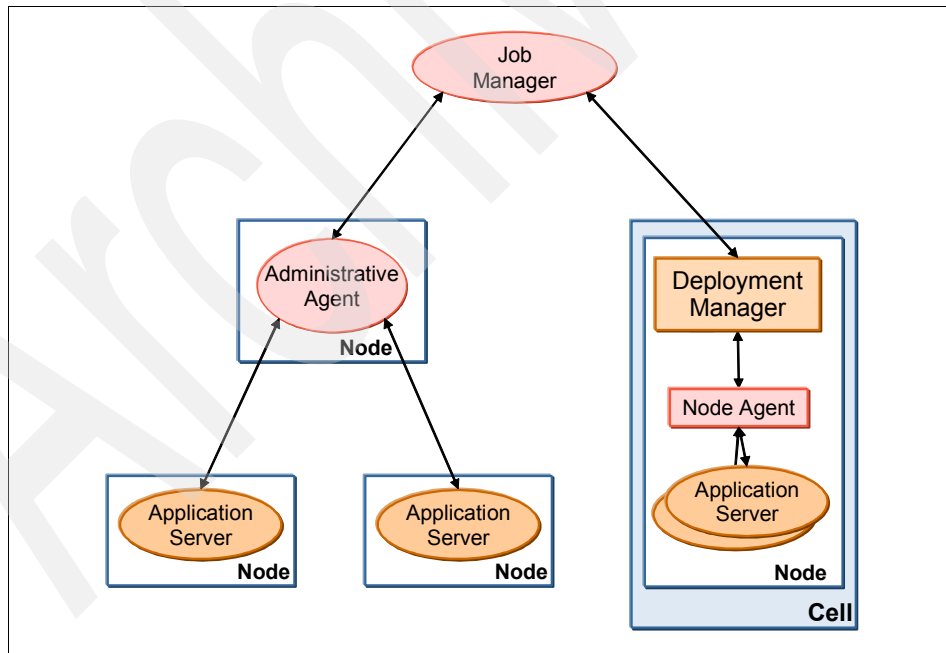


Figure 3-9 High-level overview of a job manager architecture

The purpose of the job manager is to execute daily tasks in one step for multiple installations. This includes the starting and stopping of servers, distribution and deployment of applications and various other actions.

The job manager is a new concept introduced with WebSphere Application Server V7.0 and is only available with WebSphere Application Server Network Deployment and WebSphere Application Server for z/OS.

The job manager and job manager management models are explained in 3.2.4, “Flexible management” on page 78.

3.1.9 Web servers

Although Web servers are independent products, 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 if multiple servers are used.

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. In most installations, you have managed Web server nodes behind the firewall with the WebSphere Application Server installations.
- ▶ Unmanaged nodes are not managed by WebSphere. You usually find these outside the firewall or in the demilitarized zone. You have to manually transfer the Web server plug-in configuration file to the Web server on an unmanaged node. In a z/OS environment, you have to use unmanaged nodes if the Web server is not running on the z/OS platform.

Special case: If the unmanaged Web server is an IBM HTTP Server, you can administer it from the Integrated Solutions Console. This allows you to 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.

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

Web server plug-ins

A Web server can serve static contents and requests, like 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 transfer (manually or automatically with the deployment manager) 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 (Figure 3-10).

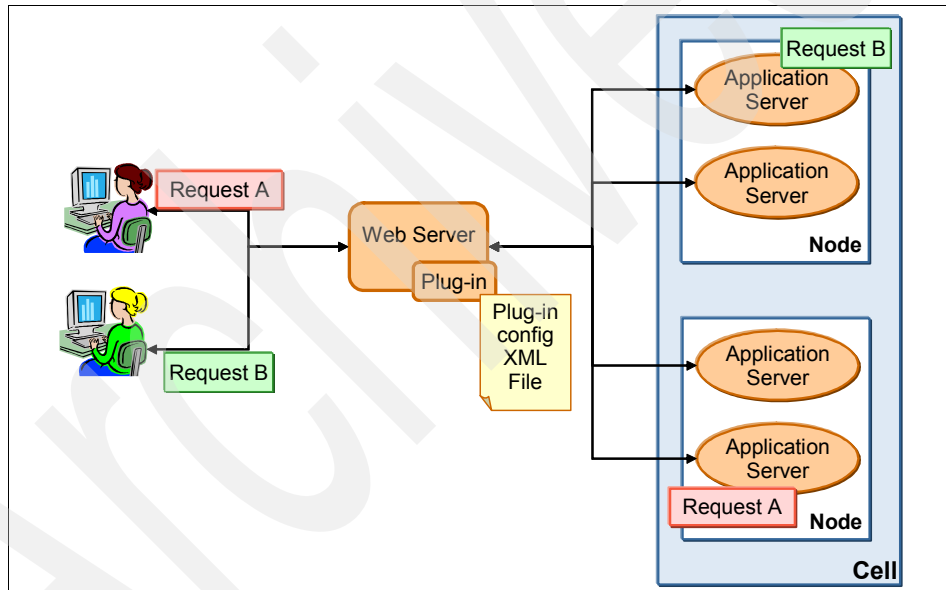


Figure 3-10 Web server plug-in concept with WebSphere Application Server

The plug-in is used for routing requests to one of multiple application servers.

About the sysplex distributor: On the z/OS platform you can also use the sysplex distributor (SD) for the distribution of requests. The SD allows you to send requests to the server that keeps its predefined response time goals best, by questioning the Workload Manager (WLM) component. The SD is an integral part of the System z operating system.

3.1.10 Proxy servers

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

WebSphere Application Server allows you to create two types of proxy servers:

- ▶ WebSphere Application Server Proxy
- ▶ DMZ Secure Proxy Server

WebSphere Application Server Proxy

WebSphere Application Server Proxy supports two protocols:

- ▶ HTTP
- ▶ SIP

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

HTTP proxy

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

You can modify an existing proxy server to perform advanced routing options, such as routing requests to a non-WebSphere application server. You may also modify a proxy server to perform caching.

Note: When using WebSphere Application Server for z/OS V7.0, the proxy server uses the Workload Management component to perform dynamic routing.

SIP proxy

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

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

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

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

WebSphere Application Server Proxy is not considered a secure proxy for demilitarized zone (DMZ) deployments. For example, it cannot bind to protected ports without being a privileged user on most operating systems and users cannot be switched after binding. WebSphere Application Server Proxy must stay in the intranet/secure zone.

DMZ Secure Proxy Server

As the WebSphere Application Server Proxy is not ready for a DMZ, WebSphere Application Server V7.0 ships a DMZ-hardened version of WebSphere Application Server Proxy. The DMZ Secure Proxy Server comes in a separate install package that contains a subset of WebSphere Application Server Network Deployment and provides new security enhancements to allow deployments inside of a demilitarized zone (Figure 3-11 on page 69).

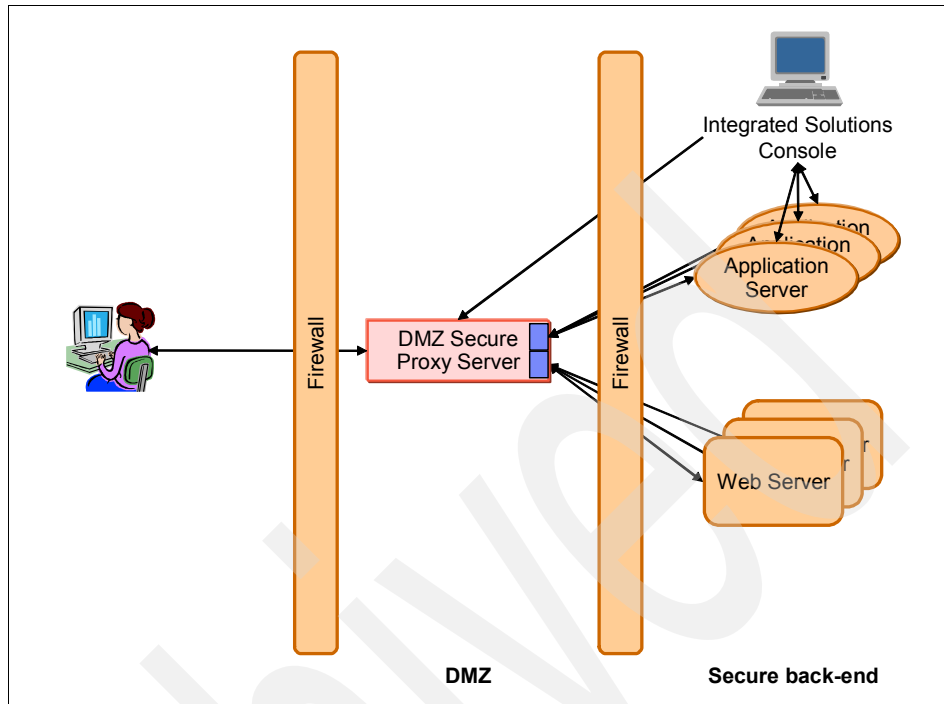


Figure 3-11 DMZ secure proxy simplified topology

The DMZ Secure Proxy is designed to improve security by minimizing the number of external ports opened, and running as an unprivileged user when binding to well-known ports. It can be managed locally or remotely using the job manager console.

Note: HTTP and SIP protocols supported by WebSphere Application Server Proxy are also supported with DMZ Secure Proxy Server.

For a sample topology using the DMZ Secure Proxy Server as reverse proxy, see 5.3.4, “Reverse proxy topology” on page 154.

3.1.11 Generic servers

A generic server is a server that is managed in the WebSphere Application Server administrative domain even though the server is not supplied by WebSphere Application Server. The WebSphere Application Server generic servers function enables you to define a generic server as an application server instance within the WebSphere Application Server administration, and associate it with a WebSphere Application Server or process.

There are two basic types of generic application servers:

- ▶ Non-Java applications or processes
- ▶ Java applications or processes

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

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

3.1.12 Business level applications

Business level application (BLA) is a notion of an application beyond Java EE's definition. This is a new administration concept that expands the options previously offered by Java EE. This grouping notion for enterprise-level applications includes WebSphere and non-WebSphere artifacts like Service Component Architecture (SCA) packages, libraries, and proxy filters under a single application definition (Figure 3-12 on page 71). Every artifact in the group is a composition unit.

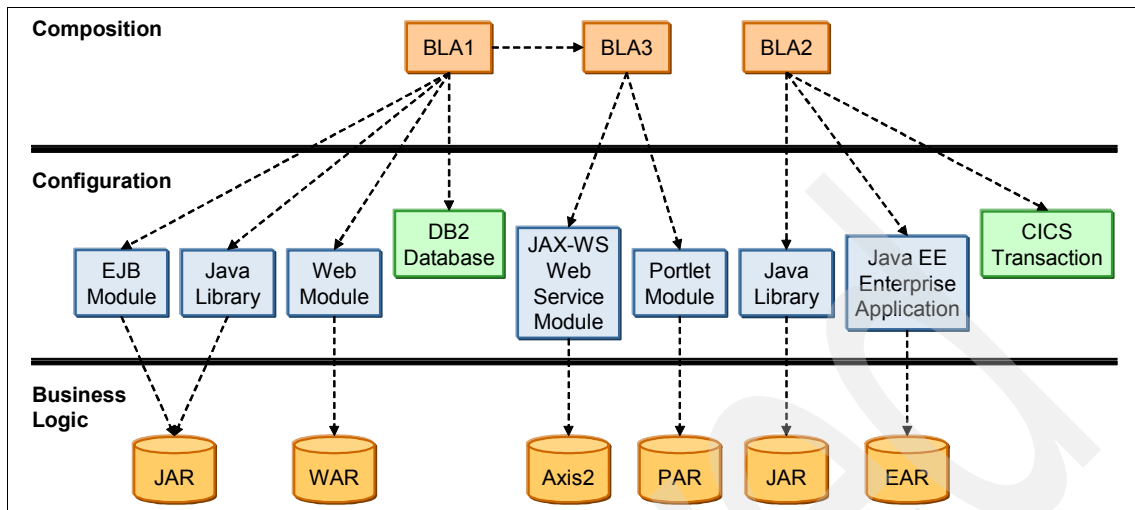


Figure 3-12 Business level applications

A business level application has the following characteristics:

- ▶ Is a configuration that lists one or more composition units, which represent the application binary files
- ▶ Might not explicitly manage the life cycle of every artifact

It is a model for defining an application and does not represent or contain application binary files

- ▶ Can span more than WebSphere Application Server deployment target runtimes, such as a proxy server, a Web server, WebSphere Application Server Community Edition, and so on
- ▶ Provides the following management features for applications:
 - Install
 - Distribute
 - Activate
 - Monitor
 - Update
 - Remove
- ▶ Supports Application Service Provider (ASP) scenarios by allowing single application binaries to be shared between multiple deployments

About ASPs: An ASP is a company providing information technology and computer-based services to its customers. Software created using the ASP model is also called on-demand software or software as a service.

- ▶ Aligns WebSphere applications closer with business as opposed to IT configuration

The overall logical application is intended to represent some function recognizable to the business.

In summary, a BLA can be useful when an application has the following characteristics:

- ▶ Is composed of multiple packages
- ▶ Applies to the post-deployment side of the application life cycle
- ▶ Contains additional libraries, or non-Java EE artifacts
- ▶ Includes artifacts that run on heterogeneous environments that include WebSphere and non-WebSphere runtimes
- ▶ Is defined in a recursive manner (for example, if an application includes other applications)

3.1.13 Centralized installation manager

In V7.0, the Network Deployment packaging adds the capability to perform centralized installations from the deployment manager to remote nodes through its centralized installation manager component.

Centralized installation manager enables a single installation to be pushed out as an installation package from the deployment manager to a series of endpoints. The endpoint can either be a node that is not part of a Network Deployment cell or an existing Network Deployment node that might need a fix pack.

With centralized installation manager, an administrator can remotely install or uninstall product components or maintenance to specific nodes directly from the Integrated Solutions Console without having to log in and repetitively complete these tasks for each node. Using centralized installation manager is a good way to shorten the number of steps that are required to create and manage your environment, and for an easier installation and patch management.

3.1.14 Intelligent runtime provisioning

Intelligent runtime provisioning is a new concept introduced with WebSphere Application Server V7.0. This mechanism selects only the runtime functions needed for an application (Figure 3-13 on page 73). Each application is examined by WebSphere Application Server during the deployment to generate an activation plan. At run time, the server uses the activation plan to start only those components that are required inside the application server.

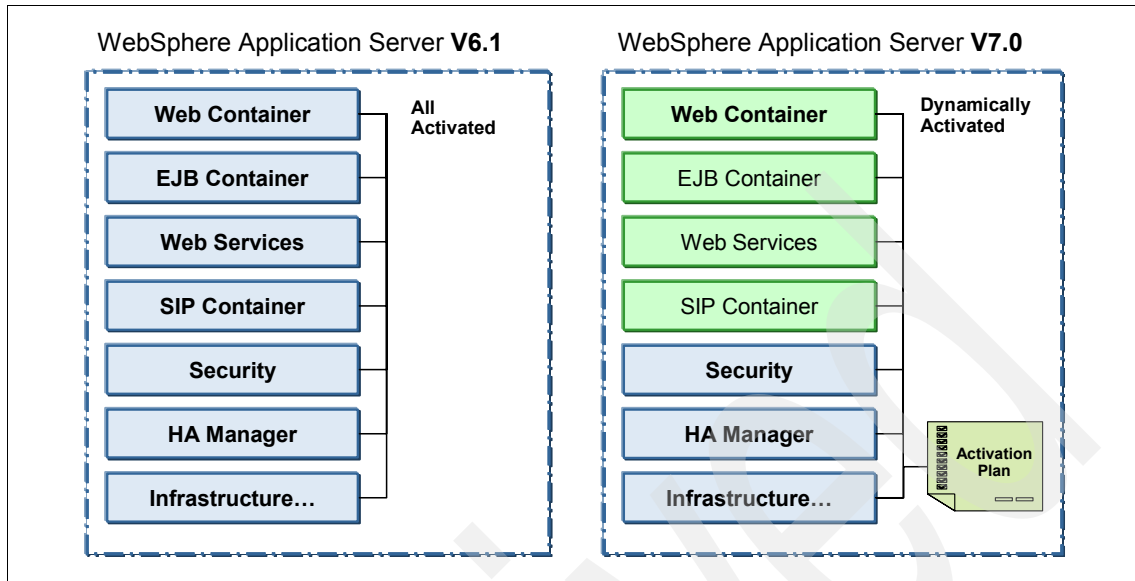


Figure 3-13 Intelligent runtime provisioning

Intelligent runtime provisioning is a feature that reduces memory footprint, application server startup time, and used CPU resources needed to start the application server.

3.2 Server environments

WebSphere Application Server enables you to build various server environments, consisting of single and multiple application servers maintained from central administrative points. This section provides information about the different configurations that can be created using WebSphere Application Server V7.0:

- ▶ “Single cell configurations” on page 74
- ▶ “Multiple cell configurations” on page 77
- ▶ “Mixed node versions in a cell” on page 77
- ▶ “Flexible management” on page 78

3.2.1 Single cell configurations

You can group nodes into a single administrative domain with cell configurations. WebSphere Application Server allows you to create two types of configurations in a single cell environment:

- ▶ Single system configurations
- ▶ Multiple systems configurations

Systems: A system can be one of the following options:

- ▶ A server machine, if it contains only one operating system
- ▶ An operating system image, if the host server machine contains multiple operating system images (for virtualization purposes for example)
- ▶ A z/OS image

Single system configurations

With the Base and Express configurations, you can only create a cell that contains a single node with a single application server (Figure 3-14).

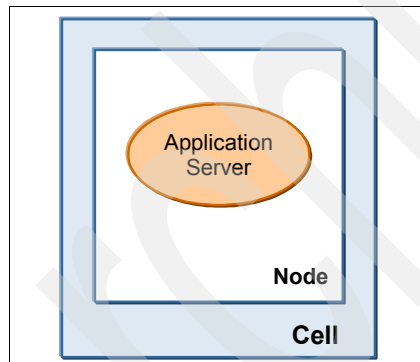


Figure 3-14 Cell configuration in Base and Express packages

Note: A server system can contain multiple Base and Express profiles installations, therefore multiple cells.

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

A single system configuration in a distributed environment includes all processes in one system (Figure 3-15).

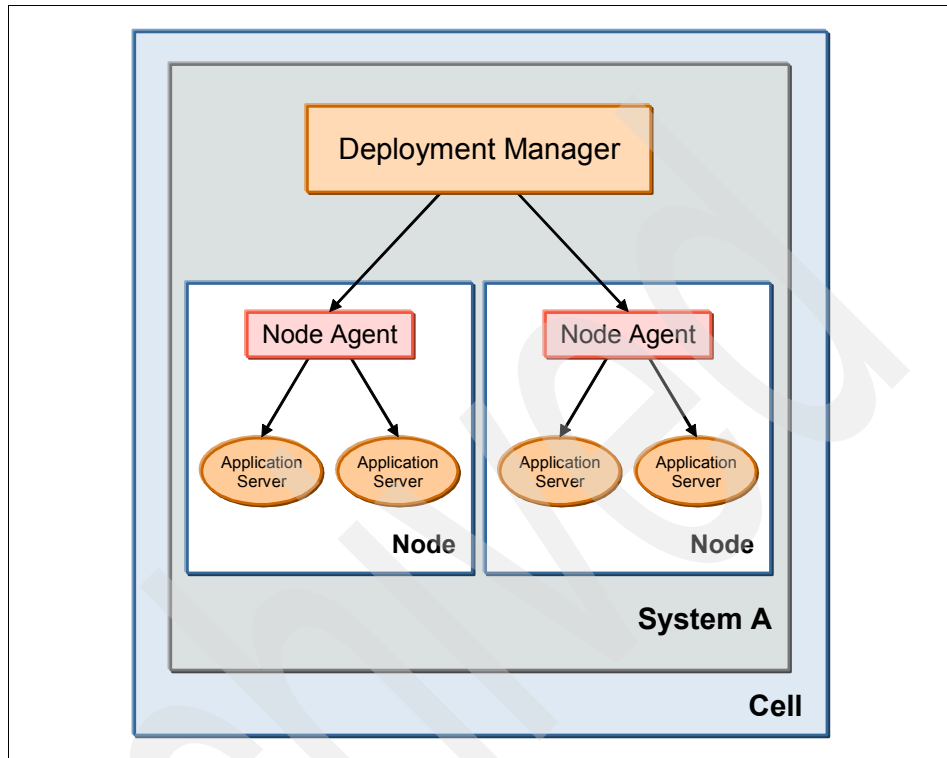


Figure 3-15 Cell configuration option in Network Deployment: Single server system

Multiple systems configurations

As an alternative to a single system configuration, the Network Deployment package allows you to create multiple systems configurations. The deployment manager can be installed on one system (System A) and each node on a different system (System B and System C), as shown in Figure 3-16 on page 76. The servers do not have to be the same platform. For example, System A can be an AIX system while System B is Microsoft Windows and System C is a z/OS image.

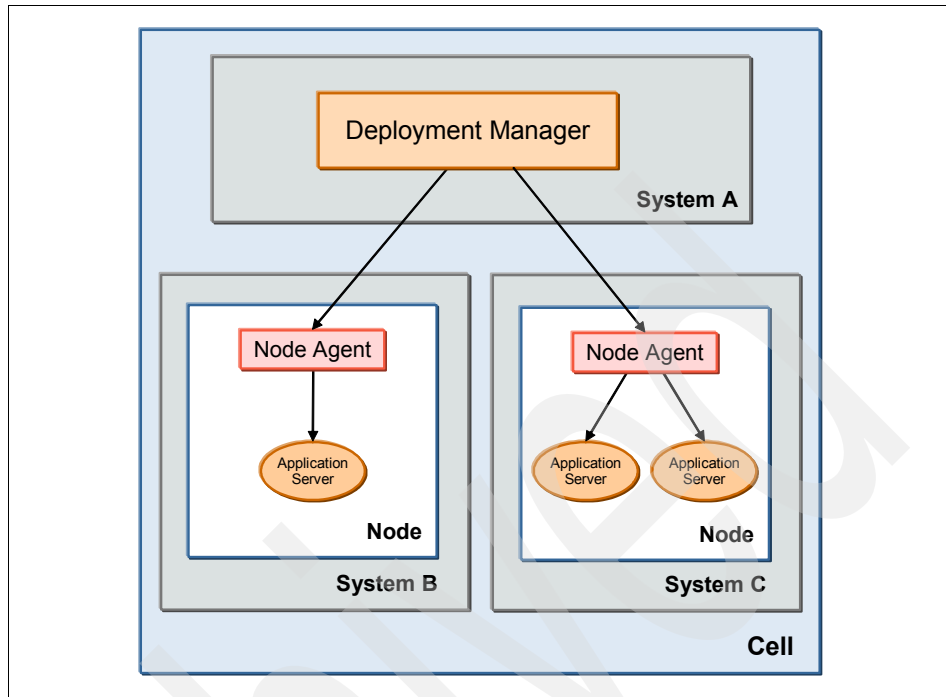


Figure 3-16 Cell configuration option in Network Deployment: Multiple server system

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

A Network Deployment environment gives you the flexibility to install the WebSphere components on server systems and in locations that suit your requirements.

3.2.2 Multiple cell configurations

Cells can reach out over operating system boundaries. This means a cell can reside entirely within one system, or be sprayed among two or more systems, as shown in Figure 3-17.

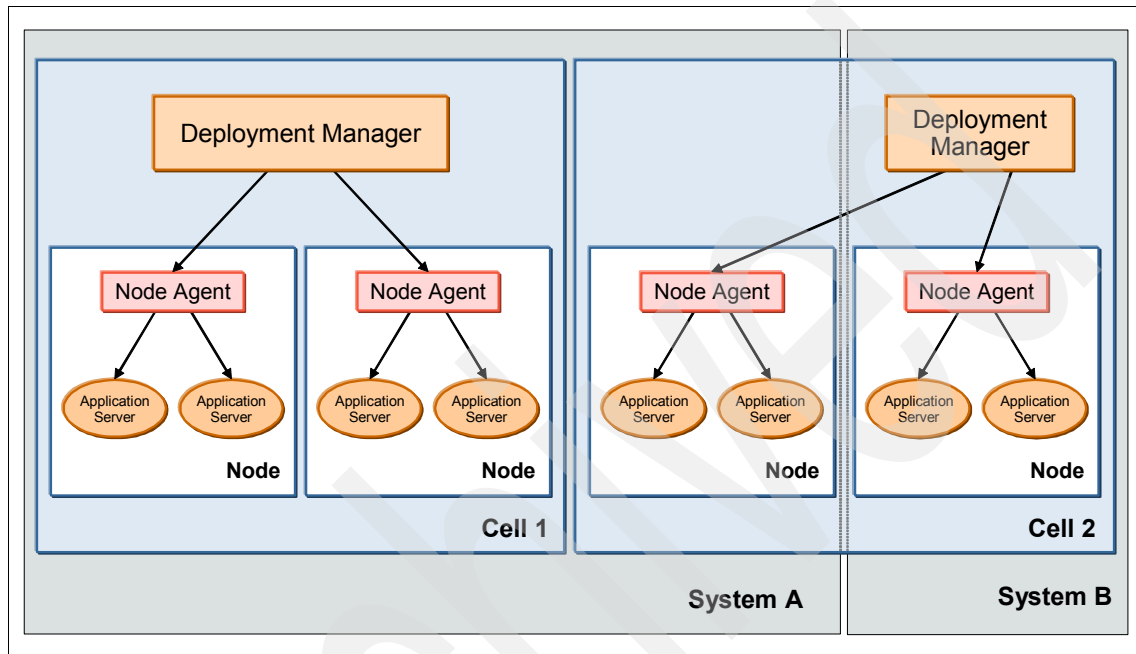


Figure 3-17 Multiple independent cells sharing server systems

3.2.3 Mixed node versions in a cell

In general, a cell can contain nodes from different WebSphere Application Server versions.

Note: Usually version n-3 is the last level that can be included in such a mixed cell. However refer to the documentation of the specific versions that you want to use, to make sure that you have a supported environment.

To administer such a topology, the deployment manager has to be at the level of the highest node version. A WebSphere Application Server Network Deployment V7.0 cell for example, can contain V5.1, V6.0, V6.1, and V7.0 nodes (Figure 3-18 on page 78). This requires that the deployment manager is at V7.0.

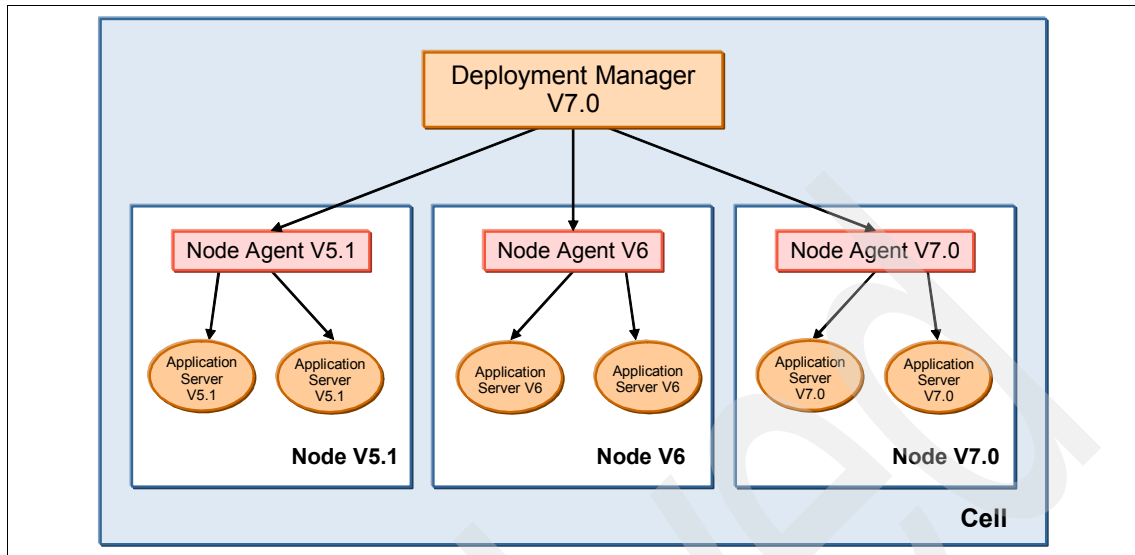


Figure 3-18 Mixed node versions in a Network Deployment cell

Multiple release levels: You can upgrade a portion of the nodes in a cell, while leaving others at a previous release level. Also, you might be managing servers that are running multiple release levels in the same cell. It is recommended to use this feature only for migration scenarios.

3.2.4 Flexible management

Flexible management is a concept introduced with WebSphere Application Server V7.0. With flexible management components like the administrative agent and the job manager, you can build advanced and large-scale topologies and manage single and multiple application server environments from a single point of control. This reduces management and maintenance complexity.

Multiple base profiles

The administrative agent component of WebSphere Application Server provides administration services and functions to manage multiple stand-alone (Express and Base) application servers that are all installed in the same system.

Figure 3-8 on page 63 shows the administrative agent management model introduced with WebSphere Application Server V7.0.

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

Job manager management model and advanced topologies

The job manager component of WebSphere Application Server Network Deployment allows you to build advanced management models and topologies for your environment.

The job manager can manage multiple administrative agents in different systems and can be the single point of control for these stand-alone server profiles (Figure 3-19).

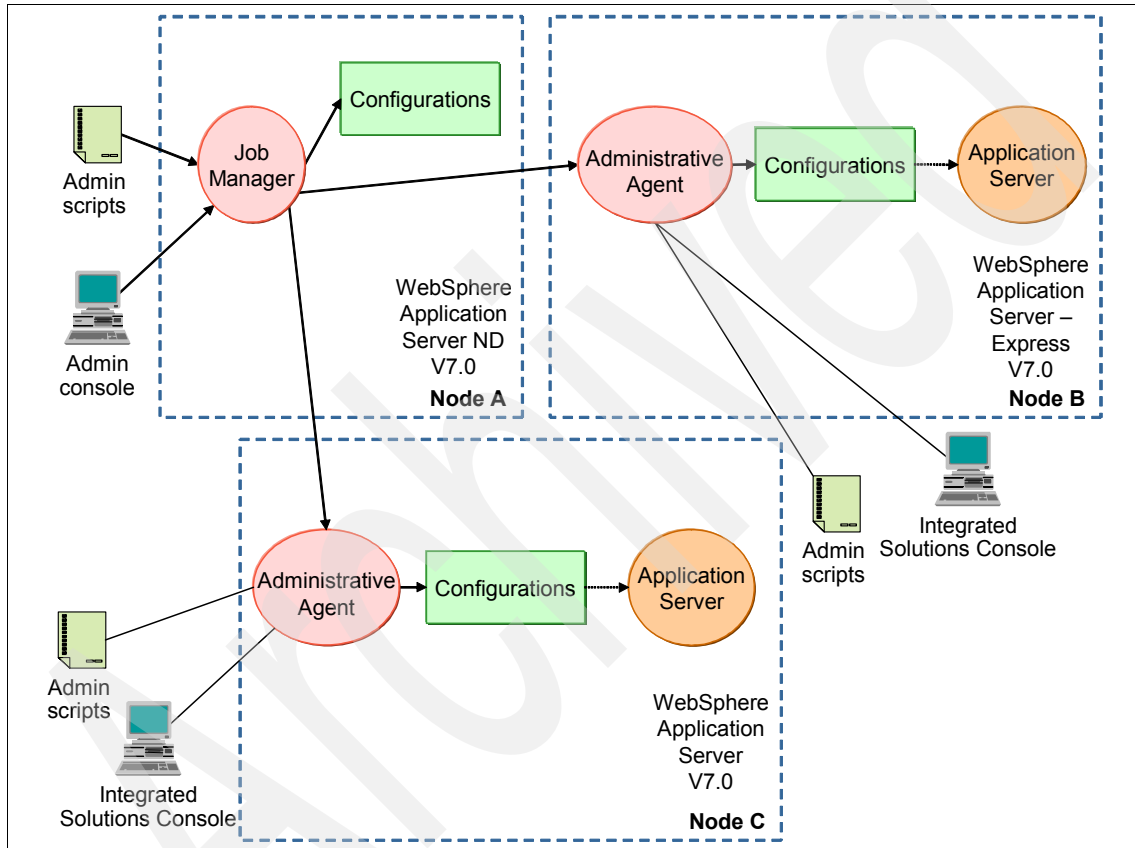


Figure 3-19 Job manager management model for multiple administrative agents

In a deployment manager environment, there is a tight coupling between application servers and node agents and also between node agents and the deployment manager. This tight coupling can impact the scalability of the administrative run time if the runtime components are not located together in close proximity using redundant, high capacity, low latency networks.

The job manager addresses the limitations inherent in the current management architecture by adopting a loosely coupled management architecture. Rather than synchronously controlling a number of remote endpoints (node agents), the job manager coordinates management across a group of endpoints by providing an asynchronous job management capability across a number of nodes.

The advanced management model relies on the submission of management jobs to these remote endpoints, which can be either a WebSphere Application Server administrative agent or a deployment manager. In turn, the administrative agent or the deployment manager executes the jobs that update the configuration, starts or stops applications, and performs a variety of other common administrative tasks (Figure 3-20).

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

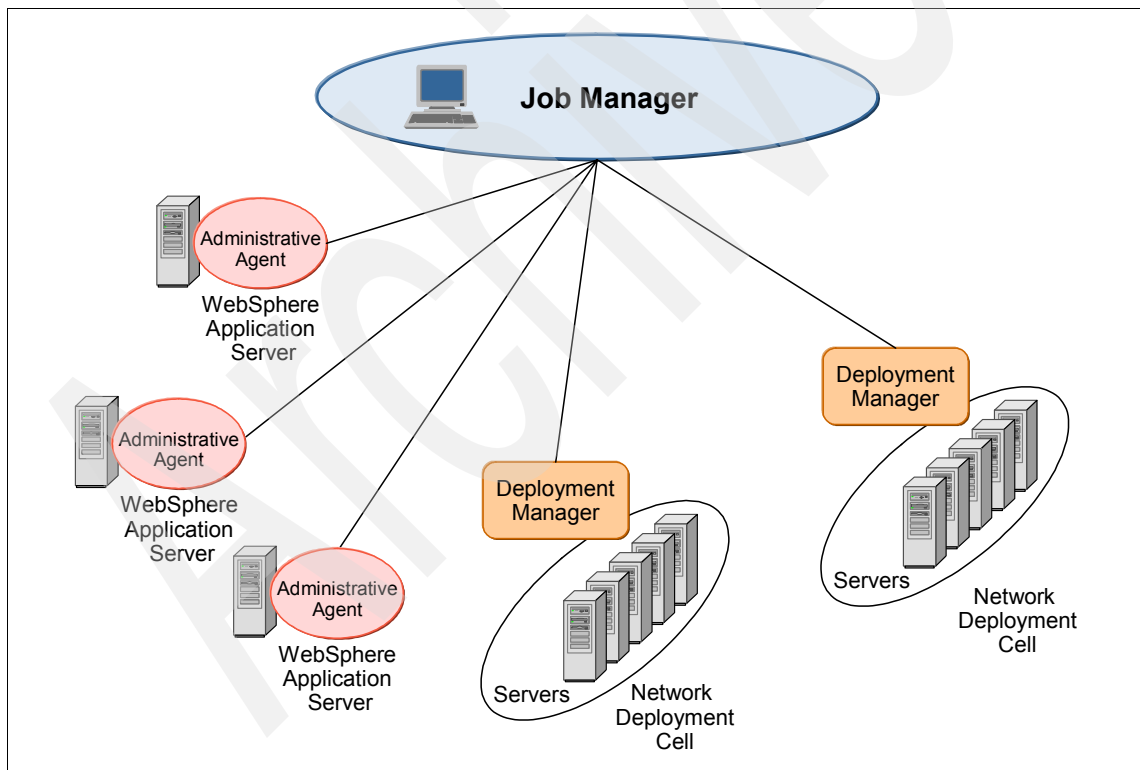


Figure 3-20 Job manager management model

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

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

3.3 Clusters

A cluster is a collection of servers managed together. Clusters behave as a single application server entity to accomplish a job by parallel processing. WebSphere Application Server provides clustering support for the following different types of servers:

- ▶ Application server clusters
- ▶ Proxy server clusters
- ▶ Generic server clusters

3.3.1 Application server clusters

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

Application servers that are a part of a cluster are called cluster members. When you install, update, or delete an application, the updates 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 to the user.

Important characteristics of the application server clusters are as follows:

- ▶ A cell can have multiple or no clusters.
- ▶ A cluster member can only belong to a single cluster.
- ▶ Clusters may span server systems and nodes, but they cannot span cells.
- ▶ A cluster cannot span from distributed platforms to z/OS.
- ▶ A node group can be used to define groups of nodes that have enough in common to host members of a given cluster. All cluster members in a cluster must be in the same node group.

Cluster workload management

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

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.

The server-weighted round robin routing policy ensures a balanced routing distribution based on the set of server weights that have been assigned to the members of a cluster. In the case of horizontal clustering where each node resides on a separate server system, the loss of one server system will not disrupt the flow of requests, because the deployment manager is only used for administrative tasks. The loss of the deployment manager would have no impact on operations and primarily affects configuration activities. You can still use administration scripts to manage your WebSphere Application Server environment.

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

Workload management on the z/OS platform introduces a finer granularity, as well as the usage of real-time performance data, to decide on which member a transaction should be processed, instead of using a static round robin procedure.

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

It is possible to classify the incoming requests in their importance. For example, requests coming from a platinum customer will be processed with higher importance (and therefore faster), than a silver ranked customer.

When there are resource constraints, the WLM component will make sure that the member processing the higher prioritized requests, gets additional resources (like CPU), protecting the response time of your most important work.

About WLM changes: The WLM component can change the amount of CPU, I/O and memory resources assigned to the different operating system processes (the address spaces). To decide whether or not a process is eligible for receiving additional resources, the system checks whether or not the process keeps its defined performance targets, and whether there is more important work in the system. This technique is performed dynamically so there is no need for manual interaction after the definitions are made by the system administrator (the system programmer).

For more information about workload management on the z/OS platform in combination with WebSphere Application Server for z/OS, refer to 14.1.6, “Workload management for WebSphere Application Server for z/OS” on page 428.

High availability

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

On the z/OS platform the WebSphere Application Server V7.0 high availability manager can now use native z/OS cluster technology, the cross-system coupling (XCF) services. This reduces the amount of CPU processing used for the keep alive check of clusters, while improving the time it takes to detect a failed member.

About XCF services: The XCF services allow applications that are located on multiple z/OS images to communicate with each other as well as to monitor their status. In the case of WebSphere Application Server for z/OS, the applications are the various cluster members.

For more information refer to 14.5, “XCF support for WebSphere HA manager” on page 438.

In a high availability environment, a group of clusters can be defined as a core group. All of the application servers defined as members of a cluster included in a core group are automatically members of that core group. The use of core

groups enables WebSphere Application Server to provide high availability for applications that must always be available to end users. You can also configure core groups to communicate with each other using the core group bridge. The core groups can communicate within the same cell or across cells.

Vertical cluster

All cluster members can reside on the same system. This topology is known as vertical scaling or vertical clustering (Figure 3-21).

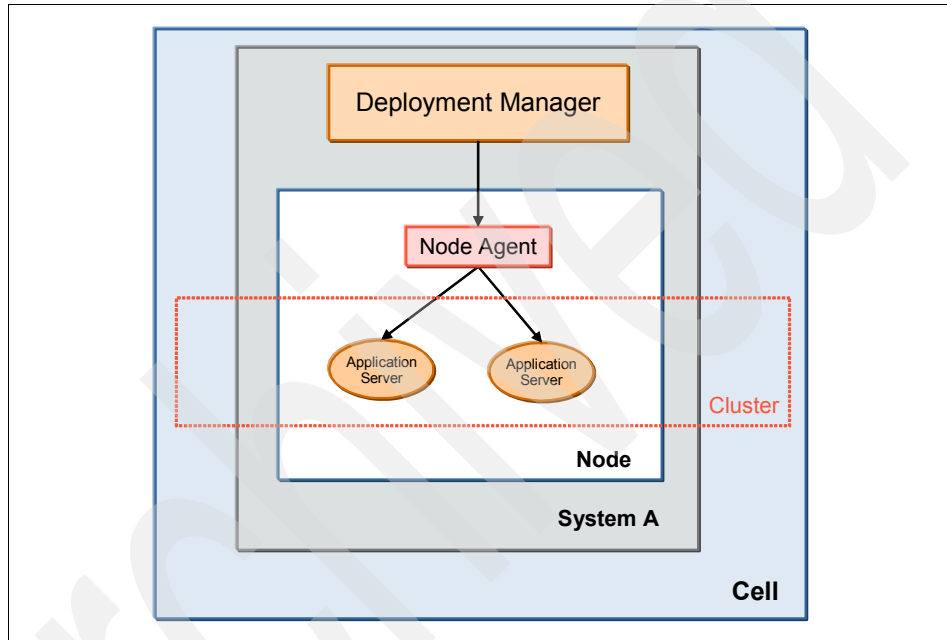


Figure 3-21 A vertical cluster

Vertical clusters offer failover support inside one operating system image and increase the resource use. For more details on vertical clusters and vertical scaling, refer to 5.3.2, “Vertical scaling topology” on page 143.

Horizontal cluster

If cluster members are spread across different server systems and operating system types, it is called horizontal scaling or horizontal clustering (Figure 3-22 on page 85). In this topology, each machine has a node in the cell holding a cluster member. The combination of vertical and horizontal scaling is also possible (Figure 3-23 on page 86).

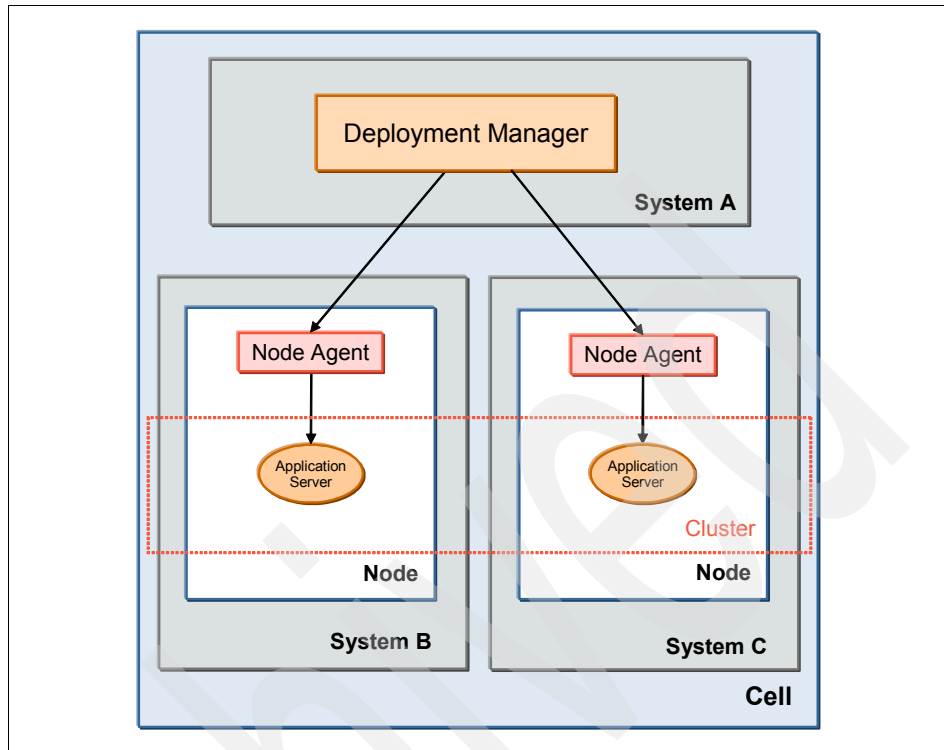


Figure 3-22 A horizontal cluster

Horizontal clusters increase availability by removing the bottleneck of using only one physical system and increasing the scalability of the environment. For more information about horizontal clusters and horizontal scaling see 5.3.3, “Horizontal scaling topology” on page 147.

Multi-servant cluster

Multi-servant clustering is a special clustering topology for WebSphere Application Server for z/OS. It basically allows each single application server (regardless of a stand-alone or a network deployment cell), to use multiple application running processes, to form a single application server image. If one of the servant fails, z/OS makes sure that a new one is started automatically.

The usage of this z/OS function allows you to improve the availability of your environment.

For more information about multi-servant regions, see 14.1.4, “Structure of an application server” on page 422.

Mixed topologies

Figure 3-23 shows a cluster that has four cluster members and is an example for the combination of vertical and horizontal scaling topologies. The cluster uses multiple members inside one operating system image (on one machine) as well as being spread over multiple physical machines.

This cluster also shows the following characteristics:

- ▶ Cluster members can reside in multiple nodes.
- ▶ Only some of the application servers in a node are part of the cluster.
- ▶ Cluster members cannot span cells.

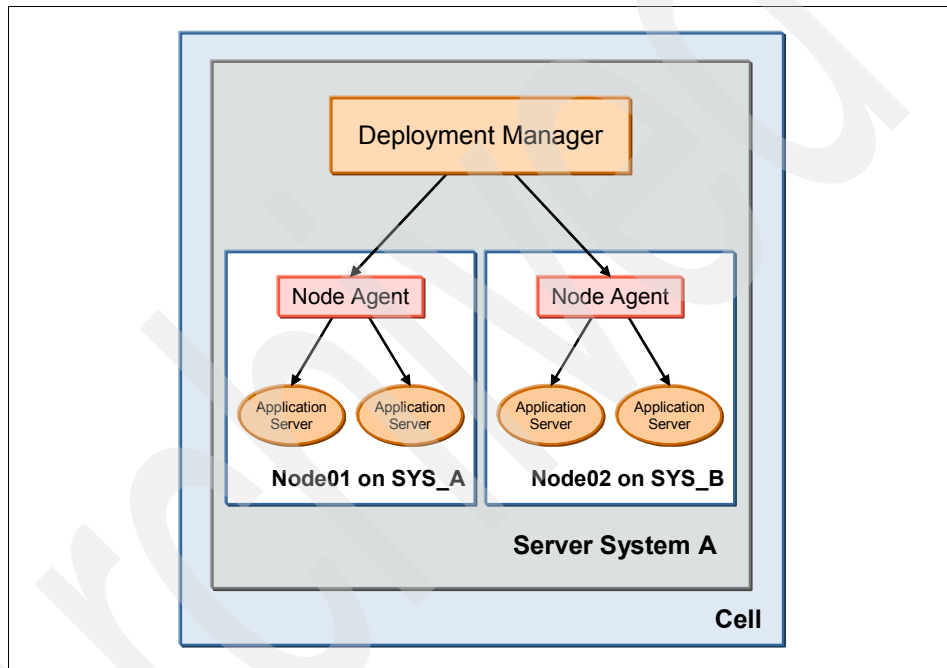


Figure 3-23 An application server cluster and its members

Mixed node versions in a cluster

A WebSphere Application Server Network Deployment V7.0 cluster can contain nodes and application servers from WebSphere Application Server V5.1, V6.0, V6.1, and V7.0. The topology shown in Figure 3-24 on page 87 contains mixed version nodes within a cluster.

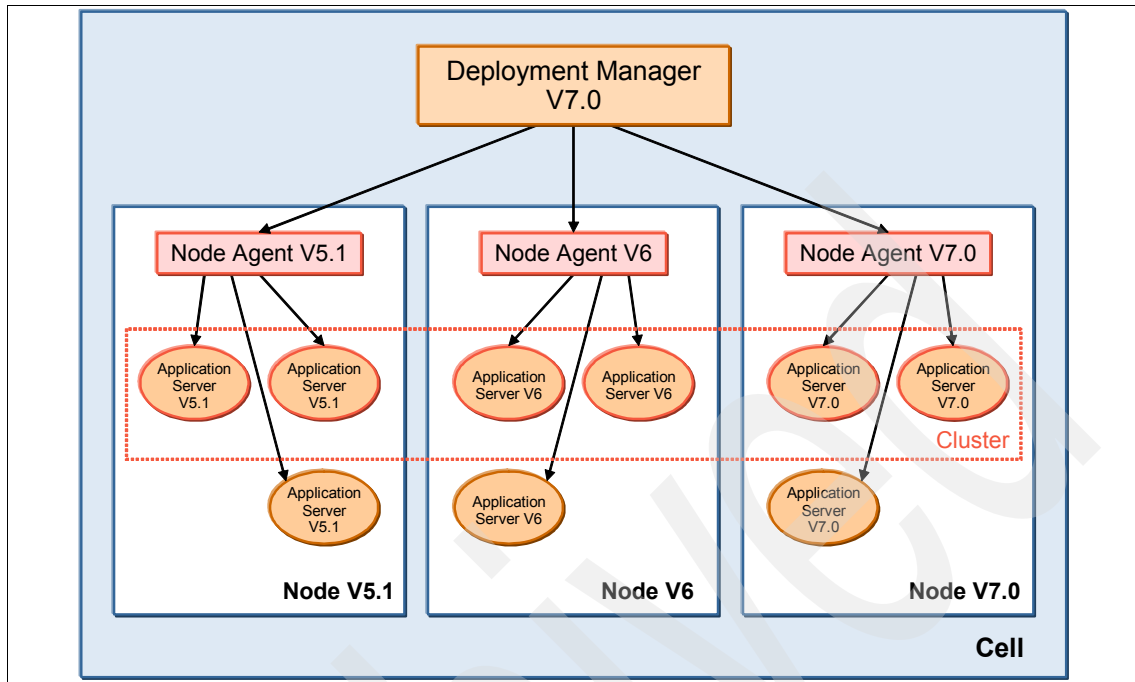


Figure 3-24 Mixed version cluster nodes in a cell

Note: You can upgrade a portion of the nodes in a cell, while leaving others at a previous release level in stable cells. It is suggested to use this feature only for migration scenarios.

3.3.2 Proxy server clusters

A proxy server cluster consists of a group of proxy servers that can route requests and traffic to applications in a cell.

3.3.3 Generic server clusters

A generic server cluster allows you to configure external servers (non-IBM application servers or a pre-V6 WebSphere Application Server) into a logical cluster that can be used by the proxy server to route requests. It defines the server endpoints to which URI groups are mapped.

With generic server clusters, generic servers host a common set of resources and can receive routing actions as a unit.

3.4 Runtime processes

In this section, we discuss how WebSphere Application Server processes execute at run time. The executable processes include application servers, node agents, administrative agents, deployment managers, and job managers. Note that cells, nodes, and clusters are administrative concepts and not executable components.

3.4.1 Distributed platforms

On distributed platforms (Windows, AIX, Linux, and so on), WebSphere Application Server is built using a single process model where the entire server runs in 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 is visible in the Windows Task Manager. Starting a node agent starts a second `java.exe` process, and each application server started is seen as a `java.exe` process (Figure 3-25).

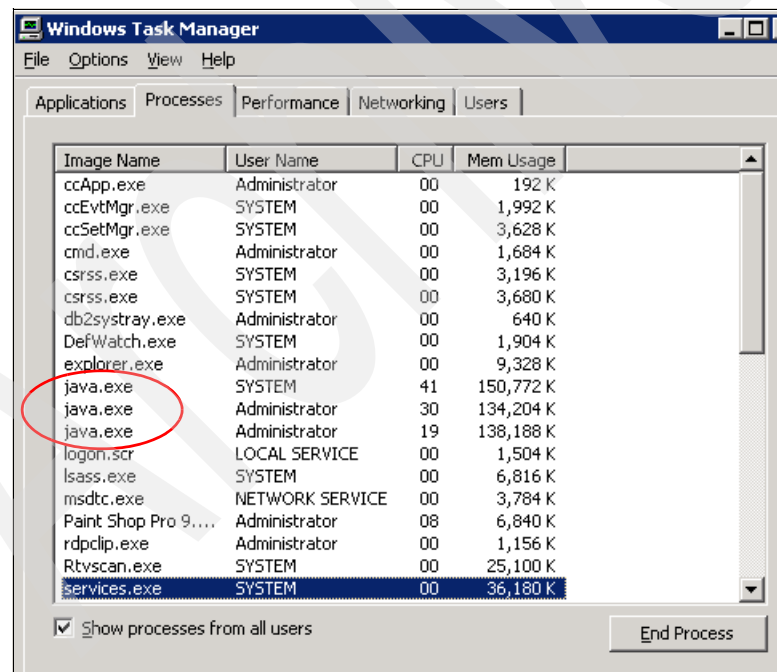


Figure 3-25 WebSphere Application Server processes on a Windows system

3.4.2 WebSphere Application Server for z/OS

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

You have the option to create multiple processes, z/OS address spaces, that run your application. This form of mini-cluster, the multi-servant cluster, enhances the performance and availability of your application, because a failure in one of these servants does not harm the others. Each servant runs its own JVM and its own copy of the application.

For detailed information about the WebSphere Application Server for z/OS runtime, see 14.1.5, “Runtime processes” on page 425 in the z/OS chapter.

3.5 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.

Managed and unmanaged nodes: When you define a Web server to WebSphere Application Server, it is associated with a node. The node is either a managed or an unmanaged. When we refer to managed Web servers, we are referring to a Web server defined on a managed node. 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.

3.5.1 Managed Web servers

Defining a managed Web server allows you to start and stop the Web server from the Integrated Solutions Console and push the plug-in configuration file to the Web server. A node agent must be installed on the Web server machine. An exception is if the Web server is the IBM HTTP Server. See 3.5.3, “IBM HTTP Server as an unmanaged Web server (special case)” on page 92 for more information. Figure 3-26 on page 90 illustrates a Web server managed node.

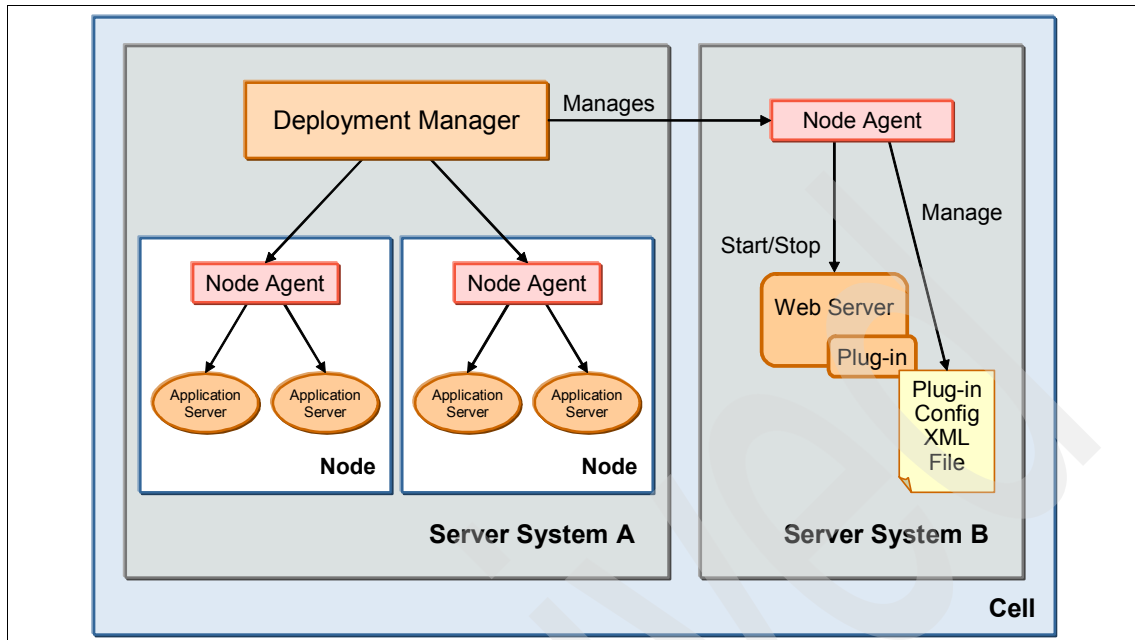


Figure 3-26 Web server managed node

3.5.2 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 3-27 on page 91 illustrates a Web server unmanaged node.

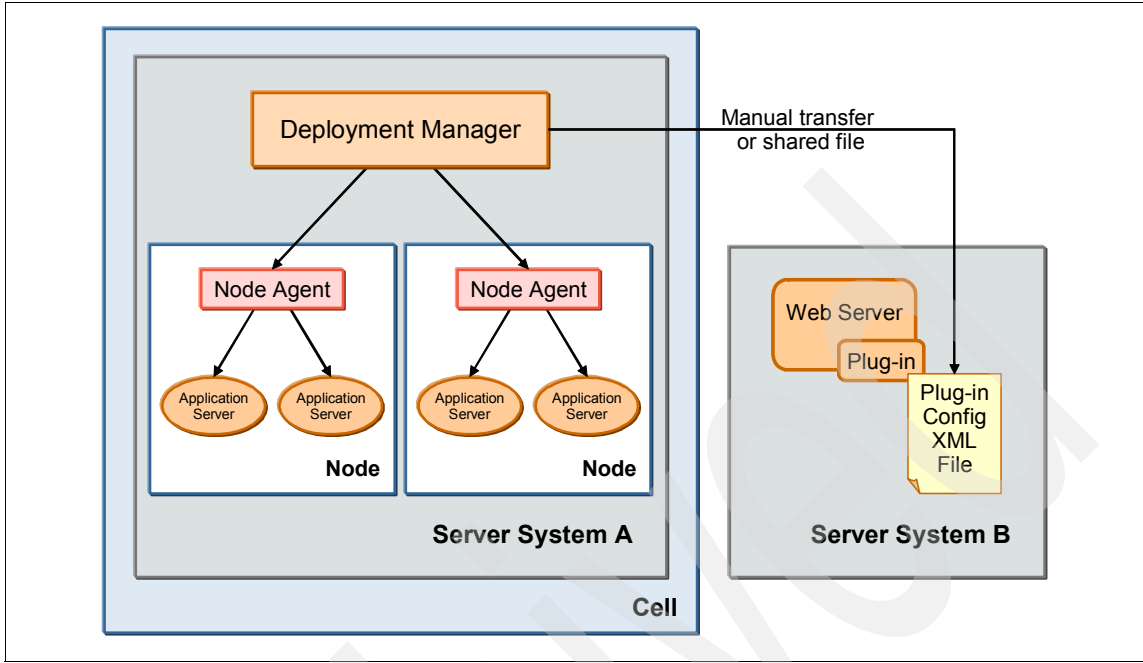


Figure 3-27 Web server unmanaged node

3.5.3 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 3-28 illustrates an IBM HTTP Server unmanaged node.

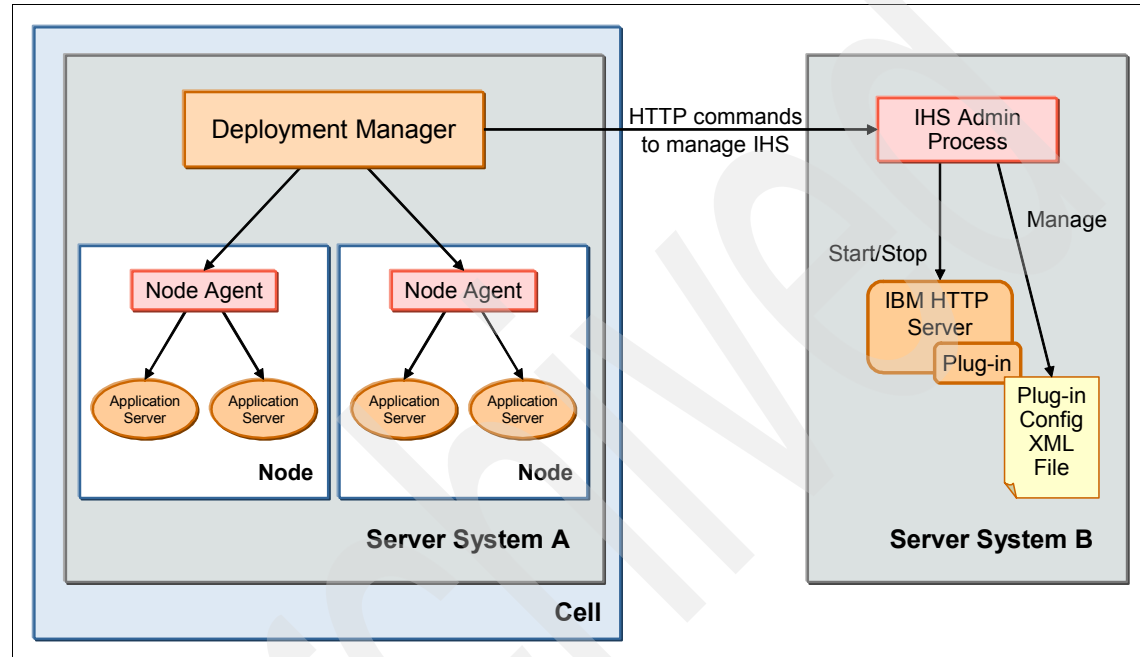


Figure 3-28 IBM HTTP Server unmanaged node

Infrastructure

There are many things to consider when planning and designing an infrastructure for a WebSphere Application Server environment. This chapter describes the most important components specific to WebSphere Application Server that you have to understand in order to run a successful WebSphere infrastructure project.

This chapter contains the following sections:

- ▶ “Infrastructure planning” on page 94
- ▶ “Design considerations” on page 95
- ▶ “Sizing the infrastructure” on page 106
- ▶ “Benchmarking” on page 107
- ▶ “Performance tuning” on page 108
- ▶ “Planning for monitoring” on page 114
- ▶ “Planning for backup and recovery” on page 117
- ▶ “Planning for centralized installation” on page 119

Note: The terms *machine* and *system* stands as a synonym for physical machines, logical partitions (LPARs) and operating system image.

4.1 Infrastructure planning

This section gives a general overview of the typical phases you have to go through during a project. It includes how to gather requirements and how to apply these requirements to a WebSphere Application Server project.

Typically, a new project starts with only a concept. Little is known about specific implementation details, especially how they relate to the infrastructure. Your development team and infrastructure team must work closely together to cater to the needs of the overall application environment.

Bringing together a large team of people can create an environment that helps hone the environment requirements. However, if unfocused, a large team can wander aimlessly and create more confusion than resolved issues. For this reason, carefully consider the size of the requirements gathering 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**
Functional requirements are usually determined by business on the use of the application and related functions.
- ▶ **Non-functional requirements**
Non-functional requirements describe the properties of the underlying architecture and infrastructure such as reliability, scalability, availability, or security.
- ▶ **Capacity requirements**
Capacity requirements include traffic estimates, traffic patterns, and expected audience size.

Requirements gathering is an iterative process. Make sure that your plans are flexible enough to deal with future changes in requirements. Always keep in mind that the plans can impact other parts of the project. To support this, make sure that dependencies and data-flows, whether application or infrastructure-related, are clearly documented.

With this list of requirements, you can start to create the first draft of your design. Target developing the following designs:

- ▶ Application design

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

- ▶ Implementation design

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

The final version of this implementation design will contain details about the hardware, processors, and software that is 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.

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 in 4.3, “Sizing the infrastructure” on page 106. In some cases, it might be appropriate to perform benchmark tests. There are many ways to perform benchmarking tests, and in 4.4, “Benchmarking” on page 107, 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 specified. For more details about how to plan for load tests, see 4.5, “Performance tuning” on page 108.

4.2 Design considerations

The following sections discuss some points to consider when designing a WebSphere Application Server deployment. They will most likely impact your design significantly. We will talk in more detail about:

- ▶ “Scalability” on page 96
- ▶ “Caching” on page 98
- ▶ “High availability” on page 99
- ▶ “Load-balancing and fail-over” on page 100
- ▶ “Disaster recovery” on page 101
- ▶ “Security” on page 102
- ▶ “Application deployment” on page 104
- ▶ “Servicability” on page 105

4.2.1 Scalability

Scalability, as used in this book, means the ability of the environment to grow as the infrastructure load grows. Designing your infrastructure to be scalable therefore, means to design your infrastructure with the ability to grow.

Scalability depends on many different factors such as hardware, operating system, middleware, and so on. For example, z/OS systems usually scale better for business transactions than other platforms.

Understanding the scalability of the components in your WebSphere Application Server 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 associated with IP load balancers, such as the following components:

- ▶ Dispatchers or edge servers
- ▶ Web presentation servers
- ▶ Web application servers
- ▶ Data servers
- ▶ Transaction servers
- ▶ Logical partitions (LPARs)

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 must be ensured that the applications are designed for scaling. It is important to understand the component flow and traffic volumes associated with existing applications and to evaluate the nature of new applications. It is essential to understand each single component and system being used in the transaction flow. Evaluate how scaling can be applied for each of these components and applications. Different types of applications represent different workload patterns. For example, online banking applications might experience the greatest workload at the database server, while other applications might experience the greatest workload at the application server.

2. Categorize your workload.

Knowing the workload pattern for a site determines where you focus scalability efforts and which scaling techniques you need to apply. For example, a customer self-service site, such as an online bank, needs to focus on transaction performance and the scalability of databases containing 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 the following site types:

- 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. From a scalability viewpoint, the key components of the infrastructure are the load balancers, the application servers, security services, transaction and data servers, and the network. First focus on those components which are most heavily used by the key transactions of your applications.

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 compromise any of the above mentioned critical factors.

Choose from the following scaling techniques:

- Using a faster machine
- Using multiple machines
- Creating clusters
- Using appliance servers
- Segmenting the workload
- Using batch requests
- Aggregating user data
- Managing connections
- Using caching techniques
- Using intelligent virtualization techniques

5. Apply the techniques.

Testing is key to successful application deployment. It is crucial that you determine not only if the scaling techniques are effective, but that they do not adversely affect other areas. Keep in mind that all components making up the infrastructure for your application environment must be balanced. Only when 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. Scalability is not a one time design consideration, It is part of the growth of the environment.

The developerWorks article *Design for Scalability - an Update* provides a detailed discussion about these steps. It can be accessed at the following Web page:

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

We expand on some of the scaling techniques in Chapter 7, “Performance, scalability, and high availability” on page 229.

4.2.2 Caching

Caching is a widely used technique to improve performance of application server environments. If planning for a high volume web site, caching will most likely be required to achieve satisfying performance results at acceptable cost.

WebSphere Application Server provides many different caching features at different locations in the architecture.

WebSphere Application Network Deployment provides caching features at each possible layer of the infrastructure:

- ▶ Infrastructure edge:
 - Caching Proxy provided by the Edge Components
 - WebSphere Proxy Server
- ▶ HTTP server layer:
 - Edge Side Include (ESI) fragment caching capabilities provided by the WebSphere plug-in
 - Caching capabilities of the HTTP server itself (like the Fast Response Cache Accelerator (FRCA) as provided by most implementations of IBM HTTP Server)
- ▶ Application server layer:
 - Dynamic caching service inside the application server’s JVM
 - WebSphere Proxy Server

In addition to the caching features provided by WebSphere Application Server Network Deployment, you might consider using third party caching devices or external caching infrastructures provided by IBM and third parties.

To use the caching mechanisms provided by WebSphere Application Server and other components of your environment, the application must be designed for caching as well. Therefore it is suggested to design caching in close cooperation with the application architect.

After you are done with designing your cache locations complete your implementation design to include the caching components.

4.2.3 High availability

Designing an infrastructure for high availability means to design an infrastructure in a way that your environment can survive the failure of one or multiple components. High availability implies redundancy by avoiding any single point of failure on any layer (network, hardware, processes, and so forth). The number of failing components your environment has to survive without losing service depends on the requirements for the specific environment.

The following list helps to identify the high availability needs in your infrastructure:

- ▶ Talk to the sponsor of your project to identify the high availability needs for each of the services used. Because high availability in most cases means redundancy, this also means that high availability will increase the cost of the implementation.

Not every service has the same high availability requirements. Therefore it would be a waste of effort to plan for full high availability for these types of services. Be careful when evaluating, as the high availability of the whole systems depends on the least available component. Carefully gather where and what type of high availability you really need to be able to meet the service level agreement (SLA) and non-functional requirements.

- ▶ When you have gathered the high availability requirements, review every component of the implementation design you developed in 4.1, “Infrastructure planning” on page 94, and determine how significant this component is for the availability of the service, and how a failure would impact the availability of your service.
- ▶ Evaluate every component you identified in the previous step against the following checklist:

- How critical is the component for the service?

The criticality of the component will have an impact on the investments you are willing to take to make this component highly available.

- Consider regular maintenance.
In addition to failures of components, consider maintenance and hang situations.
- Is the service under your control?
Sometimes you are deal with components in the architecture that are out of your control. For example, external services provided by someone else.
If the component is out of your control, document this as an additional risk and inform the project sponsor.
- What needs to be done to make the component highly available?
Sometimes you have more than one option to make a component highly available. You have to make a selection as to which best fits your requirements.
- Does the application handle outages in a defined way?
Check with the application developers on how the application handles an outage of a component. Depending on the component and the error situation, the application might need a specific design or error recovery coded before using high availability features of the infrastructure.
- Prioritize your high availability investments.
Decide the high availability implementation based on the criticality of the component and the expected outage rate. Document any deviations from the requirements gathering.
- Size every component in a way that it can provide sufficient capacity even in case of a failure of a redundant part.

After you are done with your high availability design, update your implementation design to include the high availability features.

4.2.4 Load-balancing and fail-over

As a result of the design considerations for high availability you will most likely identify a number of components that need redundancy. Having redundant systems in an architecture requires you to think about how to implement this redundancy to ensure getting the most benefit from the systems during normal operations, and how you will manage a seamless fail-over in case a component fails. This introduces the following techniques to your design:

- ▶ **Load-balancing**

Load-balancing is the technique to spread load across multiple available copies of a component for an optimum usage of the available resources.

► Fail-over

Fail-over is the capability to automatically detect the outage of a component and route requests around the failed component. When the failed resource becomes available, it will be determined automatically by the system and transparently rejoin workload processing.

For designing load-balancing and fail-over, you need to know each component's load-balancing and fail-over capabilities and how they can be used. Depending on what features you use and how you achieve these capabilities, you will require additional hardware and software to gain high availability.

In a typical WebSphere Application Server environment, there are a variety of components that need to be considered when implementing load-balancing and fail-over capabilities:

- Caching proxy servers
- HTTP servers
- Containers (such as the Web, SIP, and Portlet containers)
- Enterprise JavaBeans (EJB) containers
- Messaging engines
- Back-end services (database, enterprise information systems, and so forth)
- User registries

While load-balancing and fail-over capabilities for some of these components is incorporated into WebSphere Application Server, others require additional hardware and software.

For example, to achieve high availability for your HTTP servers or reverse proxy servers, you need IP splitters in front of the HTTP servers or reverse proxy servers. In contrast, fail-over capabilities for WebSphere default messaging is provided by the WebSphere Application Server High Availability Manager (HAManager) when using WebSphere clusters. Keep in mind that the HAManager requires shared storage between all cluster members to be able to fail-over the messaging service. After you have finished your load-balancing and fail-over discussion, update your implementation design to include these features as well.

4.2.5 Disaster recovery

Disaster recovery is a different requirement than high availability. In 4.2.3, “High availability” on page 99, we discussed some considerations about what needs to be done to make sure that our services are highly available. For example, an outage of a single component will not bring our service down. High availability can usually be achieved by avoiding any single point of failure in the architecture and providing sufficient resources.

Disaster recovery concentrates on the actions, processes, and preparations to recover from a disaster that has struck your infrastructure. Important points when planning for disaster recovery are as follows:

- ▶ Recovery Time Objective (RTO)
How much time can pass before the failed component must be up and running?
- ▶ Recovery Point Objective (RPO)
How much data loss is affordable? The RPO sets the interval for which no data backup can be provided. If no data loss can be afforded, the RPO would be zero.

Planning for disaster recovery is a complex task, and requires an end-to-end planning for all components of your data center. For information on WebSphere Application Server and disaster recovery, refer to the developerWorks article *Everything you always wanted to know about WebSphere Application Server but were afraid to ask, Part 5* available at the following Web page:

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

4.2.6 Security

WebSphere Application Server V7.0 is a powerful Java-based middleware application server ready to run your mission critical applications and business transactions. For this reason we need to design proper security in our WebSphere Application Server infrastructure from the start of the design phase.

WebSphere Application Server provides a continuously expanding and flexible security infrastructure you can use to make the access to your applications and data more secure. The security model of WebSphere Application Server was enhanced in Version 7.0 by adding the following features (for more information see 12.1, “What is new in V7.0” on page 380):

- ▶ Support for multiple security domains
- ▶ Security auditing
- ▶ Improvements for certificate management
- ▶ SPNEGO and Kerberos¹ support
- ▶ Java EE 5 Security Annotations support

¹ Kerberos support is not part of the base products but is planned to be available in a future fix pack

It is the responsibility of the WebSphere Application Server infrastructure to provide all the services required by applications to run in a secured environment. It is imperative then, to have security in mind when planning for a WebSphere Application Server infrastructure. In many cases you will find reusable components of a security infrastructure already in place.

Consider how security will affect your infrastructure:

- ▶ Understand the security policy and requirements for your future environment.
- ▶ Work with a security subject matter expert to develop a security infrastructure that adheres to the requirements and integrates in the existing infrastructure.
- ▶ Make sure that sufficient physical security is in place.
- ▶ Make sure the application developers understand the security requirements and code the application accordingly.
- ▶ Consider the user registry (or registries) you plan to use. WebSphere Application Server V7.0 supports multiple user registries and multiple security domains.
- ▶ Make sure that the user registries are not breaking the high availability requirements. Even if the user registries you are using are out of scope of the WebSphere Application Server project, considerations for high availability need to be taken and requested. For example, make sure that your LDAP user registries are made highly available and are not a single point of failure.
- ▶ Define the trust domain for your environment. All computers in the same WebSphere security domain trust each other. This trust domain can be extended, and when using SPNEGO / Kerberos, even out to the Windows desktop of the users in your enterprise.
- ▶ Assess your current implementation design and ensure that every possible access to your systems is secured.
- ▶ Consider the level of auditing required and how to implement it.
- ▶ Consider how you will secure stored data. Think of operating system security and encryption of stored data.
- ▶ Define a password policy, including considerations for handling password expirations for individual users.
- ▶ Consider encryption requirements for network traffic. Encryption introduces overhead and increased resource costs, so use encryption only where appropriate.
- ▶ Define the encryption (SSL) endpoints in your communications.

- ▶ Plan for certificates and their expiration:
 - Decide which traffic requires certificates from a trusted certificate authority and for which traffic a self-signed certificate is sufficient. Secured connections to the outside world usually use a trusted certificate, but for connections inside the enterprise, self-signed certificates are usually enough.
 - Develop a management strategy for the certificates. Many trusted certificate authorities provide online tools which support certificate management. But what about self-signed certificates?
 - How are you going to back up your certificates? Always keep in mind that your certificates are the key to your data. Your encrypted data is useless if you lose your certificates.
 - Plan how you will secure your certificates. Certificates are the key to your data, therefore make sure that they are secured and backed up properly.

Note: Further hints and tips for WebSphere Application Server security related considerations are described in IBM WebSphere Developer Technical Journal article *WebSphere Application Server V6 advanced security hardening -- Part 1*, available from the following Web page:

http://www.ibm.com/developerworks/websphere/techjournal//0512_botzum/0512_botzum1.html

After you have finished your security considerations, update your implementation design to include all security related components.

4.2.7 Application deployment

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

- ▶ Performance.

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

Deploying EJB modules to separate application servers, however, gives a lot more flexibility and avoids duplication of code.

- ▶ Number of application servers

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

- ▶ Cluster communication

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

- ▶ Platforms

WebSphere Application Server supports building cross-platform cells. This means you can create one cell containing servers on multiple platforms.

While this option increases the complexity of the environment and makes the administration more complicated, it gives a lot of flexibility when it comes to integration of existing software components.

After you know how your applications will be deployed, update your implementation design.

4.2.8 Servicability

Plan for servicability and problem determination tasks for your environment. Planning for servicability includes planning for tools, education, disk requirements for debug data, communications requirements, required agents for your tools, and so on.

Note: WebSphere Application Server V7.0 comes with IBM Support Assistant V4.0 which provides many tools for problem analysis and data collection. Refer to the following Web pages for more information:

<http://www-01.ibm.com/software/support/isa/>

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/ttrb_supportasst.html

4.3 Sizing the infrastructure

After determining the initial application and infrastructure design and scalability techniques, you need to determine the system resources required for the project to keep the service level agreements (SLAs). Whenever making sizing decisions ensure that all design decisions are still honored.

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

- ▶ Scaling capabilities of the platform
- ▶ Platforms WebSphere Application Server supports
- ▶ Performance, security, and high availability requirements of your environment.

Next, determine your scaling approach:

- ▶ Scaling up—vertical scaling

Scaling up is done inside a single system, typically on a multi-processor machine. Even if this approach might be sufficient from the performance perspective, the underlying hardware is the single points of failure.

- ▶ Scaling out—horizontal scaling

Scaling out means increasing the number of machines. Scaling out is typically done to improve high availability by limiting single points of failure as well as for scalability reasons when hardware resources are the limiting factor. You have to support and maintain multiple machines. If you decide to use the scale out approach keep in mind that your design must be capable of processing the workload even if one system fails.

Sizing estimates are based solely on your input, which means the more accurate 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 is 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 4.4, “Benchmarking” on page 107.

Based on your estimate, you might have to update the implementation design for all of your planned environments. Changes to the production environment should be incorporated into the development and testing environments if at all possible. If this is not possible for budgetary reasons, ensure that the integration and development environment are functionally equivalent.

Make sure to validate the sizing with a load test before starting the production.

4.4 Benchmarking

Benchmarking is the process used to determine the capacity of an application 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.

Many 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 benchmarking. For those that do not, alternatives such as the IBM Test Center are available. There are also third-party benchmark companies that provide this service.

Note: We recommend including a benchmark or load test as part of the development process. This will prevent a lot of performance problems and help improve the code quality of the application, as well as the robustness of the overall environment.

IBM Test Center

IBM Global Services offers Performance Management, Testing, and Scalability services. This team will come on-site and assess the overall site performance. This investigation is platform neutral. Offerings include, but are not limited to the following services:

- ▶ 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 use this data to adjust your current environment.

4.5 Performance tuning

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

When your project is finished and you switch your environment into production, you need to be confident that the environment can handle the user load. Performance problems are by far the most user-visible problem that you can have. Most users are willing to accept small functional problems when a system is rolled out, but performance problems are unacceptable to most users and affect everyone working on the system. Make sure to perform load tests that represent a realistic user load against your system.

4.5.1 Application design issues

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

Note: It is suggested when developing applications to use application profiling techniques. This allows the development team to identify bottlenecks in the applications and hot spots where a lot of CPU resources are consumed. Usually, many hot spots can be removed with little effort.

It takes much more effort and money to correct issues after they 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.

4.5.2 Understand your requirements

Without a clear understanding of your requirements, you have no goal to tune against. It is important when doing performance tuning to have specific objectives as follows:

- ▶ Transactions per second
- ▶ Throughput
- ▶ A maximum time frame, for batch style applications
- ▶ Maximum used resources

Do not waste time performance tuning a system that was improperly sized and cannot withstand the load. To identify a good objective you have to understand the non-functional requirements as well as the sizing of your environment.

4.5.3 Test environment setup

When executing performance tests, follow these general tips:

- ▶ Execute your tests in a production-like environment.

Using an environment that is as close as possible to the production environment enables you to extrapolate the test results to the production environment. If you are starting with a new environment use the future production environment for your testing purposes before going live.

- ▶ Exclusive access to the environment for the test.

Make sure that nobody is using the test systems and that no background processes are running that consume more resources than what you find in production. For example, if the intent is to test performance during the database backup, make sure the backup is running.

- ▶ Usage of monitoring options.

When implementing a WebSphere Application Server environment we suggest using monitoring tools to check the health of the environment. During performance tests there are two level of monitoring to be performed:

- Debug monitoring

The goal is to identify possible bottlenecks or reasons for problems. The debugging level is detailed and will use additional resources, usually affecting the test results.

- Production monitoring

After identifying and solving performance issues with the debug monitoring, perform a test with the same set of monitoring options that are later used in the productive environment. With this setting, you should be able to satisfy the service level agreements.

- ▶ Monitor resource use.
Check for processor, memory, and disk use before, during, and after each test run to see if there are any unusual patterns. If the target environment is using shared infrastructure, 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, though, to flood ports used for routing off the network to other networks or even the switch backbone for heavy traffic. Make sure that your network isolates the testing environment as much as possible prior to starting, because performance degradation of the network can create unexpected results.
- ▶ Perform repetitive tests.
Reset the environment to a defined start state. This includes the database, caches, and so on. Databases that use up datasets need to be reset in particular. Make sure you have enough test data available.

4.5.4 Load factors

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

- ▶ 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.

Request rate

The request rate represents the number of requests per time unit. In Web-based environments this is mostly expressed as number of HTTP requests per second. It is critical that the request rate is close to the real world load expectations.

Concurrent users

The *concurrent users* number expresses the numbers of users concurrently requesting service from your environment at a point in time. This number of users is really firing requests to your system at a given point of time.

In contrast to the concurrent users there are:

- ▶ **Active users**

Active users expresses all users currently using resources (for example, in the form of session data) in your environment. It also includes the users who are reading the response, entering data, and so on.

- ▶ **Named users**

Named users are users that are defined in the overall environment. This is usually a large number compared to the number of concurrent users.

Usage patterns

At this point in the project, it is important that you consider 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 will use.

Usage patterns consist of the following factors:

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

Combining weights with click streams is important because it shows you how many users you expect in 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 vice versa. These assumptions also have an impact on your caching strategy.

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 suggest 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.

4.5.5 Production system tuning

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

Important: To perform tuning in the production environment, 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 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 only have 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 and tuning the WebSphere parameters, evaluate your results and compare them to your objectives to see how all of this worked out for you.

4.5.6 Conclusions

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

- ▶ Performance meets your objectives.

If the performance meets your objectives, make sure that you have planned for future growth and that you are meeting all of your performance goals. After that, we suggest 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 is essential when adding additional replicas of some components in the system, because they need to be configured to the same settings used by the current resources.

- ▶ Performance is slower than required.

If 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 the bottleneck components in your environment. 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 must 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. 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 Assembly and Deploy V7.5 and Rational Application Developer for WebSphere Software V7.5 can help.

4.6 Planning for monitoring

As most WebSphere-based applications are internet-based applications, a 24×7 availability is a must. The tolerance of internet users for unavailable sites is low and users usually navigate to the next site if your site is not operable. This means you just lost a potential customer. It is required then, to track and monitor the availability of your site so that you recognize when things are going wrong and you can react in a timely manner.

An efficient monitoring, combined with a sophisticated alerting and problem handling procedure, can increase the availability of your service significantly. That is why you must plan for monitoring and problem handling. Do not wait until your environment becomes unproductive.

4.6.1 Environment analysis for monitoring

Careful planning for monitoring is essential and must start with a detailed analysis of the environment to be monitored. It is important to ensure that the full environment is monitored and no single component is overseen.

To analyze the monitoring requirements for your environment consider the following factors to give you an overview of what needs to be done:

Components to be monitored

It is essential that each component required to run your service is monitored. For each component you identify, answer the following questions:

- ▶ What are the possible states of the component and how can you retrieve them?
- ▶ What is the impact of each of the possible states the component can have?
- ▶ What specific attributes of the component can be monitored?
- ▶ For each attribute you can monitor, define the following values:
 - Which attribute values (or range of attribute values) show a normal status of the component?
 - Which attribute values (or range of attribute values) show a situation which requires attention of the administrator (warning level)?
 - Which attribute values (or range of attribute values) show a critical condition for the component and require immediate administrator action (alert)?

Prioritize each of the components monitoring results and define actions to be taken.

Monitoring software

To provide efficient 24×7 monitoring, it is required to use some kind of monitoring software. Many organizations have some monitoring infrastructure in place. Determine if you can reuse this for the WebSphere Application Server infrastructure as well.

Monitoring agents

Depending on the monitoring software in use, monitoring agents for certain components might be available. Otherwise, most monitoring software provides some scripting interfaces that allow you to write your own scripts. The scripts will check and output the results that the monitoring software can analyze.

Infrastructure requirements

When running monitoring in your environment you need to plan for additional resources. Monitoring affects your environment in almost all aspects. Monitoring requires memory, CPU cycles, network communications, and might even require separate additional systems for gateways (or as server systems) for the monitoring solution. Ensure that all the nonfunctional requirements for your infrastructure are applied to these systems as well.

Monitoring levels

Monitoring must be in place in all layers of the infrastructure. You must ensure a comprehensive monitoring of the environment. You will likely end up with multiple monitoring tasks and solutions for different purposes.

Network monitoring

Network monitoring covers all networking infrastructure such as switches, firewalls, routers, and so forth. It must also monitor the availability of all the communication paths, including redundant communication paths.

Operating system monitoring

Most monitoring solutions provide monitoring capabilities for supported operating systems. Using these features allows you to keep track of the health of your environment from the operating system perspective and allows you to monitor components like CPU use, memory use, file systems, processes, and so forth.

Middleware components monitoring

When using middleware components like application servers and databases, monitoring on the operating system level is not sufficient, because the operating system has no knowledge of the middleware state. You need specific monitoring to the middleware that provides the runtime environment for your application. WebSphere Application Server provides various interfaces that allow the monitoring of your application server infrastructure. Many popular monitoring

products, like the IBM Tivoli Composite Application Monitoring suite, support these interfaces and provide ready-to-use agents to monitor your WebSphere Application Server environment.

Transaction monitoring

The purpose of transaction monitoring is to monitor the environment from the user perspective. Transaction monitoring uses pre-recorded transactions or click sequences and replays them whereby the response for each replayed user interaction is verified against expected results.

4.6.2 Performance and fault tolerance

Keep in mind that monitoring your environment (no matter at which level) consumes additional resources. Ensure your monitoring setup does not cause unacceptable impact to your environment.

The more you monitor, and the shorter the intervals between your monitoring cycles, the quicker you can determine something out of the ordinary. But the more you monitor, and the shorter the intervals between your monitoring cycles, the higher the overhead you generate. The key to success is to find a good balance between monitoring in sufficient short intervals to determine failures without consuming an unacceptable amount of resources.

In addition to the performance impact, you need to make sure that any problems in your monitoring infrastructure do not impact your environment. Monitoring, even if there is something wrong in the monitoring infrastructure, must never be the cause for any service outage.

4.6.3 Alerting and problem resolution

Monitoring alone is not enough to keep track of the health of your environment, as monitoring does not solve issues. You improve availability if you combine your monitoring with proper alerting to the responsible problem resolvers. What is the use of monitoring if nobody knows that there is a problem? Some thoughts you need to consider when planning for alerting:

- ▶ Who is alerted for which event?
- ▶ What are the required response times?
- ▶ How will the responsible persons be alerted?
- ▶ How will you avoid repeated alerts for the same events?
- ▶ How will alerts and the resolution of the alerts be documented?
- ▶ Who will track the alerts and problem resolution?
- ▶ Who is in charge of the alert until it is finally resolved?
- ▶ Who will perform the root cause analysis to avoid reoccurrences of the alert?

Alerting is just a first part of your incident and problem management. For further information and more details about incident and problem management refer to the ITIL® pages at the following Web page:

<http://www.itlibrary.org/index.php?page=ITIL>

4.6.4 Testing

As with each component in your environment, do not forget to test your monitoring infrastructure regularly. Especially if the implementation is new, test every single monitoring alert and make sure that your monitoring detects each condition of your system properly.

Do not stop your testing when you see a monitoring situation raised. Test the whole process, including alerting and incident management and ensure that conditions are reset automatically as soon as the situation is back to normal.

4.7 Planning for backup and recovery

In general, computer hardware and software is reliable, but sometimes failures can occur and damage a machine, network device, software product, configuration, or more importantly, business data. Do not underestimate the risk of a human error that might lead to damage. It is important to plan for such occurrences. There are a number of stages to creating a backup and recovery plan, which is discussed in the following sections.

4.7.1 Risk analysis

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

4.7.2 Recovery strategy

When critical areas have been identified, develop a strategy for recovering those areas. There are numerous backup and recovery strategies available that vary in recovery time and cost. In most cases, the cost increases as the recovery time decreases. The key to the proper strategy is to find the proper balance between recovery time and cost. The business impact is the determining factor in finding the proper balance. Business-critical processes need quick recovery time to minimize business losses. Therefore, the recovery costs are greater.

4.7.3 Backup plan

With your recovery strategy, a backup plan needs to be created to handle the daily backup operations. 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. 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.

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

Any physical backup must be stored at a remote location to be able to recover from a disaster. New technologies make remote electronic vaulting a viable alternative to physical backups. Many third-party vendors offer this service.

4.7.4 Recovery plan

If a disaster occurs, 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 is 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 the recovery plan must address is 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.

4.7.5 Update and test process

You must revise the backup and recovery plan on a regular basis to ensure that the recovery plan meets your current needs. You also 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.

4.8 Planning for centralized installation

One of the new features introduced in WebSphere Application Server Network Deployment V7.0 is the centralized installation manager (CIM). Using the CIM greatly simplifies the installation and the update of machines in a Network Deployment cell. The administrator of the cell only has to install Network Deployment on one machine and create a deployment manager profile. All CIM related operations are available through either the Integrated Solutions Console or through the `wsadmin` command.

The IBM white paper *Centralized Installation Manager for IBM WebSphere Application Server Network Deployment Version 7.0* provides detailed information about the CIM. This should allow you to plan your environment for centralized information accordingly. The white paper can be found at the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.cim.whitepaper.doc/cimwhitepaper.pdf>

Archived

Topologies

Topology refers to what devices and computers are going to be used to set up a WebSphere Application Server 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 as well as some 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” on page 122
- ▶ “Terminology” on page 134
- ▶ “Topologies in detail” on page 139

5.1 Topology selection criteria

There are many ways to architect a system that satisfies your need. In Chapter 4, “Infrastructure” on page 93, we introduced some planning considerations for a WebSphere Application Server environment. This section provides an overview of the primary considerations in selecting a topology:

- ▶ “High availability” on page 122
- ▶ “Disaster recovery” on page 125
- ▶ “Security” on page 126
- ▶ “Maintainability” on page 126
- ▶ “Performance” on page 127
- ▶ “Application deployment” on page 129

5.1.1 High availability

High availability means that a system can tolerate a certain amount of failure and errors, before it is no longer working properly. High availability is an important consideration to take into account when designing your architecture. The importance comes from the fact that your design for high availability is a key criteria for the capability of your service to meet your expected service level agreement (SLA). High availability is a complex topic that affects every single component required to run your environment. High availability is not only a design challenge for WebSphere but for every component on every layer of the environment.

High availability is achieved by introducing redundancy in your architecture to be fault tolerant. This means you need redundancy at different levels depending on your availability requirements (for example: power supplies, network cables, switches, processes, machines and so forth). WebSphere Application Server Network Deployment provides a vast variety of options to provide a highly available runtime environment for your applications. In the subsequent sections we discuss many of WebSphere Application Server’s high availability features and how you can benefit from them.

Avoid single points of failure

To avoid a single point of failure and to maximize system availability, the topology needs to have some degree of redundancy. WebSphere Application Server high availability topologies typically involve horizontal scaling across multiple machines. Vertical scaling can improve availability by creating multiple processes, but the machine itself still remains a single point of failure.

When introducing redundancy, the need for a load distributor or front-end traffic handler arises. An IP sprayer can direct Web client requests to the next available Web server, bypassing any server that is not available. The IP sprayer server can be configured with a cascade backup (hot standby) that takes over the operation in case the primary server fails. This configuration eliminates the IP sprayer as a single point of failure in the architecture.

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.

Hardware redundancy

Eliminate single points of failure in a system by including hardware and redundancy. Hardware redundancy can be invented at different levels. In the topology considerations we are considering hardware redundancy at the system level only.

You can use the following techniques:

- ▶ Use horizontal scaling to distribute application servers (and applications) across multiple physical machines or z/OS images. If a hardware or process failure occurs, clustered application servers 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 secured rooms to prevent site-related problems.

Process redundancy

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

- ▶ Deploy the Web server on a different machine than the application servers. This configuration ensures that problems with the application servers do not affect the Web server and vice versa. Separate systems also increase the security level.
- ▶ Use horizontal scaling, placing application servers on different systems.
- ▶ The usage of vertical scaling provides process isolation as related to WebSphere processes only.

Load balancing

Use load balancing techniques to make sure that individual servers are not overwhelmed with client requests while other servers are idle. This includes the following techniques:

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

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

WebSphere Application Server provides these other load balancing mechanisms:

- ▶ The WebSphere Application Server HTTP server plug-in to spread requests across cluster members.
- ▶ The Enterprise JavaBeans (EJBs) workload management mechanism, which is built into WebSphere Application Server to balance EJB workload across cluster members
- ▶ The usage of partitioned queues. If the application permits, you can configure partitioned queues to split message processing workload across multiple servers.

Fail-over support

The environment must be able to continue processing client requests, even if one or more components are offline. Some ways to provide fail-over support are as follows:

- ▶ Use horizontal scaling with workload management to take advantage of failover support.
- ▶ Use an IP sprayer to distribute requests across the Web servers in a configuration.
- ▶ Use HTTP server plug-in support to distribute client requests among application servers.
- ▶ Use EJB workload management to re-align EJB requests if an application servers goes down.
- ▶ Use WebSphere high availability manager to provide fail-over support of critical services, the singletons, like messaging engines, transaction service, and so forth.

Operating system based clustering

The WebSphere Application Server high availability framework provides integration into an environment that uses other high availability frameworks, such as operating system-based clustering software like HACMP™ on AIX, or parallel sysplex on z/OS, to provide high availability for resources for which WebSphere does not provide specific high availability functions. Consider such a technique for WebSphere Application Server components such as a deployment manager, node agents, single server environments, and so on.

5.1.2 Disaster recovery

The main considerations in disaster recovery planning are how to bring up a fully operative environment after a disaster struck the system, how much data loss you can afford, and how can ensure that your data remains consistent.

Cluster isolation¹, for example, is a potential threat to your data consistency and must be avoided in all circumstances. After the data consistency issues are resolved (meaning that your problems regarding your WebSphere data, such as configuration repository, logs and so forth are also resolved) you can resume your planning for WebSphere disaster recovery.

Note: A design approach to spread your WebSphere Application Server cell over multiple data centers is not a sufficient response to a disaster recovery design challenge. Refer to the developerWorks article *Everything you always wanted to know about WebSphere Application Server but were afraid to ask -- Part 2*, available at the following Web site:

http://www-128.ibm.com/developerworks/websphere/techjournal/0512_col_alcott/0512_col_alcott.html

There is no common solution for WebSphere Application Server in a disaster recovery scenario as this depends on the existing environment, requirements, applications and budget. For some considerations on disaster recovery in your WebSphere environment, refer to the Developer works article *Everything you always wanted to know about WebSphere Application Server but were afraid to ask, Part 5*, available at the following Web page:

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

¹ Cluster isolation in this context means a condition that each member of a clustered system considers the other to be gone and therefore taking over the service. The result would be to have two systems manipulating data.

5.1.3 Security

Security is a critical consideration 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

Physical security means protection against physical threats such as controlling physical access to the systems, but also includes tasks to protect the environment of the systems.

- ▶ Logical security.

Logical security is connected to a specific IT solution, architecture, and application design. It deals with all aspects of access to runtime resources and data.

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 and finalizing a topology, review the information in 4.2.6, “Security” on page 102 and Chapter 12, “Security” on page 379.

5.1.4 Maintainability

Maintainability means that the complexity of the system must be kept in a manageable state. When designing a WebSphere Application Server infrastructure, many different non-functional requirements have to be considered. This can increase the complexity of the environment to an extent that it becomes difficult and error-prone to manage.

An easily maintained environment allows maintenance work to be done on the fly and minimizes planned system outages for maintenance work. This includes the ability to upgrade application releases while the system is running, as well as to run different versions of servers, operating systems, and applications in the same administrative unit. Not only does the infrastructure contribute to maintainability, but the application as well. You will need an application development policy that assures a sufficient number of application releases can work together. Especially in large environments, it is often impossible to upgrade all instances of an application at the same time.

5.1.5 Performance

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

When talking about performance there are two widely used approaches:

- ▶ Response time

Response time is a generic approach for a single type of request. It defines the maximum amount of time a request should take until it is finished. This metric is most often used in online workload, where a request has to achieve a real-time goal.

Note: When using this metric make sure that the response time is not only achieved in a single user-single transaction scenario, but also when the projected production load is used against the system.

- ▶ Throughput

The throughput metric measures the overall amount of work processed in a certain amount of time. It is usually used for batch-kind workloads that need to be finished in a certain time window.

WebSphere Application Server Network Deployment enables you to cluster application servers so that you have multiple server instances running the same application available to handle incoming requests. Clustering generally provides improvements for performance, due to an optimized scaling.

Scaling

Scaling represents the ability of the system to grow as the load on the system grows. There are multiple ways to achieve scaling. For example, multiple machines can be configured to add processing power, improve security, maximize availability, and balance workloads. Scaling may be vertical or horizontal (Figure 5-1). Which you use depends on where and how the scaling is taking place.

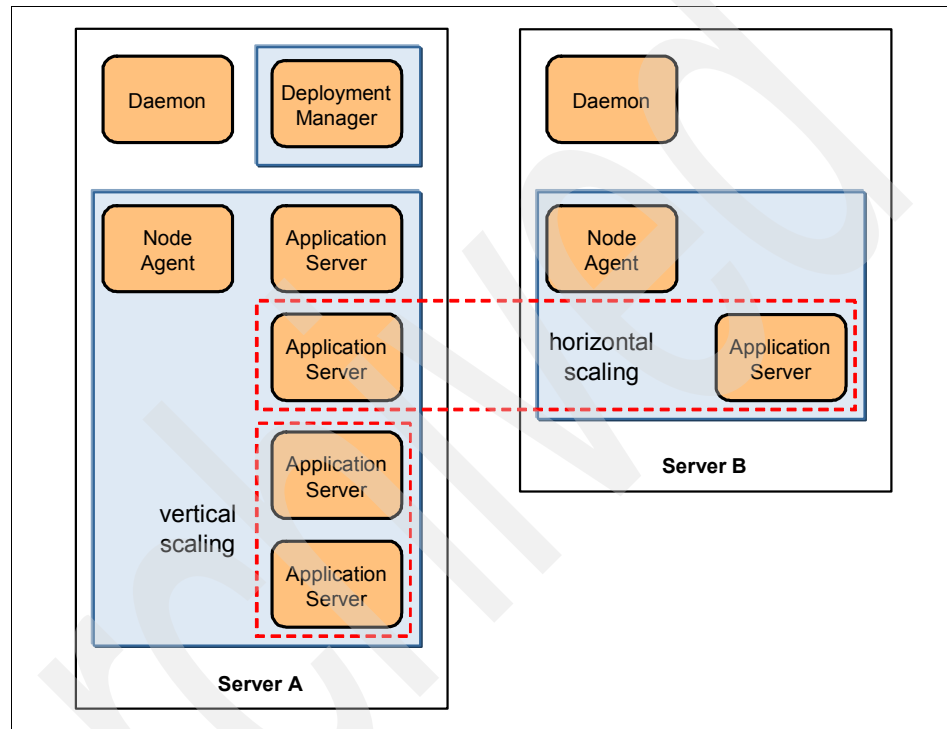


Figure 5-1 Vertical and horizontal scaling with WebSphere Application Server

► Vertical scaling

Vertical scaling involves creating additional application server processes on a single physical machine or z/OS image, providing application server failover as well as load balancing across multiple application servers. This topology does not provide an 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

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

The following components provide functions for configuring scalability:

- ▶ 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 Application Server can be used to distribute requests among converged 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 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 and other TCP/IP-based servers whose protocol is understood by the IP sprayer.

5.1.6 Application deployment

Various application deployment decisions affect topology decisions. The following application deployment-related considerations should be clarified before finalizing the architecture.

Deployment of your EJBs

The way you deploy your EJBs will have a significant impact on your application topology and of the performance you can expect. You have the choice to deploy EJBs to the same application servers and clusters as the client modules invoking the EJBs, or to separate, dedicated application servers and clusters running the EJBs only.

Both of these options are valid approaches depending on your environment and requirements. For this discussion it is assumed that the EJBs provide local and remote interfaces to be invoked.

To determine what is the best for your environment, consider the advantages and disadvantages of deploying the EJBs to a different EJB servers and clusters as the EJB clients application server:

► Advantages

- More consistency in the business functions, as all your enterprise applications have to use the same version of the EJBs.
- Better re-use of EJB code, as the same code is used by all enterprise applications requiring the same functionality.
- Application servers running the EJBs can be tuned individually.
- You can benefit from runtime provisioning as the EJB container only gets loaded on the EJB servers, while the converged container only gets loaded on the application servers running your WAR files.
- If you are running WebSphere Application System Client applications, the client calls do not use the Java resources of your Web applications Java virtual machine (JVM).
- You can run the EJBs on different systems or even different platforms from your Web modules
- Easier deployment of the applications and EJBs

► Disadvantages

- Every EJB call is an out-of-process call. You will pay a performance penalty for this configuration as this introduces a lot of serialization and de-serialization overhead and even network traffic with encryption if the EJBs are deployed to a different system.
- Re-authentication during EJB calls might be required.
- All the enterprise applications must be compatible with the existing version of deployed EJBs. This requires coordination of the application development.
- More application server processes are running, which increases the memory footprint of the deployment.
- The number of application servers in the cell will increase, resulting in a more complex administration.
- Upgrades of the EJBs are more complex, as multiple applications are affected.

Assignment of applications to clusters

When you are running multiple applications in your environment you must decide if you want to deploy all your applications to the same application servers and clusters or if you want to set up separate application servers and clusters for each application.

To determine what is best for your environment, consider the advantages and disadvantages of deploying all applications to separate application servers and clusters (the one application to a server approach):

► Advantages

- You will get better process and application isolation and a faulty application will not affect other applications (unless it is a common component).
- The administration and maintenance work for application and cluster-related tasks is easier.
- Easier tuning of the application servers. If you are running each application separated from the others, you can tune the application servers specifically for the need of this application. Often applications are different in their tuning requirements so that it is almost impossible to tune an application server properly when the applications are deployed to the same application server and cluster.
- As fewer applications are running in the application server, the Java heap requirements for the application will likely be less. You can configure a smaller heap, which results in faster garbage collection cycles and better and more consistent response times.

Note: The heap size of a JVM is finite. Even when using a 64-bit implementation of WebSphere Application Server, you must be careful with the heap sizing to avoid performance problems during garbage collection.

- You might benefit from the runtime provisioning capabilities of WebSphere Application Server V7.0.

► Disadvantages

- You will end up with more application servers and clusters, which means larger cells and the administration of your cell will become more complex.
- As you will have more application servers in your environment, you will have larger cells. Larger high availability manager cells will result in increased startup times

- You will more likely have to call your EJBs out-of-process and pay a performance penalty for that.
- The memory footprint of the environment will increase as each of the JVMs has a basic memory footprint.

Location of the embedded messaging infrastructure

When using the WebSphere Application Server embedded messaging infrastructure, you must decide in which application servers the messaging service should run. Whether you run the embedded messaging service on a separate application server and cluster or co-located on an application server running your applications depends on your needs.

To determine what is best for your environment, consider the advantages and disadvantages of running the embedded messaging infrastructure on a separate set of application servers and clusters:

► Advantages

- The messaging service runs on a separate JVM and will not use Java resources of your application server. If multiple applications and JVMs are using the messaging service this might be more important.
- It is possible to tune each application server for its specific usage. This includes Java heap tuning as well as Java runtime tuning.
- Application servers and clusters can be started, stopped, and restarted independently from the availability of the messaging service without causing a failover of the messaging service (and vice versa).
- You can deploy your messaging infrastructure to systems independently from the other application servers and clusters. This allows you to deploy your messaging infrastructure to, for example, systems where message persistence can be implemented easiest.
- You will benefit from the runtime provisioning capabilities of WebSphere Application Server V7.0.

► Disadvantages

- The number of application servers in your cell increases, resulting in a more complex administration of the cell.
- Due to the higher number of application servers, your cell and your core group grows. You will see longer cell startup times and more overhead caused by high availability manager.
- Memory footprint of the environment increases due to additional servers.
- The deployment of your applications might become more complex, especially when using mediation modules.

5.1.7 Summary: Topology selection criteria

Table 5-1, Table 5-2, and Table 5-3 on page 134 list some requirements for topology selection, and possible solutions.

Table 5-1 Topology selection based on availability requirements

Requirement = availability	Solution/topology
Web server	Load Balancer (with hot backup) or a comparable high availability solution, based on other products.
Application server	<ul style="list-style-type: none"> ▶ Horizontal scaling (process and hardware redundancy) ▶ Vertical scaling (process redundancy) ▶ A combination of both ▶ Multi servant regions on z/OS
Database server	Database or operating system-based high availability solution
User registry	Depends on the user registry in use. WebSphere provides backup support for some user registries

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 Servers in conjunction with Load Balancer ▶ 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) ▶ Deploy EJBs to the same JVM as the invoking client ▶ Dynamic caching at the application server. ▶ Offload static content to be served from the Web server and therefore offload the application servers^a ▶ Workload management and transaction classes on z/OS to keep response times.
Database server	<ul style="list-style-type: none"> ▶ Separate database server ▶ Use partitioned database servers

a. For details, see the following Web page:

<http://www.redbooks.ibm.com/abstracts/TIPS0223.html>

Table 5-3 Topology selection based on security requirements

Requirement = security	Solution/topology
Web server	<ul style="list-style-type: none"> ▶ Separate the Web server into a DMZ, either on LPAR or separate system. ▶ Use a DMZ secure proxy instead of Web server with WebSphere plug-in. ▶ Separate administrative traffic from productive traffic
Application server	<ul style="list-style-type: none"> ▶ Implement WebSphere Application Server security. ▶ Create a separate network tier for the application server. ▶ Separate the application servers from the database and EIS layer ▶ Separate administrative traffic from productive traffic

5.2 Terminology

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

5.2.1 Load balancers

A load balancer, also referred to as an IP sprayer, enables horizontal scalability by dispatching TCP/IP traffic among several identically configured servers. Depending on the product used for load balancing, different protocols are supported.

In our topologies, the load balancer is implemented using the Load Balancer Edge component provided with the Network Deployment package, which provides load balancing capabilities for HTTP, FTP, SSL, SMTP, NNTP, IMAP, POP3, Telnet, SIP, and any other TCP based application.

Note: On the z/OS platform, the function that provides intelligent load balancing is the Sysplex Distributor. It balances incoming requests based on real-time information about whether the possible members achieve their performance goals. The member with the best performance rating processes the incoming request.

5.2.2 Reverse proxies

The purpose of a reverse proxy is to intercept client requests, retrieve the requested information from the content servers, and to deliver the content back to the client. Caching proxies provide an additional layer of security hiding your servers from the clients. The caching proxy products provided by WebSphere Application Server V7.0 (namely, the (deprecated) Edge Components Caching Proxy, the DMZ secure proxy, and the WebSphere proxy server) provide the capability to store cacheable content in a local cache. Subsequent requests for the same content can be served out of this cache. This allows faster response and decreases the load on the servers as well as the internal network.

Edge proxy

The caching proxy as provided with WebSphere Application Server V7.0 with the Edge Components can be configured as a reverse and as a forwarding proxy. In this book, it is considered as a reverse proxy only. This proxy server supports the following protocols: HTTP, HTTPS, FTP and Gopher.

Note: The caching proxy (provided with WebSphere Application Server V7.0 as part of the Edge Components) was declared deprecated in WebSphere Application Server V6.1. For a current list of deprecated, stabilized, and removed features in WebSphere Application Server V7.0, see the WebSphere Information Center article *Deprecated, stabilized, and removed features* at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rmig_deprecationlist.html

DMZ secure proxy

The DMZ secure proxy server is a new feature of WebSphere Application Server V7.0. Using it allows you to install proxy servers in the demilitarized zone (DMZ) at a reduced security risk, compared to installing an application server to host a proxy server. This is achieved by removing all the features from the application server that are not required to provide the proxy functionality. For example, there is no Web container or EJB container in a DMZ secure proxy.

The DMZ secure proxy server supports the following protocols with and without encryption: HTTP and SIP. To implement a DMZ secure proxy you need to install the DMZ secure proxy product and create a profile using the secureproxy profile template.

Note: Do not mix up the DMZ secure proxy with the WebSphere Application Server Proxy that you can configure in a network deployment manager cell.

WebSphere Application Server Proxy

The WebSphere Application Server Proxy is a proxy server you configure in a WebSphere Application Server Network Deployment cell. This proxy runs inside the secure zone of the network as an application server and has access to cell information and the current state of all servers and applications inside the cell.

5.2.3 Domain and protocol firewall

A firewall is a hardware and software system that manages the flow of information between networking zones like the Internet and an organization's private network. Firewalls can prevent unauthorized Internet users from accessing services on private networks that are connected to the Internet, especially intranets. In addition, firewalls can block some virus attacks, if those viruses attacks have to cross the network boundaries protected by the firewall. Another typical usage of firewalls is to prevent denial of service attacks against services.

A firewall can separate two or more parts of a local network to control data exchange between departments, network zones, and security domains. Components of firewalls include filters or screens, each of which controls transmission of certain classes of traffic. Firewalls provide the first line of defense for protecting private information. Comprehensive security systems combine firewalls with encryption and other complementary services, such as content filtering and intrusion detection.

Firewalls control access from a less trusted network to a more trusted network. Traditional firewall services include the following implementations:

- ▶ Screening routers (the protocol firewall)
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 a 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.

5.2.4 Web servers and WebSphere Application Server Plug-in

Most WebSphere Application Server topologies will have a Web server which receives HTTP-based requests from clients. For security reasons the Web server should be placed in a separate network zone secured by firewalls (a DMZ).

Usually the Web server, in conjunction with the WebSphere Application Server Plug-in, provides the following functionality in the topology:

- ▶ Serves requests for static HTTP content like HTML files, images, and so forth.
- ▶ Requests for dynamic content like Java Server Pages (JSPs), servlets, and portlets are forwarded to the appropriate WebSphere Application Server through the WebSphere Application Server Plug-in.
- ▶ Allows caching of response fragments using the Edge Side Include (ESI) cache.
- ▶ Breaks the secured socket layer (SSL) connection from the client (unless this is done by another device in the architecture) and optionally opens a separate secured connection from the Web server to the Web container on the application server system.

WebSphere Application Server comes with Web server plug-ins for all supported Web servers.

Note: For information about Web servers supported by WebSphere Application Server V7.0, organized by platform, refer to the following Web page:

<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27012369>

The plug-in uses a configuration file (plugin-cfg.xml) that contains settings describing how to pass requests to the application server. The configuration file is generated on the application server. Each time a change on the application server affects the request routing of requests (for example, a new application is installed) the plug-in must be regenerated and propagated to the Web server machine again.

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, by using the administrative instance of IBM HTTP Server.

WebSphere Application Server V7.0 ships with IBM HTTP Server V7.0 on distributed platforms and on z/OS, which is based on Apache 2.2.8 plus its additional fixes. For more details about what is new in IBM HTTP Server V7.0 see the Infocenter article *What is new in this release*, which can be found at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.ihs.doc/info/ihs/ihs/cihs_newfunction.html

5.2.5 Application servers

Application servers are the heart of your topology. This layer in the architecture provides the runtime environment for your Java Platform, Enterprise Edition (Java EE) applications.

To provide all the flexibility and functionality offered by WebSphere Application Server, various profiles types are available. Some of the profiles types are for management purposes only. Others are required to process user requests at runtime. The management-related components of the runtime environment are implemented through specific application servers with predefined names. These application servers are created for you when creating certain profiles. Your topology has to consider which of these management servers are needed and where they are placed.

5.2.6 Directory and security services

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

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

5.2.7 Messaging infrastructure

WebSphere Application Server can either connect to and use an existing messaging infrastructure, or it can provide its own infrastructure for messaging through embedded messaging. The messaging service of the embedded messaging provider in WebSphere can run in any user-created application server.

For more information, refer to the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/welc6tech_msg.html

5.2.8 Data layer

The data layer in the topology refers to a variety of back-end resources holding real business data and logic for the enterprise. The enterprise applications running on WebSphere Application Server access these resources to build responses for the users and to update data based on user input. This can be a database, an enterprise information system (EIS), a transaction monitor such as CICS, a Web service, and so forth.

5.3 Topologies in detail

Due to the vast amount of configuration possibilities, WebSphere Application Server provides many different configuration options to fit almost every requirement. In this section we discuss some basic configuration topologies (which can also be combined), depending on the requirements of your environment.

Notes:

- ▶ WebSphere Application Server V7.0 allows you to create profiles either graphically using the Profile Management Tool (pmt) or through the **manageprofiles** command. As for traceability reasons, the command-based creation is preferred. The samples in this chapter are based on the **manageprofiles** command.
- ▶ On the z/OS platform, all the topologies introduced in this section profit from the workload management capabilities offered from the Workload Manager component and the WebSphere Application Server for z/OS. This management allows you to set and keep performance-focused SLAs on a transactional level.

For more information about the workload management capabilities for the WebSphere Application Server for z/OS refer to 14.1.6, “Workload management for WebSphere Application Server for z/OS” on page 428.

5.3.1 Stand-alone server topology

The topologies in this section all use a Web server as a front-end device. This has the benefit of not deploying an application server in the DMZ as well as using the Web server for caching purposes.

Application server

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 only. This topology does not provide any load balancing or high availability capability at all. You should be aware of the risks when implementing such a topology.

Note: A stand-alone application server on the z/OS platform offers some degree of load balancing and high availability for the application itself. WebSphere Application Server for z/OS uses multi-servant regions, best thought of as a special application cluster to build each application server. This provides one application image to the user while running multiple, independent instances of the application.

The use of multiple application images or not can be configured by the system administrator. For more information refer to 14.1.4, “Structure of an application server” on page 422.

Web server

Although you can install the Web server on the same system as WebSphere Application Server, and you can even direct HTTP requests directly to the application server, you should have a Web server in a DMZ 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 a separate network.

Figure 5-2 illustrates such a topology.

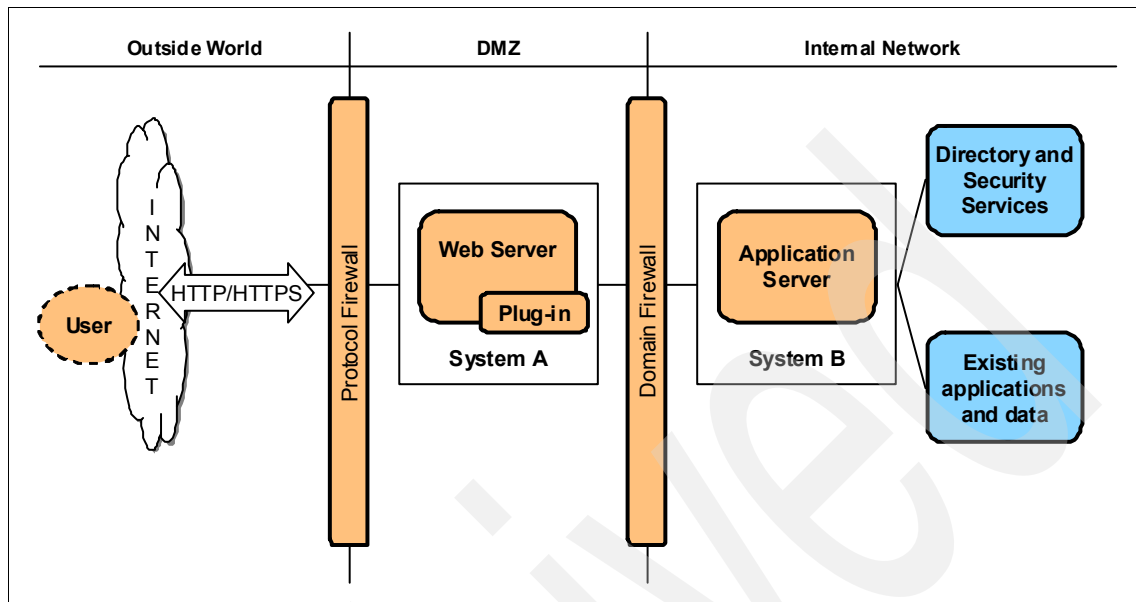


Figure 5-2 Stand-alone Server topology with Web server in a DMZ

Advantages

The advantages to a stand-alone server topology are as follows:

- ▶ Allows sizing and configuration of servers appropriately to each task
By installing components (Web server and application server) on separate systems or z/OS images, each can be sized and configured to optimize the performance of the various component.
- ▶ Removes 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 does not affect its ability to service dynamic requests. The same applies when the Web server serves dynamic content using other technologies, such as common gateway interface (CGI).
- ▶ Increases maintainability due to component independence
Server components can be reconfigured, or replaced, without affecting the installation of other components, because they are on separate machines or LPARs.

- ▶ Increased security by using a DMZ

Isolating the Web server in a DMZ protects the business applications and data in the internal network by restricting access from the public Web site to the servers and databases where this information is stored. We suggest avoiding topologies in which servers in the DMZ have direct access to the database storing business or other security sensitive data.

Disadvantages

The disadvantages to a stand-alone server topology are as follows:

- ▶ Requires additional deployment

The plug-in configuration file is generated on the WebSphere Application Server machine and must be copied to the Web server machine each time a configuration change occurs, which affects requests for applications. Although WebSphere Application Server V7.0 provides tools to automate this step, not every environment is suitable to use these tools.

- ▶ Possible drop of performance

Depending on the network capacity and the distance of the Web server, the network response time for communications between the application server and Web server can limit the application response time. To prevent this, ensure that you have an adequate network bandwidth between the Web server and the application server.

- ▶ Additional security overhead due to SSL communication

When using SSL communication from the client to the Web server, the communication from the plug-in to the application server must be encrypted to avoid sensitive data being “sniffed” on the network. This additional encryption introduces a performance penalty and increased resource use. From a security perspective, we suggest configuring the connection from the plug-in to the Web container, so that the plug-in and Web container must mutually authenticate each other using certificates. This prevents unauthorized access to the Web container.

- ▶ Additional systems to administer

With the Web server running on a separate system, there is one more system to manage and to operate, which increases the operation cost of the environment.

Installation and configuration

To set up an environment as illustrated in Figure 5-2 on page 141 the following installation and configuration steps are required:

System A

Perform the following steps to install and configure System A.

1. Install and configure a supported Web server.
2. Install and configure the WebSphere Application Server Plug-in for the Web server.

System B

Perform the following steps to install and configure System B.

1. Install WebSphere Application Server V7.0
2. Create an application server profile using the following profile template:
`<app_server_root>/profileTemplates/default`
3. Create a Web server definition through the Integrated Solutions Console or through the wsadmin scripting interface.

Note: The Web server definition is used by the application server to generate the plug-in configuration file. In a stand-alone topology, only one Web server can be defined to the configuration and it must be an unmanaged Web server.

5.3.2 Vertical scaling topology

Vertical scaling (depicted in Figure 5-3 on page 144) refers to configuring multiple application servers on a single machine or LPAR and creating a cluster of associated application servers all hosting the same applications. All members of the application server appear as one logical unit serving the applications deployed to the cluster.

Keep in mind that a WebSphere Application Server cluster can only be implemented with the Network Deployment or the z/OS packages.

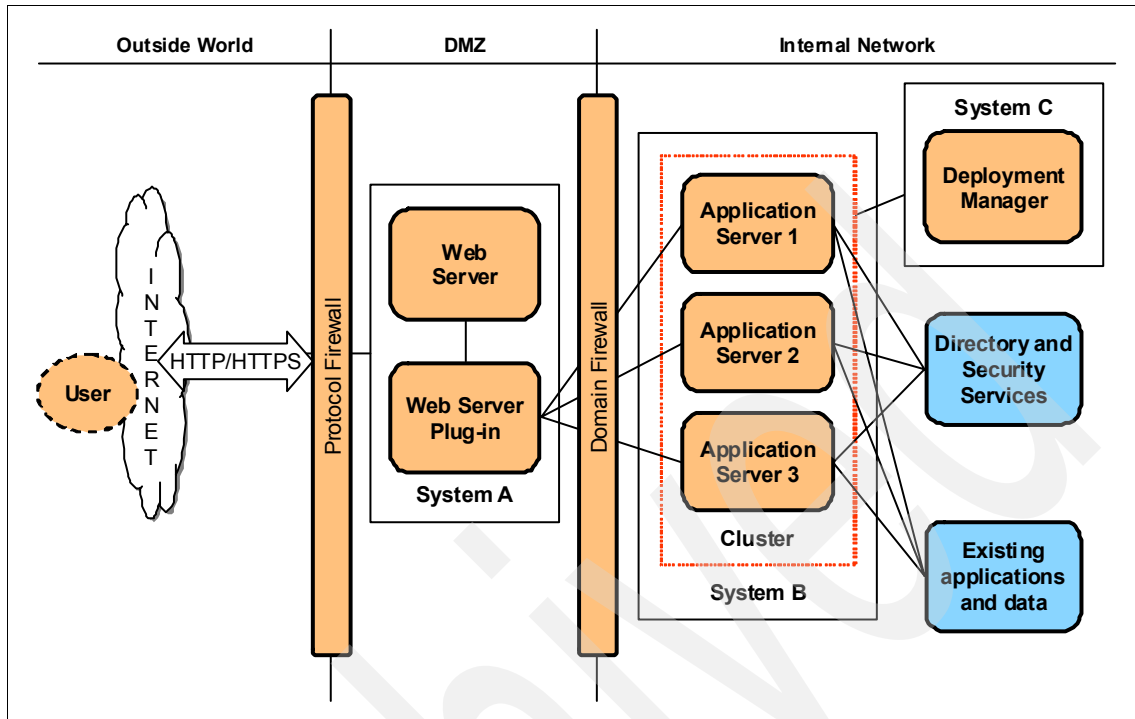


Figure 5-3 Vertical scaling topology with WebSphere Application Server

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

Note: The illustration in Figure 5-3 is intended to show a vertical scaling topology of application servers but still contains several single points of failures.

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

Advantages

Implementing vertical scaling in your topology provides the following advantages:

- ▶ Optimized resource use
With vertical scaling, each application server running its own JVM uses a portion of the machine's processor and memory. The number of application servers on a system can be increased or decreased to optimize the resource use of the machine.
- ▶ Growth beyond the limits of a single JVM
A vertical scaling implementation allows you to grow with your implementation beyond the limits of a single JVM as you can run multiple JVMs in parallel.
- ▶ Benefits from WebSphere Application Server workload management capabilities
As vertical scaling is implemented through clusters, it allows you to benefit from WebSphere Application Server workload management.
- ▶ Failover support
Due to the fact that vertical scaling is implemented using clusters, vertical scaling topologies can also take advantage of the failover support provided by WebSphere Application Server. If one of the application server processes is stopped, the remaining cluster members will continue to process and realign the workload.

Disadvantages

If you are using vertical scaling, there are some limitations and possible drawbacks to consider:

- ▶ Single points of failure
Unless you combine the vertical scaling architecture with horizontal scaling, you still have single points of failure (like hardware, operating system processes and so on) in your architecture .
- ▶ Additional overhead
To implement vertical scaling you need WebSphere Application Server Network Deployment. You need additional application server processes like the deployment manager and the node agent process to manage such an environment.
- ▶ Additional planning and implementation work required
To benefit from the load balancing and failover capabilities, you need to plan for these scenarios. For example, to benefit from failover mechanism, you need to think about what is required for a successful failover (like session data and so forth) and size for all possible situations carefully.

Installation and configuration

The following list indicates the minimum software configuration that you need for the topology shown in Figure 5-3 on page 144:

System A

Perform the following steps to configure System A:

1. Install and configure a supported Web server.
2. Install and configure the WebSphere Application Server Plug-in for the Web server.

System B

Perform the following steps to configure System B:

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the `<app_server_root>/profileTemplates/managed` profile template. Federate this profile to the deployment manager on server C either during profile creation or by running the **addNode** command after the profile creation.

Alternatively you can create an application server profile using the `<app_server_root>/profileTemplates/default` profile template and federate the node to the deployment manager running on System C by using the **addNode** command.

System C

Perform the following steps to configure System C:

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the profile template `<app_server_root>/profileTemplates/dmgr` template or use the `<app_server_root>/profileTemplates/management` template and specify `-serverType` as `DEPLOYMENT_MANAGER`.
3. Create a Web server definition through the Integrated Solutions Console or through the `wsadmin` scripting interface.
4. Create a WebSphere Application Server cluster with three cluster members on System B.

5.3.3 Horizontal scaling topology

Horizontal scaling means to create one logical unit of servers across multiple systems or LPARs where each member of the unit is able to serve each request. Horizontal scaling at the application server tier does not require an IP sprayer. If you want to scale at the Web server tier as well, we suggest that you implement an IP sprayer.

We first introduce a topology without an IP sprayer and then one with the IP sprayer component (see “Horizontal scaling topology with an IP sprayer” on page 150).

Horizontal scaling topology without an IP sprayer

The topology shown in Figure 5-4 lets a single application span multiple machines, while presenting itself as a single logical image. In this example, the WebSphere Application Server cluster spans System B and System C, each with one application server. The deployment manager is installed on a separate server, System D.

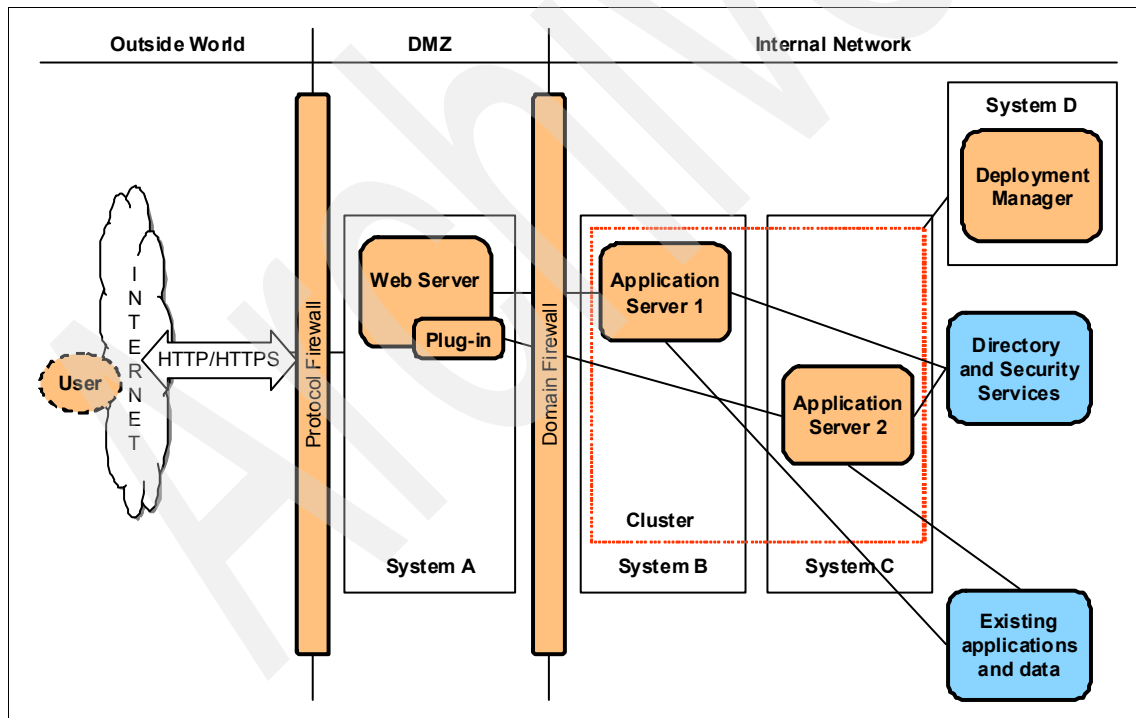


Figure 5-4 Horizontal scaling with cluster

The Web server plug-in distributes requests to the cluster members on each server performing load balancing and offers an initial failover just as it does in the vertical clustering topology. If any component (hardware or software) on System B fails, the application server on System C can continue to serve requests, and vice versa.

Note: The illustration in Figure 5-4 is intended to show a horizontal scaling topology of application servers but still contains single points of failure (namely, the Web server). To avoid these single points of failures you have to enhance the topology as shown in “Horizontal scaling topology with an IP sprayer” on page 150.

Advantages

Horizontal scaling using clusters provides the following advantages:

- ▶ Improved throughput
As multiple systems are servicing your client requests simultaneously without competing for resources, you can expect improved throughput from your installation.
- ▶ Improved response times
Hosting cluster members on multiple machines enables each cluster member to make use of its machine's processing resources, avoiding bottlenecks and resource contention. Therefore, response times will improve in most scenarios.
- ▶ Benefits from WebSphere Application Server workload management capabilities
As horizontal scaling is implemented through clusters, it benefits from WebSphere Application Server workload management.
- ▶ Provides enhanced failover support
As the cluster members are spread over multiple systems, this topology provides hardware failover capabilities. Client requests can be redirected to cluster members on other machines if a machine goes offline. The outage of a system or an operating system failure does not stop your service from working.

Disadvantages

Horizontal scaling using clusters has the following disadvantages:

- ▶ Increased resources used

As multiple systems are required to implement this topology, the hardware cost will increase.

To implement vertical scaling you need WebSphere Application Server Network Deployment. Therefore, you need additional application server processes like the deployment manager and the node agent process to manage such an environment. This increases overhead and the memory footprint of the installation.

- ▶ More complex administration

The maintenance and administration of the environment are more complex as the number of systems increases.

Installation and configuration

The following sections indicate the minimum software configuration steps that you need to perform for the topology shown in Figure 5-4 on page 147:

System A

Perform the following steps to configure System A.

1. Install and configure a supported Web server.
2. Install and configure the WebSphere Application Server Plug-in for the Web server.

System B and System C

Perform the following steps to configure System B and System C.

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the `<app_server_root>/profileTemplates/managed` profile template and federate this profile to the deployment manager on system D either during profile creation or by running the **addNode** command after the profile creation.

Alternatively, you can create an application server profile using the `<app_server_root>/profileTemplates/default` profile template and federate the node to the deployment manager running on System D by using the **addNode** command.

System D

Perform the following steps to configure System D.

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the profile template `<app_server_root>/profileTemplates/dmgr` template, or use the `<app_server_root>/profileTemplates/management` template and specify `-serverType` as `DEPLOYMENT_MANAGER`
3. Create a Web server definition through the Integrated Solutions Console or through the `wsadmin` scripting interface.
4. Create a WebSphere Application Server cluster with one cluster member on System B and one cluster member on System C.

Horizontal scaling topology with an IP sprayer

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

Figure 5-5 on page 151 illustrates a horizontal scaling configuration that uses an IP sprayer to redistribute requests between Web servers on multiple machines.

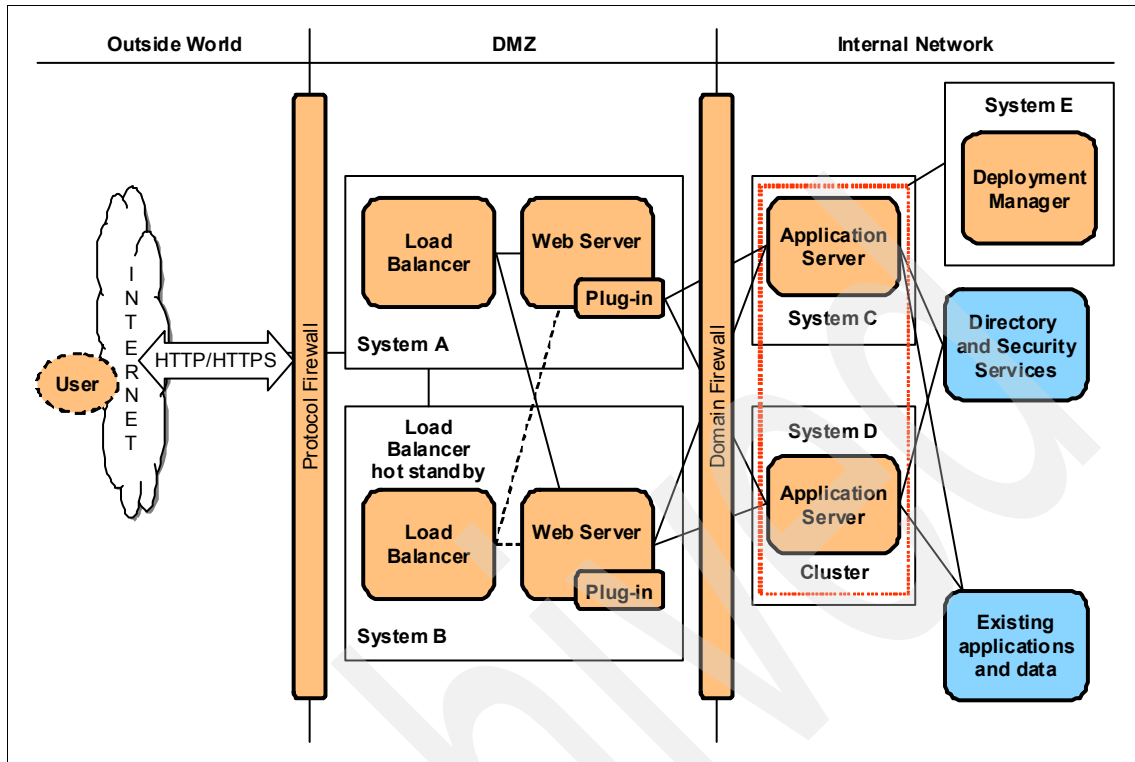


Figure 5-5 Simple IP sprayer horizontally scaled topology

Note: To reduce the number of systems, and to demonstrate the co-location capabilities of the Edge components Load Balancer, in Figure 5-5 the Load Balancer and the Web server are installed on the same system. The Web servers can be installed on separate systems in the DMZ as well. Consult the Edge Components, Version 7.0 Information Center for supported network configuration. See the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.edge.doc/welcome.html>

The active Load Balancer hosts the highly available TCP/IP address, the cluster address of your service and sprays requests to the Web servers. At the same time, the Load Balancer keeps track of the Web servers health and routes requests around Web servers that are not available. To avoid having the Load Balancer be a single point of failure, the Load Balancer is set up in a hot-standby cluster. The primary Load Balancer communicates its state and routing table to the secondary Load Balancer. The secondary Load Balancer monitors the

primary Load Balancer through heartbeat and takes over when it detects a problem with the primary Load Balancer. Only one Load Balancer is active at a time.

Both Web servers are active at the same time and perform load balancing and failover between the application servers in the cluster through the Web server plug-in. If any component on System C or System D fails, this should be detected by the plug-in and the other server can continue to receive requests.

Advantages

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

- ▶ Improved performance
By distributing incoming TCP/IP requests (in this case, HTTP requests) among a group of Web servers, the workload is spread and you can expect improved performance.
- ▶ Increased capacity
The usage of multiple Web servers increases the number of connected users that can be served at the same time.
- ▶ Elimination of the Web server as a single point of failure.
Used in combination with load balancers this topology eliminates the Web server as a single point of failure.
- ▶ Improved throughput and performance
By maximizing parallel processor and memory usage you can expect increased throughput and performance.

Disadvantages

The usage of Load Balancer has some drawbacks most of which are cost related, namely:

- ▶ Increased complexity
This configuration requires the Load Balancer component to be installed and maintained. Therefore, it increases the installation and configuration complexity. As the Load Balancer is running on a separate system there are more systems to manage and to operate, which in turn, increases the cost for the operation of the environment.

Installation and configuration

The following sections indicate the minimum software configuration that you need for the topology shown in Figure 5-5 on page 151:

System A and System B

Perform the following steps to configure System A and System B.

1. Install WebSphere Edge components Load Balancer.
2. Configure the Load Balancer component according to your network topology.
3. Install and configure a supported Web server.
4. Install and configure the WebSphere Application Server Plug-in for the Web server.

System C and System D

Perform the following steps to configure System C and System D.

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the `<app_server_root>/profileTemplates/managed` profile template and federate this profile to the deployment manager on system E either during profile creation or by running the **addNode** command after the profile creation.

Alternatively you can create an application server profile using the `<app_server_root>/profileTemplates/default` profile template and federate the node to the deployment manager running on System E by using the **addNode** command.

System E

Perform the following steps to configure System E.

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the profile template `<app_server_root>/profileTemplates/dmgr` template or use the `<app_server_root>/profileTemplates/management` template and specify `-serverType` as `DEPLOYMENT_MANAGER`.
3. Create the Web server definitions through the Integrated Solutions Console or through the `wsadmin` scripting interface.
4. Create a WebSphere Application Server cluster using the Integrated Solutions Console or `wsadmin` with one cluster member on System C and one cluster member on System D.

5.3.4 Reverse proxy topology

Reverse proxy servers, like the one provided with the Edge components or the DMZ secure proxy, are typically used in DMZ configurations for two major reasons:

- ▶ To provide additional security between the public Internet and the Web servers (and application servers)
- ▶ To increase performance and reduce the load on the servers by content caching

The topology shown in Figure 5-6 illustrates the use of a reverse proxy server.

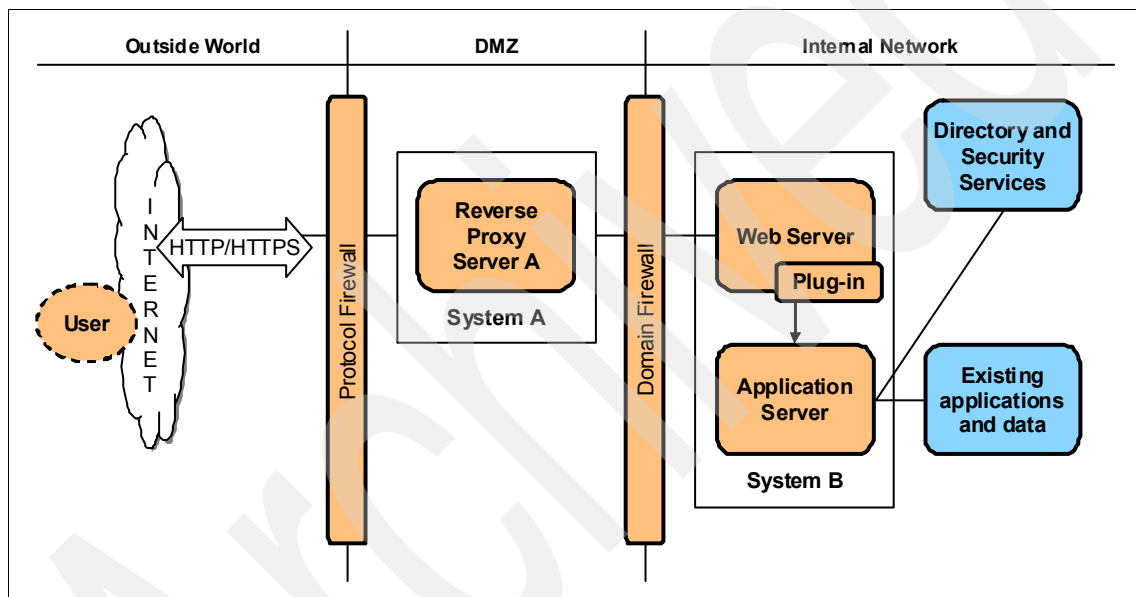


Figure 5-6 Topology using an Edge Components reverse proxy

Note: The illustration in Figure 5-6 is intended to provide an overview of what a topology, containing a reverse proxy might look like. The reader should be reminded that high availability is not incorporated here. To achieve high availability it would be required to have multiple reverse proxy servers and two sets of Load Balancer cluster addresses. One cluster address for the proxy servers and one cluster address for the Web servers.

In this example, a reverse proxy in the DMZ listens on the HTTP and HTTPS port (typically port 80 and 443) for requests. The reverse proxy intercepts the incoming requests and verifies if there is a valid copy of the requested object in its cache. If a valid, cached version is found, the cached copy is returned to the client. If no valid copy of the requested object is found in the local cache it 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 from the clients and allowing the proxy server to store a copy of the object in the local cache, if the configuration permits.

Reverse proxy configurations support high-performance DMZ solutions that require as few open ports in the firewall as possible. The reverse proxy requires only one open port per protocol to access the Web server behind the firewall.

Advantages

Advantages of using a reverse proxy server in a DMZ configuration are as follows:

- ▶ Independent configuration

The reverse proxy installation has no effect on the configuration and maintenance of a WebSphere application.

- ▶ Improved performance due to caching

As the reverse proxy server provides caching capabilities, in most cases it can respond faster to client requests because objects can be served out of the cache. In most cases the response time and performance will improve.

- ▶ Off loading the Web servers

The reverse proxy servers delivered with WebSphere Application Server V7.0 provide caching capabilities, off-loading the Web servers and the application servers if dynamic caching is supported as well.

Disadvantages

Disadvantages of a reverse proxy server in a DMZ configuration are as follows:

- ▶ Increased complexity

This configuration requires the Caching Proxy component to be installed and maintained, increasing the installation and configuration complexity. As the Load Balancer is running on a separate system there are more systems to manage and to operate, which in turn, increases the cost for the operation of the environment. Consider these costs against the advantages you realize.

- ▶ Increased latency for non-cacheable objects

Requests for non-cacheable objects increase network latency and lower performance. To be effective, a sufficiently high cache hit rate is required.

Installation and configuration: Edge Proxy

The following sections indicate the minimum software configuration that you need for the topology shown in Figure 5-6 on page 154.

System A

Perform the following steps to configure System A.

- a. Install WebSphere Edge components.
- b. Configure the required proxy directives in `ibmproxy.conf`.

System B

Perform the following steps to configure System B.

- a. Install and configure a supported Web server.
- b. Install and configure the WebSphere Application Server plug-in for the Web server WebSphere Application Server (Express, Base, or Network Deployment).
- c. Create an application server profile.
- d. Install WebSphere Application Server V7.0.
- e. Create an application server profile using the `<app_server_root>/profileTemplates/default profile template` .
- f. Create an Web server definition through the Integrated Solutions Console or through the `wsadmin` scripting interface.

Installation and configuration: DMZ secure proxy

When you plan to use the DMZ secure proxy instead of the Edge Proxy you must be aware that you need WebSphere Application Server Network Deployment. The DMZ secure proxy is not supported when using the base version of WebSphere Application Server. Therefore, the topology will slightly change and look similar to the illustration as shown in Figure 5-7 on page 157.

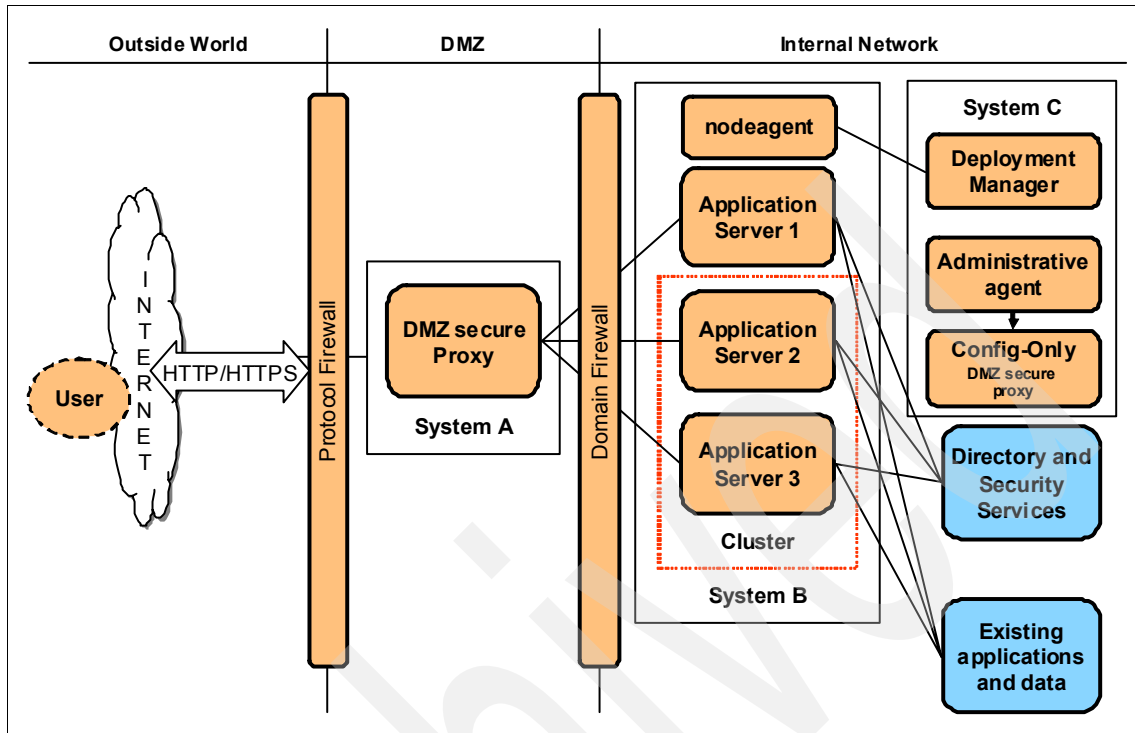


Figure 5-7 Topology using a DMZ secure proxy server

The following sections indicate the minimum software configuration steps you need to perform for the topology shown in Figure 5-7. In this example, we are assuming that the DMZ secure proxy is set up with security level HIGH and therefore supports only static routing.

System A

Perform the following steps to configure System A.

1. Install the DMZ secure proxy software package.
2. Create an Application Server Profile using the `<app_server_root>/profileTemplates/secureproxy` template.

System B

Perform the following steps to configure System B.

1. Install WebSphere Application Server V7.0
2. Create an application server profile using the `<app_server_root>/profileTemplates/managed` profile template. Federate this profile to the deployment manager on system C either during profile creation, or by running the **addNode** command after the profile creation.

Alternatively you can create an application server profile using the `<app_server_root>/profileTemplates/default` profile template and federate the node to the deployment manager running on System C by using the **addNode** command.

System C

Perform the following steps to configure System C.

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the profile template `<app_server_root>/profileTemplates/dmgr` template or use the `<app_server_root>/profileTemplates/management` template and specify `-serverType` as `DEPLOYMENT_MANAGER`.
3. Create WebSphere Application Server clusters or un-clustered servers on System B.
4. Deploy the applications to the application servers and make sure they are started.
5. Because we are setting up a DMZ secure proxy with security level HIGH, only static routing is supported. We have to export the routing information by performing the following steps:
 - a. Go to the `<profile_root>/bin` directory of the deployment manager profile
 - b. Start **wsadmin.bat (sh) -lang jython** using the Jython scripting language and run the commands shown in Example 5-1 to export the static routing information.

Example 5-1 Sample Jython code to export static routing information

```
mbean=AdminControl.queryNames('*:*',type=TargetTreeMbean,process=dmgr')
AdminControl.invoke(mbean, 'exportTargetTree',
'<directory>/targetTree.xml')
```

6. Copy the file `<directory>/targetTree.xml` to System A.

7. Create an application server profile using the `<app_server_root>/profileTemplates/management` profile template and specify `-serverType` as `ADMIN_AGENT` to create the administrative agent profile.
8. Create an Application Server Profile using the `<app_server_root>/profileTemplates/secureproxy` template.

Note: The profiles for the administrative agent and the DMZ secure proxy are for administration purposes only. The DMZ secure proxy server is a configuration-only profile. This means that the server cannot be started or used for any work. This server is an administrative place-holder for the DMZ secure proxy server on System A. If you try to start the configuration-only profile, it will fail with the following error message in the `SystemOut.log`:

Caused by:

```
com.ibm.ws.proxy.deployment.ProxyServerDisabledException: This
secure proxy server is part of a configuration-only installation
and cannot be started.
```

Tip: It is recommended to name the proxy server and the node name in the configuration-only profile on System C the same as you did when creating the proxy server on System A using parameters: `-serverName` and `-nodeName` when running `manageprofiles`.

9. Register the DMZ secure proxy configuration-only profile to the Administrative agent.
10. Use the Integrated Solutions Console from the administrative agent to manage the DMZ secure proxy configuration-only template.
11. Export the configuration-only DMZ secure proxy to move the changes over to the real DMZ secure proxy. You need to perform the following steps:
 - a. Go to the `<profile_root>/bin` directory of the configuration-only DMZ secure proxy profile.
 - b. Start `wsadmin -lang jython -conntype NONE`.
 - c. Run the command shown in Example 5-2 to export the proxy profile.

Example 5-2 Sample code to export proxy profile

```
AdminTask.exportProxyProfile('[-archive
<directory>/DMZProxy.car]')
```

12. Copy the file `<directory>/DMZProxy.car` to System A.

System A

1. Copy the file targetTree.xml to the <profile_root>/staticRoutes directory.

Note: The static routing information is not updated automatically. Whenever there is a change (for example when an application is installed or removed) the routing information needs to be manually refreshed. The DMZ secure proxy servers needs to be restarted after each refresh of the routing information to activate the change. If this is not feasible you need to switch the dynamic routing to use a lower security level.

2. Go to the <profile_root>/bin directory and start `wsadmin -lang jython -conntype NONE`.
3. Import the profile changes from <directory>/DMZProxy.car by running the `wsadmin` command shown in Example 5-3.

Example 5-3 Sample code to import proxy profile

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

4. Start the DMZ secure proxy server.

5.3.5 Topology with redundancy of multiple components

To remove single points of failure in a topology, redundant components are added. Most components in a WebSphere Application Server topology provide the options to implement redundancy. For example, Load Balancer hot standby server with the primary Load Balancer server, clustered Web servers, clustered application servers, and so forth. The topology shown in Figure 5-8 on page 161 illustrates the minimum WebSphere components used in an installation with the usual high availability requirements (whereby the number of application servers might vary).

Figure 5-8 illustrates a topology with redundancy of several components.

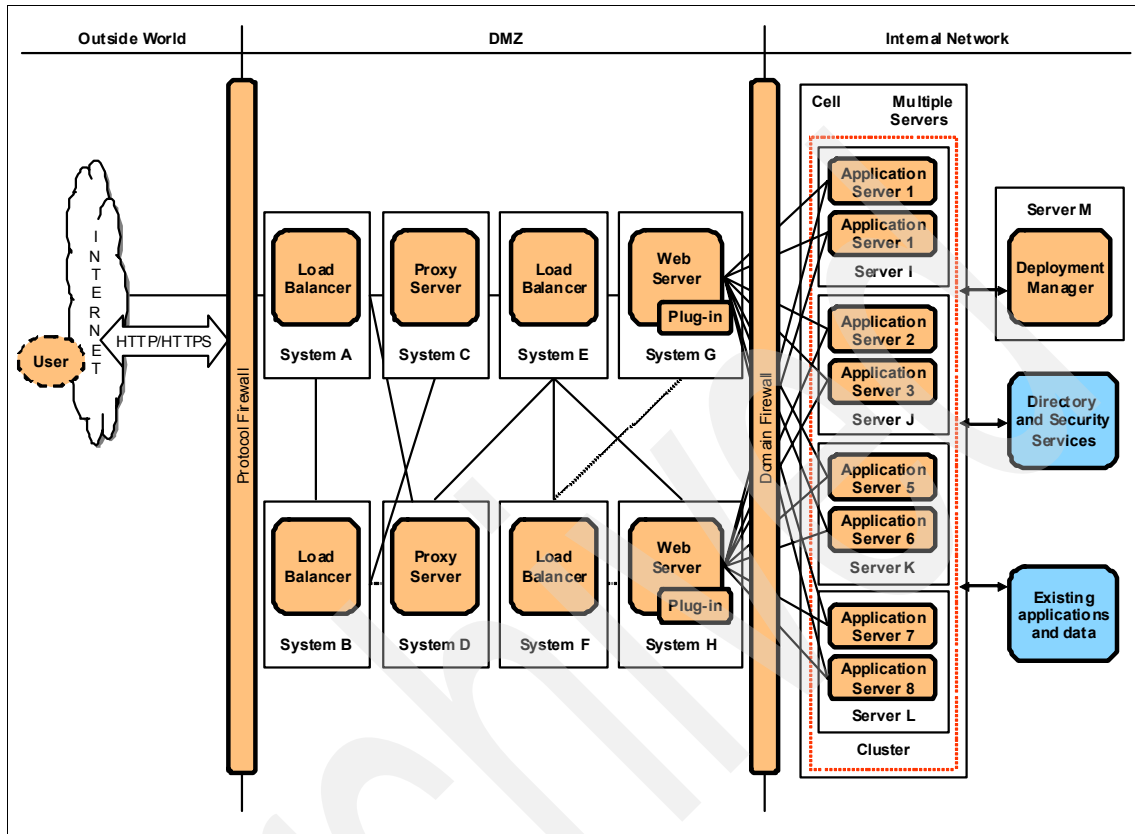


Figure 5-8 Topology with redundancy of several components

The following components are redundant in this example:

- ▶ Two clusters of Load Balancers

In the illustrated topology there are two clusters of load balancers. Each of the clusters provides a highly available cluster address. The first cluster, which runs on System A with System B as hot standby, provides the cluster address for the reverse proxies. This cluster address is used by the clients to access the service.

The second cluster runs on System E, with System F as hot standby, provides the cluster address for the Web servers, and is used by the proxy servers to retrieve content if user requests cannot be served out of the cache. Refer to “Horizontal scaling topology with an IP sprayer” on page 150 for a high level overview how Load Balancers work.

Note: Each Load Balancer installation can host multiple cluster addresses. You do not necessarily need a separate installation for each cluster address.

▶ Two reverse proxy servers

Both of the reverse proxy servers (running on System C and System D) are receiving requests from the Load Balancer and share the requests coming from the clients. Each proxy server is installed on a different system to provide a maximum level of redundancy. Keep in mind that both reverse proxy servers must have an identical configuration.

▶ Two Web servers

Each Web server, one running on System G and the other one on System H, receives requests from the second Load Balancer cluster and shares the requests that come from the reverse proxies. Each Web server is installed on a different machine but they must have an identical configuration.

▶ An application server cluster

The cluster is spread across four server systems and implements a combination of vertical and horizontal scaling. The cluster consists of eight cluster members, two on each server. Although the application servers are grouped together in one cluster, they might originate from different installations. The application servers on System I, for example, could be two separate installations of WebSphere Application Server. The application servers on System J could be profiles of a single installation.

▶ Two database servers

The database servers need to be made highly available using database system specific tools or operating system-based clustering software.

▶ Two LDAP servers

The LDAP servers could use a high availability software product (like another Load Balancer) or the backup LDAP server support (as provided through federated repositories user registry in WebSphere Application Server). The LDAP servers must have identical structure and user population.

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

Advantages

The major advantages of this topology are as follows:

- ▶ Most single points of failure eliminated

There are no single points of failure. The Load Balancer node, reverse proxy server, Web server, application server, database server, and LDAP server are set up in a redundant way.

Note: There are still components in this topology, such as the deployment manager, cell-wide services like the HAManager, and JNDI, which are not highly available.

- ▶ Horizontal scaling in place

Horizontal scaling is done by using both the IP sprayer (for the reverse proxy and the Web server nodes) and application server cluster technology to maximize availability. The benefits from horizontal scaling are discussed in 5.3.3, “Horizontal scaling topology” on page 147.

- ▶ Improved application performance

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

- Hosting application servers on multiple physical machines, z/OS images, or both to optimize the usage of available processing power.
- Using clusters to scale application servers vertically, which makes more efficient use of the resources of each machine.

- ▶ Using workload management technologies

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

- Load Balancer Network Dispatcher to distribute client HTTP requests to each reverse proxy server
- Load Balancer Network Dispatcher to distribute requests from the proxy servers to each Web server
- WebSphere Application Server Network Deployment workload management feature to distribute work among clustered application servers

Disadvantages

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

Note: As this topology is a combination of topologies described before, the disadvantages of the other base topologies apply here as well.

Installation and configuration

The following sections indicate the minimum software configuration that you need for the topology shown in Figure 5-8 on page 161:

System A, System B, System E and System F

Perform the following steps to configure System A, System B, System E and System F.

1. Install WebSphere Edge components Load Balancer.
2. Configure the Load Balancer component according to your network topology.

Note: Keep in mind that System A and System B form one cluster and that System E and System F form another, different cluster

System C and System D

To configure System C and System D, install and configure the Edge Components Caching Proxy.

Note: If you plan to implement the DMZ secure proxy you would not need the Systems G and H.

System G and System H

Perform the following steps to configure System G and System H.

1. Install and configure a supported Web server.
2. Install and configure the WebSphere Application Server Plug-in for the Web server.

Servers I, J, K, and L

Perform the following steps to configure Servers I, J, K, and L.

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the `<app_server_root>/profileTemplates/managed` profile template. Federate this profile to the deployment manager on System M either during profile creation or by running the **addNode** command after the profile creation.

Alternatively you can create an application server profile using the `<app_server_root>/profileTemplates/default` profile template and federate the node to the deployment manager running on System C by using the **addNode** command.

Server M

Perform the following steps to configure Server M.

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the `<app_server_root>/profileTemplates/dmgr` profile template, or use the `<app_server_root>/profileTemplates/management` template and specify `-serverType` as `DEPLOYMENT_MANAGER`.
3. Create an Web server definitions through the Integrated Solutions Console or through the wsadmin scripting interface.
4. Create a WebSphere Application Server cluster with two cluster members on System I, J, K, and L.

5.3.6 Heterogeneous cell

Cells can span servers across multiple heterogeneous operating systems like z/OS sysplex environments and distributed platforms. For example, z/OS nodes, Linux nodes, UNIX nodes, and Microsoft Windows nodes can exist in the same application server cell. This kind of configuration is referred to as a heterogeneous cell. With WebSphere Application Server V7.0, there are many different topologies that are possible to compose a heterogeneous cell, as shown in Figure 5-9.

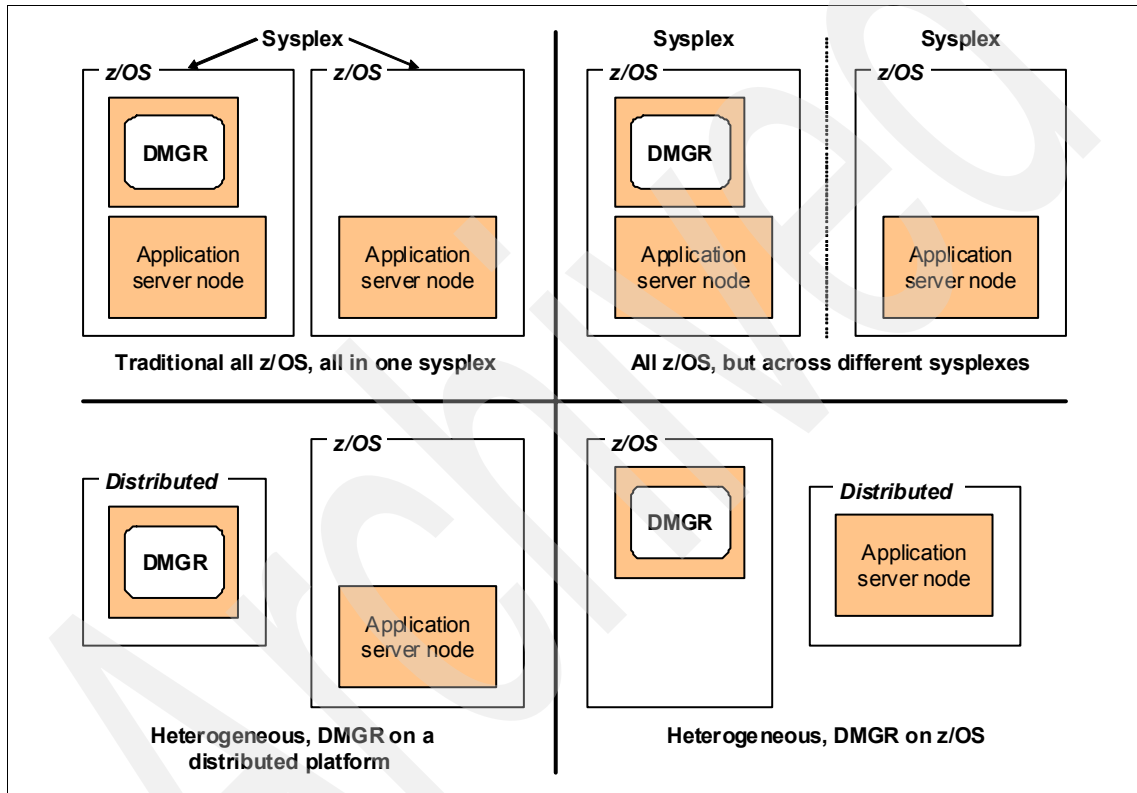


Figure 5-9 Different possibilities with a heterogeneous cell

WebSphere Application Server Version 7.0 products can coexist with the following supported versions:

- ▶ IBM WebSphere Application Server Version 5.1
- ▶ IBM WebSphere Application Server Network Deployment Version 5.1
- ▶ IBM WebSphere Application Server Version 5.1 for z/OS
- ▶ IBM WebSphere Application Server Version 6.0
- ▶ IBM WebSphere Application Server Network Deployment Version 6.0
- ▶ IBM WebSphere Application Server Version 6.0 for z/OS
- ▶ IBM WebSphere Application Server Version 6.1
- ▶ IBM WebSphere Application Server Network Deployment Version 6.1
- ▶ IBM WebSphere Application Server Version 6.1 for z/OS

WebSphere Application Server Version 5.1, Version 6.0, and Version 6.1 clients can coexist with Version 7.0 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 are as follows:

- ▶ Benefits from horizontal and vertical scaling described in previous sections

- ▶ Flexible deployment of components

Components can be deployed to systems on which they provide the best value and effectiveness.

- ▶ Easier integration and reuse of existing software components

As multiple systems can be included in the cell, the integration of existing, platform-specific software components is much easier.

- ▶ Easier migration

Running different versions and platforms of WebSphere Application Server in a cell is a possible migration approach for migrating WebSphere Application Server versions. Although this is a supported environment mixed version cells should not be considered a permanent solution.

Disadvantages

The disadvantages of this topology are as follows:

- ▶ Complex administration
Due to the heterogeneous environment, the administration is complex and requires administrator knowledge for all platforms.
- ▶ Increased administration and operational costs
This combined topology has the disadvantages of costs in hardware, configuration, and administration. Consider these costs in relation to gains in performance, throughput, and reliability.

For information about planning and system considerations required to build a heterogeneous cell, see the IBM White Paper *WebSphere for z/OS -- Heterogeneous Cells*, available at the following Web page:

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

5.3.7 Multi-cell topology

The topologies introduced in 5.3.5, “Topology with redundancy of multiple components” on page 160, and in 5.3.6, “Heterogeneous cell” on page 166 provide a high level of availability and redundancy of all sorts of WebSphere components. Nevertheless, application software problems or malfunctioning of cell level components such as high availability manager, though rare, are potential threats to the availability of your service.

High availability and disaster recovery are two terms that need to be addressed differently. A possible approach would be a two cell architecture as outlined in Figure 5-10 on page 169

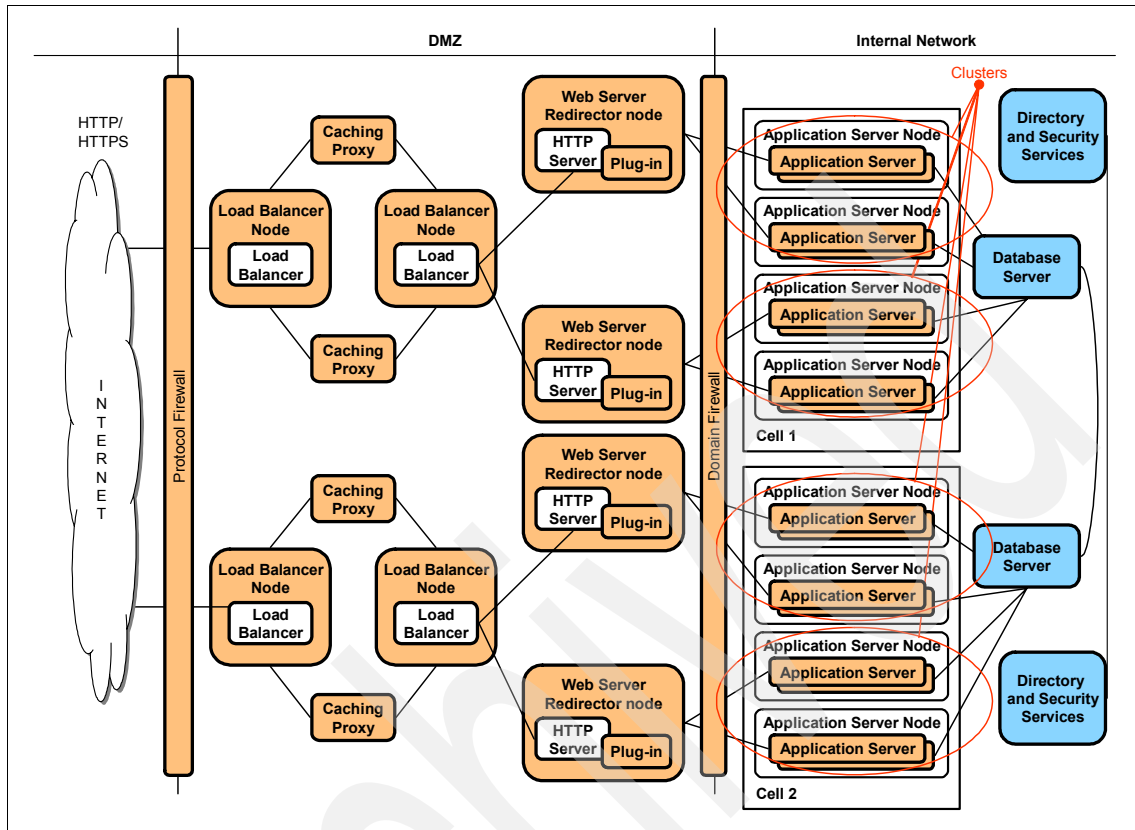


Figure 5-10 Multi-cell architecture

This topology is basically a duplication of the topology introduced in 5.3.5, “Topology with redundancy of multiple components” on page 160. Here we are implementing independent cells, both providing service. Even cell level problems can be handled quickly in this topology, as full cells can be activated and deactivated as needed.

Advantages

This topology provides the following advantages:

- ▶ Provides all the advantages defined in 5.3.5, “Topology with redundancy of multiple components” on page 160
- ▶ Allows you to react quickly to cell level problems
- ▶ Allows stepwise WebSphere upgrades

This topology allows independent releases of WebSphere Application Server software in each cell. Therefore, each cell can be upgraded on its own. This lowers the risk of an upgrade and provides a fall-back scenario in case of upgrade problems.

- ▶ Allows stepwise application upgrades

Using this topology allows independent application releases in each cell and provides a quick fall back scenario in case you determine application problems in your production environment².

- ▶ Possible approach for disaster recovery

If the cells are located in different data centers, it is a possible approach for a disaster recovery solution from a WebSphere perspective.

Note: The topology illustrated in Figure 5-10 is not a complete solution for disaster recovery. For a real disaster recovery solution implementation several issues need to be addressed:

- ▶ How do you route traffic to each of the cells?
- ▶ How will you handle affinity of requests?
- ▶ How will you handle session data?
- ▶ How will you handle security data?
- ▶ How will you address the data replication and consistency challenge?
- ▶ How will you handle a cell fail-over for each type of requests in your application? Web requests, SIP requests, EJB Requests, Web services and so forth

- ▶ Co-location of cells is possible

You can collocate the two cells on the same systems to achieve the software release independence described before. Be aware that this limits the disaster recovery usability.

² Of course this requires some sort of compatibility between application software releases

Disadvantages

This topology offers the following disadvantages:

- ▶ Increased cost

This multi-cell topology has the disadvantages of costs in hardware, complexity, configuration, and administration. But you need to consider these costs in relation to gains in performance, throughput, and reliability. You will likely have specific requirements to consider an architecture like that.

Note: As this topology is a combination of topologies described before the disadvantages of these base topologies apply here as well.

Installation and configuration

The installation and configurations steps for this topology are the same as in “Installation and configuration” on page 164.

5.3.8 Advanced topology using an administrative agent

Even if single server deployments of WebSphere Application Server do not provide any load-balancing and fail-over capabilities (with the exception of WebSphere Application Server for z/OS), there are several installation scenarios where a single server installation is sufficient according to the requirements. To optimize such single server installations, it makes sense to run multiple single server environments on one system to fully use the available system capacity.

Flexible management is an improvement in WebSphere Application Server V7.0 to reduce administration cost of large WebSphere deployments. One of the enhancements of WebSphere Application Server V7.0 addressing this issue is the administrative agent. For further information about the administrative agent see 3.1.7, “Administrative agent” on page 63. To optimize the administration of large single server deployments on one system, the administrative agent can be used. The purpose of the administrative agent is to reduce the administration cost and overhead.

The topology in Figure 5-11 shows a possible topology using an administrative agent to manage all the single server installations on System B. Instead of running the Configuration service, Integrated Solutions Console application, and so forth in each of the application servers, these services are running in the administrative agent for all profiles. The administrative agent therefore not only reduces the administrative overhead for the installation but also simplifies the administration, as all the administrative access uses one central point. Therefore, you have one URL for the administration instead of multiple URLs.

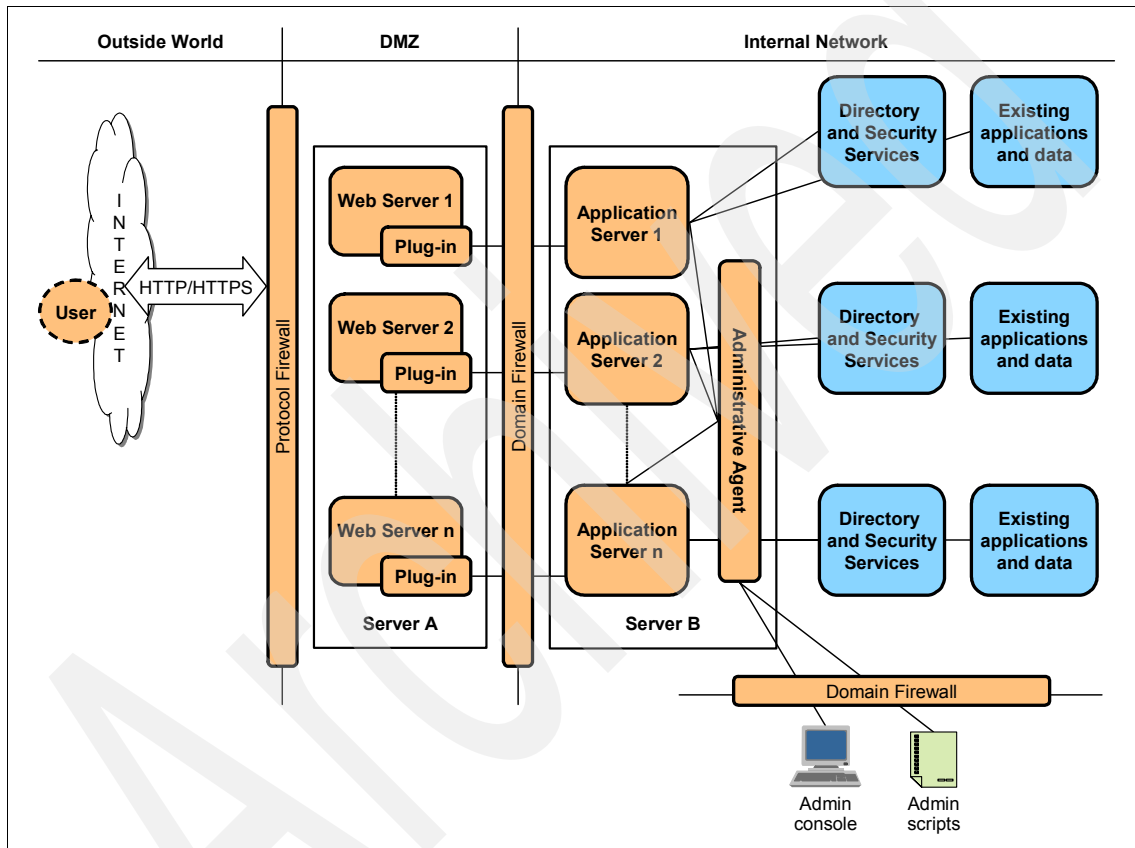


Figure 5-11 Topology containing administrative agent

Advantages

The implementation of an administrative agent profile provides several benefits for the installation as follows:

- ▶ Reduced administrative footprint

The administrative footprint is reduced when using multiple single server installations in the same system.

- ▶ Central access to administration tools
The access to the administrative tools (Integrated Solutions Console, wsadmin) is simplified, as all access is through the administrative access. Only one URL is used instead of multiple URLs, as without administrative agents.
- ▶ Less firewall ports required
If there is a firewall between administrator's workstation and the servers, less ports need to be opened on the firewall. You only need accessibility to the administrative agent instead to each application server.

Disadvantages

The implementation of an administrative agent profile has the disadvantage of requiring an additional JVM.

An additional JVM is running on the system. It requires multiple single servers to manage through the administrative agent to avoid an increased memory footprint for the overall solution.

Installation and configuration

To setup an environment as illustrated in Figure 5-11 on page 172 the following installation and configuration steps are required:

System A

Perform the following steps to configure System A.

1. Install and configure a supported Web server.
2. Install and configure the WebSphere Application Server plug-in for the Web server.

Note: In this topology there are multiple ways to install and configure your Web server. You can run one instance of a Web server, multiple instances of the same Web servers or even multiple installations if different Web servers. All you have to make sure that each Web server's plug-in points to the correct application server on System B.

System B

Perform the following steps to configure System B.

1. Install WebSphere Application Server V7.0.
2. Create as many application server profiles using the `<app_server_root>/profileTemplates/default` profile template as required. These are your single server profiles running your applications.

3. Create an application server profile using the `<app_server_root>/profileTemplates/management` profile template and specify `-serverType` as `ADMIN_AGENT`.
This step creates the administrative agent profile.
4. Go to the binary directory of your Administrative profile and register each single server profile to the administrative agent (**registerNode**).
5. Open the Integrated Solutions Console or a `wsadmin` session to the administrative agent and select the application server you want to manage.
6. For each single server installation, create an Web server definition as needed for your environment.

5.3.9 Advanced topology using a job manager

Under the aspect of flexible management in WebSphere Application Server V7.0, the job manager function was introduced as a second component beside the administrative agent. For more information about the job manager, see 3.1.8, “Job manager” on page 64.

The job manager addresses scalability issues of the administrative runtime if the components are spread over multiple remote locations. An example of such a deployment is a typical branch deployment where central management is desired but the nodes themselves are in branch locations.

The job manager uses a loosely coupled asynchronous administration model to manage a number of remote endpoints. The job manager introduces different administrative options and flexibility to set up a centralized administration model. The topology illustrated in Figure 5-12 on page 175 demonstrates a possible usage of a job manager for the central administration of multiple heterogeneous environments.

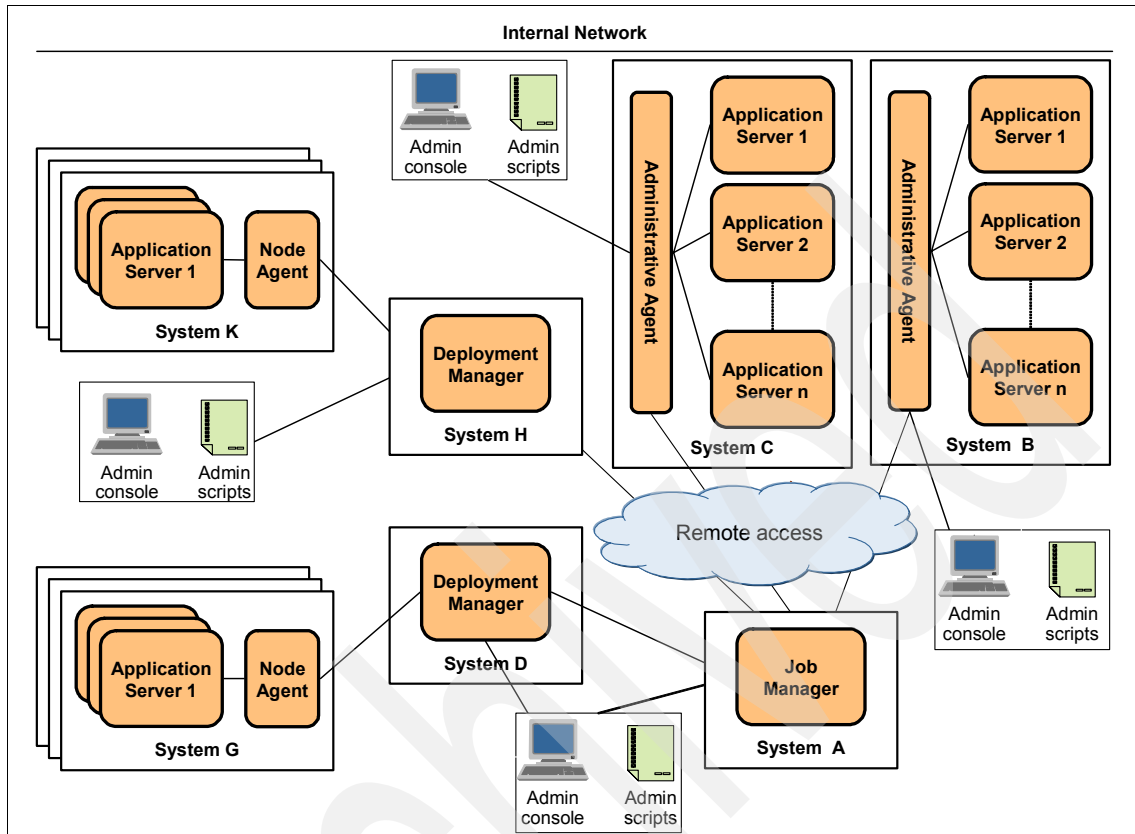


Figure 5-12 Topology using a job manager

The topology shown in Figure 5-12 is a possible usage scenario for the job manager. The job manager node on System A acts as a coordinator across multiple deployment managers (System H and System D) and administrative agents (System B and System C) through its asynchronous job management capabilities. Keep in mind that the job manager is no replacement for the deployment managers or administrative agents. The job manager relies on the local management capabilities to execute the management jobs.

Advantages

Running a job manager in your environment provides several advantages for the administration of your deployments:

- ▶ Allows central, remote management of multiple different administrative entities through WAN networks
- ▶ Allows local and remote management of each installation
- ▶ Enhances the existing management models

Disadvantages

There are no real disadvantages specific to the job manager, except that you need an additional JVM (namely the jobmgr application server) running.

Installation and configuration

To set up the environment illustrated in Figure 5-12 on page 175, the following installation and configuration steps are required:

System A

Perform the following steps to configure System A.

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the `<app_server_root>/profileTemplates/management` profile template and specify `-serverType` as `JOB_MANAGER`.

This step creates the job manager profile.

System B and System C

Perform the following steps to configure System B and System C.

1. Install WebSphere Application Server V7.0.
2. Create as many application server profiles using the `<app_server_root>/profileTemplates/default` profile template as required. These are your single server profiles running your applications.
3. Create an application server profile using the `<app_server_root>/profileTemplates/management` profile template. Specify `-serverType` as `ADMIN_AGENT`.

This step creates the administrative agent profile.

4. Go to the binary directory of your Administrative profile and register each single server profile to the administrative agent (**registerNode**).
5. Use **wsadmin** in the binary subdirectory of the Administration agent profile directory and register the administrative agent with the job manager by running the **AdminTask.registerWithJobManager** task.

System D and System H

Perform the following steps to configure System D and System H.

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the `<app_server_root>/profileTemplates/dmgr` profile template, or use the `<app_server_root>/profileTemplates/management` template and specify `-serverType` as `DEPLOYMENT_MANAGER`.
3. Use `wsadmin` in the binary subdirectory of the deployment manager profile directory and register the deployment manager with the job manager by running the `AdminTask.registerWithJobManager` task.

Systems E, F, G, I, J, and K

Perform the following steps to configure Systems E, F, G, I, J, and K.

1. Install WebSphere Application Server V7.0.
2. Create an application server profile using the `<app_server_root>/profileTemplates/managed` profile template. Federate this profile to the proper deployment manager either during profile creation or by running the `addNode` command after the profile creation.

Alternatively, you can create an application server profile using the `<app_server_root>/profileTemplates/default` profile template and federate the node to the proper deployment manager by using the `addNode` command.

Archived

Installation

This chapter provides general guidance for planning the installation of WebSphere Application Server V7.0 and many of its components. To effectively plan an installation, you need to select a topology, hardware, and operating system, and prepare your environment for WebSphere Application Server installation.

Note: Detailed pre-requisite documentation for WebSphere Application Server V7.0 can be found in the article *System Requirements for WebSphere Application Server V7.0*, which is available at the following Web page:

<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27012284>

This chapter contains the following sections:

- ▶ “What is new in V7.0” on page 180
- ▶ “Selecting a topology” on page 181
- ▶ “Selecting hardware and operating systems” on page 181
- ▶ “Planning for disk space and directories” on page 182
- ▶ “Naming conventions” on page 184
- ▶ “Planning for the load balancer” on page 184
- ▶ “Planning for the DMZ secure proxy” on page 186
- ▶ “Planning for the HTTP server and plug-in” on page 187
- ▶ “Planning for WebSphere Application Server” on page 197
- ▶ “IBM Support Assistant” on page 226
- ▶ “Summary: Installation checklist” on page 227

This document does not describe the installation process itself. Refer to the product documentation for details about installing WebSphere Application Server V7.0. The installation documentation for WebSphere Application Server can be found in the Information Center at the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/welc6topinstalling.html>

6.1 What is new in V7.0

The following installation-related items are new or changed significantly for the WebSphere Application Server V7.0 editions for *distributed systems*:

- ▶ Non-root installation
 - When installing as a non-root user, the installers for the IBM HTTP Server and the Web server plug-in installers will now install a private copy of the IBM Global Security Kit (GSKit). The private copy of the GSKit utility is installed to sub-directory `gsk7` of the `products` root directory for the product which installs GSKit. So this is either `<web_server_root>/gsk7` or `<plugins_root>/gsk7`. This allows for root and non-root installations to use secure socket layer (SSL) support.
 - Verification of file systems permission before install

Before installing the product, users can verify that the adequate file permissions exist. This verification can be invoked either during a check box on the installation summary panel or the `-OPT checkFilePermissions="true"` option in the response file when installing silently. File system verification takes place before the install begins so as to avoid an undefined product state due to insufficient file permissions.
 - Changing ownership to another user after installation

After the installation of WebSphere Application Server, the `<app_server_root>/instutils/chutils` command can be used to change the file ownership to another user or group for future operations.

Note: There are still some limitations when installing WebSphere Application Server as non-root user. Check the WebSphere Information Center for details:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.installation.nd.doc/info/ae/ae/cins_nonroot.html?

- ▶ Installation language support

The installer for WebSphere Application Server provides fine-grained language support for the Integrated Solutions Console and the application server runtime. This allows you to update language or add new packs at a later time.

- ▶ Installing maintenance packages

The IBM Update Installer for WebSphere Software V7.0 supports multiple releases (namely V6.0.2 (from 6.0.2.21 onward), V6.1 and V7.0). A single installed instance of the Update Installer can be used to install updates to the above mentioned versions of WebSphere Application Server.

Similar to the WebSphere Application Server installer, the Update Installer provides a file permission verification feature that allows you to verify file system permissions before the actual update begins, which reduces the risk that the installation fails in an undefined state.

6.2 Selecting a topology

Chapter 5, “Topologies” on page 121 describes some common configurations. 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, map the components from that topology to a specific hardware and operating system and plan for the installation of the required products.

6.3 Selecting hardware and operating systems

After selecting a topology, the next 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:

- ▶ Existing conditions
- ▶ Future growth
- ▶ Cost
- ▶ Skills within your company

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

Along with selecting the hardware 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 and to get support. Keep in mind that not every product you receive with WebSphere Application Server V7.0 is supported on each operating system and platform.

For an updated list of the hardware and software requirements and supported platforms for WebSphere Application Server V7.0, review the system requirements for WebSphere Application Server V7.0, available at the following Web page:

<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27012284>

6.4 Planning for disk space and directories

Before installing WebSphere Application Server components you must provide sufficient disk space for a successful installation and for an operation of the environment.

Note: In this section the term *file system* stands as a synonym for *manageable disk storage*. This can be file systems on UNIX-based systems, disk partitions, and drive letters on Windows, HFS/zFS for z/OS, and so on.

Although WebSphere Application Server products provide a default directory structure for their components, it might not be the best choice as the default structure might limit the flexibility or become inconsistent in terms of naming. Keep in mind that your directory names are bound to the naming rules.

There is no general rule if you should stick to one file system following the default directory structure or create multiple file systems using different mount points. Generally, disk space management is more flexible and efficient if you split the installation in different file systems. Planning your directory structure and file systems allows you to consider other criteria like performance, backup requirements and capabilities, availability, and so on.

Different file systems for the following purposes are useful:

▶ Application binaries

This file system stores the product binaries as dumped by the installer. When designing this file system keep in mind that installing maintenance causes this file system to grow.

▶ Profiles

This file system stores the profile-specific data that defines your runtime environment. The minimum disk space required depends on the profile type you create. The amount of user data needed depends on the applications deployed to the profile.

▶ Log files

The purpose of this file system is to hold the log files of the application servers. If this file system is not mounted under the default mount point you have to change the server configuration for each server. This can be accomplished either through scripting or by using a custom server template.

The size depends on the application and on the log retention policy of the application servers.

▶ Dumps

System core dumps and Java heap dumps can be large and quickly fill a file system. Therefore, it is a good practice to redirect system dumps, Java heap dumps, and the Java core dumps to a dedicated directory.

This avoids a dumping process or Java virtual machine (JVM) filling up file systems and impacting other running applications. It also allows you to locate them easily. The size depends on the number of JVMs dumping to this directory, their individual sizes, the number of dumps you want to retain, and so on.

▶ Maintenance packages/CIM repositories

For easier management of your installation packages and your fix packs or individual fixes you should keep them in a central repository. If you use the centralized installation manager (CIM), this repository is located on the deployment manager system.

Note: You do not need this file system on every server. Just maintain one repository containing all your installation media, fixes, fix packs, and so on, to have all installables at hand in case you need them.

▶ User data and content

This file system should be used to store other user data and content being used in the applications.

6.5 Naming conventions

Naming conventions make the runtime environment more comprehensible. A consistent naming convention helps standardize the structure of the environment and allows for easy expansion of the environment and each component.

Naming conventions should be developed, established, and maintained for the hardware and networking infrastructure, as well as the WebSphere Application Server infrastructure, applications, and resources. When it comes to naming, most companies have already developed a working naming convention for 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. With a proper naming convention, you should be able to understand the purpose of an artifact by just looking at its name.

When you develop a naming convention, take into consideration which hardware and software components are affected and what naming restrictions apply. On many systems there are naming restrictions in terms of specific characters and length of the names. In a heterogeneous environment this might become a pitfall. Experience shows that it is best to avoid any special or national language specific characters in the names.

6.6 Planning for the load balancer

Before starting the installation of the load balancer, several planning tasks must be finished. The first pre-requisite to install the Load Balancer is that the detailed network planning for your environment is finished and that you have an exact understanding of the data flow in your environment. Edge Components V7.0 are shipped with two versions of the Load Balancer:

- ▶ Load Balancer for IPv4
- ▶ Load Balancer for IPv4 and IPv6

Note: Several features of the Load Balancer for IPv4 were deprecated when WebSphere Application Server V6.1 was announced. For a detailed list of the deprecated features see the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rmig_depfeat.html#depv61

Unless you have a specific requirement to use the Load Balancer for IPV4 you should consider the newer Load Balancer: Load Balancer for IPv4 and IPv6.

6.6.1 Installation

Before starting with the installation, consult the IBM Information Center for the type of Load Balancer you will install to ensure that all the required hardware and software pre-requisites are met. The hardware and software requirements for the Edge Components can be found at the following Web page:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27012372>

The installation for both types of Load Balancer is done through the platform-specific package manager. Detailed installation documentation for the Load Balancer Product can be found at the following Web pages:

- ▶ For Load Balancer for IPv4, see the following Web page:
<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.edge.doc/lbip4/LBguide.htm#HDRINSTALL>
- ▶ For Load Balancer for IPv4 and IPv6, see the following Web page:
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.edge.doc/lb/info/ae/tins_installlb.html

6.6.2 Configuration

After the installation of the product you have to configure the Load Balancer for your environment. The two available Load balancers provide a wide variety of configuration options. You have multiple ways to configure the product and how the Load Balancer should forward packets. The following sections give an overview about some of these configuration tools and configuration options.

Configuration interfaces: Load Balancer for IPv4

The Load Balancer for IPv4 provides the following methods for configuration:

- ▶ Command line
- ▶ Graphical User Interface (GUI)
- ▶ Configuration Wizard

Configuration interfaces: Load Balancer for IPv4 and IPv6

The Load Balancer for IPv4 and IPv6 provides the following methods for configuration:

- ▶ Command line
- ▶ Scripts
- ▶ GUI
- ▶ Configuration Wizard

Forwarding method

Each of the available versions of the Load Balancer provides different methods to forward packages to the servers to which they are dispatching.

The Load Balancer for IPv4 provides the following forwarding methods:

- ▶ Media access control (MAC) level routing
- ▶ Network address translation (NAT)/Network address port translation (NAPT)
- ▶ Content-based routing (CBR)

Note: NAT/NAPT and the CBR forwarding method are deprecated since WebSphere Application Server V6.1

The Load Balancer for IPv6 provides the following forwarding methods:

- ▶ MAC level routing
- ▶ Encapsulation forwarding

Advisors

Advisors are used to keep track of the health of the servers to which the Load Balancer forwards the IP packets. The settings of the advisors are critical in terms of how quickly an outage of a server can be determined. The more frequent the advisor runs the quicker an outage is determined. But as advisors are basically clients for the TCP/IP protocol used for accessing the server, frequent advisor runs increase server load and network use.

The Load Balancer products provide a wide variety of build-in advisors ready to use, but also allow the usage of custom advisors. You can write your own advisor and configure Load Balancer to make decisions based of the response of your advisor.

6.7 Planning for the DMZ secure proxy

The DMZ secure proxy is a new feature of the WebSphere Application Server Network Deployment V7.0 product. In contrast to the WebSphere proxy that was introduced in WebSphere Application Server V6.0.2, this WebSphere-based proxy solution is hardened, and therefore suitable to run in a DMZ.

Note: The DMZ follows the same base principles for the installation as WebSphere Application Server itself. This means that the DMZ secure proxy differentiates between product binaries and runtime configuration files by using profiles.

The WebSphere Application Server DMZ secure proxy is available through a separate installation media. A secureproxy profile template is created upon installation of the DMZ secure proxy server and WebSphere Application Server Network Deployment. These two profiles templates are different. The network deployment installation provides a secureproxy profile template which generates a configuration only profile. This profile can be used for the administration of the DMZ secure proxy but is not runnable.

The secureproxy profile template that comes with DMZ secure proxy server is the base for a proxy server running in the DMZ and forwarding requests to the content servers.

Due to the similarity of the installation of a DMZ secure proxy server and WebSphere Application Server, this section only outlines the differences.

The following items should be addressed before starting the installation of the DMZ secure proxy server:

- ▶ Plan your file systems and directories
- ▶ Determine whether to perform a single install or multiple
- ▶ Select an installation method
- ▶ Installing updates
- ▶ Plan for profiles
- ▶ Plan for names
- ▶ Plan for TCP/IP port assignments
- ▶ Security considerations for the installation
- ▶ Installation of IBM Support Assistant Agent

For a detailed description on these items, refer to 6.9, “Planning for WebSphere Application Server” on page 197.

6.8 Planning for the HTTP server and plug-in

The options for defining and managing Web servers depends 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: 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. It is intended to help you understand the process. For detailed information about how the Plug-ins Installation Wizard works, see the *Install Guide*, which can be found on the Supplements media of the WebSphere Application Server V7.0 in the `plugin\docs` directory.

During the plug-in installation, you are asked if the installation is local or remote. Depending on the response, a certain path through the installation will occur. Figure 6-1 on page 189 illustrates the plug-in installer behavior in more detail.

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`.

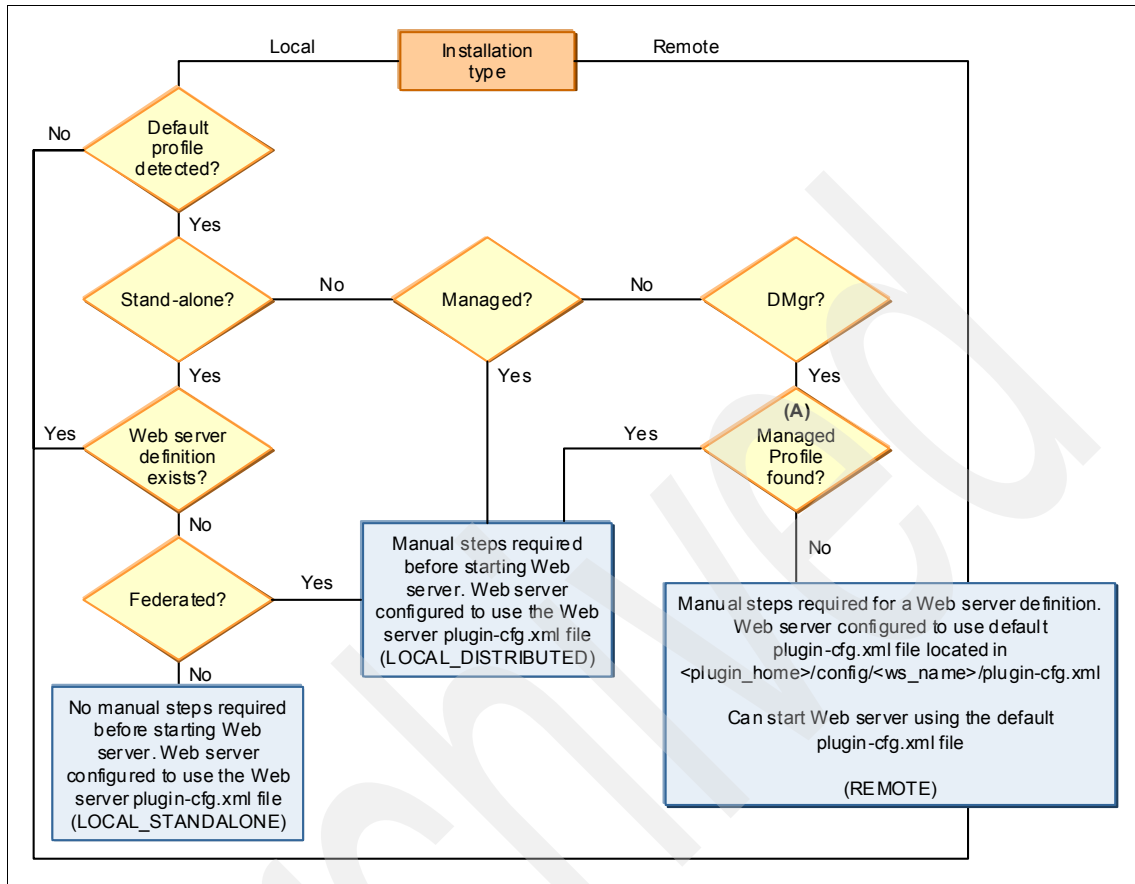


Figure 6-1 Web server plug-in installer behavior

The plug-in installer maps all possible configurations to three scenarios, (LOCAL_STANDALONE, LOCAL_DISTRIBUTED, and REMOTE) as depicted in Figure 6-1:

► LOCAL_STANDALONE

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

The Plug-ins Installation Wizard performs the following tasks in this case:

- Creates a Web server definition for the default stand-alone profile
- Configures the Web server to use the plugin-cfg.xml file

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

A LOCAL_DISTRIBUTED plug-in configuration has the following characteristics:

- A stand-alone profile federated into a deployment manager cell.
- A managed node that is either federated or unfederated in a cell.
- A managed profile found after a default deployment manager cell detected. See (A) in Figure 6-1 on page 189.

The Plug-ins installation wizard performs the following tasks in this case:

- Does not create a Web server definition for the default distributed profile.
- 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 run the snoop servlet to verify that everything is working.

► REMOTE

A REMOTE plug-in configuration has the following characteristics:

- 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.

The plug-ins installation wizard performs the following tasks in this case:

- Does not create a Web server definition for the default distributed profile.
- Configures the Web server to use the plugin-cfg.xml file in <plugin_root>/config/<webservers_name>/plugin-cfg.xml.

What is next?

If the default plugin-cfg.xml file in the <plugin_root> directory is used, start the Web server and WebSphere Application Server and select the snoop servlet to verify that everything is working. This requires that the DefaultApplication.ear enterprise application is installed. If this is the case, the snoop servlet is accessible at the following URL:

```
http://<hostname>:<web_server_port>/snoop
```

To benefit from the Web server definition, perform the following steps:

- a. Copy the configuration script to the WebSphere Application Server machine.
- b. Run the manual Web server definition creation script.
- c. Copy the generated Web server definition plugin-cfg.xml file back to the Web server machine into the <plugin_root> directory tree. For IBM HTTP Server, you can use the propagation feature.
- d. Start the Web server and WebSphere Application Server and select the snoop servlet.

6.8.1 Stand-alone server environment

In a stand-alone server environment, a Web server can be either remote or local to the application server machine, but 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 6-2.

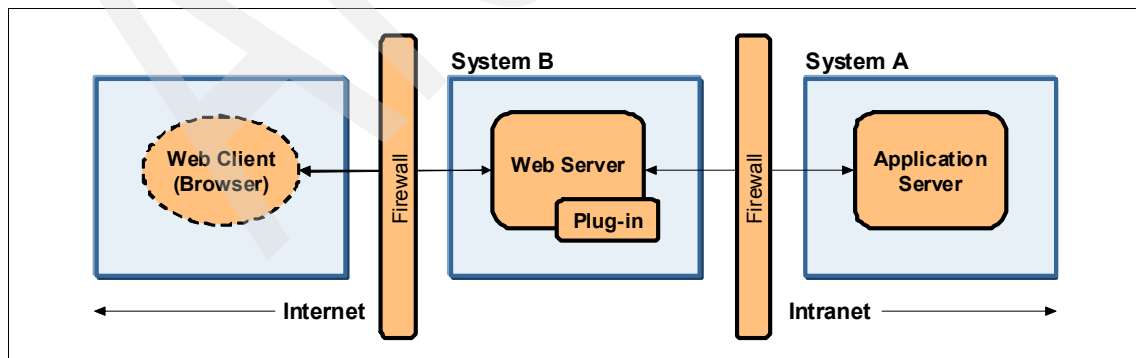


Figure 6-2 Remote Web server in a stand-alone server environment

Assume that the application server is already installed and configured on system

A. Perform the following tasks:

1. Install the Web server on system B.
2. Install the Web server plug-in on system 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 as follows:

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

During the installation, the following tasks are performed:

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

```
<plugin_root>/bin/configure<web_server_name>
```

4. At the end of the plug-in installation, copy the script to the `<app_server_root>/bin` directory of the application server machine, system A. Start the application server, 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 Web server

In this scenario, a stand-alone application server exists on system A. The Web server and Web server plug-in are also installed on system A. This topology is suited to a development environment or for internal applications. See Figure 6-3 on page 193.

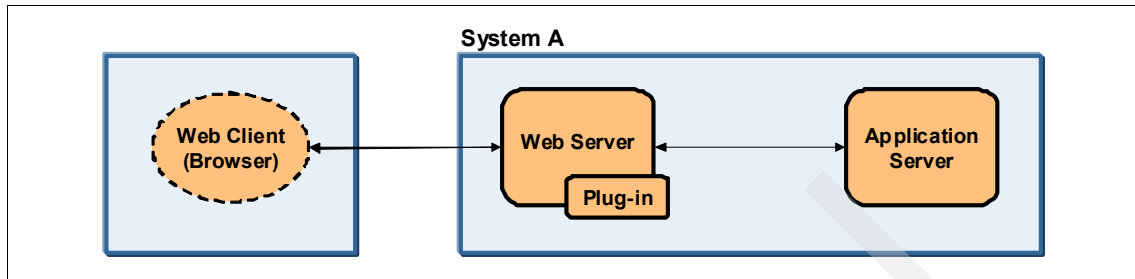


Figure 6-3 Local Web server in a stand-alone server environment

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

1. Install the Web server on system A.
2. Install the Web server plug-in on system 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 is 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 as follows:

```
<profile_root>/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:

1. The plug-in configuration file is created and placed in the location specified.
2. The Web server configuration file is updated with the plug-in configuration, including the location of the plug-in configuration file.
3. 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.

6.8.2 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. The Web servers can reside on managed or unmanaged nodes.

Remote Web server on an unmanaged node

In this scenario, the deployment manager and the Web server are on separate machines. The process for this scenario is almost identical to 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 6-4, the node is unmanaged because there is no node agent on the Web server system.

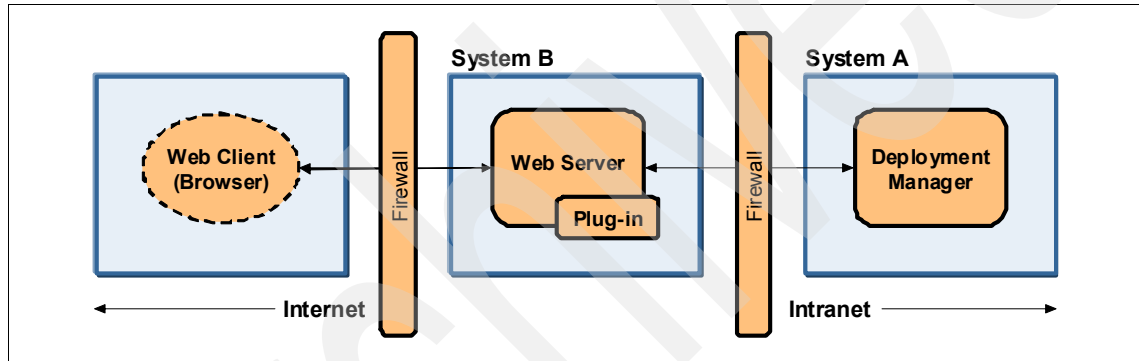


Figure 6-4 Remote Web server in a stand-alone server environment

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

1. Install the Web server on system B.
2. Install the Web server plug-in on system 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 is 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 as follows:

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

During the installation, the following tasks are performed:

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

```
<plugin_root>/bin/configure<web_server_name>
```

4. At the end of the plug-in installation, you need to copy the script to the <app_server_root>/bin directory of the deployment manager machine (system 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 system that also has a managed node. This scenario would be the same if the deployment manager was also installed on system A. See Figure 6-5.

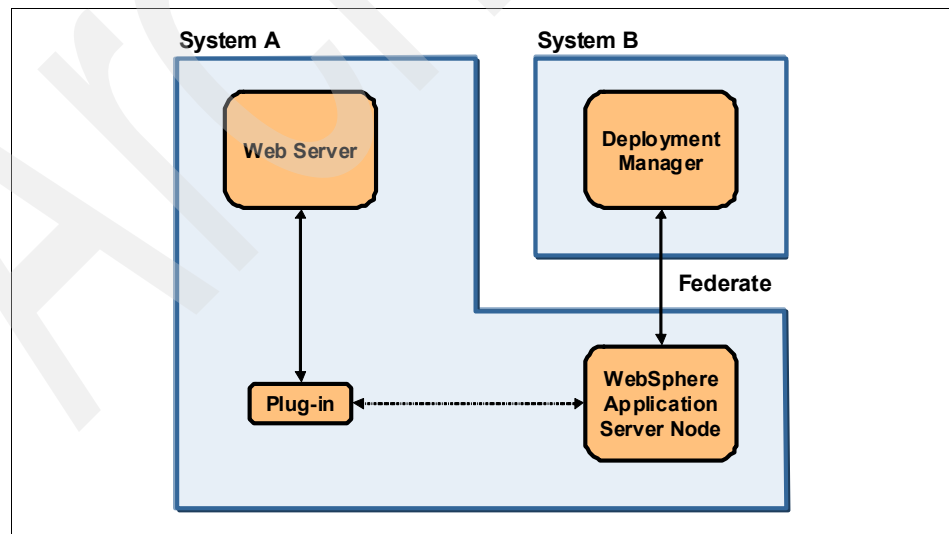


Figure 6-5 Web server installed locally on an application server system

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 system A.
2. Install the Web server plug-in on system 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 is 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 as follows:

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

During the installation, the following tasks are performed:

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

```
<plugin_root>/Plugins/bin/configure<web_server_name>
```

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

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

Note: For security reasons we do not suggest installing managed Web servers in the DMZ.

6.9 Planning for WebSphere Application Server

WebSphere Application Server V7,0 is a full product installation, not an upgrade installation. Consider the best installation method to use based on the number of systems and the complexity of the installations.

Review the documentation. The WebSphere Application Server Information Center contains planning topics for all WebSphere Application Server package that is tailored to each platform. This section gives you a high-level view at the planning tasks you need to perform.

Note: Information about installation considerations for WebSphere Application Server V7.0 for z/OS is included in 14.8, “Installing WebSphere Application Server for z/OS” on page 456.

Address the following items before starting the installation of WebSphere Application Server. These items introduced here are explained in more detail throughout this section of the book:

- ▶ File systems and directories
When installing WebSphere Application Server you are free to choose on which file systems you want to install the product and where you want to store your runtime environment, logs, and so on.
- ▶ Single install or multiple installations
The standard installation 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), and depending on your chosen topology, you might want to install multiple instances.
- ▶ Installation method
You have multiple options for the installation. Your choice is influenced by several factors, 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 Installation factory or perform a silent installation?), and if you are performing remote installations with unskilled personnel.
- ▶ Installing updates
To apply maintenance to WebSphere Application Server you need the IBM Update Installer 7.0.

- ▶ Slip install
Slip install means that custom installation packages (CIPs) are used to update and manage installed packages.
- ▶ Profile creation
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.
- ▶ Naming convention
Naming conventions can be an important management tool in large environments. Naming not only makes it easier to understand the environment but having a consistent naming convention in place is helpful if writing scripts.
- ▶ TCP/IP port assignments
Each type of server (deployment manager, node agent, application server, and so on) uses a series of TCP/IP ports. These ports must be unique on a system and must be managed properly. This is essential to avoid port conflicts if you are planning for multiple installations and profiles.
- ▶ Security considerations
Security for WebSphere falls into two basic categories:
 - Administrative security
 - Application securityDuring 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.
- ▶ IBM Support Assistant Agent
The IBM Support Assistant Agent is an optional feature that allows remote troubleshooting (such as remote system file transfer, data collections, and inventory report generation).

6.9.1 File systems and directories

WebSphere Application Server uses a default file system structure for storing the binary files and the runtime environment unless specified otherwise. Review the default directory structure and decide if this satisfies your needs. Refer to 6.4, “Planning for disk space and directories” on page 182 for a more information.

6.9.2 Single install or multiple installations

You can install WebSphere Application Server V7.0 multiple times on the same system in different directories, or you can install WebSphere Application Server V7.0 in parallel to older versions of WebSphere Application Server on the same system. These 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. You do not have to stop the other installations while applying fixes to a specific installation.

When you have a single installation of WebSphere Application Server V7.0, you can create multiple application server profiles. In this case, all profiles share the same product binaries. Therefore, you have to stop all application server JVMs of all profiles before installing fixes. When fixes are installed, they affect all profiles. Each profile has its own user data.

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

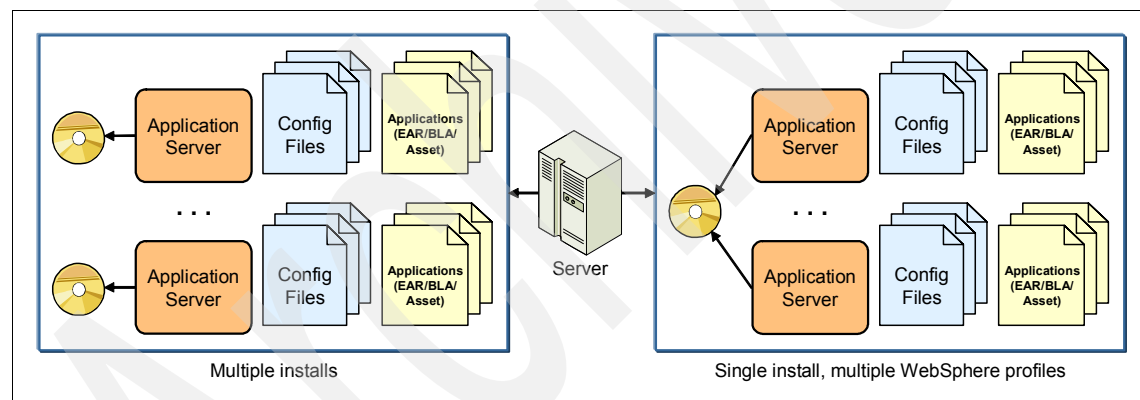


Figure 6-6 Stand-alone server installation options

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 system (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.

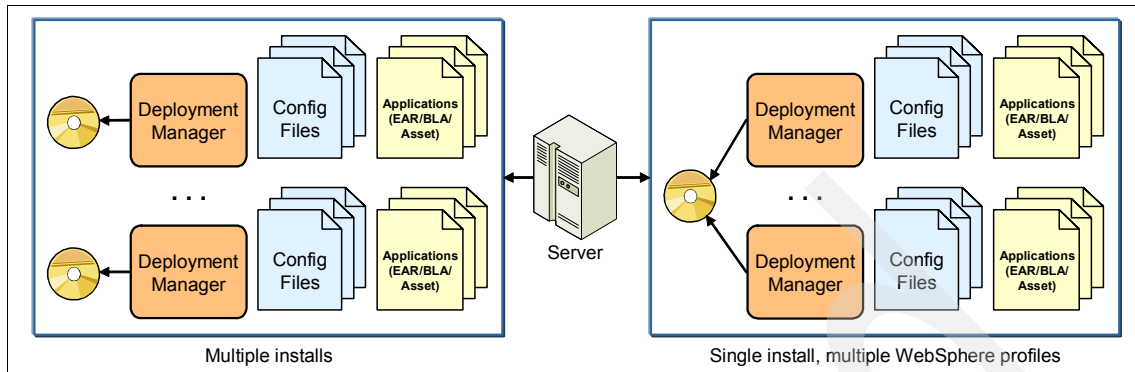


Figure 6-7 Deployment manager installation options

Using multiple installations and multiple profiles

Another possibility is the combination of multiple installation instances and multiple profiles. Figure 6-8 illustrates a Network Deployment environment where multiple runtime environments have been created using profiles.

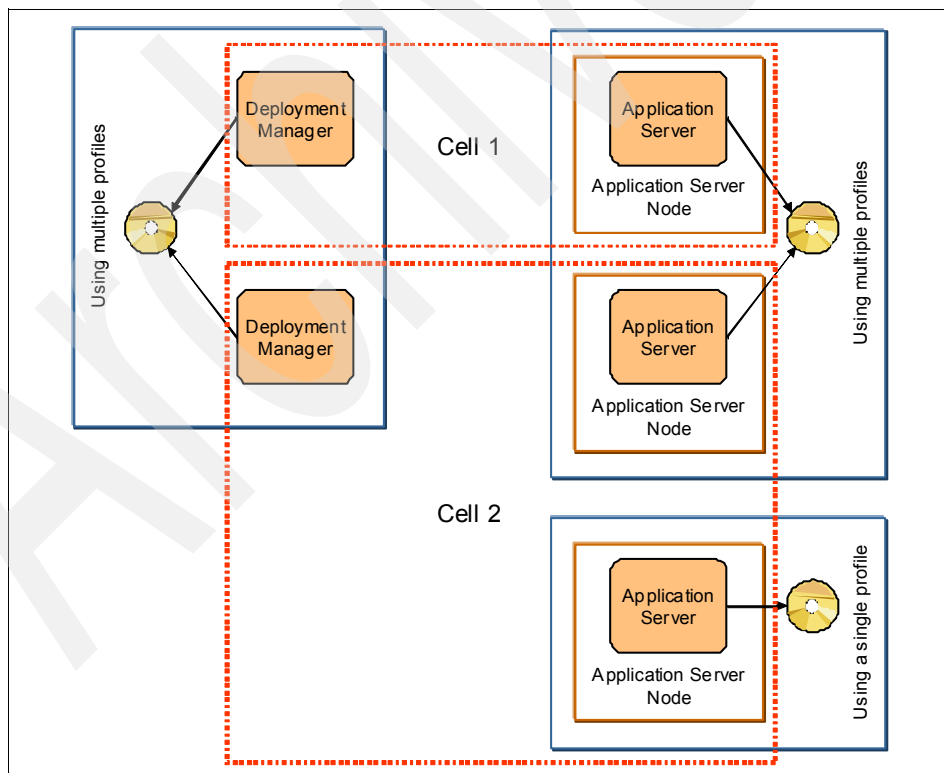


Figure 6-8 Cell configuration flexibility

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.

6.9.3 Installation method

Before starting installation activities, review the options you have for installing the code and select the option that best fits your needs. On distributed systems, you have several choices for installation:

- ▶ Graphical installation
- ▶ Silent installation
- ▶ Installation factory
- ▶ Centralized installation manager

Graphical installation

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. Note that there is a danger that the installation might fail or the product might not work as expected later on.

Silent installation

To install WebSphere Application Server V7.0 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. The silent installation approach offers the same options as the graphical installer. Providing the options in a response file provides various advantages over using the graphical installation wizard:

- ▶ The installation options can be planned and prepared in advance
- ▶ The prepared response file can be tested
- ▶ The installation is consistent and repeatable
- ▶ The installation is less fault-prone
- ▶ The installation is documented through the response file

Note: In case of problems with the silent installation, refer to the following Web page:

<http://www-01.ibm.com/support/docview.wss?uid=swg21255884>

Installation factory

The Installation Factory is an Eclipse-based tool that allows the creation of WebSphere Application Server installation packages in a reliable and repeatable way tailored to your needs.

The installation factory is part of the WebSphere deliverable on a separate media and download. Updates to the installation factory are available through the following Web page:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg24020213>

The Installation Factory can produce two types of packages:

- ▶ Customized Installation Packages (CIP)

A WebSphere Application Server CIP package includes a WebSphere Application Server product, product maintenance, profile customization, enterprise archives, other user files as well as user-defined scripting.

- ▶ Integrated Installation Packages (IIP)

An IIP can be used to install a full WebSphere software stack including Application Servers, feature pack, and other user files and might even contain multiple CIPs.

The Installation Factory allows you to create one installation package to install the full product stack you need to run your applications. Using the scripting interface you can ship and install components not related to the WebSphere installation process.

Note: The IBM WebSphere Installation Factory V7.0 is backwards-compatible with WebSphere Application Server V6.1 and allows you to create WebSphere V6.1 and WebSphere V7.0 installation packages.

Depending on the platform on which you are running the Installation Factory, you can build installation packages for operating systems other than the one on which the Installation Factory is running. The Installation Factory running on AIX, HP-UX, Linux, and Solaris operating systems can create installation packages for all supported platforms. The Installation Factory running on Windows can create installation packages for Windows and i5/OS®.

As you can see from the CIP build process flow in Figure 6-9, it takes multiple steps to build a CIP.

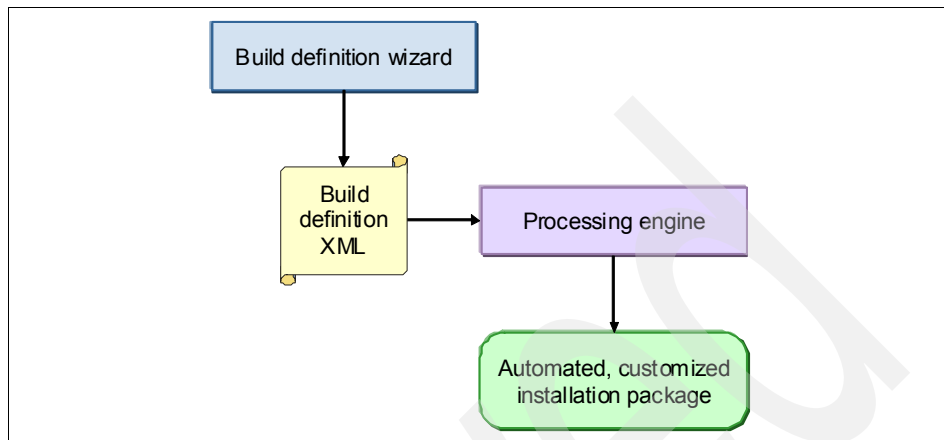


Figure 6-9 CIP build process

The CIP or IIP can be installed on the target system through two methods:

- ▶ Installation wizard
- ▶ Silent installer using a response file

The benefit of the Installation Factory is mainly in terms of installation time (fix packs, for example, are directly incorporated into the installation image) and in consistency and repeatability of the installations. This gives you a quick pay-back for the time required to build the CIP and IIP.

Centralized installation manager

Another product feature which can be used to install and update WebSphere Application Server Network Deployment installations is the CIM. More information about CIM can be found in 9.6.3, “Centralized installation manager (CIM)” on page 328.

6.9.4 Installing updates

For WebSphere maintenance tasks, you need to install the IBM Update Installer for WebSphere Software V7.0 on your system. The Update Installer is provided as a separate installer and needs to be installed on each system running WebSphere Application Server. The installer for the Update Installer can be found on the first supplemental media or through the IBM support download site at the following Web page:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg24020212>

Note: The IBM Update Installer for WebSphere Software is updated on a regular basis. It is advised to install the current version before installing any updates.

The IBM Update Installer for WebSphere Software V7.0 can be used to apply maintenance for multiple releases:

- ▶ V6.0.2 (from 6.0.2.21 onward)
- ▶ V6.1
- ▶ V7.0

Installation

The IBM Update Installer for WebSphere Software V7.0 can be installed through one of the following options:

- ▶ Graphical installation, using the installation wizard
- ▶ Silent installation, using a response file
- ▶ Centralized, using the CIM

Features

The IBM Update Installer for WebSphere Software V7.0 provides the following features:

- ▶ Installation and de-installation of code updates (fix packs or individual fixes)
- ▶ Prerequisite handling
- ▶ Logging capabilities
- ▶ Recovery functions

Fix handling

When dealing with fixes and fix packs, it is essential to plan for and to implement a proper policy on where to archive fixes and fix packs. This is best implemented by building a central repository. Having such a repository available allows quick configuration of identical systems and brings systems to the identical software level.

Note: The CIM provides a centralized repository for fix and fix pack files.

6.9.5 Profile creation

The installation process of WebSphere Application Server provides the product files required to create a runtime environment. However, the actual runtime is defined through the usage of profiles.

The product binaries remain unchanged after installation until you install maintenance. Because all profiles of an installation share the same binaries, all server processes of all profiles of an installation use the updated level of the binaries after installing the service.

Profiles can be created either during the installation process of the product or they can be created any time after the installation process is finished.

Tip: It is suggested not to create profiles during the installation. Always finish the installation and upgrade the product to the latest fix pack level first before creating the profiles.

Before you start creating the profiles consider the following questions:

- ▶ What profile types will you need?
See “Profile types” on page 206.
- ▶ How to create the profiles?
See “Creating profiles during the installation” on page 208 and “Creating additional profiles” on page 209.
- ▶ Where to store the profile configuration files?
See “Profile location” on page 219.

Profiles can be stored under the installation root for WebSphere Application Server, or in any location you choose depending on your planning for disk and directories as outlined in 6.4, “Planning for disk space and directories” on page 182.

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 as follows:

- ▶ Management profile with a deployment manager server

The deployment manager profile creates an application server named dmgr and deploys the Integrated Solutions Console (recommended). The deployment manager provides a centralized administration interface for multiple nodes with all attached servers in a single cell. The deployment manager profile is the basis for clustering, high availability, fail-over, and so on. When using manageprofiles you have two choices to create a deployment manager profile.

- Specify `-profileTemplate <app_server_root>/profileTemplates/management` and `-serverType DEPLOYMENT_MANAGER`.
- Specify `-profileTemplate <app_server_root>/profileTemplates/dmgr`.

- ▶ Management profile with an administrative agent server

The administrative agent profile creates the application server named adminagent and deploys the Integrated Solutions Console (recommended).

The administrative agent provides a centralized administration interface for multiple unfederated application server profiles on the same system without providing any support for clustering, high availability, fail-over, and so on.

When using manageprofiles specify `-profileTemplate <app_server_root>/profileTemplates/management` and `-serverType ADMIN_AGENT`.

- ▶ Management profile with a job manager server

The job manager profile creates the application server named jobmgr and deploys the Integrated Solutions Console (recommended).

The job manager provides a centralized interface for the following tasks:

- Administering multiple unfederated application server profiles through the administrative agent,
- Deployment manager profiles
- Asynchronous job submissions

No additional clustering, high availability, fail-over, and so on, capabilities are provided.

When using manageprofiles specify `-profileTemplate <app_server_root>/profileTemplates/management` and `-serverType JOB_MANAGER`.

► Application server profiles

An application server profile creates an application server and deploys the following applications:

- Default applications (optional)
- Sample applications (optional)
- Integrated Solutions Console (recommended)

The default name of the application server is `server1`, but this can be overridden through the `-serverName` parameter in the `manageprofiles` command, or when using the advanced profile creation option in the profile management tool. The application server can either run as a stand-alone application server or be federated to a deployment manager.

When using `manageprofiles`, specify `-profileTemplate <app_server_root>/profileTemplates/default` to create an application server profile.

► Custom profiles

The custom profile creates an empty node in a cell. The only application server created is an application server named `nodeagent`. No applications are deployed. The node can be federated with a deployment manager either during profile creation or at a later time.

When using `manageprofiles` specify `-profileTemplate <app_server_root>/profileTemplates/managed`.

► Cell profiles

A cell profile creates a deployment manager profile and a federated application server profile at the same time on the system using default naming conventions in certain areas. The result of this profile creation is a fully functional cell. The following applications can be deployed to the federated application server:

- Default applications (optional)
- Sample applications (optional)

On the deployment manager profile it deploys the Integrated Solutions Console (recommended). From the functional perspective it is the same as creating a management profile with a deployment manager server and an application server profile, and then federating the application server profile to the deployment manager.

When using `manageprofiles`, two portions of the cell must be created. Specify `-profileTemplate <app_server_root>/profileTemplates/cell/dmgr` to create the deployment manager portion of the profile, and specify `-profileTemplate <app_server_root>/profileTemplates/cell/default` for the cell node portion of the profile.

► Secure proxy profile

A secure proxy profile creates a proxy server that is supposed to run in the DMZ and supports HTTP and SIP protocol and the corresponding secure version of the protocol.

The default name of the application server is proxy1 but this can be overridden through the `-serverName` parameter in the `manageprofiles` command, or when using the advanced profile creation option in the profile management tool.

When using `manageprofiles`, specify `-profileTemplate <app_server_root>/profileTemplates/secureproxy` to create a secure proxy profile.

Table 6-1 shows a list of the available profile types per WebSphere Application Server edition.

Table 6-1 Available profile types per WebSphere Application Server edition

Product	WebSphere profiles available
WebSphere Application Server Express V7.0	<ul style="list-style-type: none">► Management profile with an administrative agent server► Application server profile
WebSphere Application Server V7.0 (single server)	<ul style="list-style-type: none">► Management profile with an administrative agent server► Application server profile.
WebSphere Application Server Network Deployment V7.0	<ul style="list-style-type: none">► Management profile with a deployment manager server► Management profile with an administrative agent server► Management profile with a job manager server► Application server profile► Cell profile► Custom profile► Secure proxy profile

Creating profiles during the installation

On distributed platforms, profiles can be created during the installation of the product or afterwards using either the Profile Management Tool or the `manageprofiles` command.

Note: In order to have an operational product you must create a profile.

When profiles are created during the installation of the product, they are created using typical settings using default naming conventions, default port assignments, default location (which is `<app_server_root>/profiles` directory), and so on. Be careful when creating profiles during the installation as it might violate some of your planned approach.

Express and Base installation

The installation procedure for WebSphere Application Server V7.0 Express and Base installs the core product files and optionally creates either an application server profile or an administrative agent profile. After the installation, you can create additional application server profiles using the profile management tool. 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 V7.0 installs the core product files and gives you the choice of creating any of the available profile types or no profile at all. The network deployment installation procedure gives you the option of whether a repository for the CIM should be created. If you select to create the CIM repository you can define the location of the repository and select that the current installation package should be populated to this repository.

After the installation, you can create additional profiles using the Profile Management Tool or the `manage profiles` command.

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 z/OS Profile Management Tool (zPMT) available in the Application Server Toolkit. For more information in z/OS installation and configuration steps refer to 14.8, “Installing WebSphere Application Server for z/OS” on page 456.

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 (PMT) and choosing the advanced path or the `manage profiles` command to create the profiles enables gives you more flexibility in the options you take.

Tips:

1. Use `manageprofiles.bat (sh)` to create your production profiles. Using the scripting approach allows reuse and easier documentation.
2. To determine the parameters `manageprofiles.bat (sh)` requires for a specific profile type, run `manageprofiles.bat (sh) -create -templatePath <templatePath> -help`. For example, on Windows run:

```
.manageprofiles.bat -create -templatePath  
\WebSphere\WAS70\profileTemplates\management -help
```

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 PMT.

Table 6-2 Deployment manager profile options

Typical settings	Advanced options
The administrative console is deployed by default.	You have the option to deploy the administrative console (recommended and preselected)
The profile name is Dmgrxx by default, where xx is 01 for the first deployment manager profile and increments for each one created. The profile is stored in <app_server_root>/profiles/Dmgrxx.	You can specify the profile name and its location.
The profile is not marked as the default profile.	You can choose whether to make this the default profile. (Commands run without specifying a profile are run against the default profile.)
The cell name is <host>Cellxx. The node name is <host>CellManagerxx. Host name defaults to your system's DNS host name.	You can specify the node, host, and cell names.
You can select whether to enable administrative security or not. By default Enable administrative security is preselected. If you select yes, you have to specify a user name and password that is given administrative authority.	
Creates a new default personal certificate for this profile using the DN: cn=<hostname>,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for the new certificate being created or to import an existing default personal certificate from an existing keystore
Creates a new root signer certificate for this profile using the DN: cn=<hostname>,ou=Root Certificate,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for new root signer certificate being created or to import an existing root signing certificate from an existing keystore
Default expiration date for the personal certificate is 1 year.	Allows you to enter the expiration period
Default expiration date for the signer certificate is 15 years.	Allows you to enter the expiration period
Keystore password is <i>WebAS</i>	Allows you to enter a unique password for the keystore
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.

Typical settings	Advanced options
(Windows) The deployment manager is run as a service using a local system account and startup type Automatic.	(Windows) You can choose whether the deployment manager will run as a service, under which account the service runs and what startup type is used
(Linux) The deployment manager will not run as a Linux service.	(Linux) You can create a Linux service and specify the user name from which the service runs.

Administrative agent profile options

Table 6-3 shows a summary of the options available when creating a profile for an administrative agent. The options depend on whether you take the typical or advanced path through the Profile Management Tool.

Table 6-3 Administrative agent profile options

Typical settings	Advanced options
The administrative console is deployed by default.	You have the option to deploy the administrative console (recommended and preselected).
The profile name is AdminAgentxx by default, where xx is 01 for the first administrative agent profile and increments for each one created. The profile is stored in <app_server_root>/profiles/AdminAgentxx.	You can specify the profile name and its location.
The profile is not marked as the default profile.	You can choose whether to make this the default profile. (Commands run without specifying a profile are run against the default profile.)
The cell name is <host>AACellxx. The node name is <host>AANodexx. Host name defaults to your system's DNS host name.	You can specify the node, host, and cell names.
You can select whether to enable administrative security or not. By default Enable administrative security is preselected. If you select yes, you have to specify a user name and password that is given administrative authority.	
Creates a new default personal certificate for this profile using the DN: cn=<hostname>,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for the new certificate being created or to import an existing default personal certificate from an existing keystore
Creates a new root signer certificate for this profile using the DN: cn=<hostname>,ou=Root Certificate,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for new root signer certificate being created or to import an existing root signing certificate from an existing keystore

Typical settings	Advanced options
Default expiration date for the personal certificate is 1 year.	Allows you to enter the expiration period
Default expiration date for the signer certificate is 15 years.	Allows you the enter the expiration period
Keystore password is WebAS	Allows you to enter a unique password for the keystore
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 is run as a service using a local system account and startup type Automatic. (Linux) The deployment manager will not run as a Linux service.	(Windows) You can choose whether the deployment manager will run as a service, under which account the service runs and what startup type is used (Linux) You can create a Linux service and specify the user name from which the service runs.

Job manager profile options

Table 6-4 shows a summary of the options available when creating a profile for a job manager. The options depend on whether you take the typical or advanced path through the Profile Management Tool.

Table 6-4 Job manager profile options

Typical settings	Advanced options
The administrative console is deployed by default.	You have the option to deploy the administrative console (recommended and preselected).
The profile name is JobMgrxx by default, where xx is 01 for the first administrative agent profile and increments for each one created. The profile is stored in <app_server_root>/profiles/JobMgrxx.	You can specify the profile name and its location.
The profile is not marked as the default profile.	You can choose whether to make this the default profile. (Commands run without specifying a profile are run against the default profile.)
The cell name is <host>JobMgrCellxx. The node name is <host>JobMgrxx. Host name defaults to your system's DNS host name.	You can specify the node, host, and cell names.

Typical settings	Advanced options
<p>You can select whether to enable administrative security or not. By default Enable administrative security is preselected. If you select yes, you have to specify a user name and password that is given administrative authority.</p>	
<p>Creates a new default personal certificate for this profile using the DN: cn=<hostname>,ou=<cellname>,ou=<nodename>,o=IBM,c=US</p>	<p>Allows you to enter the DN for the new certificate being created or to import an existing default personal certificate from an existing keystore</p>
<p>Creates a new root signer certificate for this profile using the DN: cn=<hostname>,ou=Root Certificate,ou=<cellname>,ou=<nodename>,o=IBM,c=US</p>	<p>Allows you to enter the DN for new root signer certificate being created or to import an existing root signing certificate from an existing keystore</p>
<p>Default expiration date for the personal certificate is 1 year.</p>	<p>Allows you to enter the expiration period</p>
<p>Default expiration date for the signer certificate is 15 years.</p>	<p>Allows you the enter the expiration period</p>
<p>Keystore password is WebAS</p>	<p>Allows you to enter a unique password for the keystore</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 (unique to the installation), use the basic defaults, or select port numbers manually.</p>
<p>(Windows) The deployment manager is run as a service using a local system account and startup type Automatic.</p> <p>(Linux) The deployment manager will not run as a Linux service.</p>	<p>(Windows) You can choose whether the deployment manager will run as a service, under which account the service runs and what startup type is used</p> <p>(Linux) You can create a Linux service and specify the user name from which the service runs.</p>

Application server profile options (non-Express V7.0)

Table 6-5 shows a summary of the options available when creating a profile for an application server. The options depend on whether you take the typical or advanced path through the Profile Management Tool.

Table 6-5 Application server profile options: V7.0

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 and preselected), the default application (preselected), and the sample applications (if installed).
The profile name is AppSrvxx by default, where xx is 01 for the first application server profile and increments for each one created. The profile is stored in <app_server_root>/profiles/AppSrvxx.	You can specify profile name and its location.
The profile is not marked as the default profile.	You can choose whether to make this the default profile. (Commands run without specifying a profile are 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 server name is server1. The host name defaults to your system's DNS host name.	You can specify the node name, server name and host name.
You can select whether to enable administrative security or not. By default Enable administrative security is preselected. If you select yes, you have to specify a user name and password that is given administrative authority.	
Creates a new default personal certificate for this profile using the DN: cn=<hostname>,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for the new certificate being created or to import an existing default personal certificate from an existing keystore
Creates a new root signer certificate for this profile using the DN: cn=<hostname>,ou=Root Certificate,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for new root signer certificate being created or to import an existing root signing certificate from an existing keystore
Default expiration date for the personal certificate is 1 year.	Allows you to enter the expiration period
Default expiration date for the signer certificate is 15 years.	Allows you the enter the expiration period

Typical settings	Advanced options
Keystore password is WebAS	Allows you to enter a unique password for the keystore
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 application server is run as a service using a local system account and startup type Automatic. (Linux) The application server will not run as a Linux service.	(Windows) You can choose whether the application server will run as a service, under which account the service runs and what startup type is used (Linux) You can create a Linux service and specify the user name from which the service runs.
Does not create a Web server definition.	Enables you to define an external Web server to the configuration.

Application server profile options (Express V7.0)

Table 6-6 shows a summary of the options available when creating a profile for an application server in WebSphere Application Server Express V7.0. The options depend on whether you take the typical or advanced path through the Profile Management Tool.

Table 6-6 Application server profile options: V7.0

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 and preselected), the default application (preselected), and the sample applications (if installed).
The profile name is AppSrvxx by default, where xx is 01 for the first application server profile and increments for each one created. The profile is stored in <app_server_root>/profiles/AppSrvxx.	You can specify profile name and its location.
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 server name is server1. The host name defaults to your system's DNS host name.	You can specify the node name, server name and host name.

Typical settings	Advanced options
You can select whether to enable administrative security or not. By default Enable administrative security is preselected. If you select yes, you have to specify a user name and password that is given administrative authority.	
Creates a new default personal certificate for this profile using the DN: cn=<hostname>,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for the new certificate being created or to import an existing default personal certificate from an existing keystore
Creates a new root signer certificate for this profile using the DN: cn=<hostname>,ou=Root Certificate,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for new root signer certificate being created or to import an existing root signing certificate from an existing keystore
Default expiration date for the personal certificate is 1 year.	Allows you to enter the expiration period
Default expiration date for the signer certificate is 15 years.	Allows you the enter the expiration period
Keystore password is WebAS	Allows you to enter a unique password for the keystore
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 application server is run as a service using a local system account and startup type Automatic. (Linux) The application server will not run as a Linux service.	(Windows) You can choose whether the application server will run as a service, under which account the service runs and what startup type is used (Linux) You can create a Linux service and specify the user name from which the service runs.
Does not create a Web server definition.	Enables you to define an external Web server to the configuration.

Cell profile options

Table 6-7 on page 217 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 6-7 Cell profile options

Typical settings	Advanced options
The administrative console and default application are deployed by default. The sample applications are not deployed.	You have the option to deploy the administrative console (recommended and preselected), the default application (preselected), and the sample applications (if installed).
The profile name for the deployment manager is Dmgrxx by default, where xx is 01 for the first deployment manager profile and increments for each one created.	You can specify the profile name
The profile name for the federated application server and node is AppSrvxx by default, where xx is 01 for the first application server profile and increments for each one created.	You can specify the profile name
Directory <app_server_root>/profiles is used as <profile_root>. The profiles are created in <profile_root>/<profilename>	You can specify the <profile_root> directory. The profiles are created in <profile_root>/<profilename>
Neither profile is made the default profile.	You can choose to make the deployment manager profile the default profile.
The cell name is <host>Cellxx. The node name for the deployment manager is <host>CellManagerxx. The node name for the application server is <host>Nodexx . The host name defaults to your system's DNS host name.	You can specify the cell name, the host name, and the profile names for both profiles.
You can select whether to enable administrative security or not. By default Enable administrative security is preselected. If you select yes, you have to specify a user name and password that is given administrative authority.	
Creates a new default personal certificate for this profile using the DN: cn=<hostname>,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for the new certificate being created or to import an existing default personal certificate from an existing keystore
Creates a new root signer certificate for this profile using the DN: cn=<hostname>,ou=Root Certificate,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for new root signer certificate being created or to import an existing root signing certificate from an existing keystore
Default expiration date for the personal certificate is 1 year.	Allows you to enter the expiration period

Typical settings	Advanced options
Default expiration date for the signer certificate is 15 years.	Allows you the enter the expiration period
Keystore password is WebAS	Allows you to enter a unique password for the keystore
TCP/IP ports will default to a set of ports not used by any profiles in this WebSphere installation instance.	You can use the recommended ports for each profile (unique to the installation), use the basic defaults, or select port numbers manually.
(Windows) The application server is run as a service using a local system account and startup type Automatic. (Linux) The product is not selected to run as a Linux service	(Windows) You can choose whether the application server will run as a service, under which account the service runs and what startup type is used (Linux) You can create a Linux service and specify the user name from which the service runs.
Does not create a Web server definition.	Enables you to define an external Web server to the configuration.

Custom profile options

Table 6-8 shows a summary of the options available when creating a custom profile.

Table 6-8 Custom profile options

Typical settings	Advanced options
The profile name is Customxx. The profile is stored in <app_server_root>/profiles/Customxx. By default, it is not considered the default profile.	You can specify profile name and location. You can also specify if you want this to be the default profile.
The profile is not selected to be the default profile	You can select this profile to be the default profile.
The node name is <host>Nodexx. The host name defaults to your system's DNS host name.	You can specify the node name and host name.
You can choose to federate the node later, or during the profile creation process. If you want to 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.	
Creates a new default personal certificate for this profile using the DN: cn=<hostname>,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for the new certificate being created or to import an existing default personal certificate from an existing keystore

Typical settings	Advanced options
Creates a new root signer certificate for this profile using the DN: cn=<hostname>,ou=Root Certificate,ou=<cellname>,ou=<nodename>,o=IBM,c=US	Allows you to enter the DN for new root signer certificate being created or to import an existing root signing certificate from an existing keystore
Default expiration date for the personal certificate is 1 year.	Allows you to enter the expiration period
Default expiration date for the signer certificate is 15 years.	Allows you the enter the expiration period
Keystore password is WebAS	Allows you to enter a unique password for the keystore
TCP/IP ports will default to a set of ports not used by any profiles in this WebSphere installation instance.	You can use the recommended ports for each profile (unique to the installation), use the basic defaults, or select port numbers manually.

Starting the Profile Management Tool

After the installation of Base, Express, or Network Deployment V7.0 on distributed systems, you can start the Profile Management Tool in the following ways:

- ▶ From the First Steps window.
- ▶ On Windows systems, from the Start menu (**Start → Programs → IBM WebSphere → Application Server [Network Deployment V7.0] → Profile Management Tool**)
- ▶ By executing the **pmt.bat (sh)** command.

For operating systems such as AIX 5L or Linux, the command is in the <app_server_root>/bin/ProfileManagement directory.

For the Windows platform, the command is in the <app_server_root>\bin\ProfileManagement directory.

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

Profile location

Profiles created as a part of the installation or using the typical settings are automatically placed in the <app_server_root>/profiles directory. When you create profiles after the installation, you can designate the location where the profiles are stored. Refer to 6.4, “Planning for disk space and directories” on page 182 for considerations about disk space and directory planning.

6.9.6 Naming convention

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

- ▶ To provide guidance during setup and configuration
- ▶ To quickly narrow down the source of any issue that arises

Naming the WebSphere Application Server infrastructure artifacts, such as cells, nodes, application servers, and so on should follow the company's normal naming conventions as far as possible. Some considerations to be taken into account when developing the key concepts for the site during the installation planning are as follows:

- ▶ 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 is 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.

► 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 is in the following 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. You can have multiple nodes on a system. Keep this in mind when planning your WebSphere names.

Naming conventions for nodes often include the physical machine name where they are running, such as NodexxAP010 if the server name is ServerAP010 and a running number to enable growth if additional nodes need to be created.

► Naming application servers

In stand-alone environments, the default server name is “server1,” but can be overridden through the `manageprofiles.bat(sh)` command, or using the advanced profile creation options in the PMT.

Note: When you federate multiple stand-alone application servers that were created using the default naming schema to a cell, you will have a unique combination of node name and server1, thus ending up with multiple server1s in the cell.

In a distributed server environment, it is more likely that new application servers are created on a federated node using the Integrated Solutions Console or another 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.

Note: 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 Integrated Solutions 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 in WebSphere Application Server:

- 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.

Tip: Avoid any language-specific characters in names.

6.9.7 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 for a single machine can involve having multiple cells, nodes, and server profiles in a single process space. Each WebSphere process requires exclusive usage of several ports and knowledge of certain other ports for communication with other WebSphere processes.

To simplify the installation and provide transparency to the ports use, the following approach is reliable and reduces the complexity of such a scenario considerably:

- ▶ With the network administration, discuss and decide on a fixed range of continuous ports for exclusive use for the WebSphere Application Server installation.
- ▶ Draw the overall 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. Make sure to document the ports in an appropriate way.

Tip: The same spreadsheet also can serve for the server names, process names, user IDs, and so forth.

When creating your profiles with the PMT using the advanced options path, you can set the ports for your profile as needed. The PMT identifies the ports used in the same installation on that system and the ports that are currently in use and will suggest unique ports to use.

The `manageprofiles.bat (sh)` command allows you to control the port numbers through the `-portsFile` and `-startingPort` parameters.

For a list of the ports used by WebSphere Application Server and their default settings, refer to the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.migration.nd.doc/info/ae/ae/rmig_portnumber.html

6.9.8 Security considerations

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.

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 are available when you need them.
- If default certificates or dummy key ring files are provided with any of the products you plan to install, replace them with your own certificates.
- If you are using self-signed certificates, plan your signer structure carefully and exchange signer certificates if necessary.

Note: In WebSphere Application Server V7.0 signer and personal certificates can be either created or imported during profile creation. If you have new certificates created you can choose the correct DN during profile creation.

► Network and physical security

- Usually one or more firewalls are part of the topology. After determining 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. For more information, see the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.installation.nd.doc/info/ae/ae/tins_install.html

Although non-root installation is also supported, some limitations apply. For more information, see the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.installation.nd.doc/info/ae/ae/cins_nonroot.html

- If there is a policy on password expiration, it should be well known so as to avoid disruption on the service (password expiration of root, 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 WebSphere Application Server V7.0 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 7.0 was enhanced and works almost the same as installing as a root user does in previous versions. There are some specifics you have to consider which are documented at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.installation.nd.doc/info/ae/ae/cins_nonroot.html

There are limitations of which you need to be aware. Refer to the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.installation.nd.doc/info/ae/ae/cins_nonroot.html#cins_nonroot_nonroot_install_limitations

Secure administration tasks

WebSphere Application Server provides a mechanism to secure the administrative interfaces. With WebSphere Application Server V7.0, 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). 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 are created in the repository and assigned the Administrator role. This ID can be used to access the administration tools and to add additional user IDs for administration. When you enable security during profile creation, LTPA is used as the authentication mechanism and the federated repository realm used is used as account repository.

On distributed systems, an XML file-based user repository is created and populated with the administrator ID. This XML file-based system can be federated with other repository types to form an overall repository system. If you do not want to use the file-based repository, do not enable administrative security during profile creation or change it afterwards. 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.

6.10 IBM Support Assistant

IBM Support Assistant V4.0 (ISA) is a free tool that is intended to help you research, analyze, and resolve problems using various support features and problem determination tools. Using IBM Support Assistant you should be able to determine the cause for most your problems faster and find solutions in a shorter time. Therefore, the availability of your installation is increased. ISA provides you many different tools for problem determination and materials collections. It allows you to organize and transfer your troubleshooting efforts between members of your team or to IBM for further support. The IBM Support Assistant consists of the following three distinct features:

- ▶ IBM Support Assistant Workbench

The IBM Support Assistant Workbench, or simply the Workbench, is the client-facing application that you can download and install on your workstation. It enables you to use all the troubleshooting features of the Support Assistant such as Search, Product Information, Data Collection, Managing Service Requests, and Guided Troubleshooting. The Workbench can only perform these functions locally. For example, on the system where it is installed (with the exception of the Portable Collector). The following Web page lists and describes the tools available in ISA:

<http://www-01.ibm.com/support/docview.wss?rs=3455&uid=swg27013116>

- ▶ IBM Support Assistant Agent

The IBM Support Assistant Agent, or simply the Agent, is the software that needs to be installed on every system that you need to troubleshoot remotely. After an Agent is installed on a system, it registers with the Agent Manager and you can use the Workbench to communicate with the Agent and use features such as remote system file transfer, data collections, and inventory report generation on the remote machine.

- ▶ IBM Support Assistant Agent Manager

The IBM Support Assistant Agent Manager, or simply the Agent Manager, needs to be installed only once in your network. The Agent Manager provides a central location where information about all available Agents is stored and acts as the certificate authority. For the remote troubleshooting to work, all Agent and Workbench instances register with this Agent Manager. Any time a Support Assistant Workbench needs to perform remote functions, it authenticates with the Agent Manager and gets a list of the available Agents. After this, the Workbench can communicate directly with the Agents.

Note: For installation instructions and more details about IBM Support Assistant, see the following Web page:

<http://www.ibm.com/software/support/isa/>

6.11 Summary: Installation checklist

Table 6-9 provides a summary of items to consider as you plan and additional resources that can help you.

Table 6-9 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 profile files (6.4, "Planning for disk space and directories" on page 182).
Develop a naming convention that includes system naming and WebSphere Application Server component naming.
Develop a strategy for managing certificates in your environment. Personal certificates as well as signer certificates.
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.
Plan for the Web server and Web server plug-in installation. Determine if the Web server is a managed or unmanaged server and note the implications. Create a strategy for generating and propagating the Web server plug-in configuration file.

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. See the following Web page:

<http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-mp&topic=>

Archived

Performance, scalability, and high availability

This chapter discusses items to consider when implementing WebSphere Application Server V7.0 so that the environment performs well, is highly scalable, and provides high availability.

The nature of these three non-functional requirements make them related to each other. For example: to increase the performance of your environment you need to add additional resources. To be able to add additional resources efficiently you need a scalable design and workload management to spread the requests across all available components. By adding additional resources in most cases you invent redundancy which is a prerequisite for high availability.

This chapter contains the following sections:

- ▶ What is new in V7.0
- ▶ Scalability
- ▶ Performance
- ▶ Workload management
- ▶ High availability
- ▶ Caching
- ▶ Session management
- ▶ Data replication service
- ▶ WebSphere performance tools
- ▶ Summary: Checklist for performance

7.1 What is new in V7.0

IBM WebSphere Application Server V7.0 provides several enhancements and improvements related to performance, scalability, and high availability.

The main enhancements can be found in the following areas, described in the following sections:

- ▶ “Runtime provisioning” on page 230
- ▶ “Java SE 6” on page 230
- ▶ “DMZ secure proxy” on page 231
- ▶ “Flexible management” on page 231

7.1.1 Runtime provisioning

Runtime provisioning is a feature of WebSphere Application Server V7.0, which makes sure that only those runtime features of an application server are activated that are needed to run the deployed applications. This results in reduced startup time and a reduced memory footprint. For more details about runtime provisioning refer to 3.1.14, “Intelligent runtime provisioning” on page 72.

7.1.2 Java SE 6

WebSphere Application Server V7.0 ships with Java SE 6. All platforms that ship with an IBM software development kit for Java (SDK) will benefit from the enhancements incorporated in the SDK. The most important enhancements related to performance are as follows:

- ▶ Improved startup performance

A major reason behind the enhanced startup performance is the improved class sharing in a cache. Class sharing in a cache was introduced in version 5 of IBM SDK. This technique was improved, and in version 6 of the SDK the cache can now be persisted. This means the cache can also survive a reboot of the system.

- ▶ Smaller memory footprint

As the Java class cache is shared across all running application servers, the memory footprint for each running application server will be reduced. It can be expected that especially large scale installations will benefit from this feature. In particular, this will reduce the footprint on the z/OS platform, with its structure of multiple Java processes and multiple Java Virtual Machines (JVM).

- ▶ Improved 64-bit performance

When using the 64-bit version of Java SE V6 the usage of compressed references provides significant improvements in memory footprint and performance. By using this technique, heap sizes up to 25 GB are possible while the overhead of a pure 64-bit implementation is reduced.

- ▶ Higher garbage collection (GC) throughput

Although the core GC technologies are the same as in version 5 of the SDK, a new implementation of the GC component provides an improved footprint and faster class loader performance. Additional performance improvements like hierarchical scanning were implemented for the gencon policy.

7.1.3 DMZ secure proxy

The DMZ secure proxy server can be placed in the architecture to replace a Web server with a plug-in. Performance tests have shown that a performance benefit of up to 15% can be achieved using the DMZ secure proxy, compared to the Web server plus plug-in configuration.

Additionally it can be expected that the DMZ secure proxy scales better than the traditional Web server plus plug-in configuration. Thus you might end up with fewer servers. You should perform proper testing in your environment to figure out what configuration performs better in your specific environment.

7.1.4 Flexible management

Under the aspect of flexible management, two types of servers: administrative agent and job manager, were introduced in WebSphere Application Server V7.0. The main purpose of these server types is to increase administrative scalability.

For more information about flexible management refer to 3.2.4, “Flexible management” on page 78.

7.2 Scalability

Scalability, in general terms, means adding hardware and software resources to improve performance. However, adding more hardware might not necessarily improve the performance of your environment if your systems are not tuned properly or if your application is not scalable. 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.

Note: In order to be scalable you need a scalable application. If an application is not designed to be scalable, the scalability options are limited.

7.2.1 Scaling overview

Consider additional hardware resources as a step for improving system performance. There are two ways to improve performance when adding hardware:

- ▶ Vertical scaling

Vertical scaling means increasing the throughput by adding resources inside the server and operating system image to handle more requests in parallel. From a hardware perspective, we can, for example, increase the number of processors for a server. By upgrading the system hardware with additional processors, the server can potentially gain a higher throughput. This concept is relatively easy to implement and can apply to any server in an environment. Examine vertical scaling at every potential bottleneck in your architecture.

- ▶ Horizontal scaling

Horizontal scaling means adding additional separated resources to handle additional load. This applies to multi-tier scenarios and can require the use of additional components like a load balancer that sits in front of the duplicated servers and makes them appear as a single instance.

Note: Adding resources usually affects the dynamics of the system and can potentially shift the bottleneck from one resource to another. Therefore it is important to have a balanced system.

In a single tier scenario, the Web application and database servers are all running on the same system. Creating a cluster and spreading application servers across systems should improve the throughput. But at the same time, additional systems introduce new communication traffic and load to the database server. Ask yourself the following questions:

- ▶ How much network bandwidth will this server configuration consume?
- ▶ What will the performance improvement by adding more systems be?

Because of these questions, we recommend that you always perform a scalability test to identify how well the site is performing after a hardware change. Throughput and response time can be measured to ensure that the result meets your expectations.

In addition, be aware that the law of diminishing returns plays a role when using either vertical or horizontal scaling technique. The law of diminishing returns is an economics principle that states that if one factor of production is increased while all 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.

Note: The benefit you get by adding resources depends on the hardware and operating system platform you are using.

7.2.2 Scaling the system

Examine the entire application environment to identify potential bottlenecks. Throwing additional resources into the environment without fully understanding what those resources are addressing can potentially worsen performance, maintainability and scalability. It is necessary to know the application environment, end-to-end, to identify the bottlenecks and ensure the appropriate response is timely.

Network

When scaling at the network layer, such as with firewalls or switches, the most common solution is vertical scaling. Network devices have processing capacity and use memory much like any other hardware resource. Adding additional hardware resources to a network device will increase the throughput of that device, which positively impacts the scaling of that device. For example, moving from 100 MBit to 1 GBit connections or even higher can significantly increase performance at the network layer.

HTTP server

When scaling at the Web server layer, the most typical 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 and 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.

Note: IBM HTTP Server for z/OS offers the unique feature of a scalable mode. This allows the server to start additional instances of itself to offer vertical scalability if the performance goals are not met.

DMZ secure server

The DMZ secure proxy provides horizontal and vertical scaling capabilities besides the scaling activities on a per server basis. When scaling vertically, make sure that you have sufficient resources. Also, be aware that this provides only limited high availability. In any scaling scenario you need an IP sprayer like the Edge Components to spread the incoming traffic across all proxy servers.

WebSphere containers

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 need to be identical in terms of physical resources.

Default messaging

With the messaging engine and the service integration bus, WebSphere offers the ability to scale messaging resources more efficiently. Using a cluster as a service integration bus member, you can create partitioned destinations. This enables a single logical queue to 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 may or may not be significant, depending on the nature of the application.

Multiple messaging engines for a cluster bus member is not the default setting. For workload management, you must take steps to add additional messaging engines. To make sure that the messaging engines are really running on different servers you must configure proper high availability policies. WebSphere Application Server V7 features a wizard for high availability policy setup which can be used for most topologies.

Data layer

At the data services layer, it is most common to implement vertical scaling. Adding multiple copies of a 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 make use of higher numbers of processors or more memory. Most database management systems can be tuned for performance with respect to I/O, pinned memory, and numbers of processors.

7.3 Performance

The scaling technique to optimize performance which best fits your environment depends on multiple factors. One of the main factors impacting your performance is the type of workload your system is processing. The type of workload determines what components of your environment (Web server, application server, database server and so on) are most heavily used.

7.3.1 Performance evaluation

Although performance is often a subjective feeling, it must be made measurable for evaluation. To evaluate the success of your scaling tasks the following concepts help identify performance:

- ▶ Throughput

Throughput means the number of requests in a certain period that the system can process. For example, if an application can handle 10 client requests simultaneously and each request takes one second to process, this site can have a potential throughput of 10 requests per second.

- ▶ Response time

Response time is the period from entering a system at a defined entry point until exiting the system at a defined exit point. In a WebSphere Application Server environment this is usually the time it takes for a request submitted by a Web browser until the response is received at the Web browser.

To measure the scaling success you need to generate a workload that meets 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.

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.

7.3.2 System tuning

The first step in performance 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, the operating system layer, and the WebSphere Application Server layer.

Network layer

Tuning at the network layer is usually done at a high level only. 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 and operating system 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.

7.3.3 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. This section, however, should generate some thoughts on what settings to consider when designing a WebSphere Application Server environment.

Web server

The Web server with the WebSphere plug-in should be tuned carefully. There are many different configuration options that impact the performance such as keep-alive settings, number of concurrent requests and, so on. The number of concurrent requests is perhaps the most critical factor. The Web server must allow for sufficient concurrent requests to make full use of the application server infrastructure, but should also act as a filter and keep users waiting in the network and avoid a flooding of the applications servers if more requests as the system can handle are coming in.

As a rough initial start value for testing the maximum concurrent threads can be set as follows:

$$\text{MaxClients} = (((\text{TH} + \text{MC}) * \text{WAS}) * 1.2^1) / \text{WEB}$$

Where:

- TH: Number of threads in the Web container
- MC: MaxConnections setting in the plugin-cfg.xml
- WAS: Number of WebSphere Application Server servers
- WEB: Number of Web servers

DMZ secure proxy server

The DMZ secure proxy server is designed as a possible replacement of the Web server with the plug-in. The same tuning considerations apply for the DMZ secure proxy as they do for the Web server with the plug-in loaded.

¹ This allows 20% of the threads to serve static content from the Web server

For the DMZ secure proxy, there are two additional main tuning areas to consider:

- ▶ JVM tuning

For tuning the JVM of the DMZ secure proxy the same rules apply as for the application server JVM. Refer to “Application server/Java virtual machine” on page 238 for additional information about JVM tuning.

- ▶ Proxy tuning

The proxy server provides specific tuning capabilities as well. The DMZ secure proxy is configured through the Integrated Solutions Console and provides performance monitoring infrastructure (PMI) data specific to the proxy module.

When configuring the DMZ secure proxy server, review the following settings closely:

- ▶ Proxy thread pool size
- ▶ HTTP proxy server settings
 - Proxy settings (such as time-outs, connection pooling and so on)
 - Routing rules
 - Static cache rules
 - Rewriting rules
 - Proxy server transports
 - Proxy cache instance configuration
 - Denial of service protection
- ▶ Performance monitoring service

When using PMI to monitor the performance of your environment be aware that there is no Integrated Solutions Console for the DMZ secure proxy. Therefore, you cannot use the Tivoli Performance Viewer to check the PMI data.

Application server/Java virtual machine

When configuring a JVM, look at the minimum and maximum heap sizes closely, as proper heap sizing is a pre-requisite for satisfactory JVM performance. The system default for the starting heap size and the maximum heap size depends on the platform. The default minimum heap size for the IBM JDK and Runtime Environment is 4 MB. The default maximum heap size depends on the platform and is documented in the diagnostic guide for the IBM JDK. The diagnostic guide can be found at the following Web page:

<http://www-128.ibm.com/developerworks/java/jdk/diagnosis/60.html>

Starting a JVM with too little memory means that the application must immediately switch context to allocate memory resources. This will slow down the server startup and 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 and a fragmented heap that requires compacting later on.

The levels should be adjusted during the testing phase to ascertain reasonable levels for both settings. In addition, the prepared statement cache and dynamic fragment caching also consume portions of the heap. You might be required to make additional adjustments to the heap when those values are adjusted.

Thread pools

Inside the application server JVM separate thread pools are used to manage different types of workload. Depending on your type of application, appropriate thread pools require careful planning.

Web container thread pool

The thread pool for the Web container should be closely monitored during initial load testing and production. This is the most common bottleneck in an 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 this setting in repetitive tests and adjust the numbers.

Enterprise JavaBeans container thread pool

The Enterprise JavaBeans (EJB) container can be another source of potential scalability bottlenecks. The inactive pool cleanup interval is a setting that determines how often unused 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.

Messaging connection pool

When using the messaging service make sure that you configured the messaging pool according to your needs. Depending on which messaging service you use and how messaging is used in the application you have different pooling options. Pools for message processing can, for example, be configured on a per connection factory basis. The messaging listener service provides a separate pool that can be configured through the Integrated Solutions Console.

To do so, navigate to **Servers** → **Application Servers** → **<server_name>** → **Messaging** → **Message Listener Service**.

Further pooling options exist when using activation specifications to invoke message driven beans (MDBs).

Mediation thread pool

If you want to run multiple mediations in your service integration bus infrastructure concurrently, you need to configure a mediation thread pool using the wsadmin command line interface.

Database connection pool

The database connection pool is another common location for bottlenecks, especially in data-driven applications. The default pool size is 10, and depending on the nature of the application and the number of requests, the default setting might not be sufficient. During implementation testing, pay special attention to the pool usage and adjust the pool size accordingly. The connections in the pool consume additional Java heap, so you might be required to go back and adjust the heap size after tuning the pool size.

Web services connection pool

Use HTTP transport properties for Java API for XML-based Web Services (JAX-WS) and Java API for XML-based RPC (JAX-RPC) Web services to manage the connection pool for HTTP outbound connections. Configure the content encoding of the HTTP message, enable HTTP persistent connection, and resend the HTTP request when a time-out occurs.

7.3.4 Application tuning

Tuning your application is the most important part of your tuning activities. While environment-related tuning is important to optimize resource use and avoid bottlenecks, it cannot compensate for a poorly written application. The majority of performance-related problems are related to application design and coding techniques. Only a well-designed application, implemented with the best practices for programming, will give you good throughput and response times.

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. An important task with which to optimize the application performance is application profiling.

7.4 Workload management

Workload management is the concept of sharing requests across multiple instances of a resource. Workload management is an important technique for high availability as well as for performance and scalability. Workload management techniques are implemented expressly for providing scalability and availability within a system. These techniques allow the system to serve more concurrent requests. Workload management allows for better use of resources by distributing load more evenly. Components that are overloaded, and therefore a potential bottleneck, can be routed around with workload management algorithms. Workload management techniques also provide higher resiliency by routing requests around failed components to duplicate copies of that resource.

7.4.1 HTTP servers

An IP sprayer component like the Edge Components Load Balancer can be used to perform load balancing and workload management functionality for incoming Web traffic. In addition, the WebSphere plug-in provides workload management capabilities for applications running in an application server.

7.4.2 DMZ proxy servers

As with HTTP servers, an IP sprayer component like the Edge Components Load Balancer can be used to perform load balancing and workload management functionality for incoming Web traffic. In addition, the DMZ proxy server provides workload management capabilities for applications running in an application server.

7.4.3 Application servers

In WebSphere Application Server, workload management is achieved by sharing requests across one or more application servers, each running a copy of the application. In more complex topologies, workload management is embedded in load balancing technologies that can be used in front of Web servers.

Workload management is a WebSphere facility to provide load balancing and affinity between nodes in a WebSphere clustered environment. Workload management can be an important facet of performance. WebSphere uses workload management to send requests to alternate members of the cluster 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 works differently from the WLM for distributed platforms. The workload management structure for incoming requests is handled by the WLM subsystem features of z/OS. Organizations can define business-oriented rules that are used to classify incoming requests and to assign service level agreement types of performance goals. This is done on a transaction level granularity compared to a server level granularity on the distributed workload management. The system then automatically assigns resources in terms of processor, memory, and I/O to try to achieve these goals.

In addition to the response times, the system can start additional processes, called *address spaces*, that run the user application if there are performance bottlenecks due to an unpredicted workload spike.

This explanation is an over-simplification of how workload management works in z/OS. For more information about workload management of z/OS and the WebSphere Application Server for z/OS refer to 14.1.6, “Workload management for WebSphere Application Server for z/OS” on page 428, or see the WebSphere article *Understanding WAS for z/OS*, available from the following Web page:

<http://websphere.sys-con.com/read/98083.htm>

7.4.4 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 7-1.

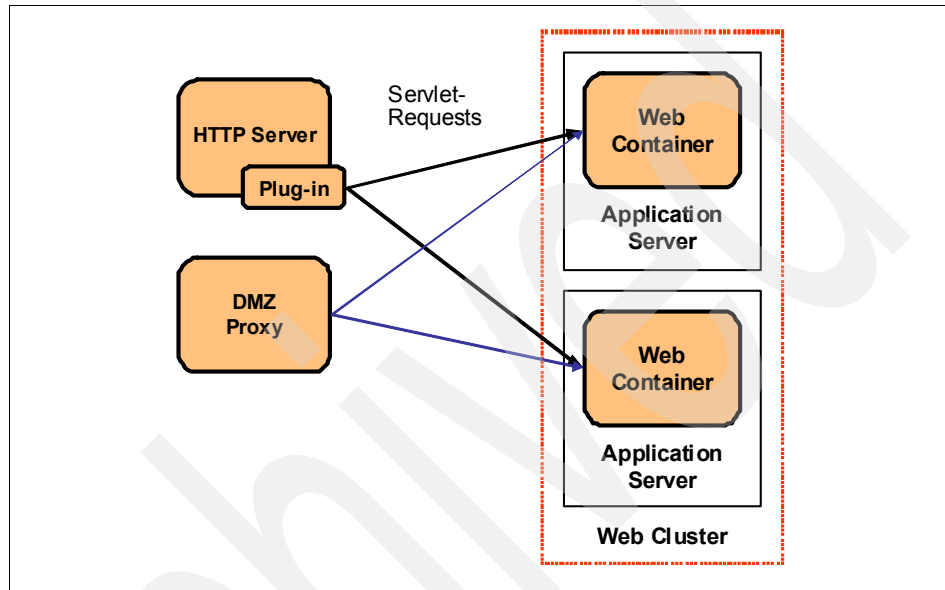


Figure 7-1 Plug-in (Web container) workload management

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.

A guideline formula for determining routing preference is:

$$\% \text{ routed to Server1} = \text{weight1} / (\text{weight1} + \text{weight2} + \dots + \text{weight}n)$$

Where there are n cluster members in the cluster.

On the z/OS platform the assignment of transactions to cluster members is performed on real-time achievement of defined performance goals. This has the benefit that the system can differentiate between light requests that only use up a small fragment of performance, and heavy requests that use more processor capacity.

For more details about Web server plug-in workload management in a cluster see the following Web page:

http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=plug-in+workload+management&uid=swg21219567&loc=en_US&cs=utf-8&lang=en

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, shown in Figure 7-2.

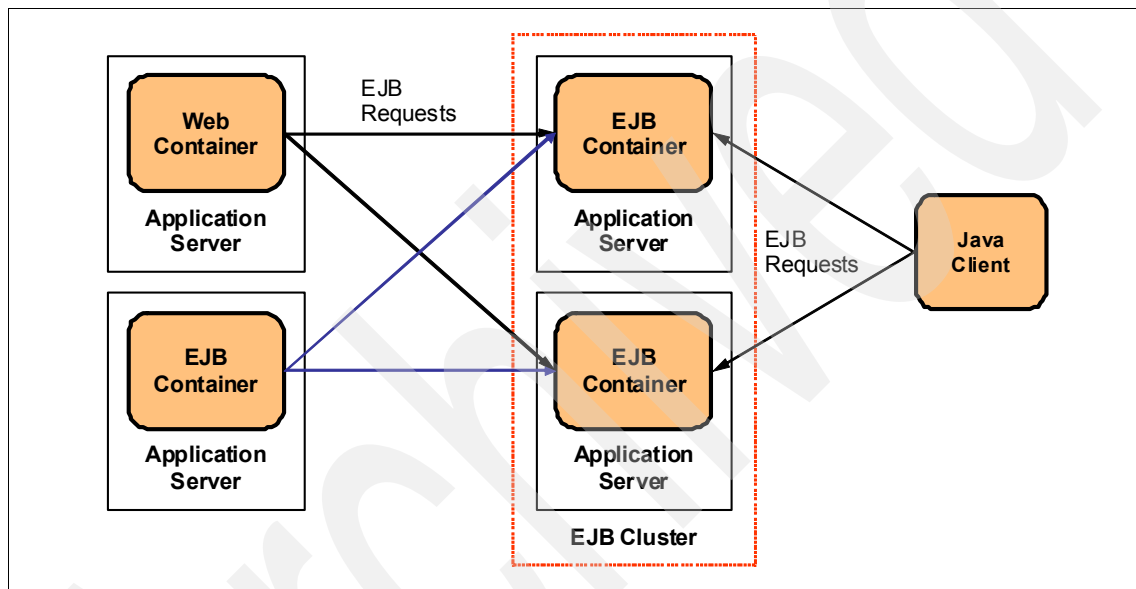


Figure 7-2 EJB workload management

In this configuration, EJB client requests are routed to available EJB servers in a round-robin fashion based on assigned server weights. The EJB clients can be servlets operating within a Web container, stand-alone Java programs using RMI/IIOP, or other 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. This policy ensures the desired distribution based on the weights assigned to the cluster members.

You can also choose that EJB requests are preferably routed to the same host as the host of the requesting EJB client. In this case, only cluster members on that host are chosen (using the round-robin weight method). Cluster members on remote host 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.

7.4.5 Scheduling tasks

WebSphere Application Server provides a Scheduler service that can be used to schedule actions to happen with the following frequencies:

- ▶ Only once
- ▶ Some time in the future
- ▶ On a recurring basis
- ▶ 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, it is executed 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 set by the Number of alarm threads parameter on the work manager.

7.5 High availability

High availability is also known as resiliency. High availability is the description of the system's ability to respond to requests no matter the circumstances. This section discusses several high availability considerations.

7.5.1 Overview

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. Add an additional 9, to achieve 99.99% availability (approximately 1 hour of unplanned outage) in a single year. The cornerstone measurement of “five nines” or 99.999% availability represents an unplanned outage of less than five minutes in a year, which can be achieved by z/OS running a parallel sysplex.

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. Ensure 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 whole system as available as possible. This can be done by minimizing the number of single points of failure (SPOF) throughout the system. If a SPOF cannot be eliminated, perform planning to mitigate the impact of that potential failure.

Note: Availability features will 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.

7.5.2 Hardware high availability

Although modern hardware is reliable and many components are fault tolerant, hardware can fail. Any mechanical component has an expected failure rate and a projected useful life until failure.

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 redundant array of inexpensive disks (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.

7.5.3 Process high availability

In WebSphere Application Server, the concept of a singleton process is used. Although not a new concept in WebSphere Application Server V7.0, 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 uses a high availability manager to provide availability for singleton processes. We discuss this further in 7.5.7, “WebSphere Application Server high availability features” on page 250,

7.5.4 Data availability

In a WebSphere Application Server environment, there are multiple places where data availability is important. Critical areas for data availability are as follows:

- ▶ Databases
- ▶ HTTP session state
- ▶ EJB session state
- ▶ EJB persistence

The majority of these requirements can be satisfied using facilities available in WebSphere Application Server. We discuss these areas in more detail in the next sections.

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 that uses a virtual IP. This enables the application to connect to one copy of the data and to transparently fail over to another working copy.
- ▶ If the data is mostly read-only, consider the option 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 an 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 in two ways:

- ▶ Using memory-to-memory replication to create a copy of the session data in one or more additional servers
- ▶ By storing the session data in an external database.

The choice of which to use is really left to the user, and performance results may vary. 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.

In contrary to the HTTP session persistence, stateful session EJB availability is handled using memory-to-memory replication only. Using the EJB container properties, you can specify a replication domain for the EJB container and enable the stateful session bean failover using memory-to-memory replication. When enabled, all stateful session beans in the container 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, bean persistence is available for all members of the cluster. Using access intent policies, you can govern the data access for the persisted bean. This EJB persistence API should not be confused with entity EJBs.

7.5.5 Clustering and failover technique

Clustering is the concept of creating highly available system processes across multiple servers. On distributed platforms it is usually deployed in a manner that only one of the servers is actively running the system resource.

Hardware-based clustering

Clustering is achieved by using an external clustering software, such as IBM HACMP on AIX systems or using operating system cluster capabilities like the Parallel Sysplex on the z/OS platform, to create a cluster of servers. Each server is generally attached to a shared disk pool through NAS, a SAN, or simply by chaining SCSI connections to an external disk array. Each system has the base software image installed. The servers stay in constant communication with each other over several connections through the use of heartbeats. Multiple paths should be configured for these heartbeats so that the loss of a switch or network interface does not necessarily cause a failover.

Note: Too few paths can create problems with both servers believing they should be the active node. This situation is called cluster isolation and is a serious threat to data consistency. Too many paths can create unnecessary load associated with heartbeat management.

Software-based clustering

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. 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 and the DMZ secure proxy server knows the location of all cluster members and route requests to all of these. In the case of a failure, the plug-in marks the cluster member as unavailable and does not send further requests to that member for a fixed interval. After the retry interval, the plug-in will mark the cluster member as available again and retry. This retry interval is configured through the Web server plug-in settings in the Integrated Solutions Console.

7.5.6 Maintainability

Maintainability is the ability to keep the system running before, during, and after scheduled maintenance. When considering maintainability in performance and scalability, remember 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.

7.5.7 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 and will assist you in planning for high availability.

High availability manager

WebSphere Application Server uses a high availability manager to eliminate single points of failure. The 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 continually polls all of the core group members to verify that they are active and healthy.

Note: The high availability manager service runs by default in each server.

For certain functions (like transaction peer recovery) the high availability manager 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 maintaining a backup for these services. WebSphere Application Server also supports other high availability solutions such as HACMP, Parallel Sysplex, and so on.

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 introduces some overhead but significantly improves application server availability.

A high availability manager focuses on recovery support and scalability in the following areas:

- ▶ Embedded messaging
- ▶ Transaction managers
- ▶ Workload management controllers
- ▶ Application servers
- ▶ WebSphere partitioning facility instances
- ▶ On-demand routing
- ▶ Memory-to-memory replication through Data Replication Service (DRS)
- ▶ Resource adapter management

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 tolerant of poor performing application servers, which might happen if the application server is overloaded, swapping, or thrashing. 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, at a specific interval. This interval can be configured in the Integrated Solutions Console by navigating to **Servers** → **Core Groups** → **<core_group_name>** → **Discovery and failure detection** → **Heartbeat transmission period**. If the JVM does not respond to heartbeats within a defined time frame, it is considered down.

This time-out can be configured in the Integrated Solutions Console by navigating to **Servers** → **Core Groups** → **<core_group_name>** → **Discovery and failure detection** → **Heartbeat timeout period**. You can use this method with multicast emulation. However, this method must be used for true multicast addressing.

A new feature in WebSphere Application Server V7.0 is the ability to configure an alternative protocol provider to monitor and manage communication between core group members. In general, alternate protocol providers, such as the z/OS Cross-system Coupling Facility (XCF)-based provider, uses less system resources than the default Discovery Protocol and Failure Detection Protocol, especially during times when the core group members are idle.

In either case, if a JVM fails, the application server on which it is running is separated from the core group, and any services running on that application server are failed over to the surviving core group members.

A JVM can be a node agent, an application server, or a deployment manager. If a JVM fails, any singletons running in that JVM are activated on a peer JVM after the failure is detected. This peer JVM is already running and eliminates the normal startup time, which potentially can be minutes.

This actually is a key difference to using operating system-based high availability. The high availability manager usually recovers in seconds while operating system-based solutions take minutes.

When an application server fails, the high availability manager assigns the failing application servers work to another eligible application server. Using shared storage for common logging facilities (like the transaction logs) allows the high availability manager to recover in-doubt and in-flight work if a component fails.

Note: The following Web page provides a testing routine you can use to determine if your shared file system is suitable for use with the high availability manager:

http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=transaction+log+failover&uid=swg24010222&loc=en_US&cs=utf-8&lang=en

Core group

A core group is a high availability domain that consists of a set of processes in the same cell that can directly establish high availability relationships. Highly available components can only fail over to another process in the same core group and replication can occur only between members of the same core group.

A cell must contain at least one core group, although multiple core groups are supported. Each core group contains a core group coordinator to manage its high availability relationships, and a set of high availability policies that are used to manage the highly available components within that core group.

WebSphere Application Server provides one standard core group: the DefaultCoreGroup that is created during installation. New server instances are added to the default core group as they are created.

In most cases one core group is sufficient for establishing a high availability environment. However, certain topologies require the use of multiple core groups. A basic rule is that all members of a core group require full IP visibility. Therefore, you have to create multiple core groups if you spread the application servers of your cell across different firewall zones.

Note: A large number of application servers in a cell increases the overhead of core group services and server startup times. You might consider creating additional core groups when you have more than 50 servers in a cell.

If you are using a DMZ secure proxy server with dynamic routing, the routing information is exchanged using core groups. In this case you need to create a tunnel access point group to establish a core group bridge tunnel between the core groups that are running on both sides of the firewall.

The core group contains a bridge service that supports cluster services that span multiple core groups. Core groups are connected by access point groups. A core group access point defines a set of bridge interfaces that resolve IP addresses and ports. It is through this set of bridge interfaces that the core group bridge provides access to a core group.

When moving core group members to new core groups, remember the following information:

- ▶ Each server process within a cell can only be a member of one core group.
- ▶ If a cluster is defined for the cell, all cluster members must belong to the same core group.

Network communication between all members of a core group is essential. The network environment must consist of a fast local area network with full IP visibility and bidirectional communication between all core group members. IP visibility means that each member is entirely receptive to the communications of any other core group member.

High availability groups

High availability groups are part of the high availability manager framework. A high availability group provides the mechanism for building a highly available component and enables the component to run in one of several different processes. A high availability group cannot extend beyond the boundaries of a core group.

A high availability group is associated with a specific component. The members of the group are the set of processes where it is possible to run that component. A product administrator cannot directly configure or define a high availability group, and its associated set of members. Instead, high availability groups are created dynamically at the request of the components that need to provide a highly available function.

High availability groups are dynamically created components of a core group. A core group contains one or more high availability groups. However, members of a high availability group can also be members of other high availability groups, if all of these high availability groups are defined within the same core group.

Every high availability group has a policy associated with it. This policy is used to determine which members of a high availability group are active at a given time. The policies that the high availability groups use are stored as part of the core group configuration. The same policy can be used by several different high availability groups, but all of the high availability groups to which it applies must be part of the same core group.

Any WebSphere Application Server highly available component can create a high availability group for its own usage. The component code must specify the attributes that are used to create the name of the high availability group for that component. For example, establishing a high availability group for the transaction manager is as follows:

- ▶ The code included in the transaction manager component code specifies the attribute `type=WAS_TRANSACTIONS` as part of the name of the high availability group that is associated with this component.
- ▶ The high availability manager function includes the default policy Clustered TM Policy that includes `type=WAS_TRANSACTIONS` as part of its match criteria.
- ▶ 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 name-value pair `type=WAS_TRANSACTIONS` included in the high availability group name is matched to the same string in the policy match criteria for the Clustered TM Policy. This match associates the Clustered TM Policy with the high availability group that was created by the transaction manager component.

- ▶ After a policy is established for a high availability group, you can change some of the policy attributes, such as quorum, fail back, and preferred servers. 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.

Note: 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 activates the messaging engine on another member in the cluster.

To accomplish the failover seamlessly, the queue information and message data must be stored in a shared location, either using an external database or a shared disk environment.

Note: For those using embedded Derby as a messaging data store, concurrent access can be a concern. The embedded Derby does not support multiple servers running the Derby engine, so there would be no ability to have multiple servers communicating with the same shared file system.

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 active cluster member failure.

7.6 Caching

Caching is a facility to off-load work to one or more external devices so that the application server is not required to do all of the work associated with user requests. 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 varied.

In this section we will discuss the following caching capabilities of WebSphere Application Server V7.0:

- ▶ Edge caching on page 256.
- ▶ Dynamic caching on page 257.
- ▶ Data caching on page 258.

7.6.1 Edge caching

Edge caching embraces a variety of methods. There are numerous software components that can provide caching capabilities at the edge of the network:

- ▶ Reverse proxy servers, such as the Caching Proxy of the Edge components or the DMZ secure proxy.
- ▶ Software components at the Web server, such as the Fast Response Cache Accelerator (FRCA).
- ▶ External caching proxy providers that can provide content off loading 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 (FRCA)

FRCA is a caching technique used by the IBM HTTP server to cache static content like HTML, text, graphics, and so on. Because the path length of such a static request is reduced, the response time for these static requests will be decreased. The amount of data processed by WebSphere Application Server will decrease as well because only requests for dynamic elements are passed to the application server. This cache can even be configured to enable high speed caching of servlets and JSP files by using the Adaptive Fast Path Architecture (AFPA) adapter bean through the external caching configuration section of the administrative console.

Note: On the z/OS platform, WebSphere Application Server V7.0 can be configured to use the FRCA technique, which provides better performance than the default WebSphere Application Server implementation.

Caching proxy

By using the Edge components Caching Proxy, you can intercept requests and 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 result in significant performance improvements and reduction in response time. This

cache can be configured to offload static content or dynamic content (but only on a page level). Using the WebSphere Application Server Integrated Solutions Console, you can also control how the content is loaded and invalidated.

DMZ secure proxy

The DMZ secure proxy server is a caching engine that allows you to offload request processing from the core application servers. Using the DMZ secure proxy server, you configure multiple security levels and routing policies. The DMZ secure proxy server also allows you to serve static content directly, thereby increasing performance for these requests.

The DMZ secure proxy server is capable of caching static and dynamic content at the edge of the network. Depending on the routing policy used, it can dynamically determine the availability of applications on the application servers.

Hardware caching

There are multiple network equipment providers that offer hardware cache devices. These serve the same purpose as software caches do, namely to offload content. The main difference is that these appliances are usually not running full versions of an operating system. Instead they use a specialized operating system that is dedicated to performing the caching function. This can include custom file systems that offer higher performance than the operating system file system and a significantly reduced instruction set. By placing dedicated appliances, instead of software caching, in your architecture, you may 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 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.

7.6.2 Dynamic caching

Dynamic caching refers to the methods employed by WebSphere Application Server either to provide fragment caching or to reuse components within the application server engine. Fragment caching means that only some portions of a page are cached.

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 servlet/JSP level. The caching options in cachespec.xml file must include sufficient details to allow the dynamic cache service to build a unique cache-key. This cache-key is used to uniquely identify each object. This might be achieved by specifying request parameters, cookies, and so on. The cachespec.xml also file allows you to define cache invalidation rules and policies.

Note: Pay special attention to the servlet caching configuration, because you create unexpected results by returning a cached servlet fragment that is stale.

Another dynamic caching option available is Edge Side Include (ESI) caching. ESI caching is an in-memory caching solution implemented through the Web server plug-in, WebSphere proxy server, or the DMZ secure proxy server. If dynamic caching is enabled at the servlet Web container level, the plug-in uses ESI caching.

An additional header is added to the HTTP request by the caching facility, called the Surrogate-Capabilities header. 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.

7.6.3 Data caching

Data caching is used 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. There are multiple methods for configuring data caching:

- ▶ Keep a local copy in a database within the same network realm as the application.
- ▶ Cache data from a localized database in memory to minimize database reads.
- ▶ Use EJB persistence to keep the data in the memory space of the running application.

Sometimes data is provided by an external provider. Making live calls to this data can prove to be a single point of failure and a slower performer. 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.

Database data caching

To minimize direct reads from the database, database systems usually offer one or more of the following options:

- ▶ Fetch ahead constructs attempt to anticipate that additional pages from that table are required and then 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 data will be requested again.

Both of these constructs reduce disk access, opting instead for reading the data from the memory, increasing performance. These facilities assume that the data is predominately read only. If the data has been written, the copy in memory can be stale, depending on the write implementation of the database. Also, memory buffers can be used to store data pages, reducing disk access. The key is to make sure that the system has enough memory to provide to the database.

Application data caching

Another option is to cache some of the database or Web page data inside an application by creating objects that are instantiated when the application server is started. Those objects pull the necessary information in memory, improving performance as 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 if a dynamic cache that might grow over time is implemented.

EJB persistence implies loading the data into an EJB after a call to the data provider. This is similar to database caching, except that 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.

7.7 Session management

Multi-system 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 always independent.

7.7.1 Overview

A client often makes a request, waits for the result, and then makes one or more subsequent requests. The processing of these subsequent requests requires information about the data processed in previous requests. Session management allows linking requests that belong together.

In terms of session management there are two kinds of requests:

- ▶ Stateless

A server processes requests based solely on information provided with each request and does not rely on information from earlier requests. Therefore, the server does not need to maintain state information between requests.

- ▶ Stateful

A server processes requests based on both the information that is provided with each request and information that is stored from earlier requests. To achieve this, 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. For stateful interactions, the server that processes a request needs access to the state information necessary to execute that request. Either the same server will process all requests that are associated with dedicated state information, or the state information can be shared by all servers that require it. If the data is shared across multiple systems, this will have a performance impact.

The load distribution facilities in WebSphere Application Server use several different techniques for maintaining state information between client requests:

- ▶ Session affinity

The load distribution facility (for example, the Web server plug-in) recognizes the existence of a client session and attempts to direct all requests within that session to the same server.

- ▶ Transaction affinity

The load distribution facility recognizes the existence of a transaction and attempts to direct all requests within the scope of that transaction to the same server.

- ▶ Server affinity

The load distribution facility recognizes that although multiple servers might be acceptable for a given client request, a particular server is best suited for processing that request.

The 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 cluster members.

7.7.2 Session support

Information entered by a user in a Web application is often needed throughout the application. For example, a user selection might be used to determine the path through future menus or options to display content. This information is kept in a session.

A session is a series of requests to a servlet that originate from the same user. Each request arriving at the servlet contains a session ID. Each ID allows the servlet to associate the request with a specific user. The WebSphere session management component is responsible for managing sessions, providing storage for session data, 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
- ▶ Session storage options.

The following sections outline the planning considerations for each.

Note: Before finishing your session management planning, review the article *Sessions* in the Information Center at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/cprs_sess.html

Application design

Although using session information is a convenient method for the developer, this usage should be minimized. Only objects really needed for processing of subsequent requests should be stored in the session. If sessions are persisted during runtime, there is a performance impact if the session data is too big.

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 Web browser to be stored as a cookie.

URL rewriting

URL rewriting requires the developer to use special encoding APIs and to set up the site page flow to avoid losing the encoded information. The session identifier is stored in the page returned to the user. WebSphere encodes the session identifier as a parameter on URLs that have been encoded programmatically by the Web application developer.

URL rewriting can only be used for pages that are dynamically generated for each request, such as pages generated by servlets or JSPs. If a static page is used in the session flow the session information is lost. URL rewriting forces the site designer to plan the user's flow in the site to avoid losing their session ID.

SSL ID tracking

With SSL ID tracking, SSL session information is used to track the session ID. Because the SSL session ID is negotiated between the Web browser and an HTTP server, it cannot survive an HTTP server failure. However, the failure of an application server does not affect the SSL session ID. In environments that use WebSphere components with multiple HTTP servers, you must use an affinity mechanism for the Web servers when SSL session ID is used as the session tracking mechanism.

Note: SSL tracking is supported in IBM HTTP Server. Session tracking using the SSL ID is deprecated in WebSphere Application Server V7.0

When the SSL session ID is used as the session tracking mechanism in a clustered environment, either cookies or URL rewriting must be used to maintain session affinity. The cookie or rewritten URL contains session affinity information that enables the Web server to properly route requests back to the same server after the HTTP session has been created on a server. The SSL ID is not sent in the cookie or rewritten URL but is derived from the SSL information.

The main disadvantage of using SSL ID tracking is the performance degradation due to the SSL overhead. If you have a business requirement to use SSL, this is probably a good choice.

Selecting multiple tracking mechanisms

It is possible to combine multiple options for a Web application.

- ▶ Use of SSL session identifiers has preference to cookie and URL rewriting.
- ▶ Use of cookies has preference to URL rewriting.

If selecting SSL session ID tracking, we suggest 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 as follows:

- ▶ Local sessions (non-persistent)
- ▶ Database persistent sessions
- ▶ Memory-to-memory replicated persistent sessions

The last two options allow session data to be accessed by multiple servers and should be considered when planning for failover. Using a database or session replication is also called session persistence.

Storing session data external to the system can have drawbacks in performance. The amount of impact depends on the amount of session data, the method chosen, and the performance and capacity of the external storage. Session management implements caching optimizations to minimize the impact of accessing the external storage, especially when consecutive requests are routed to the same application server.

Local sessions (non-persistent)

If the session data is stored in the application server memory only, the session data is not available to any other servers. Although this option is the fastest and the simplest to set up, an application server failure ends the session, because the session data is lost.

The following settings can help you manage the local session storage:

- ▶ **Maximum in-memory session count**

This setting enables you to define a limit to the number of sessions in memory. This prevents the sessions from acquiring too much of the JVM heap and causing out-of-memory errors.

- ▶ **Allow overflow**

This setting permits an unlimited number of sessions. If you choose this option, monitor the session cache size closely.

Note: Session overflow is enabled by default in WebSphere Application Server V7.0

- ▶ **Session time-out**

This setting determines when sessions can be removed from cache.

Database persistent sessions

You can store session data in an external database. The administrator must create the database and configure the session database in WebSphere through a data source.

The Use 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 when session attributes are large and few changes are required to the session attributes.

Memory-to-memory replicated 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 effort of maintaining a production database and eliminates the single point of failure that can occur with a database. Test to determine which persistence mechanism is the best one in your environment.

The administrator sets up memory-to-memory replication by creating a replication domain and adding application servers to it. You can 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 in one of the following manners:

- ▶ To one server (single replica)
- ▶ To every server (entire domain)
- ▶ 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 into many servers. By configuring more replicas, your system becomes more tolerant to possible failures of application servers 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 its own sessions to other application servers.
- ▶ **Client mode**
In this mode, a server only broadcasts or sends copies of its own sessions. It does not receive copies of sessions from other servers.
- ▶ **Both mode**
In this mode, the server is capable of sending its own sessions and receiving sessions from other application servers. Because each server has a copy of all sessions, this mode uses the most memory 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** → **Server Types** → **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** → **Application Types** → **WebSphere 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** → **Application Types** → **WebSphere enterprise applications** → **<app_name>** → **Manage modules** → **<Web_module>** → **Session management** → **Distributed environment settings** → **Memory-to-memory replication**.

The following session management properties can be set:

- ▶ Session tracking mechanism
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.
- ▶ Maximum in-memory session count
Select Maximum in-memory session count and whether to allow this number to be exceeded, or overflow.
- ▶ Session time-out
Session time-out specifies the amount of time to allow a session to remain idle before invalidation.
- ▶ Security integration
Security integration specifies a user ID be associated with the HTTP session.
- ▶ Serialize session access
Serialize session access determines if concurrent session access in a given server is allowed.
- ▶ Overwrite session management
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
Distributed environment settings select how to persist sessions (memory-to-memory replication or a database) and set tuning properties.

Note: Memory-to-memory persistence is only available in a Network Deployment distributed server environment.

7.8 Data replication service

The data replication service (DRS) is the WebSphere Application Server component that replicates data. Session manager, dynamic cache, and stateful session EJBs are the three consumers of the replication service.

To use data replication for these services, you must first create the replication domains:

1. Create one replication domain for dynamic cache. The replication domain must be configured for full group replication (both mode).
2. 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 7.7.2, “Session support” on page 261.

Configure dynamic cache replication through **Servers** → **Server Types** → **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** → **Server Types** → **Application servers** → **<server_name>** → **EJB Container**.
- ▶ Application
Navigate to **Applications** → **Application Types** → **WebSphere enterprise applications** → **<app_name>** → **Stateful session bean failover settings**.
- ▶ EJB module
Navigate to **Applications** → **Application Types** → **WebSphere enterprise applications** → **<app_name>** → **Manage modules** → **<EJB_module>** → **Stateful session bean failover settings**.

7.9 WebSphere 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 the 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 following tool sets:

- ▶ Performance monitoring considerations on page 268
- ▶ Tivoli performance viewer on page 271
- ▶ WebSphere performance advisors on page 271
- ▶ WebSphere request metrics on page 273

7.9.1 Performance monitoring considerations

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 the Performance Data Framework as defined in the J2EE 1.4 standard.

Configured through the Integrated Solutions 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:

- ▶ Application resources
 - Custom PMI
 - Enterprise bean counters
 - JDBC connections counters
 - J2C connections counters
 - Servlets/JSPs counters
 - SIP counters
 - Web services counters
 - ITCAM counters
 - Any custom that runs in the environment
- ▶ WebSphere runtime resources
 - JVM memory
 - Thread pools
 - Database connection pools
 - Session persistence
 - Dynamic caching

- Proxy counters
- ORB counters
- Transactional counters
- Workload management counters
- ▶ System resources
 - Processor usage
 - Total free memory
 - Components that are controlled outside the WebSphere environment but that are vital in healthy application state

Note: PMI offers the custom PMI application programming interface (API). This enables you to insert your own custom metrics and have them captured and available to the standard monitoring tools.

When determining which metrics to capture, you can select from the following monitoring statistics sets:

- ▶ Basic (enabled by default)
 - 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 with the option to inspect the state of the JVM. This interface was introduced with 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 became deprecated in 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 gathered through the JVMTI are 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 V7.0, this impact has been minimized, though obviously the more statistics you capture, the more processing power is required.

ITCAM for Web Resources Data Collector (eCAM)

WebSphere Application Server V7.0 offers an optional enhancement to PMI called eCAM. eCAM is a separate data collector which is bootstrapped at startup of the application server and monitors class loads and instruments only Web and EJB containers. This data collector allows gathering of request oriented data, elapsed time, CPU data, and counts.

The data collected by the eCAM data collector is exposed using an Mbean registered in PMI. The collected performance data can be viewed through the standard Tivoli Performance Viewer (TPV) of WebSphere Application Server.

The following counts collected by eCAM are exposed through TPV:

- ▶ RequestCount
- ▶ AverageResponseTime
- ▶ MaximumResponseTime
- ▶ MinimumResponseTime
- ▶ LastMinuteAverageResponseTime
- ▶ 90%AverageResponseTime
- ▶ AverageCPUUsage
- ▶ MaximumCPUUsage
- ▶ MinimumCPUUsage
- ▶ LastMinuteAverageCPUUsage
- ▶ 90%AverageCPUUsage

For details about the data collected by eCAM, refer to *IBM Tivoli Composite Application Manager for WebSphere Application Server counters* in the WebSphere Application Server Information Center at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/rprf_tpmcounter.html

7.9.2 Tivoli performance viewer

Tivoli Performance Viewer (TPV) is included with WebSphere Application Server V7.0 and is used to record and display performance data. Since WebSphere Application Server V6.0, TPV is integrated into the Integrated Solutions Console.

Using Tivoli Performance Viewer, you can perform the following tasks:

- ▶ 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.
- ▶ Log performance data
Using TPV you can log real-time performance data and review the data at a later time.
- ▶ View server performance logs
You can record and view data that has been logged by TPV in the Integrated Solutions Console.

You can use TPV to create summary reports. These reports let you monitor the server's real-time performance and health. TPV enables you to work with the performance modules. With these modules, you can drill down on specific areas of interest, even old logs. Use the log analysis tools to detect trends over time. TPV can also save performance data for later analysis or problem determination.

As the TPV runs inside the Integrated Solutions Console, the performance impact depends on which edition of WebSphere Application Server you run. When running the single server edition, the TPV runs in the same JVM as your application. In Network Deployment, the TPV runs in the JVM of the deployment manager. Certain functions (like the advisor), however, require resources in the node agents or in the application servers.

7.9.3 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 perform tests after any change is implemented. There are two types of advisors:

- ▶ Performance and Diagnostic Advisor
- ▶ Performance Advisor in Tivoli Performance Viewer.

Performance and Diagnostic Advisor

This advisor is configured through the Integrated Solutions Console. It 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. It offers advice about the following components:

- ▶ J2C Connection Manager
 - Thread pools
 - LTC Nesting
 - Serial reuse violation
 - Plus various different diagnostic advises
- ▶ Web Container Session Manager
 - Session size with overflow enabled
 - Session size with overflow disabled
 - Persistent session size
- ▶ Web Container
 - Bounded thread pool
 - Unbounded thread pool
- ▶ Orb Service
 - Unbounded thread pool
 - Bounded thread pool
- ▶ Data Source
 - Connection pool size
 - Prepared statement cache size
- ▶ Java virtual machine (JVM)
 - Memory leak detection

If you need to gather advice about items outside this list, use the Tivoli Performance Viewer Advisor.

Performance Advisor in Tivoli Performance Viewer

This advisor is slightly different from the Performance and Diagnostic Advisor. The Performance Advisor in Tivoli Performance Viewer is invoked only through the TVP interface of the Integrated Solutions Console. It runs on the application server you are monitoring, but the refresh intervals are based on selecting refresh through the console. Also, the output is routed to the user interface

instead of to an application server output log. This advisor also captures data and gives advice about more components. Specifically, this advisor can capture the following types of information:

- ▶ ORB service thread pools
- ▶ Web container thread pools
- ▶ Connection pool size
- ▶ Persisted session size and time
- ▶ Prepared statement cache size
- ▶ Session cache size
- ▶ Dynamic cache size
- ▶ JVM heap size
- ▶ DB2 performance configuration

The Performance Advisor in Tivoli Performance Viewer provides more extensive advice than the Performance and Diagnostic Advisor. Running the Performance Advisor in Tivoli Performance Viewer can require plenty of resources and impact performance. Use it with care in production environments.

7.9.4 WebSphere request metrics

PMI provides information about average system resource usage statistics but does not provide any correlation between the data. Request metrics, in contrast, provide data about each individual transaction and correlate this data.

Overview

Request metrics gather information about single transactions within an application. The metric tracks each step of a transaction and determines the process time for each of the major application components. Several components support this transaction metric:

- ▶ Web server plug-ins
- ▶ Web container
- ▶ EJB container
- ▶ JDBC calls
- ▶ Web services engine
- ▶ Default messaging provider

The amount of time that a request spends in each component is measured and aggregated to define the complete execution time for that transaction. Both the individual component times and the overall transaction time can be useful metrics when trying to gauge user experience on a site. The data allows for a hierarchical by response time view for each individual transaction. When debugging resource constraints, these metrics provide critical data at each component. The request metric provides filtering mechanisms to monitor

synthetic transactions or to track the performance of a specific transaction. By using test transactions, you can measure performance of the site end-to-end.

From a performance perspective, using transaction request metrics can aid in determining if an application is meeting service level agreements (SLAs) for the client. The metrics can be used to alert the user when an SLA target is not met.

Request metrics help administrators answer the following questions:

- ▶ What performance area should the user 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 and do not violate the SLAs

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 as follows:

- ▶ 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
- ▶ Source IP filters

The performance impact is less than 5% when all incoming transactions are being instrumented. This is not a significant amount, but factor this in when implementing the metrics. The path for implementation in the Integrated Solutions Console 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 impact on the system. The higher a tracing level, the greater the performance penalty 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 Integrated Solutions Console by navigating to **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, the agent, or both at the same time.

Note: It is suggested not to 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 the information on the following Web page:

<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 V7.0 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 Web pages:

- ▶ 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>

7.10 Summary: Checklist for performance

Table 7-1 on page 277 provides a summary of items to consider as you plan, and additional resources that can help you.

Table 7-1 Planning checklist for Web applications

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 (8.12, "Mapping application to application servers" on page 307).
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
Examine initial settings for performance tuning parameters and adjust if necessary. Reevaluate these periodically: <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

Planning item
Plan for clustering: <ul style="list-style-type: none"> ▶ Number of application servers ▶ Physical location ▶ Server weights ▶ Prefer local setting
Consider the Scheduler service to run intensive tasks in off-peak hours.
Evaluate session management needs: <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)
For messaging applications using the default messaging provider, consider: <ul style="list-style-type: none"> ▶ Quality of service settings ▶ Bus topology

References

The WebSphere Application Server Information Center also contains a lot of useful information.

For a good entry point to monitoring topics, see the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6topmonitoring.html>

For a good entry point to performance topics, see the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6toptuning.html>

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. It contains items of interest for application developers, WebSphere infrastructure architects, and system administrators. This chapter includes the following topics:

- ▶ What is new in V7.0
- ▶ End-to-end life cycle
- ▶ Development and deployment tools
- ▶ Naming conventions
- ▶ Source code management
- ▶ Automated build process
- ▶ Automated deployment process
- ▶ Automated functional tests
- ▶ Test environments
- ▶ Managing application configuration settings
- ▶ Planning for application upgrades in production
- ▶ Mapping application to application servers
- ▶ Planning checklist for applications

8.1 What is new in V7.0

The following list highlights the features added since WebSphere Application Server V7.0:

- ▶ Rational Application Developer Assembly and Deploy V7.5

Rational Application Developer Assembly and Deploy V7.5 ships with WebSphere Application Server V7.0. It is fully licensed with WebSphere Application Server V7.0. It can be used to build, test, and deploy Java Platform, Enterprise Edition (Java EE) applications on a WebSphere Application Server V7.0 environment (but not on any previous release). It has support for all Java EE artifacts supported by WebSphere Application Server V7.0, such as servlets, JSPs, Enterprise JavaBeans (EJBs), XML, and Web services. It also supports developing Java EE 5 applications.

- ▶ Rational Application Developer for WebSphere Software V7.5

Rational Application Developer for WebSphere Software V7.5 ships with WebSphere Application Server V7.0 which contains a 60 day trial license. It is a fully featured integrated development environment that can be used to build, test, and deploy Java EE applications on a WebSphere Application Server V7.0 environment (and also supports the previous releases V6.0 and V6.1). It has support for all Java EE artifacts supported by WebSphere Application Server V7.0, such as servlets, JSPs, EJBs, XML, SIP, Portlet, and Web services. It also supports developing Java EE 5 applications.

- ▶ Portlet application support enhancement

The portlet container in WebSphere Application Server V7.0 provides the runtime environment for JSR 268 compliant portlets. Portlet applications are intended to be combined with other portlets collectively to 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 is Rational Application Developer for WebSphere Software. Review *Introduction: Portlet applications* in the WebSphere Application Server Information Center at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/welc6tech_port_intro.html

Portlets are packaged in WAR files.

JSR 268 provides improvements to the portlet programming model. It allows for events and shared parameters between portlets. It improves coordination with other Web frameworks, and better serves resources and usage of cookies. It provides an easy way to test portlets without the need to set up a full portal.

For more information, see the following Web pages:

- JSR 268 Specification

<http://www.jcp.org/en/jsr/detail?id=268>

- IBM developerWorks article *Exploiting the WebSphere Application Server V6.1 portlet container: Part 1: Introducing the portlet container*

http://www.ibm.com/developerworks/websphere/library/techarticles/0607_hesmer/0607_hesmer.html

- ▶ Session Initiation Protocol (SIP) support

SIP applications are Java programs that use at least one SIP servlet written conforming 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.

Rational Application Developer 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 Java EE application archive (EAR file), just like other Java EE components.

For more information, see the following Web pages:

- JSR 116 SIP Servlet API Specification

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

- RFC 3261

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

- ▶ IBM Software Developer Kit (SDK) for Java Version 6 support

WebSphere Application Server V7.0 supports the Java Development Kit (JDK) version 6. This new JDK has enhanced the virtual machine, garbage collection scheme, and Just-in-time (JIT) compiler implementations from version 5. The Java Virtual Machine (JVM) is the runtime component of the JDK. The SDK is required for both the runtime and any remote Java client.

For more information, see:

- Sun's Java SE 6 Release Notes

<http://java.sun.com/javase/6/webnotes/features.html>

- Sun's Java SE 6 Compatibility Notes

<http://java.sun.com/javase/6/webnotes/compatibility.html>

▶ Enterprise JavaBeans (EJB) thin client

WebSphere Application Server V7.0 introduces an EJB thin client. An EJB client is a Java application that accesses the remote EJB from a server through Java Native Directory Interface (JNDI) lookup. The EJB client is thin, as it only requires the EJB thin client runtime library, as opposed to a complete application client or application server installation.

For more information, see the following Web pages:

- RCP Project

<http://www.eclipse.org/home/categories/rcp.php>

- RCP tutorial

<http://www.eclipse.org/articles/Article-RCP-1/tutorial11.html>

▶ Just in time deployment for EJB 3.0 module

Prior to WebSphere Application Server V7.0, to run enterprise beans on WebSphere Application Server you needed to use the EJBDeploy tool to generate the deployment code for the enterprise beans. The EJBDeploy tool manually resolved each EJB reference to its target EJB JNDI name. WebSphere Application Server V7.0 introduces a new feature that performs just-in-time deployment for EJB 3.0 modules. The application server automatically locates the correct target EJB reference binding.

▶ WebSphere Business Level Application

WebSphere Business Level Application expands the notion of applications beyond Java EE 5. It enables new formats for deployment packages, and a new way of representing applications. It enables you to create a new logical application that represents a meaningful business function.

We discuss this more in 3.1.12, “Business level applications” on page 70.

Also for more information, see the WebSphere Application Server Information Center at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_app_bla.html

8.2 End-to-end life cycle

The WebSphere Application Server V7.0 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 as follows:

- ▶ 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 feedback into itself and that software grows as the life cycle is repeated. The opposite is a waterfall model where the output of each stage spills into the subsequent stage.

This iterative behavior of RUP occurs at both the macro and micro 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 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 8-1 on page 284 gives an overview of the RUP.

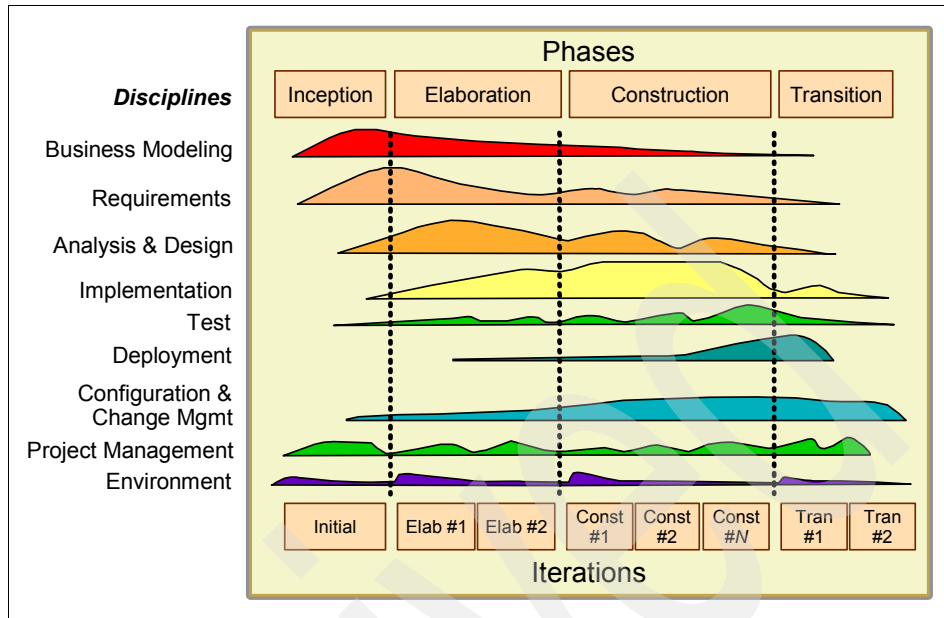


Figure 8-1 Rational Unified Process overview

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 simply 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 followed as much or as little as is required.

For more information about RUP, see the following Web page:

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

8.3 Development and deployment tools

The WebSphere Application Server V7.0 environment comes with a rich set of development tools. All editions of WebSphere Application Server V7.0 include a full licensed version of the Rational Application Developer Assembly and Deploy V7.5, as well as a 60 day trial version of the Rational Application Developer for WebSphere Software V7.5.

Rational Application Developer Assembly and Deploy V7.5 supports only the version of the WebSphere Application Server with which it ships. This means that Rational Application Developer Assembly and Deploy V7.5 supports most new features of WebSphere Application Server V7.0 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.

Rational Application Developer for WebSphere Software V7.5 supports all new features of WebSphere Application Server V7.0 and is a fully featured integrated development environment for developing SIP, Portlet, Web services, and Java EE applications. It supports previous versions of WebSphere Application Server (V6.0 and V6.1) as an integrated test environment.

8.3.1 Rational Application Developer for Assembly and Deploy V7.5

Rational Application Developer Assembly and Deploy V7.5 is based on the Eclipse 3.4 platform and provides the following features:

- ▶ Java EE support
- ▶ Development of standard Java 2 Enterprise Edition (J2EE) and Java EE artifacts, such as servlets, JSPs, and EJBs complying with J2EE 1.4 and 1.5 specifications
- ▶ Development of static Web projects (HTML, CSS style sheets, JavaScript)
- ▶ SIP development, including support for JSR 116 SIP servlets
- ▶ XML tools to build and validate XML artifacts, including schemas, DTDs, and XML files
- ▶ WebSphere Enhanced EAR support
- ▶ Profile management tool

- ▶ Support for WebSphere Application Server V7.0 test environments in either a local or remote configuration, but no support for any previous versions of WebSphere Application Server (such as V6.0 or V6.1)
- ▶ 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 and Rational ClearCase®

To summarize, Rational Application Developer Assembly and Deploy V7.5 is a development environment that provides you with the tooling necessary to create, test, and deploy the various artifacts supported by WebSphere Application Server V7.0.

It does not include the productivity-enhancing features and visual editors found in Rational Application Developer for WebSphere Software V7.5. 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 V6.0 or V6.1) as test environments.

8.3.2 Rational Application Developer for WebSphere Software V7.5

Rational Application Developer for WebSphere Software V7.5 includes all the features of Rational Application Developer Assembly and Deploy V7.5 and adds more productivity-enhancing features, which makes it an even more appealing development environment for advanced Java EE 5 development.

In addition to the features in Rational Application Developer Assembly and Deploy V7.5, Rational Application Developer for WebSphere Software V7.5 brings features such as the following:

- ▶ Full support for Java EE 5, including EJB 3.0 support
- ▶ Portal application development with support for WebSphere Portal V6.0 and V6.1 test environments
- ▶ EJB test client for easy testing of EJBs
- ▶ Support for annotation-based development
- ▶ Web 2.0 development features for visual development of responsive Rich Internet Applications with Asynchronous JavaScript and XML (AJAX) and Dojo
- ▶ 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 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
- ▶ Crystal Reports for developing visual data reports
- ▶ Component test automation tools to automate creating tests and building and managing test cases

8.3.3 WebSphere rapid deployment

WebSphere rapid deployment is a set of tools and capabilities (non-graphical command line interface) included in the WebSphere Application Server V7.0 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 the following 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. The developer can then concentrate on the Java source code rather than metadata files.

Deployment automation is where applications 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, which 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. There are limitations and rules you need to follow when working with these utilities, and most of the time you are significantly more productive using a full-blown development environment, such as Rational Application Developer Assembly and Deploy V7.5 or Rational Application Developer for WebSphere Software V7.5.

Note: You can use rapid deployment tools for packaging applications at J2EE 1.3 or 1.4 specification levels. The rapid deployment tools do not support Java EE 5 nor J2EE 1.2 specification level.

8.3.4 Which tools to use

Which tool you choose depends on your requirements. If you need to develop, deploy, and test applications on WebSphere Application Server V7.0 for fast turnaround times, choose Rational Application Developer Assembly and Deploy V7.5.

If you are developing applications that require the new features only available in the WebSphere Application Server V7.0, such as JSR 286 events, you have to use Rational Application Developer for WebSphere Software V7.5. It is feature-rich and has lots of productivity-enhancing features not found in the Rational Application Developer Assembly and Deploy V7.5.

8.4 Naming conventions

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

8.4.1 Naming for applications

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 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 the 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.

8.4.2 Naming for resources

When naming resources, associate the resource to both the application using it and the physical resource to which it refers. As an example 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 Integrated Solutions Console sorts resources by name, you might want to include the name of the application first in the resource, such as in `OrderApplication_Order`. This gives you the possibility to sort your resources according to the application using them.

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

8.5 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 Rational Application Developer Assembly and Deploy V7.5 or Rational Application Developer for WebSphere Software V7.5 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 the following advantages:

- ▶ 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 Application Developer Assembly and Deploy V7.5 and Rational Application Developer for WebSphere Software V7.5 support Rational ClearCase, CVS, and Subversion as SCM systems.

8.5.1 Rational ClearCase

Rational ClearCase organizes its code repositories as Versioned Object Bases (VOBs). VOBs contain versioned file and directory elements. Users of Rational ClearCase are organized according to their roles. Each user has their own view of the data that is in the VOB on which they are working. Rational ClearCase tracks VOBs, views, and coordinates the checking in and checking out of VOB data to and from views.

As the role-based model suggests, Rational ClearCase is 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 as follows:

- ▶ Rational Enterprise Suite Tools
- ▶ Rational Unified Process
- ▶ 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.

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

The software is scalable. Rational ClearCase LT is a scaled down version of Rational ClearCase for small-to medium-sized teams. It can be upgraded seamlessly to Rational ClearCase as a user's needs change. Additionally, you may use a Rational ClearCase MultiSite® add-on to support use of the software in geographically dispersed development teams. In short, although Rational ClearCase is an SCM system, it is also an integral part of the Rational toolset and RUP. For more information about Rational software, see the following Web page:

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

8.5.2 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 the following Web page:

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

8.5.3 Subversion

Subversion is a free open source version control system that tracks the entire file system and files. It versions directories and individual files and stores them into a repository. Each time a change is made to the files or directories, the change will be recorded in the repository. You can track the history of changes on files or directories by reviewing the log files maintain by Subversion. Each file or directory has a corresponding log file. Subversion is easy to configure and offers rich graphical and command line interfaces to manage files and directories. For more information about Subversion, see the following Web page:

<http://subversion.tigris.org/>

8.5.4 Choosing an 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), what the SCM and development process requirements are at present, and what those requirements will be in the future. If a team uses CVS or Subversion and an existing, successful, development process, Rational 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 Clear 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 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 and Subversion do not shield the user from this.

Cost

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

Change management process

If the development team uses CVS or Subversion rather than Rational ClearCase, the team does not get a prescribed change management process for CVS and Subversion such as the UCM. If 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, Subversion, or Rational ClearCase LT are good choices. As team size and complexity grows, Rational ClearCase and 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.

8.6 Automated build process

The major impetus 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 as follows:

- ▶ Failures 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 areas is unavailable.
- ▶ You cannot reproduce a problem on production because you do not know what version of files are in production at the moment.

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 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 perform the following tasks:

- ▶ Install and uninstall applications.
- ▶ Run EJB 1.x, 2.x and 3.x 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 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 page:

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

Another product to consider is IBM Rational Build Forge®. IBM Rational Build Forge is a product that provides a framework to automate the entire process end-to-end. IBM Rational Build Forge automates individual tasks, but also the hand-offs among the various steps in the process. IBM Rational Build Forge was designed to integrate existing scripts and tools, so there is no need to replace your pre-existing assets. It offers a comprehensive Application Development Process Management solution that provides complete management and control of the application development life cycle. IBM Rational Build Forge automates, standardizes, and optimizes complex processes, integrating diverse tool sets to deliver a repeatable and reliable application development life cycle process.

8.7 Automated deployment process

Automating the application deployment is something to consider if it is done more than one time. Successful automation will provide an error free and consistent application deployment approach. Most of the application deployment not only involves installing the application itself, but it also needs to create other WebSphere objects, configure the Web servers, file systems and others. These tasks can be automated using shell scripting depending on the operating systems, Java Common Language (JACL) and Jython. For more information, see the IBM developerWorks article *Sample Scripts for WebSphere Application Server Version 5 and 6* at the following Web page:

<http://www.ibm.com/developerworks/websphere/library/samples/SampleScripts.html>

8.8 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 since they are not automatically generated, so someone has to create the scripts at least one time. It is possible to create a script to cover all functions in your application, but it would be 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 the article *Rational Functional Tester* at the following Web page:

<http://www.ibm.com/software/awdtools/tester/functional/>

8.9 Test environments

Before moving an application into production, it is 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 creating the test cases and with sufficient developed functionality in the application, start testing.

Figure 8-2 shows an overview of a recommended test environment setup.

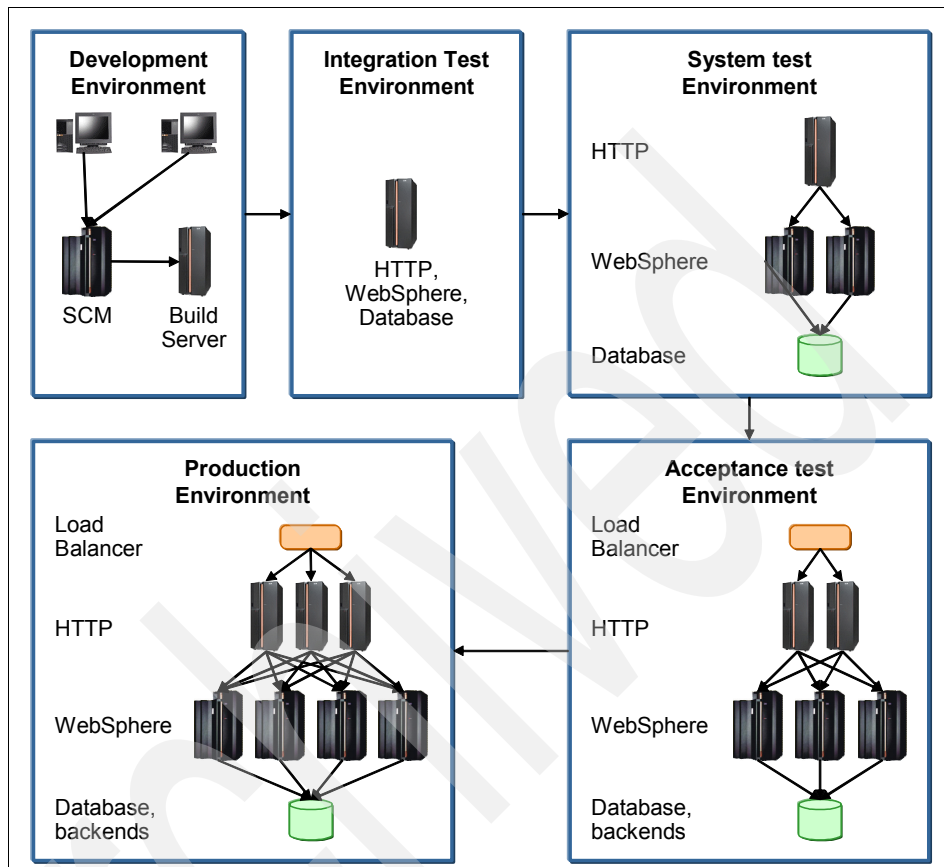


Figure 8-2 Test environments

Whether you choose to use some of these test environments, all of them, or use 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 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 desktop machine.

When each developer has committed their code to the integration stream in the SCM system, a development lead or integration team usually performs a clean build of the whole application, bringing together code developed by different developers. This is usually done on a special build server and is controlled by automatic build scripts. See 8.6, "Automated build process" on page 294. This server might need to have a copy of the Rational Application Developer for Assembly and Deploy V7.5 or Rational Application Developer for WebSphere Software V7.5 installed.

The development team should also create a Build Verification Test process (see 8.8, "Automated functional tests" on page 296), where each new build is executed before making the build available to the team. A Build Verification Test covers test cases or scenarios that verify that critical paths through the code are operational. Build Verification Test scripts are often controlled by JUnit.

It is every developer's responsibility to perform basic code profiling. By using the profiling tools in Rational Application Developer Assembly and Deploy V7.5 or Rational Application Developer for WebSphere Software V7.5, 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 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 five 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. If 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 virtualization 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-functional 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 specifications such as determining whether the system executes the business rules defined, whether the user interface shows 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 Application Server, 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 synchronized.

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 the same configuration settings as the production environment.

The purpose of the acceptance test environment is to give the operations team a chance to familiarize themselves with the application and its procedures (such as installation, backup, failover, and so on). It also provides an opportunity to test unrelated applications together, because previous environments 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. This can be because the production database is much larger than the databases used in the acceptance test environment. It is important that the test databases be populated with representative data. Ultimately, a copy of the production database should be used, but this may not be 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.

8.10 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 Rational Application Developer Assembly and Deploy V7.5 or Rational Application Developer for WebSphere Software V7.5. Other settings, such as the JVM maximum heap size and database connection pool size, are stored in the WebSphere Application Server configuration repository and modified using the WebSphere administrative tools. Finally, 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 after deploying the application.

8.10.1 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 depending on the 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.

8.10.2 Managing configuration setting

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 Java EE specification 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 the most common way of configuring internal settings for Java applications. It is an easy and straightforward method to accomplish the task. 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. The 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 code snippet in Example 8-1.

Example 8-1 Loading a property file using the class loader code snippet

```
Properties props = new Properties();
InputStream in =
MyClass.class.getClassLoader().getResourceAsStream("my.properties");
props.load(in);
in.close();
```

Property files packaged in a JAR file in the EAR file are a good solution for property files that should not be modified after the application has been deployed (the application-specific category described in 8.10.1, “Classifying configuration settings” on page 301).

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 either make the folder available on the class path for the application (and use the code snippet in Example 8-1) or you use an absolute path name and the code snippet in Example 8-2 (assuming the file to load is `/opt/apps/OrderApp/my.properties`).

Example 8-2 Absolute path name code snippet

```
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.

A lesser known, but 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 Integrated Solutions 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 use that `InputStream` as input to the `java.util.Properties` class to load the property files.

This approach to access property files is also more Java EE compliant because it does not rely on the `java.io` package for file access, which is prohibited according to the Java EE specification.

The method also gives you the opportunity to load the property files using HTTP and FTP. This allows you to set up an HTTP server serving properties files from a central location.

Unless you are using the previous technique with the HTTP or FTP protocol, it is convenient to manage all property files in a central location, on the deployment manager. However, property files 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. Because this folder is not known to the WebSphere Application Server infrastructure, it will not happen automatically when the contents are changed. You need to force a synchronization with the nodes. This propagates the property files to the `<profile_home>\config\cells\<cell_name>\appconfig` directory on each node. You can include that folder on the class path 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 of the setting referred 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 Application Server, it is distributed to each node in the cluster using the WebSphere Application Server file transfer mechanism. At each node, the EAR file is expanded and laid out on the file system so that WebSphere Application Server can execute it. 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 it 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 Application Server does not fully expand the contents of the EAR file to the file system on the deployment manager (it only extracts from the EAR file the deployment descriptors needed to configure the application in the WebSphere Application Server 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 Application Server 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 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.

8.11 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, it is also the responsibility of the developer. 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 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. 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 Application Server 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 require training for users to learn a new user interface or navigation system.

We do not go into depth on this subject, but suggest investigating the IBM developerWorks article *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. It is available from the following Web page:

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

8.12 Mapping application to application servers

A question that often arises when deploying multiple applications to WebSphere Application Server 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 consider so that you can make the right decision for your particular environment.

The advantages of deploying each application to its own application server are as follows:

- ▶ 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 Application Server 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 that multiple applications can be deployed to the same application server and use the same heap, if 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 Application Server rollout update feature, which requires the application server be stopped. Although it can be used 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, this is not a problem because applications often use, for example, log4j (which is configurable) to perform logging instead of plain system out print statements.
- ▶ It is simple to diagnose problems as there is only one application in the JVM.

The advantages of deploying multiple applications to the same application server are as follows:

- ▶ 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 it 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 Application Server 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.

8.13 Planning checklist for applications

Table 8-1 lists items to consider as you plan. Following the table are additional resources that can help you.

Table 8-1 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.
Determine where applications will be deployed. (all on one server?)

Resources

The WebSphere Application Server Information Center contains a lot of useful information. For a good entry point to information about application development and deployment, go to the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6topdeveloping.html>

For detailed information about application development using Rational Application Developer, refer to *Rational Application Developer V7 Programming Guide*, SG24-7501.

Archived

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 V7.0” on page 312
- ▶ “Administrative security” on page 314
- ▶ “WebSphere administration facilities” on page 314
- ▶ “Automation planning” on page 320
- ▶ “Configuration planning” on page 321
- ▶ “Change management topics” on page 326
- ▶ “Serviceability” on page 329
- ▶ “Planning checklist for system management” on page 334

9.1 What is new in V7.0

The following list highlights the changes in administrative tools and processes in WebSphere Application Server V7.0:

- ▶ Simplified administration of multiple Base profiles with an administrative agent

Prior to WebSphere Application Server V7.0, whether a Base or Express installation, the WebSphere Administrative Console resided in the default WebSphere Application Server. If you created multiple profiles within the Base, each profile had a WebSphere Application Server instance and the WebSphere Administrative Console resided in each WebSphere Application Server instance. To administer each WebSphere Application Server instance, you had to log into each individual WebSphere Administrative Console.

In WebSphere Application Server V7.0, you can use the administrative agent to centrally manage all WebSphere Application Server instances within the same physical server created by Base profiles. This is done through the Integrated Solutions Console, where you select which WebSphere Application Server instance to administer. This new feature reduces the individual WebSphere Application Server footprint associated with administration. It also simplifies administration of multiple Base profiles WebSphere Application Server instances.

We discuss this more in 3.1.7, “Administrative agent” on page 63.

- ▶ Centralized administration of multiple WebSphere domains with a job manager

In WebSphere Application Server V7.0, the job manager has been introduced as a loosely coupled management model based on asynchronous implementation, so that it provides centralized administration for multiple deployment managers and multiple Base profiles (through an administrative agent). It centralizes the distribution of applications, applications updates, and WebSphere Application Server configuration updates across a large number of WebSphere administrative domains. This dramatically increases the scale of administration for WebSphere Application Server implementations. The job manager does not replace deployment manager but augments it.

We discuss this more in 3.2.4, “Flexible management” on page 78.

- ▶ New runtime provisioning service

The new runtime provisioning service commissions only those components in the Java Virtual Machine (JVM) that are required by the installed applications. At application installation time, WebSphere Application Server examines the application and creates an application-specific activation plan. At WebSphere Application Server startup, it only starts those components that are detailed in

the activation plan. This reduces the WebSphere Application Server footprint and the resources needed for a given application portfolio.

We discuss this more in 3.1.14, “Intelligent runtime provisioning” on page 72.

- ▶ Administrative scripting enhancement

Administrative scripting wsadmin has been enhanced in WebSphere Application Server V7.0. It provides a set of sample Jython libraries to accelerate automation implementations. It introduces new administrative tasks, improves help with Command Assistant, and supports wildcard usage.

- ▶ Properties file based configuration

WebSphere Application Server V7.0 supports properties file based configuration, so that you can extract a WebSphere Application Server configuration from a server into a property file. Manipulating the content in the property file, such as the environment name, allows you to use the modified property file to configure the same WebSphere Application Server configuration in another server. Alternatively, you can also alter the value of a field and use the modified property file to make the new WebSphere Application Server configuration change in the same server.

- ▶ Manage WebSphere DataPower through the Integrated Solutions Console

In WebSphere Application Server V7.0, you can manage WebSphere DataPower through the Integrated Solutions Console. It provides flexibility to have all administration in one place.

We discuss this more in 2.5, “WebSphere DataPower” on page 41.

- ▶ Fine-grained administrative security

WebSphere Application Server V7.0 supports fine-grained security control, which expands the administration scope. We can restrict access based on the roles on the cell, node, server, cluster, application and node group level. This capability is valuable in big organizations where you have different teams responsible for supporting different applications.

- ▶ Centralized installation manager

The centralized installation manager (CIM) allows you to install and uninstall WebSphere Application Server binaries and maintenance patches from a centralized location (the deployment manager) to any servers in the network. This capability allows flexibility so that you can perform installation and uninstallation from a central place.

We discuss this more in 3.1.13, “Centralized installation manager” on page 72.

9.2 Administrative security

We suggest enabling administrative security to prevent unauthorized access to the administrative tasks. Enabling administrative security only secures administration tasks, not applications.

After administrative security is enabled, there is a security check when the Integrated Solutions Console or other administrative facilities are accessed. The security check makes sure that the accessing user is authenticated and has been mapped to one of the console security roles. Depending on the console role to which the user is mapped, different functions will be available.

Proper planning for system management includes identifying the people who will need access as well as the level of access to the administrative tools. You will need to consider and design a set of groups, users, and roles that fit your needs.

You should review the available roles and access levels in the WebSphere Application Server Information Center, at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.doc/info/ae/ae/csec_globalserver.html

WebSphere Application Server 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 set of XML files and be mapped to the Administrator role. Additional users can be added after profile creation using the administrative tools.

9.3 WebSphere administration facilities

WebSphere Application Server V7.0 provides a variety of administrative tools for configuring and managing your runtime environment, as follows:

- ▶ Integrated Solutions Console

The Integrated Solutions Console is a browser-based client that uses a Web application running in the Web container to administer WebSphere Application Server.

- ▶ WebSphere scripting client (wsadmin)

The wsadmin client is a non-graphical scripting interface that can be used to administer WebSphere Application Server from a command line prompt.

It can connect to WebSphere Application Server using one of the two communication mechanisms:

- Simple Object Access Protocol (SOAP) by communicating with the embedded HTTP server in the Web container.
 - Remote Method Invocation (RMI), to communicate with the administrative services.
- ▶ Task automation with Ant
- With Another Neat Tool (Ant), you can create build scripts that compile, package, install, and test your application on WebSphere Application Server.
- ▶ Administrative applications
- You can develop custom Java applications that use the Java Management Extensions (JMX™) based on the WebSphere Application Programming Interface (API).
- ▶ Command line utilities
- WebSphere Application Server provides administrative utilities to help manage your environment, including the following features:
- Called from a command line.
 - Can be used to perform common administrative tasks such as starting and stopping WebSphere Application Server, backing up the configuration and so on.
 - Work on local servers and nodes only, including the deployment manager.

The combination of administrative tools you employ ultimately depends on the size and complexity of your runtime environment. Where you have few resources, but many tasks, we suggest the use of automation and scripts. Where you have multiple administrators that will perform different tasks, you might want to consider defining different access control roles. This is important where you want non-administrators to be able to perform limited roles such as application deployment.

Updates to configuration done through Integrated Solutions Console or the wsadmin client are kept in a private temporary area called a workspace. These changes are not copied to the configuration repository until an explicit save command is issued. The workspace is kept in the <profile_root>\wstemp directory. The use of a workspace allows multiple clients to access the configuration concurrently. Care must be taken to prevent change conflicts. Clients will usually detect such conflicts and allow you to control how to handle them. For example, the wsadmin client has a property called setSaveMode that can be set to control the default save behavior in case of conflict.

9.3.1 Integrated Solutions Console

The Integrated Solutions Console connects to a running stand-alone server or, in a distributed environment, to a deployment manager. In WebSphere Application Server V7.0, it also connects to an administrative agent and a job manager.

Non-secure administration access

If administrative security is not enabled, the Integrated Solutions Console is accessed with a Web browser through the following URL:

```
http://<hostname>:<WC_adminhost>/ibm/console/unsecureLogon.jsp
```

You can gain access to the console without entering a user name. If you do enter a name, it is not validated 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 Integrated Solutions Console is accessed with a Web browser through the following URL (note the use of https:// versus http://):

```
https://<hostname>:<WC_adminhost>/ibm/console/Logon.jsp
```

You must enter an authorized user ID and password to log in. The actions that you can perform within the console are determined by your role assignment.

9.3.2 WebSphere scripting client (wsadmin)

The WebSphere scripting client (`wsadmin`) provides the ability to execute scripts. You can use the `wsadmin` tool to manage a WebSphere Application Server V7.0 installation and configuration. This tool uses the Bean Scripting Framework (BSF), which supports a variety of scripting languages to configure and control your WebSphere Application Server installation and configuration.

The `wsadmin` launcher makes Java objects available through language-specific interfaces. Scripts use these objects for application management, configuration, operational control, and for communication with Manageable Beans (MBeans) running in WebSphere server processes.

With the release of WebSphere Application Server V7.0, the stabilized process for the Java Application Control Language (Jacl) syntax associated with `wsadmin` has been announced. This means that Jacl syntax for `wsadmin` will continue to remain in the product and there is no plan to deprecate or remove this capability in a subsequent release of the product. But future investment will be focused on Jython.

You can run the wsadmin tool in interactive and unattended mode. Use the wsadmin tool to perform the same tasks that you perform with the Integrated Solutions Console.

WebSphere Application Server V7.0 adds command assistance in the Integrated Solutions Console that maps your administrative activities to wsadmin scripting commands written in Jython. These commands can be viewed from the Integrated Solution Console, and if you want, you can log the command assistance data to a file. You can also allow command assistance to emit JMX notifications to Rational Application Developer Assembly and Deploy V7.5. Rational Application Developer Assembly and Deploy V7.5 has Jython development tools that help you develop and test Jython scripts.

9.3.3 Task automation with Ant

WebSphere Application Server V7.0 provides a copy of the Ant tool and a set of Ant tasks that extend the capabilities of Ant to include product-specific functions. Ant has become a popular tool among Java programmers.

Apache Ant is a platform-independent, Java-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 WebSphere Application Server. It integrates with wsadmin scripts and uses Ant as their invocation mechanism.

For information about Apache Ant, see the following Web page:

<http://ant.apache.org>

9.3.4 Administrative programming

WebSphere Application Server V7.0 supports access to the administrative functions through a set of Java classes and methods. You can write a Java application that performs any of the administrative features of the WebSphere Application Server administrative tools. You can also extend the basic WebSphere Application Server administrative system to include your own managed resources.

JMX, a Java specification part of Java Enterprise Edition (Java EE), and the specification for the Java EE Management API (JSR-077) are the core of the WebSphere Application Server V7.0 management architecture. For information about JMX, see the following Web page::

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/cxml_javamangementx.html

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.

9.3.5 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 Integrated Solutions 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 when no profile option is specified.

9.3.6 Administrative agent

The administrative agent provides a single interface administration for multiple unfederated WebSphere Application Servers in the same physical server. This will involve creating an administrative agent profile and registering the node that you would like the administrative agent to manage using the `registerNode` command. There is also a `deregisterNode` command to undo the use of the administrative agent.

Non-secure administration access

If administrative security is not enabled, the Integrated Solutions Console is accessed with a Web browser through the following URL:

```
http://<hostname>:<WC_adminhost>/ibm/console/profileSelection.jsp
```

Select a node that you want to manage. You can gain access to the console without entering a user name. If you enter a name, it is not validated and is used exclusively for logging purposes, 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 Integrated Solutions Console is accessed with a Web browser through the following URL (note the use of https:// versus http://):

```
https://<hostname>:<WC_adminhost_secure>/ibm/console/profileSelection.jsp
```

Select a node that you want to manage. You must enter an authorized user ID and password to log in. The actions that you can perform within the console are determined by your role assignment.

For more information on the administrative agent, see the following Web page::

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/cagt_adminagent.html

9.3.7 Job manager

The job manager allows the management of multiple WebSphere Application Server domains (multiple deployment managers and administrative agents) through a single administration interface. It involves creating a job manager profile and using the `wsadmin registerWithJobManager` command to register the deployment manager or administrative agent with the job manager.

For more information on the job manager, see the following Web page::

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/tagt_jobmgr.html

9.4 Automation planning

To emphasize the need for automated administration, consider the fact that companies typically have multiple WebSphere Application Server environments to support activities in the different phases of the software development life cycle. Each of these environments is susceptible to being the target of the same type of administrative tasks. What if there was a way to do a task manually once only, and have subsequent requests be done automatically or with a lot less effort? That is the main goal of automation.

Automating common procedures and actions is one of the keys to maintaining a stable and efficient WebSphere environment. You can reduce the possibility of human error by eliminating human intervention in complicated tasks or mundane procedures that are prone to mistakes. Automating WebSphere Application Server installation and configuration also provides an administrator the opportunity to schedule recurring maintenance and backup procedures, and any other types of administrative tasks.

Every action that you can perform manually using the Integrated Solutions Console can be automated using WebSphere Application Server's wsadmin tool and command line utilities.

- ▶ Installation response files
 - Allow you to specify installation options once and use them for multiple installations of WebSphere Application Server.
 - Enable silent execution mode.
- ▶ Command line utilities
 - Shell scripts on UNIX or batch files on Windows
 - Run from standard shell or command prompt
 - Allow you to control different aspects of the WebSphere Application Server environments
- ▶ WebSphere Ant tasks
 - Facilitate build and deploy processes to WebSphere Application Server.
- ▶ JMX framework
 - Provides standards-based capabilities to control and manage a WebSphere Application Server.
 - Allows you to create custom Java clients to access managed resources.
- ▶ wsadmin scripting tool
 - Scripting interface that allows you to execute administrative commands interactively or by running a script of commands.

- ▶ Installation factory

Complementary tool to combine the installation of WebSphere Application Server with maintenance packages and fix packs in a single step.

Although scripting requires up front development costs, in the long term it provides savings through automation and increases reliability. In addition, in many organizations, the Integrated Solutions Console is prohibited by security policy and infrastructural constraints. Scripted administration provides an alternative way to manage the WebSphere Application Server environment.

Automation is not necessary if the task is only a one off.

9.5 Configuration planning

This section describes global configuration planning topics. Configuring and managing the WebSphere Application Server runtime environment can be complex. This section addresses the following items to consider at the initial installation time:

- ▶ Configuration repository location and synchronization
- ▶ Configuring application and application server startup behaviors
- ▶ Custom application configuration templates
- ▶ Planning for resource scope use

9.5.1 Configuration repository location and synchronization

WebSphere Application Server uses one or more configuration repositories to store configuration data. In a stand-alone server environment, one repository exists within the application server profile directory structure. In a distributed server environment, multiple repositories exist. The master repository is stored within the deployment manager profile directory structure. Each node also has a repository 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.

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.

9.5.2 Configuring application and application server startup behaviors

One feature of WebSphere Application Server is the ability to manage the startup of applications and application 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.

- ▶ Startup order setting

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 important for applications that have been split into sub-applications that need to start in a certain order due to dependencies between them.

- ▶ Launch application before server completes startup setting

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.

- ▶ Create MBeans for resources setting

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.

Access these settings in the Integrated Solutions Console by navigating to **Applications** → **Application Types** → **WebSphere enterprise applications** → **<your_application>** → **Startup behavior**.

The Parallel start setting for an WebSphere Application Server lets you specify whether to have the server components, services, and applications in an application server start in parallel rather than sequentially. This can shorten the startup time for a server.

The Parallel start setting can affect how an application server starts. Access this setting by navigating to **Servers** → **Server Types** → **WebSphere application servers** → **<your_server>**.

The deployment manager, node agents, and application servers can start in any order they are discovered, with the exception that the node agent must start before any application server on that node. Communication channels are established as they start up 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** → **Application Types** → **WebSphere enterprise applications** → **<application_name>** → **Target specific application status** and disable auto start for the application.

9.5.3 Custom application 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. To propagate the server configuration across cell boundaries, the server configuration must be exported to a configuration archive, after which it can be imported to another cell.

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 more efficient to create a custom template and use that template to create your WebSphere Application Server. 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.

9.5.4 Planning for resource scope use

Resource scope is a powerful concept to prevent duplication of resources across lower-level scopes. For example, if a data source can be used by multiple servers in a node, 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. 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 Integrated Solutions Console. Use Rational Application Developer Assembly and Deploy V7.5, 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 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 WebSphere Information Center at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/cdat_testcon.html

9.6 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 V7.0 operational environment. Topics are as follows:

- ▶ Application update
- ▶ Changes in topology
- ▶ Centralized installation manager (CIM)

9.6.1 Application update

WebSphere Application Server V7.0 permits fine-grained updates to applications. It allows application components to be supplied and the restart of only required 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:

- ▶ Integrated Solutions 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 Integrated Solutions Console wizard.
- ▶ 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 installed 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 z/OS console **Modify** command to pause the listeners for an application server, perform the application update, and then resume the listeners. If you use this technique, you do not have to stop and then start the server to perform the application update.

9.6.2 Changes in topology

In a distributed server environment, the deployment manager node contains the master configuration files. Each node has its required configuration files available locally. 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

You can turn on automatic synchronization using the Integrated Solutions Console. The default file synchronization interval is 60 seconds and starts when the application server starts.

- ▶ Manual

You can perform manual synchronization using the Integrated Solutions Console, the wsadmin tool, or using the **syncNode** command located in the <install_root>/bin directory of the note being synchronized.

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.

9.6.3 Centralized installation manager (CIM)

The CIM allows you to install and uninstall WebSphere Application Server binaries and maintenance patches from a centralized location (the deployment manager) to any servers in the network.

CIM is a new feature added in WebSphere Application Server V7.0. CIM is supported on the following operating systems:

- ▶ Unix-based systems
- ▶ Windows
- ▶ IBM System i

Using CIM, the following tasks can be performed:

- ▶ Installation of WebSphere Application Server V7.0 and creation of a managed profile that gets federated to the deployment manager automatically.
- ▶ Installation of the Update Installer for WebSphere Application Server V7.0.
- ▶ Installation of a customized installation package (CIP) as created using the Installation Factory.
- ▶ Central download of interim fixes and fix packs from the IBM support side. The downloaded packages are stored in the installation manager's repository.
- ▶ Installation of fixes and fix packs on nodes within the deployment manager's cell.

Tip: When installing CIM make sure that the Windows firewall allows connections from the CIM

The repository for the CIM can either be created during the installation of WebSphere Application Server Network Deployment V7.0 or afterwards using the Installation Factory. The installation using the CIM provides a good approach to perform centralized remote installations and upgrades. The drawback is that you cannot control the naming of the profiles created when performing standard installation. This problem can be avoided by using custom installation packages created through the installation manager.

For more details about CIM, refer to the IBM White Paper *Centralized Installation Manager for IBM WebSphere Application Server Network Deployment Version 7.0*, available on the following Web page:

<http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-dist&topic=was-nd-dist-dw-cim1>

Also see the Information Center at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.installation.nd.doc/info/ae/ae/tins_cim.html

9.7 Serviceability

A major challenge of problem management is dealing with unanticipated problems. It is much like detective work: find clues, make educated guesses, verify suspicions, and so on. The most important skills are common sense, focus, thoroughness, and rigorous thinking. A proactive approach to problem management is always the best. This section outlines general practices to follow.

Perform the following checks to avoid issues with the runtime environment:

- ▶ Check that you have the necessary prerequisite software up and running.
- ▶ Check that the proper authorizations are in place.
- ▶ Check for messages that signal potential problems. Look for warnings and error messages in the following sources:
 - Logs from other subsystems and products, such as TCP/IP, RACF, 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.

9.7.1 Log 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 Integrated Solutions 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 with the following names:

- `<profile_home>/logs/<server_name>/native_stderr.log`
- `<profile_home>/logs/<server_name>/native_stdout.log`

► IBM service log (activity.log)

The service log is a special log 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:

- Log Analyzer tool

We suggest 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.

- Showlog tool

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 `<profile_home>/logs/` directory.

9.7.2 Fix management

Applying regular fixes is one of the key factors to reduce the probability and impact of problems. A fix plan establishes how you will do this on a regular basis. In addition to regular scheduled fixes, you may also need to perform emergency changes or fixes to a system in response to a newly-diagnosed problem. The emergency fix plan outlines how to do this safely and effectively. Overall, the best approach is to have a strong fix plan that outlines regular small fix updates and reasonable re-testing before each fix. For available fixes, see the WebSphere Application Server support at the following Web page:

<http://www.ibm.com/software/webservers/appserv/was/support/>

9.7.3 Backing up and restoring the configuration

Back up the WebSphere Application Server configuration to a 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.

We recommend running **backupConfig** utility before each major change to the WebSphere Application Server configuration.

9.7.4 MustGather documents

MustGather documents provide instructions on how to start troubleshooting a problem and what information to provide to IBM Support if opening a Problem Management Report (PMR). MustGather documents can be accessed from within IBM Support Assistant (ISA) or on the IBM Support Web site. See an example at the following Web page:

http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=troubleshooting&uid=swg21201625&loc=en_US&cs=utf-8&lang=en

On the IBM Support site, many MustGather documents are categorized as troubleshooting and analyzing data. The troubleshooting document is what you check before you decide that you need to go through the MustGather document. The analyzing data document gives you pointers for how to interpret the information that you have just collected from the MustGather document.

A majority of MustGather documents for WebSphere Application Server now have a corresponding AutoPD script in ISA. You can either follow the steps from the MustGather document by hand, or run the AutoPD script, which will do the work more or less automatically.

9.7.5 IBM Support Assistant

IBM Support Assistant (ISA) improves your ability to locate IBM Support, development and educational information through a federated search interface (one search, multiple resources). It provides quick access to the IBM Education Assistant and key product education roadmaps, and simplifies access to the following IBM resources through convenient links:

- ▶ product home pages
- ▶ product support pages
- ▶ product forums or newsgroups

In addition, problems can be submitted to IBM Support by collecting key information, then electronically creating a PMR from within IBM Support Assistant.

IBM Support Assistant includes a support tool framework allowing for the easy installation of support tools associated with different IBM products and a framework for IBM software products to deliver customized self-help information into the different tools within it. The workbench can be customized through the built-in Updater feature to include the product plug-ins and tools specific to the environment.

For more information about ISA, see the following Web page:

<http://www.ibm.com/software/support/isa>

9.7.6 Information Center

The WebSphere Application Server V7.0 Information Center includes a troubleshooting section. For more information, see the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/welc6toptroubleshooting.html>

9.8 Planning checklist for system management

Table 9-1 lists a summary of items to consider as you plan and additional resources that can help you.

Table 9-1 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, Integrated Solutions 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.
Define a strategy for automation.
Consider whether to use automatic or manual synchronization to nodes.
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. It also includes strategies for changes in cell topology and updates to WebSphere Application Server binaries.
Create a strategy for problem management. Identify a location and naming convention for storing WebSphere Application Server logs. Configure the processes to use those locations.
Create a strategy for backup and recovery of the installation and configuration files.

Resources

The WebSphere Application Server Information Center contains a lot of useful information about system management. For a good entry point to system management topics, see the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6topmanaging.html>

Messaging

In this chapter, we discuss planning for a WebSphere Application Server V7.0 environment that uses messaging facilities. This chapter asks the following questions in the sections that follow:

- ▶ “Messaging overview: What is messaging?” on page 336
- ▶ “What is new in V7.0” on page 336
- ▶ “Messaging options: What things do I need?” on page 337
- ▶ “Messaging topologies: How can I use messaging?” on page 340
- ▶ “Messaging features: How secure and reliable is it?” on page 350
- ▶ “Planning checklist for messaging” on page 355

This chapter briefly describes the concepts required to understand messaging. The WebSphere Application Server Information Center contains a lot of useful additional information. For a good entry point to messaging topics, see the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6tech_msg.html

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 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, who 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 popular facility for exchanging data between applications and clients of different types. It is also an excellent tool for communication between heterogeneous platforms. WebSphere Application Server recognizes the power of messaging and implements a powerful and flexible messaging platform within the WebSphere Application Server environment, called the service integration bus.

10.2 What is new in V7.0

WebSphere Application Server V7.0 introduces some improvements to the service integration bus. We briefly describe these here. For those new to WebSphere Application Server messaging, we explain the concepts later in this chapter.

- ▶ Additional option for Message Driven Bean (MDB) in a cluster

The default option is if MDBs are deployed to a cluster bus member, only the MDB endpoints in servers that have a messaging engine started locally are eligible to be driven by available messages. In V7.0, there is an option to allow a MDB to process messages whether or not the server also hosts a running messaging engine. This allow the full processing power of MDBs in a cluster.

- ▶ Message visibility

In WebSphere Application Server V7.0, a consumer can choose to turn on the option of accessing all queue points of a queue. This means that no messages are left on an unattended queue point. However, there is a performance cost in scanning all queue points for messages.

- ▶ Enhancements in configuring cluster bus members

In WebSphere Application Server V7.0, the Integrated Solutions Console provides consumability improvements that guide you through the configuration of a cluster bus member. It provides you with a pattern-based approach. You select from the patterns, and the Integrated Solutions Console makes the configuration corresponding to the pattern.

- ▶ Improved administrative control of MDBs

In WebSphere Application Server V7.0, the Integrated Solutions Console provides visualization between MDBs and service integration bus destinations.

- ▶ Thin Java Message Service (JMS) client

WebSphere Application Server V7.0 updated the functionality of the thin JMS client. You can use the thin JMS client as opposed to a full WebSphere Application Server client installation. It supports the Open Service Platform as instantiated by the Eclipse Rich Client Platform (RCP) level 3.2.

There are a significant number of other improvements and new additions to messaging in WebSphere Application Server V7.0, including performance improvements, additional clustering options, security panels, and so forth. For a full list, see the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.multiplatform.doc/concepts/cjover_sib.html

10.3 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 in the following instances:

- ▶ Messaging provider standards
- ▶ Choosing a messaging provider

10.3.1 Messaging provider standards

To implement messaging within your application, either as a message producer, consumer, or both, your application needs to communicate with a messaging provider. Examples of messaging providers include the default messaging provider in WebSphere Application Server, WebSphere MQ, Oracle Enterprise Messaging Service, SonicMQ, and many others.

Your application code can interact with these providers in a number of ways. We recommend using the JMS API, but you could also use provider-specific client libraries, or the J2EE Connector Architecture API (JCA). This section briefly discusses the first two options.

Java Messaging Service

Java Messaging Service (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 V7.0 supports version 1.1 of the specification, which forms part of the overall Java EE 5 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 suggest using JMS in preference to anything else when writing an application to run within WebSphere Application Server. We suggest this for the following reasons:

- ▶ It is a tried-and-tested, consistent, and non-proprietary API that has been around for enough time to have plenty of skilled resources available.
- ▶ Applications that use it remain portable across many messaging providers.
- ▶ The API, while specific to messaging, has been expanded to support many message types and architectures, providing flexibility and versatility in the vast majority of applications.

Important: For the rest of the chapter, we assume that JMS is the chosen method to access the messaging middleware provider.

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, with the following important exceptions:

- ▶ They are proprietary and do not usually conform to any open standard.
- ▶ The use of the client libraries renders your applications non-portable across enterprise systems, and probably non-portable 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 suggest 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.

10.3.2 Choosing a messaging provider

WebSphere Application Server supports several JMS messaging providers:

- ▶ WebSphere Application Server default messaging provider
This fully featured messaging provider comes free with WebSphere Application Server. It is a robust and stable messaging platform that can handle point-to-point queues, topics in a publish-subscribe environment, and Web service endpoints.
- ▶ WebSphere MQ messaging provider
WebSphere MQ is the premier messaging middleware provided by IBM. We suggest 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 the WebSphere Application Server default messaging provider and WebSphere MQ messaging provider.

- ▶ WebSphere Application Server V5 default messaging provider

The WebSphere Application Server V5 default messaging provider is supported for migration purposes only.

10.4 Messaging topologies: How can I use messaging?

Choosing a topology depends largely on the answers to questions about the topology of the application and your own messaging requirements. Some of the more important questions are as follows:

- ▶ 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 outline what topology best fits 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, even the simplest topology for messaging, it is important that you understand how the default messaging provider handles messages.

10.4.1 Default messaging provider concepts

This section describes some concepts briefly in order to understand the basic intent of the topologies.

Service Integration Bus

The Service Integration Bus (SIBus, or just the bus) provides a transport mechanism for WebSphere Application Server default messaging provider. It is a group of interconnected WebSphere Application Servers and a cluster of WebSphere Application Servers 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.

Message Engine

The Messaging Engine (ME) is a component of WebSphere Application Server. It communicates messages to destinations, in cooperation with the other MEs 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

A foreign bus is an external messaging product that is either another SIBus or a WebSphere MQ network. You can set up a link to it so that messages traverse from one bus to another. The WebSphere MQ network can be seen as a foreign bus by the WebSphere Application Server default messaging provider using a WebSphere MQ link.

JMS and the default messaging provider

Java Enterprise Edition (Java EE) applications (producers and consumers) access the SIBus and the bus members through the JMS API. JMS destinations are associated with SIBus destinations. A SIBus destination implements a JMS destination function. Session Enterprise JavaBeans (EJBs) use a JMS connection factory to connect to the JMS provider. Message Driven Beans (MDBs) use a JMS activation specification to connect to the JMS provider. See Figure 10-1 on page 343.

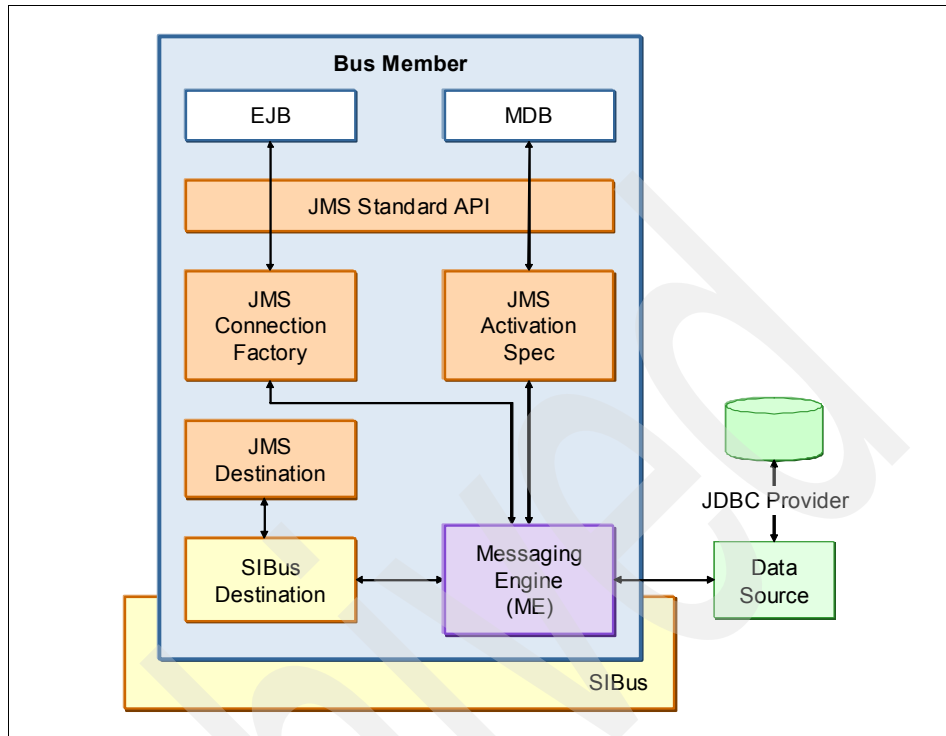


Figure 10-1 WebSphere default messaging provider and JMS

10.4.2 Choosing a messaging topology

The following topologies are some of the ones implemented by the WebSphere Application Server default messaging provider (in increasing complexity) using the previously defined concepts.

One bus, one bus member (single server)

This is the simplest and most common topology. It is used when applications deployed to the same application server need to communicate among themselves. Additional application servers that are not members of the bus and only need to use bus resources infrequently can connect remotely to the messaging engine. See Figure 10-2.

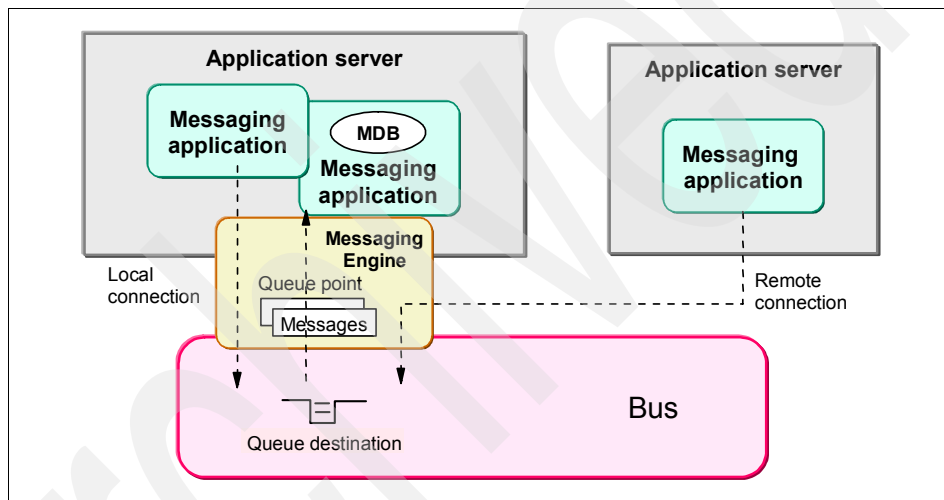


Figure 10-2 Single bus with an application server member

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 (a cluster)

With this variation, the bus member is a cluster. By default, only one application server in a cluster has an active messaging engine on a bus. If the server fails, the messaging engine on another server in the cluster is activated. 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 10-3.

Because everything is tunnelled through one messaging engine, performance might still be an issue.

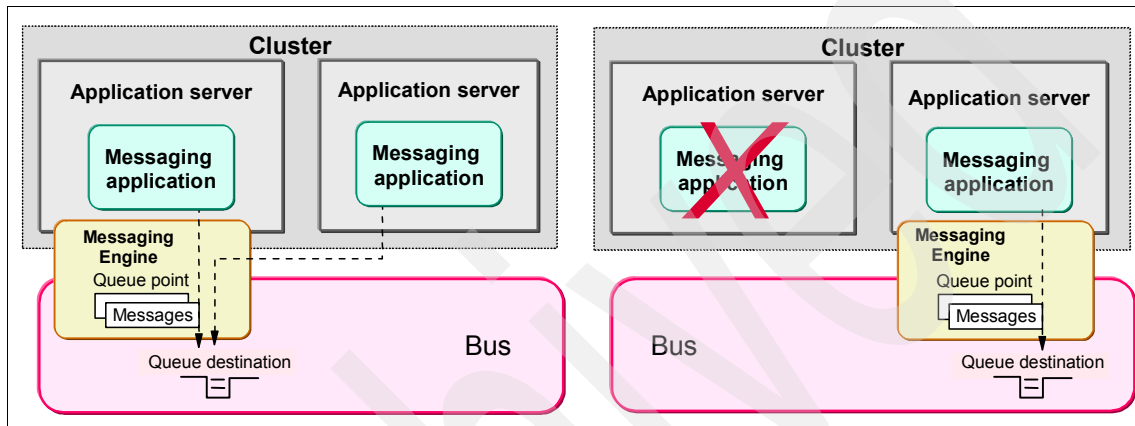


Figure 10-3 Single bus with a cluster member: High availability

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 10-4). Note that because messaging engines can run on any server, if one server goes down, both messaging engines will run on the remaining server.

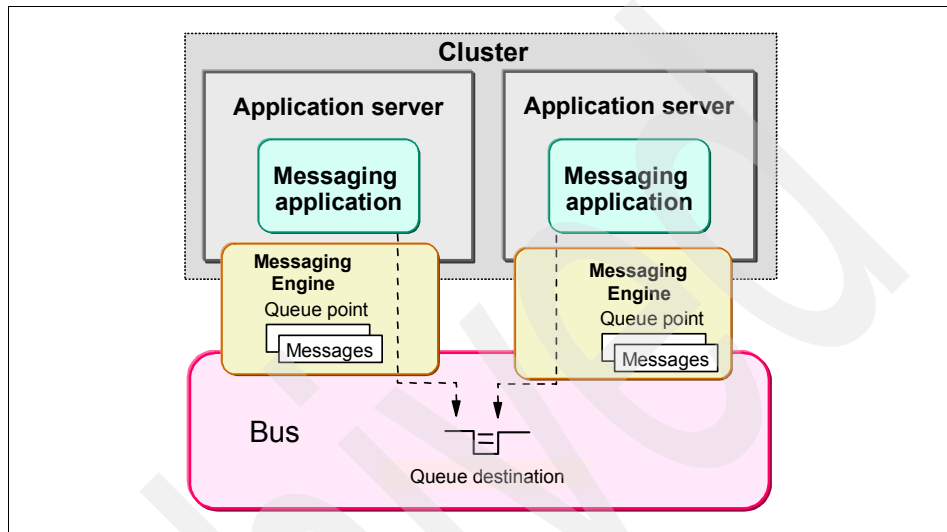


Figure 10-4 Single bus with a cluster member: Workload management

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.

One bus, multiple bus members

In this topology, there are multiple non-clustered application servers connected as members of the bus (Figure 10-5). 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.

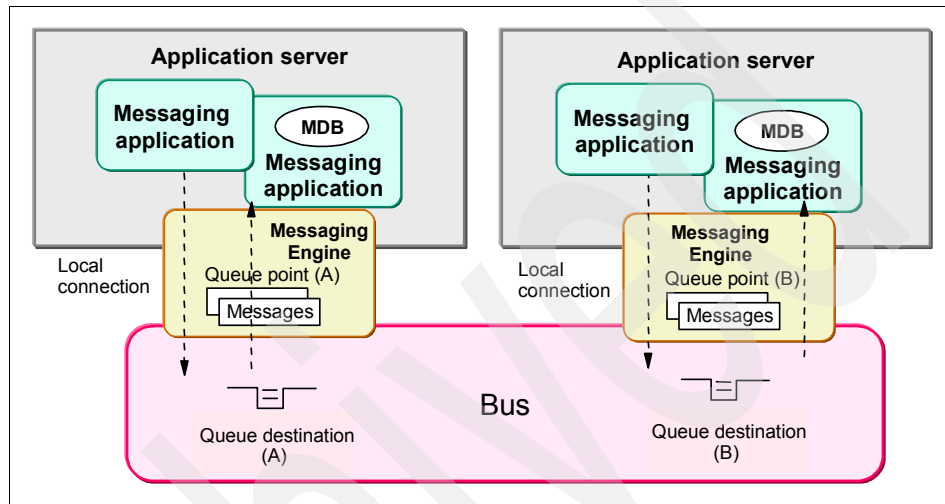


Figure 10-5 Single bus with multiple application server members

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 expand beyond the edge of a WebSphere Application Server cell. When you need to use messaging resources in multiple cells, you can connect the buses of each cell to each other. An enterprise might also deploy multiple interconnected service integration buses for organizational reasons. For example, an enterprise with several autonomous departments might want 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. This is achieved through the user of an MQ link.

Figure 10-6 illustrates how a service integration bus can be connected to another service integration bus and to a WebSphere MQ network.

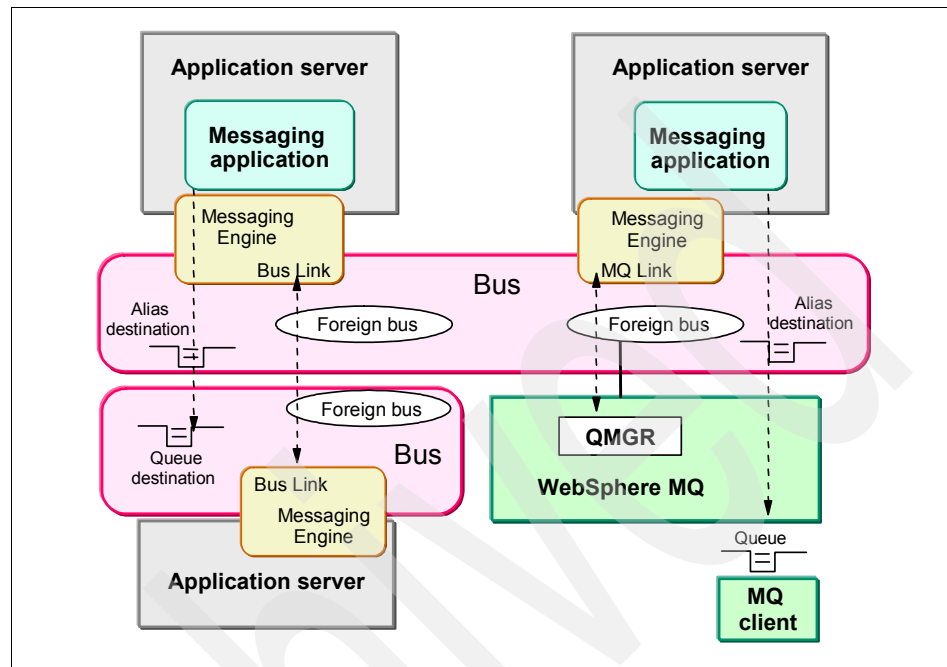


Figure 10-6 Multiple bus scenario

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 can send 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 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 10-7). The WebSphere MQ server defines properties for the connection to the queue manager or queue sharing group. With V7.0, this construct can also be applied to distributed (non z/OS platforms) queue managers as well.

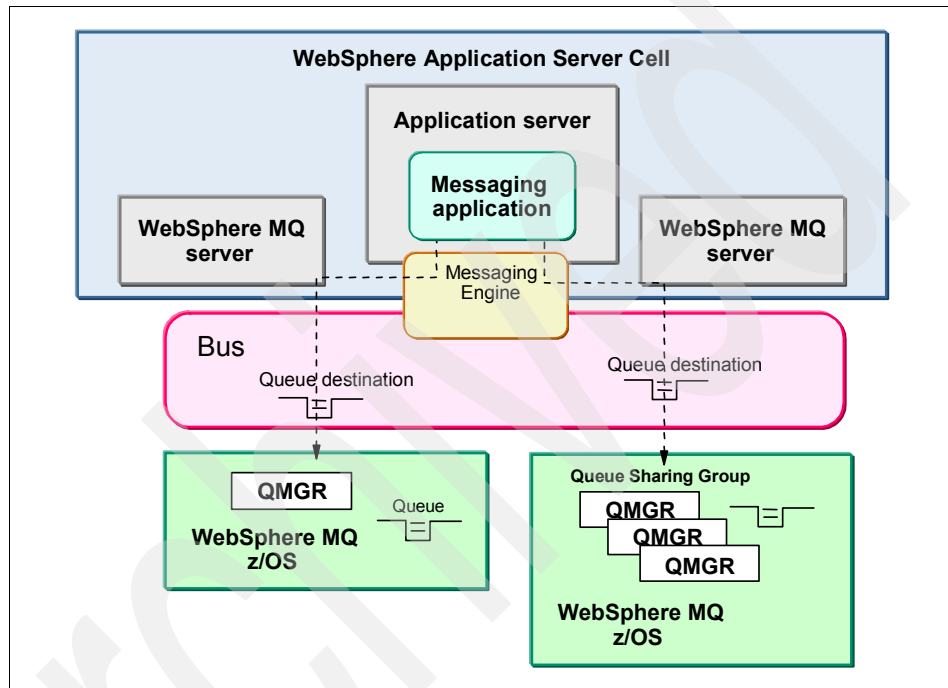


Figure 10-7 Multiple bus scenario

When you add a WebSphere MQ server as a member of the bus, the messaging engines establish connections to that WebSphere MQ server to access queues on WebSphere MQ.

To the WebSphere MQ server, the MQ queue manager or queue sharing group is regarded as a mechanism to queue messages for the bus. The WebSphere MQ server is regarded by the WebSphere MQ network as just another MQ client attaching to the queue manager or queue sharing 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.

10.5 Messaging features: How secure and reliable is it?

This section 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 into any planning. This section contains the following topics:

- ▶ “More messaging concepts” on page 351
- ▶ “Planning for security” on page 351
- ▶ “Planning for high availability” on page 352
- ▶ “Planning for reliability” on page 353

10.5.1 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 application 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.

10.5.2 Planning for security

There are two main areas in messaging security:

- ▶ Authorization and authentication of users and groups that want to connect to a bus.
- ▶ Securing the transportation of the message from source to destination.

Authentication and authorization

All access to a service integration bus must be both authorized and authenticated if bus security is turned on.

Authentication is done through an external access registry, such as 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 will be denied access even if the credentials are valid.

Other roles that affect permissions for users and groups are as follows:

- ▶ 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 messaging 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.

Relevant questions to ask when designing secure message transportation solutions are as follows:

- ▶ 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?

10.5.3 Planning for high availability

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 it 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 server if you have not already.

10.5.4 Planning for reliability

The JMS specification supports two modes of delivery for JMS messages:

- ▶ Persistent
- ▶ Non-persistent

The WebSphere administrator can select the mode of delivery on the JMS destination (queue/topic) configuration:

- ▶ Application (persistence is determined by the JMS client)
- ▶ Persistent
- ▶ Non-persistent

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 as follows:

- ▶ Best effort non-persistent
- ▶ Express non-persistent
- ▶ Reliable non-persistent
- ▶ 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 WebSphere Application Server Information Center article *Message reliability levels - JMS delivery mode and service integration quality of service* contains a table that outlines what happens to a message under various circumstances depending on delivery mode and reliability setting. See the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/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* article in the WebSphere Application Server Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/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 of the following types:

- ▶ *File stores* (or, flat files)

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 fewer administration costs, as well as no database administrator.

► *Data stores* (or, tables inside a database)

Data stores are the equivalent of file stores, but are implemented inside a relational database as a series of tables. They are administered by the facilities provided by the database. You can use any supported database product. Data stores might be preferable for larger organizations with an existing database infrastructure and skills.

Both types of message store can be subject to security, such as file system/database encryption and physical security access.

10.6 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 messaging

Planning item
Determine if and how messaging will be used.
Choose a JMS messaging provider (default messaging, WebSphere MQ, or generic).
Design a messaging topology. If using the default messaging provider, determine the number of buses to be used and if connections to other buses or WebSphere MQ are required.
Determine what destinations (queues, topics) are required initially, and the reliability levels for those destinations.
Determine the type of message data store to use.
Design a security strategy for messaging: <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, the WebSphere Application Server Information Center contains numerous messaging resources:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6tech_msg.html

Archived

Web services

This chapter describes Web services and the considerations administrators should make when planning for their usage on a WebSphere Application Server V7.0 architecture.

This chapter contains the following sections:

- ▶ “Introduction to Web services” on page 358
- ▶ “What is new in V7.0” on page 359
- ▶ “Important aspects in using Web services” on page 361
- ▶ “Web services architecture” on page 363
- ▶ “Support for Web services in WebSphere Application Server” on page 371
- ▶ “Planning checklist for Web services” on page 376

11.1 Introduction to Web services

Web services promote component reusability and a service-oriented approach to development. Because of that, they are commonly used as part of a service-oriented architecture (SOA). SOA is an approach to building enterprise applications that focuses on services (or loosely-coupled components) that can be composed dynamically. 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 WebSphere Application Server V7.0 provides. Any implementation of a SOA, including Web services, must have the following characteristics:

- ▶ Interoperability between platforms, systems, and programming languages
- ▶ Clear and unambiguous service interface description language
- ▶ Dynamic search and retrieval capabilities of a service at 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 use 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 are language independent.
The client and the server can be on different platforms, on different machines, or in different countries. There are no restrictions on the language used, if it supports XML and SOAP.
- ▶ Web services are based on mature and open standards.
The major underpinning technologies, such as XML and HTTP, were developed as open source standards themselves, with no proprietary technologies. As such, they 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 services. 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 V7.0

The main change in WebSphere Application Server V7.0 regarding Web services is the previously existing Feature Pack for Web Services for WebSphere Application Server V6.1, with added enhancements, has now been included in the base product.

WebSphere Application Server V7.0 offers some improvements to the Web services implementation from V6.1. Depending on whether or not you were using the Feature Pack, the new features can be considered separately.

11.2.1 What was in Feature Pack for V6.1

The Feature Pack for Web Services for WebSphere Application Server V6.1 extended the capabilities of V6.1 to enable Web services messages to be sent asynchronously, reliably, and securely, focusing on interoperability with other vendors. It also provided support for the Java API for XML Web Services (JAX-WS) 2.0 programming model.

The Feature Pack added support for the following features (also present in WebSphere Application Server V7.0):

- ▶ Web services standards
 - Web Services Reliable Messaging (WS-RM) 1.1
 - Web Services Addressing (WS-Addressing) 1.0
 - Web Services Secure Conversation (WS-SC) 1.0
 - SOAP 1.2
 - SOAP Message Transmission Optimization Mechanism (MTOM) 1.0
- ▶ Standards-based programming models
 - Java API for XML Web Services (JAX-WS) 2.0
 - Java Architecture for XML Binding (JAXB) 2.0
 - SOAP with Attachments API for Java (SAAJ) 1.3
 - Streaming API for XML (StAX) 1.0

11.2.2 Features added to WebSphere Application Server V7.0

In addition to the enhancements brought about by the Feature Pack for Web services, there are other major Web services enhancements in WebSphere Application Server V7.0:

- ▶ Web Services for Java Enterprise Edition (Java EE) 1.2 (JSR-109) specification is now supported.
- ▶ Support for JAX-WS 2.1, extending the functionality of JAX-WS 2.0 to provide support for the WS-Addressing in a standardized API.
- ▶ Added support for W3C (World Wide Web Consortium) SOAP over JMS (Java Message Service) 1.0 (submission draft) for JAX-WS.
- ▶ By default, HTTP chunking is used when sending MTOM (Message Transmission Optimization Mechanism) attachments, thus allowing for greater scalability and improved performance. This feature does not require the applications to be changed, but it can be disabled if desired.
- ▶ Added support for WS-Policy 1.5 and related specifications such as WS-SecurityPolicy 1.2.
- ▶ WS-Addressing is now based on JAX-WS.

- ▶ Security has been enhanced with the adoption of OASIS (Organization for the Advancement of Structured Information Standards) WS-SecureConversation 1.2.
- ▶ Enterprise Java Bean (EJB) 3.0 components may now be exposed as JAX-WS Web services.
- ▶ The Integrated Solutions Console has been improved with enhanced support to policy sets, application bindings, and endpoints management.
- ▶ Enhanced Configuration Archive support for Web services specific metadata (policy sets, policy types, and bindings) has been added.

11.3 Important aspects in using Web services

There are some business and technical aspects that need to be considered when deciding to use 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?
- ▶ Do you need a more consumable interface to existing exposed function?
Web services can be used as an easier way to expose application programming interfaces (APIs) to consumers. Wrapping existing APIs in Web services provides a more friendly interface to them.
- ▶ What business functionality do you want to expose to external parties?
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?

The technical issues that might affect your decision on the use of Web services includes:

- ▶ Does the business logic you wish to expose have state dependency?

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. Multiple request 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 of which to be wary. 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.

You can find information about the current status of the different available Web services standards on the following Web page:

<http://www.ibm.com/developerworks/webservices/standards/>

- ▶ 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 Web services architecture

The basic SOA consists of the following three primary components, as shown in Figure 11-1:

- ▶ **Service provider (or service producer)**
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.
- ▶ **Service requestor (or service consumer)**
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.
- ▶ **Service broker (or service registry)**
The service broker is responsible for making the service interface and implementation access information available to any potential service requestor. The service broker is not necessary to implement a service if the service requestor already knows about the service provider by other means.

Each component can act as one of the two other components. For example, if a service provider needs 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 11-1 shows the operations each SOA component can perform.

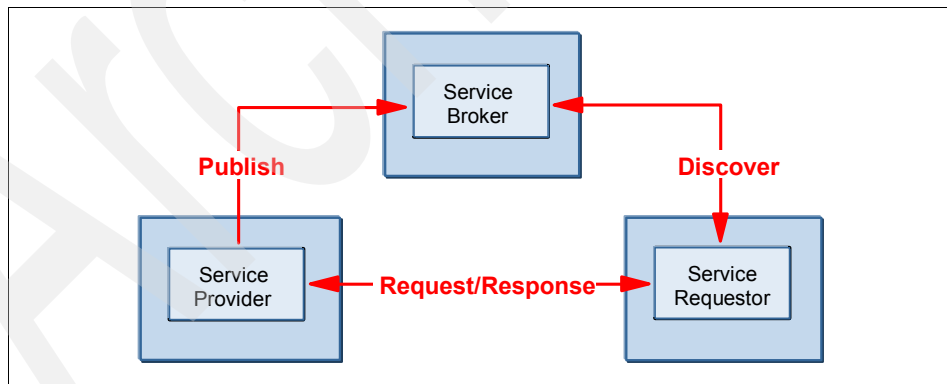


Figure 11-1 SOA components and operations

Before we look at the architecture from a Web services-specific view, 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.
- 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.
- 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.

Figure 11-2 on page 365 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.

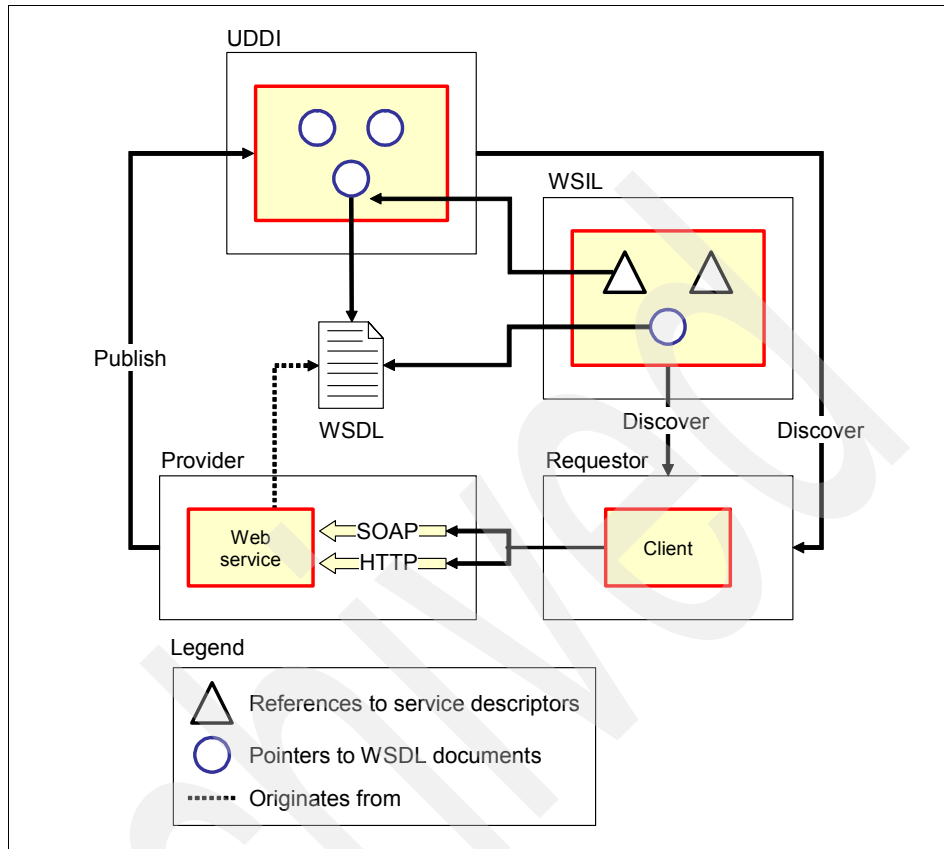


Figure 11-2 Main building blocks in an SOA approach based on Web services

11.4.1 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 patterns 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 as follows:

- ▶ One-way
- ▶ Asynchronous two-way
- ▶ Request-response
- ▶ Workflow-oriented
- ▶ Publish-subscribe
- ▶ Composite

One-way

In this 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 11-3). 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).

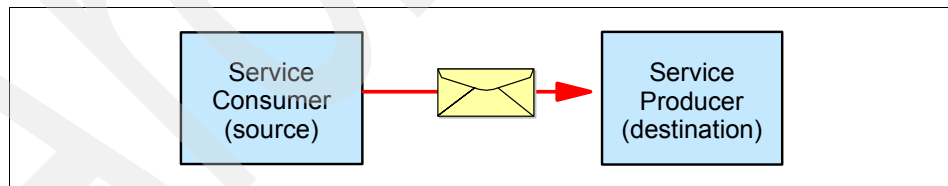


Figure 11-3 One-way message exchange pattern

Asynchronous two-way

In this message exchange pattern (Figure 11-4 on page 367), the service requestor expects a response, but the messages are asynchronous in nature (for example, where the response might not be available for many hours). Both sides must implement a Web service to receive messages. In general, the Web service

provided by the Service 2 Producer component has to relate a message it receives to the corresponding message that was sent by the Service 1 Consumer component.

Technically, this message exchange pattern is the same as the one-way pattern, with the additional requirement that there has to be a mechanism to associate response messages with their corresponding request message. This can be done at the application level or by using SOAP protocol.

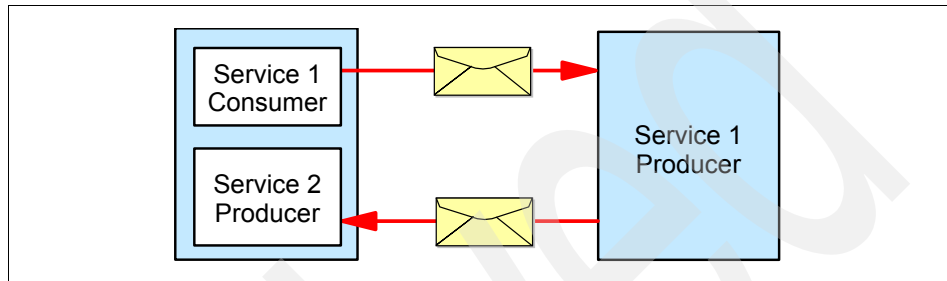


Figure 11-4 Asynchronous two-way message exchange pattern

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 11-5). In this message exchange pattern, the underlying transport protocol provides an implicit association between the request message and the response message.

In situations where the message exchange pattern is truly synchronous, such as when a user is waiting for a response, there is little point in decoupling the consumer and producer. In this situation, the use of SOAP/HTTP as a transport provides the highest level of interoperability. In cases where reliability or other quality of service requirements exist (such as prioritization of requests), alternative solutions might have to be sought.

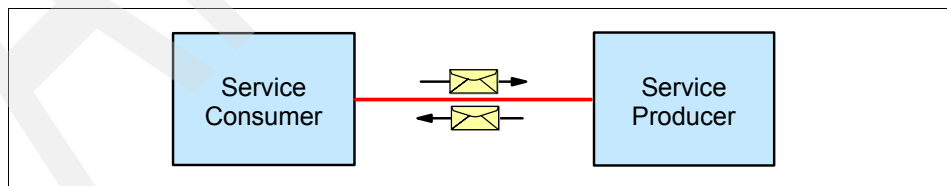


Figure 11-5 Request-response message exchange pattern

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 11-6).

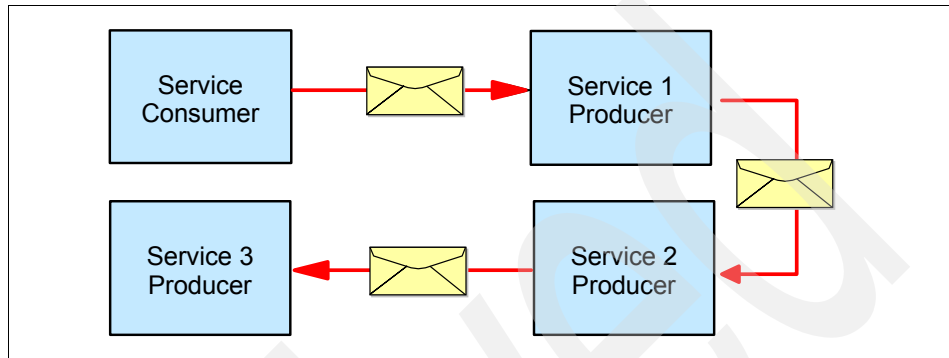


Figure 11-6 Workflow-oriented message exchange pattern

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 11-7 on page 369).

Implementation of this pattern at the application level is one possible architecture. Alternatively, the Service 1 Producer component can publish SOAP messages to a messaging infrastructure that supports the publish-subscribe paradigm.

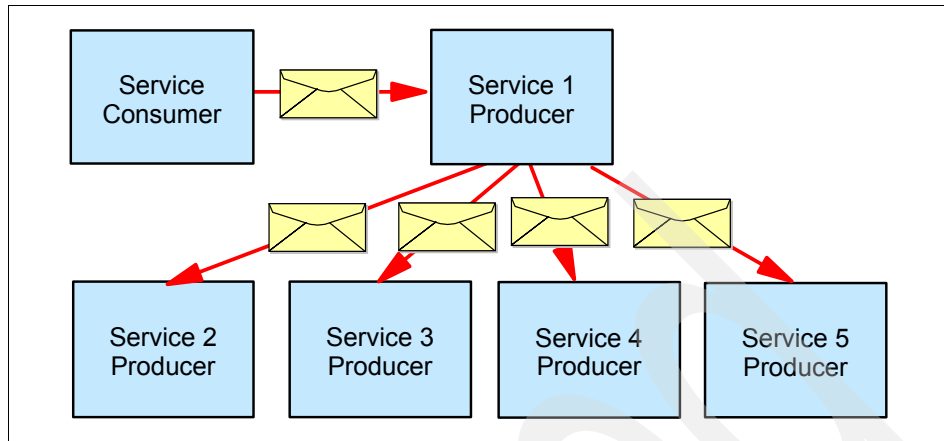


Figure 11-7 Publish-subscribe message exchange pattern

An example of a publish-subscribe message exchange pattern is a news syndication system. A news source publishes an article to the Service 1 Provider Web service. The Service 1 Provider Web service, in turn, sends the article to all interested parties.

Composite

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 11-8 on page 370).

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.

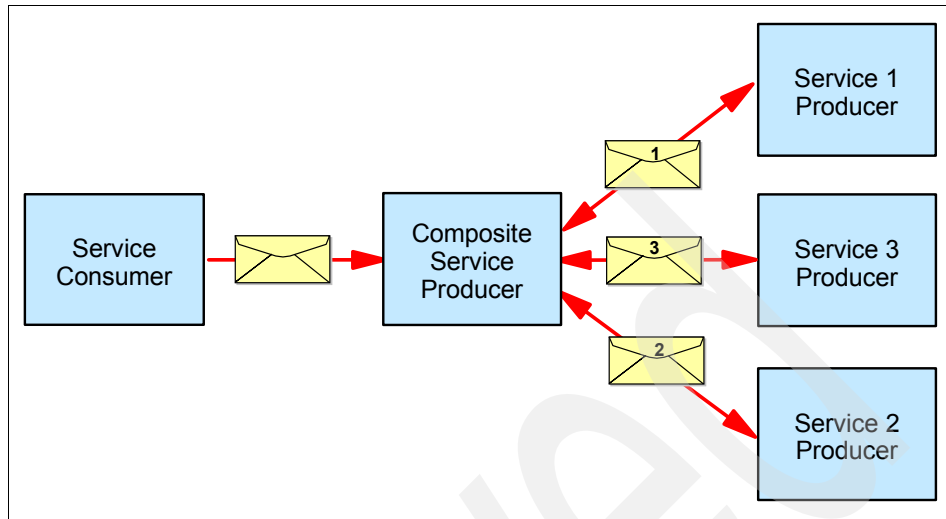


Figure 11-8 Composite message exchange pattern

SOAP processing model

At the application level, a typical Web service interaction occurs between a service consumer and a service provider, optionally with a lookup to a service registry. At the infrastructure level, additional intermediary SOAP nodes might be involved in the interaction (Figure 11-9).

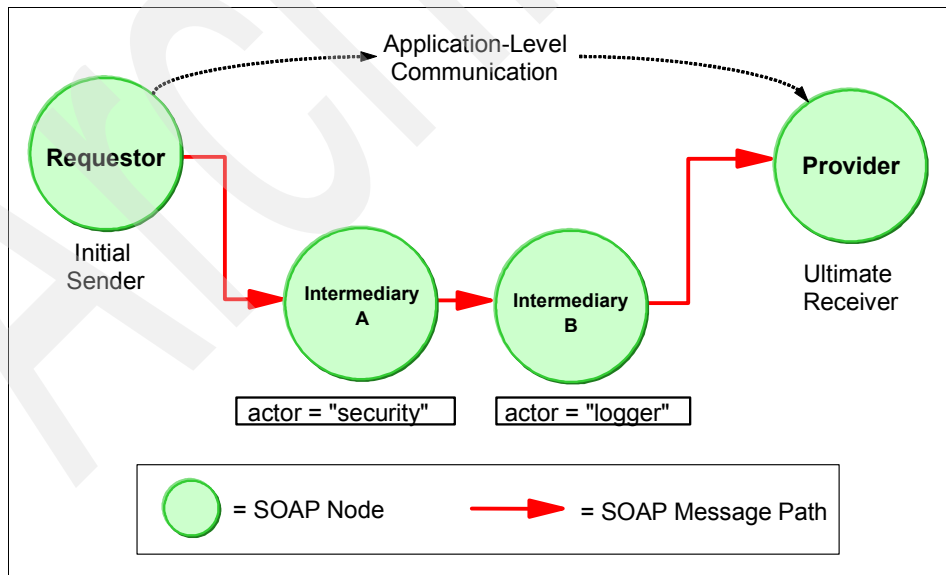


Figure 11-9 SOAP processing model

These intermediary nodes might handle quality of service and infrastructure functions that are non-application specific. Examples include message logging, routing, prioritization, and security. In general, intermediaries should not alter the meaning of the message body.

A typical situation where you need to use intermediary SOAP nodes is where you have an existing internal Web service implementation within your enterprise that you now want to expose externally. 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. Regardless of where it is placed, it can provide some or all of the following functions:

- ▶ Automatic publishing of WSDL files to an external UDDI or WSIL registry
- ▶ Automatic protocol/transport mappings
- ▶ Security functions
- ▶ Mediation of message structure
- ▶ Proxy server for Web service communications through a firewall
- ▶ Auditing of SOAP messages
- ▶ Operational management and reporting of published interfaces
- ▶ Web service threat detection and defense

11.5 Support for Web services in WebSphere Application Server

WebSphere Application Server V7.0 implements the Java EE 5 and Java SE 6 standards, and therefore it provides support for SOAP and for XML. The support for Web services in WebSphere Application Server can also be combined with the capabilities of an Enterprise Service Bus (ESB) product such as WebSphere Enterprise Service Bus or WebSphere Message Broker.

This section addresses how WebSphere Application Server V7.0 supports a Web services architecture.

11.5.1 Supported standards

WebSphere Application Server V7.0 supports a set of Web services standards that makes possible the creation and administration of interoperable, securable, transactionable, and reliable Web services applications. For a complete list of supported standards and specifications, see the Web services section in the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/rovr_specs.html

11.5.2 Service integration bus

There are many advantages to using a service integration bus that apply both to the application and to the enterprise at large. The advantages are as follows:

- ▶ 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

When the service integration bus is in place, any application that is Web service-enabled can reuse this infrastructure.

- ▶ Messaging support

The bus is built around support for JMS. 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.

- ▶ Support for standards

The bus is part of the Java EE 5 implementation and thus supports the major Web services standards that are also part of Java EE 5. These include the following standards:

- WS-I Basic Profile 1.1
- JAX-WS (JSR-224)
- JAX-RPC (JSR-101) 1.1
- UDDI V3
- WS-I Security
- 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.

11.5.3 Universal Description, Discovery, and Integration registries

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.

An 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 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.

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 world to communicate with their service with unknown and possibly malicious intent. Secondly, because the registry is accessible by anyone, it often possesses 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. 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.

11.5.4 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 as follows:

- ▶ 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.

11.5.5 Security

WebSphere Application Server V7.0 includes many security enhancements for Web services. There are a number of things that can be configured within 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 WS-Security specification focuses on the message authentication model and therefore it can be subject to several forms of attack.
- ▶ WS-SecureConversation provides session-based security, allowing secure conversations between applications using Web services.
- ▶ The endpoint for a Web service can be configured to be subject to authentication, security roles, and constraints.
- ▶ The underlying transport can be encrypted (for example, HTTPS).
- ▶ The bus can be configured to use authenticating proxy servers. Many organizations use these proxy servers to protect data and services.
- ▶ A JAX-WS client application can be also secured using the Web Services Security API

Note that, as always, the more security you have, the more performance is likely to suffer.

11.5.6 Performance

Unfortunately, performance of Web services is still poor 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.

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 cacheable. Not all of them should be (for example, dynamic or sensitive data). This can be a 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.

Note: Dynamic Caching is only included in the Network Deployment edition of WebSphere Application Server V7.0.

For SOAP, some performance improvements can be achieved with the MTOM standard through the optimization of the messages it provides. Avoiding the use of large messages can also help.

Finally, the StAX standard constitutes an alternative and more efficient method to change and traverse XML data and should be considered.

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 and how Web services will be used.
Determine how Web service clients will call providers (directly, through the service integration bus, or through an ESB).
Determine 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.
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.

Note: JAX-RPC is included in the Java EE 5 specification. If you are new to Web services, JAX-WS should be the preferred programming model of choice.

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 having a copy of this book available as you plan your Web services environment. This book is based on V6.1 with the Feature Pack and does not cover the additions and changes made to WebSphere Application Server V7.0.

The WebSphere Application Server Information Center also contains a lot of useful and up-to-date information. For a good entry point to Web services topics, see the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/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.

Archived

Security

WebSphere Application Server provides security infrastructure and mechanisms to protect sensitive resources and to address enterprise end-to-end security requirements.

This chapter covers the most important aspects that are inherent in planning security for a WebSphere Application Server installation. 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 V7.0” on page 380
- ▶ “Security in WebSphere Application Server” on page 381
- ▶ “Security configuration considerations” on page 402
- ▶ “Planning checklist for security” on page 405

12.1 What is new in V7.0

This section describes the major new features added to WebSphere Application Server V7.0.

- ▶ Multiple security domains

This feature provides the possibility of having different security settings in the same cell, therefore allowing separate security environments for administrative applications and user applications. A security domain can be enabled at cell, node, or application server scope.

- ▶ Security auditing

As a new subsystem of the WebSphere Application Server security infrastructure, security auditing achieves two primary objectives:

- Confirming the effectiveness and integrity of the existing security configuration.
- Identifying areas where improvement to the security configuration might be needed.

- ▶ Certificate management enhancements

New certificate management functions have been provided to improve the security of communications between a server and a client:

- Creating and using a certificate authority (CA) client to enable users to connect to a CA server to request, query, and revoke certificates
- Creating and using chained personal certificates to allow a certificate to be signed with a longer life span
- Creating and revoking CA certificates to ensure secure communication between the CA client and the CA server
- For WebSphere Application Server for z/OS, performing certificate management on System Authorization Facility (SAF) keyrings

- ▶ Security annotations

Security annotations, which are an alternative means of defining security roles and policies, can be used instead of, or in addition to, defining roles and policies in the deployment descriptor.

- ▶ Fine-grained administrative security in the Integrated Solutions Console

In addition to the existing support in the `wsadmin` command tool, fine-grained security can now be configured in the Integrated Solutions Console as well.

- ▶ Kerberos (KRB5) authentication mechanism
Although not available in the base release of V7.0, support for Kerberos as an authentication mechanism will be included in a future release.
- ▶ Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Web authentication
SPNEGO Web authentication has been introduced as a substitute for SPNEGO TAI. It can be configured by using the Integrated Solutions Console. It allows the dynamic reload of SPNEGO without the need of a server restart.

12.2 Security in WebSphere Application Server

The fundamental reason for having security in your systems is to protect them from intruders (external or internal to your organization) and to ensure that there is no accidental or intentional misuse of your systems or the data flowing through your systems.

When planning security for WebSphere Application Server, it is important to have a comprehensive security policy in place that coordinates neatly with the overall environment security. WebSphere Application Server adheres to standard Java Enterprise Edition (Java EE) specifications and integrates 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. The major concerns are access to the site and protection against environmental conditions. Commonly, such areas are physically secured and access is limited to a small number of individuals.
- ▶ Logical security
Logical security includes the mechanisms provided to protect systems and applications from having unauthorized accesses through the system console or through the network. User authentication is the most common logical security mechanism but there are others like encryption, certificates, or firewalls.

Security policy

A security policy is a guideline for an organization that describes the processes needed to implement a robust security environment.

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 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 important to be thorough. Remember to examine the two types of security: physical and logical. This can be a 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 many 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 technology available is necessary. However, on a test system with no external access, the appropriate security level can be met with simpler elements.

- ▶ 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 risk 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 and logical security as recommended in the security policy.

Security reviews

A timely and regular review of the security policy and its implementation 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

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 components that make up the operating environment for security in WebSphere Application Server.

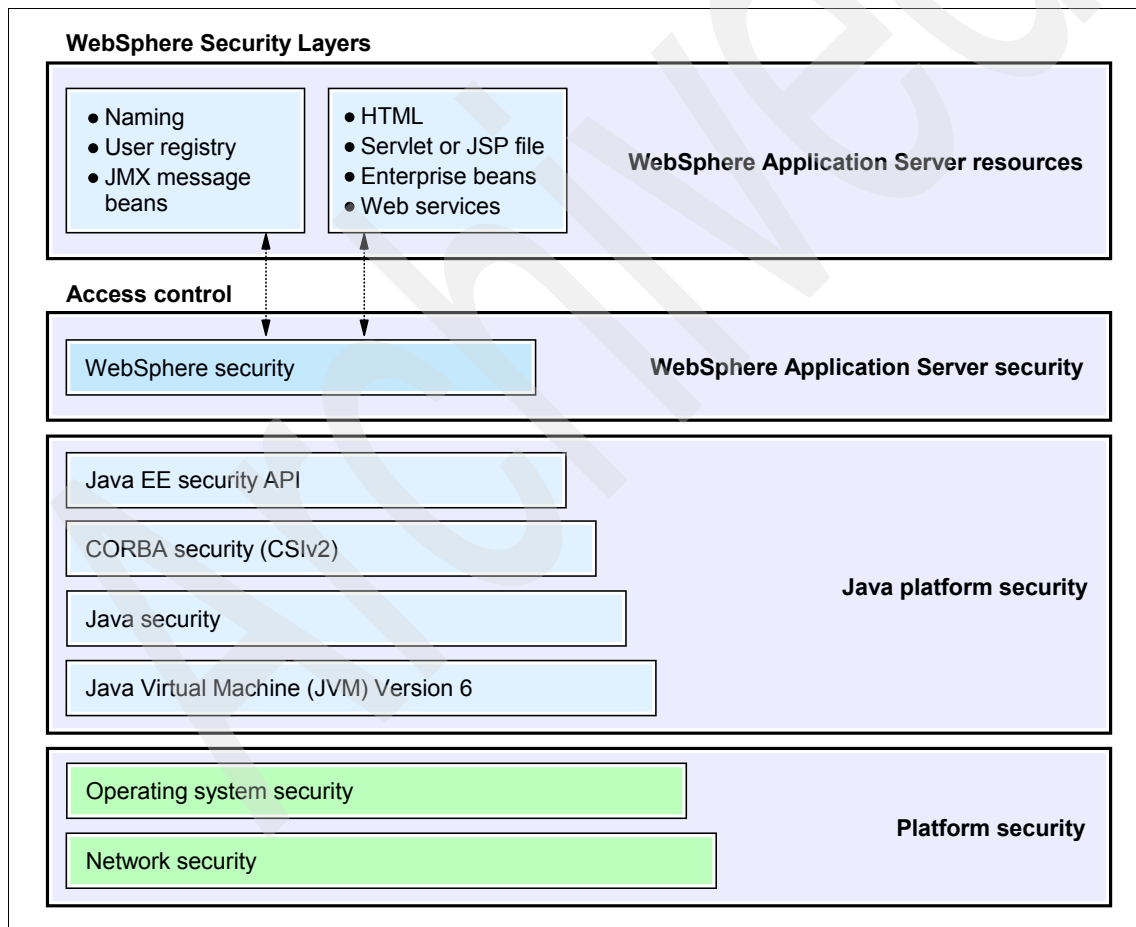


Figure 12-1 WebSphere Application Server security layers

These components consist of the following technologies:

▶ WebSphere Application Server security

WebSphere Application Server security enforces security policies and services in a unified manner on access to Web resources, enterprise beans, Web services, and JMX administrative resources. It consists of WebSphere Application Server security technologies and features to support the needs of a secure enterprise environment.

▶ Java platform security

– Java Platform, Enterprise Edition (Java EE) security API

The security collaborator enforces Java EE-based security policies and supports Java EE 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: Secure Authentication Service (SAS) security protocol is only supported between WebSphere Application Server V6.0 and previous version servers that have been federated in a Version 7.0 cell. In future releases, IBM will no longer ship or support the SAS IOP security protocol.

– Java security

The Java security model offers access control to system resources including file system, system property, socket connection, threading, class loading, and so on. Application code must explicitly grant the required permission to access a protected resource.

– Java virtual machine (JVM) 6.0

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. This option is only recommended for z/OS systems. 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.

Note: If you are interested in protecting your system from applications, WebSphere Application Server should run as a non-root user in distributed platforms so that access to root files and resources is not allowed. Be aware that, in this case, the operating system registry cannot be used.

– 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.

12.2.1 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.

Depending on the type of client, the authentication information is sent by using different protocols:

- ▶ Enterprise Beans clients use CSiv2.
- ▶ Web clients use HTTP or HTTPS.

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

- ▶ Lightweight Third Party Authentication (LTPA)
- ▶ Kerberos
- ▶ Rivest Shamir Adleman (RSA) token authentication

Note: Simple WebSphere Authentication Mechanism (SWAM) is deprecated in WebSphere Application Server V7.0 and will be removed in a future release.

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.

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 is then 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.

LTPA keys are generated automatically during the first server startup and regenerated before they expire. You can disable automatic regeneration by WebSphere Application Server so you can generate them on a schedule.

Kerberos

Warning: At the time of writing, the Kerberos and LTPA option is not available. It has been included in this book because it will be supported in a future update.

Although being new to WebSphere Application Server V7.0, Kerberos is a mature, standard authentication mechanism that enables interoperability with other applications that support Kerberos authentication. It provides single sign on (SSO) end-to-end interoperable solutions and preserves the original requester identity. Kerberos is composed of three parts: a client, a server, and a trusted third party known as the Kerberos Key Distribution Center (KDC). The KDC provides authentication and ticket granting services.

The KDC maintains a database or repository of user accounts for all of the security principals in its realm. Many Kerberos distributions use file-based repositories for the Kerberos principal and policy database and others use Lightweight Directory Access Protocol (LDAP) as the repository.

A long-term key for each principal¹ is maintained by the KDC in its accounts database. This long-term key is derived from the password of the principal. Only the KDC and the user that the principal represents should know the long-term key or password.

¹ A principal is a unique identity which represents a user.

There are some benefits in using Kerberos:

- ▶ When using Kerberos authentication, the user clear text password never leaves the user machine. The user authenticates and obtains a Kerberos ticket granting ticket (TGT) from a KDC by using a one-way hash value of the user password. The user also obtains a Kerberos service ticket from the KDC by using the TGT. The Kerberos service ticket that represents the client identity is sent to WebSphere Application Server for authentication.
- ▶ A Java client can participate in Kerberos SSO using the Kerberos credential cache to authenticate to WebSphere Application Server.
- ▶ J2EE, Web services, .NET, and Web browser clients that use the HTTP protocol can use the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) token to authenticate to WebSphere Application Server and participate in SSO by using SPNEGO Web authentication. Support for SPNEGO as the Web authentication service is new to WebSphere Application Server V7.0.
- ▶ Server-to-server communication using Kerberos authentication is provided.

Note: WebSphere Application Server can support both Kerberos and Lightweight Third-Party Authentication (LTPA) authentication mechanisms at the same time.

Rivest Shadler Adleman token authentication

The Rivest Shadler Adleman (RSA) token authentication mechanism, new to WebSphere Application Server V7.0, aids the flexible management objective to preserve the base profiles configurations and isolate them from a security perspective. This mechanism permits the base profiles managed by an administrative agent to have different Lightweight Third-Party Authentication (LTPA) keys, different user registries, and different administrative users.

Note: The RSA token authentication mechanism can only be used for administrative requests. As such, the authentication mechanism choices for administrative authentication are part of the Global Security panel of the Integrated Solutions Console.

This authentication mechanism ensures that after the RSA root signer certificate (which has a 20-year lifetime) is exchanged between two administrative processes, there is no need to synchronize security information among disparate profiles for administrative requests. The RSA personal certificate (1-year lifetime) is used to perform the cryptographic operations on the RSA tokens and can be verified by the long-lived RSA root. RSA token authentication is different from LTPA where keys are shared and if one side changes, all sides need to change.

Because RSA token authentication is based on a private key infrastructure (PKI), it benefits from the scalability and manageability of this technology in a large topology.

An RSA token has more advanced security features than LTPA; this includes a nonce value that makes it a one-time use token, a short expiration period (because it is a one-time use token), and trust, which is established based on certificates in the target RSA trust store.

RSA token authentication does not use the same certificates as used by Secure Sockets Layer (SSL). This is the reason RSA has its own keystores. In order to isolate the trust established for RSA, the trust store, keystore, and root keystore, need to be different from the SSL configuration.

User registries

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 in a certain scope. WebSphere Application Server supports the following types of user registries:

- ▶ Local operating system
- ▶ Standalone Lightweight Directory Access Protocol (LDAP)
- ▶ Federated repository (a combination of a file-based registry and one or more LDAP servers in a single realm).
- ▶ Custom registry

In the event that none of the first three options are feasible, 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.

Lightweight Directory Access Protocol

LDAP is a directory service. The information it contains 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.

WebSphere Application Server supports various LDAP servers.

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. WebSphere Application Server provides delegation through the use of RunAs roles and mappings.

Single Sign On

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. Other authentication mechanisms may have different requirements.

- ▶ All SSO participating servers must use the same user registry (for example, the LDAP server).
- ▶ All SSO participating servers must be in the same Domain Name System. (cookies are issued with a domain name and 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 Web browser must be configured to accept cookies.
- ▶ Server time and time zone must be correct. The SSO token expiration time is absolute.
- ▶ All servers participating in the SSO scenario must be configured to share LTPA keys.

Single Sign On for HTTP requests is also possible with SPNEGO Web authentication. Microsoft Windows users can access WebSphere Application Server resources without requiring an additional authentication process after being authenticated by a Domain Controller. Detailed information about SPNEGO Web authentication can be found in the WebSphere Application Server V7.0 Information Center at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/csec_SPNEGO_explain.html

Java Authentication and Authorization Service

The Java Authentication and Authorization Service (JAAS) extends the Java security architecture with additional support to authenticate and enforce access control with principals and users. It implements a Java version of the standard Pluggable Authentication Module (PAM) framework and extends the access control architecture of the Java platform in a compatible fashion to support user-based authorization or principal-based authorization. WebSphere Application Server fully supports the JAAS architecture and extends the access control architecture to support role-based authorization for Java EE resources including servlets, JSP files, and EJB components.

Although the applications remain unaware of the underlying authentication technologies, they need to contain specific code to take advantage of JAAS. If a new JAAS module is plugged-in, the application will work without a single modification of its code.

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 associations

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.

Note: SPNEGO TAI has been deprecated in WebSphere Application Server V7.0.

12.2.2 Authorization

Authorization is the process of checking whether a given user has the privileges necessary to get access to a requested resource. WebSphere Application Server supports many authorization technologies:

- ▶ Authorization involving the Web container and Java EE technology
- ▶ Authorization involving an enterprise bean application and Java EE technology
- ▶ Authorization involving Web services and Java EE technology
- ▶ Java Message Service (JMS)
- ▶ Java Authorization Contract for Containers (JACC)

Java Authorization Contract for Containers

WebSphere Application Server V7.0 supports both a default authorization provider, and, alternatively, an authorization provider that is based on the Java Authorization Contract for Containers (JACC) specification. The JACC-based authorization provider enables third-party security providers to handle the Java EE authorization.

Note: We recommend using IBM Tivoli Access Manager when an external JACC provider is needed. WebSphere Application Server V7.0 includes a Tivoli Access Manager client.

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 Java EE application-based authorization decisions are delegated to the provider per the JACC specification. Figure 12-2 shows the communications flow.

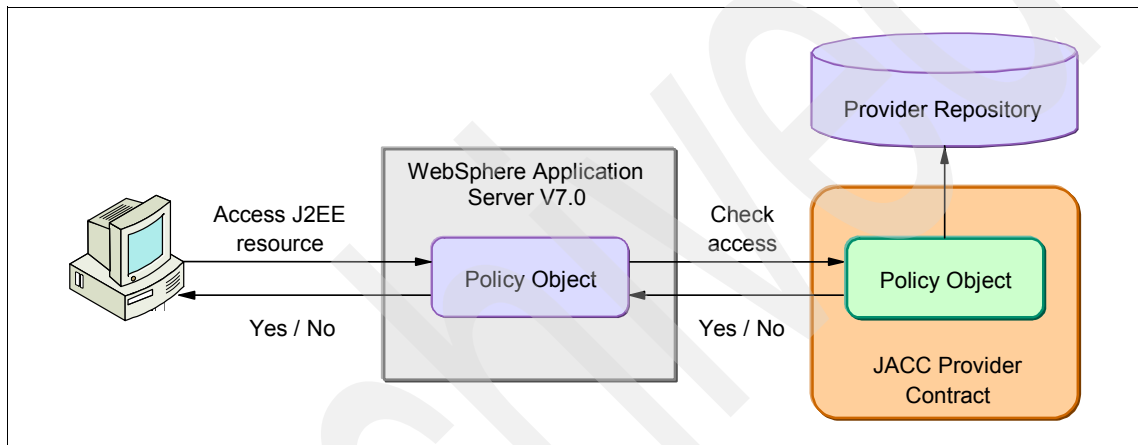


Figure 12-2 JACC provider architecture

Note: 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.

Dynamic module updates in JACC

WebSphere Application Server handles the dynamic module update with respect to JACC for Web modules. When the Web module is updated, only that particular application has to be restarted in native authorization mode. In the case of JACC being enabled, it depends on the provider support to handle the dynamic module updates specific to the security modules.

Fine-grained administrative security

Fine-grained administrative security was introduced in WebSphere Application Server V6.1, although it was only configurable with the `wsadmin` command tool. Fine-grained administrative security can grant access to each user role per resource instance instead of granting access to all of the resources in the cell, which allows a better separation of administrative duties.

WebSphere Application Server V7.0 includes new panels in the Integrated Solutions Console that simplify the fine-grained administrative security configuration.

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.
- ▶ **ISC Admins**
An individual or group that uses the ISC Admins role has Administrator privileges for managing users and groups in the federated repositories from within the Integrated Solutions Console only.

Note: The ISC Admins role is only available for Integrated Solutions Console users. It is not available for `wsadmin` users.

- ▶ **Deployer**
Users granted this role can perform both configuration actions and runtime operations on applications.

- ▶ **Admin Security Manager**

Only users who are assigned to this role can assign users to Administrative roles. When fine-grained administrative security is used, only users who are assigned to this role at cell level can manage authorization groups.

- ▶ **Auditor**

Users granted this role can view and modify the configuration settings for the security auditing subsystem. The Auditor role includes the Monitor role. This allows the auditor to view but not change the rest of the security configuration.

Security annotations

Java annotations, which are powerful programming tools resulting from the JSR-175 recommendation, are a standard way to include supported security behaviors while allowing the source code and configuration files to be generated automatically.

In Java EE 5, the security roles and policies can be defined using annotations as well as within the deployment descriptor. During the installation of the application, the security policies and roles defined using annotations are merged with the security policies and roles defined within the deployment descriptor. This merge is performed by the Annotations Metadata Manager (AMM) facility. Data defined in the deployment descriptor takes precedence over data defined in annotations.

Java annotations can be used in Enterprise Java Beans (EJB) 3.0 or Servlet 2.5 components. However, some security annotations are only available with EJB 3.0 components.

12.2.3 Secure communications

In order to prevent communications against eavesdropping, some sort of security has to be added.

WebSphere Application Server uses Java Secure Sockets Extension (JSSE) as the SSL implementation for secure connections. JSSE is part of the Java Standard Edition (Java SE) specification and is included in the IBM implementation of the Java Runtime Extension (JRE). JSSE handles the handshake negotiation and protection capabilities that are provided by SSL to ensure secure connectivity exists across most protocols. JSSE relies on a X.509 standard public key infrastructure (PKI).

A PKI represents a system of digital certificates, certificate authorities, registration authorities, a certificate management service, and a certification path validation algorithm. A PKI verifies the identity and the authority of each party

that is involved in an Internet transaction, either financial or operational, with requirements for identity verification. It also supports the use of certificate revocation lists (CRLs), which are lists of revoked certificates.

Secure Sockets Layer

Secure Sockets Layer (SSL) is the industry standard for data interchange encryption between clients and servers. SSL provides secure connections through the following technologies:

- ▶ Communication privacy
The data that passes through the connection is encrypted.
- ▶ Communication integrity
The protocol includes a built-in integrity check.
- ▶ Authentication
The server authenticates the client, interchanging digital certificates.

A certificate is an encrypted, password-protected file that includes the following information:

- ▶ Name of the certificate holder
- ▶ Private key for encryption/decryption
- ▶ Verification of sender's public key
- ▶ 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.

Certificate management

Certificates can be created and managed through the Integrated Solutions Console.

WebSphere Application Server provides mechanisms for creating and managing client CA clients and keystores, and for creating self-signed certificates and certificate authority requests.

12.2.4 Application security

The Java EE specification defines the building blocks and elements of a Java EE application that build an enterprise application. The specification provides details about security related to different elements. A typical Java EE application consists of an application client tier, a Web tier, a EJB tier, and a Web services

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 Java EE application.

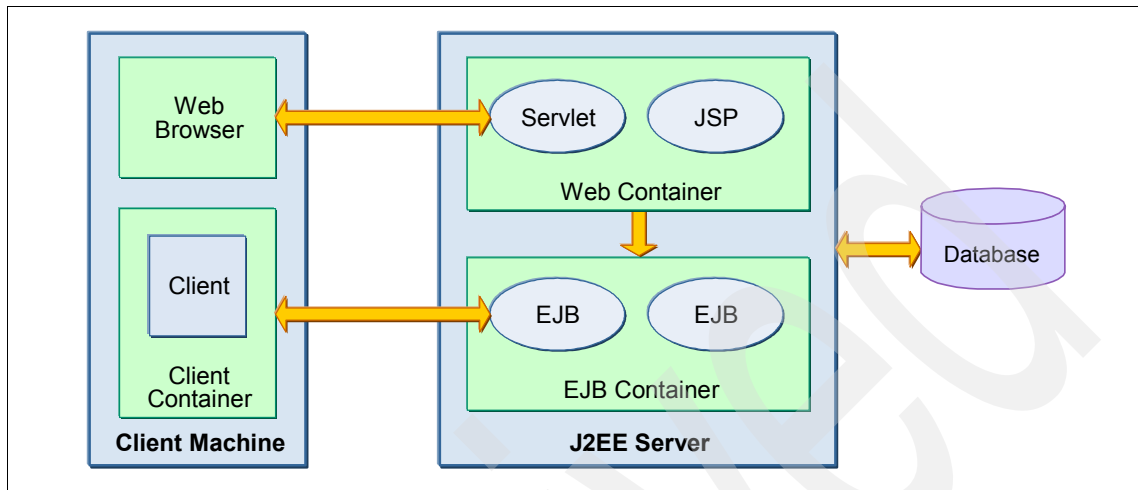


Figure 12-3 Java EE application components

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.

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 on page 398 shows an example of how roles can be mapped to users.

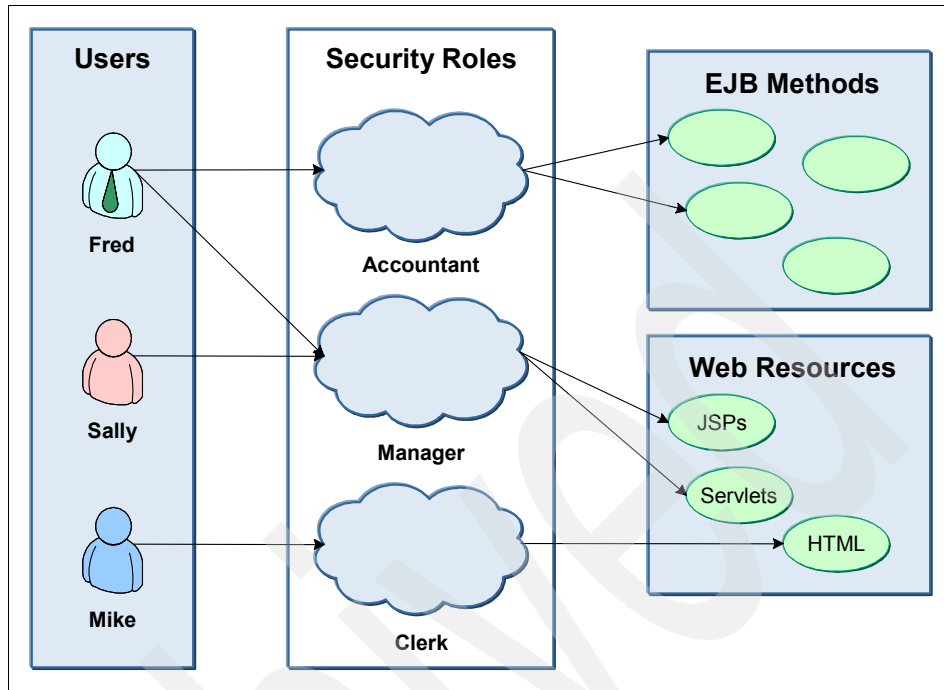


Figure 12-4 User role mapping

This two-phase approach to security gives a great deal of flexibility because deployers and administrators have control over how their users are mapped to the various security roles.

Security for Java EE resources

Java EE 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 *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. The deployment descriptor can be created and modified using Rational Application Developer for WebSphere Software V7.5.

Security policies can be defined using security annotations as well. Security annotations are included in Java code in a declarative manner.

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 robust security infrastructure that is transparent to the developer. However, there are cases where the security model is not sufficient and the developer wants greater control over to what the user has access. For such cases, there are a few security APIs that the developers can implement.

Java security

While Java EE security guards access to Web resources such as servlets, JSPs, and EJBs, Java security guards access to system resources such as file I/O, sockets, and properties.

Note: You need to be careful before enabling Java security. Java security places new requirements on application developers and administrators. Your applications might not be prepared for the fine-grain access control programming model that Java security is capable of enforcing.

For more information about Java security, refer to the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/csec_deleg.html

12.2.5 Security domains

The WebSphere security domains (WSD) provide the flexibility to use different security configurations in a WebSphere Application Server cell. WSD is also referred to as multiple security domains, or simply, security domains. With security domains you can configure different security attributes, such as the user registry, for different applications in the same cell.

Note: Be aware that some configuration options are only available as global security settings and cannot be configured in a domain, although they can be used by multiple domains:

- ▶ Federated repositories
- ▶ JACC

The global security configuration applies to all administrative functions, naming resources, and Mbeans, and is the default security configuration for user applications. Each security domain has to be associated with a scope (cell, or specific clusters, servers and service integration buses) where it will be applied.

The attributes that can be configured at the domain level are as follows:

- ▶ Application security
- ▶ Java security
- ▶ User realm (registry)
- ▶ Trust association
- ▶ SPNEGO Web authentication
- ▶ RMI/IIOP security (CSIv2)
- ▶ JAAS logins (application, system and J2C authentication data)
- ▶ Authentication mechanism attributes
- ▶ Authorization provider
- ▶ Custom properties

You do not need to configure all the attributes. Those not defined in the domain are obtained from the global configuration. When planning for security, you have to determine whether you need different security attributes for your servers or if they can use the global configuration settings. For example, you may want to use various user registries if you have different sets of users that can not be mixed (for instance, when the responsibility for user administration of each registry falls on different teams).

Note: We suggest using at least one security domain, at cell scope, in order to separate administrative users from application users.

If you plan to use a security domain for a special application with stringent security requirements, this application should be deployed in a dedicated server or cluster. The scope of the domain should include only the server(s) or cluster(s) where the application is deployed.

Note: Deploying an application in more than one server with different security domains may lead to inconsistent behavior.

12.2.6 Auditing

WebSphere Application Server V7.0 introduces a new feature as part of its security infrastructure: the security auditing subsystem.

Security auditing has two primary goals:

- ▶ Confirming the effectiveness and integrity of the existing security configuration (accountability and compliance with policies and laws)
- ▶ Identifying areas where improvement to the security configuration might be needed (vulnerability analysis)

Security auditing achieves these goals by providing the infrastructure that allows you to implement your code to capture and store supported auditable security events. During run time, all code (except the Java EE 5 application code) is considered to be trusted. Each time a Java EE 5 application accesses a secured resource, any internal application server process with an audit point included can be recorded as an auditable event.

If compliance with regulatory laws or organizational policies have to be proved, you can enable auditing and configure filters to log the events you are interested in according to your needs.

The security auditing subsystem has the ability to capture the following types of auditable events:

- ▶ Authentication
- ▶ Authorization
- ▶ Principal/credential mapping
- ▶ Audit policy management
- ▶ Administrative configuration management
- ▶ User registry and identity management
- ▶ Delegation

These events are recorded in signed and encrypted audit log files in order to ensure its integrity. Encryption and signing of audit logs are not set by default, though we suggest its use to protect those records from being altered. You will have to add keystores and certificates for encryption and signing.

Log files can be read with the audit reader, a tool that is included in WebSphere Application Server V7.0 in the form of a `wsadmin` command. For example, the following `wsadmin` command line returns a basic audit report:

```
AdminTask.binaryAuditLogReader(['-fileName myFileName -reportMode basic  
-keyStorePassword password123 -outputLocation /binaryLogs'])
```

WebSphere Application Server provides a default audit service provider and event factory², but you can change them if you have special needs. For instance, you could configure a third-party audit service provider to record the generated events to a different repository.

12.3 Security configuration considerations

When planning for security, it is important to keep in mind the difference between administrative security and application security from the WebSphere perspective:

- ▶ Administrative security protects the cell from unauthorized modification.
- ▶ 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. Since WebSphere Application Server V6.1, these security functions can be enabled separately. Administrative security can be enabled during profile creation. The default is for administrative security 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).

The default authentication mechanism is LTPA, but when Kerberos is included in the product, you can select Kerberos and LTPA later.

² The audit service provider formats and records audit events. The event factory collects the data associated to the auditable security events and sends it to the audit service provider.

After the profile has been created, you also have different options for the user account repository (or user registry):

- ▶ Federated repository (including the file-based registry created for administrative security)
- ▶ Local operating system
- ▶ Stand-alone LDAP registry
- ▶ Stand-alone custom registry

Scenarios

In order to give a better explanation of the implications if you select one of the previous options, we describe three 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. The profile creation tools create a file-based registry in the configuration file system (`profile_root/config/cells/cellname/fileRegistry.xml`), 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 Integrated Solutions Console by navigating to **Users and groups** → **Manage users**).

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 or **wsadmin**.

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 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 12-1 summarizes these options.

Table 12-1 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.	No administrative security is configured. Anyone with network 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 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.

12.4 Planning checklist for security

Table 12-2 provides a summary of items to consider as you plan, and additional resources that can help you.

Table 12-2 *Planning checklist for Web services*

Planning item
Determine when and how you will enable WebSphere Application Security.
Create a strategy for administrative security.
Plan for auditing.
Determine if multiple security domains will be used.
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 administrative security until after installation. Populate the user registry with the appropriate user IDs and groups for initial security.
Determine the authentication mechanism (LTPA is strongly suggested).
Determine the authorization method (default or JACC). If using 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.
Create a strategy for securing applications using Java EE security. Choose either declarative or programmatic. If selecting declarative, then should annotations be used or not? 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.
Review and incorporate security strategies for the service integration bus.

Resources

For a good overall reference for WebSphere Application Server security, refer to *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316.

We suggest that you have a copy of this book available as you plan to secure your environment. Be aware that this book is written with WebSphere Application Server V6.1 in mind. It does not cover the new features and changes found in V7.0.

For up to date information about WebSphere Application Server V7.0, refer to the WebSphere Application Server Information Center. A good entry point to security topics is the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.doc/info/ae/ae/welc6topsecuring.html>



WebSphere Application Server Feature Packs

A WebSphere Application Server Feature Pack is an optionally installable product extension for WebSphere Application Server that provides a set of new related standards and innovative features. With feature packs, users can take advantage of these new standards and features without having to wait for a new release of WebSphere Application Server.

This chapter describes the feature packs currently available for WebSphere Application Server V7.0 and discusses the important aspects to be considered for a successful implementation.

This chapter contains the following sections:

- ▶ “Available feature packs” on page 408
- ▶ “WebSphere Application Server Feature Pack for Web 2.0” on page 408
- ▶ “WebSphere Application Server Feature Pack for Service Component Architecture” on page 412

13.1 Available feature packs

The currently available feature packs for WebSphere Application Server V7.0 are:

- ▶ WebSphere Application Server Feature Pack for Web 2.0
- ▶ WebSphere Application Server Feature Pack for Service Component Architecture (SCA)

Two previously available feature packs for WebSphere Application Server V6.1 have been integrated into WebSphere Application Server V7.0 and therefore you can use their features without additional installation procedures. The integrated features packs are:

- ▶ WebSphere Application Server Feature Pack for Web Services
- ▶ WebSphere Application Server Feature Pack for EJB 3.0

13.2 WebSphere Application Server Feature Pack for Web 2.0

This section describes the IBM WebSphere Application Server V7.0 Feature Pack for Web 2.0. It has a short introduction to Web 2.0 technologies and an overview of the contents of the feature pack.

13.2.1 Introduction to Web 2.0

Web 2.0 is a term that describes a set of new technologies designed and created to improve existing technologies in the World Wide Web (WWW). Web 2.0 is not a specification and is subject then, to different definitions and interpretations. In this publication we have considered those technologies included in the feature pack.

Key technologies and concepts

The main technologies and concepts being used for building Web 2.0 applications are described in this section.

- ▶ Asynchronous JavaScript and XML (Ajax)

Ajax is an open technique for creating rich user experiences in Web-based applications that do not require Web browser plug-ins. It is the most important technology in the Web 2.0 world. Its main characteristic is that, by exchanging small amounts of data with the server, it renders Web pages in the Web

browser without having to reload entire pages, increasing the interactivity, usability, and speed of the user experience. Behind the scenes, this means that Ajax implements a stateful client that interacts asynchronously with the server.

▶ Representational State Transfer (REST)

REST is a software architecture for building hypermedia systems. It consists of a series of principles that outline how resources are defined and addressed. Any internet application should follow these principles, but some applications violate them (for instance, having large amounts of server-side session data). Ajax, on the contrary, follows the key principles of REST.

▶ JavaScript Object Notation (JSON)

JSON is a human readable data interchange format that is used in Ajax as an alternative format to XML. Although it is based on a subset of the JavaScript language, it is language-independent.

▶ Atom

Atom is composed of two different standards:

– Atom syndication format

Atom syndication format is an XML language used for Web feeds, which are a mechanism for publishing information and subscribing to it.

– Atom publishing protocol

Atom publishing protocol is an HTTP-based protocol for creating and updating Web applications.

Atom was developed as an alternative to RSS feeds in order to overcome RSS incompatibilities.

13.2.2 Overview of the Web 2.0 feature pack

This feature pack extends Service Oriented Architecture (SOA) by connecting external Web services, internal SOA services, and Java Platform, Enterprise Edition (Java EE) objects into highly-interactive Web application interfaces. It provides a supported, best-in-class Ajax development toolkit for WebSphere Application Server. It also provides a rich set of extensions to Ajax.

The included extensions to Ajax are as follows:

- ▶ Client side extensions
 - Ajax client runtime

The main component of the feature pack. It includes extensions to the Dojo toolkit for building Ajax applications.
 - IBM Atom library

IBM Atom library consists of utility functions, data models and widgets for creating, modifying and viewing Atom feeds.
 - IBM gauge widgets

Two gauge widgets are provided as base objects to build other widgets on them: the analogue gauge and the bar graph.
 - IBM SOAP library

IBM SOAP library implements a simple way to create a SOAP envelope around a request and a widget to connect to external SOAP services.
 - IBM OpenSearch library

IBM OpenSearch library provides an interface to servers that implement the OpenSearch specification.
- ▶ Server side extensions:
 - Ajax proxy

Ajax proxy is an implementation of a reverse proxy that accepts requests from a client and redirect them to one or more servers while letting the client believe that all the responses from its requests come from the same server.
 - Web messaging service

Web messaging service is a publish and subscribe implementation that connects a Web browser to the WebSphere Application Server service integration bus for server-side event push.
 - JSON4J libraries

JSON4J libraries are an implementation of a set of JSON classes.
 - RPC adapter libraries

RPC adapter libraries are the IBM implementation for Web remoting, which is a pattern that provides the ability to invoke Java methods on the server side from JavaScript.
 - Abdera-based feed libraries

Abdera-based feed libraries are an implementation of Apache Abdera, which is an open-source project providing feed support.

Figure 13-1 shows the main components of this feature pack.

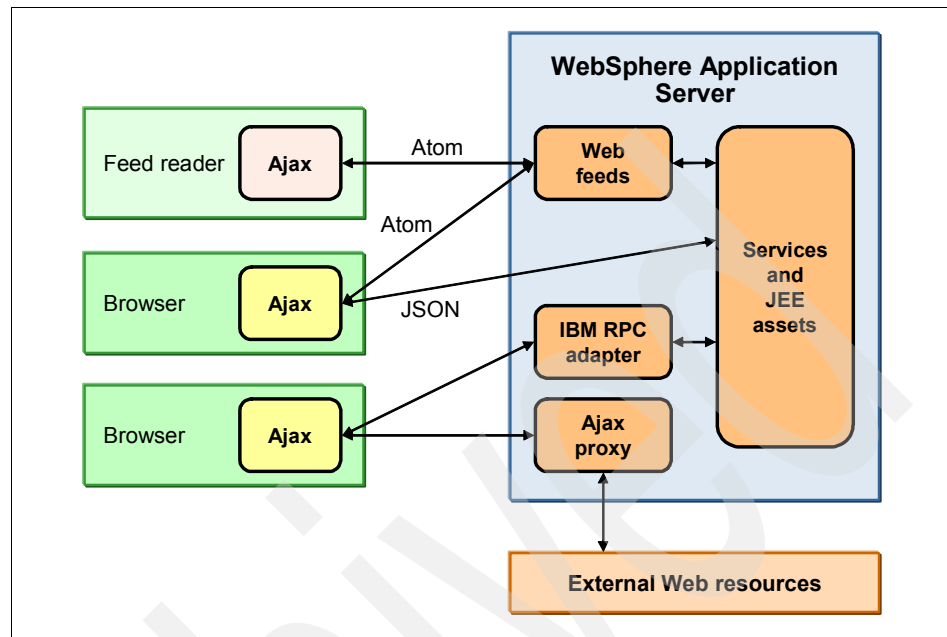


Figure 13-1 Components of WebSphere Application Server Feature Pack for Web 2.0

13.2.3 Security considerations

The security issues on Ajax are well known. Different forms of attacks can take advantage of the usage of client-side scripts that can be easily forged to consume information from untrusted sources or to collect confidential data from the user.

Adopting some measures may lead to more secure applications. Input validation, proper coding, loading scripts only from trusted sources, encryption, and a correct server security configuration on the server side are some of the measures you should consider.

13.2.4 Resources

For a complete overview and detailed information about the Feature Pack for Web 2.0, refer to the WebSphere Application Server V7.0 Information Center at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.web20fep.multiplatform.doc/info/welcome_nd.html

Another good source of information is IBM Redbooks publication *Building Dynamic Ajax Applications Using WebSphere Feature Pack for Web 2.0*, SG24-7635. This book contains an in-depth description of this feature pack and it is an essential reference for those planning to use it.

13.3 WebSphere Application Server Feature Pack for Service Component Architecture

This section describes the new IBM WebSphere Application Server V7.0 Feature Pack for Service Component Architecture (SCA). It introduces SCA and explains what is in the feature pack.

13.3.1 Introduction to SCA

SCA is a set of specifications that constitute a programming model for building applications using a service-oriented architecture (SOA). SCA extends other SOA technologies, like Web services, while providing a platform and language-neutral component model based on open standards specified by the Open SOA Collaboration (OSOA).

The main objective of SCA is allowing the creation of complex composite applications based on previously existing service components.

Key principles

SCA is based on three key principles:

- ▶ Service composition

SCA offers a composition model that allows you to build new services from existing software components. SCA provides the metadata for describing these components and the connections between them while hiding their inner workings.

- ▶ Service development

SCA has a language-neutral programming model. There are language-specific models for Java, Spring, C++, and other languages. Because SCA defines a common assembly mechanism, the language used for implementing a service does not need to be known by the service consumer.

- ▶ Service agility and flexibility

The component model provided by SCA makes the composition and assembly of business logic simple and allows a flexible reusability of components. A component can be easily replaced by another component providing the same service.

Key concepts

These are the key concepts contained in SCA:

- ▶ **Component**

This is the basic element of SCA. It is a configured instance of an implementation. Components encapsulate business functions. They have three configurable aspects:

- Services
- References
- Properties

- ▶ **Implementation**

The implementation is the actual code providing the component's functions.

- ▶ **Composite**

Also called composition or component assembly, composites are combinations of components.

- ▶ **Domain**

A domain contains one or more composites running in a single-vendor environment. Its components may be running on one or more processes and on one or more machines.

- ▶ **Service**

A service is the interface used by a consumer of the component. It specifies the operations that can be accessed by the component's client, but it does not describe how the communication happens.

- ▶ **Property**

Properties are configurable values that affect the behavior of a component.

- ▶ **Reference**

References describe the dependencies of a component on other software. Like services, they do not describe the way they have to communicate with that other software.

- ▶ **Binding**

Bindings specify how the communications with other components have to be done. A component only needs bindings for communications outside its domain because the runtime determines which bindings to use inside a domain. Multiple bindings allow different ways of communication with the component.

- ▶ **Wire**

A wire represents a relationship between a reference and a service, showing the existing dependency of a component on another component.

► Promotion

When a component's service is made available outside the composite, it is being promoted. A promotion also occurs when a reference must become a reference for the composite.

Figure 13-2 is an example of the main concepts of SCA.

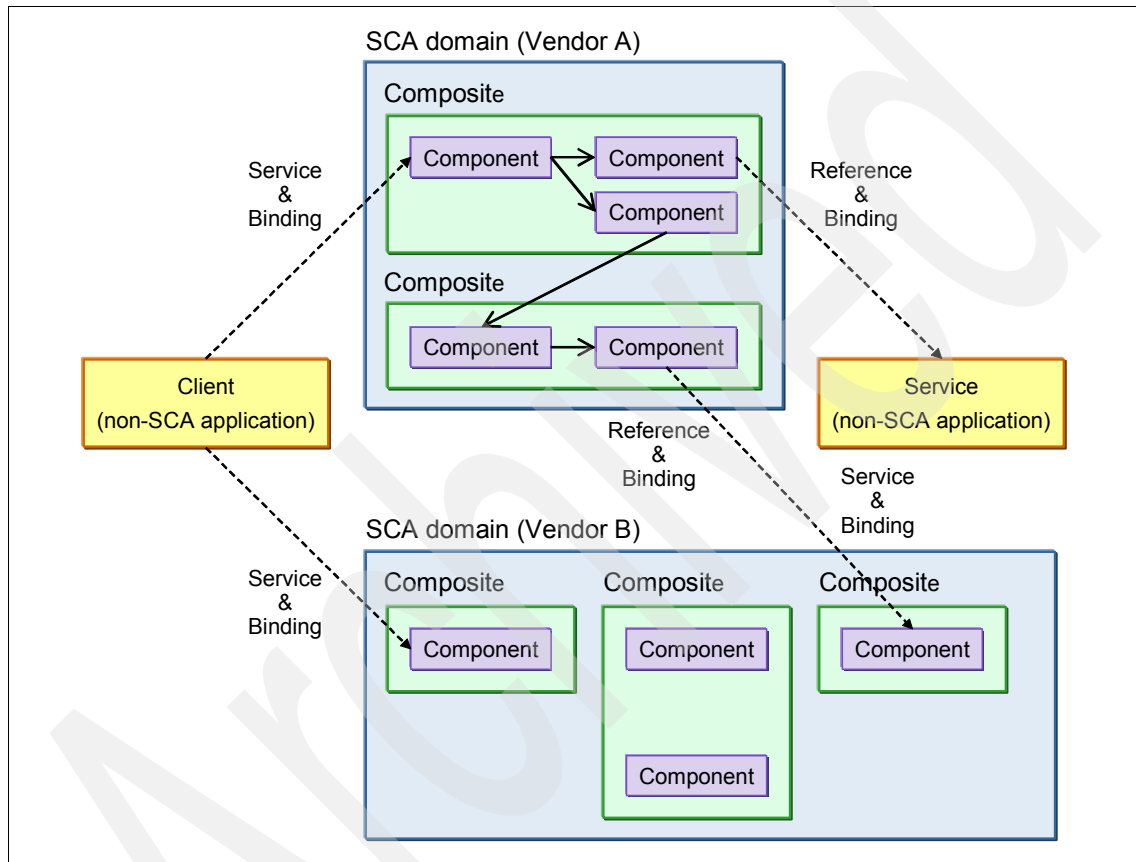


Figure 13-2 Key SCA concepts

A complete reference for SCA specifications can be found on the Open SOA Collaboration web site:

<http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications>

13.3.2 Overview of the SCA feature pack

The WebSphere Application Server Feature Pack for SCA is based on a Tuscany open source Java implementation and covers SCA 1.0.

Note: At the time of writing, the WebSphere Application Server V7.0 Feature Pack for SCA is a beta release. This chapter contains information based on this beta release.

Features covered

This feature pack follows the specifications as documented by the OSOA. Because OSOA does not define a set of compliance test suites, the implementation provided uses the following specifications as guiding principles. However, IBM is committed to providing an implementation that adheres strictly to the IBM interpretation of the specifications list below:

- ▶ SCA Assembly Model V1.0
This specification describes a model for the assembly of services as a composition of tightly or loosely coupled services. It is also a model for applying infrastructure capabilities, including security and transactions, to services and its interactions.
- ▶ SCA Java Component Implementation V1.0
This specification defines how a component must be implemented in Java, including its services, references, and properties. It extends the SCA Assembly Model and requires all the APIs and annotations defined in the SCA Java Common Annotations and APIs.
- ▶ SCA Java Common Annotations and APIs V1.0
SCA Java Common Annotations and APIs V1.0 specifies a Java syntax, including a set of APIs and annotations, for the programming concepts defined in the assembly model.
- ▶ SCA Policy Framework V1.0
This framework allows the definition of qualities of service (QoS). It supports the specification of some non-functional requirements of components (constraints, capabilities, and expectations). It is based on other standards, such as WS-Policy and WS-PolicyAttachment.
- ▶ SCA Transaction Policy V1.0
The transaction policies defined by this specification provide transactional quality of service to components (implementation policies) and their interactions (interaction policies).

- ▶ SCA Web Services Binding V1.0
This specification defines how an SCA service can be made available as a Web service and how an SCA reference can invoke a Web service.
- ▶ SCA EJB Session Bean Binding V1.0
Session bean bindings satisfy the need to expose EJBs as SCA services and to expose SCA services to clients based on the EJB programming model.

Note: As required by the SCA Assembly Model V1.0 specification, the feature pack for SCA also provides support for SCA bindings (also known as SCA default bindings).

Unsupported SCA specifications

Some sections of the SCA specifications covered by the feature pack are not supported in this version. For a complete list of unsupported sections, see the IBM WebSphere Application Server V7.0 Feature Pack for SCA Beta Information Center.

About this Information Center: At the time of writing, this Information Center cannot be accessed online. It has to be downloaded and installed on your computer. Follow the instructions in WAS_V70_SCAFP_information_Center.pdf, available at the following Web page:

<https://www14.software.ibm.com/iwm/web/cc/earlyprograms/websphere/soawas61/library.shtml>

Application packaging and management

WebSphere Application Server V7.0 Feature Pack for SCA adds support for deploying SCA applications to the application server. Both JAR and WAR files are supported.

Components that include service definitions must be packaged in a JAR file and deployed as assets for business-level applications.

SCA WAR files can be deployed as well, provided that they do not expose services over any binding type. WAR files must be deployed as WebSphere enterprise applications.

The feature pack also provides support in the Integrated Solutions Console, and for the `wsadmin` command tool, to install, delete, start, and stop SCA applications.

13.3.3 Other considerations

If you plan to use the feature pack for SCA, you must take into account the considerations detailed in the following sections.

Supported platforms

At the time of writing, the supported platforms for the beta release of the feature pack are as follows:

- ▶ Windows 2003 SP2 Standard and Enterprise
- ▶ SuSE Linux Enterprise Server 10 on VMWare Server (1.0.x)

Profile creation

After installing the feature pack, a new set of profile types are available. These profile types enable the features provided by the feature pack.

In order to deploy SCA applications, create a new profile with SCA enabled or augment an already existing profile.

Application development support

A developer using Rational Application Developer for WebSphere Software V7.5 for building SCA applications has to consider the following issues:

- ▶ Use of SCA Java annotations requires a Java compiler compliance level to be 5.0.
- ▶ Needed SCA JAR files have to be added to the classpath.
- ▶ Rational Application Developer does not provide code generation utilities or SCA tools.

13.3.4 Resources

You can get more information about the WebSphere Application Server V7.0 Feature Pack for SCA beta release at the following Web page:

<https://www14.software.ibm.com/iwm/web/cc/earlyprograms/websphere/soawas61>

Archived

WebSphere Application Server for z/OS

This chapter concentrates on the WebSphere Application Server for z/OS v7.0 features. It has the following sections:

- ▶ “WebSphere Application Server structure on z/OS” on page 420
- ▶ “What is new in V7.0” on page 432
- ▶ “WebSphere Application Server 64-bit on z/OS” on page 433
- ▶ “Load modules in the HFS” on page 436
- ▶ “XCF support for WebSphere HA manager” on page 438
- ▶ “z/OS Fast Response Cache Accelerator” on page 446
- ▶ “Thread Hang Recovery” on page 452
- ▶ “Installing WebSphere Application Server for z/OS” on page 456
- ▶ “System programmer considerations” on page 466
- ▶ “Planning checklist” on page 475

The features and functions described in this chapter are only available when the WebSphere Application Server for z/OS V7.0 is used. The new functions and features of WebSphere Application Server for z/OS V7.0, as well as the existing ones described in the other chapters of this book, are implemented on the z/OS platform but are not explicitly mentioned in this chapter.

14.1 WebSphere Application Server structure on z/OS

This section shows the added value that the implementation for WebSphere Application Server for z/OS offers compared to the distributed versions.

For those who may be not familiar with the z/OS operating system, we have included some boxes that explain z/OS terms or techniques in general IT terminology and how they might add value for your business environment.

14.1.1 Value of WebSphere Application Server on z/OS

WebSphere Application Server for z/OS V7.0 combines IBM's leading application server with the high-end server platform, z/OS. This combination offers the following unique features that can be of great value to your environment and your business:

- ▶ Service Level Agreements with Workload Management and local connections to back end

The Workload Manager component that is exploited by WebSphere Application Server for z/OS V7.0 automatically assigns resources to the application server to achieve the performance goals set for the environment. These goals can be set on a transaction level, assuring that your platinum customers get the best response time.

Local connectors to databases running in the same operating system image will enhance the throughput and decrease the used CPU resources.

- ▶ High availability reduces downtime costs

The proven technologies of the System z hardware and operating system have the highest availability in the industry. WebSphere Application Server for z/OS can directly benefit from this. In addition, the structure of the application server has been modified to expand this high availability into WebSphere Application Server itself, by forming a mini-cluster inside each application server, if activated by the administrator.

The usage of a Parallel Sysplex, the z/OS cluster technique, increases the uptime of the environment up to 99.999%. In the case of unplanned downtime, System z and z/OS offer great disaster recovery capabilities, bringing the system to a productive, industry leading state.

- ▶ Reduced management cost through excellent manageability

The management capabilities of the z/OS platform have evolved over the last 40 years, resulting in the platform with the lowest management costs in the industry while maintaining the highest degree of automation and transparency for the administrators.

- ▶ Reduced cost through lowest Total Cost of Ownership (TCO) in the market
System z is well known for to have the best TCO in the IT market. Independent consulting companies have shown that the usage of a modern mainframe will outperform distributed environments that may be cheap to purchase but more expensive to maintain. WebSphere can fully exploit features like the System z Application Assist Processor (zAAP) to reduce the software cost and the overall cost of the environment.
- ▶ Secure environment to stabilize operations and production
The usage of a central security repository, the z/OS Resource Access Control Facility (RACF) will ease the security model, because it can be used for user authentication, authorization, and the role-based security model offered by Java.
The security model of the operating system prevents unauthorized user code to harm the system and bringing down your environment.

14.1.2 Common concepts

Before we explain the enhancements, we introduce some commonalities to both implementations of the WebSphere Application Server V7.0:

- ▶ All WebSphere Application Server components described in this book are common to both distributed and z/OS platform. This includes nodes, cells, clusters, core groups, job manager, administrative agent, deployment manager, administrative console, and all the other components.
- ▶ Experience has shown that applications that run inside a WebSphere Application Server on Windows, AIX, Linux, Solaris, and so forth can also run on WebSphere Application Server for z/OS, if the application meets the requirements common to the product. Some minor modifications might be required when changing the underlying operating system.
- ▶ WebSphere Application Server administrators will find the usual control options and Web interfaces on z/OS (except the update installer).

z/OS as the operating system for WebSphere Application Server:

Using z/OS as the underlying operating system for WebSphere Application Server does not mean rebuilding your processes for administration, operation, and development, or that your administration staff needs to learn a new product.

The WebSphere API's are the same, while the z/OS operating systems offers additional capabilities that simplifies things and provides high-availability, disaster recovery, performance settings, and management options.

14.1.3 Cell component—daemon

WebSphere Application Server for z/OS introduces the location service daemon (LSD), a WebSphere cell component exclusive to the z/OS platform.

A daemon, in WebSphere Application Server for z/OS terminology, is the Location Service Agent. It provides the location name service for external clients. There is one daemon per cell per z/OS image. If your cell consists of multiple z/OS images, a daemon will be created for each z/OS image where your cell exists. If there are two cells on the same z/OS image, two daemons will be created. See Figure 14-1.

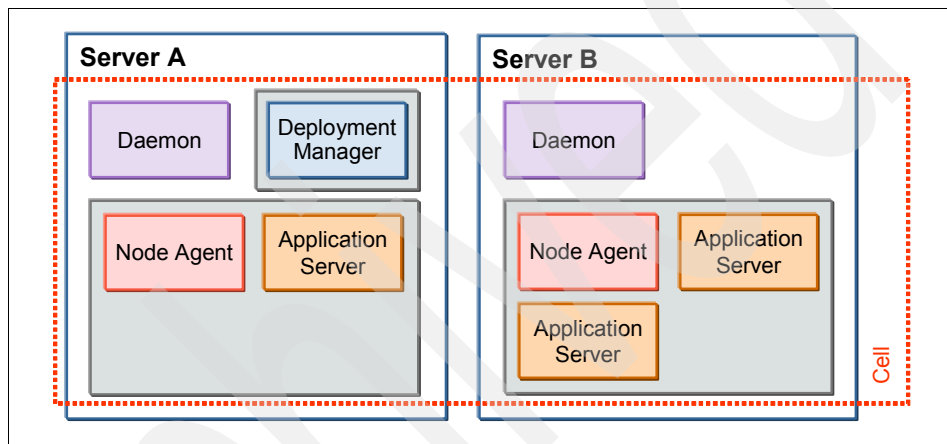


Figure 14-1 WebSphere Application Server for z/OS daemon usage in a cell

Daemon servers are started automatically when the first server for the cell on that z/OS image is started (actually when the first control region is started). If you terminate a daemon, all the WebSphere Application Server components for the cell on that z/OS image come down.

The daemon is created as part of the normal server customization process.

14.1.4 Structure of an application server

This section gives a conceptual view of an application server inside WebSphere Application Server for z/OS.

Overview

WebSphere Application Server for z/OS in general uses the same concepts to create and manage an application server. Each application server (or instance of a profile) is built out of multiple building blocks, to represent a single application server:

- ▶ Control region
- ▶ Servant region
- ▶ Control region adjunct

Figure 14-2 shows these basic building blocks and how they will form the application server. The communication between the control region and the servant regions is done using WLM queues, because the communication with the outside world is ended in the control region.

What is a WLM queue? A WLM queue is used to queue work for further processing. Each queue uses a first-in-first-out mechanism. Because it is possible to use different priorities for work requests, there are multiple queues, one for each priority. Servant regions are bound to a priority and therefore take work from the queue with the priority they are bound to.

A WLM queue is a construct allowing you to prioritize work requests on a transaction granularity, compared to server granularity on a distributed environment.

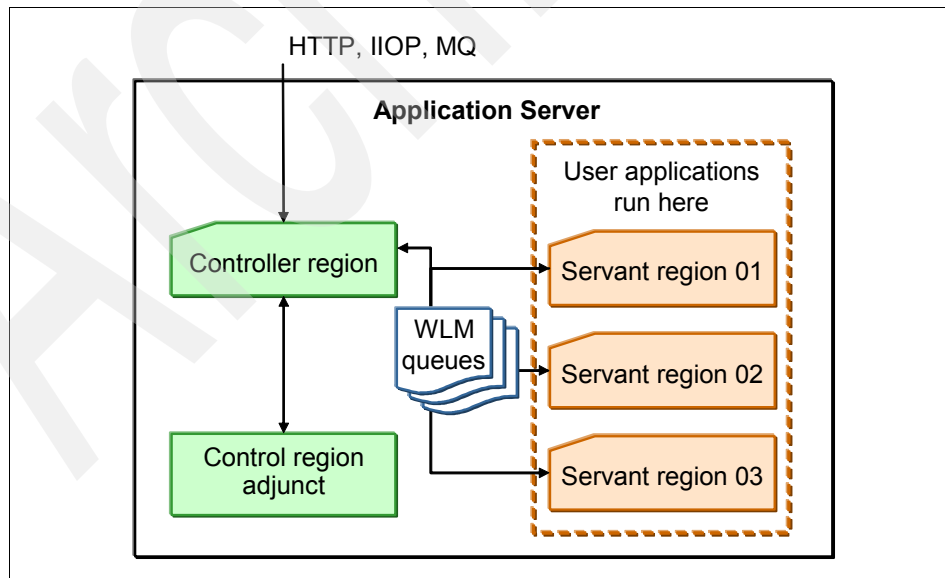


Figure 14-2 Building blocks of the WebSphere Application Server for z/OS V7.0

While WebSphere Application Server profiles on z/OS are built using multiple building blocks, it is still a single instance of an application server from an application developer, system administrator, or user perspective. This means that nearly all WebSphere variables can be defined against a server, and are not defined against the servant and control region adjuncts. However some of the settings, like heap sizes have to be done for each of the components.

All the possible profiles that can be instantiated, are built using the different regions:

- ▶ Application server
- ▶ Deployment manager
- ▶ Job manager
- ▶ Administrative agent

Control region

The control region is the face of the application server to the outside world. It is the only component that is reachable from the outside world using standard protocols and port communication. For communication with the servant regions, where the application is executed, the control region is the endpoint for TCP transportation and switches to WLM queues.

Keep the following points in mind about control regions:

- ▶ An application server can only have one control region.
- ▶ The control region contains a JVM.
- ▶ The control region will be start- and endpoint for communication.

Servant region

The servant region is the component of an application server on z/OS where the actual application is run and transactions are processed. The EJB and Web container are included here.

As seen in Figure 14-2 on page 423, it is possible to have multiple servant regions per application server. (But only one for the other profile types.) This concept is called a *multi-servant region* or *internal cluster*. Through the usage of this technique, it is possible to actually benefit from cluster benefits without the overhead of a real cluster. For continuous availability and scalability, it is suggested to build a WebSphere Application Server cluster that integrates these mini clusters. While creating a normal cluster you can still use multiple servant regions for each cluster member.

Keep in mind the following information about servant regions:

- ▶ Each servant region contains its own, independent JVM.
- ▶ All servant regions are identical to each other.
- ▶ An application runs on all servant regions connected to an application server, because it is deployed at server scope.
- ▶ An application must be WebSphere Application Server cluster-ready to use the multi-servant concept.
- ▶ The number of servant regions is transparent to the user and the application.
- ▶ Servant regions can be started dynamically by the WLM component, if response times of user transactions do not meet the defined goals. The defined maximum is the limit.
- ▶ If a single servant fails, the others will still run, keeping the application alive. Only the transactions of the crashed servant region will fail and deliver errors to the user. The other servant regions will continue to work.
- ▶ Failed servant regions will be restarted automatically by the operating system providing a miniature automation.

Note: When determining the maximum number of servant regions, make sure that the system has enough resources to use them all.

Control region adjunct

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

14.1.5 Runtime processes

This section describes the runtime behavior of the WebSphere Application Server for z/OS V7.0.

Overview

The non-z/OS platforms are built on a single process model. This means the entire application server runs in one single Java Virtual Machine (JVM) process. WebSphere Application Server for z/OS is built using a federation of JVM's, each executing in a different address space. Together, such a collection represents a single server instance, as described in Figure 14-2 on page 423.

What is an address space? An address space can be best compared to a process in the distributed world. Instead of running processes, the z/OS operating system uses a concept, called address spaces. Technically, an address space is a range of virtual addresses, that the operating system assigns to a user or separately running program, like the WebSphere Application Server for z/OS. This area is available for executing instructions and storing data.

During runtime each building block of an application server or a deployment manager opens an address space, as seen in Figure 14-3.

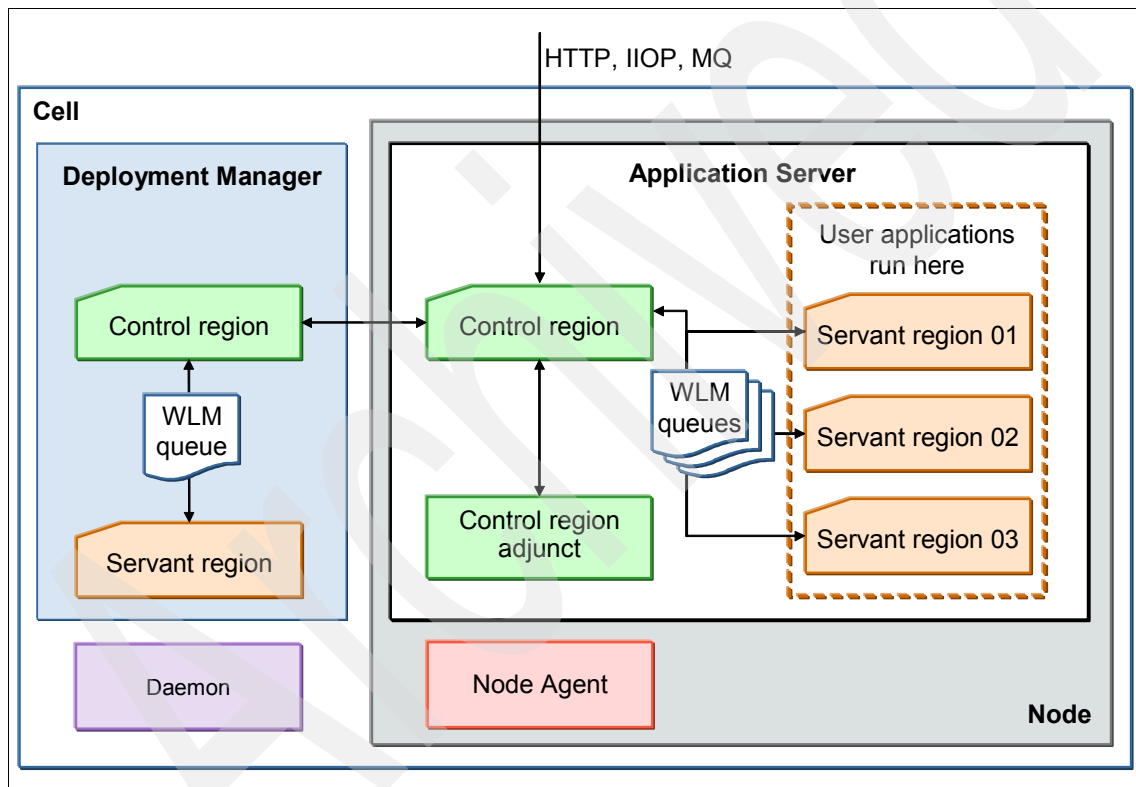


Figure 14-3 Runtime architecture of a WebSphere Application Server for z/OS Network Deployment cell

This means that for the WebSphere Application Server for z/OS environment shown in Figure 14-3 on page 426, there will be at least eight address spaces:

- ▶ Deployment manager control region
- ▶ Deployment manager servant region
- ▶ Location Service daemon
- ▶ Application server control region
- ▶ (Optional) Application server control region adjunct
- ▶ Application server servant region for each servant (here three)
- ▶ Node agent

A stand-alone server installation would consist of at least three address spaces:

- ▶ Location Service daemon
- ▶ Application server control region
- ▶ Application server servant (assumed that one servant is used)
- ▶ (Optional) Application server control region adjunct

Java Virtual Machine

Each control region and each servant region contains a JVM. For the installation shown in Figure 14-3 on page 426, this means that there are six JVMs, because we have two control regions and four servant regions.

These JVMs have special purposes. The control region JVM is used for communication with the outside world, as well as for some base WebSphere Application Server services. The servant region JVM executes the user application. So we can speak of specialized JVM's on z/OS. This specialization reduces the maximum amount of heap storage defined for the various heaps, because not all data and meta-data needs to be loaded and kept inside the memory. It also separates the user data from most of the system data needed to run the WebSphere Application Server base services.

The high number of JVMs has some implications to the system requirements as well as to the sizing of the heap. This is described in 14.8.2, "Installation considerations" on page 457:

- ▶ Amount of real storage
- ▶ Min/max size for the different heaps
- ▶ Shared class cache usage

Note: The shared class cache is a construct introduced with the JDK 5.0. The shared class cache can be used to share the content of a JVM into other JVMs. For more information about z/OS settings for the shared class cache see 14.9.3, "Java settings" on page 469.

14.1.6 Workload management for WebSphere Application Server for z/OS

This section focuses on how WebSphere Application Server for z/OS exploits the WLM subsystem of z/OS.

Workload management overview

WebSphere Application Server for z/OS V7.0 can exploit the Workload Manger (WLM) subsystem of z/OS in the following ways:

- ▶ Workload classification
 - Coarse-grained workload management on server base
- ▶ Transaction classification
 - Fine-grained workload management on transaction level
- ▶ Servant activation
 - Start additional servant regions for application processing

To use fully the provided capabilities of WLM, some configuration needs to be performed. Refer to the Information Center for a detailed step-by-step approach.

Before we go into more detail on the enhancements that the WLM offers to the WebSphere Application Server for z/OS, lets briefly explain the concepts of service classes, reporting classes and enclaves.

Service classes

A service class is the z/OS implementation of a service level agreement. A service class is used to set performance goals for different work (like incoming requests, applications or operating system tasks).

For example, a service class can be told to achieve a response time of 0.5 seconds 80% of the time for incoming requests. The WLM component of z/OS will then automatically assign resources (processor, memory, and I/O) to achieve the goals. This is done by comparing the definitions of the service class to real-time data on how the system is currently performing.

You can have multiple service classes, with multiple goals. The mapping of work to a service class is set up by the system programmer and can be based on a variety of choices, like user ID, application, or external source.

Reporting classes

While the system is processing work, a reporting class keeps track of what resources have been spent processing work. A reporting class is an administrative construct, used to keep track of consumed resources. Each unit of

work, processed by the system is charged into one reporting class. The decision of what work should be put into which report class can be defined by the z/OS system programmer (system administrator).

This grouping of used resources can then be used to tune the system or to create a charge-back to the departments using the systems. To create reports, the Resource Measurement Facility (RMF) is used.

Enclaves in an WebSphere Application Server for z/OS environment

An enclave is used to assign the user application a service class during runtime. An enclave can be thought of as a container that has a service class and reporting class attached to it. A thread can connect to this enclave and execute the thread's work with the priority of the enclave.

WebSphere Application Server for z/OS uses this technique to pass transactional work, the user application, from a servant to an enclave. The application then runs with the priority of the enclave and WLM can make sure that the performance goals for the application are achieved.

Workload classification

WebSphere Application Server for z/OS V7.0 and its prior versions are capable of classifying incoming work on a server base. To do this, the control region of an application server checks to which application server the request belongs. It will then assign the request to a WLM queue. Each servant will process work for one service class at any point in time.

As seen in Figure 14-4, incoming work is assigned a service class, based on information of the user- work request. The granularity is on the application server level.

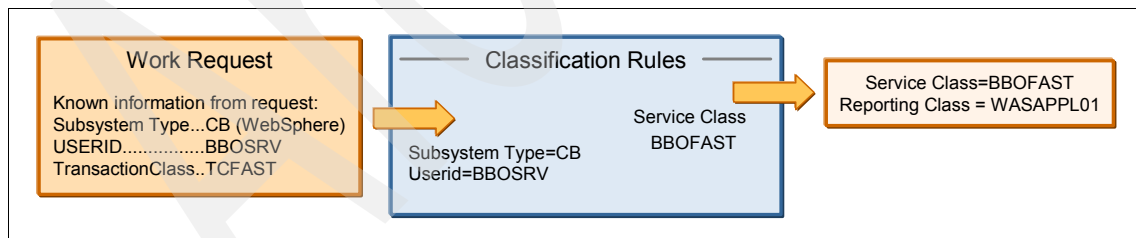


Figure 14-4 Workload classification for WebSphere Application Server for z/OS

Transaction classification

Transaction classification can be used to classify the transactions handled by your application. This technique could be used to prioritize special requests. A good example is a Web store that classifies its customers in gold and platinum customers, giving the platinum customers a better response time than the gold customers (Figure 14-5).

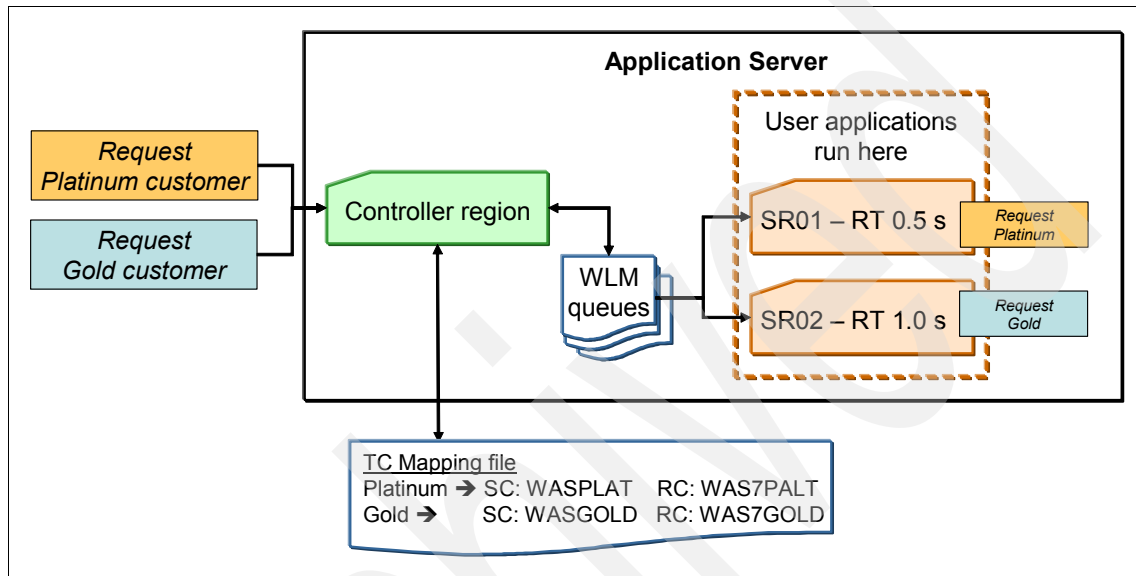


Figure 14-5 Transactional assignment of performance goals

A request that enters the system is assigned a transaction class, using the request details like used protocol, requested URI, or other metrics. The transaction class is then mapped to a service and reporting class inside the WLM subsystem, using a *workload classification document*. This is an XML file that has to be populated with mapping rules.

A workload classification document can be used with the following protocols:

- ▶ Internal classification
- ▶ IIOF classification
- ▶ HTTP classification
- ▶ MDB classification
- ▶ SIP classification

To use this technique the following steps need to be performed.

1. Talk with the application development and the business functions teams to define what transactions need performance goals.
2. Create a workload classification document.
3. Configure the server to use the classification document
4. Modify the WLM settings to use Transaction Classes.

For a detailed information, on the transaction classification refer to the Information Center article *Using transaction classes to classify workload for WLM*. This article contains links to all information sources needed, as well as samples. It can be found at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0//index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rweb_classservers.html

Servant activation

As described in “Servant region” on page 424, an application server can have multiple servant regions defined that all process the user application. If the response time goals defined for the applications cannot be kept, WLM can start additional servant regions. As within a normal cluster, incoming, or queued up, requests can now be processed faster.

The minimum and maximum number of the servant regions can be defined by the system programmer.

14.1.7 Benefits of z/OS

After reading about the deep integration you might ask: So what is the value for me? This section addresses benefits of using WebSphere Application Server V7.0 for z/OS from a security, availability, and performance perspective.

Security

In terms of security, the distinct area for user code offers you more protection for other system components running in the same Logical Partition (LPAR). In general the application server itself has more rights than the applications running inside it. This is necessary to make sure that the server can access all needed files, execute scripts, and so forth. In WebSphere Application Server for z/OS these basic functions are performed in the control region. However, the user code is executed in the servant region that generally has almost no rights. It is not possible to negatively influence system resources and services from inside the application.

Availability

The concept of a separate servant and control region greatly enhances the availability of a user application.

- ▶ Multiple servant regions can form a kind of vertical cluster running your application. In the case that one servant region goes down, users with in-flight transactions in that servant will receive an error. The other servant regions will continue to work and respond to requests, so the overall application is still available and new requests can enter the system. z/OS will automatically re-launch the failed servant.
- ▶ The control region might be identified as a single point of failure. Although the control region is unique per application server, the risk of failure is low. Only WebSphere Application Server for z/OS product code is executed in this region. To remain available with your application, keep in mind that it is also possible to create a WebSphere Application Server cluster, as in an distributed environment.

Performance

From a performance point of view the concept of different regions and the usage of WLM greatly enhances performance and scalability.

- ▶ Performance improvements are achieved by creating multiple servant regions, because more requests can be processed in parallel if the system has enough resources available.
- ▶ You can set detailed performance targets on a transactional level for the response time. The system will automatically adjust resources on a 7x24x365 basis, to make sure that these goals are kept.
- ▶ Through the usage of multiple JVMs with smaller heaps, the penalty a single transaction has to pay during garbage collection, will decrease and general throughput will increase.

14.2 What is new in V7.0

WebSphere Application Server V7.0 in general offers some new concepts, functions, and features. Refer to Chapter 3, “WebSphere Application Server concepts” on page 51 for an overview of these concepts, functions, and features.

The following new additions are specific to WebSphere Application Server V7.0 for z/OS:

- ▶ 64-bit default address mode
- ▶ Load modules in the HFS
- ▶ Cross-coupling facility (XCF) support for HA manager
- ▶ z/OS Fast Response Cache Accelerator (FRCA)
- ▶ Thread Hang Recovery

Each of these new features is discussed in detail in this chapter.

14.3 WebSphere Application Server 64-bit on z/OS

This section focuses on the 64-bit mode of WebSphere Application Server that has become the default setting in V7.0.

For detailed information about the installation and checklists for the installation go to the *Installing your application serving environment* article in the WebSphere Application Server for z/OS V7.0 Information Center. The article can be accessed on the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.installation.zseries.doc/info/zseries/ae/welc6topinstalling.html>

14.3.1 Overview

With WebSphere Application Server for z/OS V7.0, a newly created application server is automatically configured to run in 64-bit mode. This overcomes the 31-bit storage limitation.

In a 31-bit server on a z/OS system, the maximum size of the JVM heap is limited to a value between 768 and 900 MB. Although it is theoretically possible to address 2 GB of storage with a 31-bit server, this limitation comes from the size of the private virtual storage under the 2 GB line of the z/OS address spaces. The private region is limited to approximately 1.4 GB. This amount of memory is used for the heap of the Java Virtual Machine (JVM) and other infrastructure.

The usage of 64-bit removes this limitation and allows the definition of much larger heap sizes.

Note: While the 31-bit mode can still be configured manually, it is not suggested, because this mode is deprecated in V7.0.

The migration of a server from V6.1 to V7.0 does not change the bit mode. New servers start in 64-bit, but migrated servers stay in whatever bit mode they were in before.

You also can switch a server from 31 to 64-bit and back. It is not a permanent decision made at configuration time.

Deprecation message with 31-bit mode

Because support for running a server in 31-bit mode is deprecated, whenever a server that is configured to run in 31-bit mode is started, a warning message is issued to the system log, as shown in Example 14-1. The `server_name` is the name of the server that is running in 31-bit mode:

Example 14-1 31-bit deprecation message in the z/OS system log

```
BB000340W: 31-BIT MODE IS DEPRECATED FOR THE APPLICATION SERVER RUNNING
ON THE Z/OS OPERATING SYSTEM. CONSIDER USING 64-BIT MODE FOR
server_name AS AN ALTERNATIVE.
```

14.3.2 Planning considerations

Because the 31-bit operation mode for WebSphere Application Server is deprecated in V7.0, all planning activities for new installations should be done using the 64-bit mode.

The points described in the following paragraphs should be taken into consideration when planning an installation.

All components support 64-bit

Make sure that all components used in your architecture support the usage of a 64-bit JVM. While virtually all actual versions of purchased software support the usage of 64-bit, this point might be of concern for user-built applications that are migrated from a 31-bit environment.

Real and auxiliary storage

The usage of 64-bit does not itself imply that the amount of storage used will increase significantly. In general there is an overhead of approximately 10% on the needed storage of a 64-bit implementation compared to 31-bit when the same JVM heap size should be used in the system.

Note: The use of the 64-bit mode of WebSphere Application Server for z/OS does not mean that the server needs enormous amounts of additional memory. The amount of memory used by your environment still is based on the following factors:

- ▶ Applications need for storage
- ▶ Memory settings defined by your administrator.

The difference from the 31-bit addressing mode is that it is theoretically possible to use larger amounts of memory with the 64-bit addressing mode. But the amount of storage needed only increases significantly if the heap sizes are increased significantly. For example, if your application needs large heaps.

This is directly controlled through administrator interaction. If the WebSphere Application Server administrator does not change the JVM memory settings, there is no need to increase the amount of real and auxiliary storage.

If applications need to use larger heap sizes than 900 MB, make sure that there is enough real and auxiliary storage available.

Effect on the system

Keep in mind that WebSphere Application Server for z/OS is part of a larger system: the z/OS operating instance and the broader Sysplex. Consider the bigger picture when thinking about increasing WebSphere for z/OS JVM heaps significantly.

14.3.3 Administration considerations

This section focuses on the changes that need to be made in order to run WebSphere Application Server in 64-bit (default) mode.

JCL parameters

If the application server needs an large amount of JVM heap, make sure that the following Jobcard parameters do not restrain the system:

- ▶ REGION setting on the JCL JOB or EXEC statement.

This setting specifies the maximum size of the execution region (between 0 M and 2 GB) for the step in this job. A value of 0 M means it will take what it needs in that range with no limit imposed. We suggest that you specify REGION=0M so as not to limit their size.

- ▶ **MEMLIMIT** setting in the JCL or in the PARMLIB member SMFPRMxx.
It specifies the limit on the use of virtual storage above 2 GB for a single address space. If you specify a JVM heap greater than 2 GB, it may extend up into this range. A value of MEMLIMIT=NOLIMIT means it will not be limited above the 2 GB bar.
Message BBOO0331I is issued during server startup to tell you what MEMLIMIT value was used for the address space, and where it came from (an exit, JCL, and so forth).

System exits

Make sure that the IEFUSI- and JES2/JES3 exits defined on your z/OS operating system do not limit the virtual region size for the WebSphere Application Server address spaces.

Note: Although this modification/check is stated in the installation guide, there is sometimes the tendency to skip over basic steps when you already have a version of the WebSphere Application Server installed on your system.

However due to the theoretically larger heap sizes the values needs to be adjusted in most environments.

14.4 Load modules in the HFS

This section describes the new way that z/OS components are installed in WebSphere Application Server for z/OS V7.0.

14.4.1 Overview

In previous releases of WebSphere Application Server for z/OS the product was shipped as both HFS content and z/OS datasets. This was different from the other platform packaging. It also required two sets of executables to be kept in sync. If not kept in sync, difficult-to-debug code-level-mismatch problems could occur.

In WebSphere Application Server for z/OS V7.0, the load modules ship in the AppServer/lib directory with the rest of the runtime. This change reduces the installation and management complexity, and eliminates a common source of errors.

14.4.2 Installation considerations

The contents of SBBOLOAD, SBBGLOAD, SBBOLPA, and SBBOLD2 will be moved to the HFS or zFS, as shown in Figure 14-6. Those modules that have to be in Link Pack Area (LPA) are placed there by WebSphere Application Server for z/OS using directed-load and dynamic LPA services.

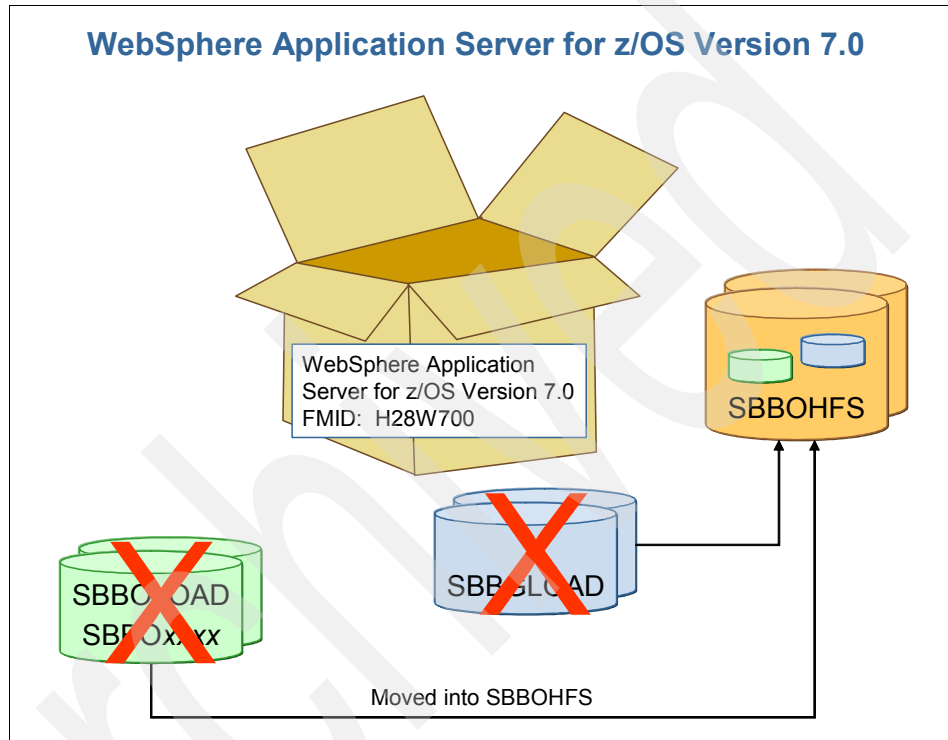


Figure 14-6 Load Modules moved into HFS

In addition, scripts are provided to help extract the load modules into a data set for those customers wishing to place the native runtime in the LPA.

Keeping the modules in the HFS is equivalent to STEPLIB in the server PROCs today.

14.4.3 HFS structure

The HFS structure of the WebSphere Application Server for z/OS V7.0 is built out of the components, seen in Table 14-1 (with default location).

Table 14-1 WebSphere Application Server for z/OS file system structure

F MID	Component	HFS path
H28W700	Application server itself	/usr/lpp/zWebSphere/V7R0
JIWO700	Optional Material	/usr/lpp/zWebSphere_OM/V7R0
JDYZ700	Secure Proxy Server	/usr/lpp/zWebSphere_SPS/V7R0
HHAP700	IBM HTTP Server V7	/usr/lpp/IHSA/V7R0

14.5 XCF support for WebSphere HA manager

This section describes the WebSphere Application Server support for the z/OS Cross Coupling Facility (XCF) protocol. It also gives an overview of WebSphere's HA manager and the default protocol used for discovery and failure detection in an application server cluster.

14.5.1 XCF support overview and benefits

WebSphere Application Server for z/OS V7.0 adds the option to use the XCF system services to monitor the status of cluster components instead of the default common code base technique. This kind of implementation has two main values:

- ▶ Reduced CPU Overhead

The usage of the XCF significantly reduces the overhead that comes through the ping packets sent by each core group member. This is noticeable during CPU idle times.

- ▶ Improved interval for failure detection

While the default interval used in the original protocol (180 seconds) was not convenient for every environment, the usage of the XCF system service reduces this time. The default settings provide information after 90 seconds. These values can still be adjusted by the system programmer.

Note: While the usage of the XCF system service is an option on the z/OS platform, the default setting for the core group member failure detection is the heartbeat technique. This default setting is chosen due to the common code base.

Due to the benefits of the XCF support for WebSphere HA manager, we suggest using the alternate protocol, if all prerequisites are met (14.5.5, “Activating XCF support for HA manager” on page 444).

14.5.2 WebSphere HA manager

The WebSphere high availability manager (HA manager) was introduced with version 6.0 to make sure that components within the WebSphere Application Server are always available. To achieve this, an HA manager instance runs within each component of a cell, including application servers, the deployment manager, and the node agent (Figure 14-7 on page 440).

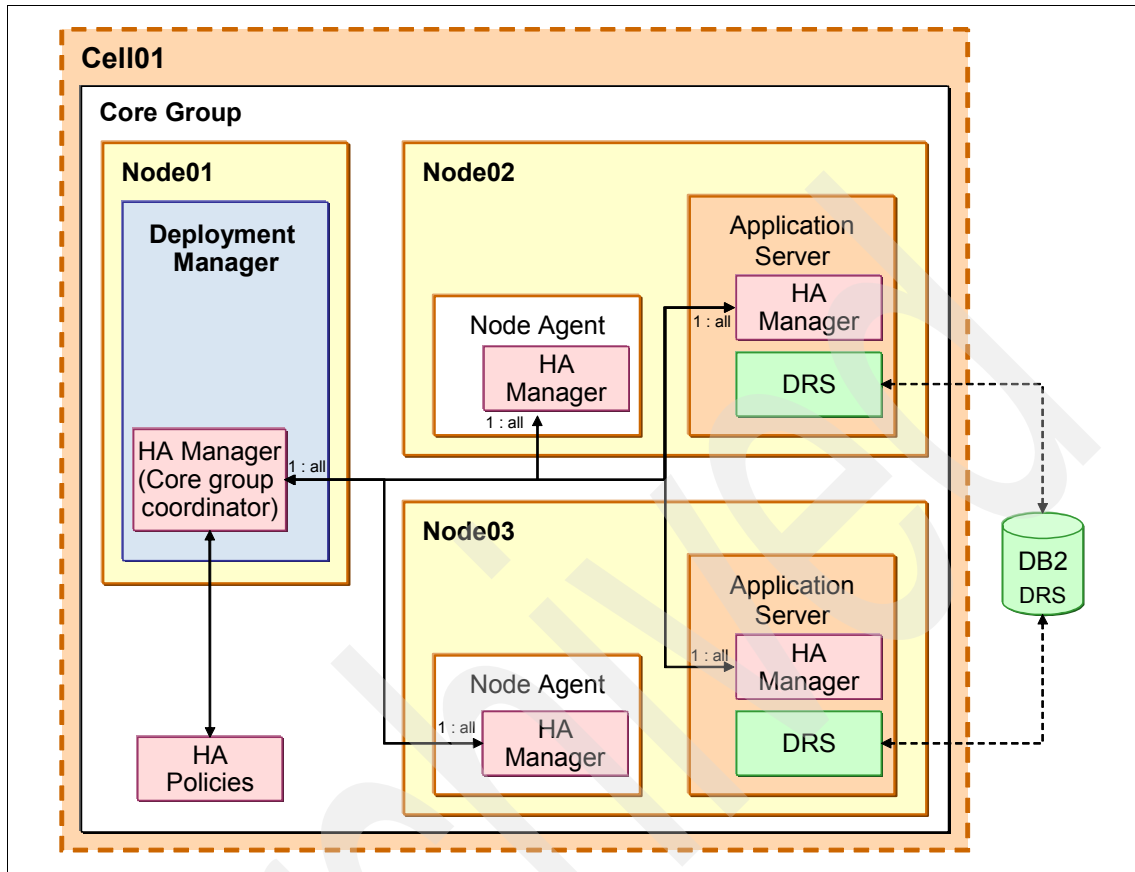


Figure 14-7 Logical concept of the HA manager

A cell can be divided into multiple availability domains, called *core groups*. Each server inside such a core group is monitored for startup and possible failure. To check if a component is still alive or if a new component is available, the core group discovery and failure detection protocol is used.

This protocol is part of the Distribution and Consistency Services (DCS). The DCS provides the following functions:

- ▶ Distribution of information among core groups
- ▶ Failure detection of said members/groups
- ▶ Forms the infrastructure used by the HA manager

14.5.3 Default core group discovery and failure detection protocol

The default implementation for the discovery and failure detection protocol works in two steps:

- ▶ Discovery service

The default service establishes network connectivity with the other members of the core group. To establish the connection, the Discovery Protocol retrieves the list of core group members and the associated network information from the product configuration settings. If the connection is successful, a message will be entered in the system log.

- ▶ Failure detection

The failure detection protocol uses two different techniques to monitor the status of the core group members. It listens for sockets used by core group members to be closed by the operating system. It sends ping packets as a heartbeat to each member. If the ping fails for a certain number of consecutive packet losses, the corresponding member is marked as down. For each kind of failure, a log entry will be issued.

After a failure is detected, the corresponding HA policy for the failed member is initiated by the HA manager to recover the member.

Because each core group member communicates with all other members of the core group, the amount of CPU cycles that the Discovery Protocol task consumes increases proportional to the number of core group members. Therefore, the default settings are a balance between CPU consumption and timely failed member detection.

While in V6 it was only possible to modify the protocol settings by creating custom variables, WebSphere Application Server V7.0 now offers the option to modify the values directly through the Integration Solutions Console. Simply navigate to **Servers** → **Core Groups** → **Core Group settings** → **(your_core_group)** → **Discovery and failure detection** and change the values.

When the core group contains members that are on version 6.x of WebSphere Application Server, then you still have to define the custom variables:

- ▶ IBM_CS_FD_PERIOD_SECS
- ▶ IBM_CS_FD_CONSECUTIVE_MISSED

Health check: If you want to check whether you have any issues with core group members, check the server output for WebSphere messages DCSV1113W or DCSV1111W for detection failures. Message code DCSV1111W reveals if any core group members are marked as failed.

14.5.4 XCF—alternative protocol on z/OS

This section describes how the XCF system service is used to enhance the WebSphere Application Server for z/OS V7.0 DCS protocol.

WebSphere Application Server for z/OS architectural change

From an architectural side, the usage of XCF adds some components to a WebSphere environment, as shown in Figure 14-8.

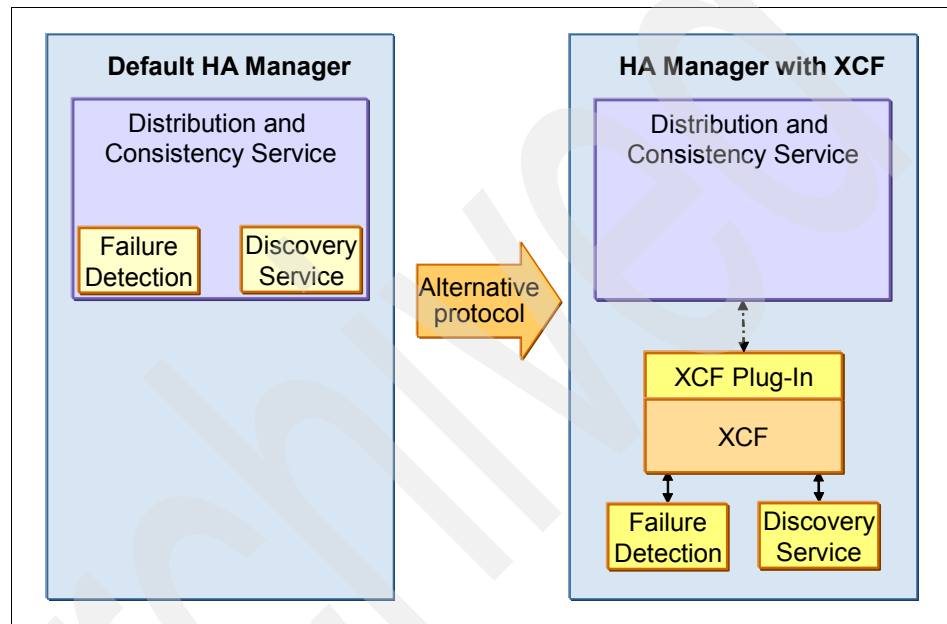


Figure 14-8 Architectural changes when using XCF support

As seen in Figure 14-8, the main changes are:

- ▶ The internal failure detection of the HA manager will be factored out of the DCS structure. Instead, XCF's failure detection will be used to notify the HA manager.
- ▶ The Discovery Service, used to communicate with the HA manager now talks to the XCF component of z/OS.
- ▶ XCF plugs into DCS to perform the alive check. This will disable the TCP/IP ping-based heartbeat.

If a member can no longer be contacted, the protocol will notify the core group members and issue message DCSV1032I in the SystemOut.log file.

Exploitation of XCF techniques

WebSphere Application Server for z/OS uses the XCF communication, which offers three different services:

- ▶ Group services
- ▶ Signaling services
- ▶ Status monitoring services

Only the group service is used for the XCF support for WebSphere HA manager. To use the XCF system service, WebSphere Application Server for z/OS V7.0 provides an XCF plug-in that contains XCF user routines, as shown in Figure 14-8 on page 442.

Become a XCF group member

When the XCF-based protocol is configured as the discovery and failure detection protocol, then for each WebSphere Application Server core group one XCF group will be created and the core group members will become active members of this XCF group.

To become an active member of a XCF group, during startup of a core group member (for example a node agent) the IXCJOIN macro is executed. As a result the member is associated with the address space in which the IXCJOIN was issued and added to the XCF group. As an active XCF member, it can perform the following tasks:

- ▶ Send and receive messages to other members
- ▶ Have its status monitored by XCF
- ▶ Be notified of status changes to other members of the group

Each WebSphere Application Server component that can become a core group member can become a XCF group member.

Status change

The XCF group service has three options of recognizing that a member is no longer active:

- ▶ The corresponding XCF macro is run that disassociates a member from the XCF group. This happens when the core group member is stopped through the Integrated Solutions Console.
- ▶ The address space (with the core group member in it) that is associated with the XCF member is terminated. This will result in the termination of the XCF member.
- ▶ The status user routine of the XCF member (and the core group member) is no longer working. To show that a member is still running, it needs to update a specific field. If the member fails to update this field within a specified time interval, XCF schedules the status user routine to determine if a problem exists.

Member notification

To notify the other core group members of a status change, an XCF group user routine is executed.

The group user routine enables XCF to notify active members of a group when there is a change in the operational state of any other member in the group, or a change to the status of any system in the sysplex.

This triggers the HA coordinator inside WebSphere Application Server for z/OS to perform necessary actions.

14.5.5 Activating XCF support for HA manager

To take advantage of the enhanced HA manager discovery and failure mechanism, the following requirements must be satisfied: (Otherwise you have to use the default policy.)

- ▶ The z/OS VTAM® component must be configured to start XCFINIT = YES, to enable TCP/IP to use the XCF service.
- ▶ All core group members must be on Version 7 of WebSphere Application Server
- ▶ All core group members must be running on z/OS
- ▶ Should the core group be bridged to another core group, then all bridged groups must be reside on z/OS in the same sysplex

When these prerequisites are met, perform the following steps to activate the XCF support in the Integration Solutions Console:

1. Select **Servers** → **Core Groups** → **Core Group settings**.
2. Select the core group you want to modify.
3. In the selected core group, select **Discovery and failure detection**.
4. Check the Use alternative protocol providers radio box and enter the fully qualified class name for the z/OS factory that is used to create the alternate protocol provider, `com.ibm.ws.xcf.groupservices.LivenessPluginZoSFactory`, as seen in Figure 14-9 on page 445.

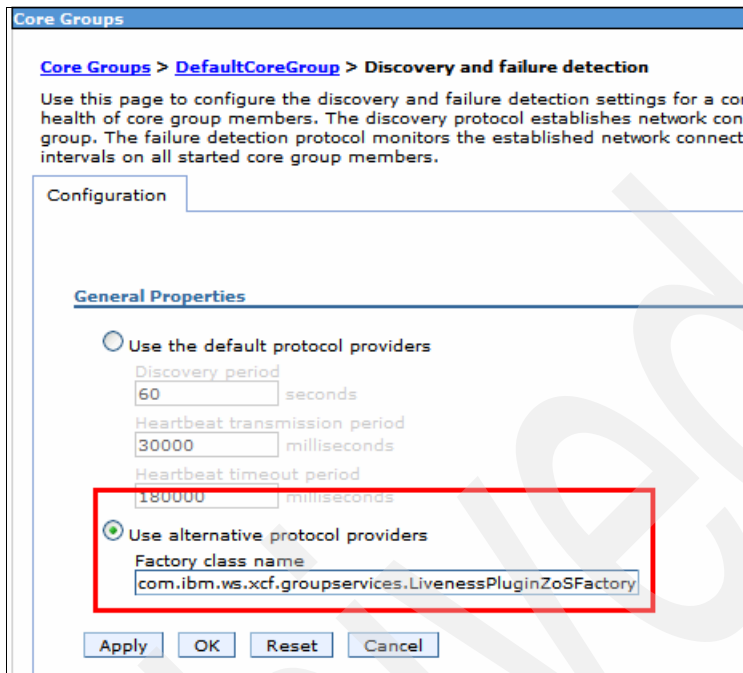


Figure 14-9 Activation setting for XCF support in the Integration Solutions Console

5. Click **OK** and save the changes to the master configuration.
6. Restart the server.
7. (Optional) If the default values for the failure detection protocol provided with the XCF support are not convenient for your environment, you can configure the following core group custom properties:

- IBM_CS_STACK_CHECK_INTERVAL_SECS

This property defines how often the alternate protocol provider checks the liveness of a core group member. The default value is 30 seconds

- IBM_CS_STACK_CHECK_FAILS

This property sets the number of attempts the alternate protocol provider will make to contact the core group member before notifying the high availability manager that a member is not active. The default value is 3.

Note: To determine the total time that it takes the alternate protocol provider to determine that a server has failed, multiply the values specified for the IBM_CS_STACK_CHECK_INTERVAL_SECS and IBM_CS_STACK_CHECK_FAILS custom properties.

Using the default values it would take 90 seconds.

14.6 z/OS Fast Response Cache Accelerator

This section gives general information about the Fast Response Cache Accelerator (FRCA).

14.6.1 Overview and benefits

WebSphere Application Server for z/OS V7.0 can be configured to use the Fast Response Cache Accelerator facility of the z/OS Communications Server TCP/IP. FRCA has been used for years inside the IBM HTTP Server to cache static contents like pictures or HTML files.

Attention: This functionality needs z/OS 1.9 or higher to be used. It is not planned to include this through PTFs in earlier versions of z/OS.

The z/OS Communications Server TCP/IP service updates to the FRCA support are required for this function to work on z/OS Version 1.9. If the updated FRCA services are not available on the system, the application server will issue error message BBOO0347E or BBOO0348E. TCP/IP uses CSM storage to maintain the cache.

The high speed cache can be used to cache static and dynamic contents, such as servlets and JavaServer Pages (JSP) files, instead of using the WebSphere Application Server Dynamic Cache.

Figure 14-10 on page 447 shows the changed flow of a request for a JSP that can be answered from the cache, assuming that the IBM HTTP server also resides on z/OS:

- ▶ Without FRCA exploitation a request has to be processed by TCP/IP, then by the IBM HTTP Server on z/OS until WebSphere Application Server itself can answer the request from its Dynacache.
- ▶ With FRCA exploitation a request to a cached JSP is recognized in the TCP/IP processing and gets answered directly.

The benefits of using the FRCA are a reduced response time and a reduced CPU cost for the serving of requests, compared to the Dynamic Cache. Tests have shown that a request served from the FRCA used approximately 8% of the processor time that the same request consumed in a Dynamic Cache environment. These advantages come from its structure, because the FRCA cache can directly serve incoming TCP/IP requests (Figure 14-10 on page 447).

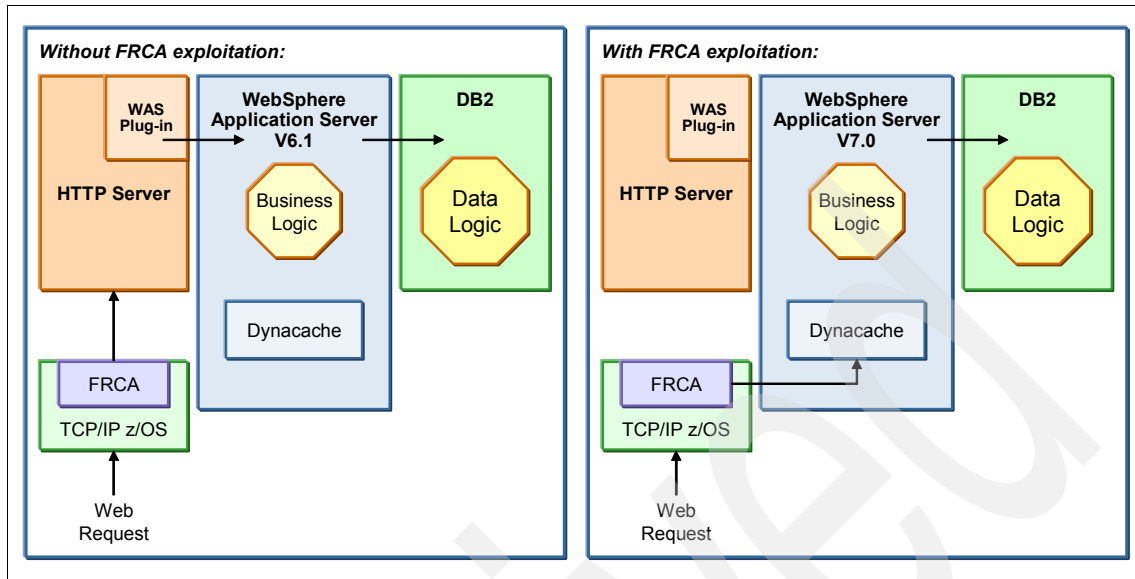


Figure 14-10 Overview of z/OS FRCA and WebSphere Application Server for z/OS

Note: At the time of writing, the FRCA cache only supports non-SSL connections.

14.6.2 Configuring FRCA

The FRCA support needs to be configured in the Integration Solutions Console as an external cache group and in `cachespec.xml`. This XML file should exist for each application in the corresponding WEB-INF directory of that application.

To configure FRCA, the following major steps need to be performed on an application server base (each step will be described in detail on the next pages). Enabling FRCA on a cell level is only possible through the use of self-written `wsadmin` scripts.

Perform the following steps to configure FRCA:

1. Create an external cache group.
2. Populate the cache group with your server.
3. Modify the cachespec.xml.
4. [Optional] configure logging for the cache.
5. [Optional] If objects larger than 10 MB are used, modify the protocol_http_large_data_response_buffer user variable to a size larger than the largest object to be cached.

Best-practice: It is suggested to configure the dynamic cache disk offload. This will prevent objects from being removed from the dynamic cache and being removed from the FRCA cache. Refer to the Information Center article *Configuring dynamic cache disk offload*, for further information, The article is found on the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/tdyn_diskoffload.html

Creating external cache group

Perform the following steps to create an external cache group:

1. Navigate to **Servers** → **Server Types** → **WebSphere application servers**.
2. Select the server that should benefit from the FRCA.
3. In the Configuration view, select **Container Settings** → **Dynamic cache service**.
4. Navigate to **Additional Properties** → **External cache group** and select **New**.
5. Enter a name that is convenient for your installation. Select **OK** and save changes to the master configuration.

Adding member to the cache group

Perform the following steps to add a member to the cache group:

1. Open the cache group just created by navigating to **Servers** → **Server Types** → **WebSphere application servers** → **(your_server)** → **Container Settings** → **Dynamic cache service** → **(your_cache_group)** → **External cache group members** and select **New**.
2. Select the configuration tab and perform the following steps:
 - a. Select **Advanced Fast Path Architecture**.
 - b. Select the Enable fast response cache adapter check box.
 - c. In the Port field, select **zero** as the port number.
3. For FRCA configuration, set the following fields:
 - Cache size
The cache size is a value that specifies the size of the FRCA cache. The maximum size is limited by the amount of available CSM memory managed by the z/OS Communications Server. The value is rounded up to a 4 K (4096) interval. The default is 102400000.
 - Cache entries
This value specifies the number of individual objects that can be placed in the FRCA cache. The maximum value is limited by the underlying support in the z/OS Communications Server. The default is 1000.
 - Max entry size
The max entry size value specifies the maximum size in bytes of a single object that can be placed in the FRCA cache. The default is 1,000,000.
 - Stack name
The stack name specifies the name of the Open Edition Physical File system supporting the TCP/IP stack containing the FRCA cache. The stack name specified must match the name on the SubFileSysType statement in the Open Edition BPXPRMxx parmlib member. This directive is only needed if the Open Edition Common Inet function is being used. Contact your system programmer to determine if Common Inet is in use, and if so, the name of the FRCA-enabled TCP/IP stack. The default is none.
 - Transaction Class
The transaction class name, which is eight characters or less, specifies the transaction class name that is used to classify the work done by FRCA. If the transaction class is specified, the FRCA processing is classified under WLM. If it is not specified, no classification will occur. The default is none.

Note: By default the FRCA cache is active on all channel chains that contain a Web container channel and do not contain an SSL Channel.

You can disable FRCA for specific channel chains and listener ports, using the configuration tab for transport channels. Navigate to **Servers** → **Server Types** → **WebSphere application servers** → **(your_server)** → **Web Container Settings** → **Web container transport chains** → **(your_transport_chain)**. Mark **Disable FRCA caching**.

Update cachespec.xml

When updating the XML file, remember that both names used for the cache in the XML file and the Integrated Solutions Console must match. Otherwise the cache cannot be used. A sample cachespec.xml is shown in Figure 14-11. For more information about the usage of a cachespec.xml, refer to the WebSphere Application Server V7.0 Information Center, at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rdyn_cachespec.html

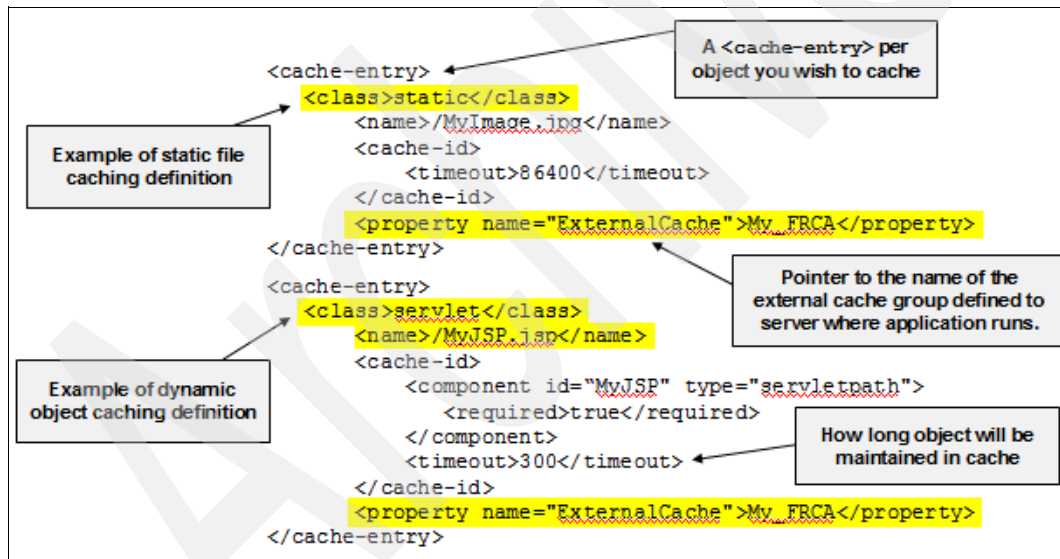


Figure 14-11 Sample cachespec.xml for usage with FRCA

[Optional] Enabling FRCA logging

If you want to enable logging for FRCA, use the Integrated Solutions Console and execute the following steps:

1. Navigate to **Servers** → **Server Types** → **WebSphere application servers** → **(your_server)**.

2. Under the Troubleshooting tab, select **NCSA access and HTTP error logging**.
3. Check the Enable logging service at server start-up check box.

If you want you can modify the other settings or keep the defaults.

Large object caching

If objects larger than 10 MB should be cached, you need to set the `protocol_http_large_data_response_buffer` custom property. The value for this property should be higher than the maximum size that should be cached.

For information about how to set custom properties, refer to the IBM Information Center at the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp>

14.6.3 Monitoring FRCA

You can use the WebSphere Application Server modify display console command to display statistics, and to list the contents of the FRCA cache. Refer to the Information Center article *Configuring high-speed caching using FRCA with the WebSphere Application Server on z/OS*, available at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/tdyn_httpserverz.html

To display cache statistics the following commands can be issued:

- ▶ From z/OS console: **f <serverName>,display, frca**
- ▶ From z/OS console: **display tcpip,,netstat,cach**
- ▶ From TSO: **netstat cach**

For monitoring large object caching activity in the Dynamic Cache, you can use the cache monitor. This installable Web application provides a real-time view of the dynamic cache state. In addition it is the only way of manipulating data inside the cache.

For more information about how to set up the cache monitor, go to the *Displaying cache information* article in the Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tdyn_servletmonitor.html

14.6.4 Resource Access Control Facility (RACF) integration

FRCA services can be restricted. If access is restricted (the SERVAUTH class and the FRCA resource are defined) in your environment, then WebSphere Application Server must be granted access.

If the access is restricted, the message seen in Example 14-2 will be issued.

Example 14-2 FRCA access denied message

```
BB00nnnnE FRCA INITIALIZATION FAILED. SERVER NOT AUTHORIZED TO USE FRCA
SERVICES. IOCTL RV=%d, RC=%d, RSN=%08X
```

The RACF command in Example 14-3 enables access to the FRCA services.

Example 14-3 RACF command to enable FRCA service

```
PERMIT EZB.FRCAACCESS.your_system_name.your_TCPIP_procname CLASS
(SERVAUTH) ID (your_control_region_userid) ACCESS (READ)
SETROPTS RACLIST (SERVAUTH) REFRESH
```

14.7 Thread Hang Recovery

This section describes the new Thread Hang Recovery option available on z/OS.

14.7.1 Overview

WebSphere Application Server for z/OS V7.0 contains a new technique called Thread Hang Recovery. A hung thread will end up with one of the following situations:

- ▶ It simply hangs around, only blocking threads and application environment resources, such as connections, tables, and so forth.
- ▶ It ends in a loop state, not only blocking other resources but in addition consuming CP or zAAP resources. What kind of processor is being used depends on whether a zAAP is available at all and in what step of the application the error occurs.

Thread Hang Recovery directly addresses both of these issues. It allows you to define actions that should be started if a timeout occurs.

It allows you to specify thresholds for processor usage and actions that should be performed if a single request exceeds this value. This is of real value if your

environment uses high timeout values, due to some long running transactions, but only with few processor resources per request. If such a transaction would suddenly consume a high amount of CPU, due to an error, this would not be detected by prior versions, unless the normal timeout occurs. However until the timeout occurs this will have a performance impact to the whole environment.

14.7.2 Pre-WebSphere Application Server V7.0 technique

In releases prior to V7.0, if a request ran into a timeout, the server assumes that the request must be hung and starts to solve the situation. Depending on the recovery setting for your installation the server has two choices of processing.

- ▶ Terminate the servant with ABEND EC3

If `protocol_http_timeout_output_recovery=SERVANT`, then the servant will be terminated and WLM will restart a new one. A dump of the servant may be generated and all work that was running in the servant is terminated. This option could end up penalizing work that was not having any problems. In addition, server throughput is affected while the a dump is being taken and a new servant is started which can take a long time

- ▶ Respond to the client and continue working

If `protocol_http_timeout_output_recovery=SESSION`, then it is assumed that there was an unusual event that caused the timeout and the request will eventually complete successfully. If this assumption is wrong, and the request is truly hung, the servant is left with one less thread for processing incoming work. In addition, by allowing the request to continue, deadlocks could occur because the request is holding locks or other resources. If this problem continues to happen on subsequent requests, multiple threads become tied up and the throughput of the servant is affected, possibly to the point where it has no threads available to process work.

14.7.3 WebSphere Application Server V7.0 technique

If a hung thread was detected, then the servant can try to interrupt the request. To allow the servant to do so, a new registry of *interruptible objects* is introduced. Certain blocking code can register so that if too much time passes, the servant can call the interruptible object in order for it to attempt to unblock the thread. A Java interruptible object will always be registered so the servant will try to have Java help interrupt the thread if all else fails.

Note: The code that is used to unblock a thread is provided by the WebSphere Application Server V7.0. To use the Thread Hang Recovery for your application serving environment, you do not have to implement code for the Interruptable Objects registry.

The results of this can be as follows:

- ▶ Thread can be freed

If this is the case, then the user whose request hung receives an exception. The administrator can define what dump-action should be taken (none, svcdump, javacore, or traceback).

- ▶ Thread cannot be freed

In the case that a thread cannot be freed, the system action depends on the administrator settings. The options are as follows:

- Abend the servant
- Keep the servant up and running
- Take a dump (defined by new variables)

See Table 14-2 on page 455.

Although the basic options if a thread cannot be freed are still the same as in prior versions of the WebSphere Application Server for z/OS product, the decision whether a servant should be abended or kept alive now depends on the following factors:

- ▶ How much CPU time is consumed by the thread? (Looping or just hanging?)
- ▶ Is the servant the last servant?
- ▶ How many threads are already in a hung state, within this servant?

For more details on the corresponding parameters, refer to 14.7.4, “New properties” on page 454.

If a thread that was reported to the controller as hung finishes, the controller is notified of that so that it is no longer considered in the threshold determination.

14.7.4 New properties

As described in the previous section, WebSphere Application Server V7.0 introduces a set of new variables that allow the administrator to configure the behavior of the application server if a hung thread cannot be freed.

The new properties will be listed by default in the system log, although they must first be created as Custom properties to benefit from them. To do so, log on to the Integrated Solutions Console and navigate to **Server** → **Server Types** → **WebSphere application servers** → (**your_server**) → **Server Infrastructure** → **Administration** → **Custom properties** → **New**. This will create the properties on a server base.

Note: When modifying one of the new parameters listed in Table 14-2 make sure to have these additional parameters configured (available in prior versions):

- ▶ control_region_timeout_delay
- ▶ control_region_timeout_save_last_servant

Table 14-2 Variables for hung thread related actions

Variable name	Values	Meaning
server_region_stalled_thread_threshold_percent	0-100	Percentage of threads that can become unresponsive before the controller terminates the servant. Default value 0 means that it behaves as in prior versions.
server_region_<type>_stalled_thread_dump_action	NONE SVCDUMP JAVACORE JAVATDUMP HEAPDUMP TRACEBACK	Specifies the dump action after hang is detected. <type> can be http(s), iiop, mdb, sip(s).
server_region_request_cputime_used_limit	variable	Amount of processor milliseconds a request can consume before servant will take an action.
server_region_cputimeused_dump_action	NONE SVCDUMP JAVACORE JAVATDUMP HEAPDUMP TRACEBACK	Type of documentation to take when a request has consumed too much processor time.
control_region_timeout_save_last_servant	0 1	Indicates whether the last available servant should be abended if a timeout situation occurs on that servant, or the last available servant should remain active until a new servant is initialized

Note: If the request exceeds the specified amount of time, UNIX Systems Services generates a signal that may or may not get delivered to the rogue request.

The signal may not get delivered immediately if the thread has invoked a native PC routine, for instance. In that case, the signal will not get delivered until the PC routine returns to the thread. When and if the signal gets delivered, a BBOO0327 message is output, documentation is gathered according to what is specified as documentation action in the `server_region_cputimeused_dump_action` property, and the controller is notified that a thread is hung.

After the signal is delivered on the dispatch thread, the WLM enclave that is associated with the dispatched request is quiesced. This situation lowers the dispatch priority of this request, and this request should now only get CPU resources when the system is experiencing a light work load.

14.7.5 Display command

There is a new command to display the dispatch threads that are currently active. The **DISPLAY, THREADS** command will display every dispatch thread in every servant region associated with the specified controller. By default, it will give you SUMMARY information but you can also specify that you want DETAILS. In the case of the `REQUEST=<value>` parameter, the default is DETAILS.

```
f <server_name>,display,threads,[ALL | TIMEDOUT | REQUEST=<value> |  
ASID=<value> | AGE=<value>]
```

The information is also available through a new `InterruptibleThreadInfrastructure` MBean.

14.8 Installing WebSphere Application Server for z/OS

This section provides an overview of the installation and configuration process for WebSphere Application Server for z/OS V7.0. For a detailed installation manual, including checklists for z/OS, see the Information Center article *How do I install an application serving environment*, available at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.installation.zseries.doc/info/zseries/ae/welc_howdoi_tins.html

14.8.1 Installation overview

To install a WebSphere Application Server for z/OS V7.0 on z/OS there are three major steps to be performed:

1. System preparation
2. Product code installation with System Modification Program/Extended (SMP/E)
3. Product configuration, including the profile creation, using Job Control Language (JCL)

14.8.2 Installation considerations

This section includes general considerations when installing a WebSphere Application Server for z/OS V7.0.

General environment considerations

Planing your environment is critical, so a dedicated chapter is introduced in this book. See Chapter 4, “Infrastructure” on page 93.

Naming convention

When installing WebSphere Application Server for z/OS V7.0, make sure you use a good naming convention. The operating system restriction to eight characters limits your naming convention to the usage of abbreviations. Keep in mind that there are new components introduced in WebSphere Application Server for z/OS V7.0 (for example the administrative agent and job manager). Make sure that your convention reflect this information as well.

Note: The naming convention guidelines from the Washington System center are included in the configuration tool for the profile creation.

Real memory defined

Keep in mind that the WebSphere Application Server for z/OS has a different blueprint than WebSphere Application Server for distributed environments. See 14.1.4, “Structure of an application server” on page 422 for more details. Because there are multiple heaps, one for every controller and servant region, this results in different memory requirements.

It is absolutely critical that the heaps defined in a WebSphere Application Server environment fit into real memory. The impact of not having the heap in the real memory would be a negative performance impact due to paging during each

garbage collection. The garbage collection for a JVM heap requires all pages of the heap to be in exclusive access and in real memory. If any pages are not in the real storage they first need to be paged in.

As an example look at the environment from Figure 14-3 on page 426. This includes a deployment manager and one application server (with three servants). Assume each control region heap has a maximum of 256 MB and each servant region a maximum of 512 MB defined. Then the real storage that the WebSphere Application Server for z/OS needs is 2.5 GB. This does not consider that there might be other middleware installed in the z/OS image.

Make sure that the LPAR that is used for the installation has enough real storage defined. Also keep in mind to add storage for the operating system and other applications running in this LPAR, like DB2, CICS, and so on.

Health check: We suggest to monitor your system and check for swapping. If you experience swapping, this will have a performance impact.

Heap sizes (min/max) defined

Usually the z/OS version will need smaller maximum heap sizes than the distributed version, because it has specialized heaps in its structure, as described in 14.1.4, “Structure of an application server” on page 422.

This is of interest when migrating an application from another platform to WebSphere Application Server for z/OS V7.0. Often, the memory size from the distributed environment is carried on from the distributed environment and reused for the controller and servant regions settings. While this may be a waste of memory resources, it can affect the performance as well. If the heap is sized too large, then the Garbage Collection will run less often, but if it runs, it takes up more time. This might reduce the general throughput.

Important: If an application is migrated to WebSphere Application Server for z/OS V7.0 from another operating system family, perform a verbose Garbage Collection analysis. This will allow you to size the heap to a good minimum and maximum value, so that the performance is good and no resources are wasted.

zAAP usage

The System z Application Assist Processor (zAAP) is a processor dedicated for the execution of Java and XML work. There are two main reasons why you should consider the usage of a zAAP in your WebSphere Application Server for z/OS environment:

- ▶ Reduced software cost

Workload that runs on the zAAP does not count to the monthly z/OS software bill. Because WebSphere Application Server is mainly written in Java, it can use the zAAP. In most environments we see about 80% of the WebSphere environment (WebSphere Application Server for z/OS and the applications inside) running on the zAAP. The use depends on the amount of Java Native Interface (JNI) calls and other non-Java functions used in the application.

- ▶ Performance gain

The zAAP implementation offers a dedicated Processor Resource/Systems Manager™ (PR/SM™) processor-pool. This means that units of work dispatched run in their own world. Because less units will compete for the processor resources, they do not have to wait as long until they can access the processor.

A zAAP has to be configured in the LPAR profile through the Hardware Management Console (HMC) of the System z. It can be used both as a shared or a dedicated processor, depending on the LPAR setting.

Note: If you currently do not have a zAAP physically installed, you can use the Resource Measurement Facility (RMF) to identify the amount of CPU seconds that could be run on a zAAP. For detailed information refer to the documentation available at the following Web page:

<http://www-03.ibm.com/systems/z/advantages/zaap/resources.html>

File system considerations

When installing a WebSphere Application Server for z/OS, the following things should be taken into consideration regarding the file system, regardless of whether HFS or zFS is used:

- ▶ Separate configuration file systems per node

Although file systems can be shared across multiple z/OS images in a parallel sysplex, from a performance perspective, it is suggested to create dedicated file systems for each node.

- ▶ Product file system mounted read only

This will improve performance as well as prevent the change of file system contents.

14.8.3 Function Modification Identifiers

WebSphere Application Server for z/OS consists of the Function Modification Identifiers (FMIDs) presented in Table 14-3.

Table 14-3 FMIDs for WebSphere Application Server V7.0 for z/OS

Component name	FMID	Compid	PSP Bucket (Upgrade=WASAS700) Subset
WebSphere Application Server V7.0 for z/OS	H28W700	5655I3500	H28W700
Optional Materials	JIWO700	---	JIWO700
Install Samples	JIWO700	5655I3509	JIWO700
IBM HTTP Server 64-bit plug-in	JIWO700	5655I3511	JIWO700
WebSphere Application Server V7.0 for z/OS DMZ Secure Proxy Server	JDYZ700	5655N0212	JDYZ700

Available as a separate product, IBM HTTP Server V7.0 (FMID HHAP700) is based on Apache. The older, 31-bit version based on Domino Go is included with z/OS and is called HTTP Server for z/OS.

14.8.4 SMP/E installation

The product code of V7.0 is still brought to your system using the SMP/E function of z/OS.

Contact the IBM Software Support Center for information about preventive service planning (PSP) upgrades for WebSphere Application Server for z/OS. For more information about PSP upgrades, see the WebSphere Application Server for z/OS: Program Directory. Although the Program Directory contains a list of required program temporary fixes (PTFs), the most current information is available from the IBM Software Support Center.

For more information about maintenance, see the Information Center article *Applying product maintenance* article, available at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.installation.zseries.doc/info/zseries/ae/tins_prodmainenance.html

14.8.5 Customization

After the SMP/E installation is complete, the next step is the product configuration as seen in Figure 14-12.

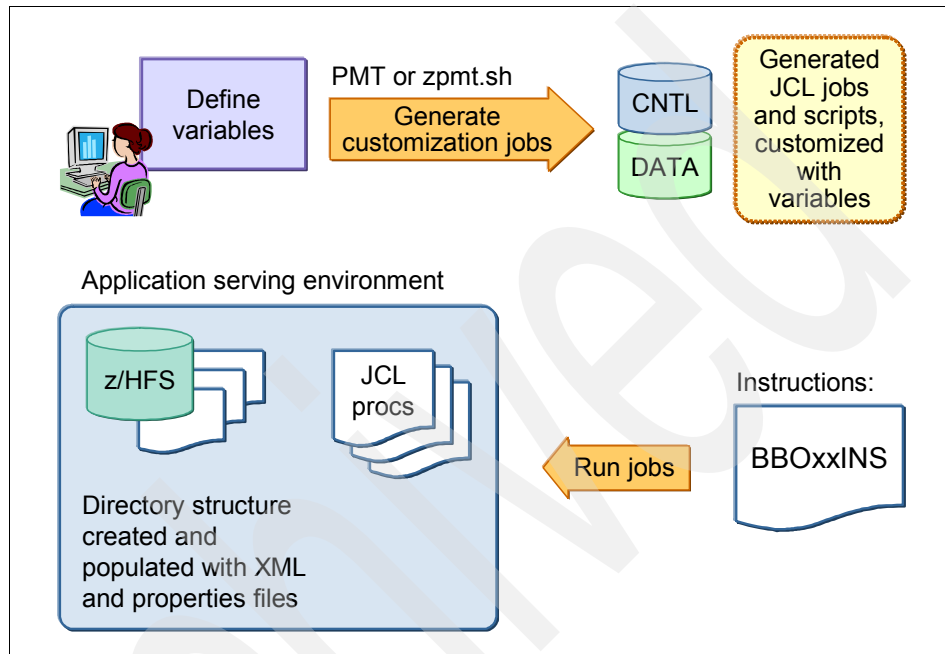


Figure 14-12 Configuration overview of WebSphere Application Server for z/OS

This includes the creation of the server profile and the execution on the host, using the following tools:

- ▶ (optional) Washington System Center Planning Spreadsheet for WebSphere Application Server V7.0 for z/OS
- ▶ WebSphere Customization Tools (graphical / interactive)
- ▶ zpmt.sh z/OS script (command line / batch style / silent)

These tools are described briefly in the following sections.

Note: For detailed information about the installation of WebSphere Application Server for z/OS V7.0 refer to the Information Center article *Installing your application serving environment*, available at the following Web page:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.installation.zseries.doc/info/zseries/ae/welc6topininstalling.html>

Washington System Center Planning Spreadsheet

The Washington System Center created a new version of the *WebSphere for z/OS Version 7 - Configuration Planning Spreadsheet* (Reference #PRS3341). The spreadsheet can be used during the customization process. It can be found at the following Web page:

<http://www-03.ibm.com/support/techdocs/atmastr.nsf/WebIndex/PRS3341>

This Techdoc contains three planning spreadsheets:

- ▶ Network Deployment Cell
- ▶ Standalone server
- ▶ DMZ (Secure Proxy)

You can use the spreadsheet to enter multiple variables to define your installation. The entered data can be saved and used as a response file for the graphical WebSphere Customization Tools or the command-line `zpmc.sh` tool. A response file can be used by these tools to generate the actual Job Control Language (JCL) that will create the profiles (as seen in Figure 14-13 on page 463).

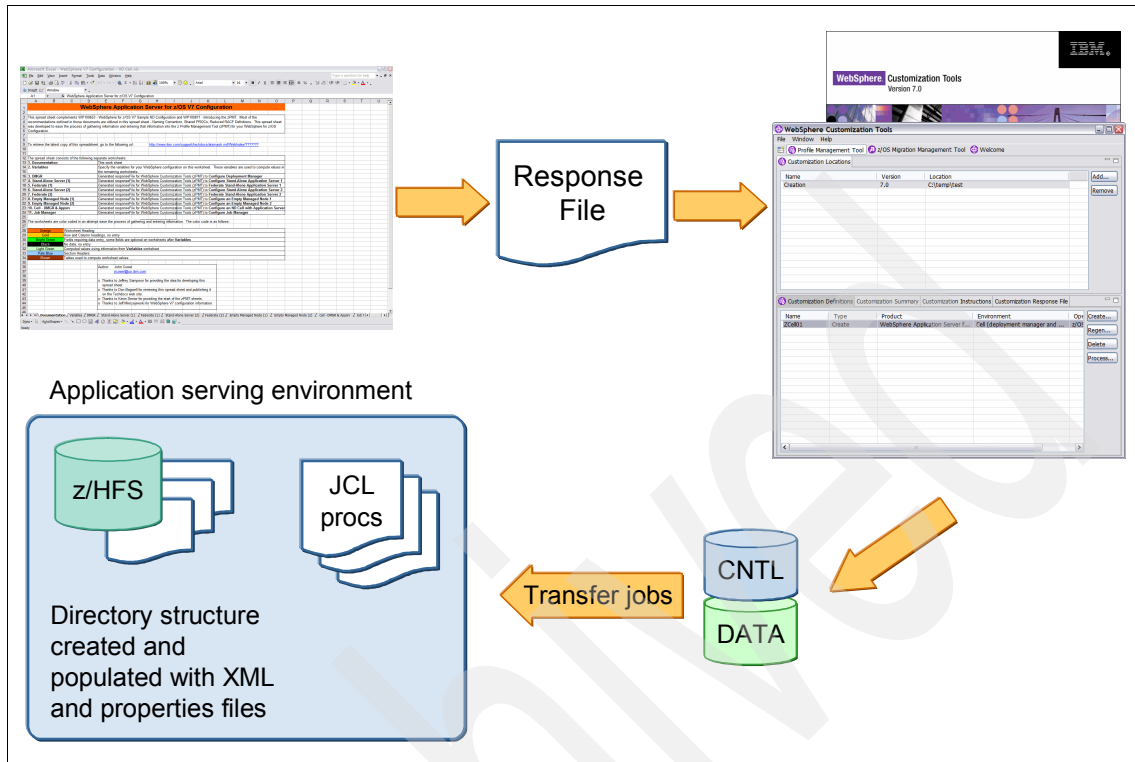


Figure 14-13 Usage of Planning Spreadsheet for WebSphere Application Server for z/OS

WebSphere Customization Tools

After the product code is brought to the system, WebSphere Application Server needs to be configured. In version 7 this is only possible using WebSphere customization tools. The tools can be accessed by searching for WebSphere Customization Tools (WCT) V7.0 on the WebSphere Fix site at the following Web page:

<http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg27004980#tools>

Note: The ISPF panel configuration is no longer available in WebSphere Application Server for z/OS V7.0.

There is a useful presentation that describes the usage of the WCT. Refer to the following Web page:

<http://www-03.ibm.com/support/techdocs/atmastr.nsf/WebIndex/PRS3357>

The set of tools is available for Windows and Linux based workstations are as follows:

- ▶ Profile Management Tool (z/OS only)
For creation of profiles
- ▶ z/OS Migration Management Tool
For migration of nodes

For detailed information about how to install and use these tools refer to the *Installation and migration* section of the IBM Education Assistant for WebSphere Application Server V7.0. The assistant can be found at the following Web page:

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.was_v7/was/7.0

Note: The Application Server Toolkit (AST) that was available in previous versions of the WebSphere Application Server is not available for V7.0.

zpmt.sh command

The zpmt.sh script is the silent implementation of the Profile Manage Tool on the z/OS host. You use a command line call to the script, including various parameters. It will then create the .CNTL and .DATA members necessary to build WebSphere Application Server corresponding to the response file. It can be configured to allocate and to copy the members from the z/OS file system to the z/OS data sets. The principal overview is shown in Figure 14-14.

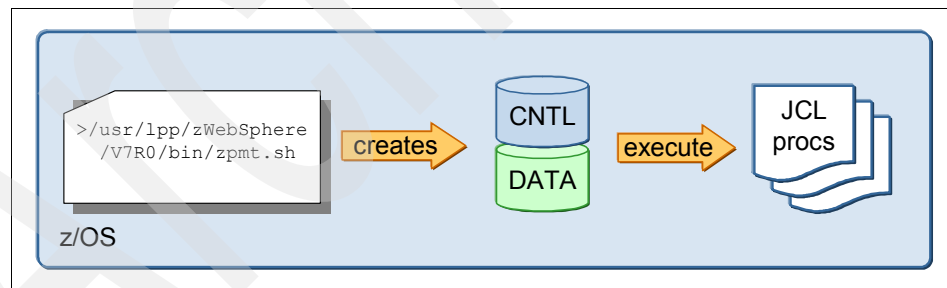


Figure 14-14 Overview of zpmt.sh configuration script

The script can be found in the default WebSphere Application Server for z/OS product directory `/usr/lpp/zWebSphere/V7R0/bin`. Executing the script places you in the OSGI command shell.

Note: An OSGI command shell is an execution environment that allows the remote management of Java application and components. It is based on the osgi open standard.

While it might first look as though nothing is happening, the shell will come up with status messages after some time.

Note: If you have to rerun a script for a profile, make sure to delete the profilePath directory. Otherwise you will receive the following message:
The profile path is not valid.

14.8.6 Execute customization jobs

The second step in the customization is the execution of the JCL that was created using one of the techniques described in 14.8.5, “Customization” on page 461.

Table 14-4 shows the jobs necessary for the customization of WebSphere Application Server for z/OS V7.0. The jobs are similar to V6.1, but there are some notable differences.

Table 14-4 Installation jobs for WebSphere Application Server for z/OS V7.0

Job name	Description
BBOxxINS	Instruction member that contains the installation steps.
BBOSBRAK	RACF scripts will be created and executed in this JCL. No more “BRAJ” to create script for “BRAK” to execute.
BBOSBRAM	Create /home directories for the WebSphere users in the OMVS part and set ownership.
BBOxBRAK	RACF scripts will be created and executed in this JCL. No more “BRAJ” to create script for “BRAK” to execute.
BBOxCFS	Set up file system (whether HFS or ZFS™).
BBOxCPY1	Copies the tailored start procedures to the cataloged procedure library.
BBOxHFSA	Populate the created HFS. The job will automatically create intermediate symlinks based on the options chosen in the WCT.
BBOWWPFx	No more HFSB job. Instead the file system initialization is included in WWPFD

14.9 System programmer considerations

This section contains some additional hints and tips for system programmers that should be taken into consideration when installing and configuring a WebSphere Application Server for z/OS V7.0 environment.

14.9.1 Systems Management Facility enhancements

WebSphere Application Server for z/OS is capable of writing z/OS Systems Management Facility (SMF) records for performance tuning and charge-back.

Prior to WebSphere Application Server for z/OS V7.0

Since Version 4 of WebSphere Application Server, SMF record 120 was used to log usage data.

SMF reserves record type 120 for WebSphere activity data. Record 120 includes a number of subtypes, each of which contains a subset of the data. This fragmented view of the data is due to internal divisions in the product. Some record 120 subtypes are created by the WebSphere Application Server runtime (subtypes 1 and 3), while others are created by the Web containers and EJB containers (subtypes 5, 6, 7, and 8). Because each subtype provides only a partial view of the activity, you need to correlate several subtypes to get a more complete picture of what the server is doing. This will however increase the overhead of SMF usage.

New in WebSphere Application Server for z/OS V7.0

In V7.0 a new subtype 9 record, called *Request Activity Record*, has been added. It can be used to create/write resource usage records without unreasonable overhead. Any data collection that adds substantially to the cost of acquiring that data is optional. You can activate or deactivate this record dynamically.

About the new subrecord: Because the new SMF 120-9 subrecord consumes less processing time than the collection of multiple subrecords, we suggest the usage of the new subrecord. The old subrecords remain valid and can be used with WebSphere Application Server for z/OS V7.0.

The subtype 9 gives you the option to monitor which requests are associated with which applications, how many requests occur, and how much resource each request uses. You can also use this record to identify the application involved and the CPU time that the request consumes. Because a new record is created for each request, it is possible to determine the number of requests that arrive at the system over a specific period of time.

Table 14-5 shows the new variables, possible values, and their meaning. A browser to display the contents of the new SMF records is provided.

Table 14-5 New SMF record 120-9 variables

Variable name	Values	Meaning
server_SMF_request_activity_enabled	Q 1	Turn record off/ on
server_SMF_request_activity_CPU_detail	Q 1	Processor usage details
server_SMF_request_activity_timestamps	Q 1	Formatted timestamps
server_SMF_request_activity_security	Q 1	Security information

Modify commands

The following modify commands can be used to activate or disable the new variables:

- ▶ F <server>,SMF,REQUEST,[ON | OFF]
- ▶ F <server>,SMF,REQUEST,CPU,[ON | OFF]
- ▶ F <server>,SMF,REQUEST,TIMESTAMPS,[ON | OFF]
- ▶ F <server>,SMF,REQUEST,SECURITY,[ON | OFF]

To show the current settings in your environment, as well as the number of records written since the last Initial Program Load (IPL), and the last non-zero Return Code (RC), use the following command:

```
F <server>,DISPLAY,SMF
```

For more information about the SMF enhancements refer to Whitepaper WP101342, *Understanding SMF Record Type 120, Subtype 9*, available at the following Web page:

<http://www-03.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP101342>

14.9.2 WebSphere Application Server settings

This section addresses the following:

- ▶ Intelligent runtime provisioning
- ▶ Workload profile setting
- ▶ Addressing mode

Intelligent runtime provisioning

The new intelligent runtime provisioning function is disabled by default. You might want to enable it in the Integrated Solutions Console to reduce the startup time and resource consumption. Improvements in the startup time of up to 10-15% can be seen. For more information about intelligent runtime provisioning see 3.1.14, “Intelligent runtime provisioning” on page 72.

Workload profile setting

WebSphere Application Server for z/OS V7.0 introduces a new value for the workload profile setting in the Object Request Broker (ORB) services advanced settings.

It is now possible to make a user defined selection for the number of threads, using the CUSTOM setting.

To change the value through the Integrated Solutions Console, click **Servers** → **Server Types** → **WebSphere application servers** → **your_server** → **Container services** → **ORB service** → **z/OS additional settings**.

Table 14-6 lists all possible values.

Table 14-6 Workload Profile settings for z/OS ORB service

Value	# Threads	Description
ISOLATE	1	Specifies that the servants are restricted to a single application thread. Use ISOLATE to ensure that concurrently dispatched applications do not run in the same servant. Two requests processed in the same servant could cause one request to corrupt another.
IOBOUND	MIN(30, MAX(5,(Number of CPUs*3)))	Specifies more threads in applications that perform I/O-intensive processing on the z/OS operating system. The calculation of the thread number is based on the number of CPUs. IOBOUND is used by most applications that have a balance of CPU intensive and remote operation calls. A gateway or protocol converter are two examples of applications that use the IOBOUND profile.
CPUBOUND	MAX((Number of CPUs-1),3)	Specifies that the application performs processor-intensive operations on the z/OS operating system, and therefore would not benefit from more threads than the number of CPUs. The calculation of the thread number is based on the number of CPUs. Use the CPUBOUND profile setting in CPU intensive applications, like XML parsing and XML document construction, where the vast majority of the application response time is spent using the CPU.

Value	# Threads	Description
LONGWAIT	40	Specifies more threads than IOBOUND for application processing. LONGWAIT spends most of its time waiting for network or remote operations to complete. Use this setting when the application makes frequent calls to another application system, like CICS screen scraper applications, but does not do much of its own processing.
CUSTOM	User defined	Specifies that the number of servant application threads is determined by the value that is specified for the <code>servant_region_custom_thread_count</code> server custom property. The minimum number of application threads that can be defined for this custom property is 1; the maximum number of application threads that can be specified is 100.

Addressing mode

The Addressing Mode (AMODE) is a JCL parameter, introduced with version 6.1, used in the START command to determine whether the server shall be started in 64-bit or 31-bit mode.

The AMODE parameter is still supported in V7.0. It is suggested not to modify the default value. In the generated procedures during the installation, the value 00 is default. This means that the value defined inside the application server's XML files is used for the decision of running 64 or 31-bit mode.

If you start the server with for example AMODE=64 and the XML files reflect a 31-bit installation (or through versa), then the server will not start.

Note: We suggest using the default value for the AMODE (AMODE=00) in the startup JCL for the WebSphere Application Server components. Double-check your automation settings.

14.9.3 Java settings

The settings described here are JVM settings that cannot be directly modified by the WebSphere Application Server V7.0. However, servers will use these underlying techniques.

For detailed information about the topics presented in this section, go to the Diagnostics Guide for the JVM V6, found at the following Web page:

<http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp>

Shared class cache

This section contains special information about the shared class cache usage on z/OS.

Overview

The shared class cache is used to share WebSphere Application Server and user classes between multiple JVMs.

JVMs that use the shared class cache start up more quickly, and have lower storage requirements than JVMs that do not. The overall cost of class loading is also reduced when JVMs use the shared class cache. When a new JVM that shares the class cache is initialized, it uses the preloaded classes instead of reading them from the file system. A JVM that shares the class cache still owns all the working data (objects and variables) for the applications that run in it. This helps to maintain the isolation between the Java applications being processed in the system.

The first JVM, after an IPL or after the cache has been destroyed, will take between 0–5% longer, to fill the cache. The startup time of subsequent JVMs will decrease by 10–40%, depending on the number of classes being loaded.

The z/OS implementation links pages in the private area of the address space that uses the cache to the frames of the original location of the cache. Because shared memory is used, BPXPRMxx parmlib settings affect the cache performance.

Important settings

Consider these factors when using shared class cache in your environment:

▶ Cache size limits

The maximum theoretical cache size is 2 GB. The size of cache you can specify is limited by the amount of physical memory and swap space available to the system. The cache for sharing classes is allocated using the System V IPC Shared memory mechanism. Because the virtual address space of a process is shared between the shared classes cache and the Java heap, if you increase the maximum size of the Java heap you might reduce the size of the shared classes cache you can create.

▶ BPXPRMxx settings for shared memory

The following settings affect the amount of shared memory pages available to the JVM.

- MAXSHAREPAGES
- IPCSHMSPAGES
- IPCSHMMPAGES
- IPCSHMMSEGS

Note: The JVM uses these memory pages for the shared classes cache. If you request large cache sizes, you might have to increase the amount of shared memory pages available.

The shared page size for a z/OS Unix System Service is fixed at 4 KB for 31-bit and 1 MB for 64-bit. Shared classes try to create a 16 MB cache by default on both 31- and 64-bit platforms. Therefore, set IPCSHMMPAGES greater than 4096 on a 31-bit system.

If you set a cache size using -Xscmx, the VM will round up the value to the nearest megabyte. You must take this into account when setting IPCSHMMPAGES on your system.

For further information about performance implications and use of these parameters, refer to IBM publications *z/OS MVS Initialization and Tuning Reference*, SA22-7592, and *z/OS Unix System Services Planning Guide*, GA22-7800.

Persistence for shared class cache

WebSphere Application Server for z/OS V7.0 uses the IBM Java Standard Edition (SE) 6. This JVM implementation offers the shared class cache that allows multiple JVM's to access the same classes, both application and system classes, without loading them multiple times into the memory.

While the IBM implementation for distributed platforms (Windows, Linux, AIX) offers the option to write the content to a file system so that it can survive an operating system restart, z/OS only supports the non-persistent cache.

Compressed references

The use of compressed references improves the performance of many applications because objects are smaller, resulting in less frequent garbage collection and improved memory cache use. Certain applications might not benefit from compressed references. Test the performance of your application with and without the option to determine if it is appropriate.

When using compressed references, the following structures are allocated in the lower area of the address space:

- ▶ Classes
- ▶ Threads
- ▶ Monitors

Because this is a JVM technique that is separated from the WebSphere Application Server product, you can only activate it by using a JVM argument on a JVM level. That means that on the z/OS platform the activation needs to be performed for all components of an application server that have a heap (adjunct, control and servant region).

In the Integrated Solutions Console navigate to **Server** → **Server Types** → **WebSphere application servers** → **(your_server)** → **Server Infrastructure** → **Java and Process Management** → **Process definition** → **server_component (Adjunct | Control | Servant)** → **Java Virtual Machine**.

Then add the following statement to the generic JVM arguments:

```
-Xcompressedrefs
```

As always when changing some JVM settings, you have to restart the server after saving and synchronizing the modifications to activate them.

14.9.4 Basic WLM classifications

The usage of WLM classification for the control and servant region address spaces is a basic z/OS approach. It is part of the installation process of the WebSphere Application Server for z/OS V7.0.

The following recommendations apply:

- ▶ Control regions should be assigned a service class with high priority in the system. Control regions should be assigned a service class with a high priority in the system such as the SYSSTC service class. A high priority is needed because controllers do some of the processing that is required to receive work into the system, manage the HTTP transport handler, classify the work, and do other housekeeping tasks.
- ▶ The servant classification should not be higher in the service class hierarchy than more important work, such as the controller and CICS, or IMS transaction servers. We suggest using a high velocity goal.
- ▶ Enclaves for WebSphere Application Server for z/OS are classified using the Subsystem CB. The performance goals set here depend on your application and the environment. Therefore, no quantitative recommendation can be made here. However, usually a percentile response time goal is advisable.
- ▶ OMVS components of WebSphere Application Server for z/OS needs to be classified as well. Some OMVS scripts are executed during server startup, therefore if these are not classified in the WLM, the server startup time will be increased.

Health check: WLM classification for OMVS components.

A step in the control region start-up procedure, invokes the applyPTF.sh script, using BPXBATCH. Because the BPXBATCH program is classified according the OMVS rules, on a busy system several minutes might pass before this step is completed.

You can minimize the impact of the BPXBATCH step by changing the WLM Workload Classification Rules for OMVS work to a higher service objective (Example 14-4).

Example 14-4 Service class definition for OMVS components

```
Subsystem Type . . : OMVS      Fold qualifier names?  Y (Y or N)
Description . . . OMVS subsystem rules mod for WAS
```

Action codes: A=After C=Copy M=Move I=Insert rule
 B=Before D=Delete row R=Repeat IS=Insert Sub-rule
 More ==>

-----Qualifier-----				-----Class-----	
Action	Type	Name	Start	Service	Report
_____	1	TN	FTPSEVE	_____	OMVSREP
_____	1	UI	OMVSKERN	_____	FTPSEVE
_____	1	TN	WSSRV*	_____	WAS70

Information about how to set WLM service class classifications, can be found in IBM Redbooks Publication *System Programmer's Guide to: Workload Manager*, SG24-6472.

Refer to the Information Center article *Controller and Servant WLM classifications*, found at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0//index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rweb_classservers.html

14.9.5 Address Space ID reuse

Address Space ID (ASID) reuse is an operating system function, introduced with z/OS 1.9. It allows the reuse of Address Space ID's, including those that are associated with cross-process services (like TCP/IP), which could not be reused in earlier releases of z/OS. WebSphere Application Server for z/OS, starting with V6.1 is able to use this function, thereby allowing the reuse of the ASID for terminated control regions.

The REUSASID parameter is automatically set to YES for any new servers that are created in WebSphere Application Server for z/OS V7.0.

If the operating system runs with the ASID reuse option disabled, the updateZOSStartArgs script, provided in the profile_root/bin of each profile can be run to enable the ASID capability for a specific WebSphere Application Server for z/OS profile. Because the script is run on a profile base, it must be run multiple times when multiple profiles or a Network Deployment installation is used. Ask the System Programmer whether the ASID reuse option is used in your installation.

For more information about ASID support in WebSphere Application Server for z/OS V7.0 refer to the information Center article *Enabling or Disabling the reusable ASID function*, available at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/xml_configasid.html

14.9.6 Deprecated features WebSphere Application Server for z/OS

As with every new version, some features are deprecated. For a complete list of deprecated features, visit the Information Center article *Deprecated features*, available at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rmig_depfeat.html

14.9.7 Java Command Language (JACL) stabilized

The JACL scripting language has been stabilized. This means that although no new development will be done for this language, it will coexist with Jython in WebSphere Application Server V7.0. Administrative scripts that use JACL do not necessarily need to be migrated to Jython.

However this stabilized status might change in future versions of WebSphere Application Server.

14.9.8 Application profiling

Application profiling allows you to analyze the application during runtime. It provides detailed information about what application step uses what amount of CPU in a graphical way. This allows you to identify critical points inside the application. While it is intended for developers, we suggest system programmers encourage the development team to use such a profiling technique. This will open up the black box application on the host.

A profile tool for z/OS is JinsightLive for z/OS. The tool is capable of analyzing 31-bit and 64-bit JVM. It can be downloaded from the following Web page:

<http://www.alphaworks.ibm.com/tech/jinsightlive>

Another tool that provides application profiling, would be the Eclipse Test and Performance Tools Platform (TPTP), a project from the Eclipse platform, found at the following Web page:

<http://www.eclipse.org/tptp/>

Note: Application profiling usually requires some level of experience with the tooling. So do not get confused when you start with the first steps. After you get used to the technique it is a powerful way of identifying CPU intensive points in an application. A lot of the critical points require only few changes in the application itself.

As a starting point you might ask your local IBM representative for some assistance.

14.10 Planning checklist

This section provides a short overview of what things to consider when you plan to install the WebSphere Application Server V7.0 for z/OS.

14.10.1 Planning considerations

Table 14-7 provides a list of planning considerations for WebSphere Application Server for z/OS V7.0.

Table 14-7 *Planning considerations for WebSphere Application Server for z/OS V7.0*

Planning item
ISPF Customization Dialog has been removed. All profile creation has to be done using the WebSphere Configuration Tools (WCT) or the line-mode <code>zpmc.sh</code> command.
Make sure you have a convenient naming convention, that can reflect the usage of the new WebSphere Application Server V7.0 components, job manager and administrative agent. (The zPMT will use the recommendations from the Washington System Center by default.)
Test the usage of the XCF support for the HA manager.
Make sure that monitoring is in place.
You might want to use the IBM Support Assistant with the following plug-ins: <ul style="list-style-type: none">▶ Visual Configuration Explorer (VCE) A graphical view on your environment as well as to keep track of configuration changes. The usage of this no-charge tool is recommended.▶ Garbage Collection and Memory Visualizer For analyzing verbose gc information, to identify a good heap size.▶ Thread Analyzer Provides analysis for Java thread dumps (or Javacores) such as those from WebSphere Application Server.
Check the amount of real memory provided for the LPAR, were WebSphere Application Server for z/OS should be installed.
Check the usage of Java Compressed references, because most of the current applications have no need for heaps larger than 900MB.
Check with the application developers, whether the application can exploit the shared class cache.
Make sure you performed a verbose garbage collection analysis, to identify and verify the heap size.

14.10.2 Resources

This section includes links and references to additional material intended to get a deeper insight on the workings of z/OS in combination with the WebSphere Application Server for z/OS.

For information about planning and system considerations required to build a heterogeneous cell, see *WebSphere for z/OS -- Heterogeneous Cells*. Note that this paper focuses on WebSphere Application Server V6.1, but the basic thoughts are still valid for V7.0. This paper is available from the following Web page:

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

For more information about the WebSphere Configuration Tools, including the zPMT and the zMMT, you might find the following Web pages useful:

- ▶ IBM WebSphere Customization Tools V7.0 for Windows
<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg24020368>
- ▶ WebSphere Application Server for z/OS V7.0 - Introducing the WCT for z/OS Document ID: PRS3357
<http://www-03.ibm.com/support/techdocs/atmastr.nsf/WebIndex/PRS3357>
- ▶ WebSphere for z/OS Version 7 - Configuration Planning Spreadsheet Document ID: PRS3341
<http://www-03.ibm.com/support/techdocs/atmastr.nsf/WebIndex/PRS3341>

To get a deeper insight on the Java options and functions used by WebSphere Application Server V7.0, the following Web pages might be useful:

- ▶ IBM Java 6.0 Diagnostics Guide
<http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp>
- ▶ Java technology, IBM style: Garbage collection policies, Part 1
<http://www.ibm.com/developerworks/java/library/j-ibmjava2/index.html>
- ▶ Java technology, IBM style: Garbage collection policies, Part 2
<http://www.ibm.com/developerworks/java/library/j-ibmjava3/>

For a comprehensive insight on application development of Java applications refer to the following Redbooks publications:

- ▶ *Java Stand-alone Applications on z/OS, Volume I*, SG24-7177
- ▶ *Java Stand-alone Applications on z/OS Volume II*, SG24-7291

Various tools are available to ease the daily life of developers and system programmers. The one listed here are free of charge:

▶ IBM Support Assistant.

This tool, together with some plug-ins provide a straight forward way of checking for configuration changes, a central repository for configuration values. In addition, you can use it to create graphical overviews of your environment. It can be found at the following Web page:

<http://www-01.ibm.com/software/awdtools/isa/support/>

▶ JinsightLive for IBM System z

The application profiling tool can be found at alpha works, at the following Web page:

<http://www.alphaworks.ibm.com/tech/jinsightlive>

▶ Eclipse Test & Performance Tools Platform (TPTP), a profiling tool plug.in for the well-known eclipse project can be found at the following Web page:

<http://www.eclipse.org/tptp/>

Migration

This chapter discusses migration considerations for moving to WebSphere Application Server V7.0. It contains the following sections:

- ▶ “Infrastructure migration considerations” on page 480
- ▶ “Application development migration considerations” on page 486
- ▶ “System management migration considerations” on page 487
- ▶ “Messaging migration considerations” on page 488
- ▶ “Web services migration considerations” on page 488
- ▶ “Security migration considerations” on page 489
- ▶ “WebSphere Application Server for z/OS migration considerations” on page 490

15.1 Infrastructure migration considerations

When migrating from an existing environment to the new version of WebSphere Application Server V7.0 you must create a migration plan. This plan might cover the following core steps:

1. Project assessment
2. Project planning
3. Skill development
4. Setup of development environment
5. Application migration
6. Runtime migration of additional environments
7. Testing and benchmarking
8. Runtime migration of the production environment
9. Lessons learned session

The steps that you might take in performing a migration are shown in Figure 15-1 on page 481.

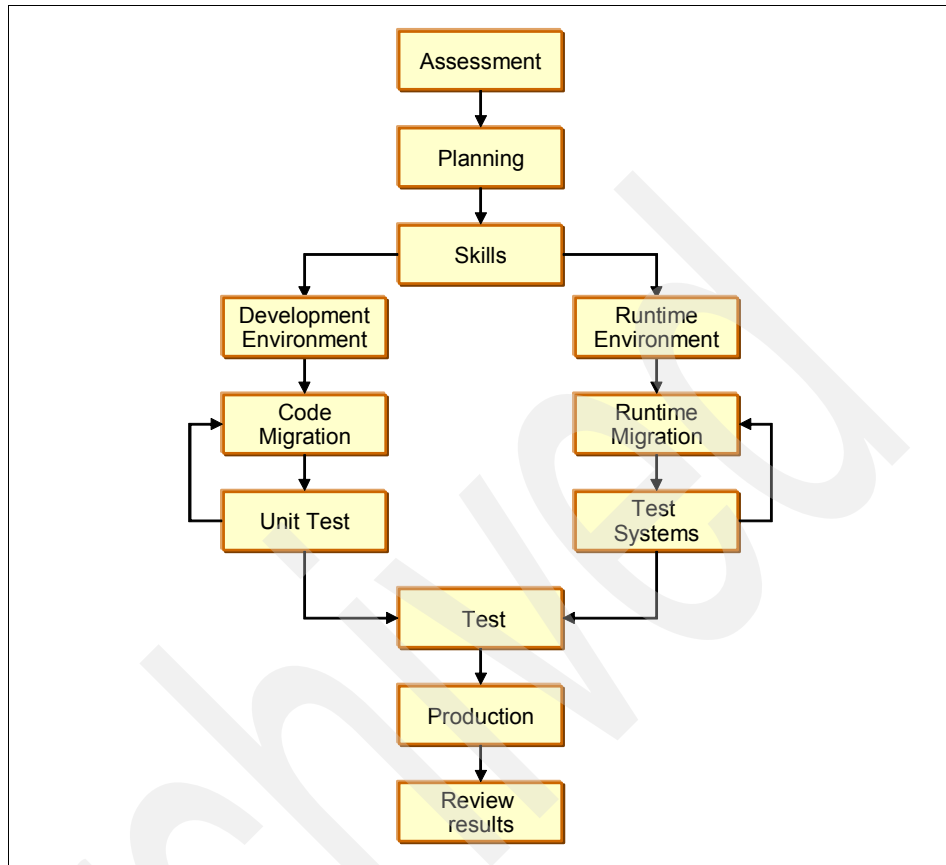


Figure 15-1 Migration path

Note: The IBM Migration Knowledge collection provides more details and updated information about WebSphere Application Server migration. It can be found at the following Web page:

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27008724>

15.1.1 Scripting migration

Since WebSphere Application Server V5.1, two scripting languages for WebSphere Application Server are available namely JACL and Jython.

With the announcement of WebSphere Application Server V7.0, JACL is declared *stabilized*. This means it will not be removed but there will be no further development in it. It is recommended to create new scripts using the Jython scripting language.

Note: The Information Center article *Deprecated, stabilized, and removed features* provides a detailed explanation of the terms and the product features affected by deprecation and stabilization. It is available at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rmig_deprecationlist.html

15.1.2 HTTP server plug-in support

The HTTP server plug-in shipped with WebSphere Application Server V7.0 can work with WebSphere Application Server V5.1, V6.0, V6.1, and V7.0.

Note: The URI in the routing rules section of the plugin-cfg.xml must be unique for each machine.

15.1.3 Coexisting versions on one system

WebSphere Application Server V7.0 can coexist with older versions of WebSphere Application Server on the same system. You can install and run these different versions of WebSphere Application Server at the same time.

Consider the following factors before starting:

- ▶ The hardware and software of the system must be supported by all versions of WebSphere Application Server you plan to coexist
- ▶ Each installation of WebSphere Application Server requires additional system resources.
- ▶ Plan for unique ports for every installed version of WebSphere Application Server.
- ▶ WebSphere Application Server V7.0 supports coexistence with WebSphere Application Server V5.1, V6.0, V6.1, and V7.0.

Note: There exist some limitations and restrictions for mixed cells. The Information Center provides more details about coexistence of multiple versions of WebSphere Application Server. See the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.migration.nd.doc/info/ae/ae/tins_coexistep.html

15.1.4 Runtime inter operability

WebSphere Application Server is interoperable and can communicate with multiple older versions of WebSphere Application Server. This means that WebSphere Application Server V7.0 supports and can inter operate with applications that use the following features:

- ▶ Security
- ▶ Transactions
- ▶ EJB workload management

This support is offered for WebSphere Application Server V5.1, V6.0, V6.1, and V7.0, allowing incremental migration of nodes in a runtime environment.

15.1.5 Runtime migration tools

WebSphere Application Server V7.0 provides various runtime migration tools depending on the platform on which WebSphere is installed. These tools allow you to copy existing configuration information from existing WebSphere Application Server installations to the new version.

The following tool support is available:

- ▶ Distributed
 - Migration Wizard
 - Migration Commands

- ▶ System i
 - Migration Commands

- ▶ zOS

System z Migration Management Tool (See 15.7.4, “z/OS Migration Management Tool” on page 491)

15.1.6 Mixed cell support

To ease the incremental upgrade of your environment, the support of mixed cells can be used. Mixed cell support means that nodes on different versions of WebSphere Application Server and on different platforms are supported in the same cell managed by WebSphere Application Server Network Deployment V7.0. Although it is a supported configuration to run WebSphere Application Server in mixed cells, this is usually considered to be transitional.

Note: There are some limitations and restrictions for mixed cells. The Information Center provides more details about coexistence of multiple versions of WebSphere Application Server. Refer to the following Web page for details:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.migration.nd.doc/info/ae/ae/tins_coexistep.html

15.1.7 Network Deployment migration strategies

There are three strategies for migrating a Network Deployment environment:

- ▶ Manual migration
- ▶ Automated migration with whole node upgrade
- ▶ Automated migration with mixed version use

Manual migration

To run a manual migration means that a totally new environment is configured and the existing configuration is not considered. This is a viable option but might be error-prone unless scripts are used comprehensively.

This approach provides the following advantages and disadvantages:

- ▶ Advantages
 - Assuming that the existing scripts are correct, you have the least risk. All the migration work can be performed independently from the running production environment.
 - The granularity of the migration is under control of the project team. The migration team can easily control which parts of the environment are migrated at what time.
 - All the scripts are yours. You have full control over the migration and do not need to depend on WebSphere tools.
 - You can easily step back to the previous version if problems arise in the new environment.

Note: It is always a good practice to have comprehensive scripts to configure the environment available as it eases various administration tasks (such as setting up a new environment for testing purposes, recreating the environment in case of a disaster, and so forth).

- ▶ Disadvantages
 - Creation and continuous maintenance of these scripts can require a lot of effort and therefore be expensive.
 - Requires every change in the environment to be scripted.
 - Administration tasks performed through the Integrated Solutions Console in the existing environment are not migrated,

Automated migration with whole node upgrade

This migration approach relies on the runtime migration tools provided by WebSphere Application Server Network Deployment. As when migrating a whole node, the existing WebSphere V5 / V6 configuration is initially recreated in WebSphere Application Server V7.0.". Subsequently, one node after the other can be migrated whereby you have to consider that all applications on a node are migrated at the same time.

This approach provides the following advantages and disadvantages:

- ▶ Advantages
 - There is no need for self-written scripts. All migration is done using WebSphere Application Server tools.
 - All the information of the current configuration is moved to WebSphere Application Server V7.0. Therefore you do not have to worry about missing something.
- ▶ Disadvantages
 - All the information of the current configuration is moved to WebSphere Application Server V7.0. The disadvantage of this approach is that you will also move configuration items that you no longer need.
 - All applications on the node being migrated must be ready for migration at the same time. Therefore, the time of migration is triggered by application availability.
 - Manual enablement of some upgraded features is still required.
 - This approach is only viable if you are not redesigning your environment

Automated migration with mixed version use

This migration approach also relies on the runtime migration tools provided by WebSphere Application Server Network Deployment. Like when using the “Automated migration with whole node upgrade” on page 485 in the first step the existing WebSphere V5 / V6 configuration is recreated exactly in WebSphere Applications Server V7.0. After that, additional new WebSphere Application Server V7.0 nodes are added and applications can be migrated one by one when they are ready. As soon as all applications of an old node are migrated, the old node can be removed from the configuration.

This approach provides the following advantages and disadvantages:

- ▶ Advantages
 - There is no need for a self-written comprehensive set of scripts. All migration is performed using WebSphere Application Server tools.
 - All the information in the current configuration is moved to WebSphere Application Server V7.0. Therefore, you do not have to worry about missing something.
 - Allows more flexibility for application migration as migrations can be migrated when they are ready.
- ▶ Disadvantages
 - All the information of the current configuration is moved to WebSphere Application Server V7.0. The disadvantage of this approach is that you will also move configuration items that you no longer need.
 - Care must be taken if you want to re-factor your topology during the migration phase.
 - Manual enablement of some upgraded features is still required.

15.2 Application development migration considerations

This section provides a general overview of what needs to be considered when migrating applications between WebSphere Application Server versions.

Consider the following points:

- ▶ Understand the new features and use the right development and deployment tools. WebSphere Application Server V7.0 provides two tools:
 - WebSphere Application Server Deploy and Assembly V7.5
 - Rational Application Developer for WebSphere Software V7.5
- ▶ Understand JDK V1.6 impacts, such as whether it will affect any of the API commands used in the application.
- ▶ Understand the changes to the administrative deployment scripts with the introduction of new components (such as Business Level Applications).
- ▶ Identify the deprecated APIs in WebSphere Application Server V7.0 and determine whether any of these APIs are used in your existing applications.

For more information about deprecated APIs, see the Information Center article *Deprecated, stabilized, and removed features*, available at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/rmig_deprecationlist.html

15.3 System management migration considerations

This section highlights migration considerations from a system management perspective.

- ▶ Understand system management constraints and limitations while performing the migration. In the process of performing a migration, one of the possible scenarios is that there could be a single cell with mixed WebSphere Application Server versions. If so, will session replication work with different versions of WebSphere Application Server?
- ▶ Understand the new features provided by the administrative agent and the job manager through the Integrated Solutions Console. Understand the new administrative commands to manage the administrative agent and job manager.
- ▶ Understand the new administrative functions introduced in the Integrated Solution Console and wsadmin so that you can use them to administer new WebSphere Application Server objects such as Business Level Applications.
- ▶ Understand the changes in the profile creation wizard and what ports are allocated to the administrative agent and job manager. Port information is crucial especially when you have firewalls in place for your WebSphere Application Server systems.

15.4 Messaging migration considerations

This section highlights migration considerations from an messaging perspective.

- ▶ Understand whether your Java 2 Enterprise Edition (J2EE) V1.4 applications will run in WebSphere Application Server V7.0 without modifications to Java Messaging Service (JMS) code.
- ▶ If the JMS provider is WebSphere MQ, ensure you have the right version of WebSphere MQ with WebSphere Application Server V7.0.
- ▶ Use the WASPostUpgrade utility to migrate V5 jmsserver to a V7 default messaging configuration.

For more information about messaging migrations, see the Information Center article *Messaging resources*, at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/welc6tech_msg.html

15.5 Web services migration considerations

If you are migrating from WebSphere Application Server V6.1, be aware of the following issues:

- ▶ While JAX-RPC is still supported, some enhancements are (and will be) based only on JAX-WS and JAXB. For example, SOAP 1.2 and MTOM support is only available within these new programming models. Existing JAX-RPC applications that need to use the new features will need to be rewritten. Note that no automatic migration from JAX-RPC to JAX-WS has been provided.
- ▶ WebSphere Application Server V7.0 supports the new industry standard for SOAP over JMS for JAX-WS. The IBM proprietary nonstandard implementation has been deprecated and may be removed in a future release.
- ▶ Support for the '2006/02' WS-Addressing WSDL binding namespace has been deprecated. Replace any uses of the '2006/02' namespace in WSDL files with uses of the '2006/05' namespace.
- ▶ The WSDM interface has been deprecated and the other standard management interfaces in WebSphere Application Server should be used instead.

- ▶ Users migrating from WebSphere Application Server V6.1 with the Web Services Feature Pack need to be aware that annotations are no longer supported by default in pre-Java EE 5 applications. Annotation support has to be added manually.
- ▶ Migration of the JAX-WS bindings in the WebSphere Application Server 6.1 Feature Pack for Web Services takes place during the product migration to version 7.0. The product migration handles most of the WS-Security migration process, but your input and action is required for specific configurations in order to complete the migration.

15.6 Security migration considerations

There are a few security considerations to take into account when planning a migration from WebSphere Application Server V6.1 to V7.0:

- ▶ SPNEGO Trust Association Interceptor (SPNEGO TAI), introduced in V6.1, has been deprecated in V7.0. SPNEGO Web authentication has taken its place and it should be used instead.
- ▶ SAS security protocol has been deprecated and is only supported between V6.0 and previous version servers that have been federated in a V7.0 cell. We recommend CSiv2 protocol to be adopted during the migration process.
- ▶ Multiple Security Domains add a new dimension to your security environment. Although you are not required to take advantage of it, we recommend its use for separating administrative security from application security.

Note: In previous releases of WebSphere Application Server, you could associate a small set of security attributes at a server level that were used by all of the applications of that server. This way of configuring the security attributes is deprecated in WebSphere Application Server V7.0 and will be removed in a future release.

15.7 WebSphere Application Server for z/OS migration considerations

This section concentrates on the topics that need to be considered when migrating an existing WebSphere Application Server for z/OS to V7.0.

15.7.1 Migration and coexistence

There are some coexistence and prerequisite conditions that must be met before attempting a migration. The lowest release level of WebSphere Application Server that can be directly migrated to WebSphere Application Server V7.0 is V5.1. Prior releases must be migrated using a two-step migration, first to a version that is supported for the migration tools.

See Table 15-1 for minimum requirements for the supported releases.

Table 15-1 WebSphere Application Server for z/OS releases for direct migration

Current release	Target release	Minimum level
V5.1	V7.0	n/a
V6.0	V7.0	V6.0.2.12, if security is enabled
V6.1	V7.0	V6.1.0

Keep in mind that the deployment manager must always be at the highest version level. For example, when migrating to V7.0, the deployment manager must be at V7.0. With mixed versions in a cell, you can minimize application downtime during migration because you can migrate one node at a time. If you have applications that run in a clustered environment, those applications can typically continue to run while the migration of one node takes place.

15.7.2 General considerations

Before describing the migration process in more detail, here are some things to remember when performing a migration:

- ▶ Use the same procedure names.
 - Before updating the STC procedures for Version 7, you should save your current procedures in case you need to fallback to the previous level.
 - If you choose to use different procedure names, you will need to update the RACF STARTED class profiles. Sample RACF commands to accomplish this are found in the migration instructions provided.

- ▶ Automation changes may also be required when changing procedure names.
- ▶ You should also use a separate HFS for each Version 7 node.
This might require new procedure names if you used a shared HFS in previous versions.
- ▶ Review information in the Information Center article *Premigration considerations*, available at the following Web page:
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/cmig_pre.html

15.7.3 Migration process overview

After the product code of WebSphere Application Server for z/OS V7.0 was brought into the system using SMP/E, the migration is performed in a three step approach:

Note: The migration is always performed on a node basis. In the network deployment configuration, you must always start with the deployment manager node.

1. Backup the old environment to have a fallback option.
2. Create and transfer the Job Control Language (JCL) jobs needed during the actual migration (CNTL and DATA datasets).
3. Run the JCL jobs to perform the migration.

To create the JCL, either the z/OS Migration Management Tool (MMT) or the zMMT.sh script can be used. Both techniques are described in the next sections.

15.7.4 z/OS Migration Management Tool

This section describes the z/OS Migration Management Tool (MMT) used during the migration process on z/OS.

Overview

The MMT is an Eclipse-based application used to create the JCL jobs for the migration. It uses *Migration Definitions*, a construct that contains all data necessary to migrate a WebSphere Application Server for z/OS node from V5.1.x (and later), to V7. It contains the Migration Instructions, personalized for each Migration Definition. It can optionally be used to transfer the JCL to the z/OS target system, if that system has an FTP server up and running.

MMT is intended for use by a systems programmer or administrator who is familiar with the z/OS target system on which the migrated Version 7.0 nodes run. Figure 15-2 shows a high level overview of the migration process with MMT used.

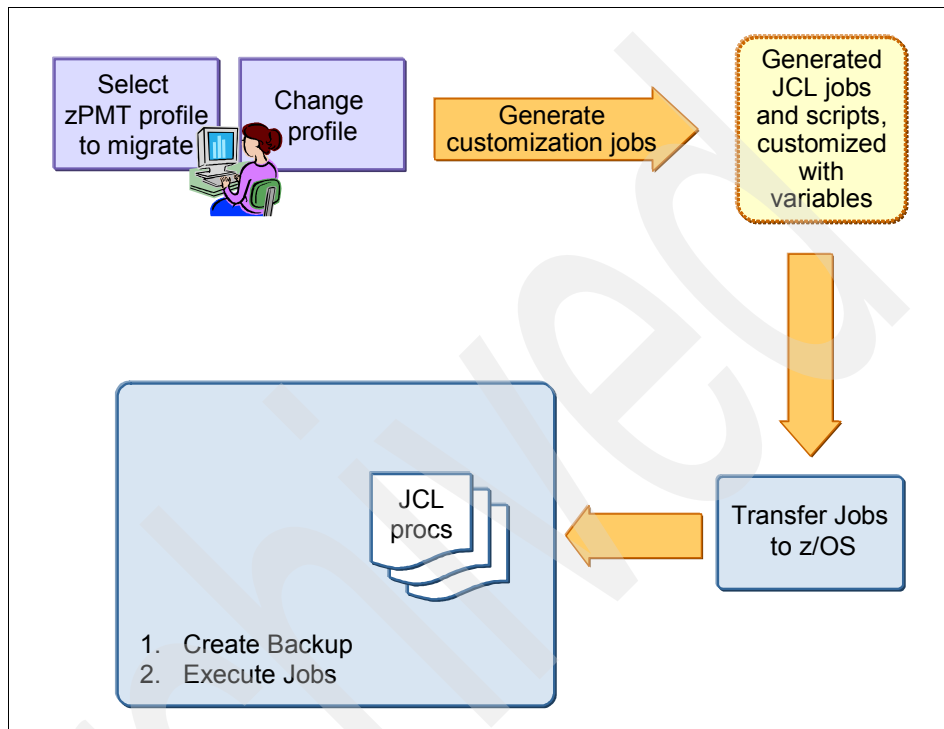


Figure 15-2 Migration process with MMT

Installing the z/OS Migration Management Tool

MMT is available for Windows and Linux-based workstations. It is included in the Optional Material package, FMID: JIWO700. You can also download the WebSphere Configuration Tools package from the internet. The download (130 MB) includes the Profile Management Tool for System z (zPMT) and the MMT. It can be found at the following Web page:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg24020368>

To install the tool, extract it in a convenient directory on your local file system and execute `install.exe`, located in the `WCT\` directory created during the extraction.

On a Windows operating system, navigate to **Start** → **Programs** → **IBM WebSphere** → **WebSphere Customization Tools V7.0**. Click **WebSphere Customization Tools** to start the program.

Note: If you are migrating a server, make sure that you have the appropriate version of the MMT installed. If the MMT is back-level, this might result in migration problems due to old JCL.

Creating a Migration Definition

To create a Migration Definition, follow the steps listed below. For help regarding the MMT, see the Information Center article *Using the z/OS Migration Management Tool to create and manage Migration Definitions*, available from the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.migration.zseries.doc/info/zseries/ae/tmig_zmmt_usemmt.html

1. Start the z/OS Migration Management Tool.
2. To specify a location where you want Migration Definition files to be stored on your workstation or to add another migration location to the Migration Locations table, perform the following steps:
 - a. Click **Add**.
 - b. Enter the path name of the location where you want to store the migration data. The migration location directory must be empty when you create a new migration location.
 - c. Enter a name to be associated with the table entry.
 - d. Select the version of WebSphere Application Server to which you are migrating.
 - e. Click **Finish**.
3. Click **Migrate**. This will bring up a multi-panel dialog.
4. Complete the fields in the panels using the values that you entered for the variables on the configuration worksheet that you created, clicking **Back** and **Next** as necessary.

Tips:

- ▶ The MMT has a good help file. Hover the mouse over a field for help information.
- ▶ On the Server Customization (Part 2) panel there is a Migration Definition identifier shown. Write down this number. This identifier is used to separate the output of individual node migrations. The identifier is also the name of the subdirectory where the JCL will be saved on your workstation.

5. When you have successfully entered all of the necessary information about the panels for this type of Migration Definition, the z/OS Migration Management Tool displays the definition type, location, and name on the Migration Summary panel.
6. Click **Create** to build the Migration Definition on your workstation.
7. Read the information about the Migration Creation Summary panel and click **Finish**.

As a result you will find a directory structure populating the path that was specified to store the Migration Definitions. As the next step, you have to upload the migration jobs using the MMT. See the next section.

Attention: You will find two subdirectories, the CNTL and DATA directory in the created structure at `..\profileTemplates\zos-migDmgr\documents\`. These will contain some JCL jobs. However these are not the complete set. You have to use the MMT to process the JCL, so that it can be used to migrate a WebSphere Application Server for z/OS profile.

Uploading migration jobs

To put the migration jobs on the z/OS, two options exist:

- ▶ Use the MMT to FTP the jobs up to the host. (This option can also allocate the necessary data sets for you.) Open the MMT, select the Migration Definition that you want to work with, and select the **Process** button (Figure 15-3).
- ▶ Create the jobs on the workstation and transfer them manually.

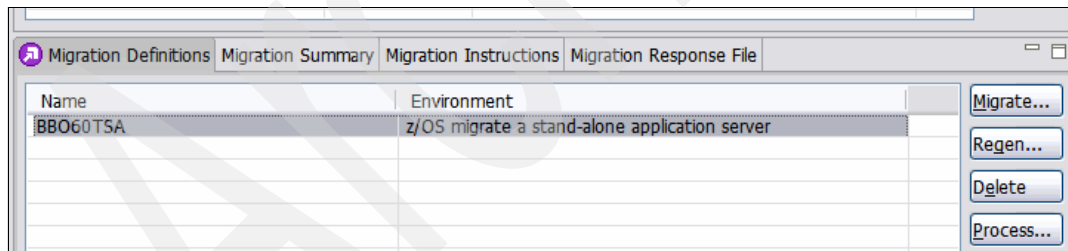


Figure 15-3 MMT options to work with a Migration Definition

For detailed migration instructions, select a Migration Definition by left-clicking it, and then clicking the Migration Instruction tab. The instructions can also be found on the file system of your workstation. The path is stated on the Migration Instruction tab. The instructions reflect the variables entered in the Migration Definition panels.

15.7.5 Migration Management Tool script

This section describes the System z Migration Management Tool script (zMMT.sh), that can be used to create the JCL needed for a node migration.

Overview

The second option allows you to create the migration jobs completely on z/OS using a shell script, zmmt.sh, found in the product bin directory (/usr/lpp/zWebSphere/V7R0/bin). The script will also create the CNTL and DATA data sets and the corresponding JCL that is needed to perform the migration. You will need a response file. This response file contains information about the node construction. There are two options on how to create a response file:

- ▶ Use the MMT
- ▶ Rebuild an example found in the Information Center

Note: We suggest using the MMT to build the response file. This makes sure that any changes brought with new PTFs to the response file are used.

Command

The command can be found in the /bin directory of the product home (default is /usr/lpp/zWebSphere/V7R0). See Example 15-1 on page 496. To run the script, the following parameters can be used:

- ▶ -responseFile
Specifies the path to your response file. This file can either be encoded in ASCII or EBCDIC. The shipped samples use ASCII.
- ▶ -profilePath
Fully qualified path name to an existing set of generated jobs. This parameter cannot be used in combination with the -responsefile option.
- ▶ -workspace
Specifies the Eclipse work space directory.
- ▶ -transfer
Copy generated jobs from a UNIX System Services file system to a pair of partitioned datasets. The zMMT.sh script first writes the customization jobs to a UNIX System Services file system.
- ▶ -allocate
Attempts to allocate the target datasets

Example 15-1 Migration management command

```
zmmt.sh -workspace /xxx -transfer -allocate -responseFile  
/xxx/ZCellcmd.responseFile
```

This command performs the following tasks:

- ▶ Generate the migration jobs to the location specified by profilePath in the response file.
- ▶ Allocate the target CNTL and DATA datasets, using the high level qualifier specified by targetHLQ in the response file.
- ▶ Transfer the jobs from the file system to the CNTL and DATA datasets.

Runtime considerations

When using the zMMT.sh script to create the migration JCL, there are two points that you must be aware of:

- ▶ The script will be run in the osgi command shell, as seen in Example 15-2.

Example 15-2 osgi shell for the zMMT.sh script

```
/usr/lpp/zWebSphere/V7R0/bin#>rkspc -responseFile  
/tmp/zDMgr01.responseFile
```

```
osgi>
```

Note: An osgi command shell is an execution environment that allows the remote management of Java application and components. It is based on the OSGI open standard.

Because the script will take a relative long time to proceed, it might look as though nothing is happening. However, the script will write messages as seen in Example 15-3.

Example 15-3 osgi messages when issuing the zMMT.sh script

```
osgi> Customization definition successfully written to /tmp/ZDMgr01  
Attempting to allocate dataset: BOSS.VICOM.BOSS0173.CNTL  
Allocation successful.  
Attempting to allocate dataset: BOSS.VICOM.BOSS0173.DATA  
Allocation successful.  
Copying CNTL files to BOSS.VICOM.BOSS0173.CNTL...  
Copy successful.  
Copying DATA files to BOSS.VICOM.BOSS0173.DATA...  
Copy successful.
```

- ▶ If you have to rerun the command, delete the profilePath directory. If the directory still exists, the osgi shell will show the error message, shown in Example 15-4:

Example 15-4 MMT error message

```
osgi> The following validation errors were present with the command
line arguments: profilePath: The profile path is not valid.
```

15.7.6 Migration jobs

The migration jobs are created using MMT or the zMMT.sh script. This section gives a short overview of these jobs.

Overview

There are multiple jobs created by the MMT. However, you only need to execute five of them. Table 15-2 gives an overview of what jobs will be used by the migration, depending on the node that should be processed.

Check the detailed migration manual that is created during the JCL built step for the needed user authorities

Table 15-2 WebSphere Application Server for z/OS V7.0 migration jobs

Jobname ^a	DMGR	Federated server	Standalone server
BBOMxZFS or BBOMxHFS	Allocates HFS or zFS	Allocates HFS or zFS	Allocates HFS or zFS
BBOMxCP	Copies tailored JCL to PROCLIB	Copies tailored JCL to PROCLIB	Copies tailored JCL to PROCLIB
BBOWMG1x	n/a	Clear transaction logs (for XA connectors only)	Clear transaction logs (for XA connectors only)
BBOWMG2x	n/a	Disable Peer Restart and Recovery mode (XA only)	Disable Peer Restart and Recovery mode (XA only)
BBOWMG3x	Perform migration	Perform migration	Perform migration

a. The value for x depends on the profile that you are migrating.

Tip: The BBOWMG3x job is long running and can cause certain error conditions (such as an ABEND 522). TIME=NOLIMIT on the JCL job card should avoid the problem. Also note that the BBOWMG1x and BBOWMG2x jobs are only needed if you have any XA connectors defined in your configuration. They do not apply to the deployment manager node migration.

Note: This table is intended to give you an overview on the tasks of each job, not as a manual. For a detailed step-by-step migration guide, we suggest printing the Migration Instruction created by the MMT. (See Figure 15-3 on page 494 for the information how to access the Migration Instruction)

Job BBOWMG3x

The BBOWMG3x job is the job that actually performs the migration. It is the job that will take the longest time to be processed. Listed below are the steps included in the job:

1. Create working directory (/tmp/migrate/nnnnn)
2. WRCONFIG: Copy dialog generated variables to the HFS.
3. WRRESP: Create a profile creation response file from dialog generated variables.
4. MKCONFIG: Gather information from existing configuration (for instance, cell name, server name).
5. VERIFY: Verify the variables generated from dialog.
6. CRHOME: Create a V7 WAS_HOME structure.
7. CRPROF: Create V7 default profile.
8. PREUPGRD: Backup some files in the HFS to be used by WASPostUpgrade.
9. UPGRADE: Run WASPostUpgrade to perform the migration (serverindex.xml renamed to serverindex.xml__disabled).
10. FINISHUP: Run Config2Native, update file permissions and attributes.

A working directory in /tmp is used to do much of the processing. The *nnnnn* is a unique number that was generated during the creation of your migration jobs. For normal migration, the space used in /tmp is small, but if you turn on tracing, the space used can be quite large. Make sure you have enough free space on /tmp.

The UPGRADE step is where the actual migration occurs and will take the longest to complete. The VERIFY step attempts to check the information provided so that the migration does not fail because of bad input parameters.

Troubleshooting for BBOWMG3x

Because a migration is complex, errors may occur. The main source for errors is the BBOWMG3x job, described in the previous section. Here are some troubleshooting tips.

- ▶ If the BBOWMG3x job fails, check the output for errors.
 - /tmp/migrate/nnnnn/BBOWMG3x.out
 - /tmp/migrate/nnnnn/BBOWMG3x.err written to JOBLOG
 - /tmp/migrate/nnnnn/logs directory can contain logs named WAS*Upgrade*<timestamp>.log
- ▶ If you need more information, turn traces on. The trace states are disabled by default. 'xxxx.DATA(BBOWMxEV)' has to be updated to enable tracing:
 - TraceState=enabled
 - profileTrace=enabled
 - preUpgradeTrace=enabled
 - postUpgradeTrace=enabled
- ▶ If the job fails in the VERIFY step, it is most likely that you made an error when specifying information used to create the jobs. Correct the information and rerun the job.
- ▶ If the job fails after the VERIFY step, you need to delete the WAS_HOME directory that was created during the failed run, before re-running the job. Check the original configuration for the serverindex.xml file being renamed to serverindex.xml_disabled also. This is done to signal that the configuration has already been migrated, to stop you from inadvertently migrating the node again. This is done by default but it is possible to change this behavior during the configuration phase. It is a check box in the z/OS migration management tool or you can set the keepDMGREnabled parameter to true in the response file.

15.7.7 Migration considerations for 64-bit mode

WebSphere Application Server V7.0 runs in 64-bit mode. Consider the following.

Application considerations

For code written in pure Java, the general experience is that there are no changes necessary to the code to run it in a 64-bit application server.

If the application uses the Java Native Interface (JNI) to call a native program, that native program must be a 64-bit program. Typically, these native programs will be code written in C or C++, or perhaps LE compliant assembler. This point is especially important to check for when using in-house applications that use older native programs.

For more information about how to convert applications to run in 64-bit mode, see the following resources:

- ▶ IBM Whitepaper WP101095 *C/C++ Code Considerations With 64-bit WebSphere Application Server for z/OS*, available at the following Web page:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101095>
- ▶ *Language Environment Programming Guide for 64-bit Virtual Addressing Mode*, SA22-7569

Bigger heap needed for applications

Simply using the 64-bit addressing mode does not mean that the sizes for the various heaps need to be increased.

In general, good minimum and maximum heap sizes should be identified by a verbose garbage collection (GC) analysis. This technique will allow you to identify these values, thereby reducing the GC overhead and saving CPU time. It is suggested to perform a verbose GC analysis on a regular basis, especially if the number of users or the amount of transactions has changed.

For more information about how to perform a verbose GC analysis refer to the Information Center article *Tuning the IBM virtual machine for Java*, available at the following Web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0//topic/com.ibm.websphere.zseries.doc/info/zseries/ae/tprf_tunejvm_v61.html

Note: In general the structure of the WebSphere Application Server for z/OS will significantly reduce the maximum heap size compared to the maximum heap sizes used on the distributed platforms.

For more information about how this is achieved see 14.1.5, “Runtime processes” on page 425.



Sample topology walkthrough

This appendix explores a complex topology and provides general guidance on setting it up.

It contains the following sections:

- ▶ “Topology review” on page 502
- ▶ “Sample topology” on page 504
- ▶ “Installation” on page 507
- ▶ “Deploying applications” on page 510
- ▶ “Configuring security” on page 511
- ▶ “Testing the topology” on page 513

Topology review

The topology shown in Figure A-1 takes elements from several of the topologies found in Chapter 5, “Topologies” on page 121 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 customers and IBM teams who are responsible for the implementation of WebSphere environments.

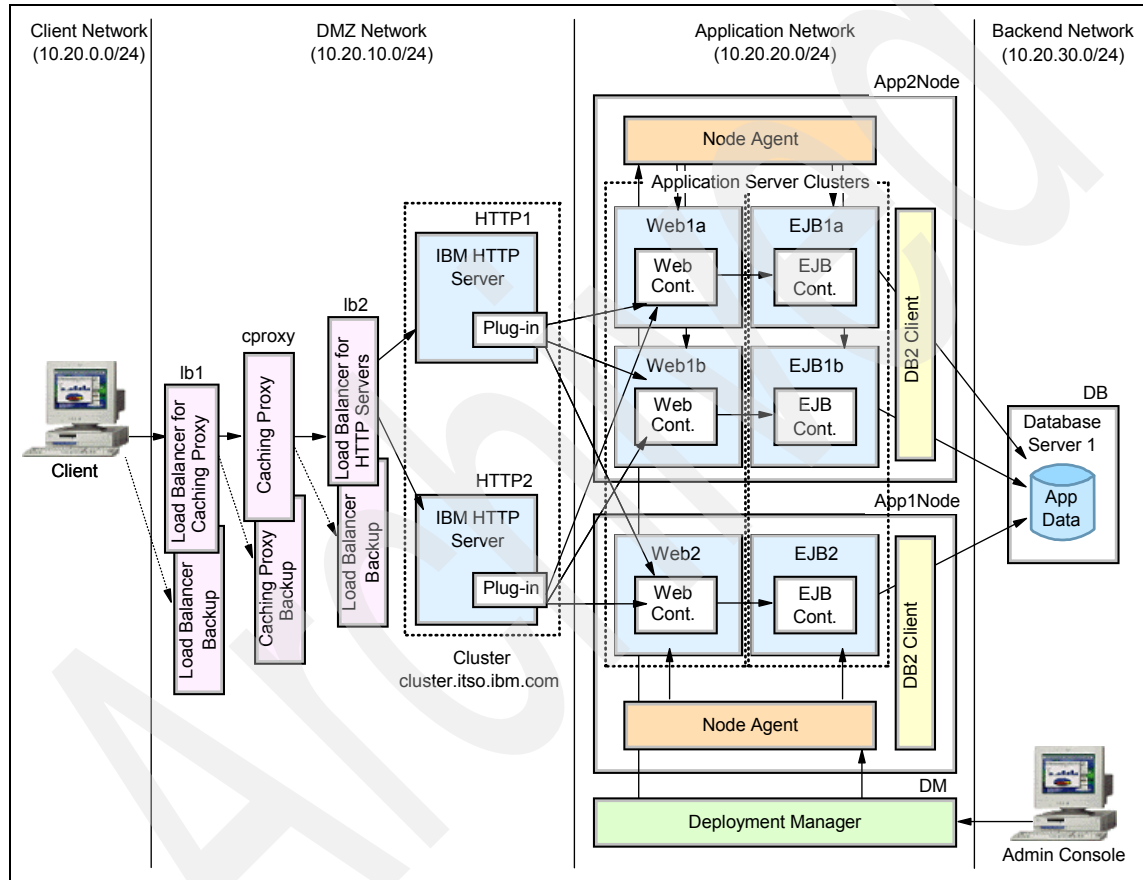


Figure A-1 Complex topology

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 that keeps a local cache of recently accessed pages
Caching Proxy is included in WebSphere Application Server Edge Components. Cacheable content includes static Web pages and JSPs with dynamically generated but infrequently changed fragments. The Caching Proxy can satisfy subsequent requests for the same content by delivering it directly from the local cache, which is much quicker than retrieving it again from the content host.
A backup server is configured for high availability.
- ▶ A Load Balancer to direct incoming requests to the Caching Proxy and a second Load Balancer to manage the workload across the HTTP servers
Load Balancer is included in WebSphere Application Server Edge Components. 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 to provide 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. 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 V9.

Advantages

This topology is designed to maximize scalability, workload management, and high availability. It 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 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 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 as follows:
 - 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

The topology has the following disadvantages:

- ▶ This appendix 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.
- ▶ Isolating multiple components and providing hardware backups means that more licenses and machines are needed, increasing the overall cost.

Sample topology

This section addresses a simplified subset of the topology described in the previous section. Figure A-2 on page 505 shows the sample topology that we will implement.

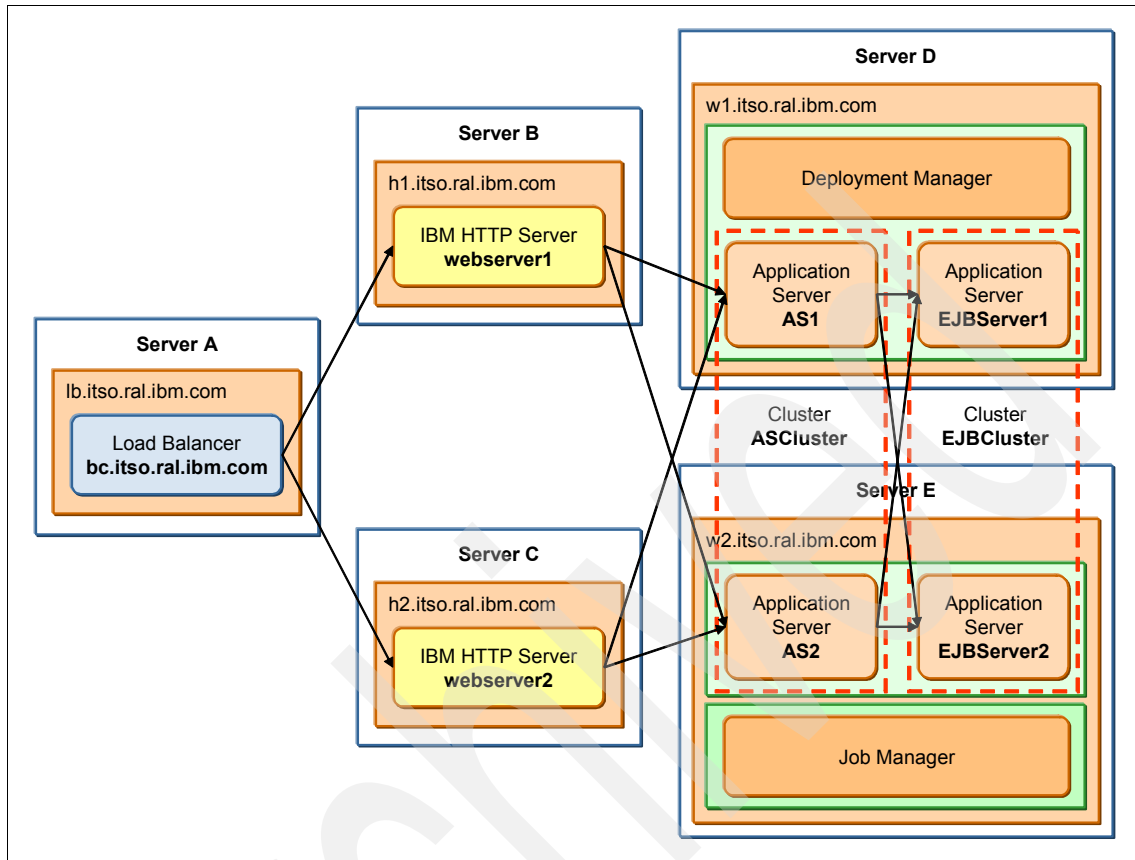


Figure A-2 Sample topology

Characteristics

The characteristics of this topology can be summarized as follows:

- ▶ The topology has been designed to allow the deployment of the BeenThere sample application that is included in WebSphere Application Server V7.0.

The BeenThere sample application demonstrates the workload management and clustering capabilities of WebSphere Application Server. Because its responses contain the application servers where requests have been processed, it is possible to test load balancing and availability with this application.

- ▶ In order to show the new security domains feature, a security domain for the application servers has been created and configured with a different user realm. The administrative application uses global security settings based on a file registry. The BeenThere application uses the operating system registry. Figure A-3 shows the security configuration.

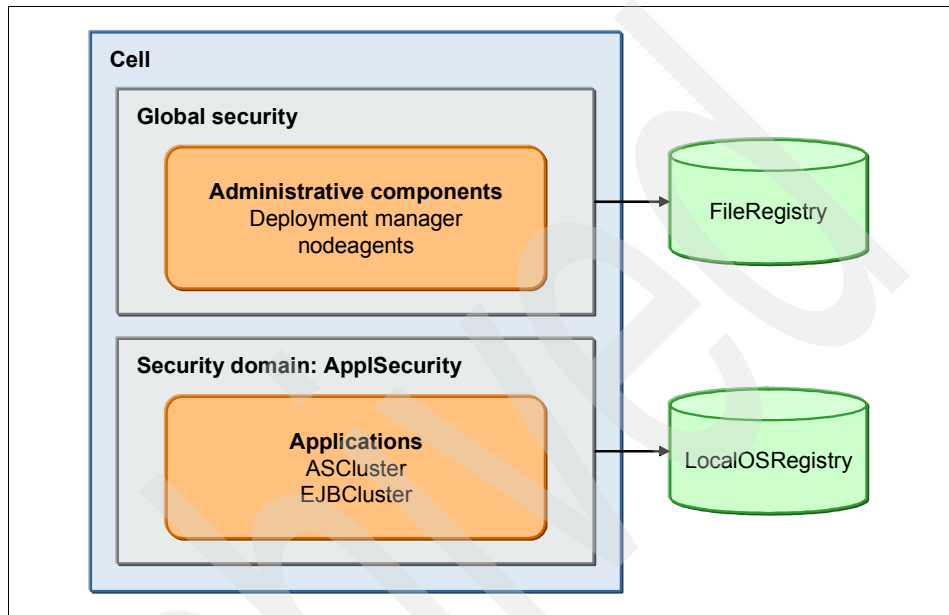


Figure A-3 Security configuration

- ▶ A job manager has also been installed to test the capabilities of this new component in our topology.
- ▶ Only one Load Balancer has been installed. This constitutes a single point of failure, but one Load Balancer is enough to demonstrate its high availability capabilities.
- ▶ No DB2 servers have been installed because the BeenThere application does not need an external data repository.

Installation

The installation process for all the components is fairly similar. All the installations are started from the launchpad for the Network Deployment package. In the following sections we describe the steps that had to be done to install each component. We used a Windows environment to perform this installation.

Installing the Load Balancer (server A)

The Load Balancer component runs in server A and is used to distribute traffic between our two Web servers.

Perform the following steps to install the Load Balancer:

1. Install the Load Balancer by performing the following steps:
 - a. Start the launchpad and click **IBM Edge Components**.
 - b. In the panel, click **Launch the installation wizard for Edge Components Load Balancer and Caching Proxy**. This launches the installer.
 - c. Go through the panels accepting the default options.
2. Configure the load balanced cluster by performing the following steps:
 - a. Launch lbadmin.
 - b. Click **Dispatcher** → **Connect to Host**.
 - c. Click **Dispatcher** → **Start Configuration Wizard**.
 - d. Following the wizard panels, create a cluster with the new IP address and add the two Web servers in it. The new IP address is the public address that clients have to use to access the BeenThere application.
3. Configure loopback adapters in Web servers by performing the following steps:
 - a. Add a loopback adapter in both HTTP server systems (server B and C).
 - b. Configure the cluster address in the loopback adapters.
 - c. Disable the Windows firewall (our servers' operating system was Windows XP).

Installing HTTP Servers (servers B and C)

Both servers B and C run a Web server that directs the incoming requests to one of the application servers.

We performed the following steps on both servers:

1. Launch the installer for the HTTP server from the launchpad.
2. Click **IBM HTTP Server Installation**.
3. In the panel, click **Launch the installation wizard for IBM HTTP Server**.
4. In the wizard panels, select the default options.
5. Start the server from **Start → Programs → IBM HTTP Server V7.0 → Start HTTP Server**.

Creating Deployment manager (server D)

Server D hosts the deployment manager used to administer the servers, applications, and resources in the WebSphere Application Server cell. To build this node:

1. Install WebSphere Application Server Network Deployment by performing the following steps:
 - a. From the launchpad, select **Launch the installation wizard for WebSphere Application Server Network Deployment**.
 - b. Navigate through the panels in the installation wizard selecting the default options.
2. Create a deployment manager profile by performing the following steps:
 - a. Launch the PMT.
 - b. Create a management profile of type deployment manager with the default options.
3. Start the deployment manager from **Start → Programs → IBM WebSphere → WebSphere Application Server Network Deployment V7.0 → Profiles → Dmgr01 → Start the deployment manager**.

Creating Application servers (servers D and E)

Server D and E host the two application server clusters needed for the BeenThere application.

The process for installing and building the WebSphere Application Server components is the same for each application server node.

We performed the following actions:

1. Install WebSphere Application Server Network Deployment by performing the following steps:
 - a. In the launchpad, click **WebSphere Application Server Installation**.
 - b. Click **Launch the installation wizard for the WebSphere Application Server Network Deployment** and select the default options.
2. Create a custom profile for the node by performing the following steps:
 - a. Launch PMT and select **Custom Profile**.
 - b. Follow the prompts taking the defaults, including the federation of the node to the cell as part of the process (the deployment manager must be installed and running).
3. Create the application server clusters by performing the following steps:
 - a. Log into the Integrated Solutions Console.
 - b. Click **Servers** → **Clusters** → **WebSphere application server clusters**.
 - c. Add the two new clusters with the names and member weights showed in Table A-1:

Table A-1 Cluster names and weights

Cluster name	Member name	Weights
ASCluster	AS1	2
	AS2	3
MyEJBCluster	EJBServer1	1
	EJBServer2	3

The chosen weights are those indicated in the BeenThere application installation instructions and are intended to illustrate the workload management capabilities of the product.

Creating Job manager (server E)

The job manager runs on server E and is used during our tests to stop and start the application servers.

Because WebSphere Application Server Network Deployment was already installed, we only had to follow these steps:

1. Create a job manager profile by performing the following steps:
 - a. Launch PMT.
 - b. Create a management profile of type job manager taking the default options.
2. Register the deployment manager to the job manager by performing the following steps:
 - a. From the deployment manager profile, run the wsadmin command tool.
 - b. Execute the following command:

```
$AdminTask registerWithJobManager {-managedNodeName  
w1CellManager01 -host w2.itso.ra1.ibm.com -port 9943 -alias  
SampleTopologyCell}
```

The parameters are as follows:

- `manageNodeName`
The name of the node to be registered. In our example, it is the cell node.
- `host`
The host name of the server running the job manager.
- `port`
The administrative port number to which the job manager listens.
- `alias`
The name that identifies the managed node in job manager.

Deploying applications

The deployment of the application was done following the deployment instructions included in the product for this task. See `readme_BeenThere.html` under the `samples` directory of your WebSphere Application Server V7.0 installation.

Perform the following steps to deploy applications.

1. Deploy the application by performing the following steps:
 - a. Log into the Integrated Solutions Console.
 - b. Click **Applications** → **New application** → **New enterprise application**.
 - c. Select the remote file system and look for the BeenThere.ear file.
 - d. Accept the default options, but map the BeenThere WAR module to ASCluster and the EJB module to MyEJBCluster.
2. Generate a new plug-in for the IBM HTTP Servers by performing the following steps:
 - a. Click **Environment** → **Update global Web server plug-in configuration**.
 - b. Click **OK** to update the plug-in file.
 - c. Copy the new plug-in file to webserver1 and webserver2. The default target directory is as follows:
`C:\Program Files\IBM\HTTPServer\Plugins\config\your_web_server.`

Configuring security

As mentioned before, we created a security domain with a different user realm for the applications servers where the BeenThere application runs.

Because the profile for the deployment manager had already been created with security enabled, we performed the following steps:

1. Change administrative user registry by performing the following steps:
 - a. Log into the Integrated Solutions Console.
 - b. Change the default user registry to **standalone custom registry** in the global security settings. This registry is configured by default to use the FileRegistrySample class, which allows us to use a users file and a groups file in the local file system as our registry.
 - c. On servers D and E, from Windows, create two files with the following names and content:
 - `c:\users.txt`
Administrator:itso4you:1:1:admin
 - `c:\groups.txt`
admins:1:Administrator:Administrative group

- d. On the Integrated Solutions Console, click the **Configure** button and add two custom properties:
 - usersFile = c:\users.txt
 - groupsFile = c:\groups.txt
- e. In the **Global security** panel, click **Set as current**.
2. Enable application security. In the **Global security** panel, set **Enable application security**.
3. Add a security domain and set its scope to application servers by performing the following steps:
 - a. Click **Security** → **Security domain**.
 - b. Add a new domain named ApplSecurity.
 - c. In the new domain settings, select the **Customize for this domain** and **Local operating system** check boxes. In the configuration for this registry, introduce a name for it: ApplRealm.
 - d. Set the scope of this domain to include the two clusters. Select **ASCluster** and **MyEJBCluster**.
4. Add a new Windows user account on servers C and D. It will be used to log into the application.
5. Assign administrative roles.

The user account being used to log onto the BeenThere application needs an administrative role, otherwise the application will not work. This is because the application needs to get the node name from WebSphere Application Server and, therefore, the monitor role is enough. To assign this role to the user, perform the following steps:

 - a. Click **Global security** → **Administrative user roles**.
 - b. Select **AppIRealm**, because this is the realm defined to be used by the application, and add the user to the list of mapped roles.
6. Assign application roles.

This application has a role (administrator) that has to be assigned to the user. Perform the following steps to assign application roles:

 - a. Click **Applications** → **Application types** → **WebSphere enterprise applications**.
 - b. Select **BeenThere** and click **Security role to user / group mapping**.
 - c. Map the user from the AppIRealm to the administrator role.
7. Restart application servers.

Servers in the scope of the new security domain have to be restarted in order to get the new configuration settings.

Note: We suggest using a centralized user repository like LDAP or, in a z/OS environment, RACF. Neither the sample file registry nor the local OS registry should be used in production environments. These two registries have been used in this chapter only for illustration purposes.

Testing the topology

After building the environment according to our sample topology, we were ready to test and verify it.

We considered two different aspects to be tested:

- ▶ Service
- ▶ Administration

Service

Because the main purpose of our tests was to demonstrate the clustering, load balancing, and high availability capabilities of our sample topology, we challenged our environment in different conditions. For each test case described below, we used a Web browser and typed in the following URL:

`http://bc.itso.ral.ibm.com/wlm/BeenThere?weights=false&count=4`

Normal functioning

In this test, every component was up and running so we could show the load balancing features in WebSphere Application Server.

Because it was our first test, we got a window (Figure A-4) asking for username and password. We used the credentials of the account we had created in the operating system registry.

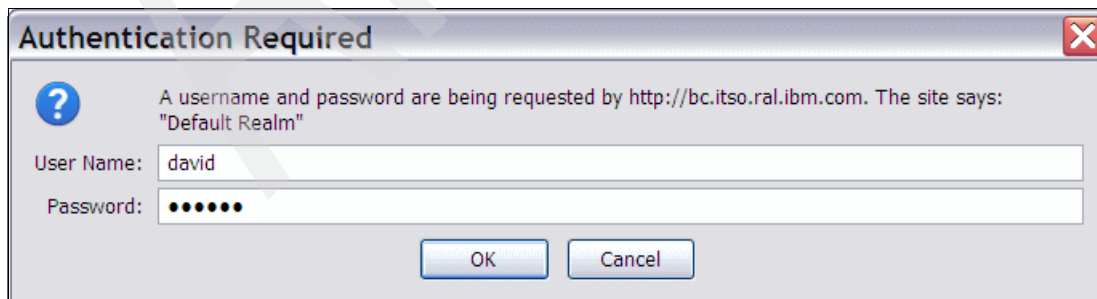
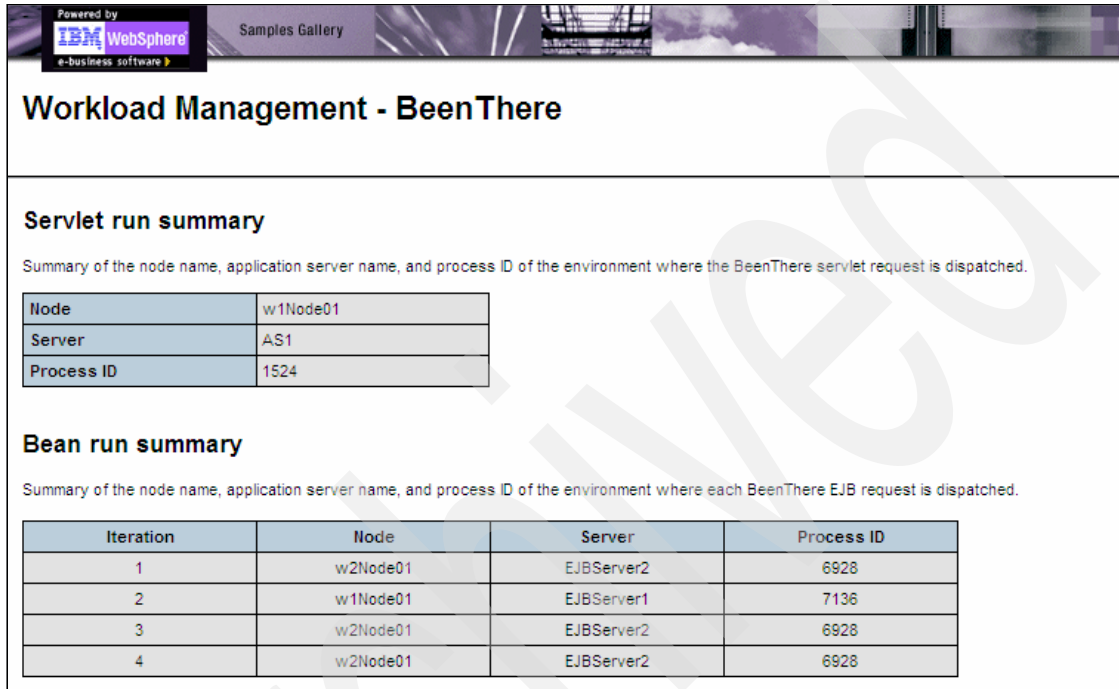


Figure A-4 Authentication dialog

After clicking **OK**, we got the answer from the application (Figure A-5). The request was processed by the servlet on node ws1node1, three of the four iterations were processed by EJBServer2 and the other one by EJBServer1. This distribution of the requests to the EJB servers is a consequence of the weights configured when we created those servers (Table A-1 on page 509).



Powered by IBM WebSphere e-business software

Samples Gallery

Workload Management - BeenThere

Servlet run summary

Summary of the node name, application server name, and process ID of the environment where the BeenThere servlet request is dispatched.

Node	w1Node01
Server	AS1
Process ID	1524

Bean run summary

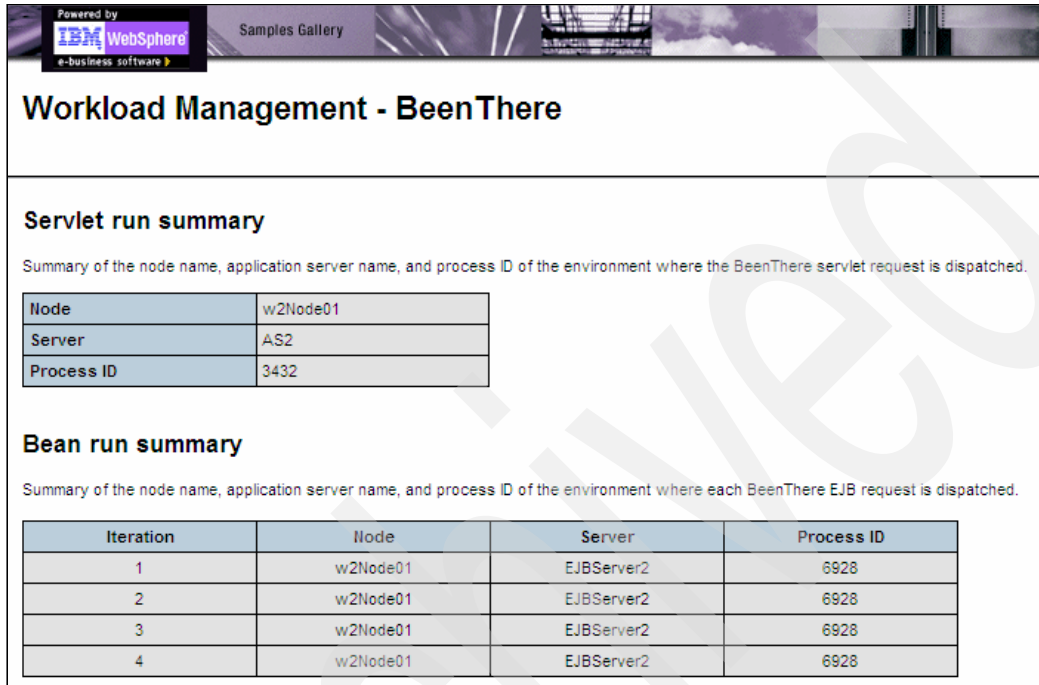
Summary of the node name, application server name, and process ID of the environment where each BeenThere EJB request is dispatched.

Iteration	Node	Server	Process ID
1	w2Node01	EJBServer2	6928
2	w1Node01	EJBServer1	7136
3	w2Node01	EJBServer2	6928
4	w2Node01	EJBServer2	6928

Figure A-5 Response from BeenThere application

One WebSphere Application Server node down

In the last of our test scenarios, we stopped server w1.itso.ral.ibm.com. The system still responded to our requests, which were all processed by w2.itso.ral.ibm.com (Figure A-7).



The screenshot shows the 'Workload Management - BeenThere' page. It includes a 'Servlet run summary' table with one row: Node (w2Node01), Server (AS2), and Process ID (3432). Below it is a 'Bean run summary' table with four rows, all showing EJBServer2 on w2Node01 with Process ID 6928.

Powered by **IBM WebSphere** e-business software

Samples Gallery

Workload Management - BeenThere

Servlet run summary

Summary of the node name, application server name, and process ID of the environment where the BeenThere servlet request is dispatched.

Node	w2Node01
Server	AS2
Process ID	3432

Bean run summary

Summary of the node name, application server name, and process ID of the environment where each BeenThere EJB request is dispatched.

Iteration	Node	Server	Process ID
1	w2Node01	EJBServer2	6928
2	w2Node01	EJBServer2	6928
3	w2Node01	EJBServer2	6928
4	w2Node01	EJBServer2	6928

Figure A-7 All responses came from EJBServer2

Administration

Our purpose was to test some of the features provided by the new job manager component.

Because we needed to stop the applications servers running on w1.itso.ral.ibm.com for the last scenario of the previous section of this appendix, we used the functionality provided by the job manager.

The following screenshots show the steps followed to stop server AS1 after we logged into the Integrated Solutions Console and clicked on **Jobs** → **Submit**:

1. In the first panel (Figure A-8), we selected the job **Stop server**.

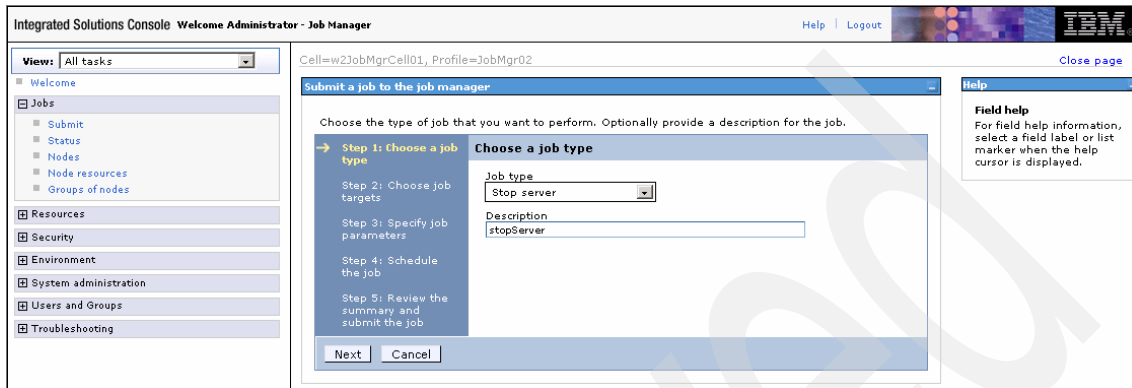


Figure A-8 Job selection

2. We selected the node we had registered to job manager (Figure A-9).

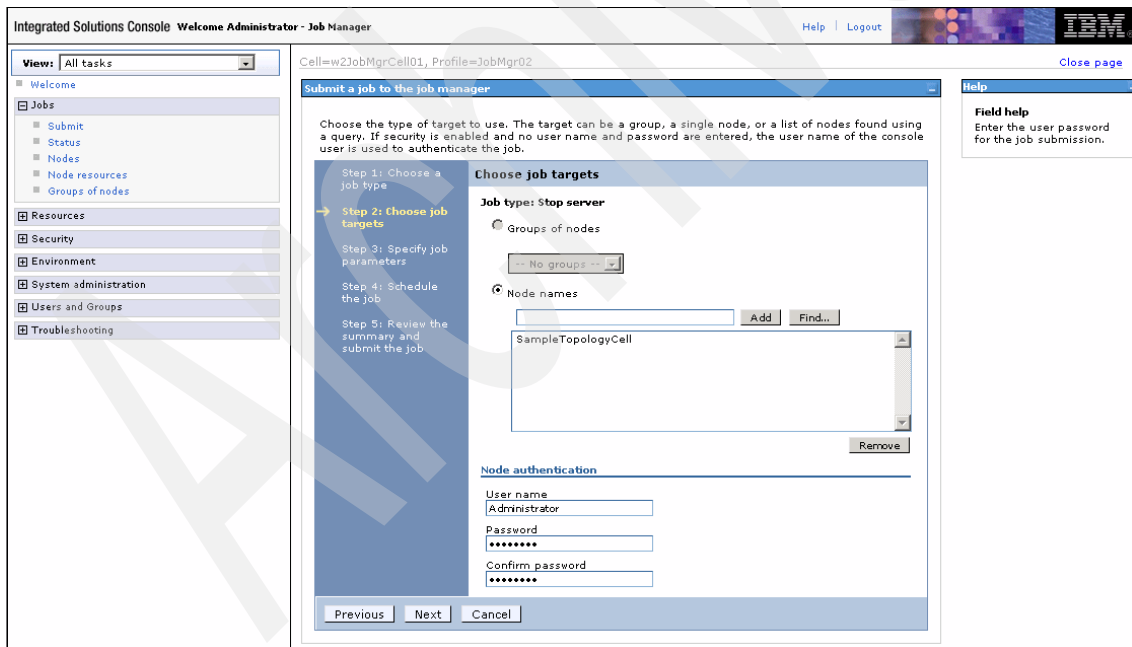


Figure A-9 Target selection

3. We selected server AS1 (Figure A-10).

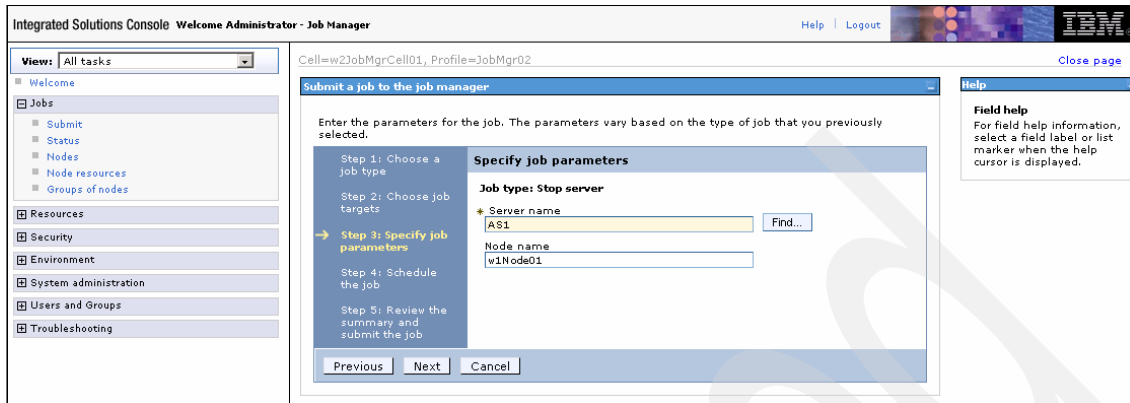


Figure A-10 Additional parameters

4. Because we wanted to immediately stop the server, we accepted the default option in the schedule panel (Figure A-11).

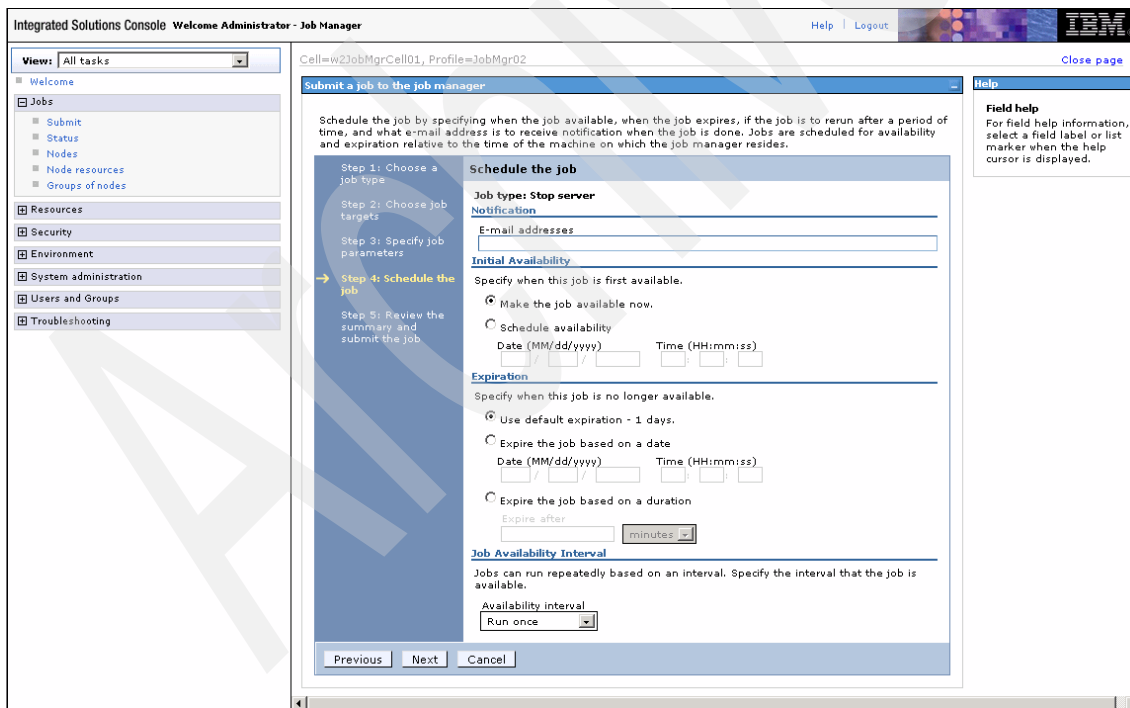


Figure A-11 Job schedule options

The next panel (Figure A-12) showed a summary of the job.

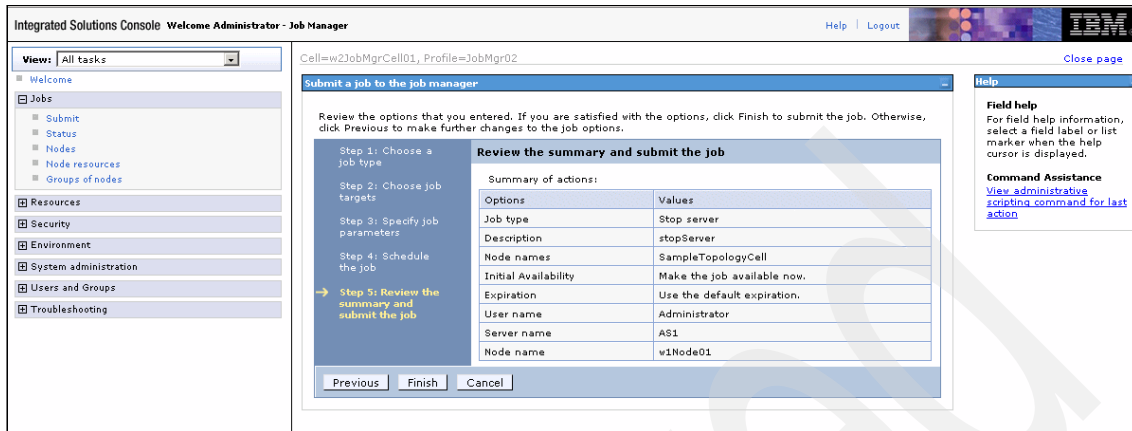


Figure A-12 Job summary

After submitting the job, we got the status window showing that our job was still pending (Figure A-13).

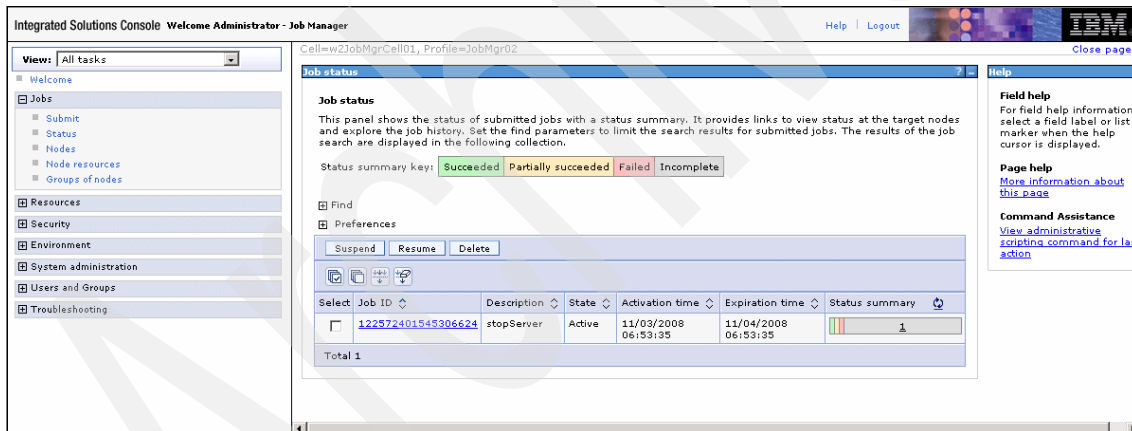


Figure A-13 Job pending

We had to wait a few minutes until the job finished. Figure A-14 shows the status window indicating that the job finished successfully.

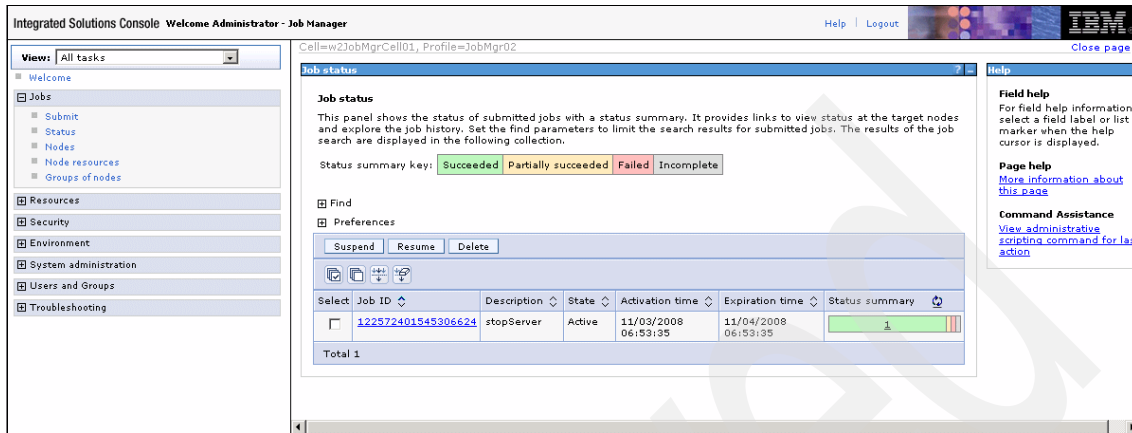


Figure A-14 Job finished

5. We repeated the same procedure to stop EJBserver1.

Summary

In this chapter, we have reviewed a commonly used complex topology. We have also presented a simplified version of this topology, and used it to illustrate the installation of the different components. Finally, we have demonstrated its excellent response to our challenges.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 522. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Building Dynamic Ajax Applications Using WebSphere Feature Pack for Web 2.0*, SG24-7635
- ▶ *Enabling SOA Using WebSphere Messaging*, SG24-7163
- ▶ *IBM Tivoli Composite Application Manager Family Installation, Configuration, and Basic Usage*, SG24-7151
- ▶ *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316
- ▶ *IBM WebSphere DataPower SOA Appliances Part I: Overview and Getting Started*, REDP-4327
- ▶ *Java Stand-alone Applications on z/OS, Volume I*, SG24-7177
- ▶ *Java Stand-alone Applications on z/OS Volume II*, SG24-7291
- ▶ *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
- ▶ *Rational Application Developer V6 Programming Guide*, SG24-6449
- ▶ *Rational Application Developer V7 Programming Guide*, SG24-7501
- ▶ *Solution Deployment Guide for IBM Tivoli Composite Application Manager for WebSphere*, SG24-7293
- ▶ *System Programmer's Guide to: Workload Manager*, SG24-6472
- ▶ *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257
- ▶ *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688
- ▶ *WebSphere Application Server V6 Migration Guide*, SG24-6369

- ▶ *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798
- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- ▶ *WebSphere Application Server V6.1: Planning and Design*, SG24-7305
- ▶ *WebSphere Security Fundamentals*, REDP-3944
- ▶ *WebSphere Service Registry and Repository Handbook*, SG24-7386

Online resources

The following Web site is also relevant as an information source:

- ▶ *What's new in WebSphere Application Server V7 developerWorks article*
http://www.ibm.com/developerworks/websphere/library/techarticles/0809_alcott/0809_alcott.html

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Redbooks

WebSphere Application Server V7.0: Concepts, Planning, and Design

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



WebSphere Application Server V7.0: Concepts, Planning, and Design



Discusses end-to-end planning for WebSphere implementations

Defines WebSphere concepts and best practices

Addresses distributed and z/OS platforms

This IBM Redbooks publication discusses the concepts, planning, and design of WebSphere Application Server V7.0 environments. This book is aimed at IT architects and consultants who want more information for the planning and designing of application-serving environments, ranging from small to large, and complex implementations.

This IBM Redbooks publication addresses the packaging and the features incorporated into 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 for WebSphere Application Server V7.0 and WebSphere Application Server Network Deployment V7.0 on distributed platforms, and WebSphere Application Server for z/OS V7.0. It also covers migration considerations for migrating from previous releases.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**