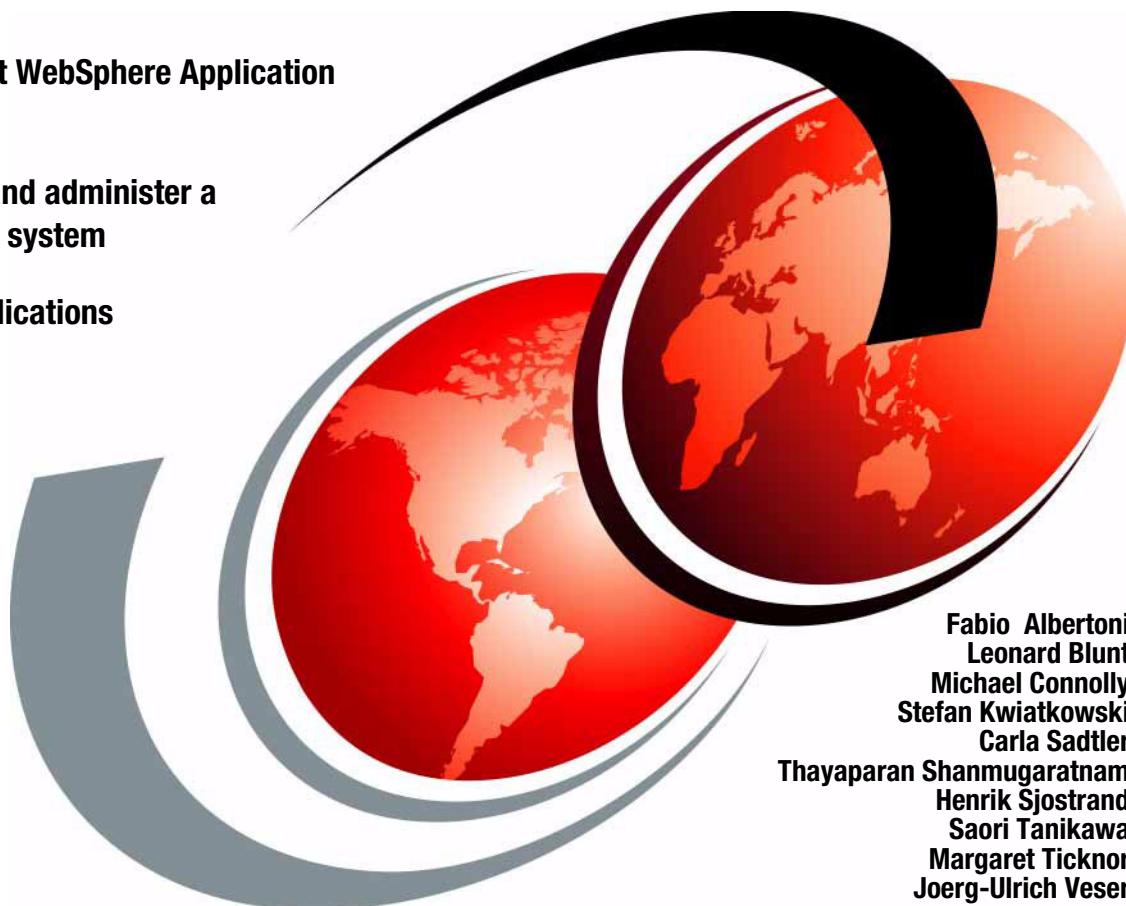


WebSphere Application Server V7 Administration and Configuration Guide

Learn about WebSphere Application Server V7

Configure and administer a WebSphere system

Deploy applications



Fabio Albertoni
Leonard Blunt
Michael Connolly
Stefan Kwiatkowski
Carla Sadtler
Thayaparan Shanmugaratnam
Henrik Sjostrand
Saori Tanikawa
Margaret Ticknor
Joerg-Ulrich Veser

Redbooks



International Technical Support Organization

**WebSphere Application Server V7
Administration and Configuration Guide**

March 2010

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

Second Edition (March 2010)

This edition applies to WebSphere Application Server V7.

Contents

Notices	xiii
Trademarks	xiv
Preface	xv
The team who wrote this book	xv
Now you can become a published author, too!	xviii
Comments welcome.....	xix
Stay connected to IBM Redbooks	xix
Summary of changes	xxi
March 2010, Second Edition	xxi
Part 1. Basic administration and configuration techniques	1
Chapter 1. System management: A technical overview	3
1.1 System management overview	4
1.1.1 Terminology	4
1.1.2 Directory conventions	5
1.1.3 Profiles	5
1.1.4 System management tools	7
1.2 System management in a standalone server environment.....	8
1.3 System management of multiple standalone servers	9
1.4 System management in a distributed server environment	12
1.4.1 Centralized changes to configuration and application data.....	14
1.4.2 Rules for process startup	15
1.4.3 Distributed process discovery.....	16
1.4.4 Configuration and application data repository.....	18
1.4.5 File synchronization in distributed server environments	25
1.5 Management of distributed and standalone servers	32
1.6 Java Management Extensions (JMX)	35
1.6.1 JMX MBeans.....	36
1.6.2 JMX usage scenarios	36
1.7 Centralized Installation Manager.....	37
1.8 IBM Support Assistant V4	37
Chapter 2. Working with profiles on distributed systems	41
2.1 Types of profiles	42
2.1.1 Application server profile.....	43
2.1.2 Deployment manager profile.....	43

2.1.3 Custom profile	44
2.1.4 Cell profile	44
2.1.5 Administrative agent profile	44
2.1.6 Job manager profile	45
2.1.7 Profile generation	47
2.2 Planning for profiles	47
2.3 Building systems with profiles	47
2.3.1 Starting the PMT	48
2.3.2 Common panels and steps for all profiles	50
2.3.3 Creating an application server profile	62
2.3.4 Creating a deployment manager profile	72
2.3.5 Creating a cell profile	76
2.3.6 Creating a custom profile	77
2.3.7 Federating nodes to a cell	82
2.3.8 Creating an administrative agent profile	90
2.3.9 Creating a job manager profile	95
2.3.10 Registering nodes to an administrative agent	95
2.3.11 Deregistering a node from the administrative agent	99
2.3.12 Registering an administrative agent node with a job manager	99
2.3.13 Registering a deployment manager with a job manager	103
2.4 Managing profiles	105
2.4.1 Using the manageprofiles command	105
2.4.2 Getting help	106
2.4.3 Getting a list of profiles	107
2.4.4 Creating a profile with the manageprofiles command	107
2.4.5 Creating a profile in silent mode with PMT	109
2.4.6 Deleting profiles	109
Chapter 3. Working with profiles on z/OS systems	113
3.1 Creating WebSphere environments	114
3.1.1 WebSphere Application Server for z/OS	116
3.1.2 WebSphere DMZ secure proxy server for z/OS	117
3.2 Getting started with the profile management tool	117
3.3 Sample environment	123
3.4 Creating a deployment manager definition	124
3.4.1 Creating the customization definition	124
3.4.2 Uploading the jobs to the z/OS system	150
3.4.3 Executing the jobs	152
3.5 Creating the base application server definition	155
3.6 Federating an application server	177
3.7 Creating a job manager profile	183
3.8 Creating an administrative agent profile	193

Chapter 4. Centralized Installation Manager	205
4.1 Planning considerations	206
4.1.1 Linux and AIX target requirements	207
4.1.2 Requirement when using CIM for installing or uninstalling maintenance on AIX target as non-root user	208
4.1.3 Update Installer	209
4.1.4 Repository directory structure	209
4.2 Installing CIM and creating the repository	210
4.3 Loading additional product packages into the repository	212
4.3.1 Installing Installation Factory	212
4.3.2 Package types	213
4.3.3 Adding product packages to the CIM repository	213
4.3.4 Adding maintenance when the deployment manager is connected to the Internet	216
4.3.5 When the deployment manager is not connected to the Internet	220
4.4 Using CIM to manage your environment	222
4.4.1 Adding additional installation targets outside of the cell	222
4.4.2 Installing a Secure Shell (SSH) public key	224
4.4.3 Removing installation target systems	226
4.4.4 Installing packages to the target systems	226
4.4.5 Product installation	227
4.4.6 Installing maintenance to target systems	230
4.4.7 Uninstalling packages	236
4.4.8 CIM AdminTask Commands	238
Chapter 5. Administration consoles and commands	239
5.1 Introducing the WebSphere administrative consoles	240
5.1.1 Starting and accessing the consoles	241
5.1.2 Logging in to a console	243
5.1.3 Changing the administrative console session timeout	245
5.1.4 The graphical interface	245
5.1.5 Finding an item in the console	252
5.1.6 Updating existing items	258
5.1.7 Adding new items	260
5.1.8 Removing items	262
5.1.9 Starting and stopping items	262
5.1.10 Using variables	263
5.1.11 Saving work	265
5.1.12 Getting help	266
5.2 Securing the console	266
5.2.1 Enabling security after profile creation	267
5.2.2 Administrative security roles	269
5.3 Job manager console	272

5.3.1	Submitting a job with the job manager	274
5.3.2	Distributing files using the job manager	283
5.4	Using command line tools	285
5.4.1	Command location	285
5.4.2	Key usage parameters	286
5.4.3	Entering commands	286
Chapter 6. Administration of WebSphere processes		289
6.1	Working with the deployment manager	290
6.1.1	Deployment manager configuration settings	290
6.1.2	Starting and stopping the deployment manager	292
6.2	Starting and stopping an administrative agent	296
6.3	Starting and stopping the job manager	297
6.4	Working with application servers	297
6.4.1	Creating an application server	298
6.4.2	Viewing the status of an application server	307
6.4.3	Starting an application server	310
6.4.4	Stopping an application server	312
6.4.5	Viewing runtime attributes of an application server	315
6.4.6	Customizing application servers	316
6.5	Working with nodes in a distributed environment	325
6.5.1	Starting and stopping nodes	326
6.5.2	Node agent synchronization	329
6.5.3	Removing a node from a cell	331
6.5.4	Renaming a node	334
6.5.5	Node groups	334
6.6	Working with clusters	337
6.6.1	Creating application server clusters	337
6.6.2	Viewing cluster topology	349
6.6.3	Managing clusters	349
6.7	Working with virtual hosts	350
6.7.1	Creating and updating virtual hosts	352
6.8	Managing your configuration files	353
6.8.1	Backing up a profile	353
6.8.2	Restoring a profile	354
6.8.3	Exporting and importing profiles	356
6.9	Managing applications	357
6.9.1	Managing enterprise applications: Administrative console	358
6.9.2	Installing an enterprise application	360
6.9.3	Uninstalling an enterprise application	366
6.9.4	Starting an enterprise application	367
6.9.5	Stopping an enterprise application	367
6.9.6	Preventing an enterprise application from starting on a server	368

6.9.7	Viewing application details	368
6.9.8	Finding a URL for a servlet or JSP	371
6.10	Enabling process restart on failure	375
Chapter 7. WebSphere Application Server for z/OS	381
7.1	WebSphere Application Server on z/OS Architecture	382
7.1.1	Architecture of a single application server on z/OS	382
7.1.2	Cell architecture of WebSphere Application Server for z/OS	389
7.2	WebSphere Application Server for z/OS operations	392
7.2.1	Structure of the configuration HFS	392
7.2.2	Load module libraries in the HFS	394
7.2.3	Changed start procedure JCLs with V7	395
7.2.4	Starting and stopping an application server	398
7.2.5	Logging and tracing	400
7.3	Maintenance for the HFS	401
7.3.1	The process of applying maintenance	402
7.3.2	The concept of intermediate symbolic links	402
7.4	Workload management	405
7.4.1	Workload management overview	406
7.4.2	Workload classification	407
7.4.3	Transaction classification	407
7.4.4	Servant activation	412
7.4.5	Basic WLM classifications	412
7.5	What is new in V7 for z/OS	414
7.5.1	z/OS Fast Response Cache Accelerator (FRCA)	415
7.5.2	Thread hang recovery	422
7.5.3	Systems Management Facility (SMF) Subtype 9	428
7.6	Thread management using the workload profile	429
7.7	Local connectivity to DB2	432
7.7.1	Prerequisites for implementing a JDBC type 2 driver	432
7.7.2	Creating a JDBC provider	433
7.7.3	Defining a type 2 data source	436
7.8	Migrating to V7	438
Chapter 8. Administration with scripting	439
8.1	Overview of WebSphere scripting	440
8.1.1	Script programming languages	440
8.2	Launching wsadmin	441
8.2.1	Scripting environment properties file	442
8.2.2	Script profile file	444
8.2.3	Connected versus local mode	445
8.3	Command and script invocation	445
8.4	wsadmin management objects	447

8.4.1 Help.....	448
8.4.2 AdminControl	449
8.4.3 AdminConfig	449
8.4.4 AdminApp	450
8.4.5 AdminTask	450
8.5 Managing WebSphere using script libraries	452
8.5.1 Invoking script libraries	453
8.5.2 Displaying help for script libraries	455
8.5.3 Application script library	455
8.5.4 Resource script library	459
8.5.5 Security script library.....	464
8.5.6 Server script library	465
8.5.7 System management script library	469
8.6 Assistance with scripting	471
8.6.1 Enabling command assistance	471
8.6.2 Building script files using command assist	474
8.7 Example: Using scripts with the job manager.....	476
8.7.1 Introduction	477
8.7.2 Creating the customized script	478
8.7.3 Submitting the job	482
8.7.4 Check the results	484
8.8 Online resources	485
Chapter 9. Accessing databases from WebSphere	487
9.1 JDBC resources	488
9.1.1 JDBC providers and data sources	488
9.1.2 WebSphere support for data sources	489
9.2 Steps in defining access to a database.....	493
9.2.1 Creating an authentication alias	494
9.3 Example: Connecting to an IBM DB2 database	495
9.3.1 Creating the JDBC provider	495
9.3.2 Creating the data source.....	499
9.4 Example: Connecting to an Oracle database	503
9.4.1 Creating the JDBC provider	503
9.4.2 Creating the data source.....	507
9.5 Example: Connecting to an SQL Server database	510
9.5.1 Creating the JDBC provider	510
9.5.2 Creating the data source.....	514
9.6 Example: Connecting to an Informix Dynamic Server database	516
9.6.1 Creating the JDBC provider	516
9.6.2 Creating the data source.....	520
9.7 Configuring connection pooling properties	523
9.7.1 WebSphere Application Server data source properties	527

9.7.2 Extended DB2 data source	531
Chapter 10. Accessing EIS applications from WebSphere	533
10.1 JCA resource adapters	534
10.1.1 WebSphere Application Server JCA support	535
10.2 Resource adapters	536
10.2.1 Installing and configuring resource adapters	537
10.3 Configuring J2C connection factories	541
10.4 Resource authentication	544
10.4.1 Container-managed authentication	545
10.4.2 Component-managed authentication	545
Chapter 11. Monitoring	547
11.1 Overview	548
11.1.1 Monitoring scenarios	549
11.2 Enabling monitoring infrastructures	552
11.2.1 PMI defaults and monitoring settings	552
11.2.2 Enable request metrics	561
11.3 Viewing the monitoring data	567
11.4 Monitoring scenarios	579
11.4.1 Database interactions	580
11.4.2 Threading resources	582
11.4.3 JVM memory usage	586
11.4.4 Request level details	587
11.5 ITCAM for WebSphere	593
11.5.1 Installing the data collector	593
11.5.2 Configuring ITCAM for WebSphere metrics	594
11.5.3 Viewing ITCAM for WebSphere data	599
11.6 Monitoring considerations summary	604
11.6.1 Other tools and considerations	605
11.6.2 Summary of monitoring tips	607
Part 2. Working with applications	609
Chapter 12. Session management	611
12.1 HTTP session management	612
12.2 Session manager configuration	612
12.2.1 Session management properties	613
12.2.2 Accessing session management properties	613
12.3 Session identifiers	615
12.3.1 Choosing a session tracking mechanism	615
12.3.2 Cookies	617
12.3.3 URL rewriting	619
12.4 Local sessions	620

12.5 General properties for session management	622
12.6 Session affinity	624
12.6.1 Session affinity and failover	626
12.7 Persistent session management	628
12.7.1 Enabling database persistence.....	631
12.7.2 Memory-to-memory replication	635
12.7.3 Session management tuning.....	646
12.7.4 Larger DB2 page sizes and database persistence	653
12.7.5 Single and multi-row schemas (database persistence).....	654
12.7.6 Contents written to the persistent store using a database	656
12.8 Invalidating sessions	659
12.8.1 Session listeners.....	660
12.9 Session security	661
12.10 Session performance considerations	663
12.10.1 Session size	663
12.10.2 Reducing persistent store I/O	666
12.10.3 Multirow persistent sessions: Database persistence	666
12.10.4 Managing your session database connection pool	668
12.10.5 Session database tuning.....	669
12.11 Stateful session bean failover	670
12.11.1 Enabling stateful session bean failover.....	670
12.11.2 Stateful session bean failover consideration.....	673
Chapter 13. Understanding class loaders.....	677
13.1 A brief introduction to Java class loaders	678
13.2 WebSphere class loader overview	682
13.2.1 WebSphere extensions class loader.....	683
13.2.2 Application and Web module class loaders	684
13.2.3 Handling JNI code	686
13.3 Configuring WebSphere for class loaders.....	686
13.3.1 Application server class loader policies	686
13.3.2 Class loading/delegation mode.....	690
13.3.3 Shared libraries	692
13.4 Class loader viewer	693
13.5 Learning class loaders by example.....	694
13.5.1 Step 1: Simple Web module packaging	695
13.5.2 Step 2: Adding an EJB module and utility jar	698
13.5.3 Step 3: Changing the WAR class loader delegation mode	700
13.5.4 Step 4: Sharing utility JARs using shared libraries	702
Chapter 14. Packaging applications for deployment.....	709
14.1 JEE 5 EAR files.....	710
14.1.1 Development tools	711

14.1.2 Working with deployment descriptors	712
14.2 EJB 3.0 modules	713
14.3 JPA persistence units	717
14.3.1 JPA access intents	718
14.4 Resource adapters	719
14.5 Web modules	720
14.5.1 WebSphere extensions to Web modules	721
14.5.2 File serving	722
14.5.3 Web application auto reload	723
14.5.4 Serve servlets by class name	724
14.5.5 Default error page	724
14.5.6 Directory browsing	724
14.5.7 Pre-compile JSPs	724
14.5.8 Automatic HTTP request and response encoding	724
14.6 Example: Packaging an application	725
14.6.1 Configuring Web module extensions	728
14.7 Exporting to an EAR file	730
14.8 WebSphere Enhanced EAR	731
14.8.1 Configuring a WebSphere Enhanced EAR	733
14.9 Packaging recommendations	745
14.10 Business-level applications	745
14.10.1 Example: Creating a business-level application	749
Chapter 15. Deploying applications	753
15.1 Preparing the environment	755
15.1.1 Creating the ITSO Bank DB2 database	755
15.1.2 Creating an environment variable	756
15.1.3 Creating the ITSO Bank application server	756
15.1.4 Defining the ITSO Bank virtual host	761
15.1.5 Creating the virtual host for IBM HTTP Server and Apache	762
15.1.6 Creating a DB2 JDBC provider and data source	764
15.2 Deploying the application	772
15.3 Deploying application clients	778
15.3.1 Defining application client bindings	780
15.3.2 Launching the J2EE client	781
15.4 Updating applications	783
15.4.1 Replacing an entire application EAR file	784
15.4.2 Replacing or adding an application module	785
15.4.3 Replacing or adding single files in an application or module	786
15.4.4 Removing application content	787
15.4.5 Performing multiple updates to an application or module	787
15.4.6 Rolling out application updates to a cluster	790
15.4.7 Hot deployment and dynamic reloading	795

Related publications	797
IBM Redbooks publications	797
Online resources	797
How to get IBM Redbooks publications	800
Help from IBM	800

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

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

AIX®	IMS™	Redbooks (logo)  ®
Build Forge®	Informix®	System z®
CICS®	MVS™	Tivoli®
DataPower®	POWER®	WebSphere®
DB2®	RACF®	z/OS®
developerWorks®	Rational®	zSeries®
i5/OS®	Redbooks®	
IBM®	Redpapers™	

The following terms are trademarks of other companies:

Adobe, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Adobe, and Portable Document Format (PDF) are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

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

ACS, Interchange, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

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

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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

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

Preface

This IBM® Redbooks® publication provides system administrators and developers with the knowledge to configure a WebSphere® Application Server V7 runtime environment, to package and deploy applications, and to perform ongoing management of the WebSphere environment.

As one in a series of IBM Redbooks publications and Redpapers™ publications for V7, the entire series is designed to give you in-depth information about key WebSphere Application Server features. In this book, we provide a detailed exploration of the WebSphere Application Server V7 runtime administration process.

This book includes configuration and administration information for WebSphere Application Server V7 and WebSphere Application Server Network Deployment V7 on distributed platforms and WebSphere Application Server for z/OS® V7.

The following publications are considered prerequisites to this book:

- ▶ *WebSphere Application Server V7.0: Technical Overview*, REDP-4482
- ▶ *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708

The team who wrote this book

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

Carla Sadtler is a Consulting IT Specialist at the ITSO, Raleigh Center. She writes extensively about WebSphere products and solutions. Before joining the ITSO in 1985, Carla worked in the Raleigh branch office as a Program Support Representative, supporting MVS™ customers. She holds a degree in mathematics from the University of North Carolina at Greensboro.



Fabio Albertoni is a Senior IT Specialist working in Integrated Technology Delivery SSO, in Hortolandia, Brazil. He has twelve years of experience in the IT and banking industries. He has spent the last eight years developing and implementing integrated solutions using WebSphere Application Server and MQ-Series. He holds a degree in Data Processing from FATEC University of Ourinhos and a Masters degree in Computer Engineering from Instituto de Pesquisas Tecnologicas of Sao Paulo, Brazil.



Leonard Blunt is a Senior I/T Specialist working in ASEAN software lab services, based in Singapore. Leonard has a history in middleware architecture design and development, with emphasis on multi-channel e-business applications and existing integrations. Leonard's origins are in building application middleware architectures, with a focus on rapid application development through product integration and the generation of code. Leonard is experienced in implementing J2EE/Java™ and SOA solutions and is passionate about producing robust hardened software that, from its inception, incorporates performance, monitoring, and security. Leonard has been working with WebSphere application Server since 2003, and he graduated from Wollongong University in New South Wales Australia with a Bachelor of Engineering (Computer) in 1999.



Michael Connolly is an IT Consultant at the ITSO, Poughkeepsie Center. He has more than 30 years of IBM software development experience in both distributed systems and the mainframe zSeries®. He holds a BA in Humanities from Villanova University. His areas of expertise include TCP/IP communications, UNIX® System Services, and WebSphere Application Server for z/OS.



Stefan Kwiatkowski is a Senior IT Specialist working in the UK. He has seventeen years of experience in the IT industry with particular focus in the defence industry. He has first hand experience of installing and integrating WebSphere products within customer environments and in his current role with Premium Support specializes in serviceability and technical problem solving. He has experience in other IBM products, including WebSphere Application Server (V4, V5, V6), WebSphere Portal, Tivoli® ITCAM and Rational® Build Forge®. He holds a degree in Electronic System Engineering from Kingston University in the UK. He is certified in WebSphere Application Server v6.1.



Thayaparan Shanmugaratnam is an Instructor with WebSphere Education in Canada. He has several years of experience in relational database, Java development tools, and middleware infrastructure. He holds a Software Engineering degree from the University of Ottawa. His areas of expertise include infrastructure architecture design, implementation, problem determination, and performance tuning on IBM WebSphere Application Server, and IBM WebSphere MQ.



Henrik Sjostrand is an IBM Certified Senior IT Specialist who has worked for IBM Sweden for 15 years. He is currently working in a technical sales role in the WebSphere team in IBM Software Group, Sweden. For the last nine years, he has focused on Java application development and WebSphere Application Server architecture, deployment, performance tuning, and troubleshooting. He is certified in WebSphere Application Server V4, V5.x, and V6.x. Recently he also started working with Web 2.0 technologies and the new IBM WebSphere sMash product. Prior to this book he has authored five Redbooks publications on various topics. Henrik holds a Master of Science in Electrical Engineering from Chalmers University of Technology in Gothenburg, Sweden, where he lives.



Saori Tanikawa is a Technical Engineer with Mizuho Information & Research Institute, Inc. in Japan. Her 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, Tivoli Access Manager and Tivoli Identity Manager. She has written guides for WebSphere Application Server and is published in developerWorks® in Japan. She is certified in WebSphere Application Server V6.1.

Margaret Ticknor is an IT Support Specialist at the IBM ITSO Center in Raleigh. She manages the production and testing environments for the development of IBM Redbooks publications in Raleigh. Prior to joining the ITSO in 1997, Margaret worked in Endicott on the WW VM platform, supporting internal VM customers. Margaret attended the Computer Science program at State University of New York at Binghamton.



Joerg-Ulrich Veser is an IT Specialist working since 2006 in the pre-Sales support for WebSphere on z/OS in Germany. His areas of expertise include infrastructure architecture design, implementation, problem determination, high availability, and security on WebSphere products for z/OS. He holds a degree in Computer Science from the University of Cooperative Education in Mannheim (Germany).

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

Rich Conway
International Technical Support Organization, Poughkeepsie Center

Jim Knutson
IBM Austin

Don Bagwell
IBM US

Raymond Kwok
IBM Canada

William Alward
IBM US

Sreenivasa Paidi
IBM US

Charles Lewis,
NSi Technology

Thanks to the authors of the previous editions of this book, *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304, published in November 2006:

Carla Sadtler, Fabio Albertoni, Bernardo Fagalde, Thiago Kleinubing, Henrik Sjostrand, Ken Worland, Lars Bek Laursen, Martin Phillips, Martin Smithson, Kwan-Ming Wan

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance

and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

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

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/pages/IBM-Redbooks/178023492563?ref=ts>

- ▶ Follow us on twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

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

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-7615-01
for WebSphere Application Server V7 Administration and Configuration Guide
as created or updated on June 11, 2013.

March 2010, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described here.

New information

- ▶ 9.6, “Example: Connecting to an Informix Dynamic Server database” on page 516



Part 1

Basic administration and configuration techniques



System management: A technical overview

In this chapter we provide a technical overview of the system management functionality of WebSphere Application Server. This information can help you understand how system administration occurs. It is particularly useful in a multi-server environment to understand the distributed administration and synchronization topics.

This chapter assumes that you are familiar with the concepts in *WebSphere Application Server V7.0: Technical Overview*, REDP-4482.

We cover the following topics:

- ▶ “System management overview” on page 4
- ▶ “System management in a standalone server environment” on page 8
- ▶ “System management of multiple standalone servers” on page 9
- ▶ “System management in a distributed server environment” on page 12
- ▶ “Management of distributed and standalone servers” on page 32
- ▶ “Java Management Extensions (JMX)” on page 35
- ▶ “Centralized Installation Manager” on page 37
- ▶ “IBM Support Assistant V4” on page 37

1.1 System management overview

At first glance, system management concepts in WebSphere Application Server might seem complex. However, the fact that the system management architecture is based on Java Management Extensions (JMX), and the fact that WebSphere Application Server provides easy-to-use administration tools, both make it fairly simple to use and understand.

1.1.1 Terminology

There are differences in how WebSphere Application Server handles administration, depending on the environment that you have set up. As you go through this book, you can see the following terms used:

- ▶ *Standalone server environment* refers to a single server that is not managed as part of a cell. (The server has not been “federated” to the cell). With the Base and Express packages, this option is your only option. You can also create a standalone server environment with the Network Deployment package.
- ▶ *Distributed server environment* refers to the situation where you have multiple servers managed from a single deployment manager in the cell. We also refer to these as *managed servers*. This is only valid with the Network Deployment package.
- ▶ *Application server* refers to the process that provides the functions that are required to support and host user applications. An application server can be a *standalone application server*, a *distributed application server* managed by a deployment manager, or an *unfederated application server* that is managed from an administrative agent.
- ▶ *Managed processes* refer to the deployment manager, nodes (node agents), and application servers.
- ▶ *Flexible management* refers to asynchronous job management through the job manager (available in Network Deployment only) and managing multiple unfederated application servers from an administrative agent (available on all packages).
- ▶ *System* refers to one physical computer.

1.1.2 Directory conventions

Throughout this book, we use the following notations when indicating the location of files and commands:

- ▶ *install_root* is used to denote the installation directory for a product. The default installation directory locations can be found in:
 - Directory conventions

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.installation.factory.doc/info/ae/ae/rins_dircon.html

- ▶ *profile_root* is used to denote the home profile for a directory. This is equivalent to:

install_root/profiles/profile_name

Special instances of *profile_root* are used to denote the profile home for the following processes:

- Deployment manager:

dmgr_profile_root

- Administrative agent:

adminAgnt_profile_root

- Job manager:

jmgr_profile_root

1.1.3 Profiles

To create an application server, node agent, deployment manager, administrative agent, or a job manager, you must first install the WebSphere Application Server core product files and then create their respective profiles. This is illustrated in Figure 1-2.

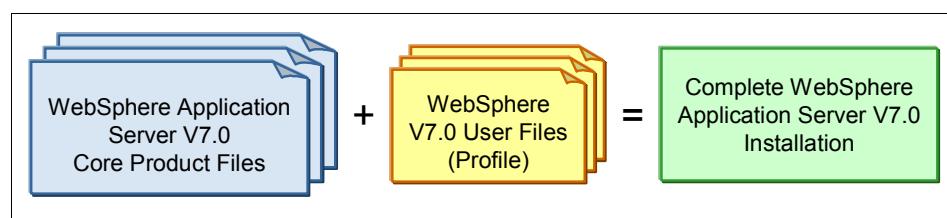


Figure 1-1 Product files and profiles

When a profile is created, the configuration details for the server are stored in a folder that is unique to the profile. You can think of the product files as the runtime component, and the profiles as the input for the runtime.

There are seven profile types that can be created, as shown in Figure 1-2.

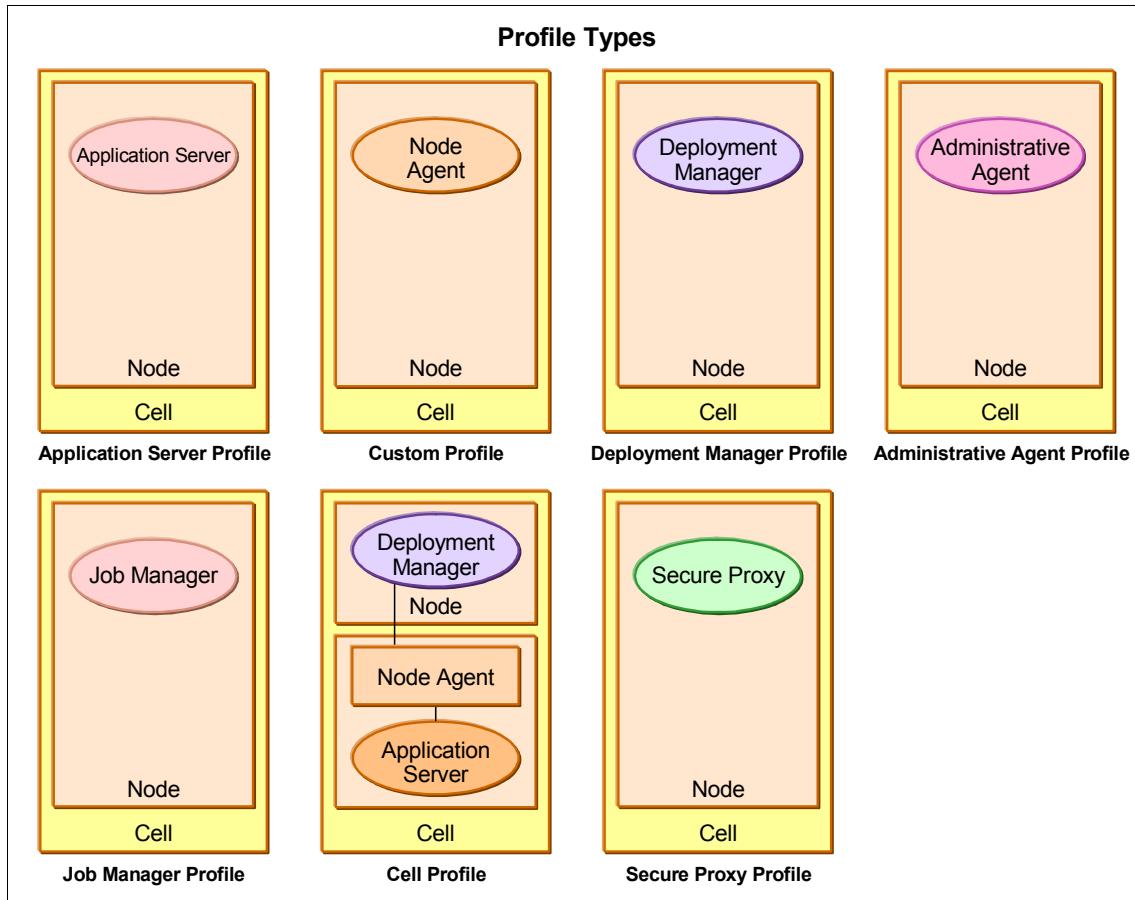


Figure 1-2 Profile types

Notice that a server always belongs to a node, and the node always belongs to a cell. The purpose of the node and cell is to define the administrative domain. The node and cell are logical concepts used to group servers, and to define their scope. Every time a profile is created, a node is created. Therefore, there is a one to one relationship between a profile and a node.

1.1.4 System management tools

WebSphere Application Server V7.0 provides a variety of administrative tools for configuring and managing your runtime environment. The choice of which combination of administrative tools you want to employ depends on the size and complexity of your runtime environment:

- ▶ Integrated Solutions Console, referred to as the *administrative console*:
The administrative console is a browser-based client that uses a Web application running in the Web container to administer WebSphere Application Server.
The administrative console uses Simple Object Access Protocol (SOAP) via HTTP or HTTPS to connect to the admin application in the application server or deployment manager. The admin application connects to the administrative service, which is responsible for modifying the configuration repository using JMX managed beans (MBeans).
- ▶ 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 provides multiple options to connect to the Mbeans: SOAP, RMI, JSR160RMI, IPC and NONE. Note that the RMI connection option is deprecated. The default connection type is SOAP.
The NONE connection type connects directly to the configuration files bypassing the application server. The NONE option allows the administrator to administer the application server even if the application server is not started, or if the embedded HTTP Server, or administrative service is not responding. When using the NONE option, the administrator must be on the same system as the application server.
- ▶ 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. These utilities are called from a command line. They can be used to perform common administrative tasks such as starting and stopping WebSphere Application Server, backing up the configuration, and so on. Command line utilities work on local processes (application servers, nodes, deployment manager) only.

There are multiple levels of administration in WebSphere Application Server:

- ▶ In the WebSphere Application Server Express and Base packages, you can administer stand-alone server instances individually.
- ▶ In the WebSphere Application Server Express and Base packages, you can administer multiple stand-alone server instances on a single system using an administrative agent.
- ▶ In the WebSphere Application Server Network Deployment package, you can administer an entire cell of application servers using a deployment manager.
- ▶ You can administer multiple stand-alone application servers, administrative agents, and deployment managers using a job manager.

1.2 System management in a standalone server environment

A standalone application server provides the necessary capabilities to run J2EE compliant applications. A standalone application server is a good starting point for development and test teams. It can also be used for proof of concept or light-weight applications that do not require intensive system resources.

To create a standalone application server, you must create an Application Server profile. The profile defines the application server, node, and cell.

The application server can be administered using the Web-based administrative console, command line utilities, and wsadmin. The administrative console and wsadmin provide remote administration access. All of the configuration data for an application server, including the installed applications, is stored in a configuration repository created when the profile is created.

Figure 1-3 shows the components of a standalone application server environment.

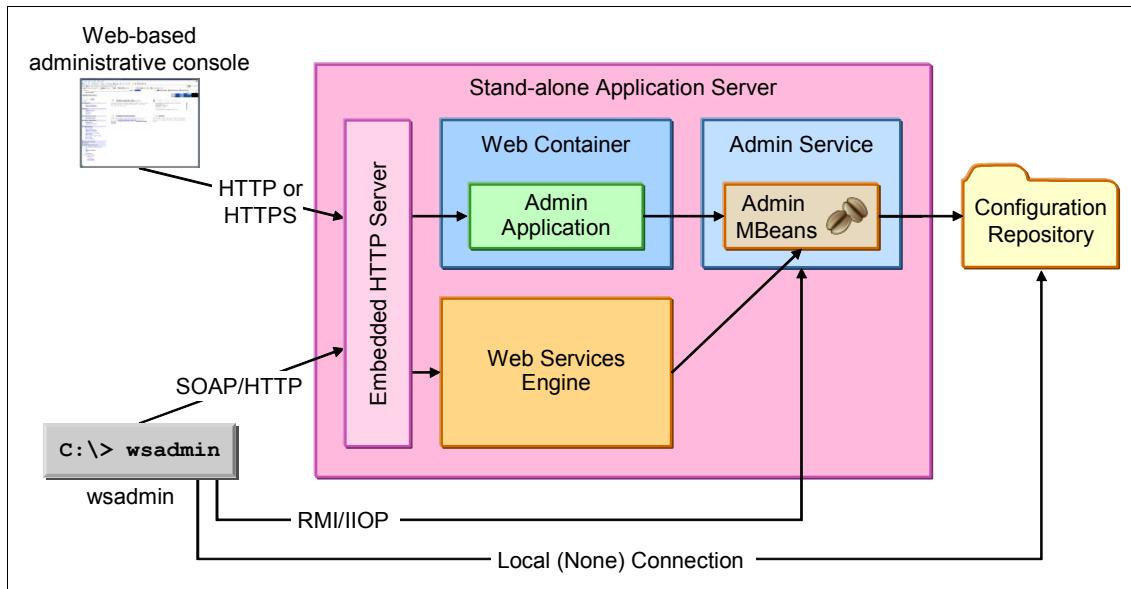


Figure 1-3 Standalone application server system management environment

1.3 System management of multiple standalone servers

Based on their business requirements, an organization can have multiple standalone application servers installed on the same system or on multiple systems. These servers might be used for development environments, unit testing, regression testing, staging environment and so forth.

When there are multiple standalone application servers installed, administration of each application server can become difficult. Each application server runs its own administrative service as well as the administrative console application. An administrator could have to juggle multiple consoles. And as the number of standalone servers increase on a given system, the system resources used for administrative functions increase.

An environment with independently managed application servers is illustrated in Figure 1-4.

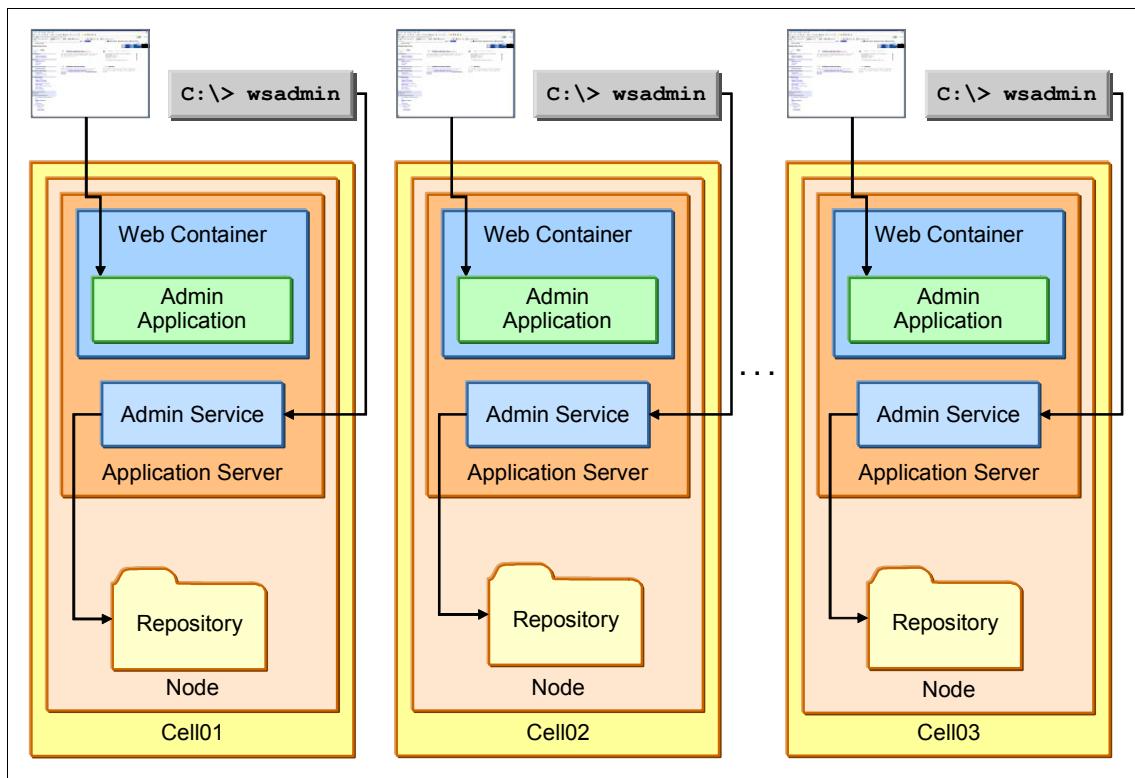


Figure 1-4 Multiple standalone servers with independent administration

With this infrastructure, the administrator must know which administrative port is used for each standalone server. Another drawback to managing standalone servers, is that when the server is not running, the administrative application and administrative service are not available. Therefore, to start the application server, or make any changes, the application servers must be locally started.

The administrative agent reduces the problems related to managing multiple standalone servers by providing a centralized point of control for all of the standalone application servers within a system. The administrative agent is responsible for running the administrative application and administrative service to manage all of the servers on the registered node within the given system. This environment is illustrated in Figure 1-5.

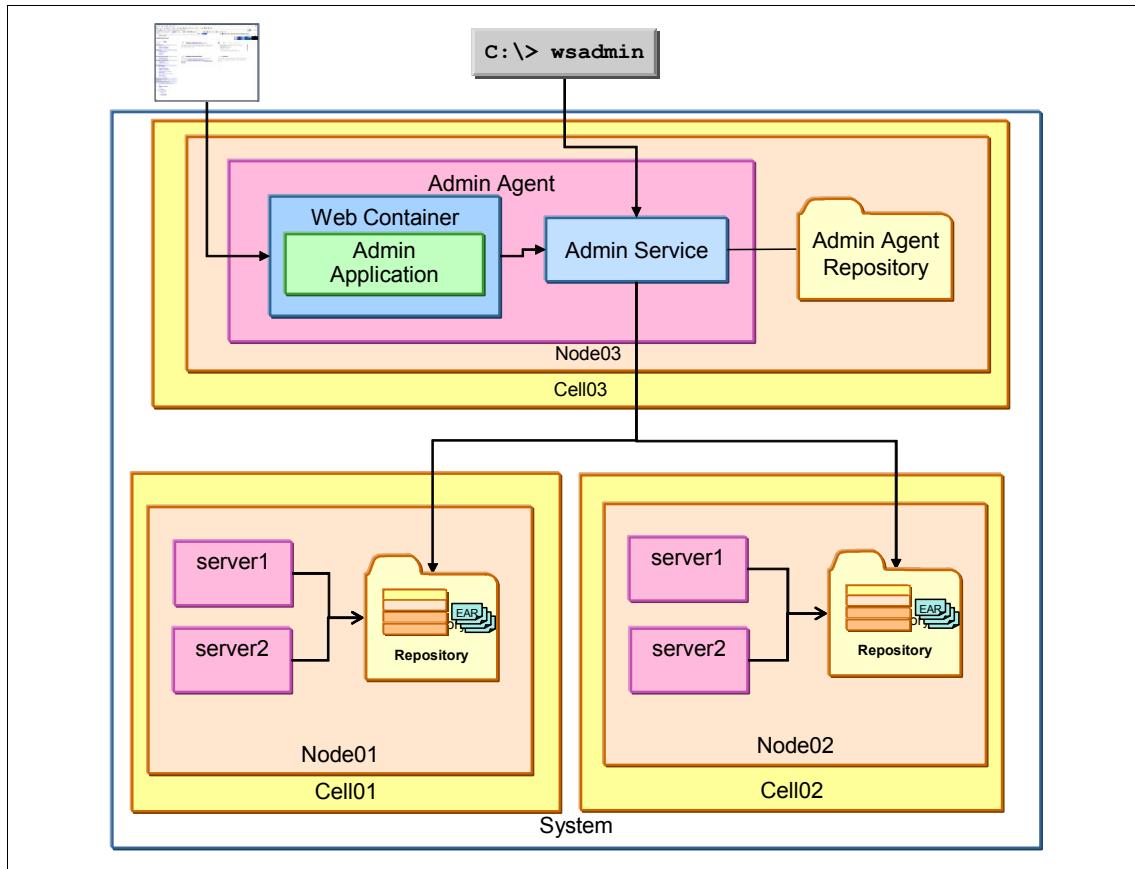


Figure 1-5 Managing multiple standalone servers with flexible management

After a node containing a standalone server is registered with the administrative agent, the administrative console application and administrative service are stopped on the application server. System resources are efficiently utilized because the administrative functionality is not enabled on multiple standalone servers. The standalone application server resources are dedicated to running applications.

Another key feature is the ability to administer an application server when it is not running. The administrative agent modifies the standalone servers configuration repository directly using the administrative service. The administrative agent can also start, stop, and create new servers within the managed node.

New in V7: Without the administrative agent, to create a standalone server, you must create an application server profile, which introduces a new cell, a new node, and an increased number of files. The capability to create multiple application server within an unmanaged node using the administrative agent is a new feature introduced in V7.

The administrative agent, along with multiple standalone servers, is a great starting point for simplifying administration. However, note that features such as failover, workload management, session data replication, and many other features cannot be configured without a distributed server environment.

1.4 System management in a distributed server environment

Although a standalone application server is sufficient to run any J2EE compliant application, there are many runtime capabilities that are not addressed, such as what happens when an application server crashes, or needs to be restarted, or if the demand on the application exceeds the resources available on the hardware. A distributed server environment allows the developer to concentrate on solving the business problem at hand, while leaving infrastructure problems to WebSphere Application Server Network Deployment.

A distributed server environment provides the capabilities to create clusters of application servers. Clustering servers provides work load balancing, session data replication and failover. The application servers are centrally managed using the deployment manager.

To build a distributed server environment, you start by creating a deployment manager profile. The deployment manager is responsible for administering the entire cell. The concept of cells and nodes become very important in a distributed server environment. A deployment manager administers one and only one cell.

After the deployment manager is created, the next step is to create a custom profile. This creates a second cell (defaultCell), a node, and a node agent. At this point, you do not have a functioning application server environment. Figure 1-6 illustrates this temporary stage of the environment.

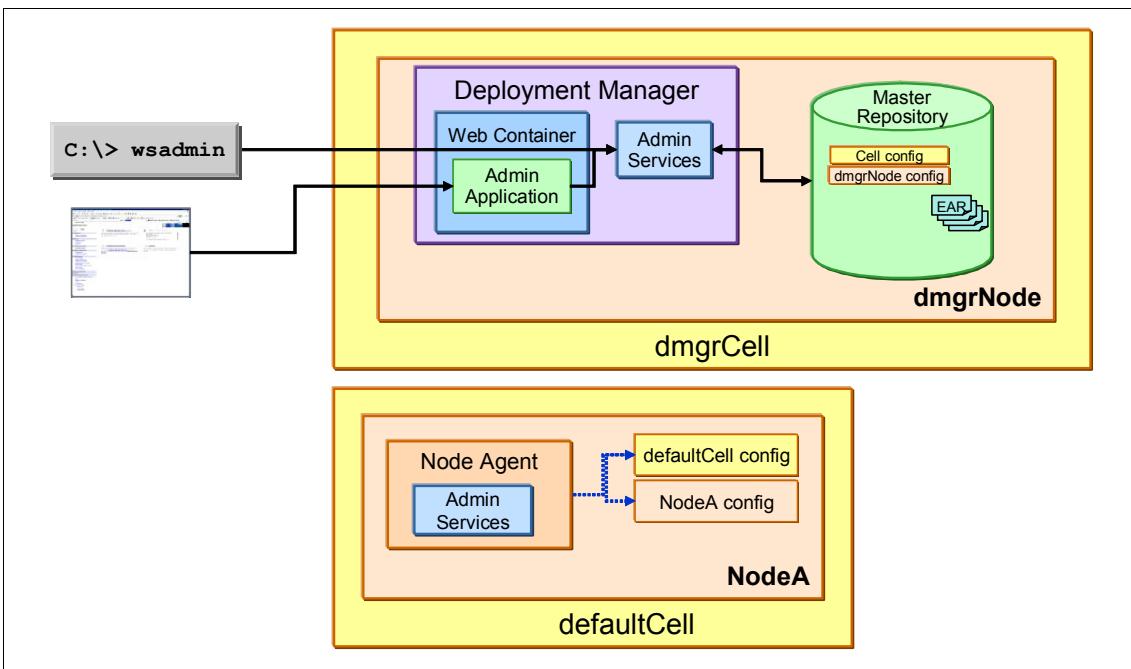


Figure 1-6 A deployment manager and unfederated custom profile

The next step is to federate the node (NodeA in Figure 1-6) to the deployment manager's cell by using the **addNode** command. After being federated, NodeA is no longer part of the defaultCell, but rather it is part of the deployment manager's cell (dmgrCell).

Now that NodeA has been federated, all administration of NodeA is delegated to the deployment manager and new application servers can be created on the node using the administrative tools for the deployment manager.

This environment is shown in Figure 1-7.

For design considerations such as scalability, caching, high availability, load balancing, fail-over, disaster recovery security, and serviceability, see *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708.

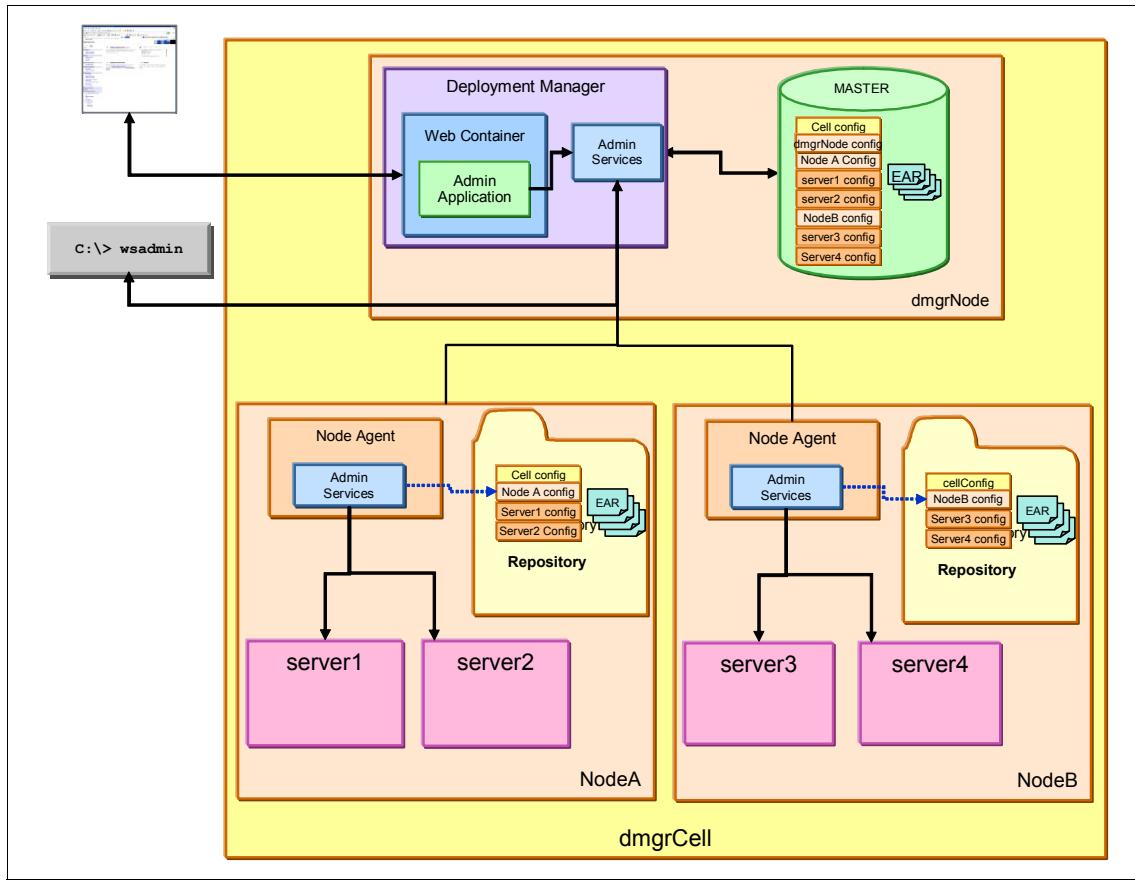


Figure 1-7 Distributed server environment

1.4.1 Centralized changes to configuration and application data

The deployment manager maintains a master repository of all the configuration files. When configuration changes are made at the deployment manager, the changes are first stored in the master repository. Automatic or manual synchronization pushes the changes down to the affected nodes.

Synchronization is discussed further 1.4.5, “File synchronization in distributed server environments” on page 25.

1.4.2 Rules for process startup

When a managed server begins its startup, it sends a discovery request message that allows other processes to discover its existence and establish communication channels with the process. This makes it possible to start the processes in a distributed server environment without following a strict order for startup. For example:

- ▶ A node agent can be running while the deployment manager is not, and vice versa. When the stopped process is started, discovery occurs automatically.
- ▶ The deployment manager can be running while a managed server is not, and vice versa. The execution of a managed server is not dependent on the presence of a running deployment manager. The deployment manager is only required for permanent configuration changes written to the master repository.

The only rule to remember is that the node agent should be started before any application servers on that node. The node agent contains the Location Service Daemon (LSD) in which each application server registers on startup. However, the node agent is purely an administrative agent and is not involved in application serving functions. Each managed server has the data necessary to start itself.

Example discovery scenarios

In this section we describe two typical situations that might occur:

Situation: The node agent is not running and the deployment manager starts:

1. The deployment manager tries to determine if the node agent is running. The process fails.
2. When the node agent is started, it contacts the deployment manager, creates a communication channel, and synchronizes data.

Situation: The node agent starts but no managed servers are started:

1. The node agent knows all about its managed servers and checks whether they are started. If so, it creates communication channels to these processes.
2. When a managed server starts, it checks whether the node agent is started and then creates a communication channel to it.

1.4.3 Distributed process discovery

Deep-dive: The distributed process discovery section provides in-depth knowledge that can be useful when debugging or testing, but it is not necessary when trying to understand the overall architecture.

Figure 1-8 shows an example of the distributed discovery process for a topology containing two nodes that are located on different machines. Note that both node agents in the figure use ports 7272 and 5000. This assumes they reside on separate physical machines. If nodes are located on the same machine, they must be configured to use non-conflicting IP ports. The profile wizard selects non-conflicting ports for you automatically.

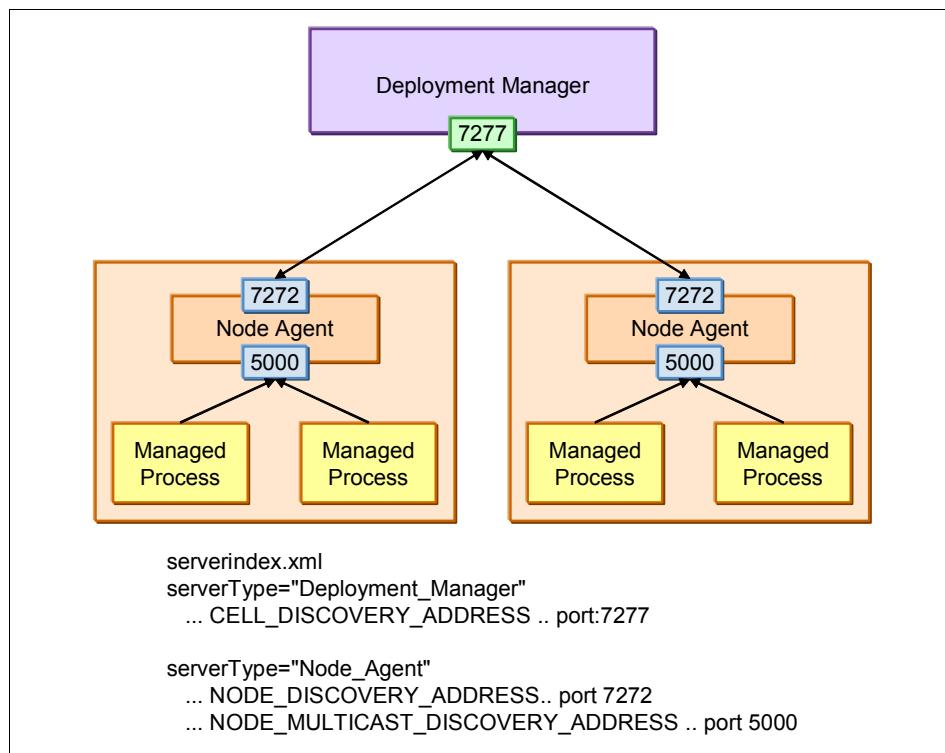


Figure 1-8 Distributed discovery process

Each node agent and deployment manager maintains status and configuration information by using discovery addresses, or *ports*. On startup, processes discover other running components, and create communication channels between them, through the discovery addresses.

During discovery, the following actions occur:

- ▶ The master repository located on the deployment manager installation contains the serverindex.xml file for each node. The deployment manager reads this file on startup to determine the host name and IP port of each node agent's NODE_DISCOVERY_ADDRESS.

The default port for the NODE_DISCOVERY_ADDRESS is 7272. You can verify this by looking at the NODE_AGENT stanza in the serverindex.xml file of each node located at:

`dmgr_profile_root/config/cells/cell_name/nodes/node_name/serverindex.xml`

You can also display this port from the administrative console by selecting **System Administration** → **Node agents**. Select each node agent and expand **Ports** under the Additional Properties section.

- ▶ The copy of the configuration repository located on each node contains the serverindex.xml file for the deployment manager. The node agent reads this file on startup to determine the host name and IP port of the deployment manager's CELL_DISCOVERY_ADDRESS.

The default port for the CELL_DISCOVERY_ADDRESS is port 7277. You can verify this by looking at the DEPLOYMENT_MANAGER stanza in the serverindex.xml file for the deployment manager node located at:

`profile_root/config/cells/cell_name/nodes/dmgr_node_name/serverindex.xml`

You can also display this port from the administrative console by selecting **System Administration** → **Deployment manager**. Expand **Ports** under the Additional Properties section.

- ▶ The copy of the configuration repository located on each node also contains the serverindex.xml file for the node. Each managed server reads this file on startup to determine the host name and IP port of the node agent's NODE_MULTICAST_DISCOVERY_ADDRESS.

A multicast address is used to prevent the usage of a large number of IP ports for managed server to node agent discovery requests. Using multicast, a node agent can listen on a single IP port for any number of local servers.

The default port for the NODE_MULTICAST_DISCOVERY_ADDRESS is 5000. You can verify this by looking at the NODE_AGENT stanza in the serverindex.xml file of the node located at:

`profile_root/config/cells/cell_name/nodes/node_name/serverindex.xml`

You can also display this port from the administrative console by selecting **System Administration** → **Node agents**. Select the node agent and expand **Ports** under the Additional Properties section.

Important: Keep the following considerations in mind:

- ▶ The discovery service uses the `InetAddress.getLocalHost()` call to retrieve the IP address for the local machine's host name. The network configuration of each machine must be configured so that `getLocalHost()` does not return the loopback address (127.0.0.1). It must return the real IP address of the correctly chosen NIC.
- ▶ A multicast address is a logical address. Therefore, it is not bound to a real, physical network interface, and is not the same as the host name (or IP address) of the host on which the node agent is executed.
- ▶ Multicast host addresses must be within a special range (224.0.0.0 to 239.255.255.255) defined by the IP standards and must never be a host name value. The default for WebSphere node agents is 232.133.104.73.

Each server has its own copy of the configuration and application data necessary for startup of the runtime and the installed applications.

1.4.4 Configuration and application data repository

The configuration and application data repository is a collection of files containing all the information necessary to configure and execute servers and their applications. Configuration files are stored in XML format, while application data is stored as EAR files and deployment descriptors.

Repository directory structure

It is important to know that configuration files defining a runtime environment are stored in profile directories. Each node containing a deployment manager, application server, administrative agent, or job manager has its own profile directory under the `install_root/profiles` directory.

Terminology: In the remainder of this book, when we talk about a specific profile directory, located at `install_root/profiles/profile_name`, we refer to it as the `profile_root` directory.

When we are speaking of a profile directory for a specific profile, we use the following terms:

- ▶ Deployment manager profile: `dmgr_profile_root`
- ▶ Administrative agent profile: `adminAgnt_profile_root`
- ▶ Job manager profile: `jmgr_profile_root`

The repository files are arranged in a set of cascading directories under each profile directory structure, with each directory containing a number of files relating to different components of the cell, as shown in Figure 1-9. The repository structure follows the same format, regardless of whether you have a standalone server environment or distributed server environment.

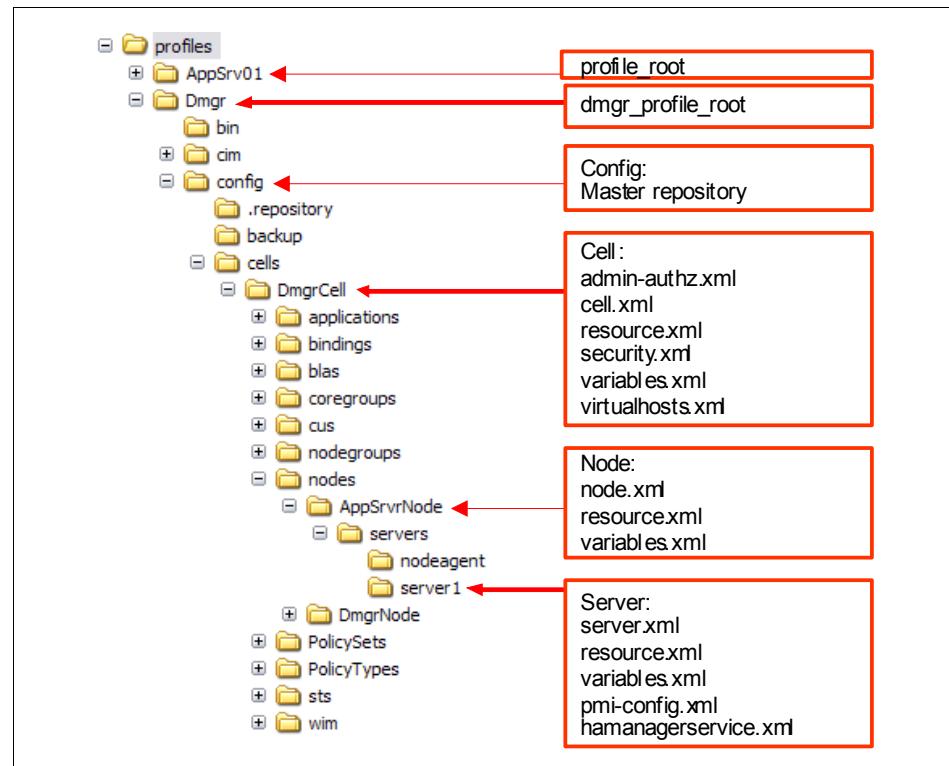


Figure 1-9 Repository directory structure

The `profile_root/config` directory is the root of the repository for each profile. It contains the following directory structure:

► `cells/cell_name/`

This is the root level of the configuration for the cell. The directory contains a number of cell-level configuration files. Depending on the types of resources that have been configured, you might see the following subdirectories:

- `cells/cell_name/applications/` contains one subdirectory for every application that has been deployed within the cell.
- `cells/cell_name/buses/` contains one directory for each service integration bus (bus) defined.
- `cells/cell_name/coregroups/` contains one directory for each core group defined.
- `cells/cell_name/nodegrups/` contains one directory for each node group defined.
- `cells/cell_name/nodes/` contains the configuration settings for all nodes and servers managed as part of this cell. The directory contains one directory per node. Each `cells/cell_name/nodes/<node>` directory contains node-specific configuration files and a server directory that in turn contains one directory per server and node agent on that node.
- `cells/cell_name/clusters/` contains one directory for each of the clusters managed as part of this cell. Each cluster directory contains a single file, `cluster.xml`, which defines the application servers of one or more nodes that are members of the cluster.

The overall structure of the master repository is the same for both a standalone server environment and a distributed server environment. The differences are summarized in the following sections.

In a standalone server environment, the structure has the following characteristics:

- The master repository is held on a single machine. There are no copies of this specific repository on any other node.
- The repository contains a single cell and node.
- There is no node agent because each application server is standalone, so there is no directory for the node agent (`nodeagent`).
- Clusters are not supported. Therefore, the repository tree does not contain the clusters directory or subdirectories.

In a distributed server environment, the structure has the following characteristics:

- ▶ The master repository is held on the node containing the deployment manager. It contains the master copies of the configuration and application data files for all nodes and servers in the cell.
- ▶ Each node also has a local copy of the configuration and application data files from the master repository that are relevant to the node.
- ▶ When changes are made to the configuration in the master repository, those changes need to be synchronized to the configuration files on the nodes.
- ▶ Changes can be made to the configuration files on a node, but the changes are temporary and are overwritten by the next file synchronization from the deployment manager. Permanent changes to the configuration require changes to the file or files in the master repository. Configuration changes made to node repositories are not propagated up to the cell.
- ▶ The applications directory of the master repository contains the application data (binaries and deployment descriptors) for all applications deployed in the cell. The local copy of the applications directory on each node only contains the directories and files for the applications deployed on application servers within that node.

Information about the individual files in each of these directories can be found in the topic, *Configuration Document Descriptions*, at:

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

Variable scoped files

Identically named files that exist at differing levels of the configuration hierarchy are termed *variable scoped* files. There are two uses for variable scoped files:

- ▶ Configuration data contained in a document at one level is logically combined with data from documents at other levels of the configuration hierarchy. In the case of conflicting definitions, the “most specific” value takes precedence. For example, if an identical entry exists in the files at the cell and node level (as with a variable defined in both the cell and node’s variables.xml), the entry at the node level takes precedence.
- ▶ Documents representing data that is not merged but is rather scoped to a specific level of the topology. For example, the namestore.xml document at the cell level contains the cell persistent portion of the name space, while the namestore.xml at the node level contains the node persistent root of the name space.

Application data files

The master repository is also used to store the application binaries (EAR files) and deployment descriptors. The `profile_root/config` directory of the master repository contains the following directory structure used to hold application binaries and deployment settings:

- ▶ `cells/cell_name/applications/`

This directory contains a subdirectory for each application deployed in the cell. Names of the directories match the names of the deployed applications.

Note: The name of the deployed application does not have to match the name of the original EAR file used to install it. Any name can be chosen when deploying a new application, as long as the name is unique across all applications in the cell.

- ▶ `cells/cell_name/applications/app_name.ear`

Each application's directory in the master repository contains:

- A copy of the original EAR, called `app_name.ear`, which does not contain any of the bindings specified during the installation of the application
 - A deployments directory, which contains a single `app_name` directory used to contain the deployed application configuration
- ▶ `cells/cell_name/applications/app_name.ear/deployments/app_name`

The deployment descriptors in this directory contain the bindings specified during application deployment. The deployment directory of each application contains these files:

- `deployment.xml`

This file contains configuration data for the application deployment, including the allocation of application modules to application servers, and the module startup order.

- META-INF/

This directory contains these files:

- application.xml - J2EE standard application deployment descriptor
- ibm-application-bnd.xmi - IBM WebSphere-specific application bindings
- ibm-application-ext.xmi - IBM WebSphere-specific application extensions
- was.policy - Application-specific Java 2 security configuration

This file is optional. If not present, then the policy files defined at the node level apply for the application.

The subdirectories for all application modules (WARs and EJB JARs) are contained in the was.policy along with each module's deployment descriptors. The subdirectories for each module do not contain application binaries (JARs, classes, and JSPs), only deployment descriptors and other configuration files.

The installation of an application onto a WebSphere Application Server application server results in:

- ▶ The storage of the application binaries (EAR) and deployment descriptors within the master repository.
- ▶ The publishing of the application binaries and deployment descriptors to each node that will be hosting the application. These files are stored in the local copy of the repository on each node.

Each node then installs applications ready for execution by exploding the EARs under the *profile_root/installedApps/cell_name/* as follows:

- ▶ *profile_root/installedApps/cell_name/*

This directory contains a subdirectory for each application deployed to the local node. The name of each application's directory reflects the name under which the application is installed, not the name of the original EAR. For example, if an application is called myapp, then the *installedApps/cell_name* directory will contain a *myapp.ear* subdirectory.

- ▶ *profile_root/installedApps/cell_name/app_name.ear/*

Each application-specific directory contains the contents of the original EAR used to install the application.

- The deployment descriptors from the original EAR. These descriptors do not contain any of the bindings specified during application deployment.
- All application binaries (JARs, classes, and JSPs)

Figure 1-10 summarizes how the node's local copy of the repository contains the application's installed deployment descriptors, while the directory under `installedApps` contains the application binaries.

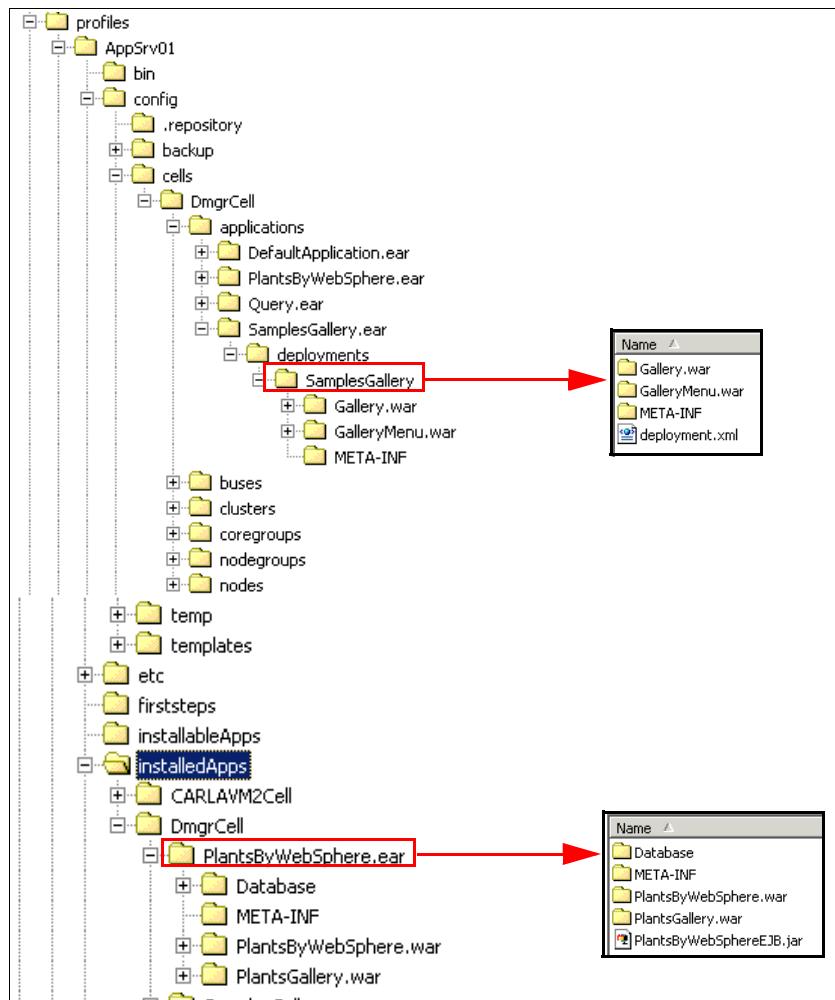


Figure 1-10 Location of application data files

By default, a WebSphere Application Server application server executes an application by performing the following tasks:

1. Loading the application binaries stored under:

profile_root/installedApps/*cell_name*/*app_name.ear*/

You can change this location by altering the Application binaries setting for the enterprise application or by altering the \$(APP_INSTALL_ROOT) variable setting.

2. Configuring the application using the deployment descriptors stored under:

profile_root/config/cells/*cell_name*/applications/*app_name.ear*/deployments/*app_name*

You can change this by modifying the Use configuration information in binary setting for the enterprise application. This is the Use Binary Configuration field on the application installation and update wizards.

By default, the setting is not enabled. Enabling it specifies that you want the application server to use the binding, extensions, and deployment descriptors located in the application EAR file rather than those stored in the deployments directory.

1.4.5 File synchronization in distributed server environments

The file synchronization service is the administrative service responsible for keeping up to date the configuration and application data files that are distributed across the cell. The service runs in the deployment manager and node agents, and ensures that changes made to the master repository are propagated out to the nodes, as necessary. The file transfer system application is used for the synchronization process. File synchronization can be forced from an administration client, or can be scheduled to happen automatically.

During the synchronization operation, the node agent checks with the deployment manager to see if any files that apply to the node have been updated in the master repository. New or updated files are sent to the node, while any deleted files are also deleted from the node.

Synchronization is one-way. The changes are sent from the deployment manager to the node agent. No changes are sent from the node agent back to the deployment manager.

Synchronization scheduling

The scheduling of file synchronization is configured using an administrative client. The available options are:

- ▶ Automatic synchronization:

Synchronization can be made to operate automatically by configuring the file synchronization service of the node agent. These settings allow you to:

- Enable periodic synchronization to occur at a specified time interval
By default, this option is enabled with a time interval of one minute.
- Enable synchronization at server startup

The synchronization occurs before the node agent starts a server. Note that if you start a server using the `startServer` command, this setting has no effect.

- ▶ Explicit/forced synchronization:

Synchronization can be explicitly forced at any time via use of an administrative client.

Tip: In a production environment, the automatic synchronization interval should be increased from the one minute default so that processing and network overhead is reduced.

How files are identified for synchronization

Deep-dive: This section provides in-depth knowledge that can be useful when debugging or testing, but it is not necessary when trying to understand the overall architecture.

When synchronization occurs, WebSphere must be able to identify the files that have changed and therefore need to be synchronized. To do this, WebSphere uses the following scheme:

- ▶ A calculated digest is kept by both the node agent and the deployment manager for each file in the configuration they manage. These digest values are stored in memory. If the digest for a file is recalculated and it does not match the digest stored in memory, this indicates the file has changed.
- ▶ An *epoch* for each folder in the repository and one for the overall repository is also stored in memory. These epochs are used to determine whether any files in the directory have changed. When a configuration file is altered through one of the WebSphere administration interfaces, then the overall repository epoch and the epoch for the folder in which that file resides is modified.

Manually updating a configuration file does not cause the digest to change. Only files updated with administration clients are marked as changed. Manually updating the files is not recommended, but if you do, a forced synchronization will include manually updated files.

- ▶ During configuration synchronization operations, if the repository epoch has changed since the previous synchronize operation, then individual folder epochs are compared. If the epochs for corresponding node and cell directories do not match, then the digests for all files in the directory are recalculated, including that changed file.

Ensuring that manual changes are synchronized

Manually changing configuration files is not recommended. This should only be done as a diagnostic measure or on the rare occasion that you need to modify a configuration setting that is not exposed by the administration clients.

The following topic lists several configuration files that have settings not exposed in the administration clients. In the event that you find it necessary to edit a file manually, this information can help make sure that you do not lose your changes. Refer to *Configuration Document Descriptions* in the Information Center at:

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

Manual editing has several drawbacks, including these:

- ▶ When using **wsadmin** and the administrative console, you have the benefit of a validation process before the changes are applied. With manual editing, you have no such failsafe.
- ▶ Updates made manually are not marked for synchronization and will be lost at the next synchronization process unless you make them in the master repository and manually force synchronization.

If a change to a configuration file is made by editing the file, then the digest for the file is not recalculated, because epochs for the directories continue to match and the synchronization process does not recognize that the files have changed.

However, manual edits of configuration files in the master cell repository can be picked up if the repository is reset so that it re-reads all the files and recalculates all of the digests. You can reset either the master cell repository epoch or the node repository epoch:

- ▶ Resetting the master cell repository causes any manual changes made in the master configuration repository to be replicated to the nodes where the file is applicable.
- ▶ Resetting the node repository causes any manual changes to the local node files to be overwritten by whatever is in the master cell repository, regardless of whether the cell repository was changed or not. Any manual changes in the master repository will be picked up and brought down to the node.

The main difference between cell reset and node reset is that cell reset is likely to impact the entire cell, not just one node.

This holds true for changes to installed applications as well. They are treated the same as other configuration files in the repository. For each installed application, there is an EAR file in the repository and configuration files associated with the deployment of the application.

If you manually change the EAR file and reset the master cell repository, the changed EAR file is replicated out to the nodes where it is configured to be served and is expanded in the appropriate location on that node for the application server to find it. The application on that node is stopped and restarted automatically so that whatever is changed is picked up and made available in the application server.

Important: Manually changing the EAR file is best performed by advanced users. Otherwise, unpredictable results can occur.

If you manually edit one of the deployment configuration files for the application and reset the repository, that change is replicated to the applicable nodes and is picked up the next time the application on that node is restarted.

Resetting the master cell repository

Note: The use of `wsadmin` is covered in Chapter 8, “Administration with scripting” on page 439. The only thing that you might need to know about `wsadmin` to complete these tasks is to start `wsadmin` on the SOAP connector port of the process on which you want to run the commands. The default is to start to port 8879. If the process you are connecting to has a different port number specified, start `wsadmin` with the `-port` argument.

To perform a reset of the master cell repository:

1. Open a command prompt and change to the *dmgr_profile_root/bin* directory and start a **wsadmin** session. Note that the deployment manager must be running. Use the following command:

```
cd dmgr_profile_root\bin  
wsadmin
```

2. Enter the following statements:

```
wsadmin>set config [$AdminControl queryNames  
*:*,type=ConfigRepository,process=dmgr]
```

```
wsadmin>$AdminControl invoke $config refreshRepositoryEpoch
```

You can see a number returned by the `refreshRepositoryEpoch` operation, for example, 1047961605195, as shown in Example 1-1.

Example 1-1 Resetting the master cell repository

```
dmgr_profile_root\bin>wsadmin  
WASX7209I: Connected to process "dmgr" on node DmgrNode using SOAP  
connector; The type of process is: DeploymentManager  
WASX7029I: For help, enter: "$Help help"  
  
wsadmin>set config [$AdminControl queryNames  
*:*,type=ConfigRepository,process=dmgr]  
  
WebSphere:name=repository,process=dmgr,platform=common,node=DmgrNode,ve  
rsion=5.0,type=ConfigRepository,mbeanIdentifier=repository,cell=DmgrCel  
1,spec=1.0  
  
wsadmin>$AdminControl invoke $config refreshRepositoryEpoch  
1237317922687  
wsadmin>
```

This resets the entire cell repository digest set. On the next synchronize operation, all files in the master cell repository will have their digests recalculated. Any manual changes will be replicated to the applicable nodes.

Resetting the node repository

There are multiple ways to reset a node repository for synchronization:

- ▶ In a **wsadmin** session connected to the deployment manager or node agent, enter the following:

```
wsadmin>set config [$AdminControl queryNames  
*:* ,type=ConfigRepository,process=nodeagent]
```

```
wsadmin>$AdminControl invoke $config refreshRepositoryEpoch
```

This resets the node digest set. Any file that does not match what is in the repository is overwritten.

Example 1-2 gives an overview of resetting the node repository.

Example 1-2 Resetting the node repository

```
profile_root\bin>wsadmin -port 8883
```

```
WASX7209I: Connected to process "nodeagent" on node AppSrvrNode using  
SOAP connector; The type of process is: NodeAgent  
WASX7029I: For help, enter: "$Help help"
```

```
wsadmin>set config [$AdminControl queryNames  
*:* ,type=ConfigRepository,process=nodeagent]  
WebSphere:name=repository,process=nodeagent,platform=common,node=AppSrv  
rNode,version=5.0,type=ConfigRepository,mbeanIdentifier=repository,cell  
=DmgrCell
```

```
wsadmin>$AdminControl invoke $config refreshRepositoryEpoch  
1237319314359
```

- ▶ From the deployment manager administrative console, select **System Administration** → **Nodes** to see a list of the nodes in the cell. Notice the Synchronize and Full Resynchronize buttons on the page. The Synchronize button causes a normal synchronize operation with no re-reading of the files. The Full Resynchronize button is the reset and recalculate function. Select the node or nodes to be updated with manual changes, then click the **Full Resynchronize** button.
- ▶ Use the **syncNode** command. This command is a standalone program that runs separately from the node agent. It has no cache of epoch values that could be used for an optimized synchronization, therefore performing a complete synchronization. For this same reason, if you restart a node agent, the very first synchronization that it performs will always be a complete synchronization. Note that this requires the node agent to be stopped.

The **syncNode** command resides in the bin directory of the base install. To use the **syncNode** command, type these commands from the command line:

```
cd profile_root\bin  
syncNode cell_host
```

Example 1-3 shows the use of the **syncNode** command.

Example 1-3 Using the syncNode command

```
profile_root\bin>stopnode  
ADMU0116I: Tool information is being logged in file  
profile_root\logs\nodeagent\stopServer.log  
ADMU0128I: Starting tool with the AppSrv01 profile  
ADMU3100I: Reading configuration for server: nodeagent  
ADMU3201I: Server stop request issued. Waiting for stop status.  
ADMU4000I: Server nodeagent stop completed.
```

```
profile_root\bin>syncnode sysvm2  
ADMU0116I: Tool information is being logged in file  
profile_root\logs\syncNode.log  
ADMU0128I: Starting tool with the AppSrv01 profile  
ADMU0401I: Begin syncNode operation for node AppSrvrNode with  
Deployment Manager sysvm2: 8879  
ADMU0016I: Synchronizing configuration between node and cell.  
ADMU0402I: The configuration for node AppSrvrNode has been synchronized  
with Deployment Manager sysvm2: 8879
```

The repository is flexible in that there is no predefined list of document types that it permits. You can add any file you want. Perhaps you have some unique configuration data that needs to be used on all nodes. You could put it in the config/cells/*cell_name* folder and it would be synchronized to all nodes. If it applies to just one node, you could put it in the folder corresponding to that node and it would be synchronized only to that node. The same applies for any additional documents in a server level folder.

For example, under normal circumstances, all application files are packaged in the EAR file for the application. However, consider a configuration file specific to an application. Any changes to that file would require that you update the EAR file and synchronize the entire application.

One possibility is to put a properties file in the application deployment directory in the master configuration repository, so that it is replicated automatically to all nodes where the application is installed but the entire EAR is not replicated. Then you could have an ExtensionMBean update the properties file in the master repository and normal synchronization would replicate just those changes out to

the nodes without the need to synchronize the whole EAR and restart the application.

1.5 Management of distributed and standalone servers

It is possible to encounter a scenario where there might be multiple distributed environments, each managed by their own deployment manager. With multiple deployment managers, they must be administered individually and there is no way of coordinating management actions between the different distributed environments. Distributed environment administration performance is affected by low latency networks because file synchronization between the deployment manager and node agent are dependent on network communication.

The job manager can be used to administer multiple distributed environments as well as standalone servers. The job manager administers the environment asynchronously using the concept of jobs. Because jobs are submitted asynchronously, a low latency network is sufficient, which can be useful when the environment is distributed over distant geographical areas.

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

To administer a distributed environment, the deployment manager is registered with the job manager. To administer standalone servers, the nodes managed by the administrative agent are registered with the job manager. This relation between the job manager and the environments it can interact with is shown in Figure 1-11.

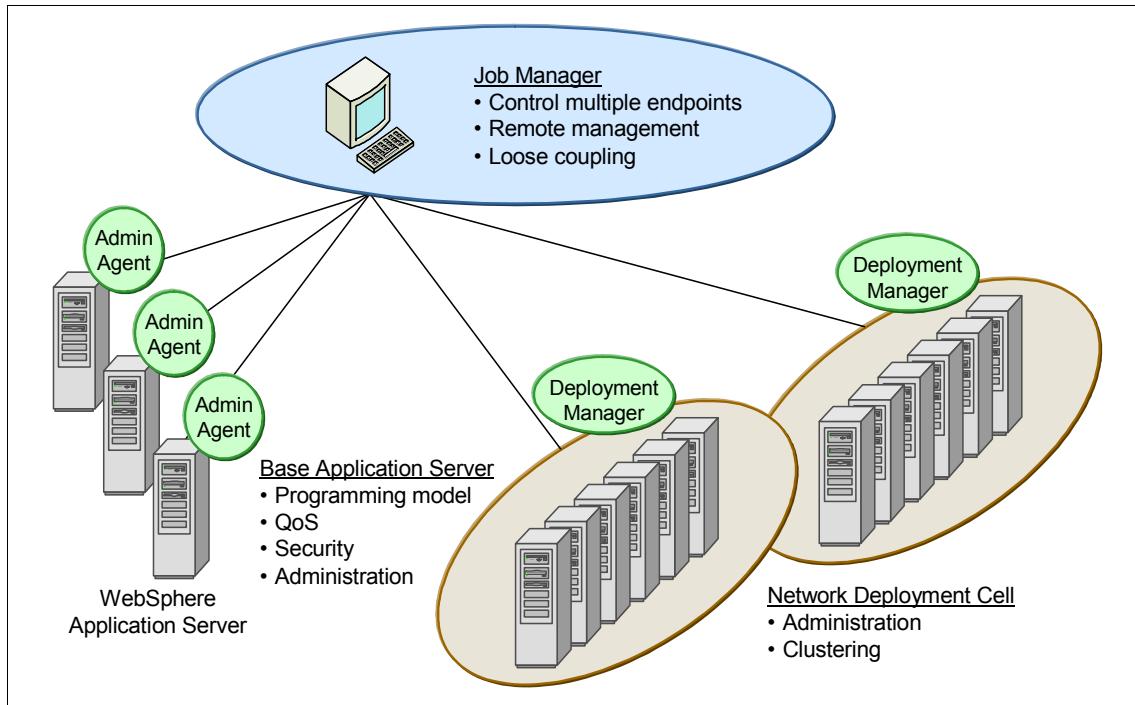


Figure 1-11 Flexible management

The job manager administers the registered environments by submitting jobs that perform tasks, for example:

- ▶ Start and stop servers
- ▶ Create and delete servers
- ▶ Install and uninstall applications
- ▶ Start and stop applications
- ▶ Run wsadmin scripts
- ▶ Distribute files

The job manager has a repository for its own configuration files, which are related to security, administration of job manager, configurations, and so on, however, it does not maintain a master repository the way a deployment manager does. Rather, the job manager allows the administrative agents and deployment managers to continue managing their environment as they would have had they not been registered with the job manager. The job manager simply provides another point of administration. This is illustrated in Figure 1-12.

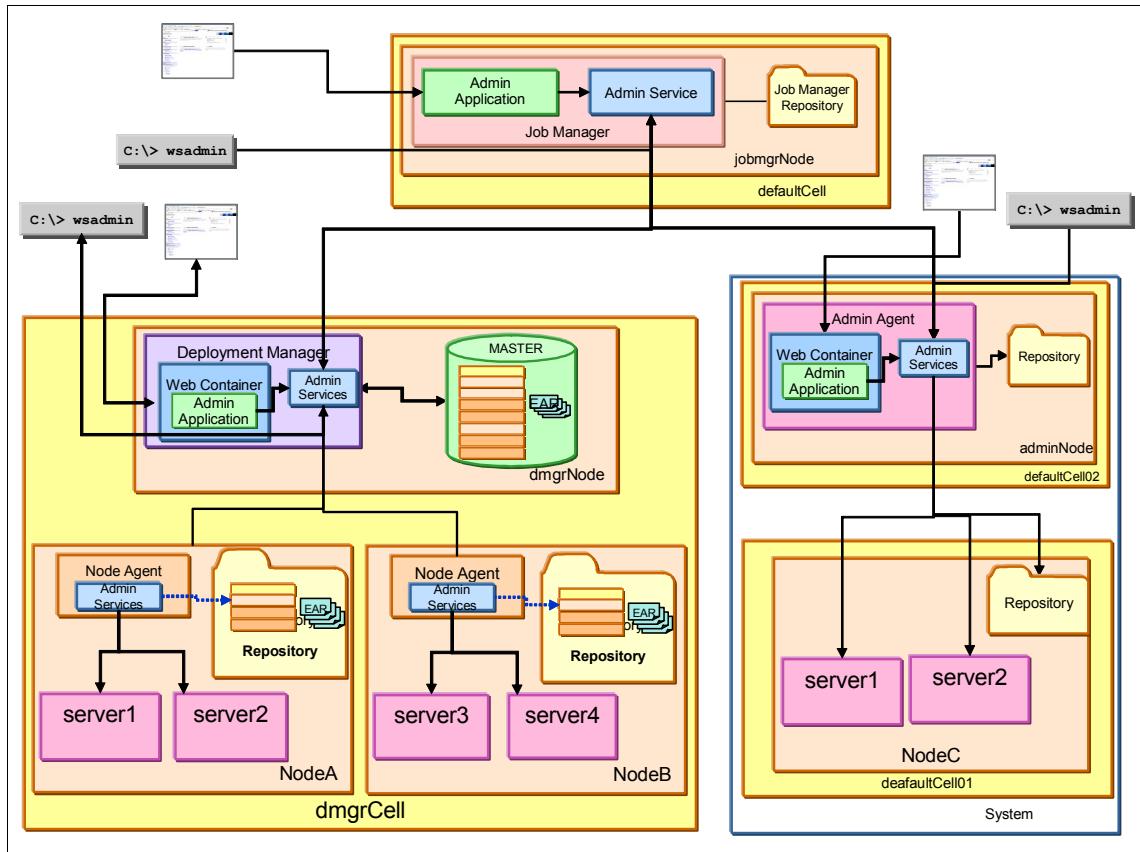


Figure 1-12 Job manager administration environment

The job manager can administer multiple administrative agents and deployment managers. Conversely, each administrative agent and deployment manager can be registered with multiple job managers.

1.6 Java Management Extensions (JMX)

Deep-dive: Extensive knowledge of JMX is not required to administer WebSphere Application Server. However, familiarity with some basic concepts, such as MBeans, can be useful when you are writing scripts for wsadmin.

The system management functionality of WebSphere Application Server is based on the use of Java Management Extensions (JMX). JMX is a framework that provides a standard way of exposing Java resources, for example, application servers, to a system management infrastructure. The JMX framework allows a provider to implement functions, such as listing the configuration settings, and allows users to edit the settings. It also includes a notification layer that can be used by management applications to monitor events, such as the startup of an application server. The use of JMX opens the door to third-party management tool providers. Users of WebSphere are no longer restricted to IBM-supplied management tools.

JMX is a Java specification (JSR-003) that is part of J2SE 1.5. A separate specification defines the J2EE management API (JSR-77) for managing a J2EE conforming application server. WebSphere Application Server provides *managed objects* (*MOs*) as defined in the JSR-77 specification and hence is manageable from third-party management products that delivers J2EE management capabilities.

IBM WebSphere Application Server V6, V6.1 and V7 implements JMX 1.2, while Version 5.x implements JMX 1.1. Due to the evolution of the JMX specification, the serialization format for JMX objects differs between the two specifications, such as javax.management.ObjectName. The WebSphere Application Server JMX runtime has been enhanced to be aware of the version of the client with which it is communicating. It makes appropriate transformations on these incompatible serialized formats so as to allow the different version runtimes to communicate with each other. This makes it possible for a V5.x administrative client to call a V7 deployment manager, node, or server. Similarly, a V7 administrative client can call a V5.x node or server.

For more information about JMX architecture, see:

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

1.6.1 JMX MBeans

Resources are managed by JMX MBeans. These are not EJBs, but simple Java beans that conform to certain design patterns outlined in the JMX specification.

Providers that want to instrument their systems with JMX need to provide a series of MBeans. Each MBean is meant to wrap, or represent, a certain runtime resource. For example, in order to expose an application server as a manageable resource, WebSphere needs to provide an application server MBean.

External applications can interact with the MBeans through the use of JMX connectors and protocol adapters. Connectors are used to connect an agent with a remote JMX-enabled management application. This form of communication involves a connector in the JMX agent and a connector client in the management application.

Each JMX enabled JVM contains an MBean server that registers all the MBeans in the system. It is the MBean server that provides access to all of its registered MBeans. There is only one MBean server per JVM.

WebSphere Application Server provides a number of MBeans, each of which can have different functions and operations available. For example:

- ▶ An application server MBean might expose operations such as start and stop.
- ▶ An application MBean might expose operations such as install and uninstall.

1.6.2 JMX usage scenarios

Some of the more common JMX usage scenarios you might encounter are:

- ▶ Internal product usage:
 - All WebSphere Application Server administration clients use JMX:
 - WebSphere administrative console
 - **wsadmin** scripting client
 - Admin client Java API

- ▶ External programmatic administration:

In general, most external users will not be exposed to the use of JMX. Instead, they will access administration functions through the standard WebSphere Application Server administration clients.

However, external users would need to access JMX in the following scenarios:

 - External programs written to control the WebSphere Application Server runtime and its resources by programmatically accessing the JMX API.
 - Third-party applications that include custom JMX MBeans as part of their deployed code, allowing the applications components and resources to be managed through the JMX API.

1.7 Centralized Installation Manager

Centralized Installation Manager (CIM) provides the capabilities to install, uninstall, and update the WebSphere Application Server environment from a deployment manager. After a deployment manager has been created, CIM can be used to install the WebSphere Application Server binaries on remote servers on the network.

There is no need for an agent to be running on the remote server in order to transfer the binary files. Rather, the CIM uses a native protocol to transfer the binaries over to the remote machine. After the binaries are transferred, custom profiles can be created on the remote server and federated into the deployment manager's cell.

CIM can also be used to download the update installer, interim fixes, and fix packs for WebSphere Application Server V6.1 and greater. When the fix packs have been downloaded using the CIM, the fix packs can be transferred to all remote servers and applied using the update installer. CIM can also be used to uninstall binaries from a remote server.

1.8 IBM Support Assistant V4

IBM Support Assistant V4.0 is a tool provided by IBM at no charge to troubleshoot a WebSphere Application Server environment. IBM Support Assistant is composed of the following components:

- ▶ IBM Support Assistant Workbench
- ▶ IBM Support Assistant Agent Manager

- ▶ IBM Support Assistant Agent

The IBM Support Assistant Workbench is an Eclipse-based client application that provides the following features:

- ▶ Search capabilities
- ▶ Product information
- ▶ Media viewer
- ▶ Data collection
- ▶ Guided troubleshooter

The majority of the workbench features are only available on the local machine where the workbench is installed, which means, all data that is collected must be manually transferred to the system where the workbench is installed. To extend the capabilities built into the workbench to import and export data to remote systems, the IBM Support Assistant manager and agent have to be installed.

The IBM Support Assistant Agent Manager needs to be installed only once within your network. After being installed, the workbench can authenticate itself against the manager, and collect data from remote servers through the use of the IBM Support Assistant Agent.

The IBM Support Assistant Agent is installed on all systems that you are interested in troubleshooting remotely. After installation, the agent must be registered with the IBM Support Assistant agent manager. The agent manager then exposes the agent to the workbench, which is now capable of transferring files to the agent, remotely collecting data from the agent, and also gathering inventory reports.

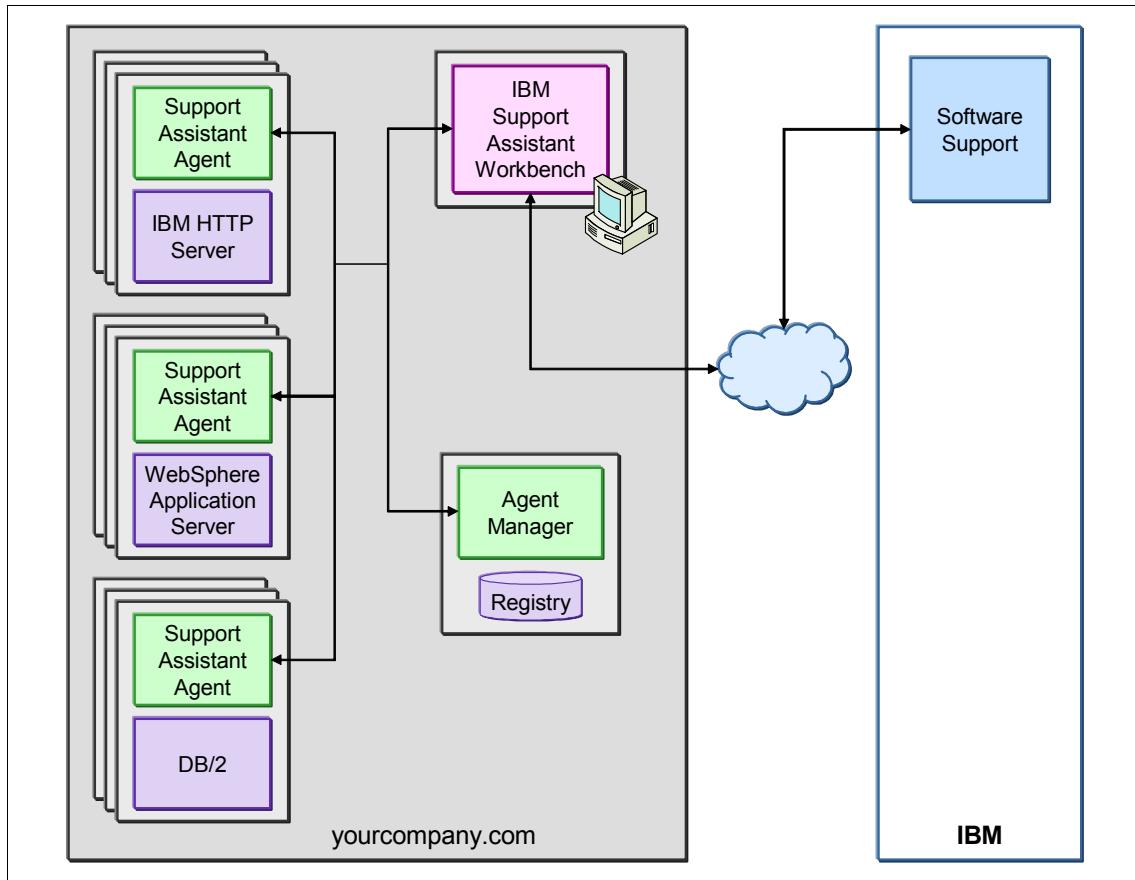


Figure 1-13 IBM Support Assistant

For installation instructions and more details about the IBM Support Assistant, see:

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

If you are new to using IBM Support Assistant, we recommend that you visit the IBM Education assistant tools on IBM Support Assistant for help in getting started:

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.selfassist/selfassist/ISAv41_Task.html



Working with profiles on distributed systems

Installing a WebSphere Application Server environment requires careful planning. A major decision point is the topology for the system. These decisions include, for example, whether you will have a stand-alone server, a distributed managed server environment, and whether you will use the new flexible management options.

Planning for topology design is covered in the IBM Redbooks publication, *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708. That book is designed to help you select a topology and develop a clear idea of what steps are needed to set up your chosen environment. Your options depend on your WebSphere Application Server package. The installation process is well-documented in the installation guide packaged with the product.

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

We cover the following topics:

- ▶ “Types of profiles” on page 42
- ▶ “Planning for profiles” on page 47
- ▶ “Building systems with profiles” on page 47
- ▶ “Managing profiles” on page 105

2.1 Types of profiles

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

With the Base and Express packages, you can only have standalone application servers. Each application server is defined within a single cell and node. The administration console is hosted within the application server and can only connect to that application server. (**New in V7**) You can consolidate administration for multiple standalone servers by registering the node for each application server to an administrative agent.

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

With the Network Deployment package, you have the option of defining multiple application servers with central management capabilities. The administration domain is the cell, consisting of one or more nodes. Each node contains one or more application servers and a node agent that provides an administration point management by the deployment manager.

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

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

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

- ▶ You can create an application server profile, then federate it to the cell. When a node is added to a cell, a node agent is created on the node, and configuration files for the node are added to the master configuration repository for the cell. The deployment manager then assumes responsibility for the configuration of all servers on the node.
- ▶ You can define a custom profile to create an empty node for federation to the cell. After federation, you further configure the node by creating application servers and clusters from the deployment manager administrative console.

With WebSphere Application Server V7.0, the job manager and administrative agent profile types have been introduced to enhance the administration capabilities.

2.1.1 Application server profile

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

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

The primary uses for this type of profile are:

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

2.1.2 Deployment manager profile

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

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

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

After you have the deployment manager, you can:

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

2.1.3 Custom profile

A custom profile is an empty node, intended for federation to a deployment manager. This type of profile is used when you are building a distributed server environment. You use a custom profile as follows:

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

2.1.4 Cell profile

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

2.1.5 Administrative agent profile

(New in V7) The administrative agent is a new profile that provides enhanced management capabilities for stand-alone application servers. This is a new concept introduced with WebSphere Application Server V7.0.

An administrative agent profile is created on the same node as the standalone servers and can manage only servers on that node. The node configuration for each standalone server is totally separate from any other servers on the system, but it can be managed using the administrative console on the administrative agent.

To participate in flexible management, standalone base servers first register themselves with the administrative agent. When a base application server registers with an administrative agent, much of the administrative code that was in the base server is consumed by the administrative agent. This results in a significantly smaller and faster starting base server (Figure 2-1).

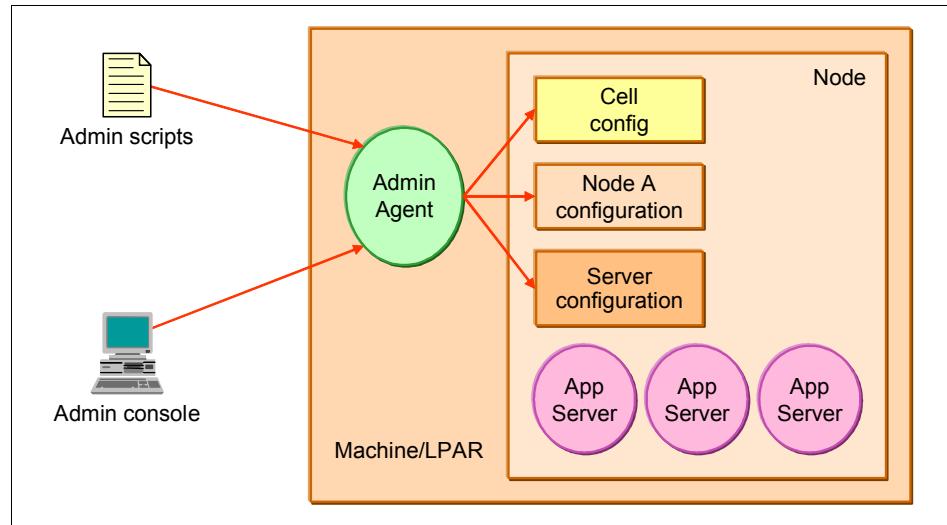


Figure 2-1 High-level overview of a administrative agent profile architecture

2.1.6 Job manager profile

(New in V7) The job manager is a new server type that was added to support flexible management. A job manager is defined by a job manager profile.

To participate in flexible management, a standalone application server first registers itself with the administrative agent. The administrative agent must then register the node for the application server with the job manager. If a deployment manager wants to participate in an environment controlled by a job manager, the deployment manager registers directly with the job manager; no administrative agent is involved in this case.

The main use of the job manager is to queue jobs to application servers in a flexible management environment. These queued jobs are pulled from the job manager by the administrative agent and distributed to the appropriate application server or servers.

Both deployment manager and administrative agents retain autonomy and can be managed without the job manager.

The units of work that are handled by the flexible management environment are known as jobs. The semantics of these jobs are typically straightforward, and the jobs require few parameters. The jobs are processed asynchronously and can have an activation time, expiration time, and a recurrence indicator. You can specify that an e-mail notification be sent upon completion of a job. Additionally, you can view the current status of a job by issuing a status command (Figure 2-2).

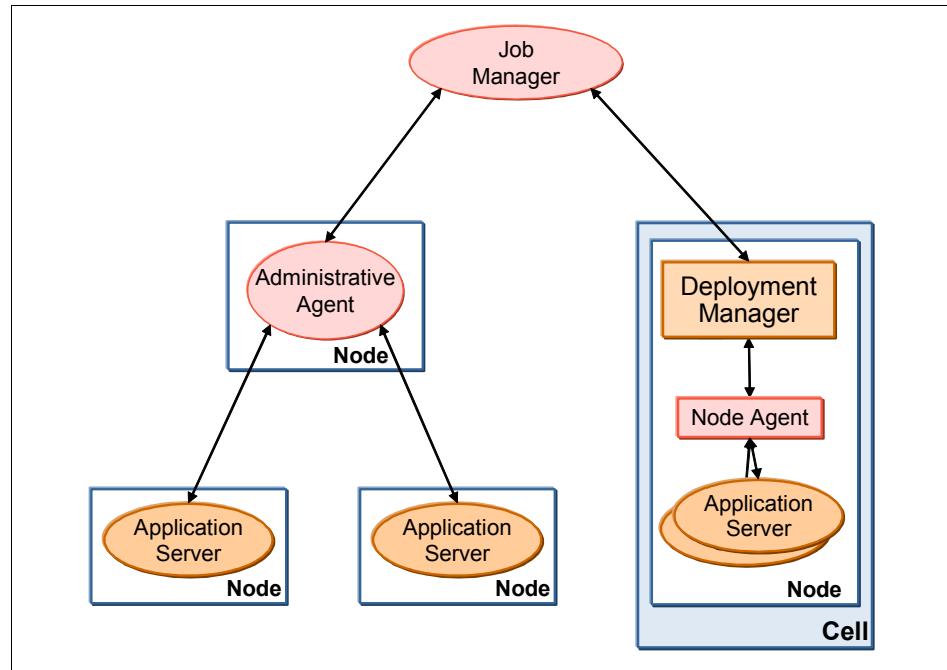


Figure 2-2 High-level overview of a job manager architecture

In Figure 2-2 we see that the administrative agent looks like it communicates directly to the job manager node - in practice the individual application server that is managed by the administrative agent is registered with the job manager directly.

2.1.7 Profile generation

Profiles can be created at any time during or after installation using graphical or command line tools. WebSphere Application Server provides the following profile management tools:

- ▶ The **manageprofiles** command: Command-line interface for profile management functions.
- ▶ Profile Management Tool (PMT): A GUI interface that gathers user input and invokes the **manageprofiles** command line tool to manage the profiles.

2.2 Planning for profiles

Profiles can be created using the PMT or in silent mode using the **manageprofiles** command. Regardless of the method you use, a minimum amount of space must be available in the directory where you create a profile. This minimum requirement is documented in the Information Center at:

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

Profiles grow when applications and associated log files are created, and therefore these increases must be considered at the planning stages.

An error can occur when you do not provide enough space to create a profile. Verify that you have, in addition to the minimum space required for a particular profile, an additional 40 MB of space, which is used for log files and temporary files.

Important: You cannot use the PMT to create profiles for WebSphere Application Server 64-bit installations except on the Linux® for zSeries platform. However, you can use the Profile Management Tool on 64-bit architectures if you use a WebSphere Application Server 32-bit installation.

2.3 Building systems with profiles

This section shows how to use the PMT to create profiles on distributed systems. Creating profiles in silent mode is discussed in 2.4.4, “Creating a profile with the **manageprofiles** command” on page 107.

2.3.1 Starting the PMT

The first steps in creating a profile are common, regardless of the type of profile you are going to create.

Follow these steps to create the profile:

1. Start the PMT using one of the following methods:
 - Windows® only:
From the Start menu, select **Start → Programs → IBM WebSphere → Application Server Network Deployment V7.0 → Profile Management Tool.**
 - For Linux, HP-UX, Solaris and AIX®:
Use the **pmt.sh** command in the *install_root/bin/ProfileManagement* directory.
 - At the end of the installation process using the install wizard, check the box to launch the Profile Management Tool.
2. When you start the wizard, the first window you see is the Welcome window.
Click the **Launch Profile Management Tool** button (Figure 2-3).

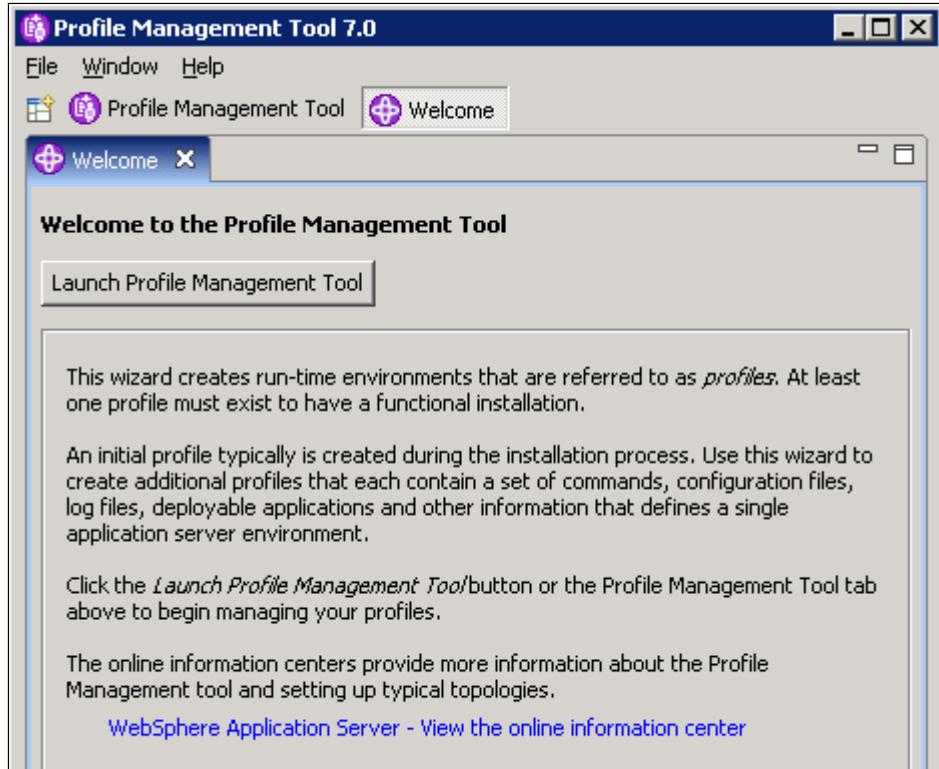


Figure 2-3 PMT Welcome window

3. Next you see a list of existing profiles. Click **Create** to start the profile creation process (Figure 2-4).

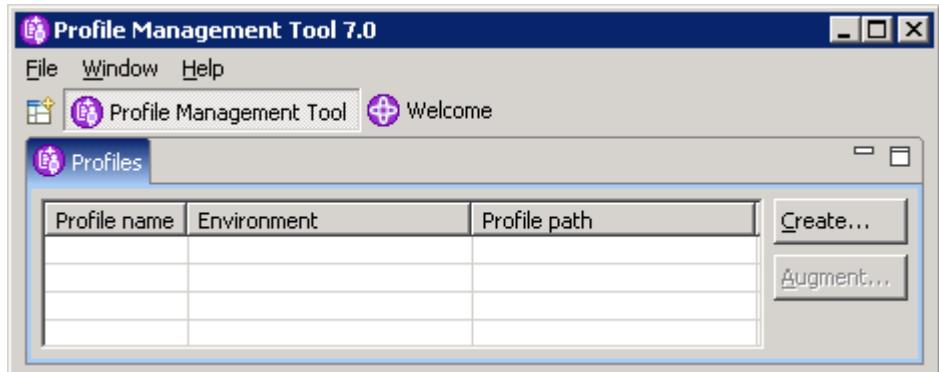


Figure 2-4 PMT List of profiles

2.3.2 Common panels and steps for all profiles

Many of the options that you have when you create a profile are the same, regardless of the type of profile.

Environment selection

During profile creation, you will be asked to select the type of profile to create (Figure 2-5).

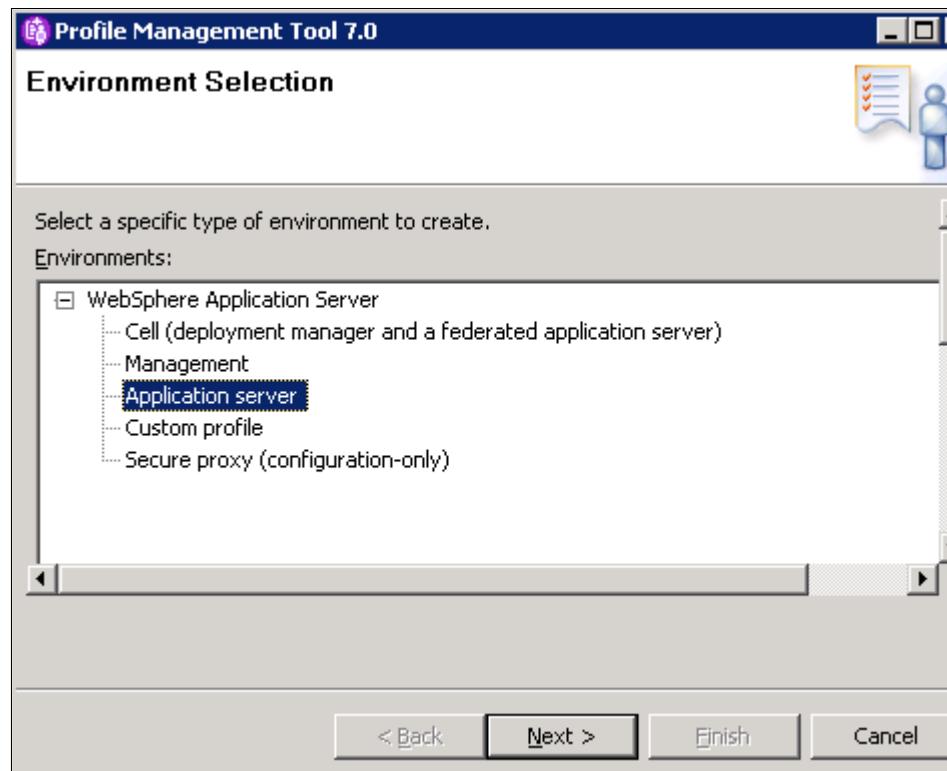


Figure 2-5 Pick Application server

The profile options are listed next. Note that the deployment manager profile is under the Management option, along with the new profile types for flexible management (administrative agent and job manager):

- ▶ Cell (deployment and a federated application server)
- ▶ Management
 - Administrative agent (*New in V7*)
 - Deployment manager
 - Job manager (*New in V7*)

- ▶ Application server
- ▶ Custom profile
- ▶ Secure proxy (configuration-only) (**New in V7**)

Profile creation options

While creating profiles, you are presented with a choice (Figure 2-6) of following the “Typical” path, where a set of default values for most settings will be used, or an “Advanced” path, which lets you specify values for each option.

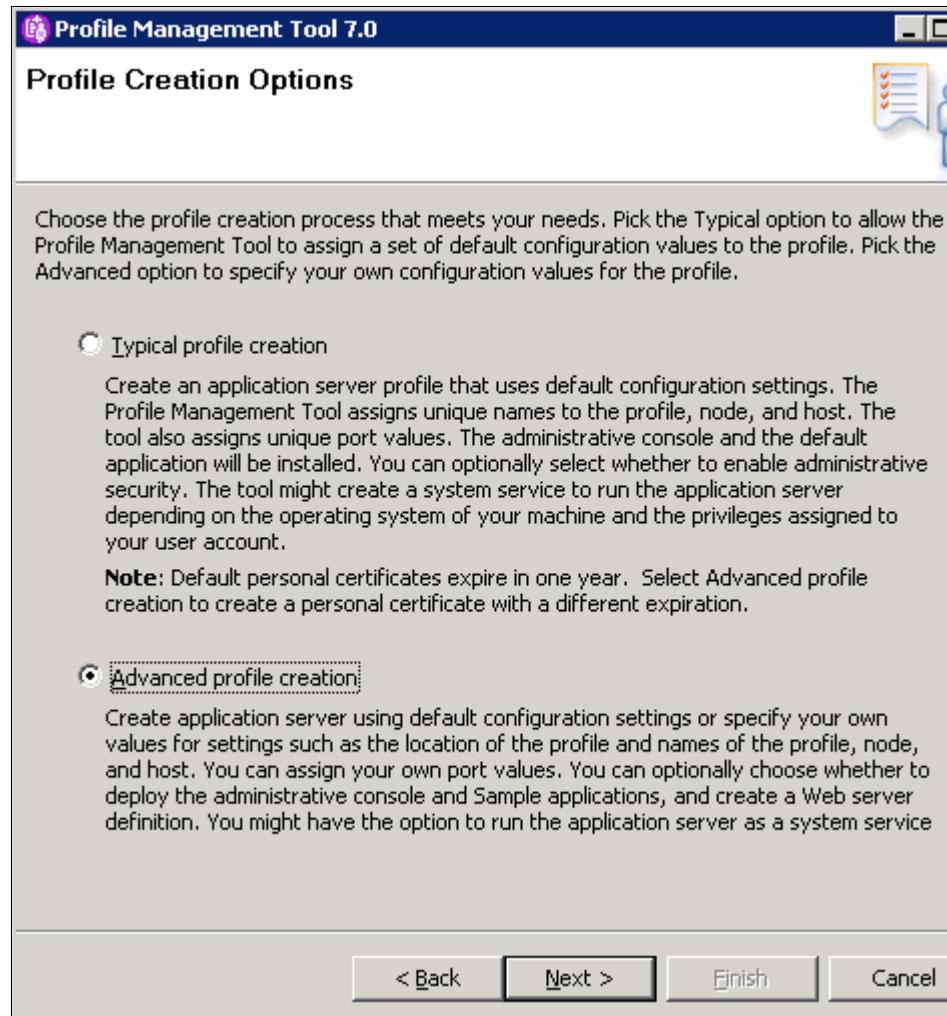


Figure 2-6 Profile creation path (typical vs. advanced option)

The Advanced path is preferred because it gives you additional control over names and settings. An overview of the Advanced profile options is shown in Table 2-1.

Table 2-1 Options available in the typical versus advanced path

Option	Application Server	Deployment manager	Administrative Agent	Job manager	Custom	Cell
Deploy Administrative Console	Yes	Yes	Yes	Yes	No	Yes
Default and Sample Applications	Yes	No	No	No	No	Yes
Profile Name and Location	Yes	Yes	Yes	Yes	Yes	Yes
Node and Host Names	Yes	Yes	Yes	Yes	Yes	Yes
Administrative Security	Yes	Yes	Yes	Yes	Yes (Federation)	Yes
Certificates (part 1 and 2)	Yes	Yes	Yes	Yes	Yes	Yes
Port Assignment	Yes	Yes	Yes	Yes	Yes	Yes (Part 1 and part 2, one for dmgr, other for App Server)
Windows Services (Windows only)	Yes	Yes	Yes	Yes	Yes	Yes
Web Server definition (Parts 1 and 2)	Yes	No	No	No	No	Yes
Summary	Yes	Yes	Yes	Yes	Yes	Yes

Profile name and location (and default profiles)

The wizard asks for a profile name and where you want the profile configuration files stored.

Directory location

By default, profiles are stored in *install_root/profiles/profile_name*. The logs for the process defined by the profile will reside within this directory structure, however, you can easily change this if space is a concern.

Default profile

The first profile that you create on a machine is the default profile. The default profile is the default target for commands that are issued from the bin directory in the product installation root when the -profileName argument is not used.

You can make another profile the default profile when you create that profile by checking **Make this profile** the default on the Profile name and location panel of the Advanced profile creation path. You can also make another profile the default profile using the `manageprofiles` command after you create the profile.

Profile name

The profile name must be unique within the installation and should follow an appropriate naming convention so you can easily identify it by the name it is given. The guidelines are as follows:

- ▶ Double-byte characters are supported.
- ▶ The profile name can be any unique name with the following restrictions.
- ▶ Do not use any of the following characters when naming your profile:
 - Spaces
 - Special characters that are not supported within the name of a directory on your operating system, such as *&?
 - Slashes (/) or (\)

Administrative security

When you create a profile for a process with the administrative functions (basically everything but a custom profile), you have the opportunity to enable administrative security. If you enable security during profile creation, you are asked for a user ID and password that will be added to a file-based user registry with the Administrator role.

We recommend that you enable administrative security. The file-based repository created during profile creation can be federated with other repositories

later to provide a robust user registry for both administrative and application security.

Note: If you are going to create a job manager and register a deployment manager, keep in mind that you cannot register a deployment manager that has security enabled to a job manager that does not. So, you should plan for administrative security across the WebSphere environment.

You can find more information about administrative security in 5.2, “Securing the console” on page 266.

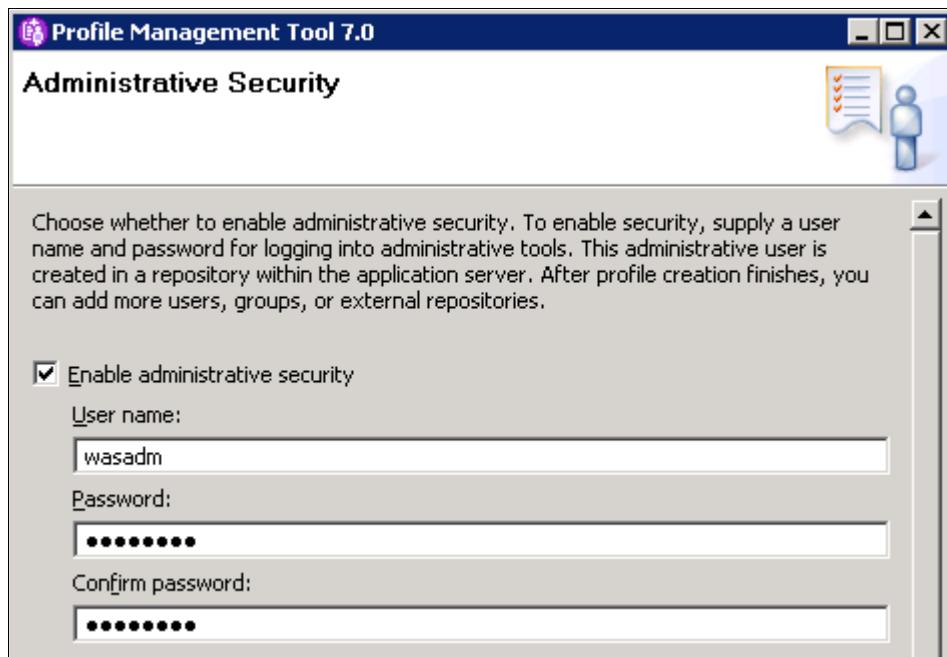


Figure 2-7 Enable administrative security

Certificates

Each profile contains a unique chained certificate signed by a unique long lived root certificate that is created when the profile was created. When a profile is federated to a deployment manager, the signer for the root signing certificate is added to the common truststore for the cell, establishing trust for all certificates signed by that root certificate.

For a full description of the certificates and the keystore password, see:
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/csec_7ssldefault_chainedcert_config.html

Two panels are used during profile creation to manage the import or creation of these certificates.

The first panel (Figure 2-8) allows you to create the certificates or import existing certificates.

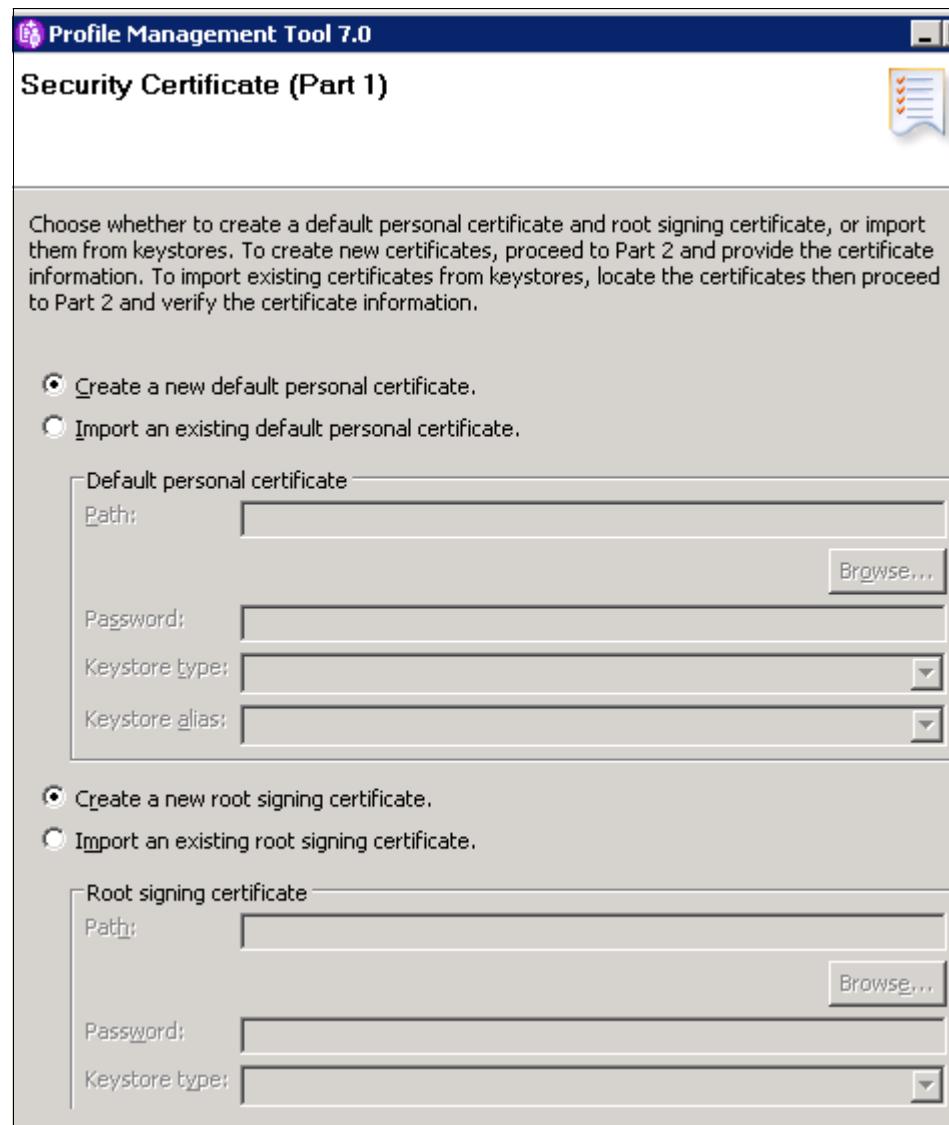


Figure 2-8 Create certificates or import existing personal or root certificates

The second panel (Figure 2-9) is used to modify the certificate information to create new certificates during profile creation. Review the expiration period and provide a new password for the default keystore. The default password is WebAS.

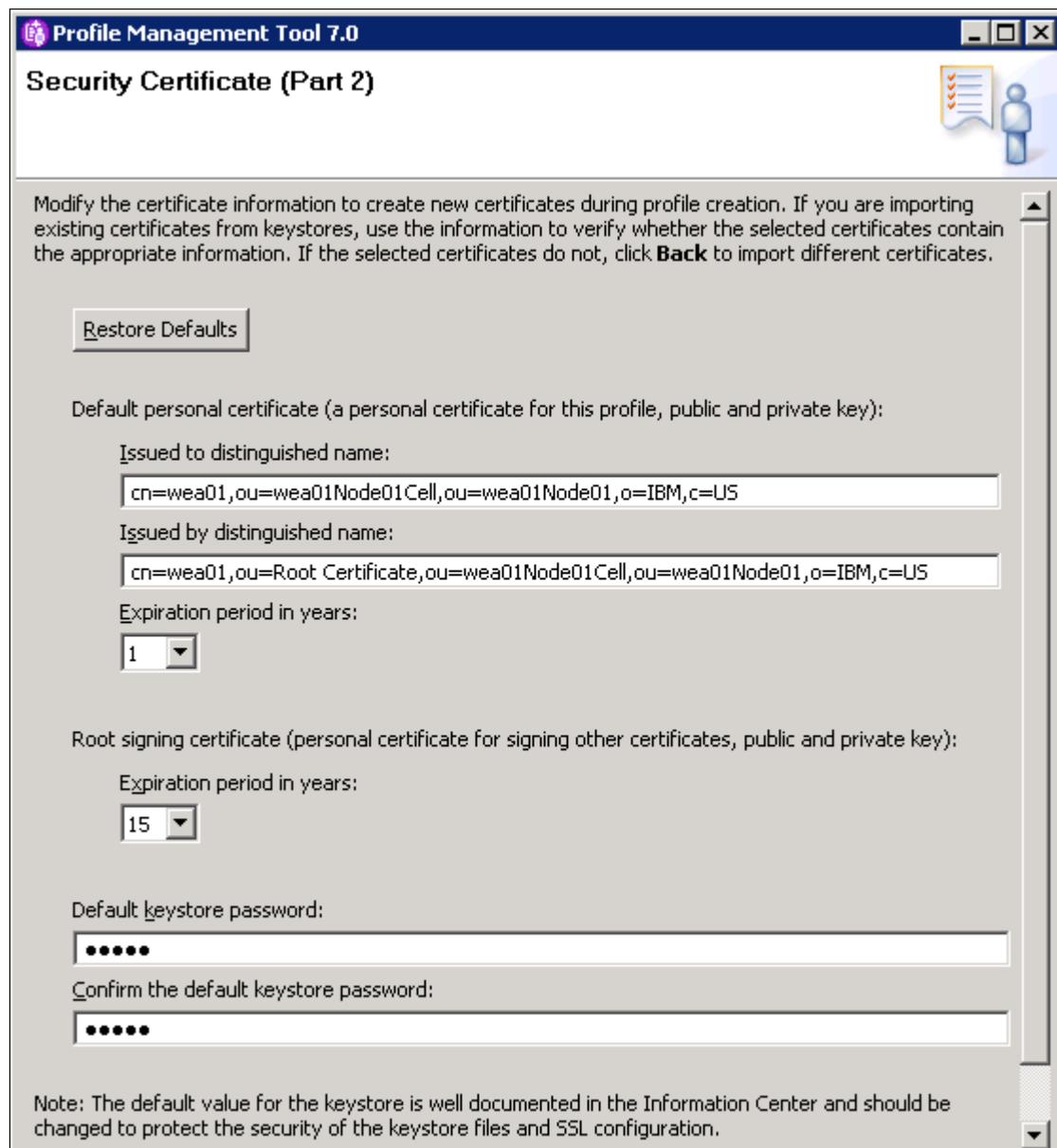


Figure 2-9 Modify certificate information at profile creation time

Port assignments

Every process uses a set of ports at runtime. These ports must be unique to a system. For the default port assignment for the distributed platform, see:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.migration.nd.doc/info/ae/ae/rmig_portnumber.html

The PMT wizard assigns unique port numbers to each profile if multiple profiles are installed in the same system. Careful planning is needed so that there are no port conflicts with other software installed on the same systems.

When you take the Advanced path through the profile wizard, you have three options:

- ▶ Use the default set of port numbers.
- ▶ Use the recommended set of port numbers. These have been selected as unique to the WebSphere installation.
- ▶ Customize the port numbers.

Run as a Windows service

When you create a profile on a Windows system, you have the option of running the application server as a Windows service. This provides you a simple way of automatically starting the server process when the system starts.

If you would like to run the process as a Windows service, check the box and enter the values for the logon and startup type. Note that the window lists the user rights that the user ID you select needs to have. If the user ID does not have these rights, the wizard automatically adds them.

When you take the Typical path through the profile creation wizard, the default is to define the process as a Windows service (Figure 2-10).

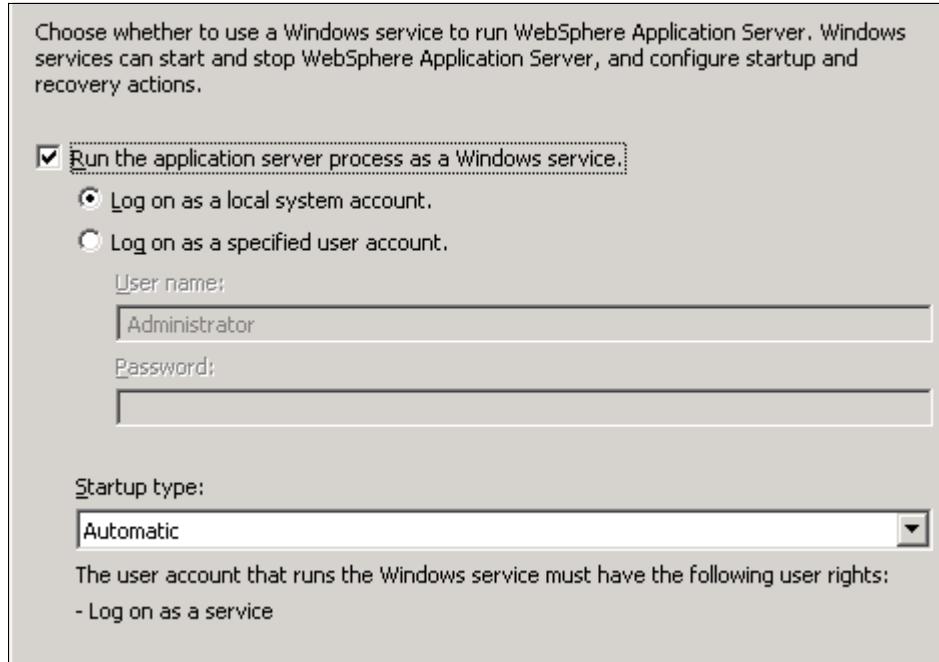


Figure 2-10 Run as a service

If you do not register the process as a Windows service during profile creation, you can do that later using the **WASService** command. For more information about the WASService command, see:

► **WASService command**

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

Verification steps

The profile will be stored in the directory structure you selected. In this IBM Redbooks publication, we refer to this directory as *profile_root*. This is where you can find, among other things, the config directory containing the configuration files, the bin directory for entering commands, and the logs directory where information is recorded.

After you create a new profile, you can take the following steps to verify that the profile is working correctly:

- View the messages produced by the profile creation.

First, note the messages that result from the profile creation (Figure 2-11).

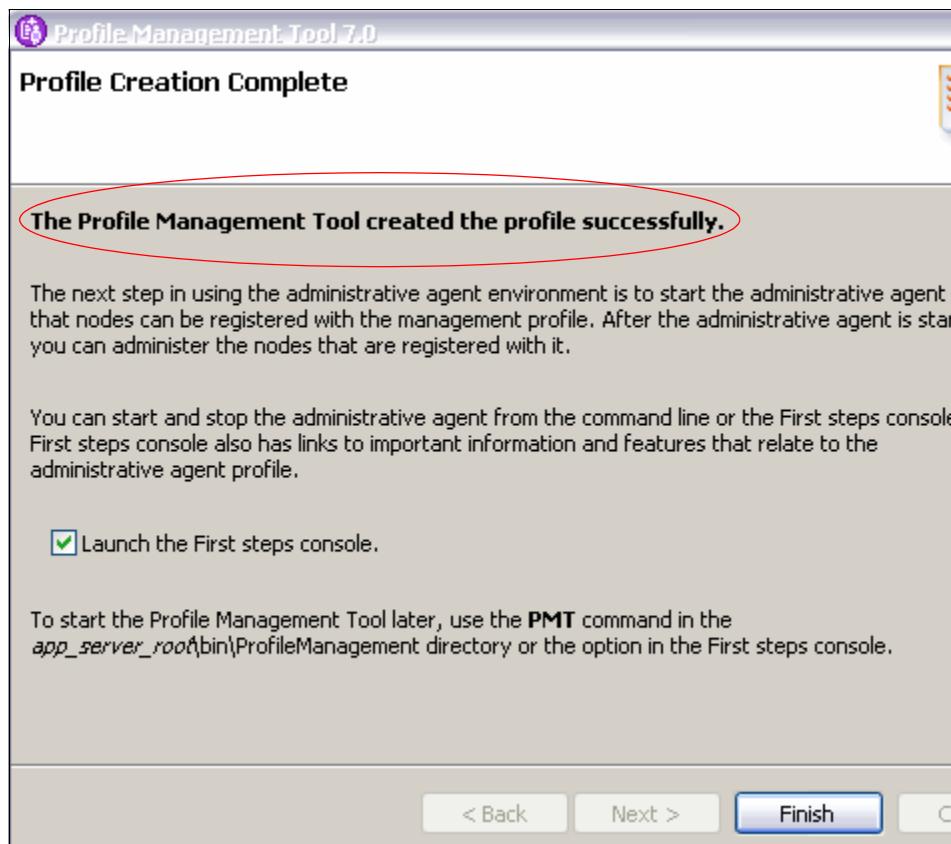


Figure 2-11 Profile creation complete - messages

If the messages indicate that the profile was not created successfully, then look in `install_root/logs/manageprofiles/profile_name_create.log` to determine what went wrong.

Troubleshooting information can be found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.installation.base.doc/info/aes/ae/tins_trouble.html

- ▶ Run the installation verification test:

Each profile has its own installation verification tests (IVT) program that starts the process defined by the profile and runs a series of verification tests. The IVT program scans the SystemOut.log file for errors and verifies core functionality of the profile.

Use the First Steps console to run the IVT. The First Steps console starts by default when you click **Finish** at end of the profile creation. You can start the First Steps console any time by using the **firststeps** command located in *profile_root/firststeps* directory. The options on this console vary depending on the profile type.

Alternatively the **ivt** command can be executed from the *profile_root/bin*. The IVT program verifies that the installation of the application server or deployment manager profile was successful.

Messages from the IVT are displayed on the First Steps window and logged in the following places:

- *profile_root/logs/server_name/startServer.log*
- *profile_root/logs/server_name/SystemOut.log*
- ▶ If applicable, log in to the administrative console hosted by the process. You can access the console from the First Steps menu or by accessing its URL from a Web browser:

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

Here is a sample URL:

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

The administrative console port is selected during profile creation (see Figure 2-16 on page 67)

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

Figure 2-12 shows the First steps console for a stand-alone application server.

First steps

Installation verification
Confirm that your server is installed and that it can start properly.

Start the server
Start the server and its applications.

Administrative console
Install and administer applications.

Profile management tool
Work with profiles.

Samples gallery
See WebSphere Application Server in action.

Information center for WebSphere Application Server
Learn more about WebSphere Application Server.

Migration wizard
Migrate WebSphere Application Server 5.1, 6.0 or 6.1 to version 7.0.
Migrate WebSphere Application Server Feature Pack for Web Services to version 7.0.

Exit

Figure 2-12 First Steps

2.3.3 Creating an application server profile

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

This section takes you through the steps of creating the application server profile: It shows the steps in the Advanced path through the profile creation:

1. Start the Profile Management Tool and click **Launch Profile Management Tool** on the Welcome page.
2. Click the **Create** button.
3. Select **Application server** as the profile type and click **Next**.
4. Select **Advanced**. Click **Next**.
5. Select the applications you want to deploy (Figure 2-13).

Installing the administrative console is recommended. However, there might be some circumstances when you would not want to install an administrative console, such as, if you plan to control all administrative tasks via scripting.

WebSphere Application Server provides sample applications that you can use to familiarize yourself with WebSphere applications. If you have installed the sample applications (optional during WebSphere Application Server installation), then you can opt to deploy these to the server during profile creation. For information about the samples available and how to install them, see the topic, *Accessing the Samples*, under *Learn about WebSphere Applications*, in the Information Center:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/welc6tech.html>

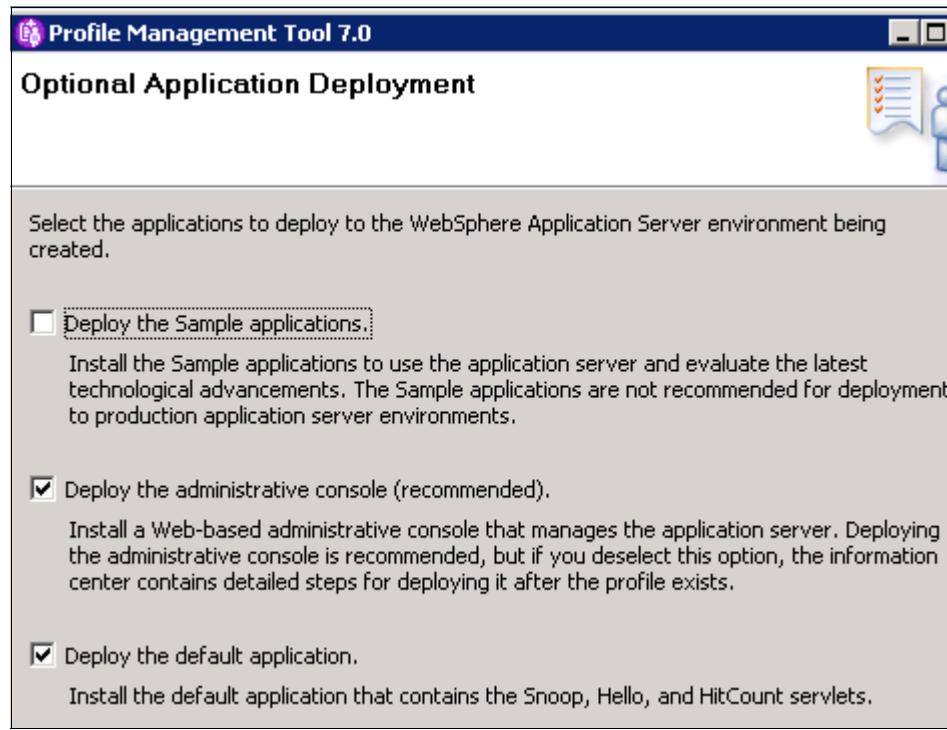


Figure 2-13 Select applications required

Click **Next**.

6. Enter a unique name for the profile or accept the default. If the application server will be used primarily for development purposes, check the option to create it from the development template. The development template reduces startup time and allows the server to run on less powerful hardware. Do not use this for production servers (Figure 2-14).

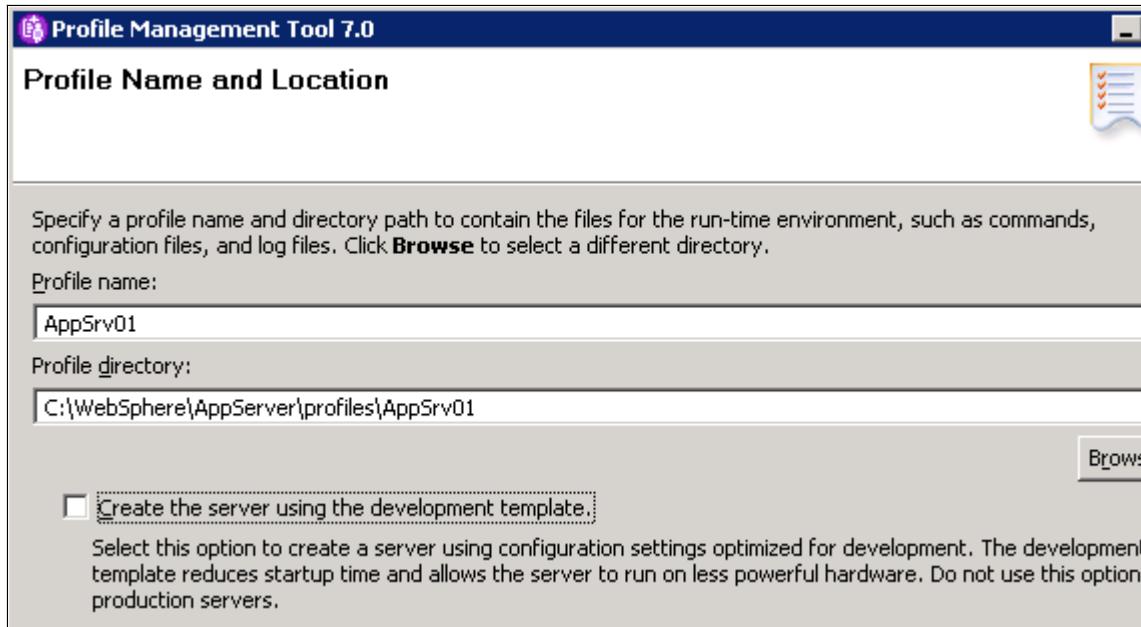


Figure 2-14 Enter name and location

Click **Next**.

7. Enter the new node name and the system host name. The node name will default based on the host name of your system. The wizard recognizes if there are existing nodes in the installation and takes this into account when creating the default node name (Figure 2-15).

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

 **Profile Management Tool 7.0**

Node and Host Names

Specify a node name, a server name, and a host name for this profile.

Node name:
wea01Node01

Server name:
server1

Host name:
wea01

Figure 2-15 Enter host and node names

Click **Next**.

8. Choose whether to enable administrative security. If you enable security, you are asked for a user ID and password that will be added to a file-based user registry with the Administrator role.

You are asked for a password for the samples if you checked the earlier option to include the samples in this application server.

Important: It might seem obvious, but do make a note of the user ID and passwords you enter here. Without the administrator ID, you will not be able to manage the application server.

Click **Next**.

9. Elect to either create new default personal and root signing certificates or to import them.

Click **Next**.

10. Review and modify the certificate information as needed.

Click **Next**.

11. The wizard presents a list of TCP/IP ports for use by the application server. If you already have existing profiles on the system (within this installation), this is taken into account when the wizard selects the port assignments, but you should verify that these ports will be unique on the system. You can select a different port by using the up and down arrows next to the port number.

Figure 2-16 shows some typical port assignments.

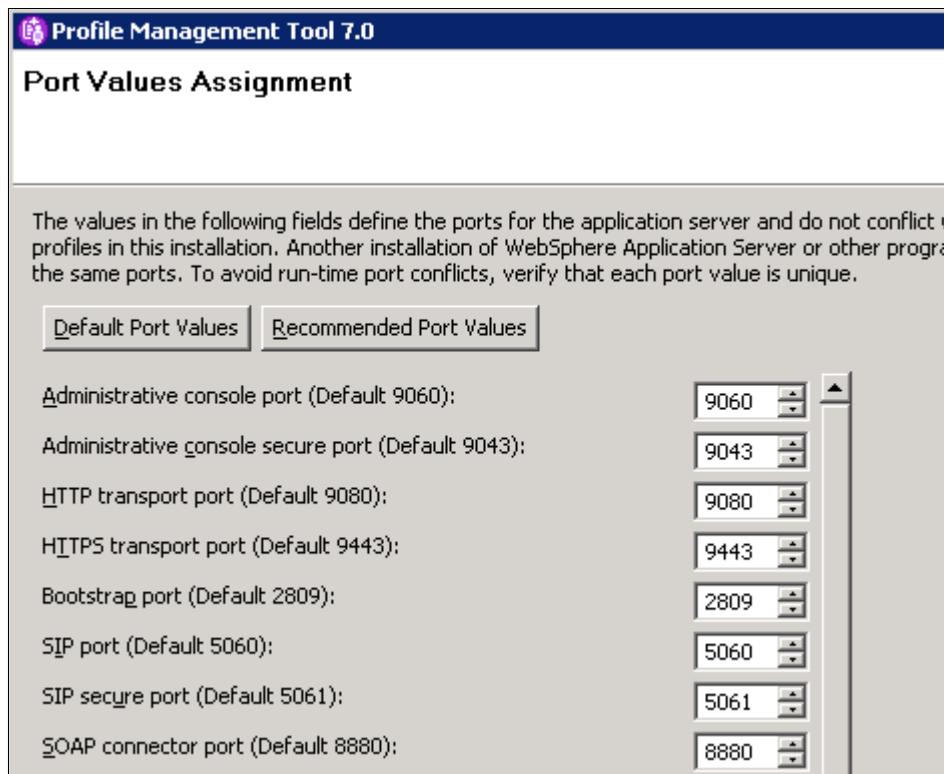


Figure 2-16 Select ports

Attention: Make a note of the following port numbers for later use:

- ▶ *SOAP connector port:* If you plan to federate this node to a deployment manager later using the deployment manager administrator console, you need to know this port number. This is also the port that you connect to when using the `wsadmin` administration scripting interface.
- ▶ *Administrative console port:* You need to know this port in order to access the administrative console. When you turn on security, you need to know the *Administrative console secure port*.
- ▶ *HTTP transport port:* This port is used to access applications running on the server directly vs. going through a Web server. This is useful in test environments.

Click **Next**.

12. If the profile is being created on a Windows system, select whether you want the server to run as a Windows service.

Click **Next**.

13. The wizard allows you to create an optional Web server definition (Figure 2-17). Web server definitions define an external Web server to WebSphere Application Server. This allows you to manage Web server plug-in configuration files for the Web server and in some cases to manage the Web server. If you have not installed a Web server or want to do this later, you can easily do this from the administrative console.

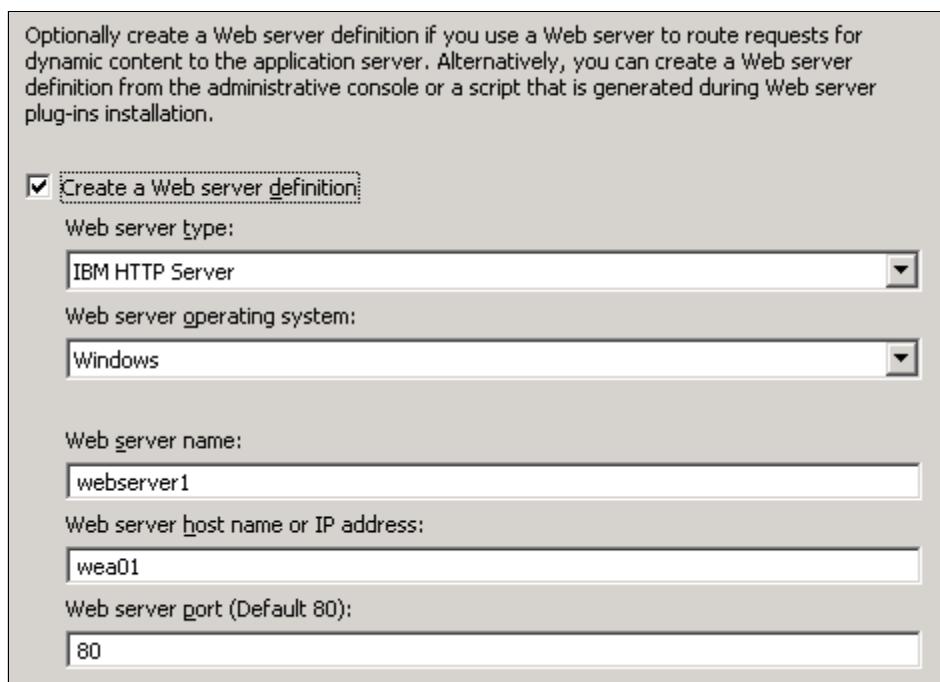


Figure 2-17 Creating a Web Server definition

Click **Next**.

14. If you elect to create a Web server definition, the next panel shows the default locations of the Web server installation and Web server plug-in path. Change this to match your installation (Figure 2-18).

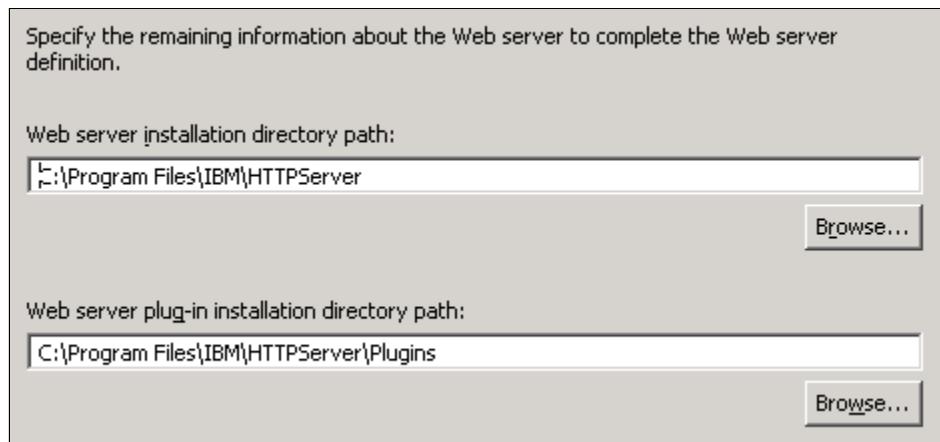


Figure 2-18 Location of Web server definition

Click **Next**.

15. Review the options you have chosen and click **Next** to create the profile.
16. The final window indicates the success or failure of the profile creation. If you have errors, check the log at:
install_root/logs/manageprofiles/profile_name_create.log
You can also find logs for individual actions stored in:
profile_root/logs
17. Click **Finish** to close the wizard and start the First Steps application.
18. Use the First Steps console to verify the installation, start the server, and login to the administrative console.

19. Display the configuration from the console. You should be able to see the following items from the administrative console:

- Application servers:

Select **Servers** → **Server Types** → **WebSphere Application servers**.
You should see the application server in the list (Figure 2-19).

The screenshot shows the Integrated Solutions Console interface. The left sidebar has a 'View' dropdown set to 'All tasks' and a list of categories: Welcome, Guided Activities, Servers, Applications, Services, Resources, Security, Environment, and System administration. Under 'Servers', there is a 'Server Types' section with options: WebSphere application, WebSphere MQ server, and Web servers. The main content area is titled 'Application servers' and contains a brief description: 'Use this page to view a list of the application servers in your environment and the status of each of these servers. You can also use this page to change the status of a specific application server.' Below this is a table with columns: Name, Node, Host Name, and Version. A single row is shown with the values: server1, wea01Node01, wea01, ND 7.0.0.0. At the bottom of the table, it says 'Total 1'.

Figure 2-19 Application server defined by the application server profile

To see the configuration of this server, click the name in the list.

b. Enterprise applications:

Select **Applications** → **Application Types** → **WebSphere enterprise applications**. You should see a list of applications. Figure 2-20 shows the WebSphere sample applications.

The screenshot shows the Integrated Solutions Console interface. The left sidebar has a 'View' dropdown set to 'All tasks' and a tree view with nodes like 'Welcome', 'Guided Activities', 'Servers', 'Applications' (selected), 'New Application', 'Application Types' (selected), 'WebSphere enterprise', 'Business-level ap...', 'Assets', 'Services', 'Resources', 'Security', 'Environment', 'System administration', 'Users and Groups', 'Monitoring and Tuning', and 'Troubleshooting'. The main content area is titled 'Enterprise Applications' with the sub-section 'Enterprise Applications'. It says 'Use this page to manage installed applications. A single application can be started or stopped from this page.' Below this are buttons for 'Start', 'Stop', 'Install', 'Uninstall', 'Update', 'Rollout Update', and 'Re'. There are also icons for creating, deleting, and cloning applications. A search bar at the top right says 'Select Name' with a dropdown arrow. Below it is a table titled 'You can administer the following resources:' with five entries: 'DefaultApplication' (checkbox checked), 'PlantsByWebSphere' (checkbox checked), 'SamplesGallery' (checkbox checked), 'iwtApp' (checkbox checked), and 'query' (checkbox checked). The table footer says 'Total 5'.

Figure 2-20 Applications installed on server1

Working with application servers: For information about starting, stopping, and viewing application servers, see 6.4, “Working with application servers” on page 297.

2.3.4 Creating a deployment manager profile

To create a deployment manager profile (Figure 2-21):

1. Start the PMT and click the **Launch Profile Management Tool** button on the Welcome page.
2. Click **Create**.
3. Select **Management**. Click **Next**.
4. Select **Deployment manager**. Click **Next**.
5. Select whether to take the typical settings or to go through the advanced windows. The options that you see next depend on the path you take.
If **Typical** is selected, then you only see one more option (to enable security).
If **Advanced** is selected, you continue with the following steps.
6. Select the option to deploy the administrative console (the default) and click **Next**.
7. Enter a unique name for the profile or accept the default. The profile name becomes the directory name for the profile files. Click the box if you want this to be the default profile for receiving commands. Select the location for the profile and click **Next**.

Specify a profile name and directory path to contain the files for the run-time environment, such as commands, configuration files, and log files. Click **Browse** to select a different directory.

Profile name:

Profile directory:

Make this profile the default.

Each installation of WebSphere Application Server always has one default profile. Commands that run without referring to a specific profile use the default profile. Select this option to make this profile the new default

Figure 2-21 Creating a deployment manager profile: Enter name and location

8. Enter the node, host, and cell names. These default, based on the host name of your system. The wizard recognizes if there are existing cells and nodes in the installation and takes this into account when creating the default names. Click **Next** (Figure 2-22).

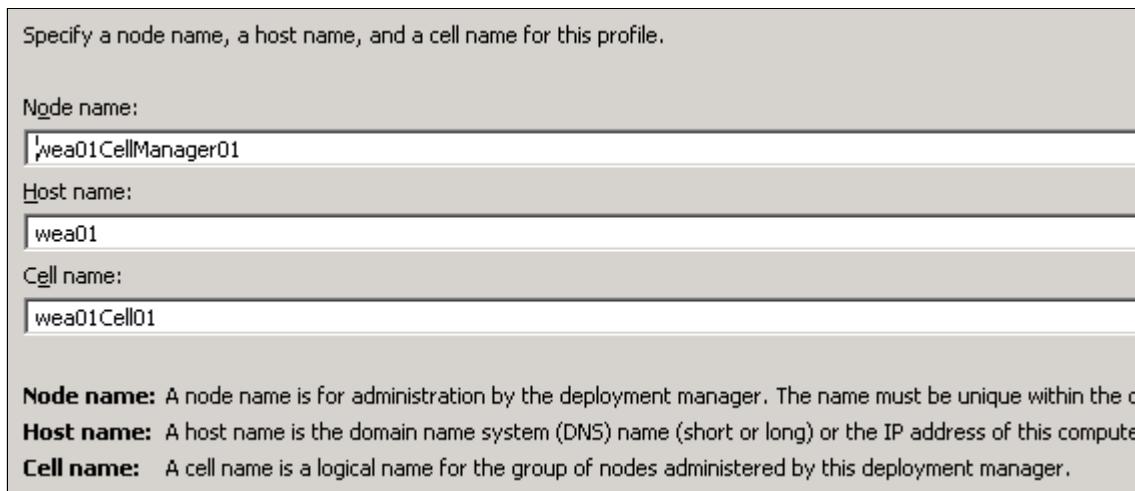


Figure 2-22 Creating a deployment manager profile: Enter cell, host, and node names

9. Choose whether to enable administrative security. If you enable security here, you are asked for a user ID and password that will be added to a file-based user registry with the Administrator role. Click **Next**.
10. Elect to either create new default personal and root signing certificates or to import them.
Click **Next**.
11. Review and modify the certificate information as needed.
Click **Next**.
12. The wizard presents a list of TCP/IP ports for use by the deployment manager. If you already have existing profiles on the system, this is taken into account when the wizard selects the port assignments. However, you should verify that these ports will be unique on the system (Figure 2-23).

Default Port Values		Recommended Port Values
Administrative console port (Default 9060):	9061	<input type="button" value="▼"/>
Administrative console secure port (Default 9043):	9044	<input type="button" value="▼"/>
Bootstrap port (Default 9809):	9809	<input type="button" value="▼"/>
SOAP connector port (Default 8879):	8879	<input type="button" value="▼"/>
Administrative interprocess communication port (Default 9632)(X):	9632	<input type="button" value="▼"/>
SAS SSL ServerAuth port (Default 9401):	9405	<input type="button" value="▼"/>
CSIV2 ServerAuth listener port (Default 9403):	9404	<input type="button" value="▼"/>
CSIV2 MultiAuth listener port (Default 9402):	9406	<input type="button" value="▼"/>
QRB listener port (Default 9100):	9101	<input type="button" value="▼"/>
Cell discovery port (Default 7277)(6):	7277	<input type="button" value="▼"/>
High availability manager communication (DCS) port (Default 9352):	9352	<input type="button" value="▼"/>
DataPower appliance manager secure inbound port (Default 5555):	5555	<input type="button" value="▼"/>

Figure 2-23 Creating a deployment manager profile: Select ports

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

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

13. If you would like to run the process as a Windows service, leave the box checked and enter the values for the logon and startup type.

Click **Next**.

14. Review the options that you have chosen (Figure 2-24). If you took the Typical path through the wizard, make sure that the default selections suit your needs. Click **Create** to create the profile.

Review the information in the summary for correctness. If the information is correct, click **Create** profile. Click **Back** to change values on the previous panels.

Application server environment to create: Management

Server type: Deployment manager

Location: C:\WebSphere\AppServer\profiles\dmgr01

Disk space required: 30 MB

Profile name: Dmgr01

Make this profile the default: False

Cell name: wea01Cell01

Node name: wea01CellManager01

Host name: wea01

Deploy the administrative console (recommended): True

Enable administrative security (recommended): True

Administrative console port: 9061

Administrative console secure port: 9044

Deployment manager bootstrap port: 9809

Deployment manager SOAP connector port: 8879

Run deployment manager as a service: True

Figure 2-24 Creating a deployment manager profile: Finish

15. The final window indicates the success or failure of the profile creation. If you have errors, check the log at:

install_root/logs/manageprofiles/profile_name_create.log

You can also find logs for individual actions stored in:

profile_root/logs

16. Click **Finish** to close the wizard and start the First Steps application

17. Verify the installation. You can do this directly from the First Steps menu. The IVT process starts the deployment manager and checks the log file for warnings or errors on start.

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

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

Here is a sample URL in the address bar:

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

19. Log in and display the configuration from the console. You should be able to see the following items from the administrative console:

- a. Cell information: Select **System administration** → **Cell**.
- b. Deployment manager: Select **System administration** → **Deployment manager**.
- c. Deployment manager node: Select **System administration** → **Nodes**.
- d. The default node group: Select **System administration** → **Node groups**.

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

- a. A node agent:

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

- b. Application servers

Working with deployment managers: For information about starting, stopping, and viewing deployment managers, see 6.1, “Working with the deployment manager” on page 290.

2.3.5 Creating a cell profile

Table 2-2 shows a summary of the options you have during a cell profile creation. Using this option actually creates two distinct profiles, a deployment manager profile and an application server profile. The application server profile is federated to the cell. The options you see are a reflection of the options you would see if you were creating the individual profiles versus a cell. The PMT panels give you basically the same options that you would see if you created a deployment manager, then an application server.

Table 2-2 Cell profile options

Typical	Advanced
The administrative console and default application are deployed by default. The sample applications are not deployed.	You have the option to deploy the administrative console (recommended), the default application, and the sample applications (if installed).
The profile name for the deployment manager is Dmgrxx by default, where xx is 01 for the first deployment manager profile and increments for each one created. The profile is stored in <i>install_root/profiles/Dmgrxx</i> .	You can specify the profile name and its location.
The profile name for the federated application server and node is AppSrvxx by default, where xx is 01 for the first application server profile and increments for each one created. The profile is stored in <i>install_root/profiles/AppSrvxx</i> .	You can specify the profile name and its location.
Neither profile is made the default profile.	You can choose to make the deployment manager profile the default profile.
The cell name is <host>Cellxx. The node name for the deployment manager is <host>CellManagerxx. The node name for the application server is <host>Nodexx. The host name is prefilled in with your system's DNS host name.	You can specify the cell name, the host name, and the profile names for both profiles.
You can enable administrative security (yes or no). If you select yes, you are asked to specify a user name and password that will be given administrative authority.	
TCP/IP ports 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 deployment manager will be run as a service.	(Windows) You can choose whether the deployment manager will run as a service.
Does not create a Web server definition.	Allows you to define an external Web server to the configuration.

2.3.6 Creating a custom profile

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

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

Note: With other profiles, you have the option of registering the processes as Windows services. This does not appear as an option when you create a custom profile.

The following steps outline the process of creating a custom profile (Figure 2-25):

1. Start the Profile Management Tool and click the **Launch Profile Management Tool** button on the Welcome page.
2. Click **Create**.
3. Select **Custom profile**. Click **Next**.
4. Select whether to take the typical settings or to go through the advanced windows. The options you see next depend on the path you take.
If **Typical** is selected, then you only see one more option (to enable security).
If **Advanced** is selected, you continue with the following steps.
5. Enter a unique name for the profile or accept the default. The profile name becomes the directory name for the profile files.
Click the box if you want this directory to be the default profile for receiving commands. Click **Next**.

Specify a profile name and directory path to contain the files for the run-time environment, such as commands, configuration files, and log files. Click **Browse** to select a different directory.

Profile name:	<input type="text" value="Custom01"/>
Profile directory:	<input type="text" value="C:\WebSphere\AppServer\profiles\Custom01"/> <input type="button" value="Browse..."/>
<input type="checkbox"/> Make this profile the default.	

Figure 2-25 Creating a Custom profile: Enter name and location

6. Enter the new node name and the system host name. The node name defaults to the host name of your system. The wizard recognizes if there are existing nodes in the installation and takes this into account when creating the default node name. Click **Next** (Figure 2-26).

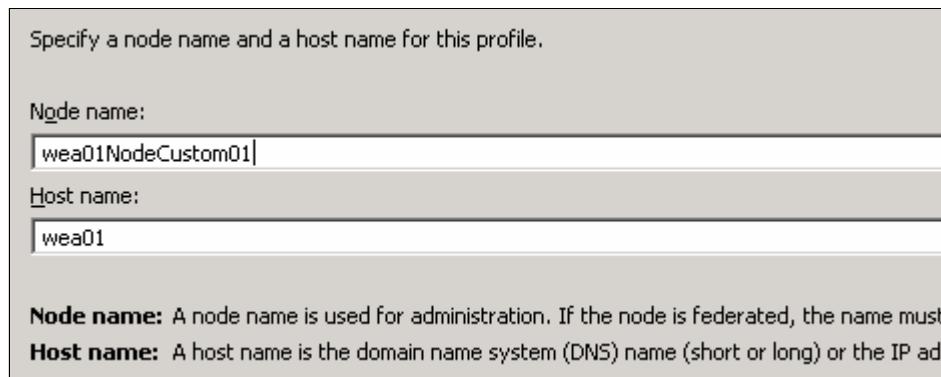


Figure 2-26 Creating a custom profile: Enter host, and node names

7. If you would like to federate the new node defined by the profile to a cell as part of the wizard process, leave the **Federate this node later** box unchecked and enter the host name and SOAP connector port for the deployment manager. Enter the user ID and password of the administrator ID for the deployment manager.

Click **Next**.

When you click **Next**, a connection is attempted to the deployment manager. If you have entered any of these values incorrectly, you will be able to correct them (Figure 2-27).

Specify the host name or IP address and the SOAP port number for an existing deployment manager. Federation can occur only if the deployment manager is running.

Deployment manager host name or IP address:
localhost

Deployment manager SOAP port number (Default 8879):
8879

Deployment manager authentication
Provide a user name and password that can be authenticated, if administrative security is enabled for the deployment manager.

User name:

Password:

Federate this node later.
You must federate this node later using the **addNode** command if the deployment manager:
- is not running
- has the SOAP connector disabled

Figure 2-27 Creating a custom profile: Federate later

8. Review the options you have chosen. If you took the Typical path through the wizard, make sure that the default selections suit your needs (Figure 2-28).

Review the information in the summary for correctness. If the information is correct, click **Create** creating a new profile. Click **Back** to change values on the previous panels.

Application server environment to create: Custom profile

Location: C:\WebSphere\AppServer\profiles\Custom01

Disk space required: 10 MB

Profile name: Custom01

Make this profile the default: False

Node name: wea01NodeCustom01

Host name: wea01

Federate to deployment manager: False

Figure 2-28 Creating a custom profile: Summary

Click **Create** to create the profile.

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

If you have errors, check the log at:

install_root/logs/manageprofiles/profile_name_create.log

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

You can also find logs for individual actions stored in:

profile_root/logs

Note: Custom profiles do not create a server process, so you cannot verify, stop, or start the profile. The only reason to launch the First Steps menu is if you want to link to the Information Center or launch the migration wizard.

10. The federation process creates a node agent for the new node, federates it to the cell, and starts the node agent. If you federated the custom profile, open the deployment manager administrative console and view the node and node agent:

- Select **System Administration → Nodes**. You should see the new node.
- Select **System Administration → Node agents**. You should see the new node agent.

- Select **System Administration** → **Cells**. Click the **Topology tab** and expand the view. From here, you can see a tree diagram of the cell (Figure 2-29).

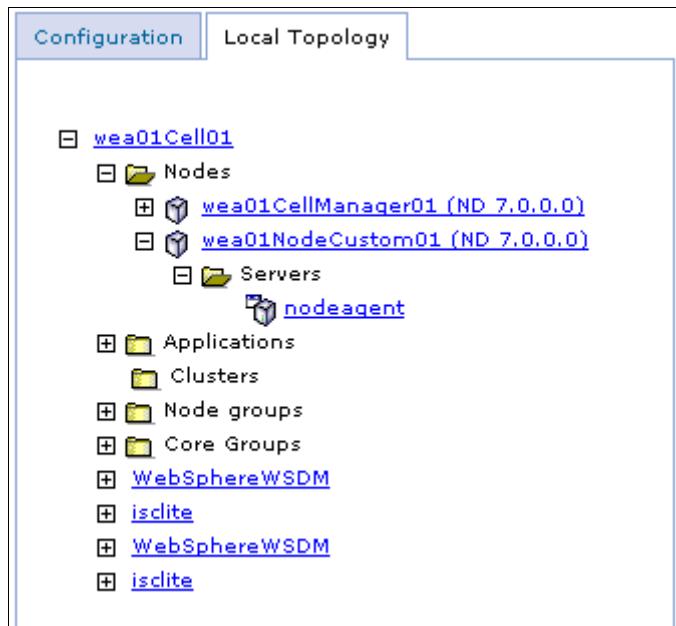


Figure 2-29 Topology view of a cell

If you have not federated the node, proceed to 2.3.7, “Federating nodes to a cell” on page 82. Otherwise, you can continue by defining an application server on the new node (see 6.4.1, “Creating an application server” on page 298).

2.3.7 Federating nodes to a cell

A custom profile defines a node that can be added to a cell. The **addNode** command is used to federate a node in a custom profile to a cell.

A stand-alone application server can also be federated to a cell with the **addNode** command, or from the deployment manager administrative console. The administrative console invokes the **addNode** command on the target system.

When you federate a node, the node name from the federated node is used as the new node name and must be unique in the cell. If the name of the node that you are federating already exists, the **addNode** operation will fail.

The addnode command

The **addNode** command is run from the *install_root/bin* or *profile_root/bin* directory of the installation for the profile that will be federated (that is, from the custom profile or application server profile installation).

Addnode command syntax

The syntax of the **addNode** command is shown in Example 2-1.

Example 2-1 The addNode command syntax

```
Usage: addNode dmgr_host [dmgr_port] [-conntype <type>] [-includeapps]
[-includebuses] [-startingport <portnumber>]
[-portprops <qualified-filename>] [-nodeagentshortname <name>]
[-nodegroupname <name>] [-registerservice] [-serviceusername <name>]
[-servicepassword <password>] [-coregroupname <name>] [-noagent]
[-statusport <port>] [-quiet] [-nowait] [-logfile <filename>]
[-replacelog] [-trace] [-username <username>] [-password <pwd>]
[-localusername <localusername>] [-localpassword <localpassword>]
[-profileName <profile>] [-excludesecuritydomains] [-help]
```

- ▶ **dmgr_host, -username, -password**

This command connects to the deployment manager, so you have to specify the deployment manager host name and a user ID/password with administrative privileges on the deployment manager.

- ▶ **dmgr_port, -conntype**

The default is to connect to the deployment manager using SOAP and port 8879. If your deployment manager was defined with this port, you do not need to specify anything. If not, you can specify the correct port, or you can use RMI as the connection type.

For SOAP connections, the port defined as the SOAP_CONNECTOR_PORT number on the deployment manager must be specified. If you choose to use an RMI connection instead, the ORB_LISTENER_ADDRESS port must be specified. You can see these in the port list of the deployment manager in the administrative console.

Tip: Port numbers are also stored in *profile_root/properties/portdef.props*

- ▶ **-startingport, -portprops <filename>**

The new node agent is assigned a range of ports automatically. If you want to specify the ports for the node rather than taking the default, you can specify a starting port using the -startingport parameter. The numbers are incremented from this number.

For example, if you specify 3333, the BOOTSTRAP_ADDRESS port will be 3333, CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS will be 3334, and so on.

As an alternative, you can provide specific ports by supplying a file with the port properties.

- ▶ -includeapps, -includebuses

If you are federating an application server, you can keep any applications that are deployed to the server and you can keep any service integration bus definitions that have been created. The default is that these are not included during federation and are lost.

For more information about the **addNode** syntax and options, see:

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

The **addNode** command performs the following actions:

1. Connects to the deployment manager process. This is necessary for the file transfers performed to and from the deployment manager in order to add the node to the cell.
2. Attempts to stop all running application servers on the node.
3. Backs up the current stand-alone node configuration to the *profile_root/config/backup/base/* directory.
4. Copies the stand-alone node configuration to a new cell structure that matches the deployment manager structure at the cell level.
5. Creates a new local config directory and definition (*server.xml*) for the node agent.
6. Creates entries (directories and files) in the master repository for the new node's managed servers, node agent, and application servers.
7. Uses the FileTransfer service to copy files from the new node to the master repository.
8. Uploads applications to the cell only if the -includeapps option is specified.
9. Performs the first file synchronization for the new node. This pulls everything down from the cell to the new node.
10. Fixes the node's **setupCmdLine** and **wsadmin** scripts to reflect the new cell environment settings.
11. Launches the node agent (unless -noagent is specified).

Federating a custom node to a cell

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

To federate the node to the cell, do the following actions:

1. Start the deployment manager.
2. Open a command window on the system where you created the custom profile for the new node. Switch to the *profile_root/bin* directory or *install_root/bin* directory.
3. Run the **addNode** command.

Example 2-2 shows an example of using the **addNode** command on a Windows system to add Node01 to the deployment manager using 8879 as the SOAP connector address.

Example 2-2 addNode command

```
C:\WebSphere\AppServer\profiles\Custom01\bin>
C:\WebSphere\AppServer\profiles\Custom01\bin>addNode localhost 8879
ADMU0116I: Tool information is being logged in file
          C:\WebSphere\AppServer\profiles\Custom01\logs\addNode.log
ADMU0128I: Starting tool with the Custom01 profile
CWPKI0308I: Adding signer alias "CN=wca01, OU=Root Certificate, " to local
              keystore "ClientDefaultTrustStore" with the following SHA digest:
              AF:60:11:60:15:5B:B3:54:0C:46:84:1A:B5:DC:C6:A9:B6:DC:0F:0E
CWPKI0308I: Adding signer alias "datapower" to local keystore
              "ClientDefaultTrustStore" with the following SHA digest:
              A9:BA:A4:B5:BC:26:2F:5D:2A:80:93:CA:BA:F4:31:05:F2:54:14:17
ADMU0001I: Begin federation of node wca01NodeCustom01 with Deployment Manager
          at localhost:8879.
ADMU0009I: Successfully connected to Deployment Manager Server: localhost:8879
ADMU0507I: No servers found in configuration under:
          C:\WebSphere\AppServer\profiles\Custom01\config\cells\wca01Node02Cell
/nodes/wca01NodeCustom01/servers
ADMU2010I: Stopping all server processes for node wca01NodeCustom01
ADMU0024I: Deleting the old backup directory.
ADMU0015I: Backing up the original cell repository.
ADMU0012I: Creating Node Agent configuration for node: wca01NodeCustom01
ADMU0014I: Adding node wca01NodeCustom01 configuration to cell: wca01Cell01
ADMU0016I: Synchronizing configuration between node and cell.
ADMU0018I: Launching Node Agent process for node: wca01NodeCustom01
ADMU0020I: Reading configuration for Node Agent process: nodeagent
```

ADMU0022I: Node Agent launched. Waiting for initialization status.
ADMU0030I: Node Agent initialization completed successfully. Process id is:
5368
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: nodeagent
ADMU0300I: The node wea01NodeCustom01 was successfully added to the wea01Cell01
cell.
ADMU0306I: Note:
ADMU0302I: Any cell-level documents from the standalone wea01Cell01
configuration have not been migrated to the new cell.
ADMU0307I: You might want to:
ADMU0303I: Update the configuration on the wea01Cell01 Deployment Manager with
values from the old cell-level documents.
ADMU0306I: Note:
ADMU0304I: Because -includeapps was not specified, applications installed on
the standalone node were not installed on the new cell.
ADMU0307I: You might want to:
ADMU0305I: Install applications onto the wea01Cell01 cell using wsadmin
\$AdminApp or the Administrative Console.
ADMU0003I: Node wea01NodeCustom01 has been successfully federated.
C:\WebSphere\AppServer\profiles\Custom01\bin>

4. Open the deployment manager administrative console and view the node and node agent:
 - Select **System Administration** → **Nodes**. You should see the new node.
 - Select **System Administration** → **Node agents**. You should see the new node agent and its status.

The node is started as a result of the federation process. If it does not appear to be started in the console, you can check the status from a command window on the node system:

```
cd profile_root\bin
serverStatus -all
```

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

```
cd profile_root\bin
startNode
```

For more information about managing nodes, see 6.5, “Working with nodes in a distributed environment” on page 325.

Creating application servers on the new node: The custom profile does not automatically give you an application server. You can follow the steps in 6.4.1, “Creating an application server” on page 298 to create a new server after the custom profile has been federated to a cell.

Federating an application server profile to a cell

If you are using the administrative console to federate an application server, keep in mind the following considerations:

- ▶ Both the deployment manager and the application server must be running.
- ▶ You need to be logged into the console with an ID that has administrator privileges.
- ▶ The command will connect to the application server. This requires you to specify the application server host name and a user ID that can connect to the server. In turn, the node has to connect to the deployment manager. Specify a user ID and password for this connection.
- ▶ You need to specify the host name, JMX connection type, and port number to use to connect to the application server. The JMX connection type can be SOAP or RMI. The default is a SOAP connection using port 8880.

To federate an application server profile to a cell (Figure 2-30), do the following steps:

1. Ensure that the application server and deployment manager are running.
2. Open the deployment manager administrative console.
3. Select **System Administration → Nodes → Add Node**.
4. Select **Managed node** and click **Next**.
5. Enter the host name and SOAP connector port of the application server profile.

If you want to keep the sample applications and any other applications you have installed, check the **Include applications** box.

Enter the administrator user ID and passwords for both the application server and the deployment manager.

Node connection

* Host
wea01.hursley.ibm.com

* JMX connector type
SOAP

* JMX connector port
8882

Application server user name
wasadm

Application server password

* Deployment manager user name
wasadm

* Deployment manager password

Config URL
file://\${USER_INSTALL_ROOT}/properties/sas.c

Options

Include applications

Include buses

Starting port

Use default

Specify

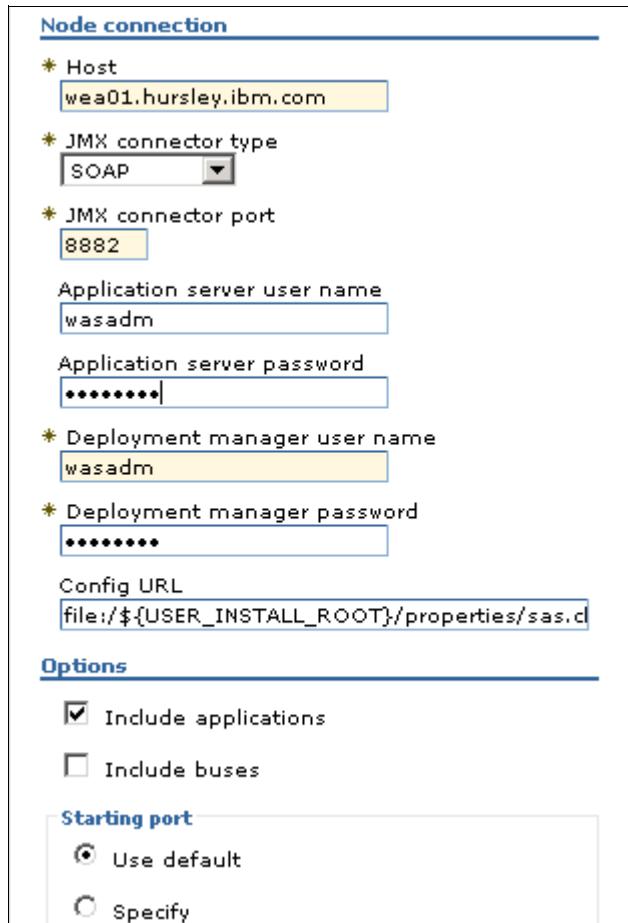


Figure 2-30 Adding a standalone application profile to a cell

Click **OK**.

6. If the node is a Windows node, you have the opportunity to register the new node agent as a Windows service. Make your selection and click **OK**.

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

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

At the completion of the process:

- ▶ The profile directory for the application server still exists and is used for the new node.
- ▶ The old cell name for the application server has been replaced in the profile directory with the cell name of the deployment manager.
profile_root/config/cells/dmgr_cell
- ▶ A new entry in the deployment manager profile directory has been added for the new node.
dmgr_profile_root/config/cells/dmgr_cell/nodes/federated node
- ▶ An entry for each node in the cell is added to the application server profile configuration. Each node entry contains the serverindex.xml file for the node.
profile_root/config/cells/dmgr_cell/nodes/federated node

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

Example 2-3 shows an example of using the **addNode** command to add an application server profile to a cell. The command specifies the deployment manager host (T60) and the SOAP connector port (8882). Applications currently installed on the application server will still be installed on the server after federation.

Example 2-3 The addNode command usage examples

```
C:\WebSphereV7\AppServer\bin>addNode t60 8882 -profileName node40b  
-includeapps -username admin -password adminpwd  
ADMU0116I: Tool information is being logged in file  
    C:\WebSphereV7\AppServer\profiles\node40b\logs\addNode.log  
ADMU0128I: Starting tool with the node40b profile  
CWPKI0308I: Adding signer alias "default_2" to local keystore  
    "ClientDefaultTrustStore" with the following SHA digest:  
    9D:99:04:63:97:8C:C0:76:19:46:5A:C4:C0:35:20:FE:DE:21:FD:29  
ADMU0001I: Begin federation of node node40b with Deployment Manager at  
    t60:8882.  
ADMU0009I: Successfully connected to Deployment Manager Server:  
    t60:8882  
ADMU0505I: Servers found in configuration:  
ADMU0506I: Server name: server40b1  
ADMU2010I: Stopping all server processes for node node40b  
ADMU0512I: Server server40b1 cannot be reached. It appears to be  
    stopped.  
ADMU0024I: Deleting the old backup directory.  
ADMU0015I: Backing up the original cell repository.
```

ADMU0012I: Creating Node Agent configuration for node: node40b
ADMU0120I: isclite.ear will not be uploaded since it already exists in the target repository.
ADMU0120I: DefaultApplication.ear will not be uploaded since it already exists in the target repository.

ADMU0016I: Synchronizing configuration between node and cell.
ADMU0018I: Launching Node Agent process for node: node40b
ADMU0020I: Reading configuration for Node Agent process: nodeagent
ADMU0022I: Node Agent launched. Waiting for initialization status.
ADMU0030I: Node Agent initialization completed successfully. Process id is: 5512
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: nodeagent
ADMU0506I: Server name: server40b1

ADMU0308I: The node node40b and associated applications were successfully added to the Cell140 cell.

ADMU0306I: Note:
ADMU0302I: Any cell-level documents from the standalone Cell140 configuration have not been migrated to the new cell.
ADMU0307I: You might want to:
ADMU0303I: Update the configuration on the Cell140 Deployment Manager with values from the old cell-level documents.

ADMU0003I: Node node40b has been successfully federated.

2.3.8 Creating an administrative agent profile

This section takes you through the steps of creating an administrative agent profile (Figure 2-31). Follow these steps:

1. Start the PMT and click the **Launch Profile Management Tool** button on the Welcome page.
2. Click **Create**.
3. Select **Management**. Click **Next**.
4. Select **Administrative agent** and click **Next**.

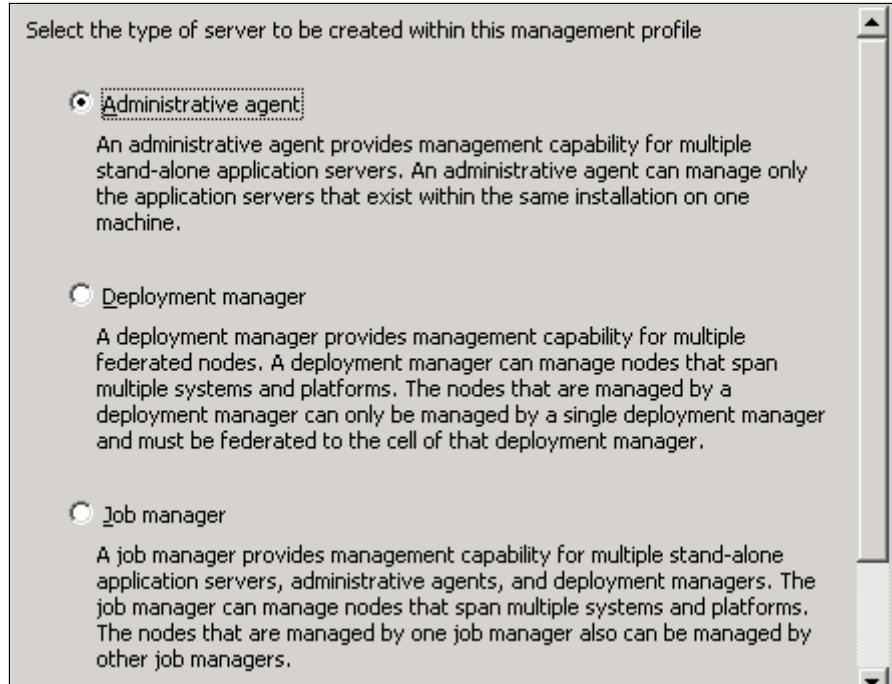


Figure 2-31 Administrative agent option

5. Select the typical or advanced path. Click **Next**.
If **Typical** is selected, then you only see one more option (to enable security).
If **Advanced** is selected, you see the next step.
6. Select the option to install the administrative console (Figure 2-32).
Click **Next**.

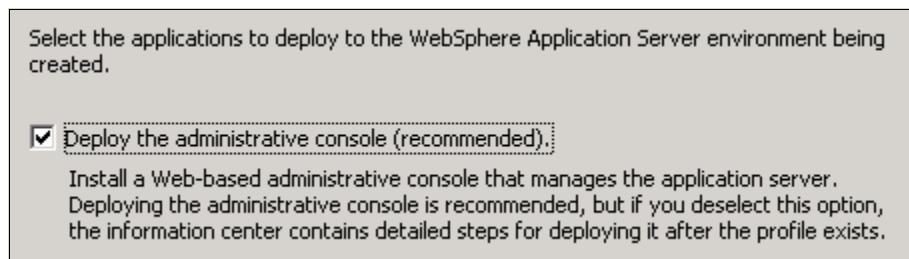


Figure 2-32 Deploy administrative console

7. Enter a unique name for the profile or accept the default. The profile name becomes the directory name for the profile.

Click the box if you want this directory to be the default profile for receiving commands (Figure 2-33). Click **Next**.

Specify a profile name and directory path to contain the files for the run-time environment, such as configuration files, and log files. Click **Browse** to select a different directory.

Profile name:

Profile directory:

Make this profile the default.

Each installation of WebSphere Application Server always has one default profile. Commands referring to a specific profile use the default profile. Select this option to make this profile the default.

Figure 2-33 Enter name and location of profile

8. Enter the new node name and the system host name. The node name defaults to the host name of your system. The wizard recognizes if there are existing nodes in the installation and takes this into account when creating the default node name (Figure 2-34). Click **Next**.

Node name:

Host name:

Cell name:

Figure 2-34 Enter host, and node names

9. Choose whether to enable administrative security. If you enable security here, you are asked for a user ID and password that will be added to a file-based user registry with the Administrative role. Click **Next**.
10. Elect to either create new default personal and root signing certificates or to import them. Click **Next**.
11. Review and modify the certificate information as needed. Click **Next**.

12. The wizard presents a list of TCP/IP ports for use by the application server. If you already have existing profiles on the system (within this installation), this will be taken into account when the wizard selects the port assignments, but you should verify that these ports will be unique on the system and alter them if required (Figure 2-35). Click **Next**.

Default Port Values	Recommended Port Values
Administrative console port (Default 9060):	9064
Administrative console secure port (Default 9043):	9047
Bootstrap port (Default 9807):	9807
SOAP connector port (Default 8877):	8877
Administrative interprocess communication port (Default 9630)(X):	9631
SAS SSL ServerAuth port (Default 9401):	9414
CSIV2 ServerAuth listener port (Default 9403):	9413
CSIY2 MultiAuth listener port (Default 9402):	9415
ORB listener port (Default 9098):	9098

Figure 2-35 Select ports

- Note the administrative console and SOAP connector ports for future use.
13. On Windows systems, select whether to run the administrative agent process as a Windows service.
Click **Next**.
14. Review the options you have chosen and click **Next** to create the profile.
15. After the wizard has finished, you are presented with the window indicating the success or failure of the process.
If you have errors, check the log at:
install_root/logs/manageprofiles/profile_name_create.log
You can also find logs for individual actions stored in:
profile_root/logs
16. Click **Finish** to close the wizard and start the First Steps application.
17. Run the IVT tests to ensure the installation was successful. The test will start the administrative agent.

18. Open the administrative agent console from the First Steps menu, or using the following URL:

`http://admin_host:port/ibm/console/`

Where port is the administrative console port selected during profile creation.

19. View the administrative agent by selecting **System Administration** → **Administrative agent** (Figure 2-36).

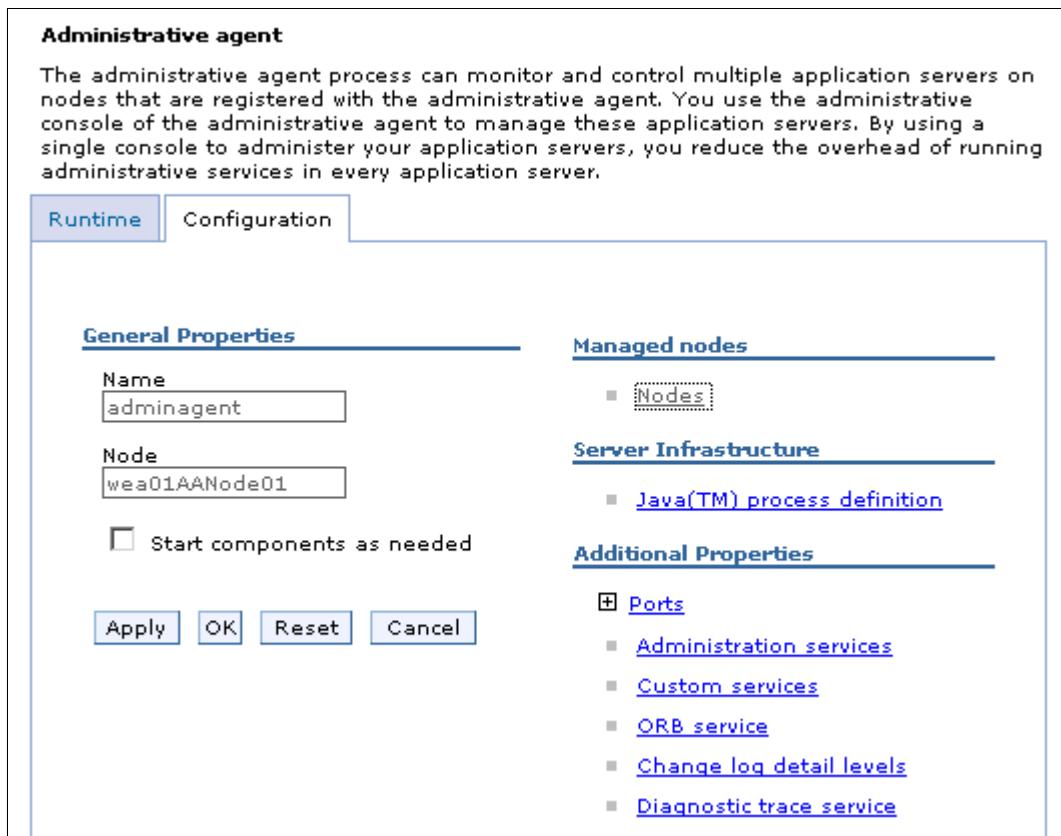


Figure 2-36 Administrative agent details in the Administration Console

Nodes that are registered to the administrative agent can be viewed by clicking on the **Nodes** link under Managed nodes. The list is initially empty. To register standalone application server nodes to the administrative agent, see 2.3.10, “Registering nodes to an administrative agent” on page 95.

2.3.9 Creating a job manager profile

The steps for creating a job manager profile are exactly the same as those for the administrative agent profile.

1. Start the Profile Management Tool and click the **Launch Profile Management Tool** button on the Welcome page.
2. Click **Create**.
3. Select **Management**. Click **Next**.
4. Select **Job manager** and click **Next**.

All further steps are exactly as described in the administrative agent profile.

You can open the job manager console from the First Steps menu, or by using the following URL:

`http://job_manager_host:port/ibm/console/`

Where port is the administrative console port selected during profile creation.

2.3.10 Registering nodes to an administrative agent

An administrative agent provides a single interface to unfederated application server nodes (standalone application server profiles).

Notes for use:

- ▶ The administrative agent and application servers must be on the same machine or sysplex.
- ▶ The administrative agent must be started before running the registerNode command
- ▶ You can only run the command on an unfederated standalone application server. When you run the command, the node for the standalone server is converted into a node that the administrative agent manages.

The **registerNode** command is used to register a node with an administrative agent. The syntax of the command is:

`registerNode [options]`

The options can be displayed using the -help parameter (Example 2-4).

Example 2-4 registerNode options

```
C:\WebSphereV7\AppServer\bin>registerNode -help
Usage: registerNode -profilePath <path to the base profile to be
registered>

[-host <adminagent host>] [-connType <SOAP | RMI | JSR160RMI | IPC>]
[-port <adminagent JMX port>] [-name <managed node name>]
[-openConnectors <SOAP,IPC,...>] [-username <adminagent user name>]
[-password <adminagent password>] [-nodeusername <base node user name>]
[-nodepassword <base node password>] [-profileName <adminagent profile
name>] [-portsFile <jmx ports filename>] [-trace] [-help]
```

For details on this command, see

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/ragt_registerNode.html

You can enter the command directly. For example, to register the application server defined by the SASrv40 profile to the administrative agent adminAgnt40, you would enter the following command:

```
registerNode.bat -profileName adminAgnt40 -profilePath
"C:\WebSphereV7\AppServer\profiles\SASrv40" -host t60 -connType SOAP
-port 8878 -username admin -password admin
```

Alternatively, you could create an execution file with the registerNode command. For example:

1. Create a file with contents similar to Example 2-5. The name of the file in this example is registerAppSrv03WithAdminAgent.bat. Modify the set statements to match your environment.

Example 2-5 Sample registerNode bat file contents

```
echo on
set WAS_HOME=C:\WebSphere\AppServer
set appAdminUser=wasadm
set appAdminPassword=wasadm11
set adminAgentAdminUser=wasadm
set adminAgentAdminPassword=wasadm11
set adminAgentProfileName=AdminAgent01
set adminAgentHostName=localhost
set adminAgentSoapPort=8877
set baseProfileName=AppSrv03
```

```
cd %WAS_HOME%\profiles\%adminAgentProfileName%\bin

registerNode.bat -profileName %adminAgentProfileName% -profilePath
"%WAS_HOME%\profiles\%baseProfileName%" -host %adminAgentHostName% -conntype SOAP
-port %adminAgentSoapPort% -username %adminAgentAdminUser% -password
%adminAgentAdminPassword% -nodeusername %appAdminUser% -nodepassword
%appAdminPassword%
```

2. Copy this file to *adminAgent_profile_root*/bin directory and run the file. You can see results similar to Example 2-6. Check the final message to make sure that the node was registered.

Example 2-6 Sample execution to register a node to the Administration profile

```
echo on

set WAS_HOME=C:\WebSphere\AppServer
set appAdminUser=wasadm
set appAdminPassword=wasadm11
set adminAgentAdminUser=wasadm
set adminAgentAdminPassword=wasadm11
set adminAgentProfileName=AdminAgent01
set adminAgentHostName=localhost
set adminAgentSoapPort=8877
set baseProfileName=AppSrv03

registerNode.bat -profileName AdminAgent01 -profilePath
"C:\WebSphere\AppServer\profiles\%baseProfileName%" -host localhost -conntype SOAP -port 8877
-username wasadm -password wasadm11 -nodeusername wasadm -nodepassword wasadm11
ADMU0116I: Tool information is being logged in file
    C:\WebSphere\AppServer\profiles\AdminAgent01\logs\registerNode.log
ADMU0128I: Starting tool with the AdminAgent01 profile
ADMU8053I: Successfully connected to AdminAgent Server: localhost:8877
ADMU8002I: Exchanging signers between adminagent and node with path
    C:\WebSphere\AppServer\profiles\%baseProfileName%.
ADMU8007I: Exchanged signers successfully.
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: server1
ADMU2010I: Stopping all server processes for node wea01Node03
ADMU0512I: Server server1 cannot be reached. It appears to be stopped.
ADMU8010I: Begin registration of Application Server with path
    C:\WebSphere\AppServer\profiles\%baseProfileName%
ADMU0024I: Deleting the old backup directory.
ADMU8004I: Backing up the original config directory of the node will be
registered.
```

ADMU8037I: Backing up the original wsadmin.properties file of the node will be registered.
ADMU8036I: Registering the node with an AdminAgent.
ADMU8042I: Node has been successfully registered.
ADMU8040I: The administrative agent is initializing the administrative subsystem for the registered node.
ADMU8014I: The administrative subsystem for registered node has been successfully initialized.
ADMU8041I: The administrative agent is starting the administrative subsystem for the registered node.
ADMU8015I: The administrative subsystem for registered node has been successfully started.
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: server1
ADMU8012I: Application Server with path
C:\WebSphere\AppServer\profiles\AppSrv03 has been successfully registered.

3. The next time you log onto the administrative agent console, you have the option to select a node to administer. In this case, you can select between the administrative agent (the top choice) and the new node you registered.

Select the administrative agent to view the new configuration (Figure 2-37).

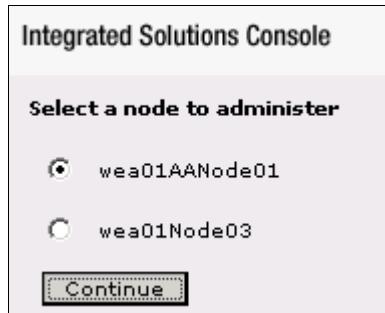


Figure 2-37 Option to pick which local Application Server to administer

4. Select **System Administration** → **Administrative agent** → **Nodes**.

This shows that this node is now registered with the administrative agent (Figure 2-38).

Administrative agent > Nodes

This panel lists all the managed nodes registered to this administrative agent. To register a managed node with a job manager, select a managed node in the collection and click Register with Job Manager. To unregister a managed node from a job manager, select a managed node in the collection and click Unregister from a Job Manager.

Preferences

Select	Name	Unique ID
<input type="checkbox"/>	wea01Node03	AppSrv03-BASE-9c056a91-ff09-47e6-af55-a067e1813184

Total 1

Figure 2-38 Node registered with administrative agent

2.3.11 Deregistering a node from the administrative agent

To deregister a node from the administrative agent use the **deregisterNode** command. Run this command from the *adminAgnt_profile_root/bin* directory.

Example 2-7 deRegisterNode command

```
deregisterNode.bat -connType SOAP -port 8877 -profilePath  
C:\WebSphere\AppServer\profiles\AppSrv03 -username wasadm -password wasadm11
```

2.3.12 Registering an administrative agent node with a job manager

This section describes the steps to register nodes within the administrative agent with a job manager:

1. Logon to the administrative agent node (Figure 2-39).

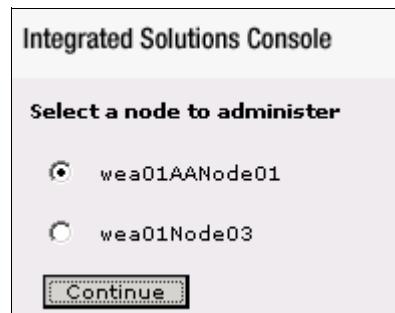


Figure 2-39 Option to pick which local Application Server to administer

Navigate to **System administration** → **Administrative agent** (Figure 2-40).

The dialog box has tabs 'Runtime' (selected) and 'Configuration'. The 'General Properties' section includes fields for 'Name' (adminagent) and 'Node' (wea01AANode01), and a checkbox 'Start components as needed'. The 'Managed nodes' section lists 'Nodes'. The 'Server Infrastructure' section lists 'Java(TM) process definition'. The 'Additional Properties' section lists 'Ports', 'Administration services', 'Custom services', 'ORB service', 'Change log detail levels', and 'Diagnostic trace service'.

Administrative agent

The administrative agent process can monitor and control multiple application servers on nodes that are registered with the administrative agent. You use the administrative console of the administrative agent to manage these application servers. By using a single console to administer your application servers, you reduce the overhead of running administrative services in every application server.

Runtime Configuration

General Properties

Name: adminagent
Node: wea01AANode01
 Start components as needed

Managed nodes

- Nodes

Server Infrastructure

- Java(TM) process definition

Additional Properties

- Ports
- Administration services
- Custom services
- ORB service
- Change log detail levels
- Diagnostic trace service

Figure 2-40 Administrative agent details

2. Click **Nodes** and select the node that you want to register with the job manager (Figure 2-41).

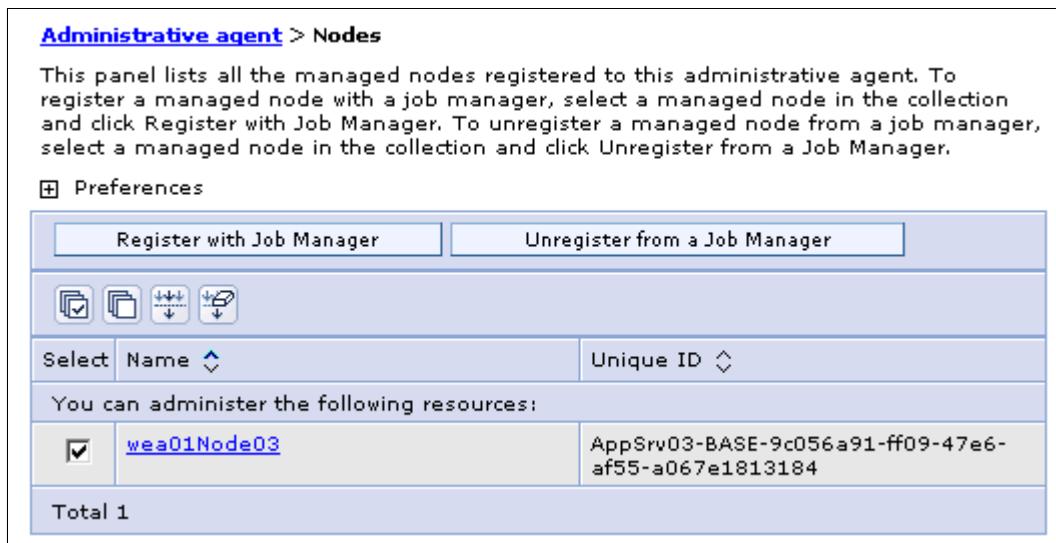


Figure 2-41 Select which node will be registered with the job manager

Note: You can register some or all of the nodes with the job manager. Or you can register some nodes with one job manager and other nodes with another job manager.

Click **Register with Job Manager**.

3. Enter the information required to connect to the job manager, including the host name, administrative host port, and user ID and password.

If the node name you are registering is already in use by the job manager, you can enter an alias for the node (Figure 2-42).

General Properties

* Managed node name	wea01Node03
Alias	
Host name	wea01.hursley.ibm.com
Port	9943
User name	wasadmin
Password	*****
Confirm password	*****

Buttons: OK, Reset, Cancel

Figure 2-42 Detailed options for registering a node with a job manager

Click **OK** to register the node.

4. Log into the job manager and go to **Jobs → Nodes** to see the newly registered node (Figure 2-43).

Display Resources ▾		
Select	Node name ▾	Version ▾
<input type="checkbox"/>	wea01Node03	ND 7.0.0.0
Total 1		

Figure 2-43 Nodes which the job manager is able to administer

For equivalent command line steps for this process, see the IBM InfoCenter at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/txml_7regmannode.htm

2.3.13 Registering a deployment manager with a job manager

This section describes the steps to register a deployment manager node with a job manager:

1. Log in to the deployment manager administrative console and go to **System Environment** → **Deployment Manager**.
2. Under Additional Properties, select **Job managers**.
3. Click the **Register with Job Manager** button (Figure 2-44).

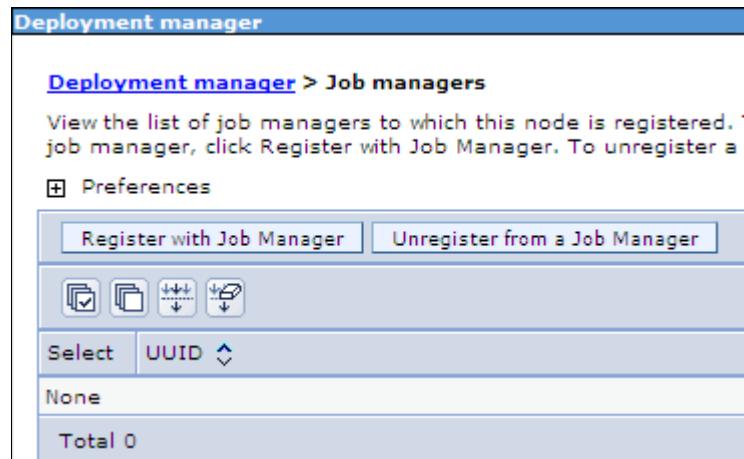


Figure 2-44 Registering a deployment manager with a job manager

4. Enter the information required to connect to the job manager, including the host name, administrative host port, and user ID and password.
If the node name you are registering is already in use by the job manager, you can enter an alias for the node.
Click **OK** (Figure 2-45).

General Properties

* Managed node name
wae06CellManager01

Alias
alias_wae06

Host name
wae01.hursley.ibm.com

Port
9943

User name
wasadmin

Password

Confirm password

OK **Reset** **Cancel**

This screenshot shows the 'General Properties' dialog box. It contains fields for managing a node: 'Managed node name' (wae06CellManager01), 'Alias' (alias_wae06), 'Host name' (wae01.hursley.ibm.com), 'Port' (9943), 'User name' (wasadmin), 'Password' (represented by six asterisks), and 'Confirm password' (also represented by six asterisks). At the bottom are 'OK', 'Reset', and 'Cancel' buttons.

Figure 2-45 Job manager details

This registers the deployment manager with the job manager.

5. To view the newly registered deployment manager, log in to the job manager console and select **Jobs** → **Nodes**. This lists the nodes and deployment managers that are registered with the job manager (Figure 2-46).

Display Resources ▾		
Select	Node name ▾	Version ▾
<input type="checkbox"/>	wae01Node03	ND 7.0.0.0
<input type="checkbox"/>	wae06CellManager01	ND 7.0.0.0
Total 2		

This screenshot shows a table titled 'Display Resources'. It has columns for 'Select' (checkboxes), 'Node name' (wae01Node03 and wae06CellManager01), and 'Version' (ND 7.0.0.0 for both). A total of 2 nodes are listed at the bottom. The table includes standard toolbar icons like copy, paste, and search.

Figure 2-46 Nodes that the job manager can administer

2.4 Managing profiles

Each profile you create is registered in a profile registry:

install_root/properties/profileRegistry.xml

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

2.4.1 Using the manageprofiles command

The **manageprofiles** command can be found in the *install_root/bin* directory.

The syntax is:

► **manageprofiles(.sh) -mode -arguments**

The modes listed in Table 2-3 are available.

Table 2-3 manageprofiles modes

Mode	Use
-create	Creates a new profile.
-delete	Deletes a profile
-augment	Augments the given profile using the given profile template
-unaugment	Unaugments the profile
-unaugmentAll	Unaugments all the profile
-deleteAll	Deletes all registered profiles.
-listProfiles	Lists the profiles in the profile registry.
-listAugments	Lists the registered augments on a profile that is in the profile registry
-getName	Returns the name of the profile at the path specified.
-getPath	Returns the path of the profile name specified.
-validateRegistry	Validates the profile registry and returns a list of profiles that are not valid
-validateAndUpdateRegistry	Validates the profile registry and lists the non-valid profiles that it purges
-getDefaultName	Returns the name of the default profile.

Mode	Use
-setDefaultName	Sets the default profile.
-backupProfile	Back ups the given profile into a zip file.
-restoreProfile	Restores the given profile from a zip file.
-response	Manage profiles from a response file
-help	Shows help.

2.4.2 Getting help

Enter **manageprofiles -mode -help** for detailed help on each mode.
 Example 2-8 shows the help information for the -create mode.

Example 2-8 Getting help for the manageprofiles command

C:\WebSphere\AppServer\bin>**manageprofiles -create -help**
 Function: Creates a new profile

Syntax:

manageprofiles -create -<argument> <argument parameter> ...

Arguments:

The following command line arguments are required for this mode:

- templatePath <argument parameter>: The fully qualified pathname of the profile template that is located on the file system.
- profileName <argument parameter>: The name of the profile.
- profilePath <argument parameter>: The intended location of the profile in the file system.

The following command line arguments are optional, and have no default values:

- isDefault <argument parameter>: Make this profile the default target of commands that do not use their profile parameter.
- omitAction <argument parameter>: Omit optional features.

Note: Command-line arguments are case sensitive.

Note: If argument accepts a parameter containing spaces, the parameter must be enclosed in "double quotes".

Note: The default profile template is "default" and may be overridden by the -templatePath switch.

Note: Each profile template will have its own set of required and optional arguments.

2.4.3 Getting a list of profiles

Enter `manageprofiles -listProfiles` to see a list of the profiles in the registry. Example 2-9 shows a sample output of `-listProfiles`.

Example 2-9 Listing profiles

```
C:\WebSphere\AppServer\bin>manageprofiles -listProfiles
[AppSrv01, Dmgr01, Custom01, AppSrv02, AdminAgent01, JobMgr01,
AppSrv03]
```

Depending on the operation used, there will be other parameters that are required, these are documented in the Information Center. To find the relevant articles, search for “`manageprofile`”.

2.4.4 Creating a profile with the `manageprofiles` command

You can use the `manageprofiles` command to create profiles.

Profile templates

Profiles are created based on templates supplied with the product. These templates are located in `install_root/profileTemplates`. Each template consists of a set of files that provide the initial settings for the profile and a list of actions to perform after the profile is created. When you create a profile using `manageprofiles`, you need to specify one of the following templates:

- ▶ Default (for application server profiles)
- ▶ Management (for deployment manager, job manager and administrative agent profiles)
- ▶ Managed (for custom profiles)
- ▶ Cell (for cell profiles)

For example, the command used to create a deployment manager with node name `TestDmgr01` under profile name `TestDmgr01` is shown in Figure 2-10.

Example 2-10 Creating a profile with the `manageprofiles` command

```
manageprofiles.bat -create -templatePath
c:/WebSphere/AppServer/profileTemplates/management -serverType
DEPLOYMENT_MANAGER -profileName TestDmgr01 -profilePath
c:/WebSphere/AppServer/profiles/TestDmgr01 -enableAdminSecurity true
-adminUserName wasadmin -adminPassword wasadmin11 -cellName TestCell01
-nodeName TestDmgr01
```

Log files that result when you run the manageprofiles command are located in:

install_root/logs/manageprofile/profilename_action.log

For example:

C:/WebSphere/AppServer/logs/manageprofiles/TestDmgr01_create.log

Additional log files are created in:

install_root/logs/manageprofile/profile_name/

For example:

C:/WebSphere/AppServer/logs/manageprofiles/TestDmgr01

Important: Do not manually modify the files that are located in the *install_root/profileTemplates* directory.

Options for specifying ports

During profile creation using the manageprofiles command, you can accept the default port values, or you can specify your port settings. If you want to specify ports, you can do so in any of the following ways:

- ▶ Specify the use of a port file that contains the port values.
- ▶ Specify the use of a starting port value.
- ▶ Specify the use of the default port values.

Port file that contains the port values

You can supply a file containing the port values that you want to use for any profile using the **-portsFile** option. See Example 2-11 for an example ports file named portdef.props.

Example 2-11 Example contents of portdef.props file

```
WC_defaulthost=39080 WC_adminhost=39060 WC_defaulthost_secure=39443
WC_adminhost_secure=39043 BOOTSTRAP_ADDRESS=32809 SOAP_CONNECTOR_ADDRESS=38880
IPC_CONNECTOR_ADDRESS=39633 SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=39401
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=39403
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=39402 ORB_LISTENER_ADDRESS=39100
DCS_UNICAST_ADDRESS=39353 SIB_ENDPOINT_ADDRESS=37276
SIB_ENDPOINT_SECURE_ADDRESS=37286 SIB_MQ_ENDPOINT_ADDRESS=35558
SIB_MQ_ENDPOINT_SECURE_ADDRESS=35578 SIP_DEFAULTHOST=35060
SIP_DEFAULTHOST_SECURE=35061
```

Incrementing port numbers from a starting point

The `manageprofiles` command can assign port numbers based on a starting port value. You can provide the starting port value from the command line, using the `-startingPort` parameter. The command assigns port numbers sequentially from the starting port number value. However, if a port value in the sequence conflicts with an existing port assignment, the next available port value is used.

The order of port assignments is arbitrary. Predicting assignments is not possible.

Use the `-startPort` option for the `manageprofiles` command

Default ports

Assigns the default or base port values to the profile. Use the `-defaultPorts` option to the `manageprofile` command.

Changing port settings after profile creation

Use the `updatePorts` tool to change port settings.

For more information, read the article, “Updating ports in an existing profile” at:

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

2.4.5 Creating a profile in silent mode with PMT

Profiles can also be created in silent mode using a response file. The command to use is:

`pmt(.sh) -options response_file -silent`

The command to start the wizard is platform-specific and is located in `install_root/bin/ProfileManagement`.

2.4.6 Deleting profiles

To delete a profile, do the following actions:

- ▶ If you are removing a custom profile or application server profile that has been federated to a cell:
 - Stop the application servers on the node.
 - Remove the node from the cell using the administrative console or the `removeNode` command. Removing a node does not delete it, but restores it to its pre-federated configuration that was saved as part of the federation process.

- Delete the profile using `manageprofiles -delete -profileName profile_name`
- Use the `manageprofiles -validateAndUpdateRegistry` command to clean the profile registry.
- Delete the `profile_root` directory.
- ▶ If you are removing an application server profile that has not been federated to a cell:
 - Stop the application server.
 - Delete the profile using `manageprofiles -delete -profileName profile_name`
 - Use the `manageprofiles -validateAndUpdateRegistry` command to clean the profile registry.
 - Delete the `profile_root` directory.
- ▶ If you are removing a deployment manager profile:
 - Remove any nodes federated to the cell using the administrative console or the `removeNode` command. Removing a node does not delete it, but restores it to its pre-federated configuration that was saved as part of the federation process.
 - Stop the deployment manager.
 - Delete the profile using `manageprofiles -delete -profileName profile_name`
 - Use the `manageprofiles -validateAndUpdateRegistry` command to clean the profile registry.
 - Delete the `profile_root` directory.

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

`install_root/logs/manageprofile/profilename_delete.log`

For example, in Example 2-12, you can see the use of the `manageprofiles` command to delete the profile named Node06.

Example 2-12 Deleting a profile using manageprofiles

```
C:\WebSphere\ND\profiles\dmgr01\bin>manageprofiles -delete -profileName
Node06
INSTCONFSUCCESS: Success: The profile no longer exists.
```

As you can see, all seems to have gone well. But, as an additional step to ensure the registry was properly updated, you can list the profiles to ensure that the profile is gone from the registry, and validate the registry. See Example 2-13.

Example 2-13 Verifying the delete profile results

```
C:\WebSphere\ND\profiles\dmgr01\bin>manageprofiles -listProfiles
[Dmgr01, AppSrv01, AppSrv02, SamplesServer, WebServer2Node, DmgrSecure]
```

```
C:\WebSphere\ND\profiles\dmgr01\bin> manageprofiles
-validateAndUpdateRegistry
[]
```

Note: If there are problems during the delete, you can manually delete the profile. For information about this, see the topic, *Deleting a profile*, in the Information Center:

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



Working with profiles on z/OS systems

The purpose of this chapter is to help you build your initial WebSphere Application Server environment using the new WebSphere Customization Tools (WCT).

We cover the following topics:

- ▶ “Creating WebSphere environments” on page 114
- ▶ “Creating a deployment manager definition” on page 124
- ▶ “Creating the base application server definition” on page 155
- ▶ “Federating an application server” on page 177
- ▶ “Creating a job manager profile” on page 183
- ▶ “Creating an administrative agent profile” on page 193

3.1 Creating WebSphere environments

Configuring a WebSphere Application Server for z/OS environment consists of setting up the configuration directory for the environment and making changes to the z/OS target system that pertain to the particular application serving environment. Configuring these application serving environments after product installation requires a fair amount of planning and coordination.

For example, when defining multiple deployment managers or application servers on a single machine or LPAR, you need to ensure that the ports and names you select for each are unique and the z/OS environment variables, generated jobs, and so on, are all set up properly. We strongly recommend that you spend time planning the installation and if possible, first practice by configuring a stand-alone application server using the default options.

You use the WebSphere Application Server for z/OS Profile Management Tool (PMT) available with the WebSphere Customization Tools (***new with V7***) to configure WebSphere Application Server environments for the z/OS platform.

The PMT is a dialog-driven tool that runs in the WebSphere Customization Tools. It is an Eclipse plug-in that allows you to perform the initial setup of WebSphere Application Server for z/OS cells and nodes. It provides the same functionality as the former ISPF dialogs plus additional features to help you.

The WCT itself does not create the cells and nodes; however, it creates batch jobs, scripts, and data files that you can use to perform WebSphere Application Server for z/OS customization tasks. These jobs, scripts, and data files form a *customization definition* on your workstation, which is then uploaded to z/OS where you submit the jobs as pictured in Figure 3-1.

The WCT for z/OS is a workstation graphical tool that is used to generate the customized jobs used to create, migrate or augment a WAS z/OS configuration

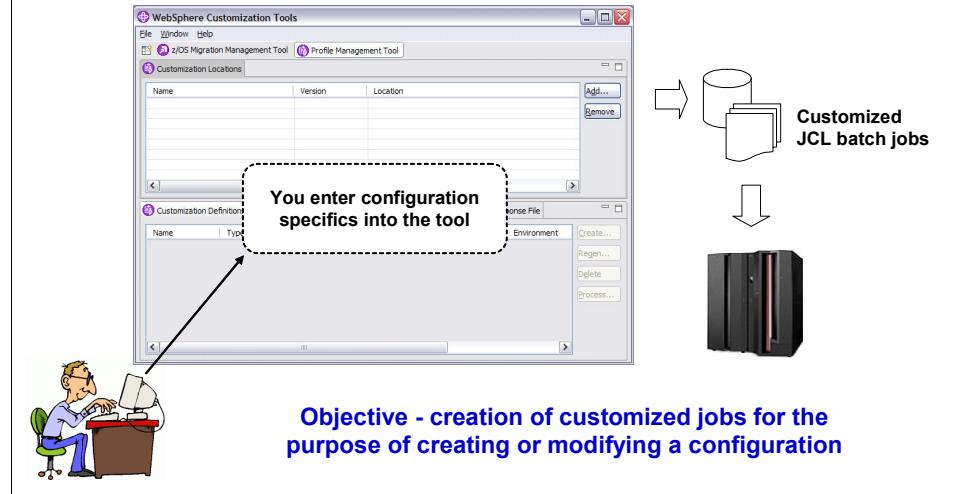


Figure 3-1 The WCT configuration flow

Note: The ISPF dialogs were deprecated in WebSphere Application Server Version 6.1 and as of WebSphere Application Server Version 7.0, the ISPF panels are not longer available.

Review the documentation:

The WebSphere Application Server information center contains planning topics for each WebSphere Application Server package that is tailored to each platform. This section gives you a high level look at the planning tasks you need to perform.

If you are planning a WebSphere Application Server for z/OS environment, we strongly suggest that you review the installation planning material for z/OS platforms available at the WebSphere Application Server Information Center at the following URL:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.installation.zseries.doc/info/zseries/ae/welc6topinstalling.html>

The PMT requires that you have the WebSphere Customization Tools (WCT) application. For more information about how to download and install the WCT, see the following URL:

<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg24020368>

For additional information about the function of the WCT, see the following document: *Introducing the WCT for WebSphere z/OS Version 7*, found at:

<http://www-03.ibm.com/support/techdocs/atstrmstr.nsf/WebIndex/PRS335>

A spreadsheet has been developed that can help you create and document your configuration variables. This spreadsheet can be downloaded from the following URL. Many of the values that you need to specify to the PMT tool can be planned using this spreadsheet:

<http://www-03.ibm.com/support/techdocs/atstrmstr.nsf/WebIndex/PRS3341>

Through the PMT, you can create environments for the following products:

- ▶ WebSphere Application Server for z/OS
- ▶ WebSphere DMZ secure proxy server for z/OS

3.1.1 WebSphere Application Server for z/OS

The PMT provides support to generate jobs to create the following WebSphere Application Server for z/OS environments:

- ▶ Create a cell environment consisting of a deployment manager and a federated application server.
- ▶ Create a management environment, which can be either:
 - A deployment manager
 - A job manager (*New in V7*)
 - An administrative agent (*New in V7*)
- ▶ Create a standalone application server environment.
- ▶ Create a managed (custom) node and federate it into an existing cell.
- ▶ Federate an application server.

3.1.2 WebSphere DMZ secure proxy server for z/OS

The DMZ secure proxy server for IBM WebSphere Application Server installation allows you to install your proxy server in the demilitarized zone (DMZ), while reducing the security risk that might occur if you choose to install an application server in the DMZ to host a proxy server. The risk is reduced by removing any functionality from the application server that is not required to host the proxy servers, but that could pose a security risk. Installing the secure proxy server in the DMZ rather than the secured zone presents new security challenges. However, the secure proxy server is equipped with capabilities to provide protection from these challenges.

The PMT Version 7.0 provides support for the following WebSphere DMZ secure proxy server for z/OS environments:

- ▶ Management:
Generates the customization jobs to create an administrative agent for the secure proxy server.
- ▶ Secure proxy:
Generates the customization jobs to create a secure proxy server.

3.2 Getting started with the profile management tool

This section explains how to prepare and start the PMT. These steps are common for any type of profile.

To start the PMT, perform the following steps:

1. Start the WebSphere Customization Tools (WCT).

On Windows, you can start it from the Windows Start menu:

Start → All Programs → IBM WebSphere → WebSphere Customization Tools V7.0 → WebSphere Customization Tools

On Linux, use the menus used to start your programs.

For example, click ***operating_system_menus_to_access_programs* → IBM WebSphere → WebSphere Customization Tools V7.0.**

2. Click the **Welcome** tab. You are presented with the Welcome window shown in Figure 3-2.

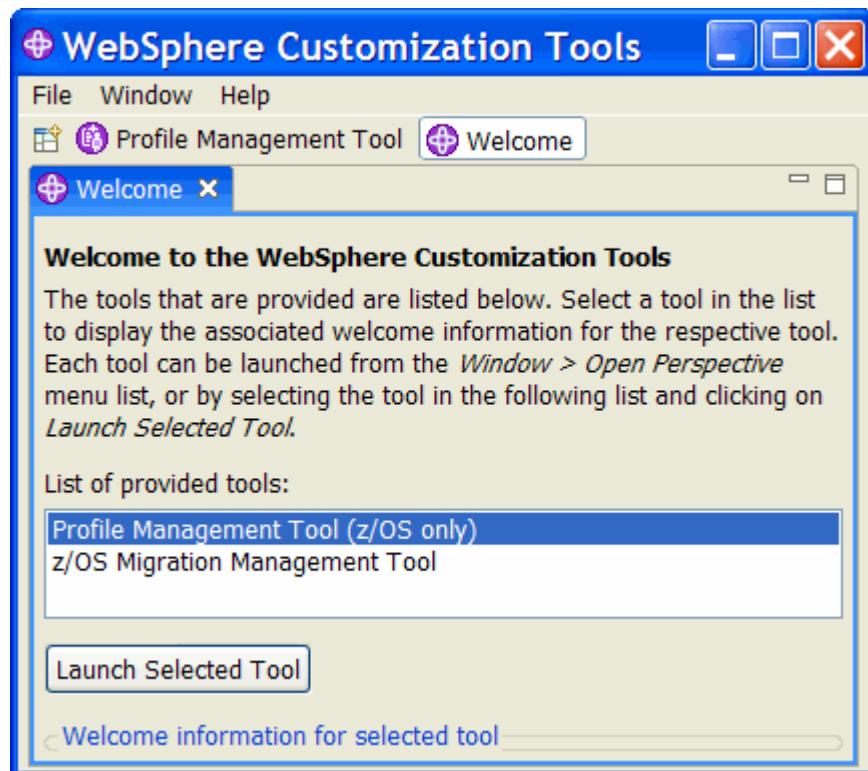


Figure 3-2 WebSphere Customization Tools welcome

3. Select the **Profile Management Tool (z/OS only)** option and click the button, **Launch Selected Tool**. You are presented with the PMT panel as shown in Figure 3-3.

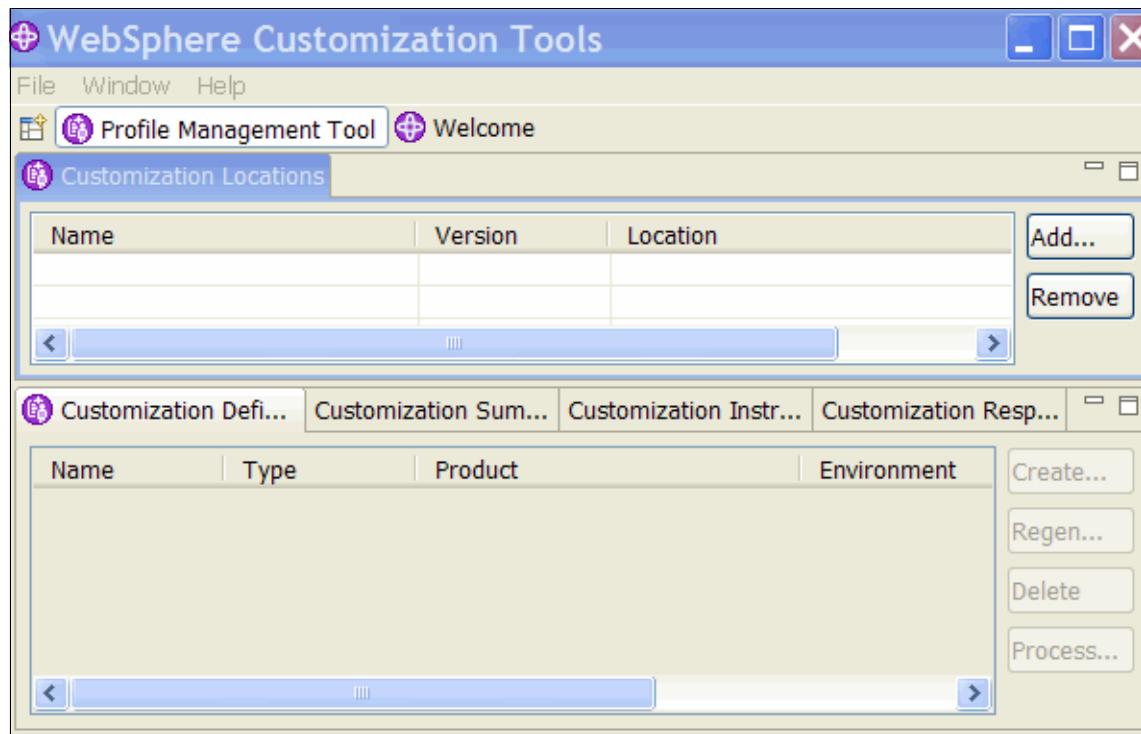


Figure 3-3 Initial Profile Management Tool panel

4. Create a new *location* by clicking the **Add** button (top right corner). The definitions containing the generated customization jobs will be stored at the location you define here:
 - a. Select **Create a new customization location** and give a name to the new *location* as shown in Figure 3-4.

Note: The PMT is compatible with WebSphere Application Server Version 6.1, therefore for a single location you can create definitions for either WebSphere Application Server V6.1 or WebSphere Application Server V7.0.

- b. Select **Version 7.0** and the corresponding folder for this *location*.

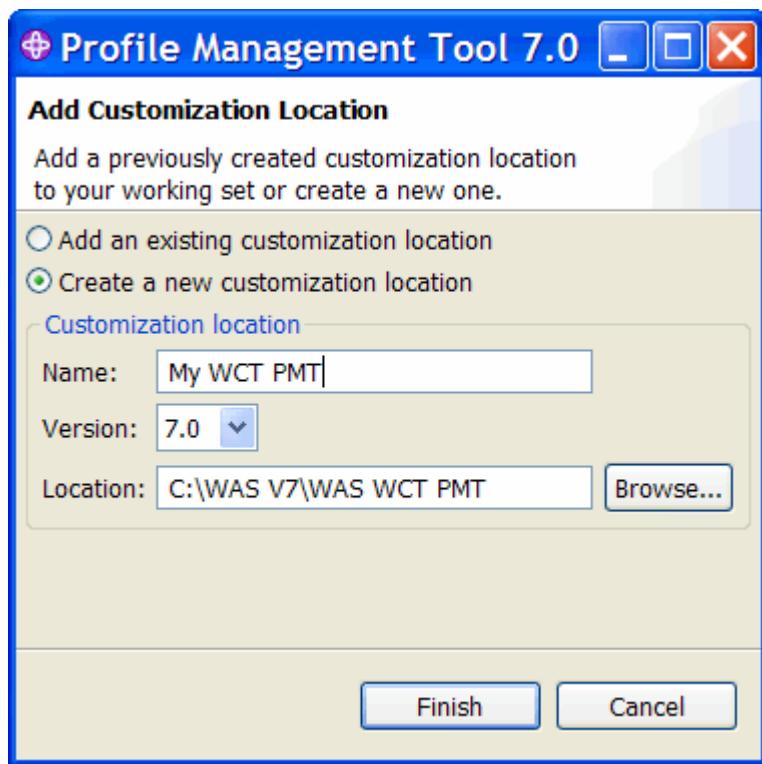


Figure 3-4 Create the customization location

- c. Click **Finish** to create this *location*.

The Profile Management Tool panel opens (Figure 3-5).

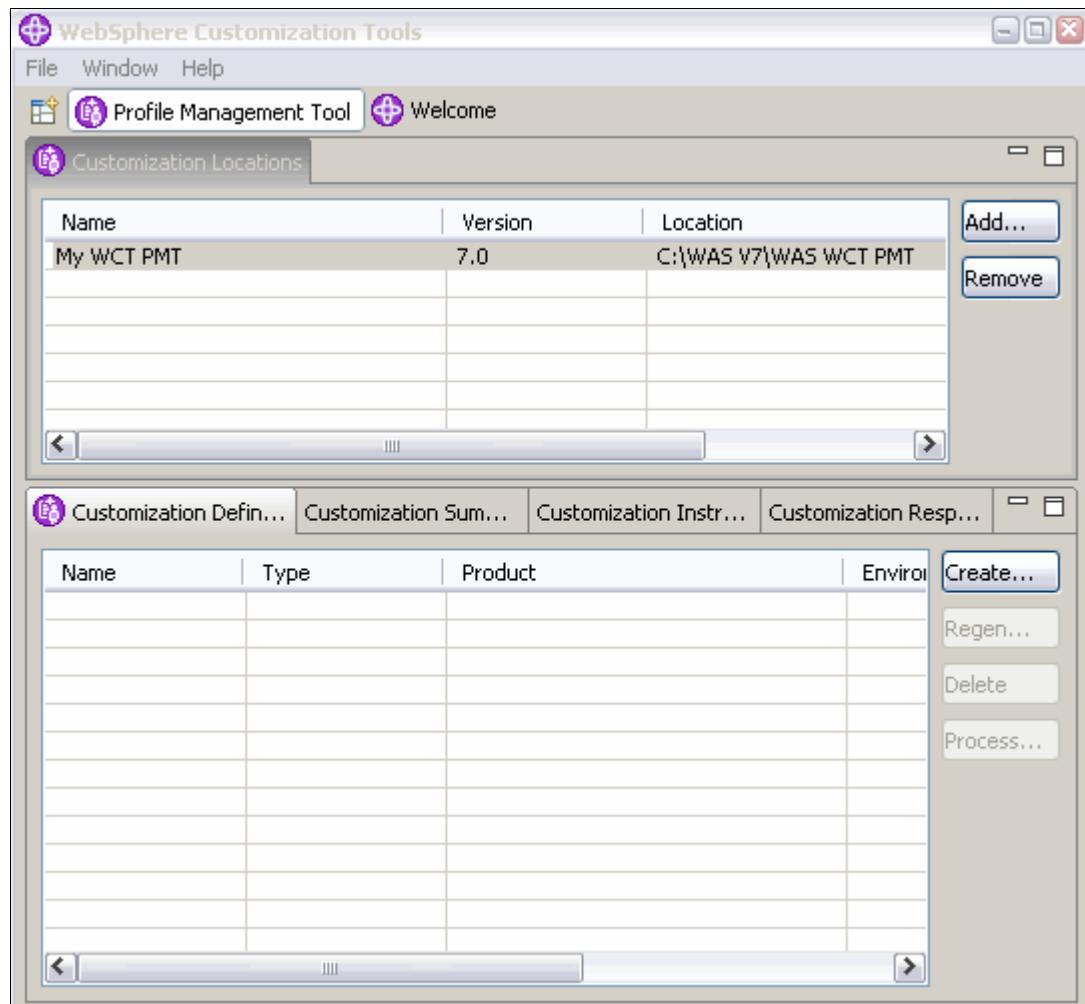


Figure 3-5 Profile management tool main panel

Using response files:

Each time you create a customization definition, it is stored in a directory in the selected customization location. A corresponding response file is generated in the same directory.

You can display this response file by switching to the Customization Response File tab in the previous panel (Figure 3-5). See the last tab in the bottom portion of the panel. You can use this response file when creating future profiles to populate the input fields with values contained in the response file.

Configure Additional Users:

If your daemon and control region adjunct processes have to run using different user IDs from the associated control region process, you should enable the following setting:

Click **Window** → **Preferences** → **Profile Management Tool** in WebSphere Customization Tools Version 7.0.0.1 or later, select **Enable unique user IDs for daemon and adjunct**, and click **Apply**.

With this setting, you get an additional panel when you build a customization definition that allows you to specify additional user IDs for processes relevant to the profile (that is, for the controller adjunct and daemon processes).

- To begin a new customization definition, click **Create**. You are now presented with the panel (Figure 3-6) welcoming you to the Profile Management tool.

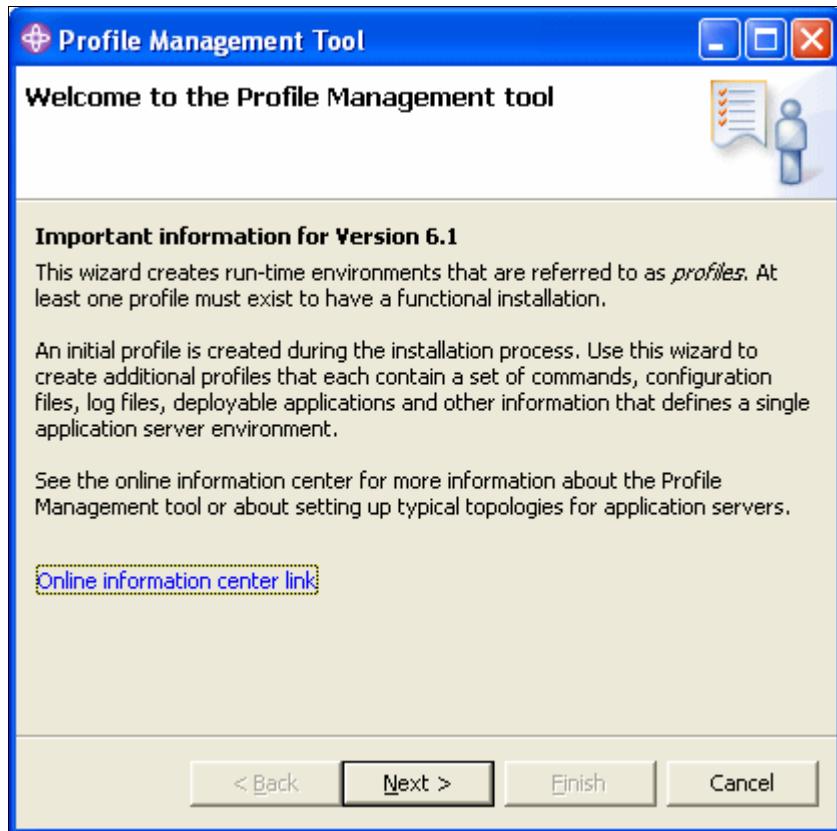


Figure 3-6 PMT Welcome panel

- Click **Next** to get started with creating profiles.

3.3 Sample environment

In this section we demonstrate how to use the PMT to create a z/OS Network Deployment cell (deployment manager and application server). We chose this configuration because it is representative and inclusive of the panels and steps that you will encounter with the other environments.

Figure 3-7 illustrates the flow that you go through in order to generate the jobs for this configuration.

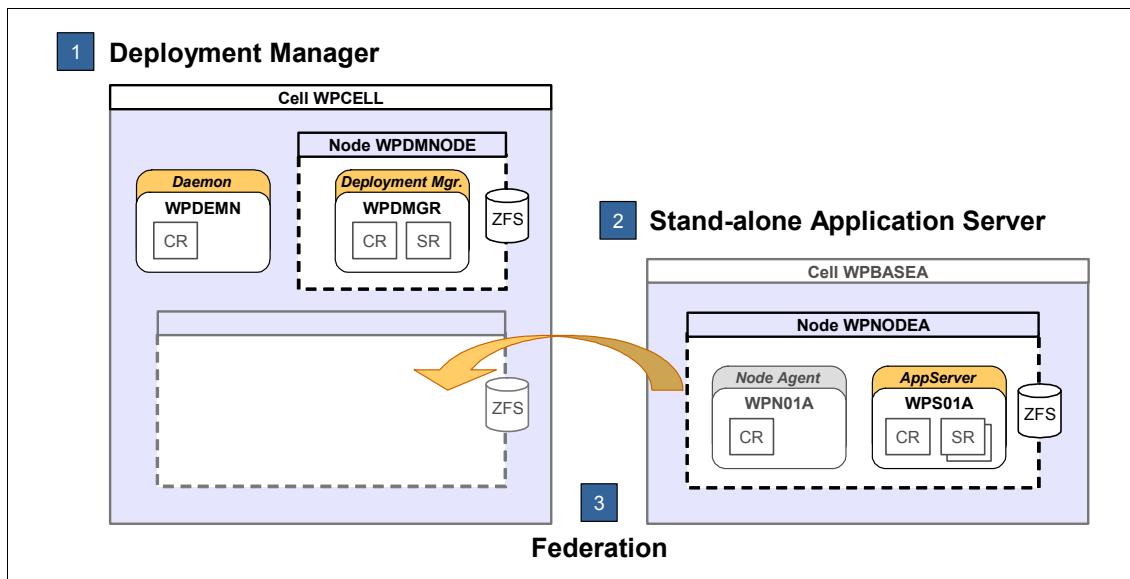


Figure 3-7 Our configuration objectives

We build the environment in three steps:

1. Create a deployment manager profile
2. Create an application server profile
3. Federate the application server to the deployment manager cell.

3.4 Creating a deployment manager definition

The first step in building the sample WebSphere environment is to create the deployment manager.

3.4.1 Creating the customization definition

To begin, run the profile management tool to create the custom definition:

1. Click **Create** on the Profile Management Tool main panel.

The first panel that you see when you click **Next** on the Profile Management Tool Welcome panel prompts you to select the type of WebSphere environment that you want to create.

2. Select **Management** and click **Next** (Figure 3-8).



Figure 3-8 Environment selection panel

3. Select **Deployment manager** and click **Next**.
4. The next window, Figure 3-9, contains the following fields:
 - Customization definition name:
Used to specify the customization profile you are about to create. This name is not transported to your host system.
 - Response file path name:
Allows you to specify a saved file with values from a previously created configuration. Doing this populates the fields throughout the windows with the values that are contained in the response file. This field is optional.
Because it is the first time that you will be creating a profile, you probably do not have this file. A response file is written each time a z/OS customization definition is created, and its name is the customization definition name itself + `.responseFile` created under the root directory for the customization definition. Normally, you should specify a response file from a customization definition of the same type as you are about to define. However, a response file from a similar customization type can be used to pre-load most of the default values.

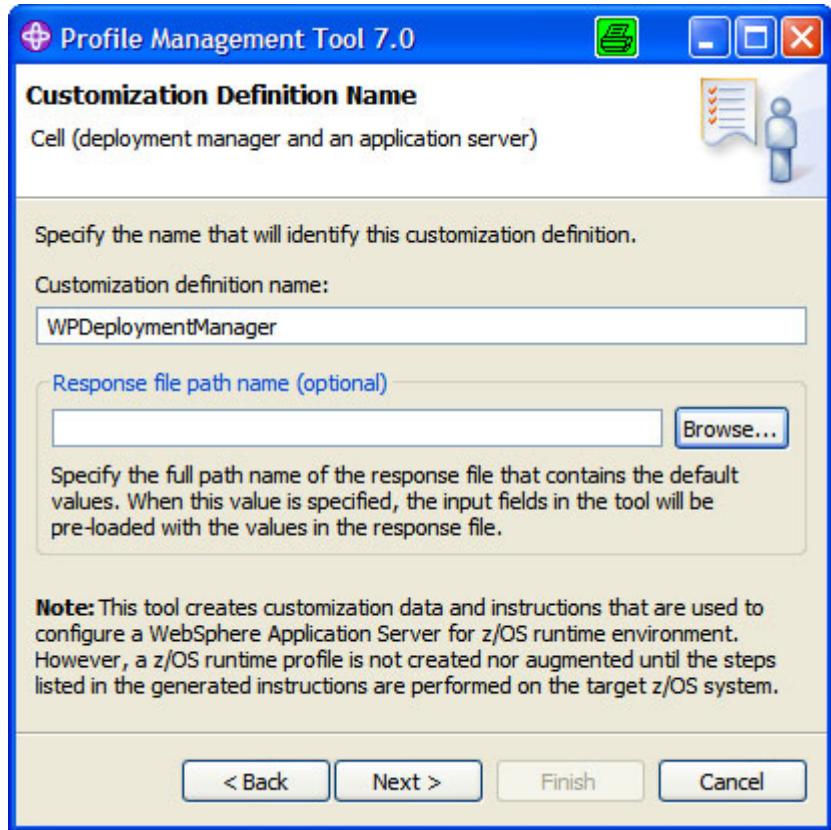


Figure 3-9 Customization definition

Click **Next**.

5. The next panel (Figure 3-10) allows you to specify defaults for GID and UID values, name and user ID defaults based on a two character prefix that identifies the cell, and allows you to specify a default range for ports assigned to the process.

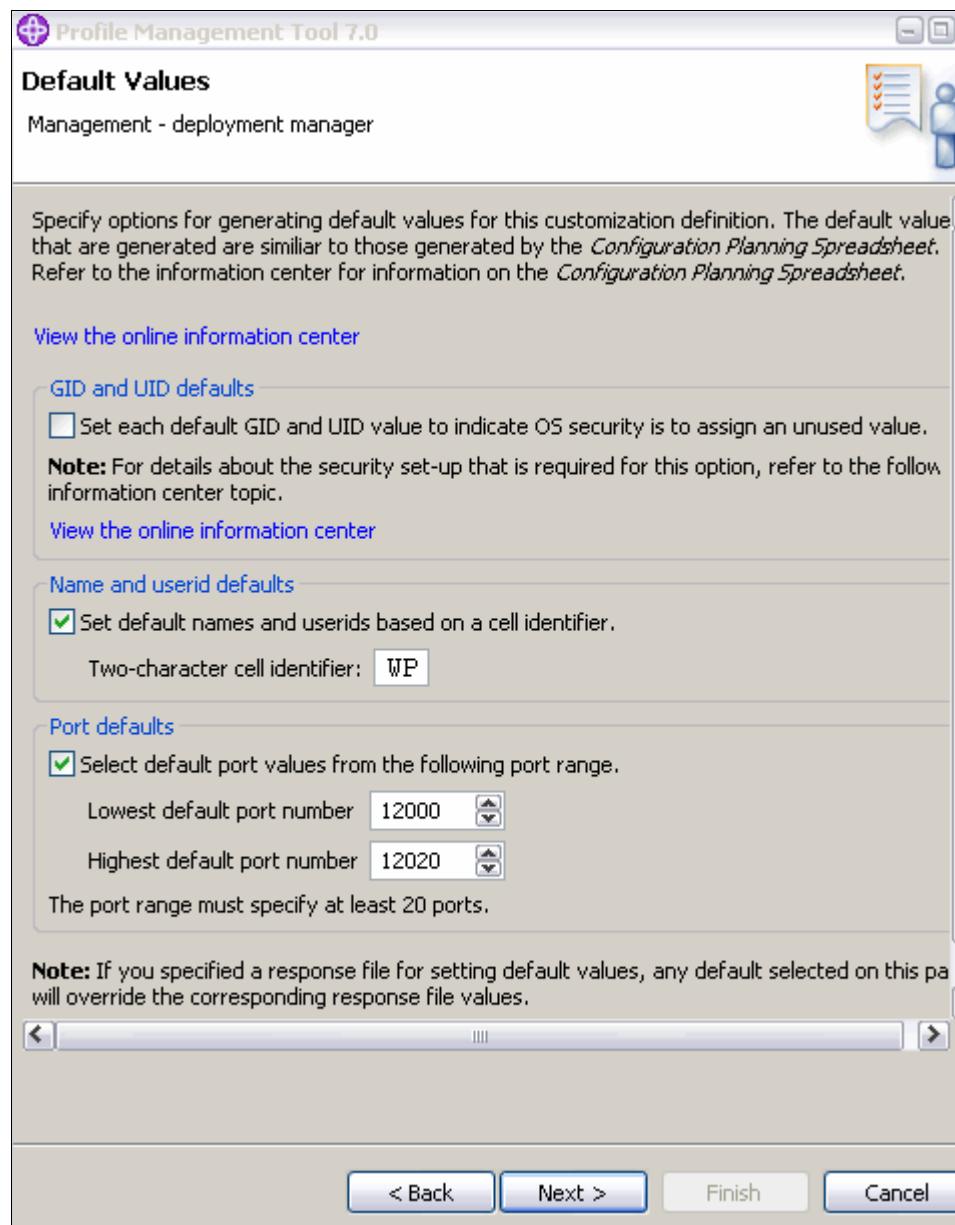


Figure 3-10 Specify default values

Click **Next**.

6. The next panel (Figure 3-11) asks you to specify a high level qualifier for the target z/OS data sets that will contain the generated jobs and instructions.

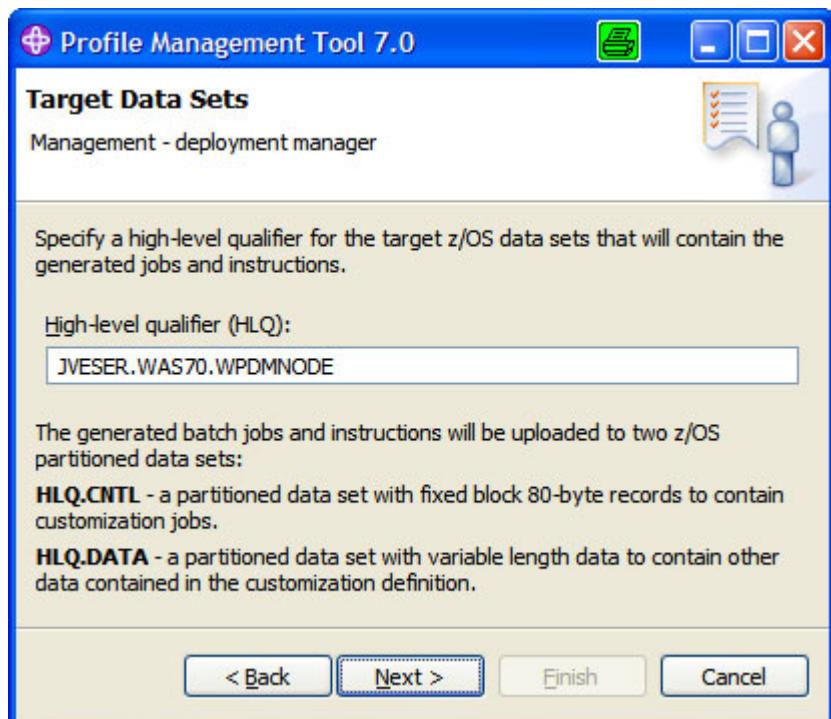


Figure 3-11 Creating a deployment manager profile: target data sets window

The high level qualifier can be composed of multiple qualifiers up to 39 characters. When a customization profile is uploaded on the target z/OS system, the generated jobs and files are written on a pair of data sets. The same data sets can be reused for a future installation; however, we strongly recommend you create a new pair of data sets for every new profile installation.

A good planning and naming convention is crucial when defining this type of information. As a best practice, try to set the high level qualifier according to the version and release of WebSphere Application Server for z/OS, the task you are performing, and the cell (and, in some cases, the node name) you are configuring.

In this case, the following data sets will be created when the customization profile is uploaded to the target z/OS system:

- JVESER.WAS70.WPDMNODE.CNTL
- JVESER.WAS70.WPDMNODE.DATA

The CNTL data set is a partitioned data set with a fixed block 80-byte records that keeps the customization jobs. The DATA data set is a partitioned data set as well, but with variable length data to contain the other customization data.

Click **Next**.

Note: After the customization profile is created, the data set names cannot be changed, because all jobs are based on these data set names.

7. The next panel (Figure 3-12) contains the fields to configure common groups.

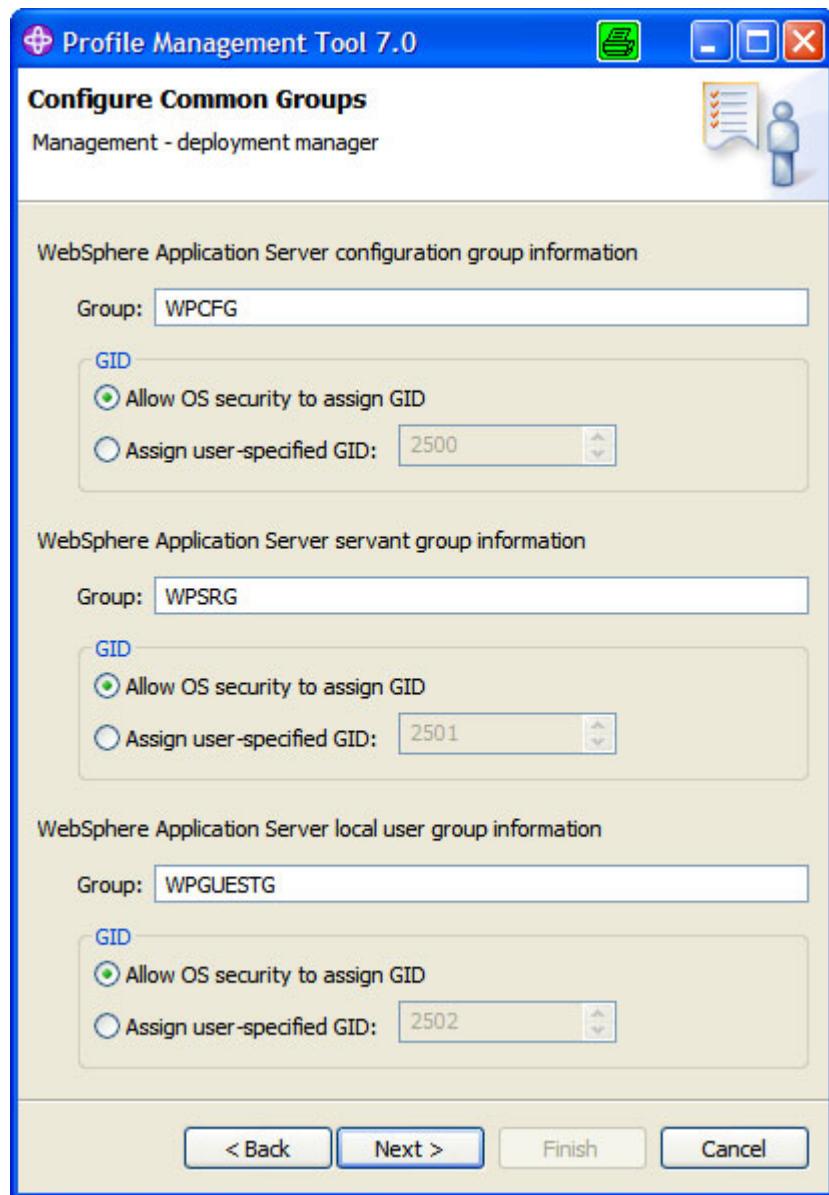


Figure 3-12 Deployment manager: Configure common groups

The information that you need to provide for this panel is as follows:

- WebSphere Application Server Configuration Group Information:
Used to specify the group name for the WebSphere Application Server administrator user ID and all server user IDs.
- WebSphere Application Server Servant Group Information:
Used to connect all servant user IDs to this group. You can use it to assign subsystem permissions, such as DB2® authorizations, to all servants in the security domain.
- WebSphere Application Server Local User Group Information:
Specify the local client group. This group provides minimal access to the cell.
- WebSphere Application Server user ID home directory:
Specify a new or existing z/OS file system directory in which home directories for WebSphere Application Server for z/OS user IDs will be created by the customization process. Note that this directory does not need to be shared among z/OS systems in a WebSphere Application Server cell.

Click **Next**.

8. The next panel (Figure 3-13) configures the common users.

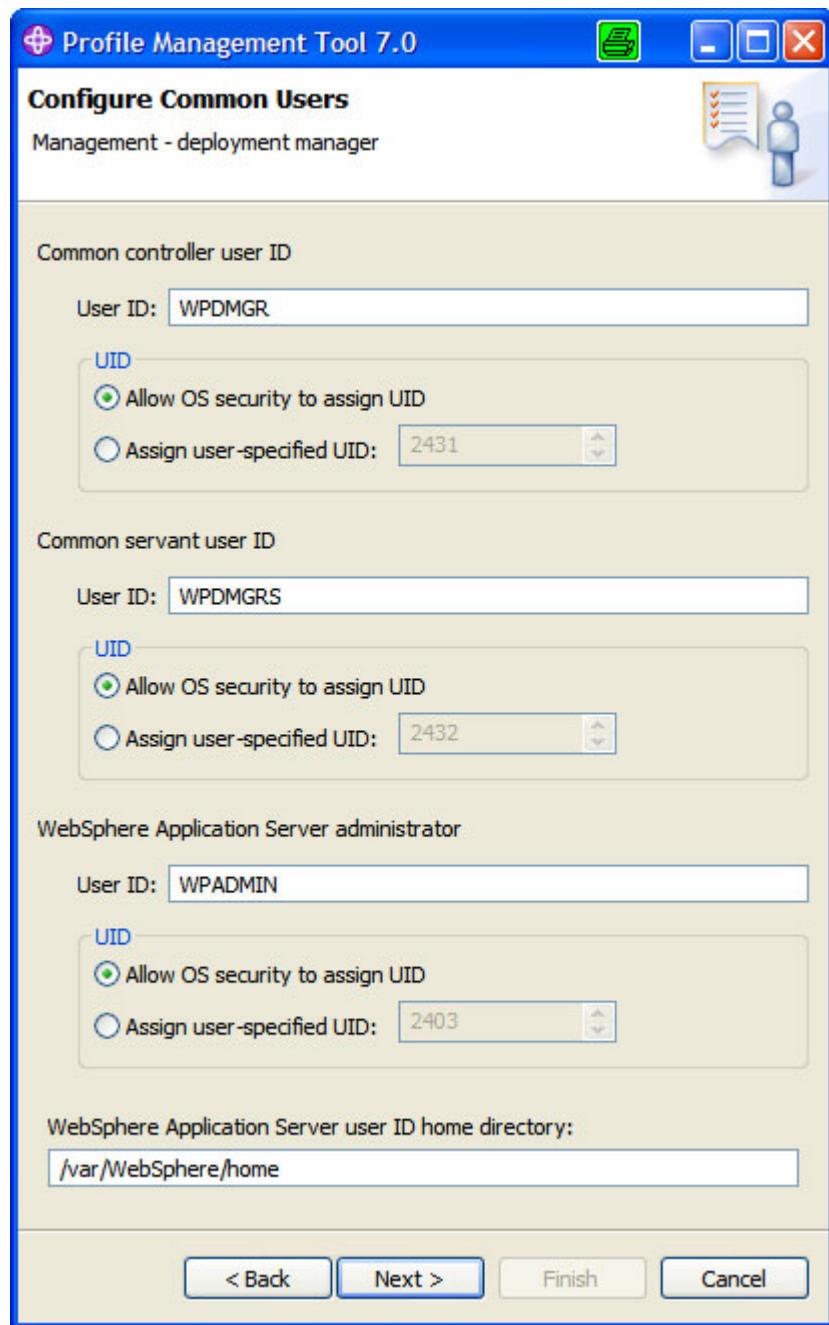


Figure 3-13 Common users

- Common controller user ID:
The user ID associated with all the control regions and the daemon. This user ID will also own all of the configuration file systems.
- Common servant user ID:
The user ID associated with the servant regions.
- WebSphere Application Server administrator user ID:
The initial WebSphere Application Server administrator. The ID must have the WebSphere Application Server configuration group as its default UNIX System Services group. The UNIX System Services UID number for the administrator user ID is specified here, and must be a unique numeric value between 1 and 2,147,483,647.

Click **Next**.

9. The next panel (Figure 3-14) requests system and data set names.

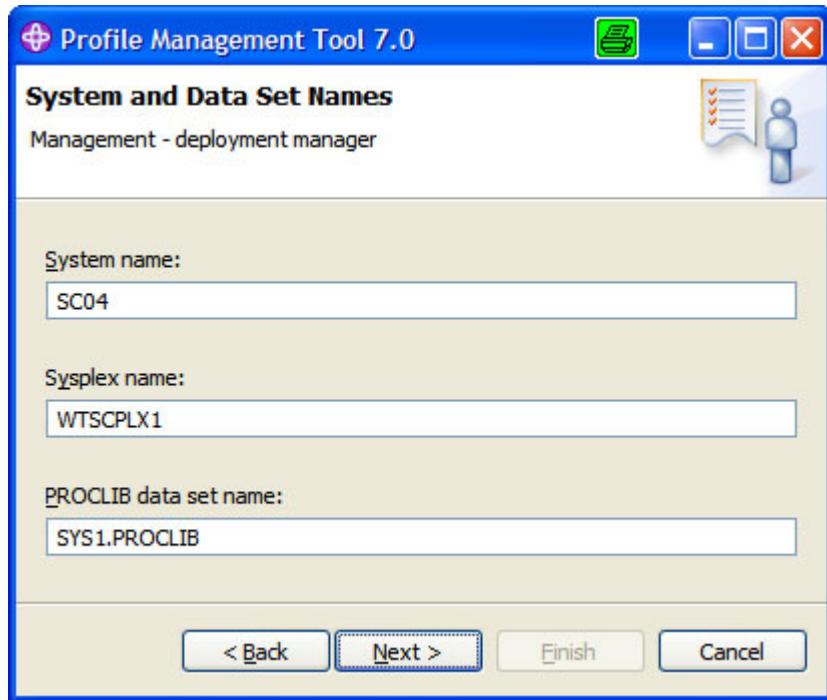


Figure 3-14 Creating a deployment manager profile: System and Dataset names

- System name: The system name of the target z/OS system.
- Sysplex name: The sysplex name of the target z/OS system.

Note: If you are not sure of the system and sysplex names for your target z/OS system, you can use the console command, D SYMBOLS, on the target z/OS system to display them.

- PROCLIB data set name: The PROCLIB data set where the WebSphere Application Server for z/OS cataloged procedures are to be added.

Click **Next**.

10. The next panel (Figure 3-15) requests the cell, node, and server names.

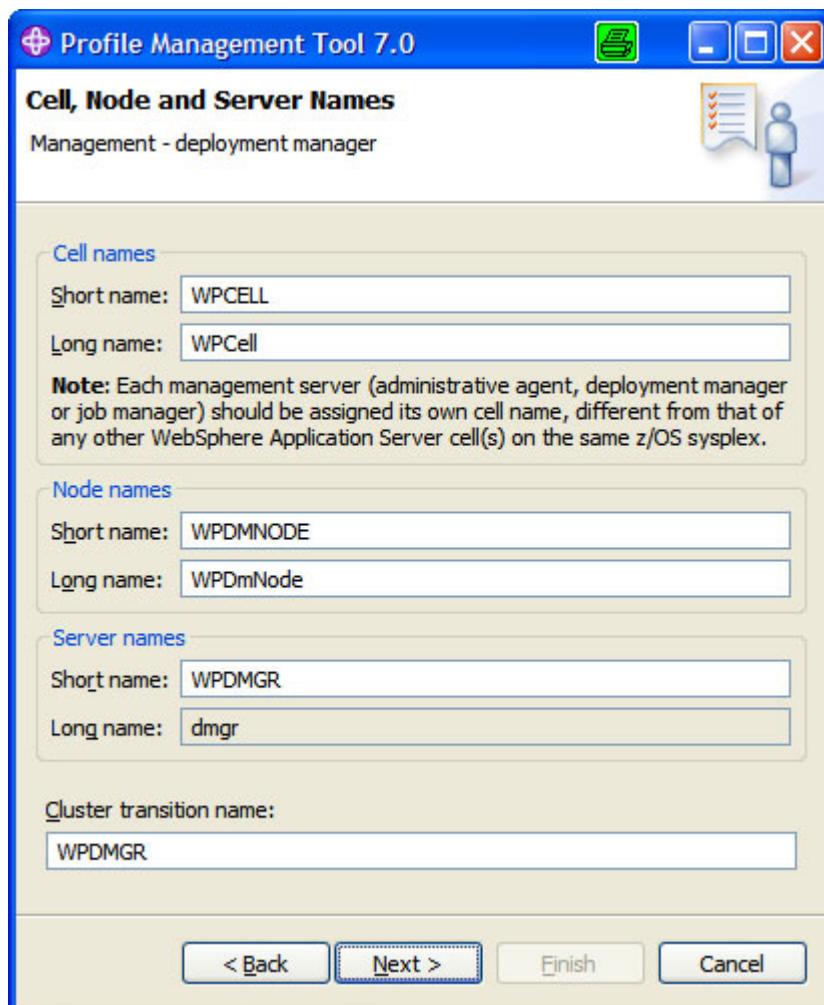


Figure 3-15 Deployment manager - cell, node and server names

- Cell short name:
Identifies the cell to z/OS facilities, such as SAF.
- Cell long name:
The primary external identification of this WebSphere Application Server for this z/OS cell. This name identifies the cell as displayed through the administrative console.
- Deployment manager short name:
The name that identifies the node to z/OS facilities, such as SAF.
- Deployment manager long name:
The primary external identification of this WebSphere Application Server for the z/OS node. This name identifies the node as displayed through the administrative console.
- Deployment manager server short name:
Identifies the server to z/OS facilities, such as SAF. The server short name is also used as the server JOBNAME.
- Deployment manager server long name:
The name of the application server and the primary external identification of this WebSphere Application Server for the z/OS server. This name identifies the server as displayed through the administrative console.
- Deployment manager cluster transition name:
The WLM application environment (WLM APPLENV) name for the deployment manager. If this is a server that is converted into a clustered server, this name becomes the cluster short name. The cluster short name is the WLM APPLENV name for all servers that are of the same cluster.

Click **Next**.

New in V7: With Version 7.0, the WebSphere Application Server LOADLIBs are located in the product filesystem instead of separate data sets as in Version 6.1. Therefore the STEPLIBs are not needed anymore. Consequently, in Version 7.0, the Z-member in PROCLIB disappears.

11. The next window (Figure 3-16) requests configuration file system information for your z/OS system. The file system can be either HFS or zFS. It is used to hold WebSphere Application Server configuration information.

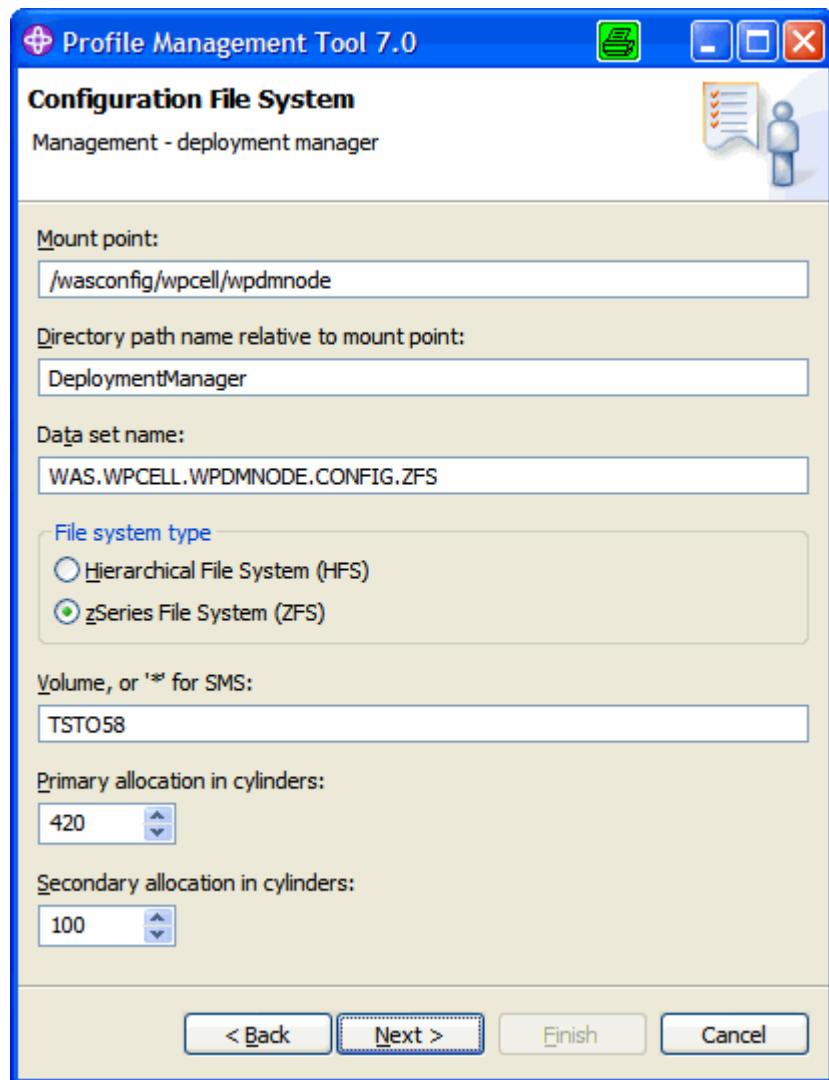


Figure 3-16 Deployment manager - configuration file system

- Mount point:
The read/write HFS directory where application data and environment files are written. The customization process creates this mount point if it does not already exist.
- The directory path name relative to the mount point:
Directory to be used for the deployment manager home directory.
- Data set name:
The file system data set you will create and mount at the specified mount point above.
- Volume, or '*' for SMS:
Specify either the DASD volume serial number to contain the above data set or "*" to let SMS select a volume. Using "*" requires that SMS automatic class selection (ACS) routines be in place to select the volume. If you do not have SMS set up to handle data set allocation automatically, list the volume explicitly.
- Primary allocation in cylinders:
The initial size allocation for the configuration file system data set. In the application server, the total space needed for this data set increases with the size and number of the installed applications. The minimum suggested size is 250 cylinders (3390).
- Secondary allocation in cylinders:
The size of each secondary extent. The minimum suggested size is 100 cylinders.
- File System type:
Select to allocate and mount your configuration file system data set using as either HFS or zFS.

Click **Next**.

1. The next panel (Figure 3-17) defines the product file system directory and allows you to set up an intermediate symbolic link (see Chapter 7, “WebSphere Application Server for z/OS” on page 381 for more information).

In the past, the intermediate symbolic links had to be set up manually.

Recommendation: Use intermediate symbolic links for flexibility when applying maintenance and running with different versions of the code.

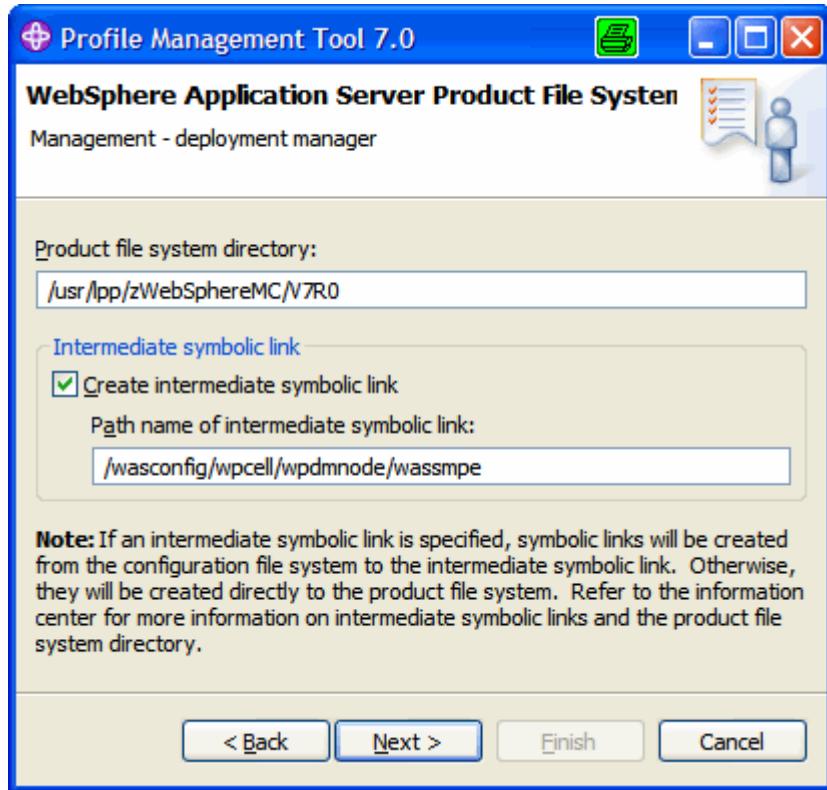


Figure 3-17 Deployment manager - WebSphere Application Server product file system

Click **Next**.

2. Enter the job names, procedure names, and user IDs to use for each process in the panel shown in Figure 3-18.

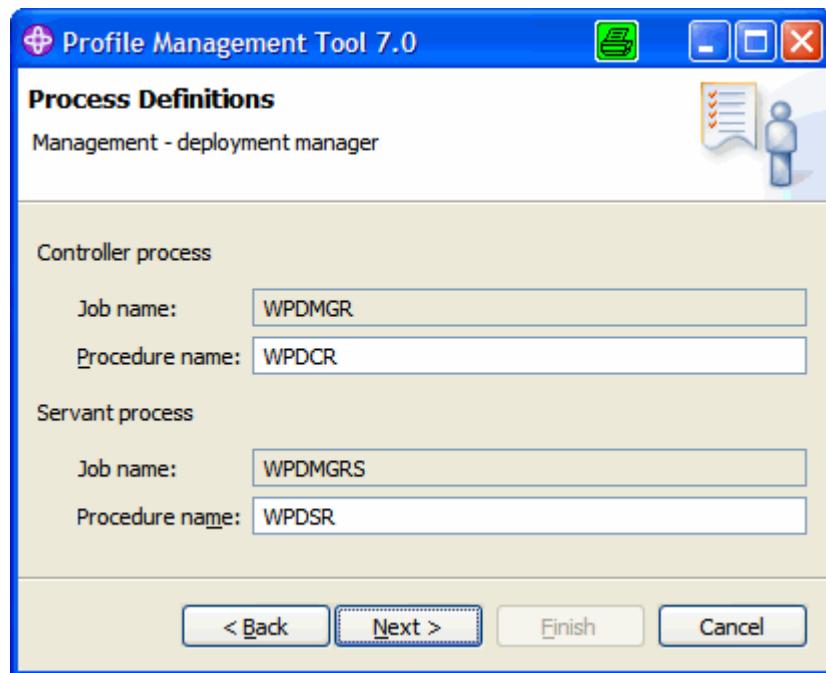


Figure 3-18 Deployment manager - processes

– Deploy manager controller process:

The job name is specified in the MVS START command JOBNAM parameter, associated with the control region. This is the same as the server short name and it cannot be changed during customization. The procedure name is the member name in your procedure library to start the control region. The User ID is the user ID associated with the control region.

– Deploy manager servant process:

Specify the job name used by WLM to start the servant regions. This is set to the server short name, followed by the letter "S," and it cannot be changed during customization. The procedure name is the member name in your procedure library to start the servant regions. The User ID is the user ID associated with the servant regions.

Click **Next**.

3. The next panel (Figure 3-19) requires you to specify the ports to use for each process. Planning is very important to avoid port conflicts, so be sure that you have all values you need in order to complete this panel.

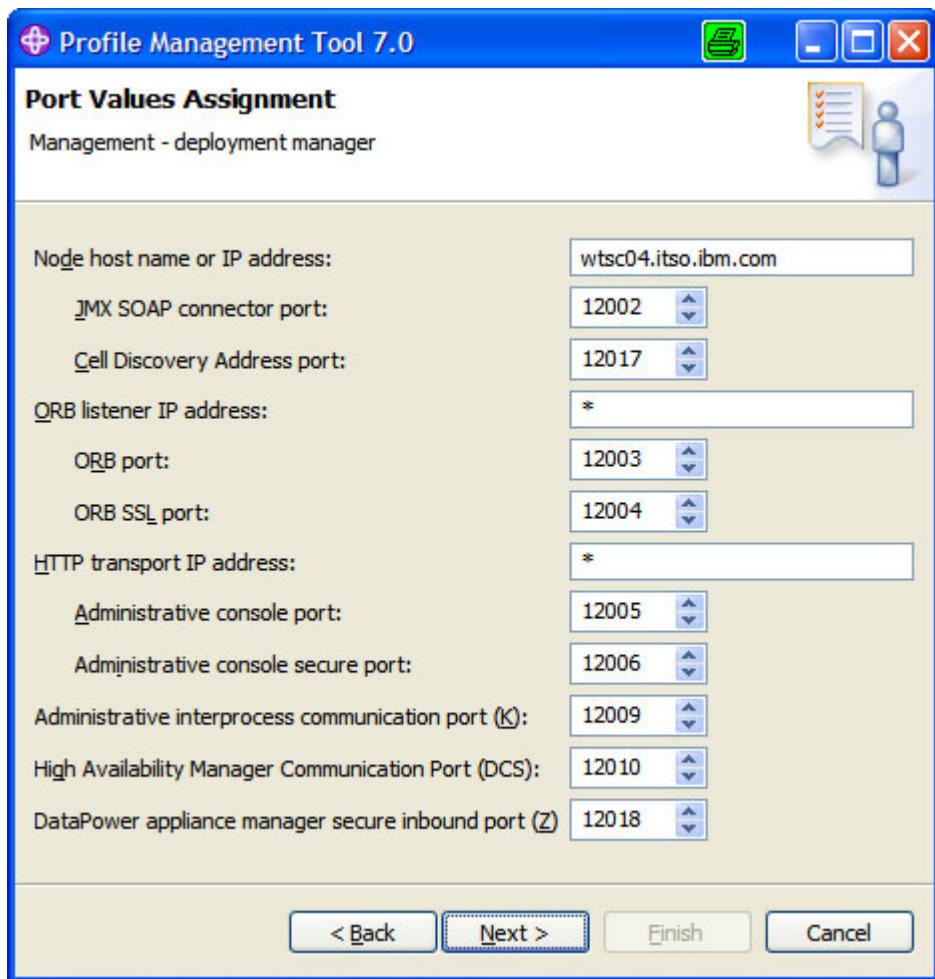


Figure 3-19 Deployment manager - port values

Comparing the ports with V6.1, two additional ports have been added—the Administrative interprocess communication port as well as the DataPower® appliance manager secure inbound port.

After completing the required ports, click **Next**.

4. The next panel (Figure 3-20) requests the location service daemon settings. The location daemon service is the initial point of client contact in WebSphere Application Server for z/OS. The server contains the CORBA-based location

service agent that places sessions in a cell. All RMI/IOP IORs (for example, enterprise beans) establish connections to the location service daemon first, then forward them to the target application server.

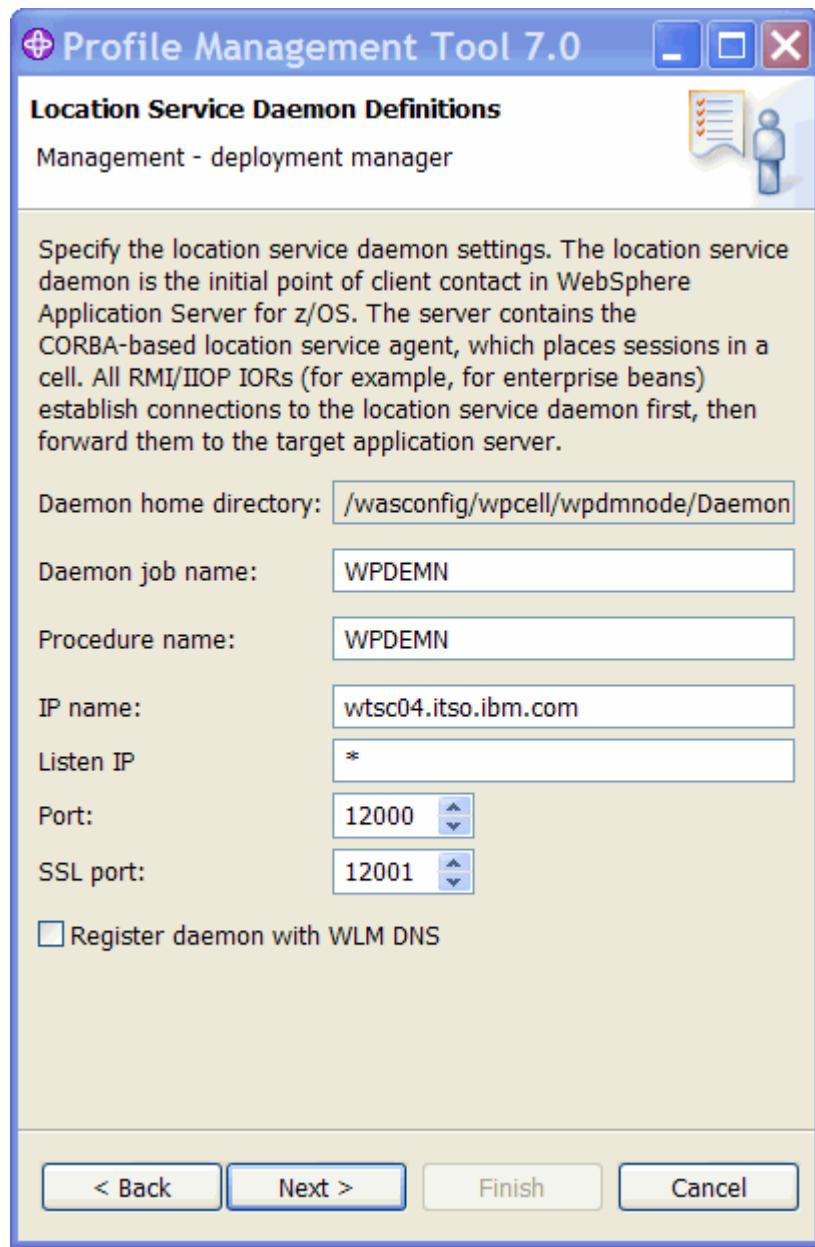


Figure 3-20 Location Service Daemon Definitions

- Daemon home directory:
The directory in which the location service daemon resides. This is set to the configuration file system mount point/Daemon and cannot be changed.
- Daemon job name:
Specifies the job name of the location service daemon, specified in the JOBNAME parameter of the MVS start command used to start the location service daemon. When configuring a new cell, be sure to choose a new daemon job name value. A server automatically starts the location service daemon if it is not already running.
- Procedure name:
Member name in your procedure library to start location service daemon.
- IP name:
The fully qualified IP name, registered with the Domain Name Server (DNS), that the location service daemon uses. The default is your node host name. In a sysplex, you should consider using a virtual IP address (VIPA) for the location service daemon IP name. Select the IP name for the location service daemon carefully. You can choose any name that you want, but, after being chosen, it is difficult to change, even in the middle of customization.
- Listen IP:
The address at which the daemon listens. Select either * or a dotted IP address for this value.
- Port:
Specify the port number on which the location service daemon listens.
- SSL port:
The port number on which the location service daemon listens for SSL connections.

Note: Choose the IP name and port number carefully, because these are difficult to change, even in the middle of customization.

- Register daemon with WLM DNS check box:
If you use the WLM DNS (connection optimization), you must register your location service daemon. Otherwise, do not register your location service daemon. Only one location service daemon per LPAR can register its domain name with WLM DNS; if you have multiple cells in the same LPAR and register more than one location service, it will fail to start.

Click **Next**.

5. The next panel allows you to enter SSL customization values as shown in Figure 3-21.

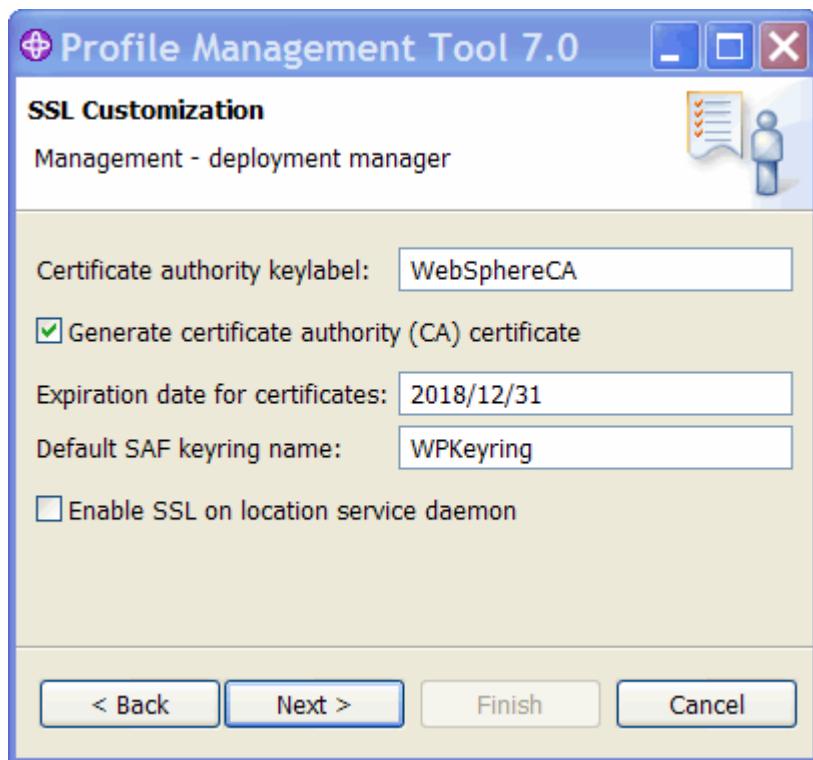


Figure 3-21 SSL Customization

- Certificate authority keylabel:
The name that identifies the certificate authority (CA) to be used in generating server certificates.
- Generate certificate authority (CA) certificate check box:
Selected to generate a new CA certificate. Do not select this option to have an existing CA certificate generate server certificates.
- Expiration date for certificates:
Used for any X509 Certificate Authority certificates created during customization, as well as the expiration date for the personal certificates generated for WebSphere Application Server for z/OS servers. You must specify this even if you have not selected **Generate Certificate Authority (CA) certificate**.

- Default SAF keyring name:

The default name given to the RACF® keyring used by WebSphere Application Server for z/OS. The keyring names created for repertoires are all the same within a cell.

- Enable SSL on location service daemon check box:

Select this option if you want to support secure communications using Inter-ORB Request Protocol (IOP) to the location service daemon using SSL. If selected, a RACF keyring will be generated for the location service daemon to use.

After completing the required SSL information, click **Next**.

6. The next window allows you to select the user registry to be used for administrative security. You can choose from the following options:

- z/OS security product option: This option uses the z/OS system's SAF compliant security product, such as IBM RACF or equivalent, to manage WebSphere Application Server identities and authorization according to the following rules:
 - The SAF security database will be used as the WebSphere user repository.
 - SAF EJBROLE profiles will be used to control role-based authorization, including administrative authority.
 - Digital certificates will be stored in the SAF security database.

Note: Select the z/OS security product option if you are planning to use the SAF security database as your WebSphere Application Server registry or if you plan to set up an LDAP or custom user registry whose identities will be mapped to SAF user IDs for authorization checking. For this security option, you must decide whether to set a security domain name, and choose an administrator user ID and an unauthenticated (guest) user ID.

- WebSphere Application Server security option:

The WebSphere Application Server administrative security option is used to manage the Application Server identities and authorization according to these rules:

- A simple file-based user registry will be built as part of the customization process.
- Application-specific role binds will be used to control role-based authorization.
- The WebSphere Application Server console users and groups list will control administrative authority.
- Digital certificates will be stored in the configuration file system as keystores.

Note: Choose this option if you plan to use an LDAP or custom user registry without mapping to SAF user IDs. (The file-based user registry is not recommended for production use.)

- No security:

Although it is not recommended, you can disable administrative security. If you choose this security option, there are no other choices to make. Your WebSphere Application Server environment will not be secured until you configure and enable security manually. You can enable security manually later via the administrative console or using Jython scripts.

Select an option and click **Next**. The next window that you see will depend on the security option you choose.

7. Figure 3-22 shows the parameters to enter if you selected the z/OS security product option.



Figure 3-22 Setting z/OS security

- SAF profile prefix: (Formerly known as Security domain identifier):

This optional parameter is used to distinguish between APPL or EJBROLE profiles based on security domain name. It provides an alphanumeric security domain name of one to eight characters. Internally, this sets SecurityDomainType to the string cellQualified.

All servers in the cell will prepend the security domain name you specify to the application-specific J2EE role name to create the SAF EJBROLE profile for checking. The security domain name is not used, however, if role checking is performed using WebSphere Application Server for z/OS bindings.

The security domain name is also used as the APPL profile name and inserted into the profile name used for CBIND checks. The RACF jobs that the Customization Dialog generates create and authorize the appropriate RACF profiles for the created nodes and servers.

If you do not want to use a security domain identifier, leave this field blank.

- WebSphere Application Server unauthenticated user ID:

Associated with unauthenticated client requests. It is sometimes referred to as the “guest” user ID. It should be given the RESTRICTED attribute in RACF, to prevent it from inheriting UACC-based access privileges. The UNIX System Services UID number for the user ID is specified here and is associated with unauthenticated client requests. The UID value must be unique numeric values between 1 and 2,147,483,647.

- Enable Writable SAF Keyring support

In previous versions of WebSphere Application Server, keystores and truststores that pointed to an SAF keyring could only be used in read only mode. Certificates stored in SAF could not be created, deleted, imported, or exported. These operations needed to be performed by the SAF administrator. In WebSphere V6.1, certificates stored in SAF could be viewed and their expirations could be monitored in the administrative console.

With WebSphere Application Server for z/OS V7, the writable keyring support has been introduced. This feature enhances the capabilities of the administrative console to consistently manage keyrings and certificates stored in SAF. With the writable keyrings support, certificates can be created and signed by a CA, connected to keyrings, removed from keyrings, imported, exported and renewed using the administrative console. This major enhancement increases the efficiency of SSL management within WebSphere Application Server for z/OS while SAF still has control over all certificates and keyrings.

All certificates created with the writable keyring support are generated and signed by Java code and not by SAF. In this case the writable keyring support only uses SAF to store the generated certificates.

The writable keyring support is completely optional. New keystores and truststores marked as read-only can be created independent from the writable keyring support. When using the read-only JCERACFKS and JCECCARACFS keystores, the certificates in the appropriate SAF keyring can still be viewed in the administrative console.

Click **Next**.

8. The next window (Figure 3-23) allows you to tailor the JCL for the customization jobs. Enter a valid job statement for your installation on this window. The profile creation process will update the job name for you in all the generated jobs, so you need not be concerned with that portion of the job statement. If continuation lines are needed, replace the comment lines with continuation lines.

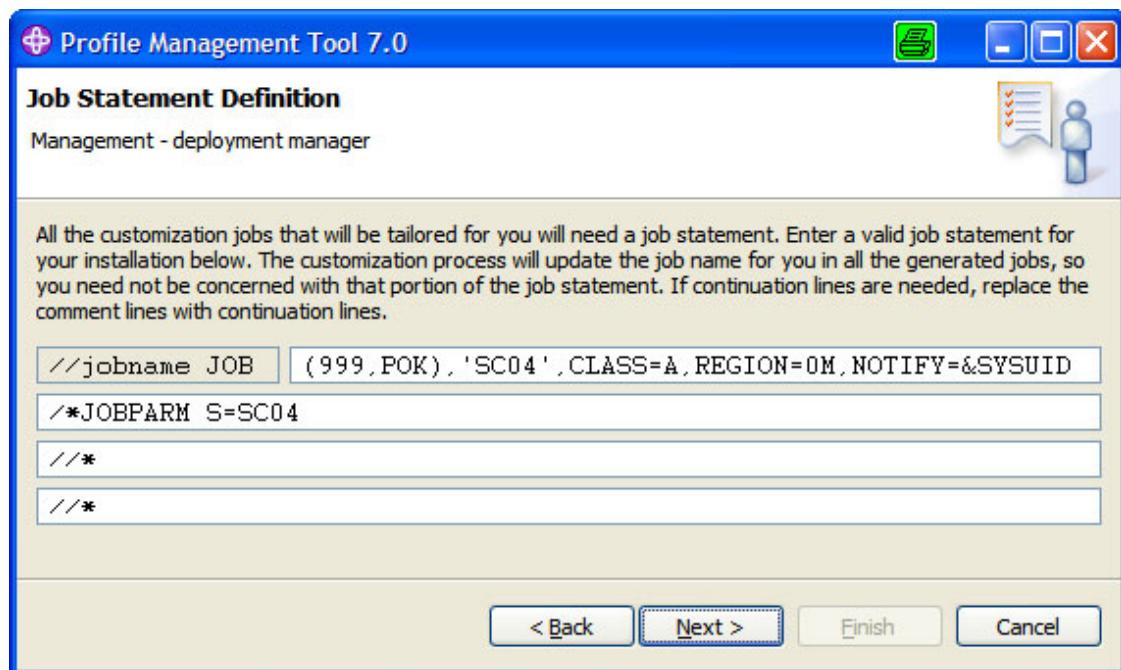


Figure 3-23 Deployment manager: job statement definition

Click **Next**.

9. The last window shows a short summary of the customization, including profile type and where the generated jobs will be stored. To change the characteristics of this profile, click the **Back** button; otherwise, click **Create** to generate your z/OS Customization jobs.
10. The PMT will display a summary window (Figure 3-24) that indicates whether the jobs were created successfully or not. If the jobs were not created, a log file containing failure information will be identified.

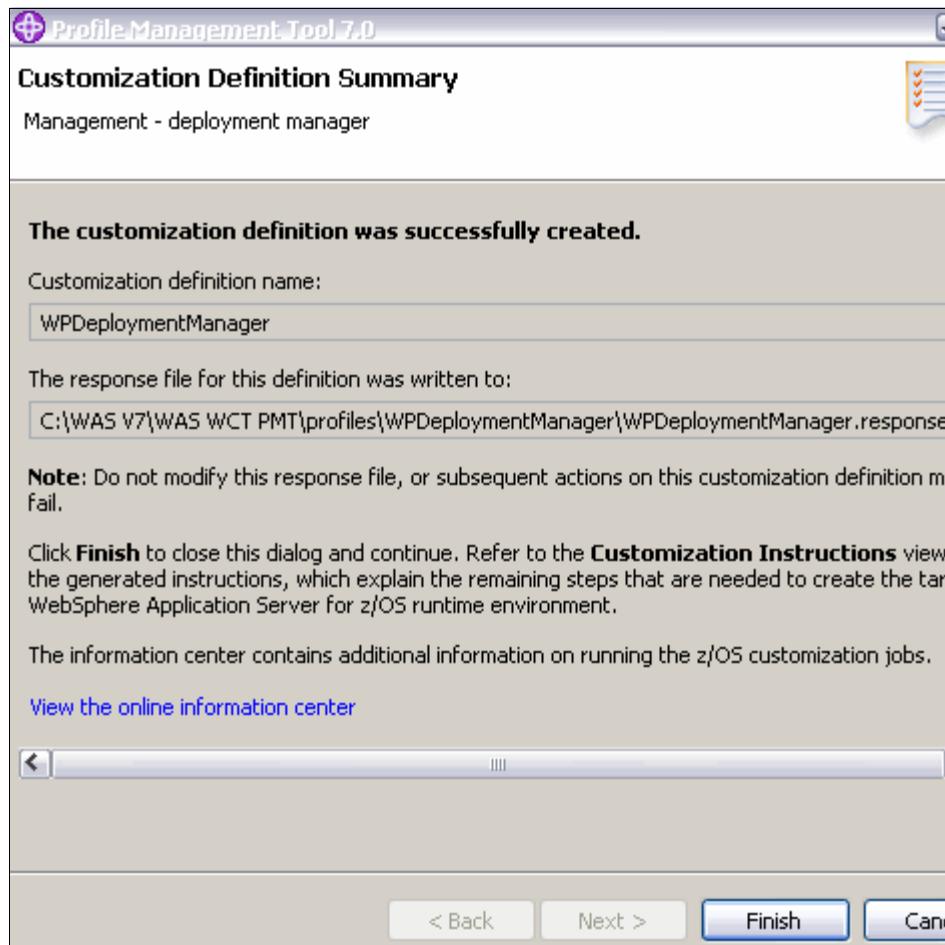


Figure 3-24 Deployment manager: Customization definition summary

Click **Finish** to return to the Profile Management Tool main panel. The new deployment manager definition will be listed in the Customization Definitions tab.

3.4.2 Uploading the jobs to the z/OS system

If successful, the next step in the z/OS customization process is to upload these jobs and the associated instructions to a pair of z/OS partitioned data sets. To do this,

1. On the main window, select the customization definition for the profile and click the **Process** button. To upload the generated jobs to the target z/OS system select **Upload to target z/OS system** and click **Next**.
2. In the upload customization definition window (Figure 3-25 on page 151), enter the target z/OS system. This must be fully qualified or the upload will fail.

Use the **Allocate target z/OS data sets** check box to specify whether to allocate the data sets if they do not exist (box check). If the data sets exist and are to be reused, clear the box.

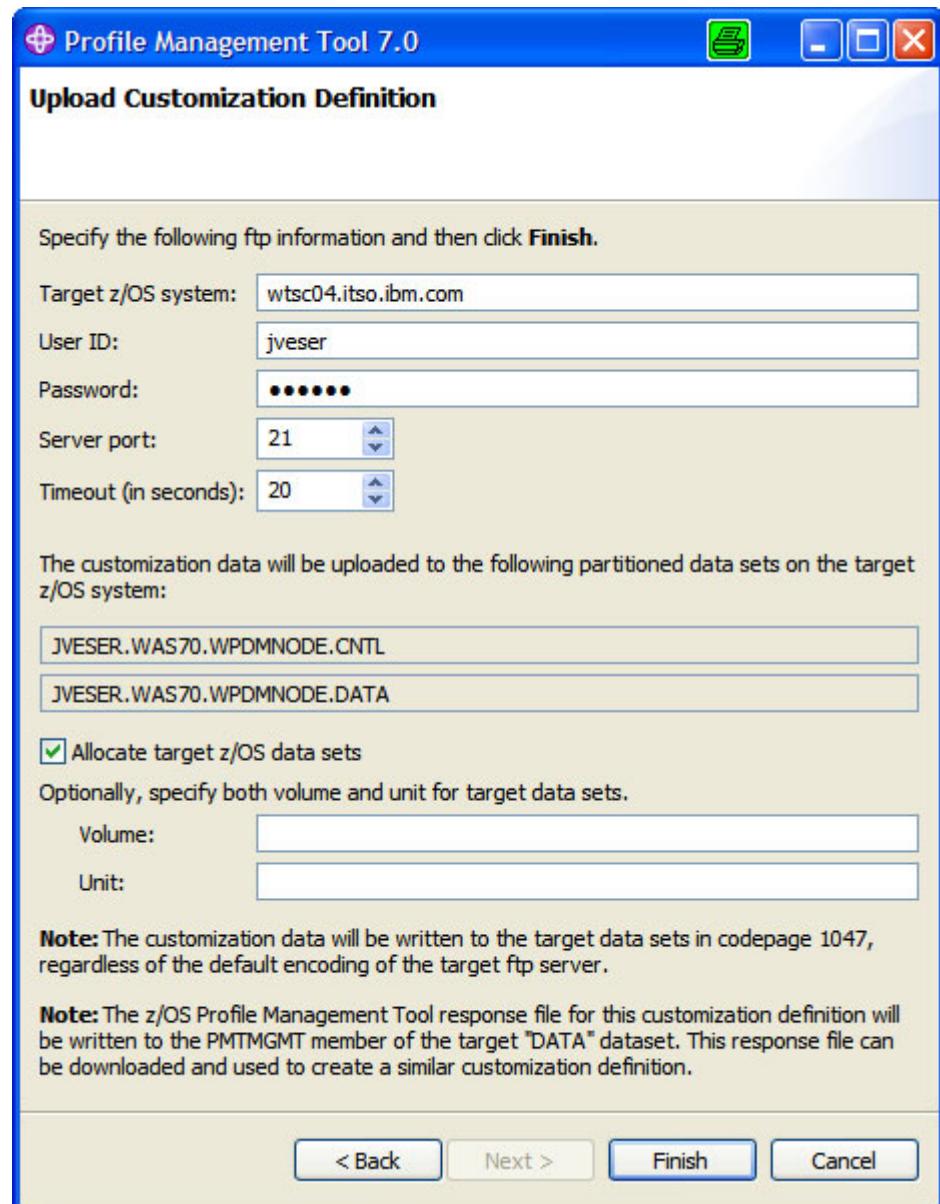


Figure 3-25 Upload the customization definition

Click **Finish**.

You can see a progress information window while the upload is occurring.

3.4.3 Executing the jobs

After the customization profile is uploaded, the next step is to execute the jobs. The instructions for preparing for and executing the jobs can be found in the Profile Management Tool. Select the customization definition and switch to the **Customization Instructions** tab (Figure 3-26). These instructions are also contained in a job that has been loaded to the host.

The screenshot shows the WebSphere Customization Tools interface. The title bar reads "WebSphere Customization Tools". The menu bar includes "File", "Window", and "Help". The toolbar has icons for "Profile Management Tool" and "Welcome". A tab bar at the top has "Profile Management Tool" selected, followed by "Welcome", "Customization Locations", "Customization Definitions", "Customization Summary", "Customization Instructions" (which is the active tab), and "Customization Response File". Below the tabs, a URL is displayed: "file:///C:/WAS%20V7/WAS%20WCT%20PMT/profiles/WPDeploymentManager/managementInstruction". The main area contains a table titled "Customization Locations" with one row:

Name	Version	Location
My WCT PMT	7.0	C:\WAS V7\WAS WCT PMT

Below the table, the "Customization Instructions" tab is active, displaying the following text:

WebSphere Application Server for z/OS customization instructions:

Deployment manager WPDMGR (cell WPCELL, node WPDMNODE)
Tailored on 2009/05/05 at 13:14 by sadtler
WCT version 7.0.0.3 build cf030908.27

The Profile Management Tool has created a response file based on the information you provided. These instructions tell you how to convert this response file into a set of customization jobs and how to run the jobs to customize WebSphere Application Server for z/OS. When you upload the customization jobs to the target system, a text version of these instructions will be written to:

JVESER.WAS70.WPDMNODE.CNTL (BBOCCINS)

Figure 3-26 Customization definition instructions

The instructions help you determine what jobs to run, the order to run them in, and the expected results. They also tell you how to start the environment after you are done.

After the jobs have been run successfully, the deployment manager profile is complete.

Figure 3-27 shows an example of the jobs generated to create a deployment manager and the sequence they should run in.

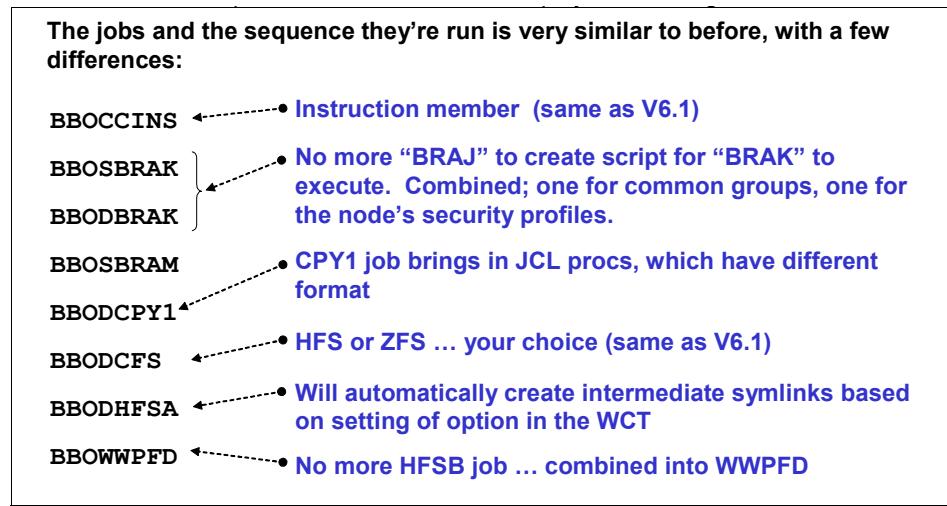


Figure 3-27 Deployment manager configuration jobs created by PMT

Figure 3-28 illustrates how the jobs are mapped to the components of the configuration.

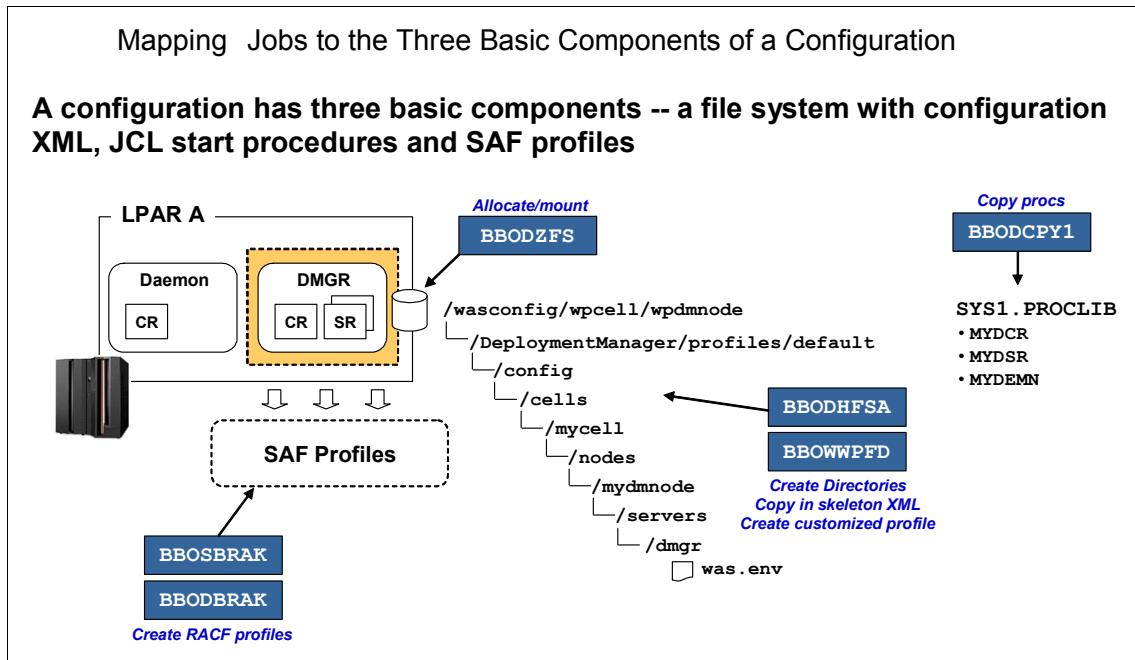


Figure 3-28 Mapping generated jobs to configuration functions

3.5 Creating the base application server definition

The next step in building the sample environment is to create a standalone application server.

To begin, run the profile management tool to create the customization definition:

1. Click **Create** on the Profile Management Tool main panel.

The first panel that you see when you click **Next** on the Profile Management Tool Welcome panel prompts you to select the type of WebSphere environment that you want to create.

2. In the environment selection panel, select **Application server** as shown in Figure 3-29 and click **Next**.



Figure 3-29 Selecting Application server environment

3. Specify a name for the customization definition and optionally a response file path, and click **Next**.

4. The next panel (Figure 3-30) lets you establish configuration default values.

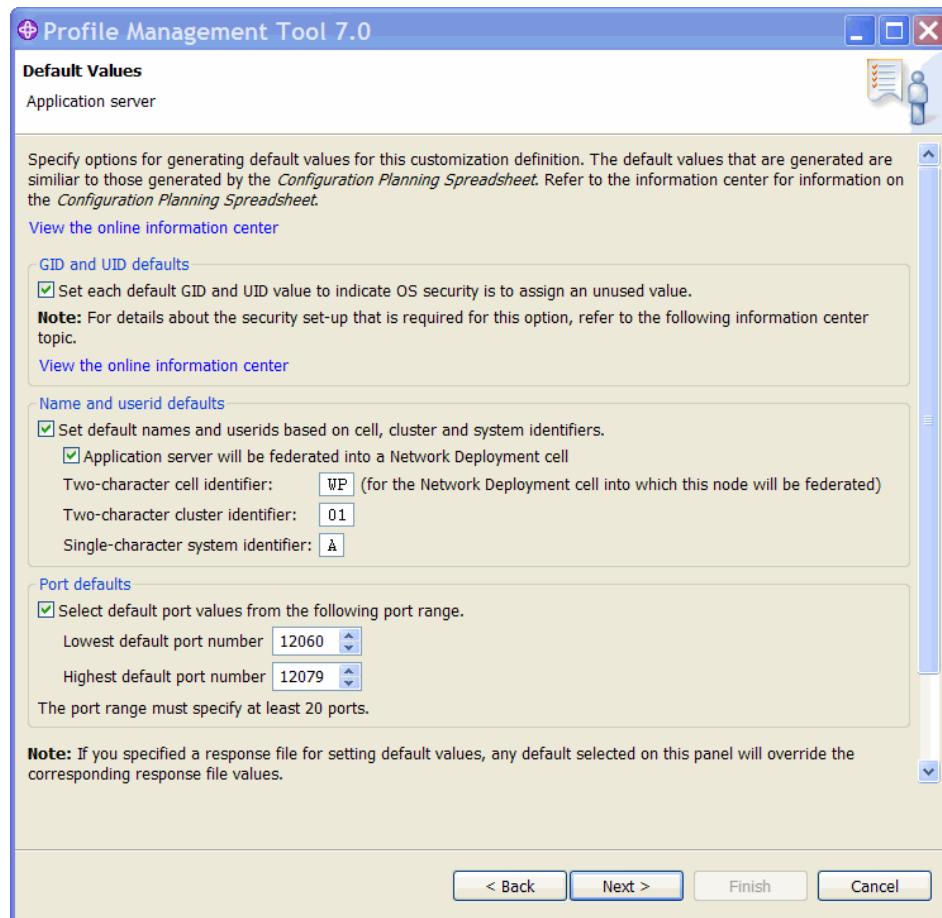


Figure 3-30 Application server default values

Note: If you specified a response file for setting default values, any default selected here will override the corresponding response file values.

The GID and UID defaults section contains the following selectable option:

- **Set each default GID and UID value to indicate that operating-system security is to assign an unused value:**

When this option is selected, each GID and UID value is defaulted to allow operating-system security to assign an unused value. When this option is not selected, each GID and UID value is defaulted to an IBM-provided number.

The Name and userid defaults contains the following selectable options:

- **Set default names and user IDs based on cell, cluster, and system identifiers:**

When this option is selected, default cell, node, server, cluster, and procedure names as well as group names and user IDs are based on cell, cluster, and system identifiers.

- **Application server will be federated into a Network Deployment cell:**

Select this option to indicate that the application server will be federated into a Network Deployment cell at some point in time.

- Two-character cell identifier:

Enter a two-character cell identifier to be used to create default names and user IDs.

If you have selected the option to federate to a cell, specify the two-character cell identifier of the target Network Deployment cell.

Rule: The first character must be an alphabetic character and the second character must be an alphanumeric character. Alphabetic characters can be entered in lowercase or uppercase. The case of alphabetic characters will be adjusted as appropriate for each generated default value.

- Two-character cluster identifier:

Two-character cluster identifier to be used to create default names and user IDs. The characters will be appended to the cluster transition name.

Rule: The characters must be alphabetic characters. The alphabetic characters can be entered in lowercase or uppercase. The case of alphabetic characters will be adjusted as appropriate for each generated default value.

- Single-character system identifier:

Single-character system identifier to be used to create default names and user IDs. It will be appended to the short and long names for the cell, node, and server and to the appropriate process names.

Rule: The character must be an alphanumeric character. An alphabetic character can be entered in lowercase or uppercase. The case of the alphabetic character will be adjusted as appropriate for each generated default value.

- Port defaults:
 - Select default port values from the following port range.
When this option is not selected, each port value defaults to an IBM-provided number. When this option is selected, each port default value is selected from the following port number range.
The port range must contain at least 20 ports.
- Lowest default port number:
Lowest number that can be assigned as a default port number
- Highest default port number:
Highest number that can be assigned as a default port number

Click **Next**.

5. You are now prompted to specify the high-level qualifier for the target z/OS datasets that will contain the generated jobs and instructions that are being created. (See Figure 3-11 on page 128).

The generated batch jobs and instructions will be uploaded to two z/OS partitioned datasets:

- *HLQ.CNTL*
Partitioned dataset with fixed block 80-byte records to contain customization jobs
- *HLQ.DATA*
Partitioned dataset with variable-length data to contain other data contained in the customization definition

Note: A multilevel high-level qualifier can be specified as the dataset high-level qualifier

Click **Next**.

6. The Configure Common Groups panel appears as shown in Figure 3-31.

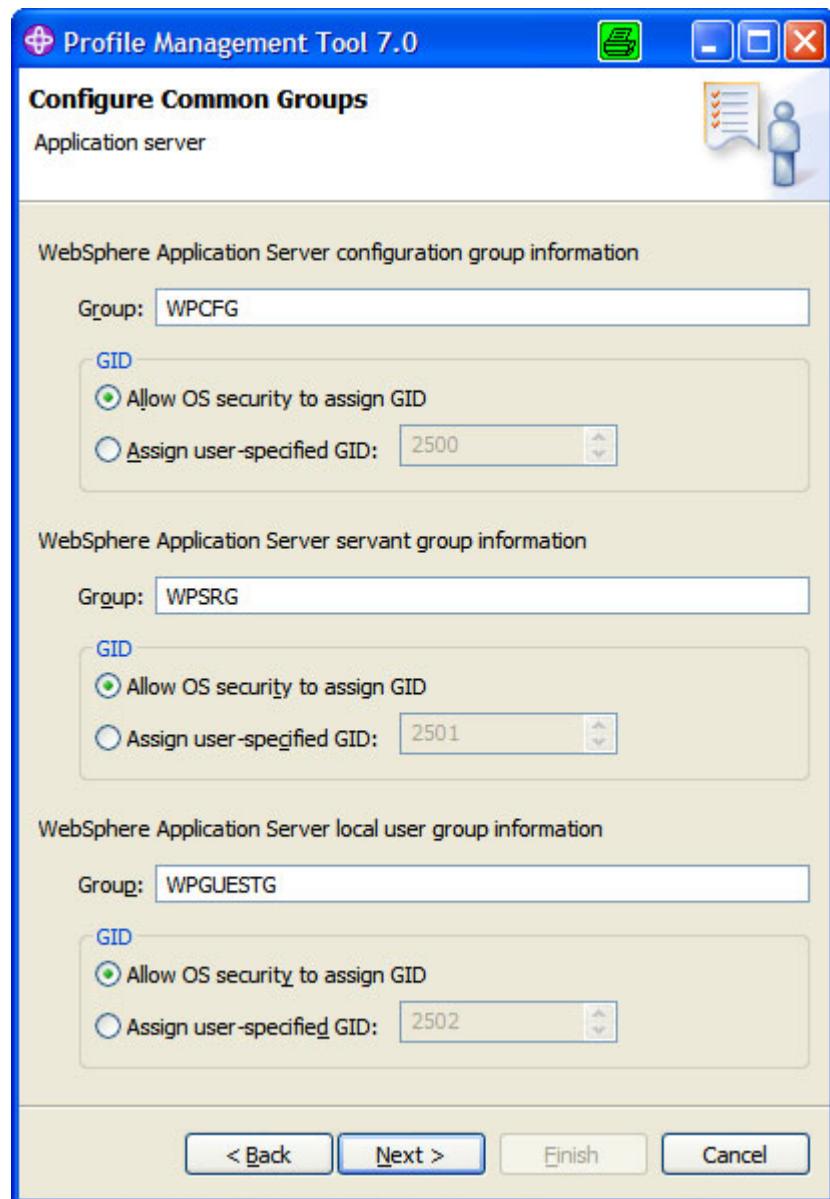


Figure 3-31 Configure Common Groups

- WebSphere Application Server configuration group information:
Specify the default group name for the WebSphere Application Server administrator user ID and all server user IDs.
Select whether to allow the OS security system (RACF) to assign an unused GID value, or assign a specific GID.
- WebSphere Application Server servant group information:
Specify the group name for all servant user IDs. You can use this group to assign subsystem permissions, such as DB2 authorizations, to all servants in the security domain.
Select whether to allow the OS security system (RACF) to assign an unused GID value, or assign a specific GID.
- WebSphere Application Server local user group information:
Specify the group name for local clients and unauthorized user IDs (provides minimal access to the cell).
Select whether to allow the OS security system (RACF) to assign an unused GID value, or assign a specific GID.

GID values: The specified GID is the UNIX System Services GID number for the WebSphere Application Server configuration group. GID values must be unique numeric values between 1 and 2,147,483,647.

Click **Next**.

7. The Configure Common Users panel appears as shown in Figure 3-32.

Common controller user ID

User ID: WPACR

UID

Allow OS security to assign UID

Assign user-specified UID: 2431

Common servant user ID

User ID: WPASR

UID

Allow OS security to assign UID

Assign user-specified UID: 2432

WebSphere Application Server administrator

User ID: WPADMIN

UID

Allow OS security to assign UID

Assign user-specified UID: 2403

Asynchronous administration user ID

User ID: WPADMSH

UID

Allow OS security to assign UID

Assign user-specified UID: 2504

WebSphere Application Server user ID home directory:

/var/WebSphere/home

< Back Next > Finish Cancel

Figure 3-32 Configure Common Users

- Common controller user ID:

UIDs: UIDs must be unique numbers between 1 and 2,147,483,647 within the system.

Enter the user ID to be associated with all the control regions and the daemon. This user ID will also own all of the configuration file systems. If you are using a non-IBM security system, the user ID might have to match the procedure name. Refer to your security system's documentation.

Select whether to allow the OS security system (RACF) to assign an unused UID value, or assign a specific UID to be associated with the control region user ID.

- Common servant user ID:

Enter the user ID to be associated with the servant and control adjunct regions.

Select whether to allow the OS security system (RACF) to assign an unused UID value, or assign a specific UID to be associated with the servant region user ID.

- WebSphere Application Server administrator:

Enter the user ID for the initial WebSphere Application Server administrator. The user ID must have the WebSphere Application Server configuration group as its default UNIX System Services group.

Select whether to allow the OS security system (RACF) to assign an unused UID value, or assign a specific UID to be associated with the administrator user ID

- Asynchronous administration user ID:

Enter the user ID to be used to run asynchronous administration operations procedure. This ID must be a member of the WebSphere Application Server configuration group.

Select whether to allow the OS security system (RACF) to assign an unused UID value, or assign a specific UID for the asynchronous administration task user ID.

- WebSphere Application Server user ID home directory:

This field identifies a new or existing file system directory in which home directories for WebSphere Application Server for z/OS user IDs will be created by the customization process. This directory does not need to be shared among z/OS systems in a WebSphere Application Server cell.

Click **Next**.

8. Provide the system and data set names to be used (See Figure 3-14 on page 133):
 - Specify the system and sysplex name for the target z/OS system on which you will configure WebSphere Application Server for z/OS.
 - Enter the name of an existing procedure library where the WebSphere Application Server for z/OS cataloged procedures are added.

Click **Next**.

9. Next the Cell, Node, and Server names panel is presented as shown in Figure 3-33 on page 164.
 - Specify the long and short names for the cell, node, and servers. Short names identify the process to z/OS facilities, such as SAF. Long names are used as the primary external identification for the process. This is the name you will see in the administrative console.
 - Cluster transition name:
If this server is converted into a clustered server, this name becomes the cluster short name. The cluster short name is the WLM APPLENV name for all servers that are part of the same cluster.
 - Specify the JVM mode. With Version 7.0 either 31 bit or 64 bit can be selected. The default value is 64 bit, because the 31 bit has been deprecated.

Click **Next**.

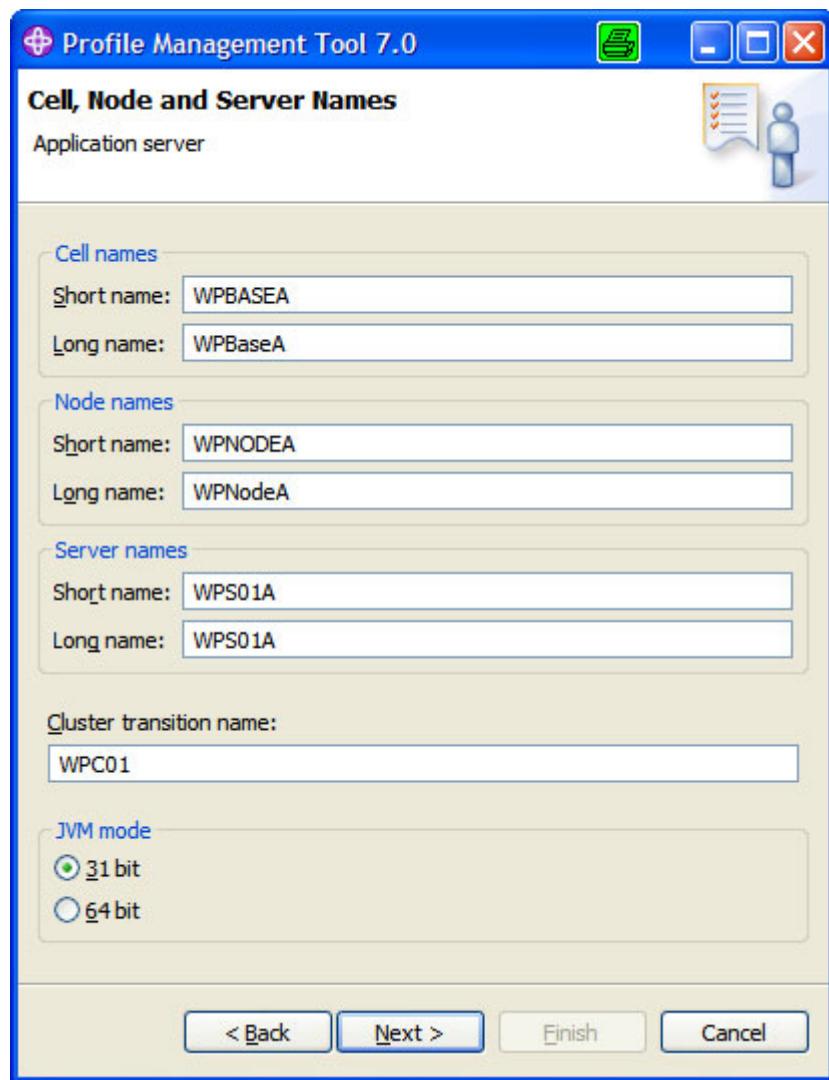


Figure 3-33 Cell, Node, and Server Names

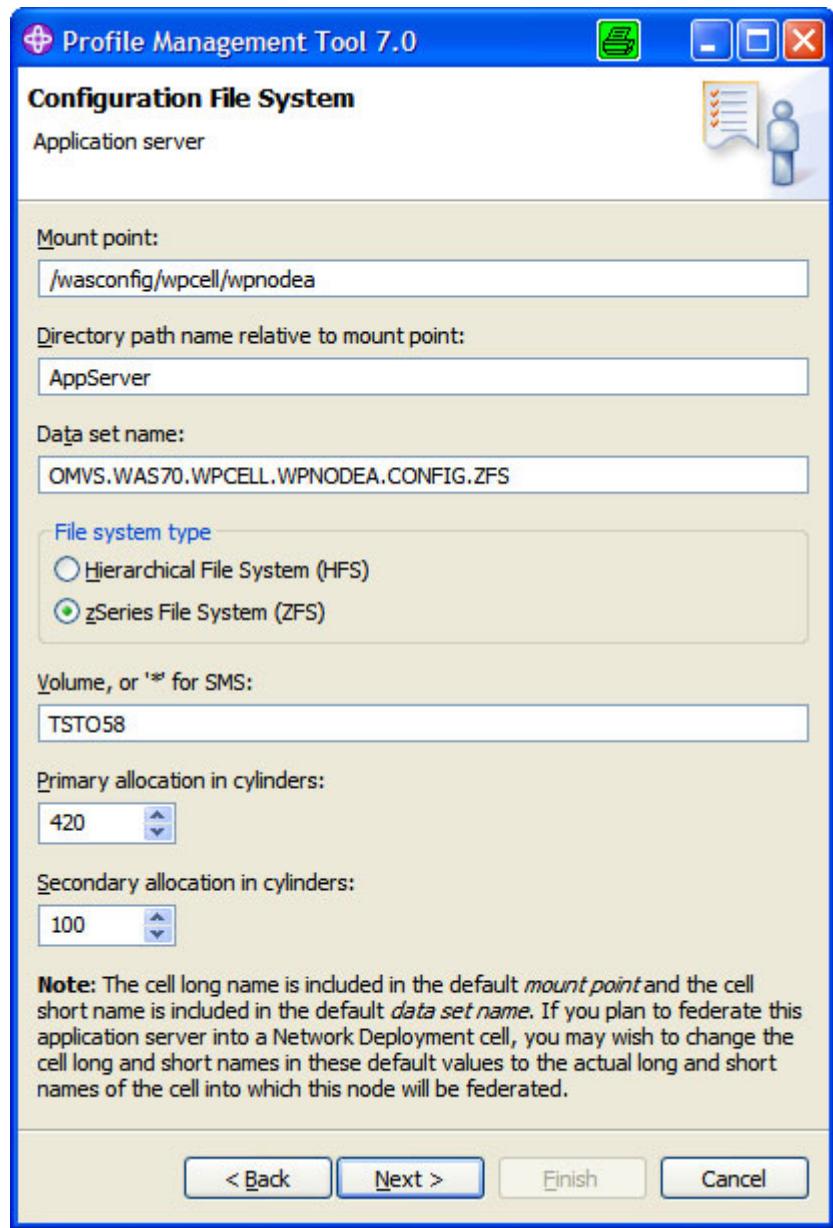


Figure 3-34 Configuration file system settings

10.The Configuration File System values are entered as shown in Figure 3-34 on page 165.

- Mount point:

Application server configuration file system mount point: Specifies the Read/write file system directory where the application data and environment files are written. This field is not writable here, but was specified earlier on the “System Environment: Configuration file system information” window.

- Directory path name relative to mount point:

The relative path name of the directory within the configuration file system in which the application server configuration resides.

- Data set name:

The file system data set you will create and mount at the specified mount point above.

- Volume, or ^{1*} for SMS:

Specify either the DASD volume serial number to contain the above data set or "*" to let SMS select a volume. Using "*" requires that SMS automatic class selection (ACS) routines be in place to select the volume. If you do not have SMS set up to handle data set allocation automatically, list the volume explicitly.

- Primary allocation in cylinders:

The initial size allocation for the configuration file system data set. In the application server, the total space needed for this data set increases with the size and number of the installed applications. The minimum suggested size is 250 cylinders (3390).

- Secondary allocation in cylinders:

The size of each secondary extent. The minimum suggested size is 100 cylinders.

- File System type:

Select to allocate and mount your configuration file system data set using HFS or zFS.

11.Specify the information for the product file system as shown in Figure 3-35.

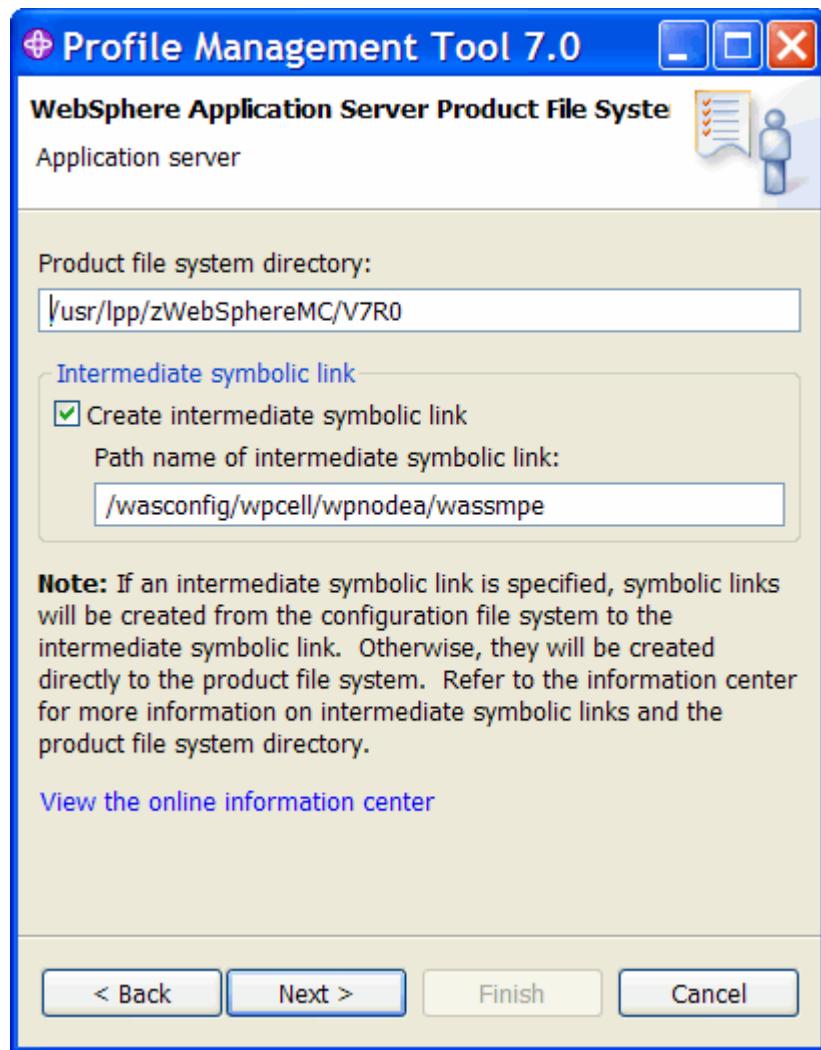


Figure 3-35 Product file system

- Specify the name of the directory where the product files for WebSphere Application Server for z/OS were stored during installation.
- Select the option to allow to set up an intermediate symbolic link and specify the path name.

Click **Next**.

12. The next panel allows you to select the applications to deploy onto the environment that you are creating. The choices are:

- Administrative console (recommended)
- Default application
- Sample applications

Click **Next**.

13. Enter the process information into the panel shown in Figure 3-36. The job names for the processes are provided in the panel and cannot be changed. Specify the procedure name for each process.

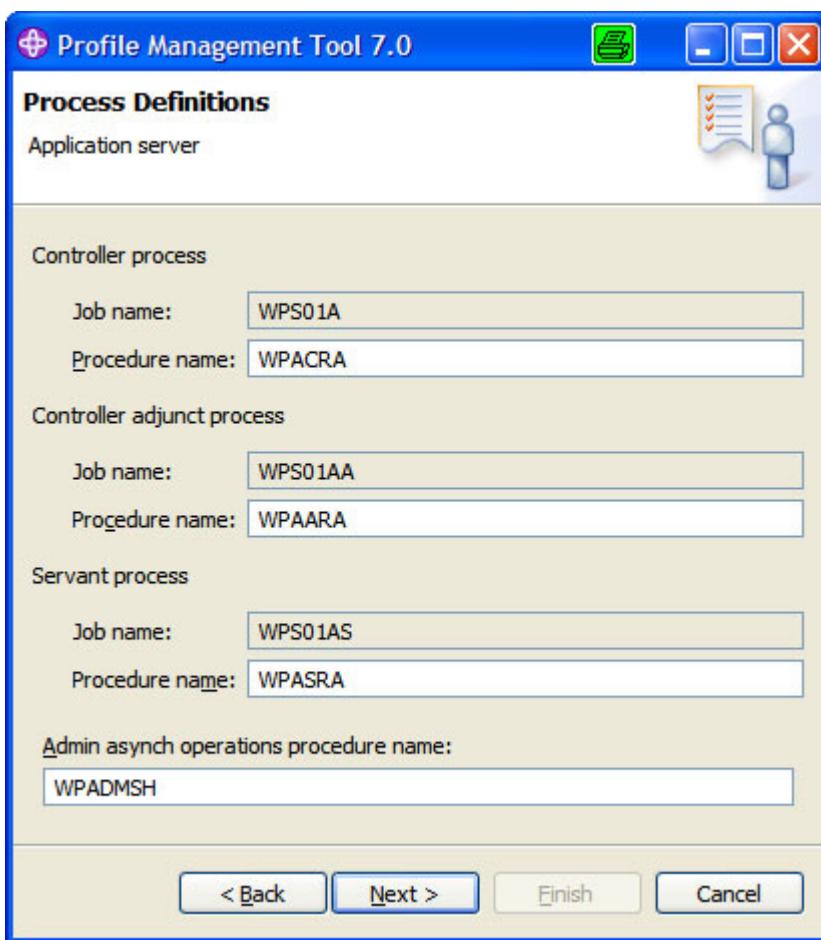


Figure 3-36 Application server names

- Controller process job and procedure name:
The job name for the control region is the same as the server short name. This is the name used in the MVS START command to start the region.
- Controller adjunct process job and procedure name:
The job name is used by WLM to start the control region adjunct. This is set to the server short name followed by the letter “A.”
- Servant process job and procedure name:
The job name is used by WLM to start the servant regions. This is set to the server short name followed by the letter “S.”
- Admin asynch operations procedure name:
Specify the JCL procedure name of a started task to be launched by way of the START command by node agents or application servers to perform certain asynchronous administrative operations (such as node synchronization) and add and remove a node.

Click **Next**.

14. Specify the application server port values into the panel shown in Figure 3-37.

Good planning is very important to avoid port conflicts, so be sure that you have all values you need in order to fill out this panel.

<u>Node host name or IP address:</u>	wtsc04.itso.ibm.com
<u>JMX SOAP connector port:</u>	12062
<u>ORB listener IP address:</u>	*
<u>ORB port:</u>	12063
<u>ORB SSL port:</u>	12064
<u>HTTP transport IP address:</u>	*
<u>Administrative console port:</u>	12065
<u>Administrative console secure port:</u>	12066
<u>HTTP transport port:</u>	12067
<u>HTTPS transport port:</u>	12068
<u>Administrative interprocess communication port (K):</u>	12069
<u>High Availability Manager Communication Port (DCS):</u>	12070
<u>Service Integration port:</u>	12071
<u>Service Integration Secure port:</u>	12072
<u>Service Integration MQ Interoperability port:</u>	12073
<u>Service Integration MQ Interoperability Secure port:</u>	12074
<u>Session Initiation Protocol (SIP) Port:</u>	12075
<u>Session Initiation Protocol (SIP) Secure Port:</u>	12076

[**< Back**](#) [**Next >**](#) [**Finish**](#) [**Cancel**](#)

Figure 3-37 Application server port values

Click **Next**.

15.Location Service Daemon Definitions are shown in Figure 3-38.

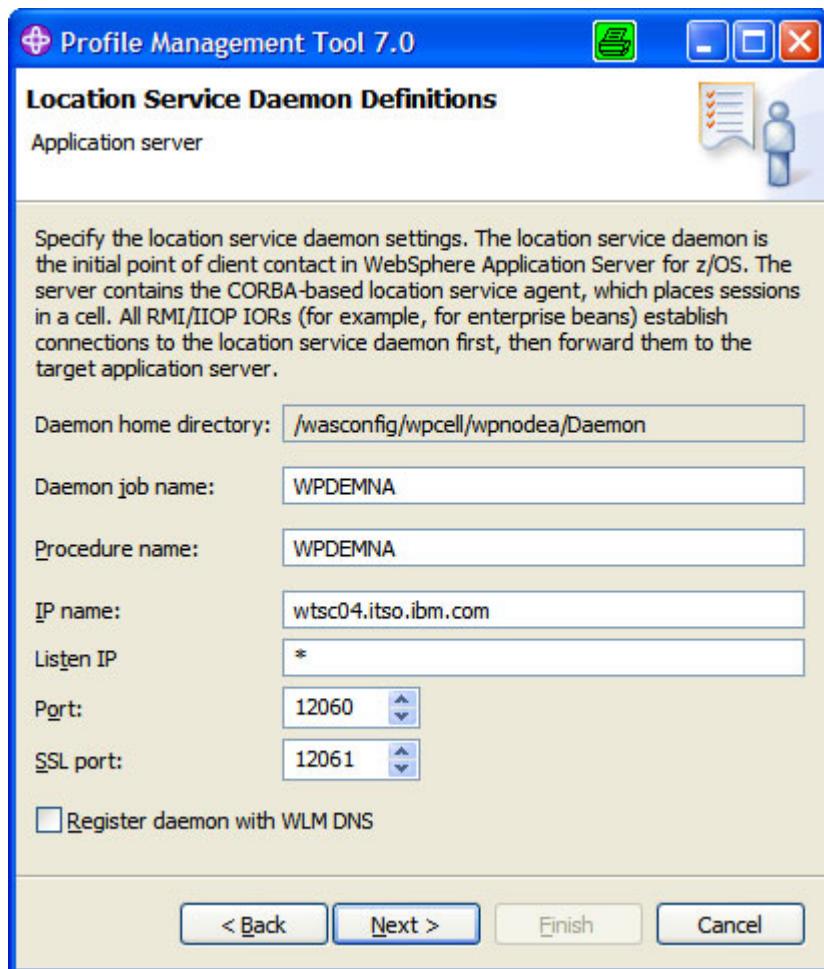


Figure 3-38 Location Service daemon definitions

- Daemon home directory: Directory in which the location service daemon resides. This is set to the configuration file system mount point/Daemon and cannot be changed.
- Daemon job name: Specifies the job name of the location service daemon, specified in the JOBNAME parameter of the MVS start command used to start the location service daemon.
- Procedure name: Name of the member in your procedure library to start the location service daemon.

- IP Name: The fully qualified IP name, registered with the Domain Name Server (DNS), that the location service daemon uses.
- Listen IP: Address at which the daemon listens.
- Port: Port number on which the location service daemon listens.
- SSL port: Port number on which the location service daemon listens for SSL connections.
- Register daemon with WLM DNS: If you use the WLM DNS (connection optimization), you must select this option to register your location service daemon with it; otherwise, do not select it.

Click **Next**.

16. Next the SSL Customization panel is presented. This is similar to the deployment manager panel shown previously in Figure 3-21 on page 143.

Enter the information required for SSL connections and click **Next**.

17. The next panel allows you to select the user registry that will be used to manage user identities and authorization policy. You can select from one of the following choices:

Note: If you plan to federate this application server, we recommend that you set the application server's SAF profile prefix to be the same as that of the Network Deployment Cell.

- z/OS security product option: This option uses the z/OS system's SAF compliant security product, such as IBM RACF or equivalent, to manage WebSphere Application Server identities and authorization according to the following rules:
 - The SAF security database will be used as the WebSphere user repository.
 - SAF EJBROLE profiles will be used to control role-based authorization, including administrative authority.
 - Digital certificates will be stored in the SAF security database.

Note: Select the z/OS security product option if you are planning to use the SAF security database as your WebSphere Application Server registry or if you plan to set up an LDAP or custom user registry whose identities will be mapped to SAF user IDs for authorization checking. For this security option, you must decide whether to set a security domain name, and choose an administrator user ID and an unauthenticated (guest) user ID.

- WebSphere Application Server security option: The WebSphere Application Server administrative security option is used to manage the Application Server identities and authorization as follows:
 - A simple file-based user registry will be built as part of the customization process.
 - Application-specific role binds will be used to control role-based authorization.
 - The WebSphere Application Server console users and groups list will control administrative authority.
 - Digital certificates will be stored in the configuration file system as keystores.

Note: Choose this option if you plan to use an LDAP or custom user registry without mapping to SAF user IDs. (The file-based user registry is not recommended for production use.)

- No security:

Although it is not recommended, you can disable administrative security. If you choose this security option, there are no other choices to make.

Your WebSphere Application Server environment will not be secured until you configure and enable security manually. You can enable security manually later via the administrative console or using Jython scripts.

Select an option and click **Next**. The next window you see will depend on the security option you choose.

18. The next panel displayed will depend on the Security option you selected.

We previously selected the **Use a z/OS security product** option and were presented with the panel shown in Figure 3-39.



Figure 3-39 z/OS security settings

- SAF profile prefix: (Formerly known as Security domain identifier):

This optional parameter is used to distinguish between APPL or EJBROLE profiles based on security domain name. It provides an alphanumeric security domain name of one to eight characters. Internally, this sets SecurityDomainType to the string cellQualified.

All servers in the cell will prepend the security domain name you specify to the application-specific J2EE role name to create the SAF EJBROLE profile for checking. The security domain name is not used, however, if role checking is performed using WebSphere Application Server for z/OS bindings.

The security domain name is also used as the APPL profile name and inserted into the profile name used for CBIND checks. The RACF jobs that the Customization Dialog generates create and authorize the appropriate RACF profiles for the created nodes and servers.

If you do not want to use a security domain identifier, leave this field blank.

- WebSphere Application Server unauthenticated user ID:

Associated with unauthenticated client requests. It is sometimes referred to as the “guest” user ID. It should be given the RESTRICTED attribute in RACF, to prevent it from inheriting UACC-based access privileges. The UNIX System Services UID number for the user ID is specified here and is associated with unauthenticated client requests. The UID value must be unique numeric values between 1 and 2,147,483,647.

- Enable Writable SAF Keyring support

In previous versions of WebSphere Application Server keystores and truststores that pointed to a SAF keyring could only be used in read only mode. Certificates stored in SAF could not be created, deleted, imported or exported. These operations needed to be performed by the SAF administrator. In WebSphere V6.1, certificates stored in SAF could be viewed and their expirations could be monitored in the administrative console.

With WebSphere Application Server for z/OS V7 the writable keyring support has been introduced. This feature enhances the capabilities of the administrative console to consistently manage keyrings and certificates stored in SAF.

With the writable keyrings support, certificates can be created and signed by a CA, connected to keyrings, removed from keyrings, imported, exported and renewed using the administrative console. This major enhancement increases the efficiency of SSL management within WebSphere Application Server for z/OS while SAF still has control over all certificates and keyrings.

All certificates created with the writable keyring support are generated and signed by Java code and not by SAF. In this case the writable keyring support only uses SAF to store the generated certificates.

The writable keyring support is completely optional. New keystores and truststores marked as read-only can be created independent from the writable keyring support. When using the read-only JCERACFKS and JCECCARACFS keystores, the certificates in the appropriate SAF keyring can still be viewed in the administrative console.

Click **Next**.

19. The next panel allows you to customize the JCL for the generated jobs. The panel is the same as seen in the deployment manager scenario (Figure 3-23 on page 148.)

Make any adjustments necessary and click **Next**.

20. The Customization Summary is the final panel presented. Click **Create** to store this application server environment definition for later transfer to the intended z/OS host system.

21. Click **Finish** when the jobs are complete.

If successful, the next step in the z/OS customization process is to upload these jobs and the associated instructions to a pair of z/OS partitioned data sets. To do this,

1. On the main window, select the customization definition for the profile and click the **Process** button. To upload the generated jobs to the target z/OS system select **Upload to target z/OS system** and click **Next**.
2. In the upload customization definition window (Figure 3-25 on page 151), enter the target z/OS system. This must be fully qualified or the upload will fail.

Use the **Allocate target z/OS data sets** check box to specify whether to allocate the data sets if they do not exist (box check). If the data sets exist and are to be reused, clear the box.

Click **Finish**.

You will see a progress information window while the upload is occurring.

3. After the customization profile is uploaded, select the customization definition in the Profile Management Tool and switch to the **Customization Instructions** tab. This will provide you with complete instructions on how to build the profile using the jobs.

These instructions will help you determine what jobs to run, the order to run them in, and the expected results. It will also tell you how to start the environment after you are done.

After the jobs have been run successfully, the application server profile is complete.

3.6 Federating an application server

This definition is used to federate the base application server into the previously created deployment manager node.

1. Select **Federate an application server** in the Environment Selection panel as shown in Figure 3-40.

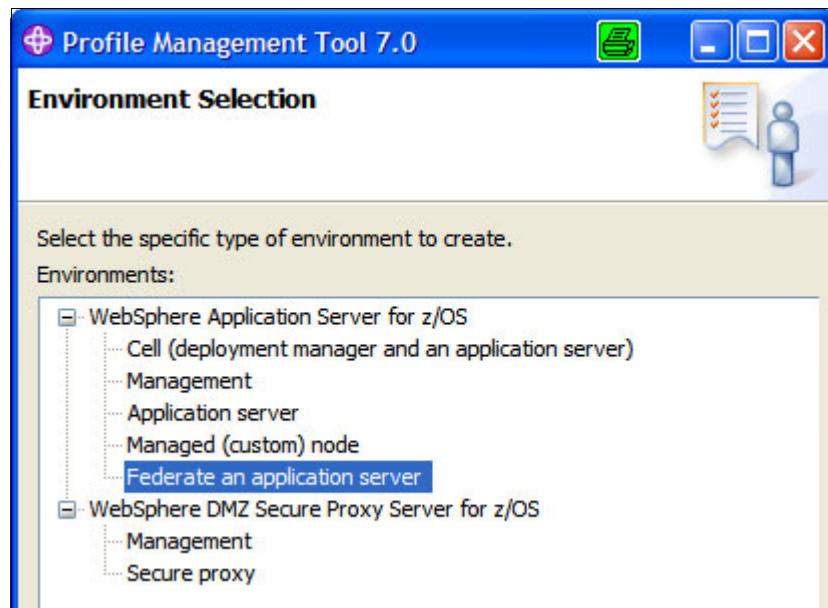


Figure 3-40 Federate an application server selection

Click **Next**.

2. Enter a name for the customization definition and click **Next**.
3. The Default Values panel next appears as shown in Figure 3-41

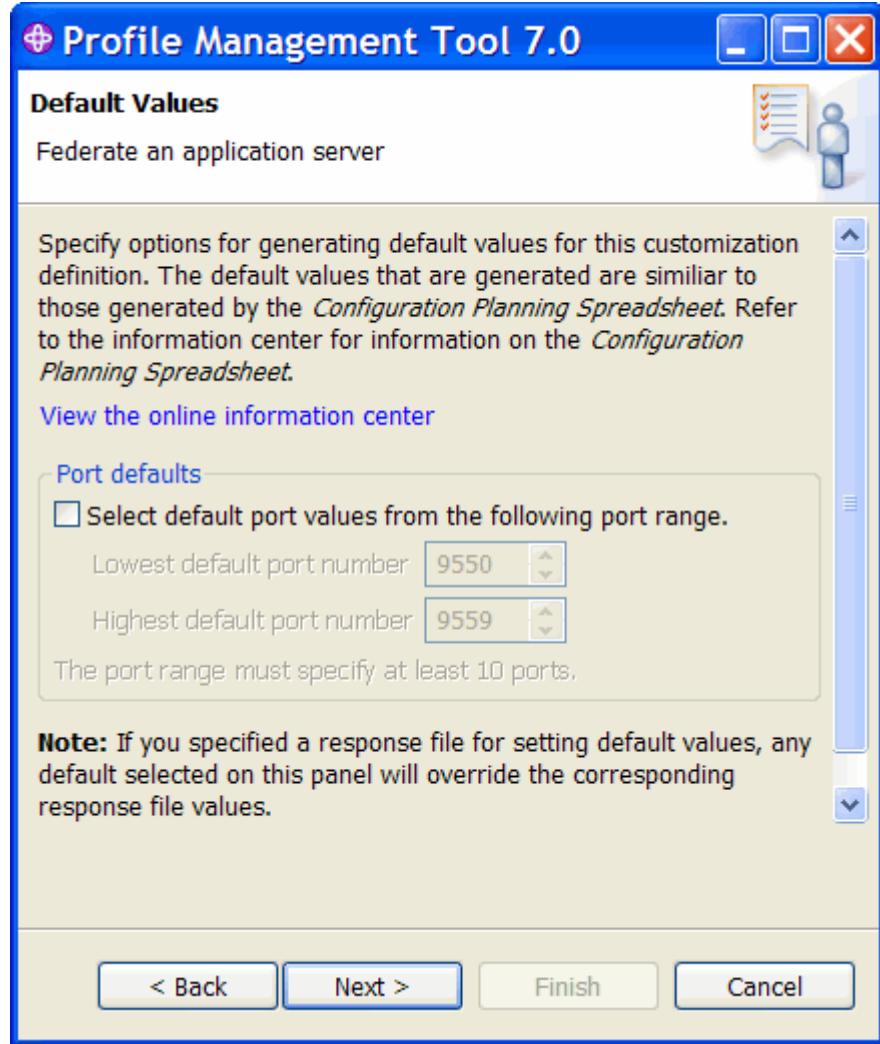


Figure 3-41 Default port settings

This panel allows you to specify a default range of ports to be used for the node agent. When this option is not selected, each port value defaults to an IBM-provided number.

Click **Next**.

4. The Target Data Sets panel allows you to provide the high-level qualifier for the pair of data sets (CNTL and DATA) that will contain the generated jobs. (See Figure 3-11 on page 128).

Click **Next**.

5. The Federate Application Server (Part 1) panel appears as shown in Figure 3-42. This panel is used to specify the information needed to access the application server and the deployment manager.

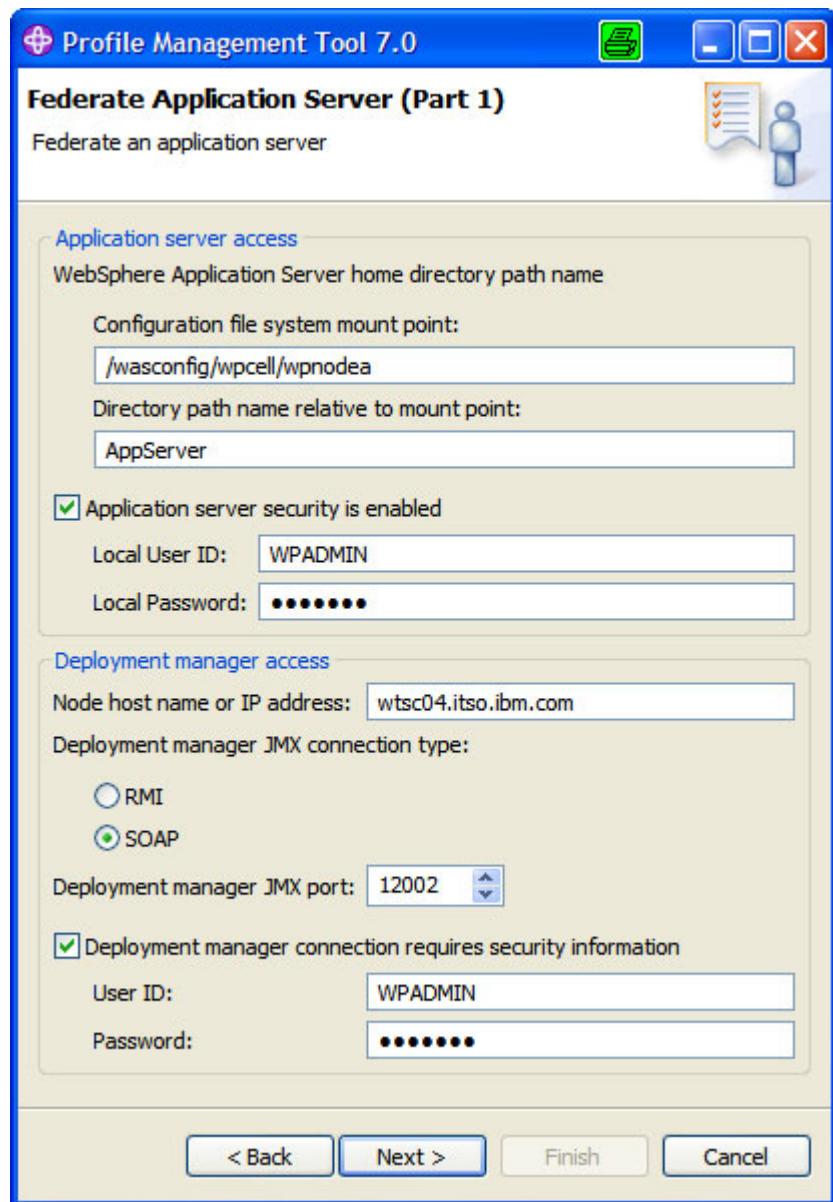


Figure 3-42 Specify Application server settings (Part 1)

- Application server access information:

This section contains the information needed to find and access the application server node configuration file system. This information includes the file system mount point, the directory path name, and if administrative security is enabled on the application server, the administrator user ID and password.

- Deployment manager access information:

Federating the application server requires that the deployment manager be running and accessible by the federation process. This section provides the information required to connect to the deployment manager, including the host name or IP address, JMX connection type and port, and the administrator user ID and password.

The node host name must always resolve to an IP stack on the system where the deployment manager runs. The node host name cannot be a DVIPA or a DNS name that, in any other way, causes the direction of requests to more than one system.

The user ID and password are required when global security is enabled on the Network Deployment cell unless an RMI connector is being used. If an RMI connector is being used, the identity information will be extracted from the thread of execution of the addNode job if the user ID and password are not specified.

Click **Next**.

6. The next panel is shown in Figure 3-43. The output of this panel is used to configure the node agent required for the application server to participate in the distributed environment cell.

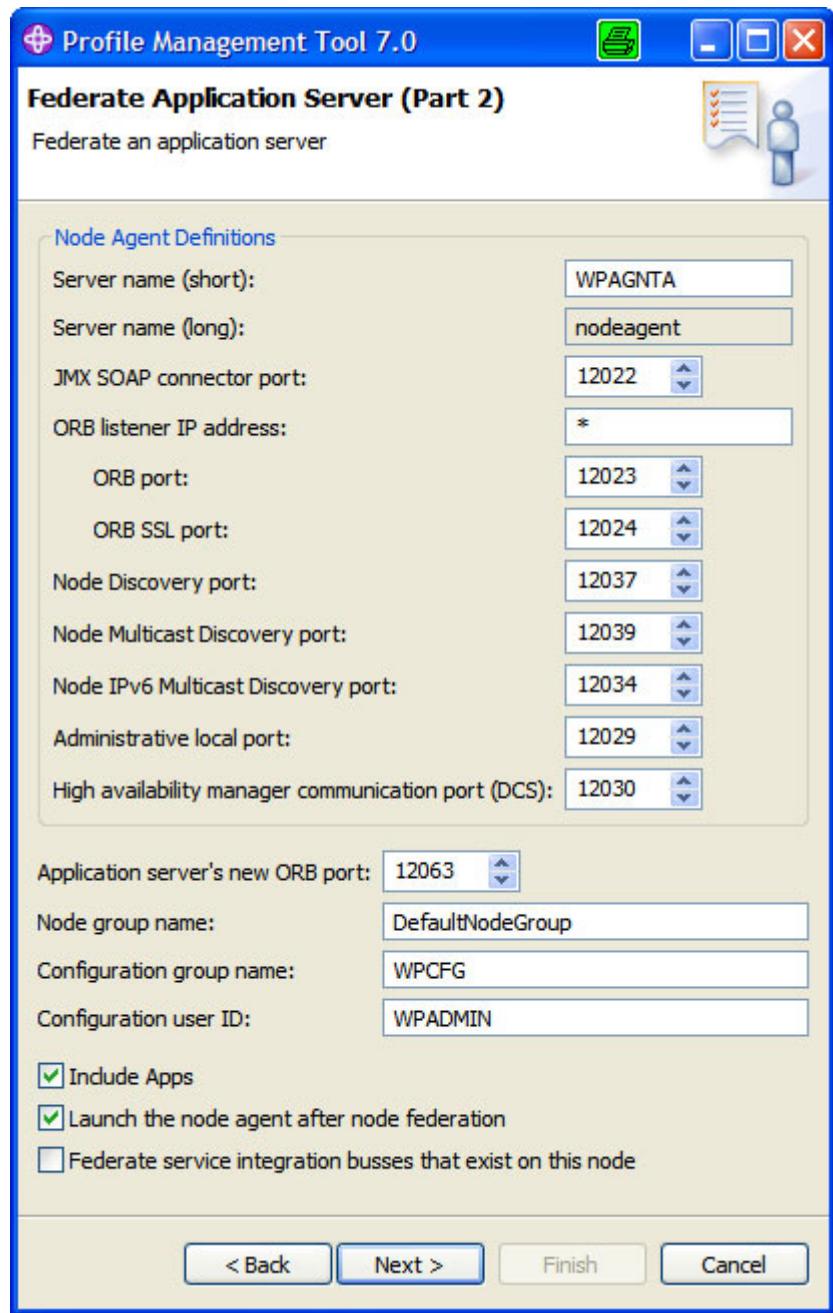


Figure 3-43 Specify Application server settings (Part 2)

- Specify the short name for the node agent process:
The short name is the server's jobname, as specified in the MVS START command JOBNAMES parameter. (The node agent server long name is set to the fixed value of **nodeagent**.)
- Specify the IP addresses and ports to be used by the node agent and a new ORB port to be used by the application server.
- The node group, configuration group, and configuration user ID.
- Select the relevant federation options for your environment. If you have installed applications on the application server or defined service integration buses, you can choose to have those included in the newly federated application server.

Click **Next**.

7. The Job Statement Definition panel is displayed next (see Figure 3-23 on page 148). All the customization jobs that will be created will need a job statement. Enter a valid job statement for your installation and click **Next**.
8. Finally, a summary is displayed. Click **Create**, then **Finish**.

If successful, the next step in the z/OS customization process is to upload these jobs and the associated instructions to a pair of z/OS partitioned data sets. To do this:

1. On the main window, select the customization definition for the profile and click the **Process** button. To upload the generated jobs to the target z/OS system, select **Upload to target z/OS system** and click **Next**.
2. In the upload customization definition window (Figure 3-25 on page 151), enter the target z/OS system. This must be fully qualified or the upload will fail.

Use the **Allocate target z/OS data sets** check box to specify whether to allocate the data sets if they do not exist (box checked). If the data sets exist and are to be reused, clear the box.

Click **Finish**.

You will see a progress information window while the upload is occurring.

3. After the customization profile is uploaded, select the customization definition in the Profile Management Tool and switch to the **Customization Instructions** tab. This provides you with complete instructions on how to build the profile using the jobs.

These instructions can help you determine what jobs to run, the order to run them in, and the expected results. It also tells you how to start the environment after you are done.

After the jobs have been run successfully, the application server profile is complete.

3.7 Creating a job manager profile

In a flexible management environment, the job manager allows you to asynchronously submit and administer jobs for large numbers of unfederated application servers and deployment managers over a geographically dispersed area. Many of the management tasks that you can perform with the job manager are tasks that you can already perform with the product, such as application management, server management, and node management. However, with the job manager, you can aggregate the tasks and perform the tasks across multiple application servers or deployment managers.

The next panel prompts you to select the type of WebSphere environment that you want to create.

1. To generate the definitions for the Job Manager select **Management**. in the PMT environment selection panel and click **Next**.
2. In the next panel, select **Job manager** as the type and click **Next**.
3. Specify a name for the customization definition and click **Next**.
4. The next panel allows you to specify defaults for GID and UID values, name and user ID defaults based on a two character prefix that identifies the cell, and allows you to specify a default range for ports assigned to the process. Click **Next**.
5. The next panel allows you to specify the high level qualifier for the CNTL and DATA data sets that will hold the generated jobs and instructions (see Figure 3-11 on page 128). Specify the value and click **Next**.

6. The next panel (Figure 3-44) allows you to configure the common groups.

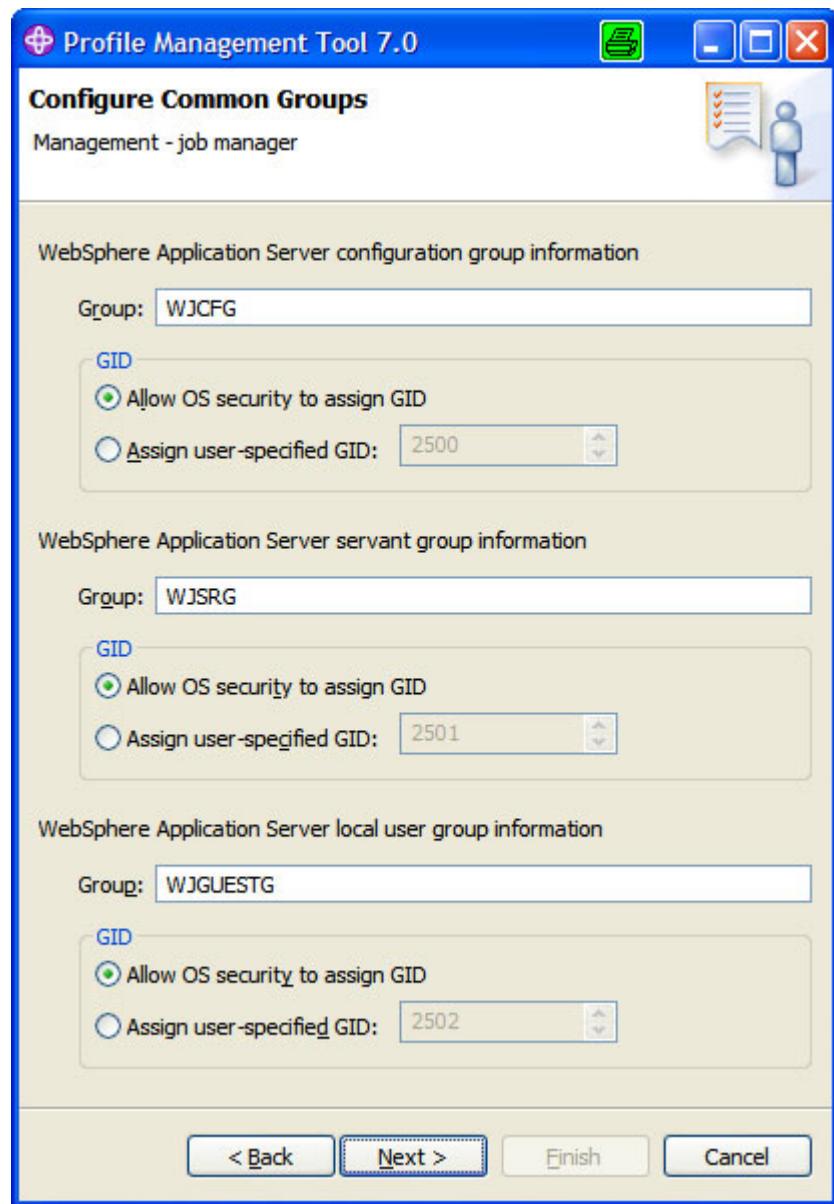


Figure 3-44 Job Manager: Common group settings

Provide the appropriate settings for your environment into the common groups panel and click **Next**.

7. Next, configure the user ID settings (Figure 3-45).

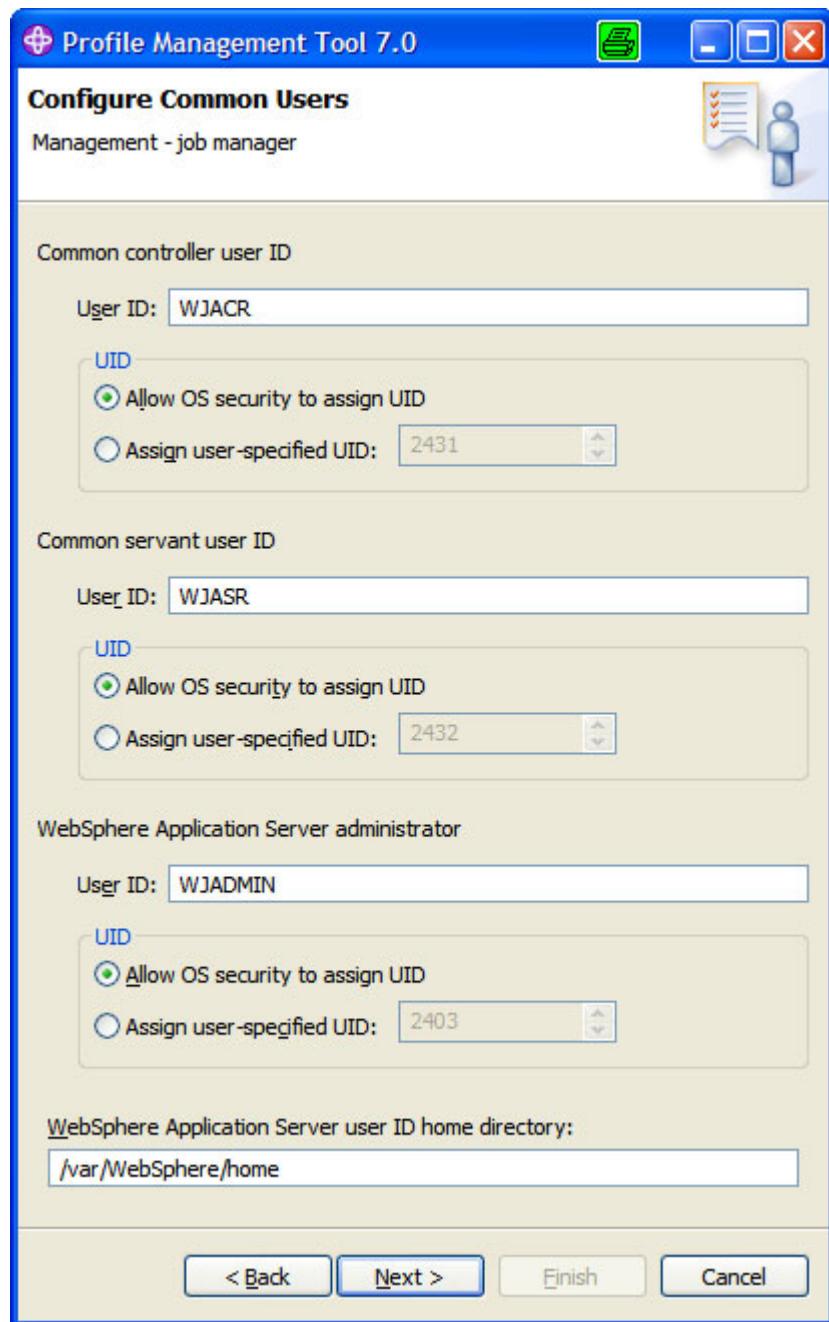


Figure 3-45 Job Manager: Common Users settings

Provide the appropriate settings for your environment and click **Next**.

8. Enter the required information into the System and Data Set Names panel as shown in Figure 3-14 on page 133. Click **Next**.
9. Enter the cell, node, and server names to use for this profile into the panel shown in Figure 3-46 on page 186.

Note: Each management server (administrative agent, deployment manager, or job manager) should be assigned its own cell name that is different from that of any other WebSphere Application Server cell on the same z/OS sysplex.

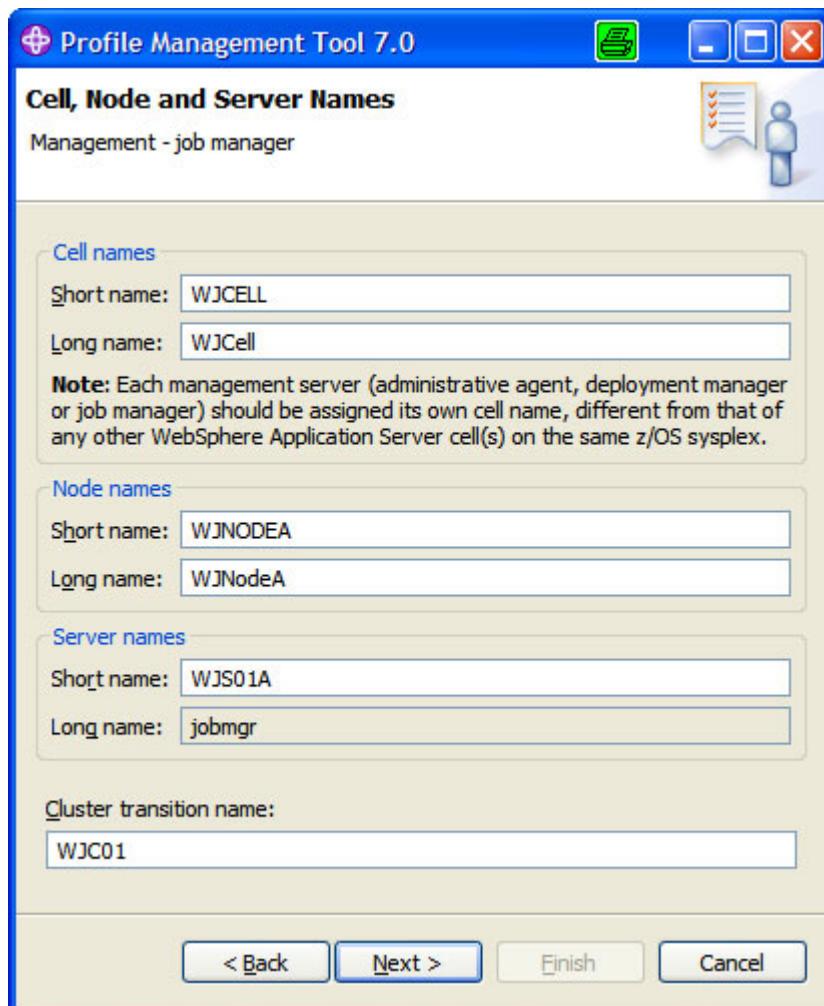


Figure 3-46 Job Manager: cell, node and server names

Click **Next**.

10. Enter the file system information to be used for the job manager configuration data as shown in Figure 3-47 and click **Next**.

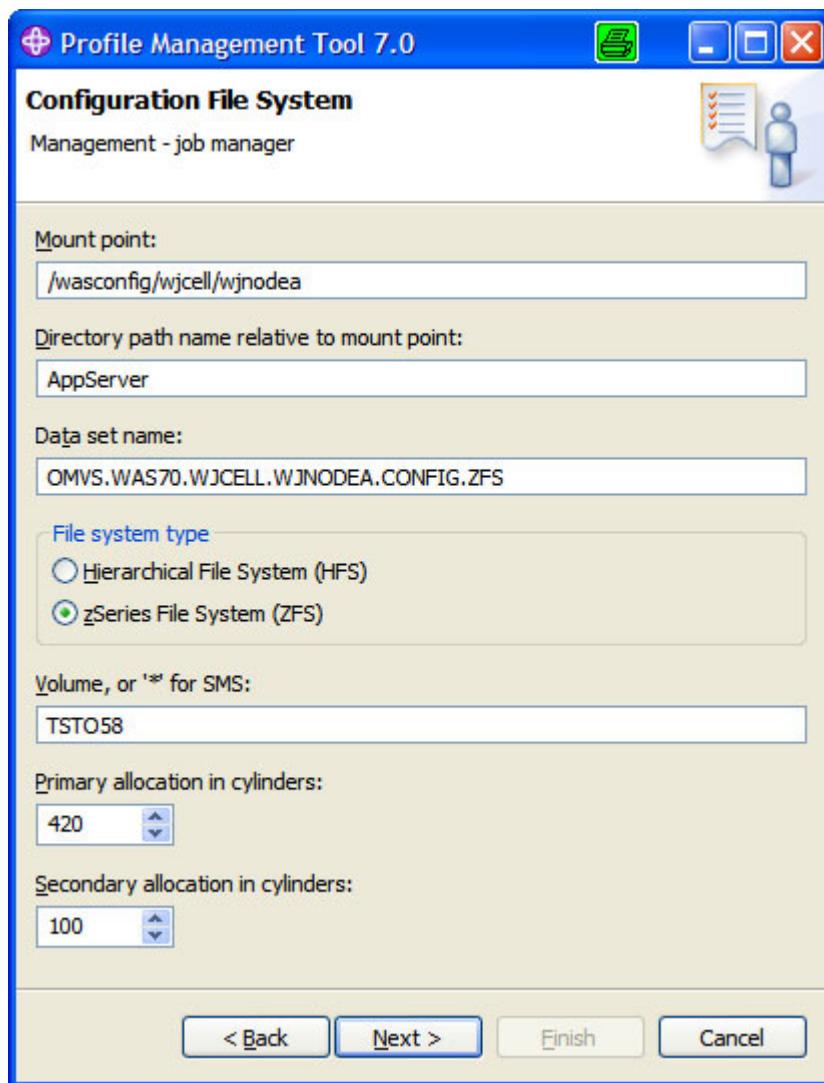


Figure 3-47 Job Manager: file system settings

11. Enter the product file system information for the job manager as shown in Figure 3-48 similarly to what was described for the Application Server environment shown in Figure 3-35 on page 167. Click **Next**.

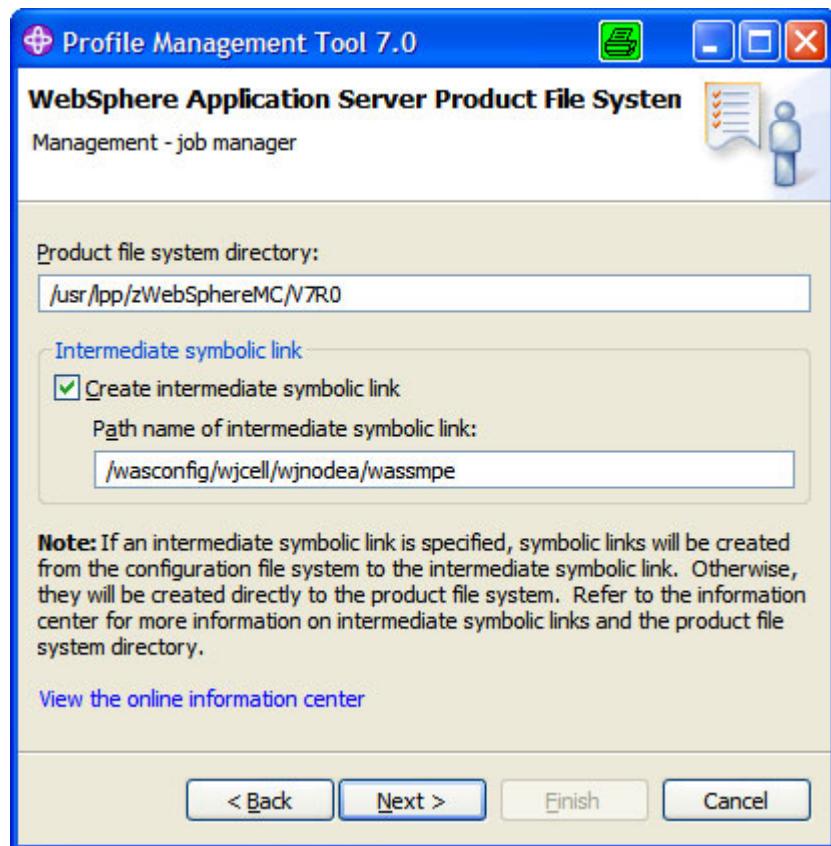


Figure 3-48 Job Manager: product file system location

12. Enter the procedure names to be used for the job manager controller process and server process as shown in Figure 3-49. Click **Next**.

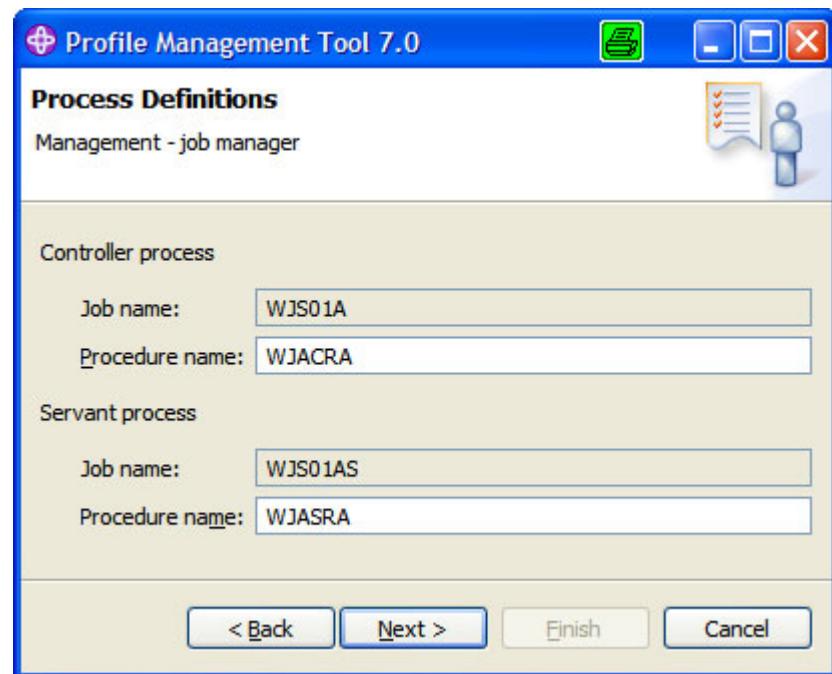


Figure 3-49 Job Manager: Controller and Servant names

13. Enter the job manager port values as shown in Figure 3-50. Click **Next**.

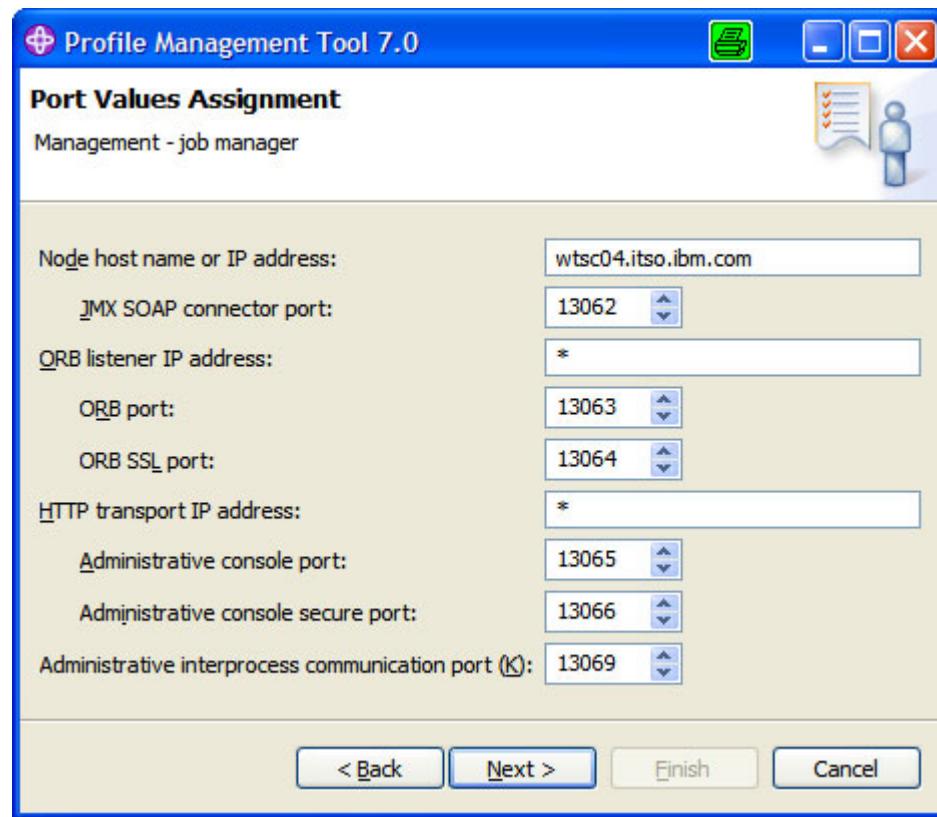


Figure 3-50 Job manager port values

14. Enter the appropriate values into the Location Service Daemon Definitions panel as shown in Figure 3-51. Click **Next**.

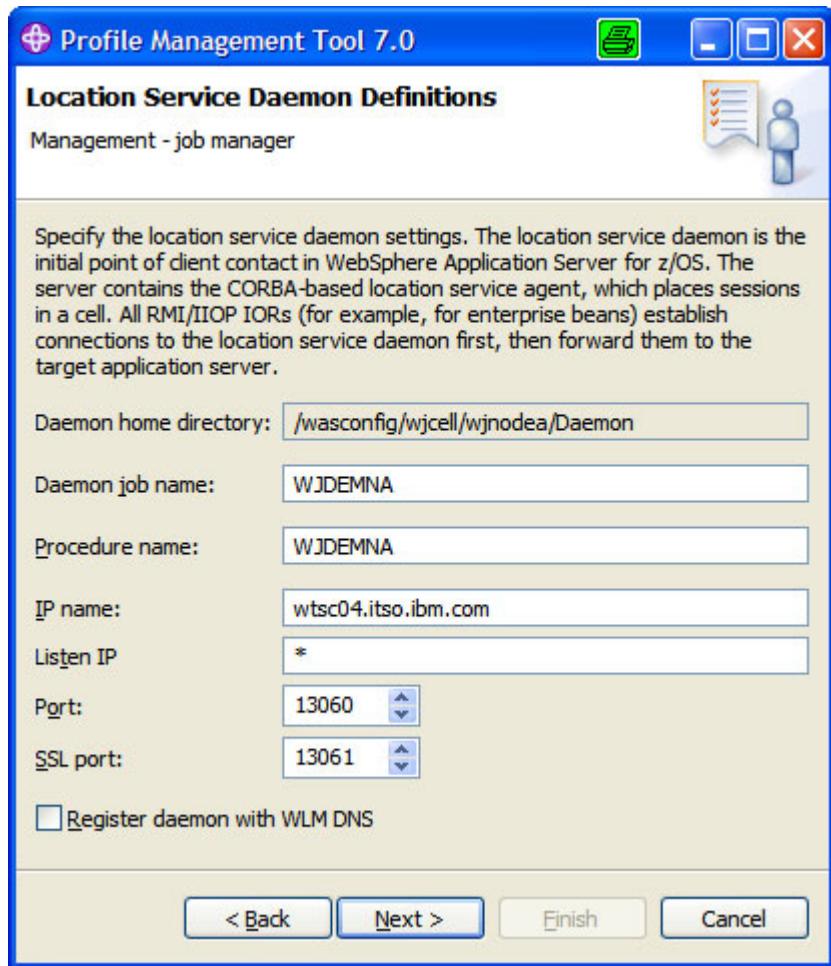


Figure 3-51 Job manager: Location Service Daemon settings

15. The SSL Customization panel is presented next. Enter the information required for SSL connection. (see Figure 3-21 on page 143). Click **Next**.
16. The next panel allows you to select the user registry. Select from one of the following choices:
- Use a z/OS security product
 - Use WebSphere Application Server
 - Do not enable security

Click **Next**. The next panel you see will depend on the user registry option you select.

17. In this example, the **z/OS security product** user registry option was selected in the previous step. The next panel allows you to configure parameters specific to this option (Figure 3-52).

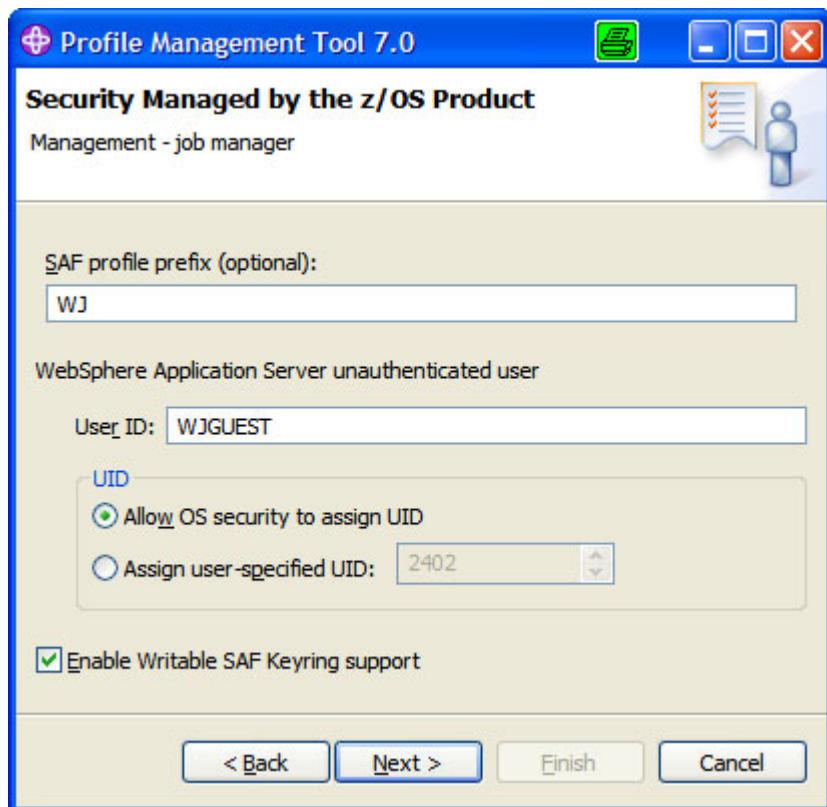


Figure 3-52 Job manager: Security settings

Click **Next**.

18. The next panel allows you to modify the JCL to be used in the generated jobs. Modify the JCL or accept the defaults and click **Next**.
19. The last panel is a summary. Click **Create**, then **Finish** to store this Job manager environment definition for later transfer to the intended z/OS host system.

If successful, the next step in the z/OS customization process is to upload these jobs and the associated instructions to a pair of z/OS partitioned data sets.

To do this:

1. On the main window, select the customization definition for the profile and click the **Process** button. To upload the generated jobs to the target z/OS system select **Upload to target z/OS system** and click **Next**.
2. In the upload customization definition window (Figure 3-25 on page 151), enter the target z/OS system. This must be fully qualified or the upload will fail.

Use the **Allocate target z/OS data sets** check box to specify whether to allocate the data sets if they do not exist (box checked). If the data sets exist and are to be reused, clear the box.

Click **Finish**.

You will see a progress information window while the upload is occurring.

3. After the customization profile is uploaded, select the customization definition in the Profile Management Tool and switch to the **Customization Instructions** tab. This will provide you with complete instructions on how to build the profile using the jobs.

These instructions will help you determine what jobs to run, the order to run them in, and the expected results. It will also tell you how to start the environment after you are done.

After the jobs have been run successfully, the application server profile is complete.

3.8 Creating an administrative agent profile

The administrative agent provides a single interface to administer multiple application server nodes in, development, unit test, or server farm environments, for example. The administrative agent provides a single interface to administer multiple unfederated application server nodes in, for example, development, unit test, or server farm environments. By using a single interface to administer your application servers, you reduce the overhead of running administrative services in every application server.

1. To generate the definitions for the administration agent select **Management** in the PMT environment selection panel and click **Next**.
2. Select **Administrative agent** as the server type and click **Next**.
3. Enter a name for the customization definition and click **Next**.
4. The next panel allows you to specify defaults for GID and UID values, name and user ID defaults based on a two character prefix that identifies the cell, and allows you to specify a default range for ports assigned to the process.

Click **Next**.

5. The next panel allows you to specify the high level qualifier for the CNTL and DATA data sets that will hold the generated jobs and instructions (see Figure 3-11 on page 128). Specify the value and click **Next**.
6. Configure the common group settings as shown in Figure 3-53.

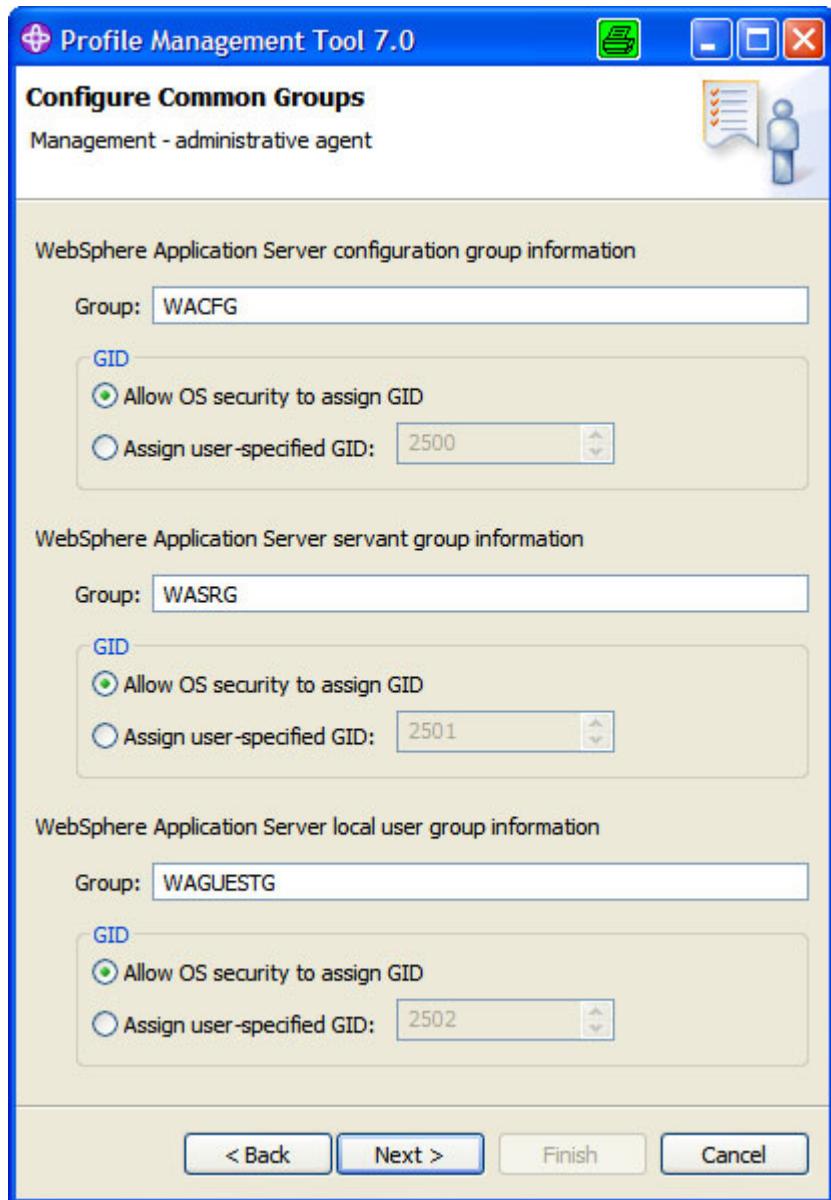


Figure 3-53 Common groups settings

Provide the appropriate settings for your environment and click **Next**.

7. Configure the common user settings as shown in Figure 3-54.

The screenshot shows a configuration dialog for administrative users. It is divided into four sections: 'Common controller user ID', 'Common servant user ID', 'WebSphere Application Server administrator', and 'WebSphere Application Server user ID home directory'. Each section contains a 'User ID' input field and a 'UID' configuration group. In the 'UID' group, the 'Allow OS security to assign UID' option is selected. The 'User ID' fields contain 'WAACR', 'WAASR', and 'WAADMIN' respectively. The 'UID' fields contain '2431', '2432', and '2403'. The 'WebSphere Application Server user ID home directory' field contains '/var/WebSphere/home'. At the bottom are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

Common controller user ID			
User ID:	WAACR		
UID			
<input checked="" type="radio"/> Allow OS security to assign UID			
<input type="radio"/> Assign user-specified UID:	2431		
Common servant user ID			
User ID:	WAASR		
UID			
<input checked="" type="radio"/> Allow OS security to assign UID			
<input type="radio"/> Assign user-specified UID:	2432		
WebSphere Application Server administrator			
User ID:	WAADMIN		
UID			
<input checked="" type="radio"/> Allow OS security to assign UID			
<input type="radio"/> Assign user-specified UID:	2403		
WebSphere Application Server user ID home directory:			
/var/WebSphere/home			
< Back	Next >	Finish	Cancel

Figure 3-54 Administrative agent: common users

Provide the appropriate settings for your environment and click **Next**.

8. Next enter the system name, sysplex name, and PROCLIB data set name (See Figure 3-14 on page 133). Click **Next**.
9. Enter the cell, node, and server names as shown in Figure 3-55 and click **Next**.

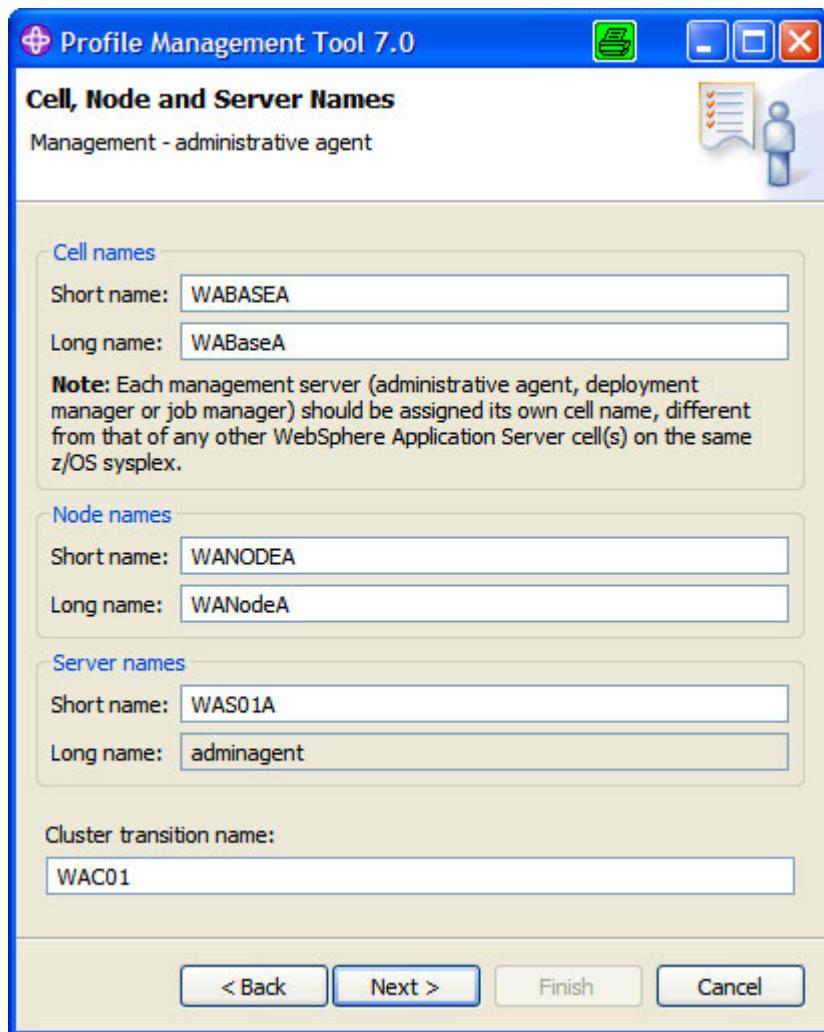


Figure 3-55 Cell, node and server names

10. Enter the file system information for the administrative agent as shown in Figure 3-56.

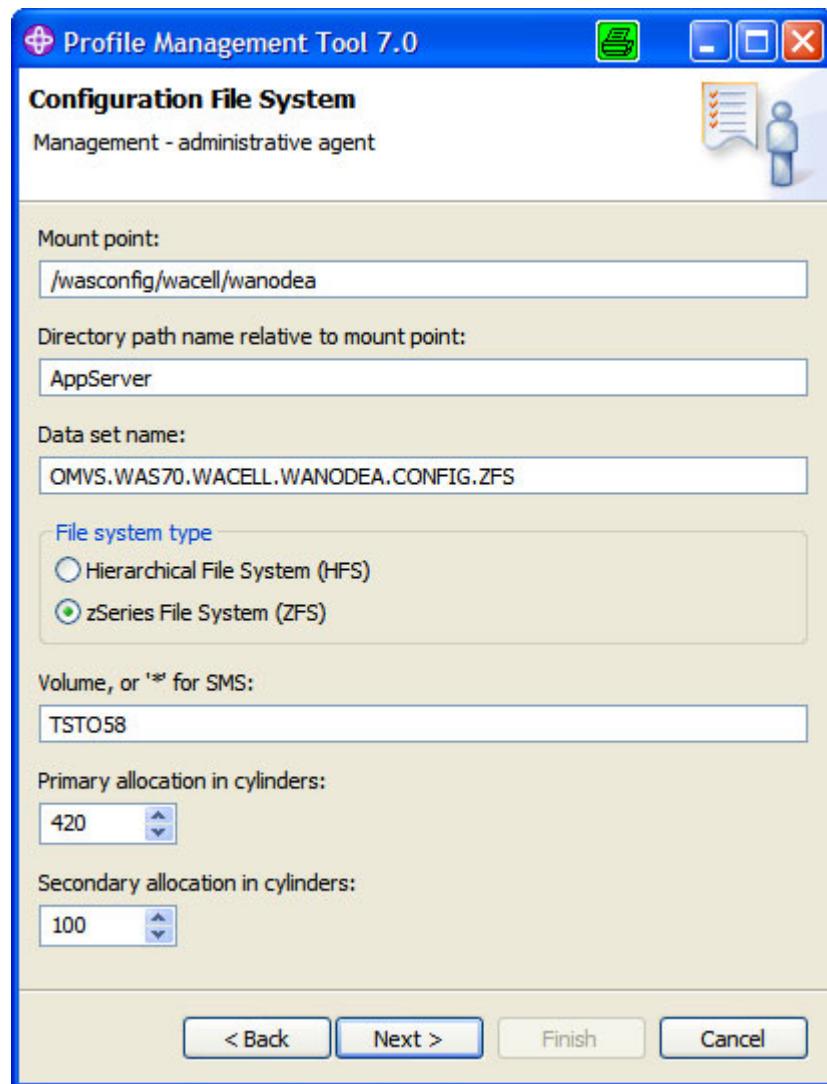


Figure 3-56 Configuration file system settings

11. Enter the product file system information for the administrative agent as shown in Figure 3-57 and click **Next**.

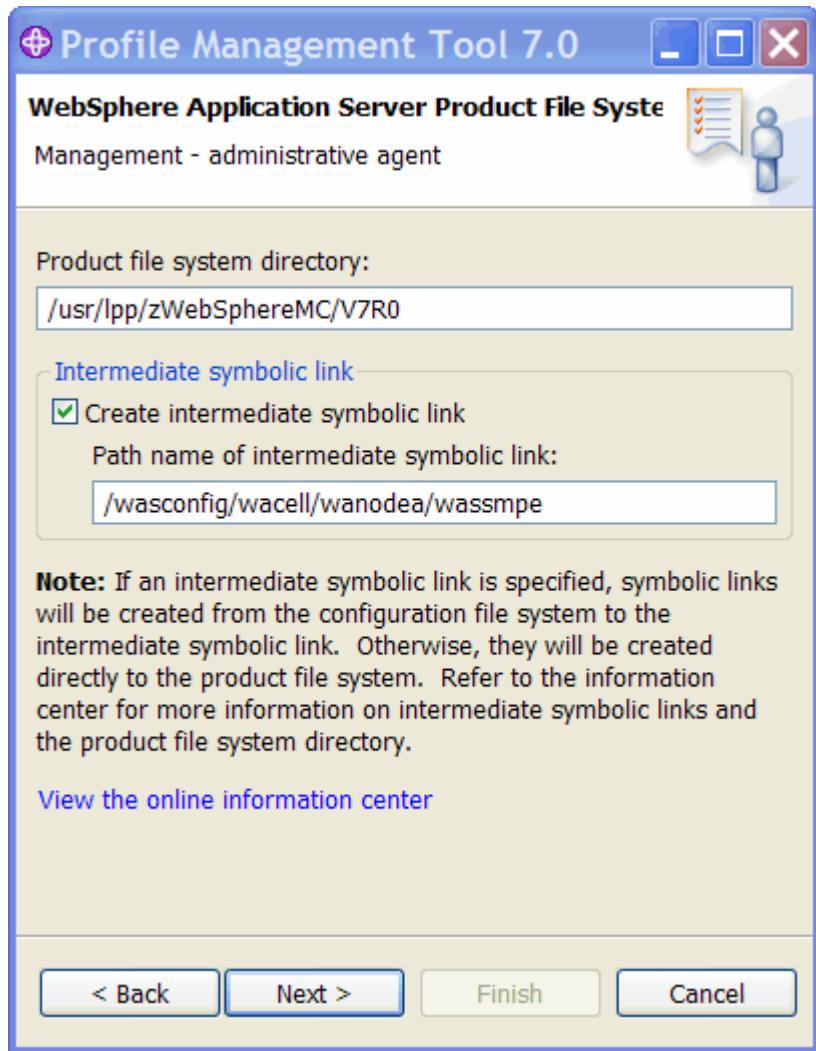


Figure 3-57 Administrative agent: product file system

12. Enter the job manager controller process and server process names as shown in Figure 3-58 and click **Next**.

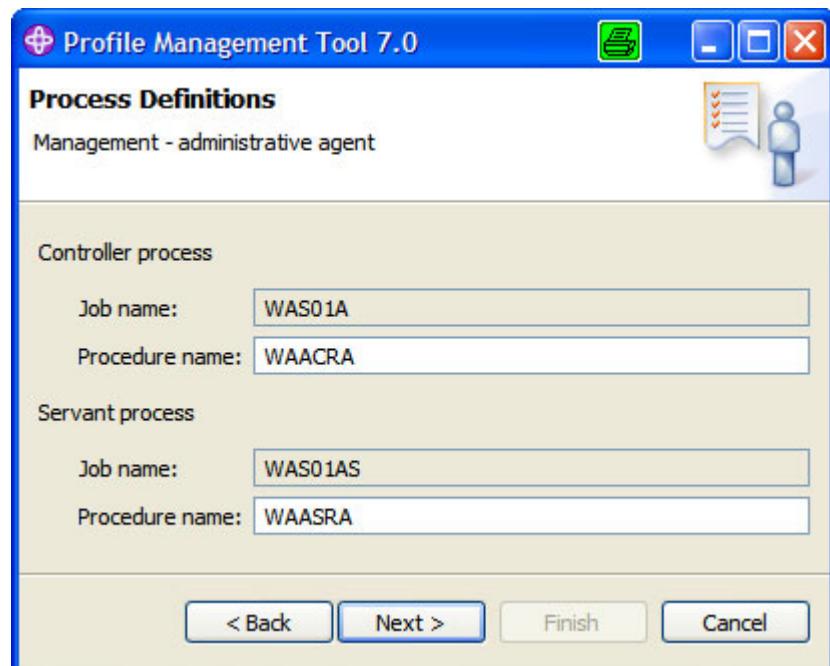


Figure 3-58 Controller and servant names

13. Enter the administrative agent port values as shown in Figure 3-59 similarly and click **Next**.

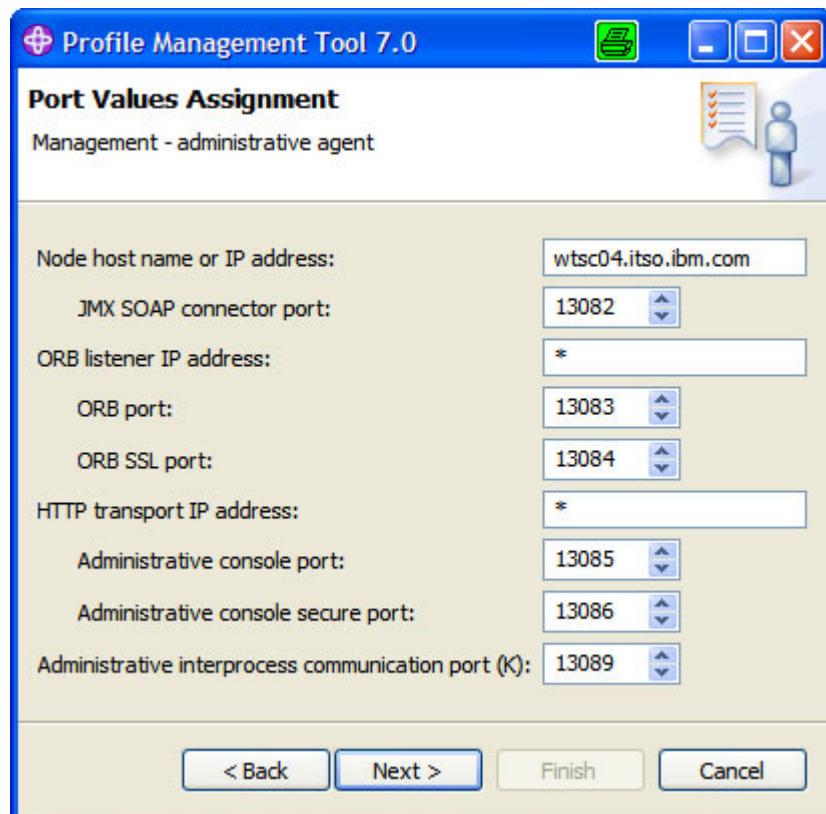


Figure 3-59 Administrative agent: port assignments

14. Enter the appropriate values into the Location Service Daemon Definitions panel as shown in Figure 3-60 and click **Next**.

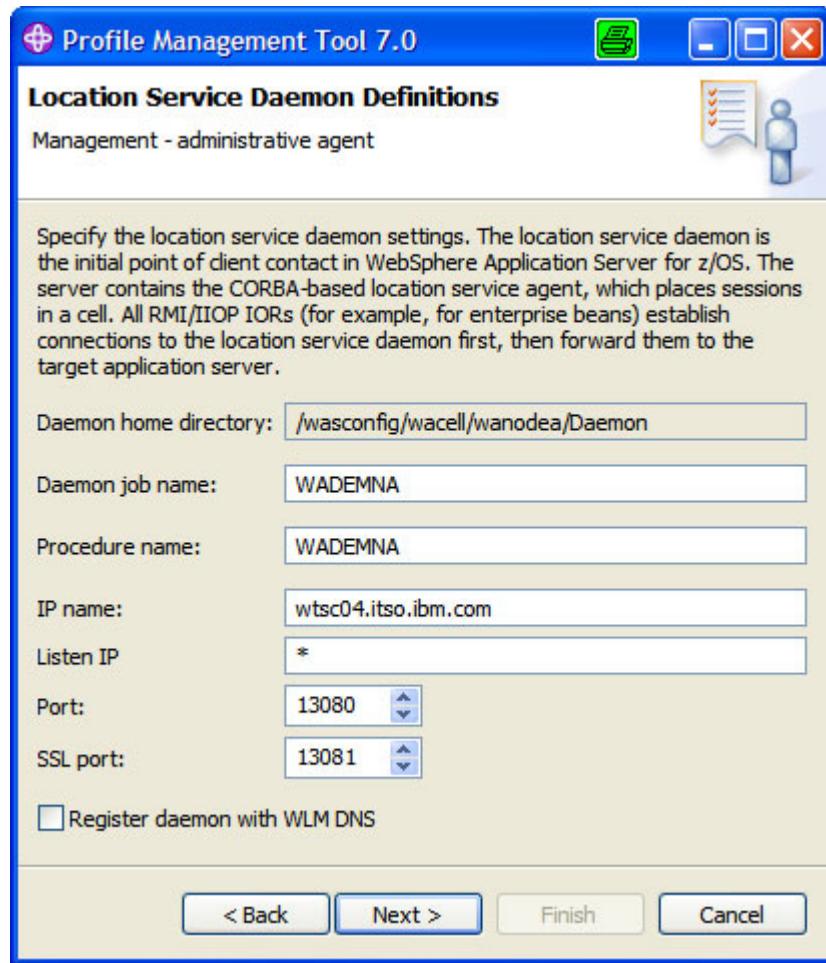


Figure 3-60 Location service daemon settings

15. The SSL Customization panel is presented next. Enter the information required for SSL connection. (see Figure 3-21 on page 143). Click **Next**.
16. The next panel allows you to select the user registry. Select from one of the following choices:
- Use a z/OS security product
 - Use WebSphere Application Server
 - Do not enable security

Click **Next**. The next panel you see will depend on the user registry option you select.

17. In this example, the **z/OS security product** user registry option was selected in the previous step. The next panel allows you to configure parameters specific to this option (Figure 3-61).

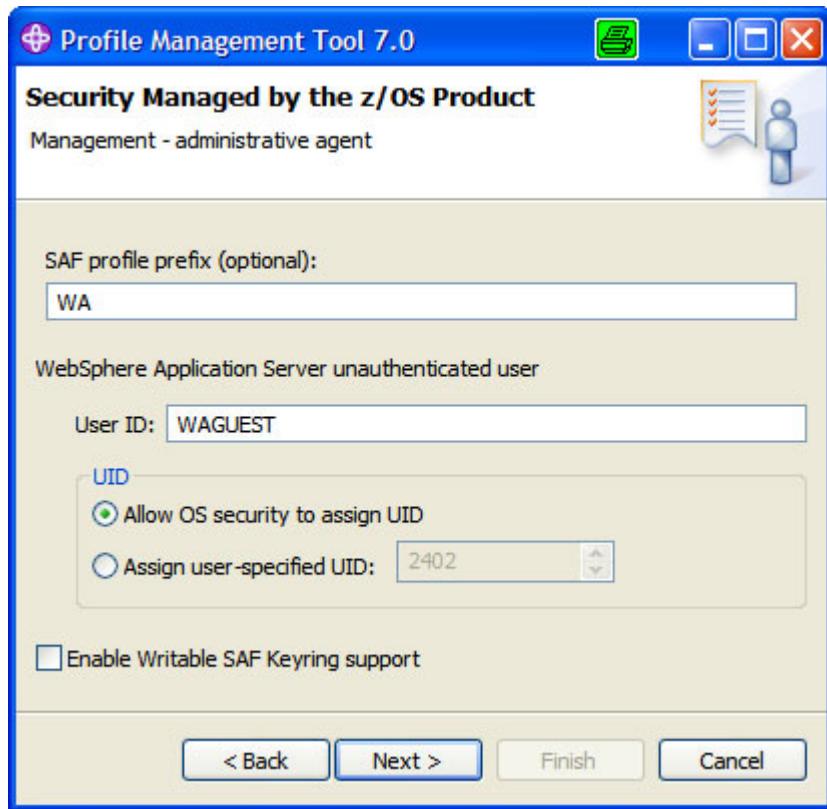


Figure 3-61 Administration agent security settings

18. The next panel allows you to modify the JCL to be used in the generated jobs. Modify the JCL or accept the defaults and click **Next**.
19. The last panel is a summary. Click **Create**, then **Finish** to store this Job manager environment definition for later transfer to the intended z/OS host system.

If successful, the next step in the z/OS customization process is to upload these jobs and the associated instructions to a pair of z/OS partitioned data sets.

To do this:

1. On the main window, select the customization definition for the profile and click the **Process** button. To upload the generated jobs to the target z/OS system select **Upload to target z/OS system** and click **Next**.
2. In the upload customization definition window (Figure 3-25 on page 151), enter the target z/OS system. This must be fully qualified or the upload will fail.

Use the **Allocate target z/OS data sets** check box to specify whether to allocate the data sets if they do not exist (box check). If the data sets exist and are to be reused, clear the box.

Click **Finish**.

You will see a progress information window while the upload is occurring.

3. After the customization profile is uploaded, select the customization definition in the Profile Management Tool and switch to the **Customization Instructions** tab. This will provide you with complete instructions on how to build the profile using the jobs.

These instructions will help you determine what jobs to run, the order to run them in, and the expected results. It will also tell you how to start the environment after you are done.

After the jobs have been run successfully, the application server profile is complete.

After the administrative agent and standalone application servers have been created on the system, use the **registerNode** command to register the application server nodes to the administrative agent.



Centralized Installation Manager

(New in V7) The Centralized Installation Manager (CIM) is a new feature with WebSphere Application Server Network Deployment, Version 7.0 for distributed platforms. This feature can be used to consolidate and simplify the steps required to perform installations and apply maintenance on systems in the Network Deployment cell.

Prior to CIM, administrators had to log on to every machine in the cell, and manually install the code and maintenance. With CIM, the administrator only has to select the machine name, provide login credentials, and CIM handles the rest. CIM allows for installing and uninstalling of the WebSphere application server binaries and maintenance patches on the following components:

- ▶ WebSphere Application Server Network Deployment Version 7
- ▶ WebSphere Application Server Version 7 refresh packs, fix packs, and interim fixes
- ▶ WebSphere Application Server Version 6.1 refresh packs, fix packs, and interim fixes
- ▶ Update Installer for WebSphere Application Server Version 7

Only Network Deployment packages and Network Deployment customized installation packages are supported in a CIM repository.

4.1 Planning considerations

The CIM does not take the place of Update Installer or Installation Factory for WebSphere Software. CIM pushes the product binary files or maintenance to the remote targets and invokes the standard installer or update installer tool to perform the installation or apply updates on the target systems.

CIM can be used to complete the following tasks:

- ▶ 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 site. 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.
- ▶ Monitor downloads and installation status through the administrator console.

Each WebSphere Application Server installation only contains one CIM repository. This repository is shared among all the deployment managers in the installations. Both products and maintenance can be made available in this repository. The CIM repository can be created during the installation process of WebSphere Application Server Version 7.0 or the IBM WebSphere Installation Factory can be used later to create it.

Approximately 3 GB of disk space is required for each platform you decide to have in the repository. If you plan to create Customized Installation Packages (CIPs) for use with CIM, additional disk space will be required.

The CIM relies on current information regarding the versions of WebSphere Application Server that are installed on each node. This information is kept current on the deployment manager configuration by the node agent running on each node. The deployment manager is aware of the correct versions of WebSphere Application Server that are installed on each node, if the node agent of each node is started at least once after an update is applied. To ensure that the deployment manager receives this information, the CIM automatically starts the node agent after each installation or uninstallation of maintenance.

The CIM relies on the node agent to effectively stop the server processes on the target node, if the node agent is not running, it is up to the administrator to ensure that all the server processes are stopped on the target node before initiating any maintenance update operations on the node.

4.1.1 Linux and AIX target requirements

The CIM, through RXA, uses SSH Version 2 to access UNIX and Linux target workstations. This usage requires the use of either OpenSSH 3.6.1 or, if accessing AIX targets, OpenSSH 4.7 on the target hosts.

Using Secure Shell (SSH) protocol

Remote Execution and Access does not supply SSH code for UNIX operating systems. You must ensure SSH is installed and enabled on any target you want to access using CIM.

In AIX and Linux environments the Bourne shell (sh) is used as the target shell. To communicate with Linux and other SSH targets using password authentication, you must edit the /etc/ssh/sshd_config file on the targets and set the following property:

```
PasswordAuthentication yes
```

The default value for the PasswordAuthentication property is no.

After changing this setting, stop and restart the SSH daemon using the following commands:

```
/etc/init.d/sshd stop  
/etc/init.d/sshd start
```

Installing a secure shell public key to access remote targets

UNIX platforms generally support the use of SSH protocol. To use the SSH public/private key as an authentication method for accessing your remote workstations, SSH must be installed and enabled on the installation target system. On AIX and Linux systems issue the following command to ensure SSH is enabled on the installation target:

```
ps -e | grep sshd
```

You can generate an RSA private key and its corresponding public key using the ssh-keygen command as in the following example:

```
ssh-keygen t rsa
```

Take the default location for storing the private key and make note of it. If you specify a non-empty string for the passphrase prompt, make sure that you can remember the string, because you will need it when you want to use the generated private key.

Additionally, you must know the location of the SSH public key file on the deployment manager, and the administrative ID and password for the installation target. This is the same administrative ID and password that you use to later install or uninstall software packages on the same installation target.

4.1.2 Requirement when using CIM for installing or uninstalling maintenance on AIX target as non-root user

Before using the CIM to install or uninstall maintenance on IBM AIX operating systems as a non-root user, you must install and configure sudo, an open-source tool, on the target AIX operating systems.

To install and configure sudo, logon to root on the local system and download sudo from the IBM AIX Toolkit website:

<http://www-03.ibm.com/systems/power/software/aix/linux/toolbox/download.html>

After it is downloaded, issue the following command to install sudo:

```
rpm -i sudo-*.rpm
```

You can download an AIX installp image for the rpm package manager for POWER® from the previous download website if your AIX system does not already have rpm installed.

Authorize a non-root user ID, which you specify, to run the `slibclean` command as a root user without providing a password. Issue the `visudo` command to add the following entry to the /etc/sudoers configuration file (replace *userid* with the real user id you will be using):

```
userid ALL = NOPASSWD: /usr/sbin/slibclean
```

Log in with the specified user ID, and issue the `sudo -l` command. If successful, a message that is similar to the following example is displayed:

```
User userid may run the following commands on this host:  
(root) NOPASSWD: /usr/sbin/slibclean
```

If you do not have sudo installed, or sudo is installed but not configured correctly for the specified user ID, error messages are displayed.

4.1.3 Update Installer

The CIM installs an appropriate level of the Update Installer on the target systems, which it will use to install fix packs and other maintenance. If you previously installed the Update Installer tool on any of the target hosts in a directory location other than *install_root*/UpdateInstaller, you might want to consider uninstalling the Update Installer using its uninstall process. That copy will not be used by the CIM. It is not required to uninstall the previous copy of Update Installer for CIM to work properly.

When you install fix packs or other maintenance on the target systems, CIM will install the Update Installer tool, if the option is selected. If the version of the Update Installer tool found in *install_root*/UpdateInstaller directory does not meet the minimum version required by the interim fix or fix pack, the CIM automatically installs a newer version on the target, if the newer version has been downloaded to your CIM repository.

You cannot use CIM to install the Update Installer on nodes that are not federated to the deployment manager cell.

4.1.4 Repository directory structure

The CIM repository consists of directories that contain the installation images for product files, maintenance files and the update installer. Below is the list of the directory names and files types contained in the directories:

- ▶ UPDI70: This directory holds the 7.0.0.0-WS-UPDI-*.zip file, which contains the installation images of Update Installer for the operating systems that you want in your repository. Example of files that would be copied into this directory.
7.0.0.1-WS-UPDI-AixPPC32.zip
- ▶ WAS70Updates: This directory holds the .pak files, which contain interim fixes for WebSphere Server Network Deployment Version 7.0. These files can be removed when no longer required.
7.0.0.1-WS-WAS-IFPK75887.PAK
- ▶ WAS70FPn: This directory contains the .pak files that make up a specific fix pack for WebSphere Server Network Deployment Version 7.0. Below are examples of the files that would be copied into the WAS70FP1 directory. Refer to the WebSphere Application Server Version 7.0 support website for the list of files required for each fix pack.

For example, for WebSphere Application Server Network Deployment Version 7.0 Fix Pack 1, copy the following .pak files to the WAS70FP1 directory:

7.0.0-WS-WAS-*platform_architecture*-FP0000001.pak
7.0.0-WS-WASSDK-*platform_architecture*-FP0000001.pak

- ▶ ND61Updates: This directory holds the .pak files, which contain interim fixes for WebSphere Server network Deployment Version 6.1.
- ▶ ND61FPn: This directory contains the .pak files that make up a specific fix pack for WebSphere Application Server Version 6.1. Refer to the WebSphere Application Server Version 6.1 support website for the list of files required for each fix pack. For example, for WebSphere Application Server Network Deployment Version 6.1 Fix Pack 23, copy the following .pak files into the ND61FP23 directory:

6.1.0-WS-WAS-*platform_architecture*-FP0000023.pak
6.1.0-WS-WASSDK-*platform_architecture*-FP0000023.pak
6.1.0-WS-WASWebSvc-*platform_architecture*-FP0000023.pak
6.1.0-WS-WASEJB3-*platform_architecture*-FP0000023.pak

- ▶ WAS70: This directory is created during the installation of your CIM repository. It contains the product installation file for the operating systems in your environment:

jdk.7000.aix.ppc32.zip
was.nd.7000.aix.ppc32.zip
- ▶ Descriptors: This directory is created at the time of the CIM install and will contain the descriptor files:

InstallPackageND61FP15.xml

4.2 Installing CIM and creating the repository

During the installation of WebSphere Application Server Network Deployment Manager, you will be presented with the panel in Figure 4-1. To have the repository initialize and the current installation package loaded to the repository, follow these steps:

1. Launch the Installation wizard for WebSphere Application Server Network Deployment
2. To create and add the installation package to the CIM repository, select the box to **Create a repository for Centralized Installation Manager**.
3. Enter the directory path where you want the repository installed.

4. Select the box to populate the repository with this installation package (Figure 4-1).
5. Click **Next** to continue.

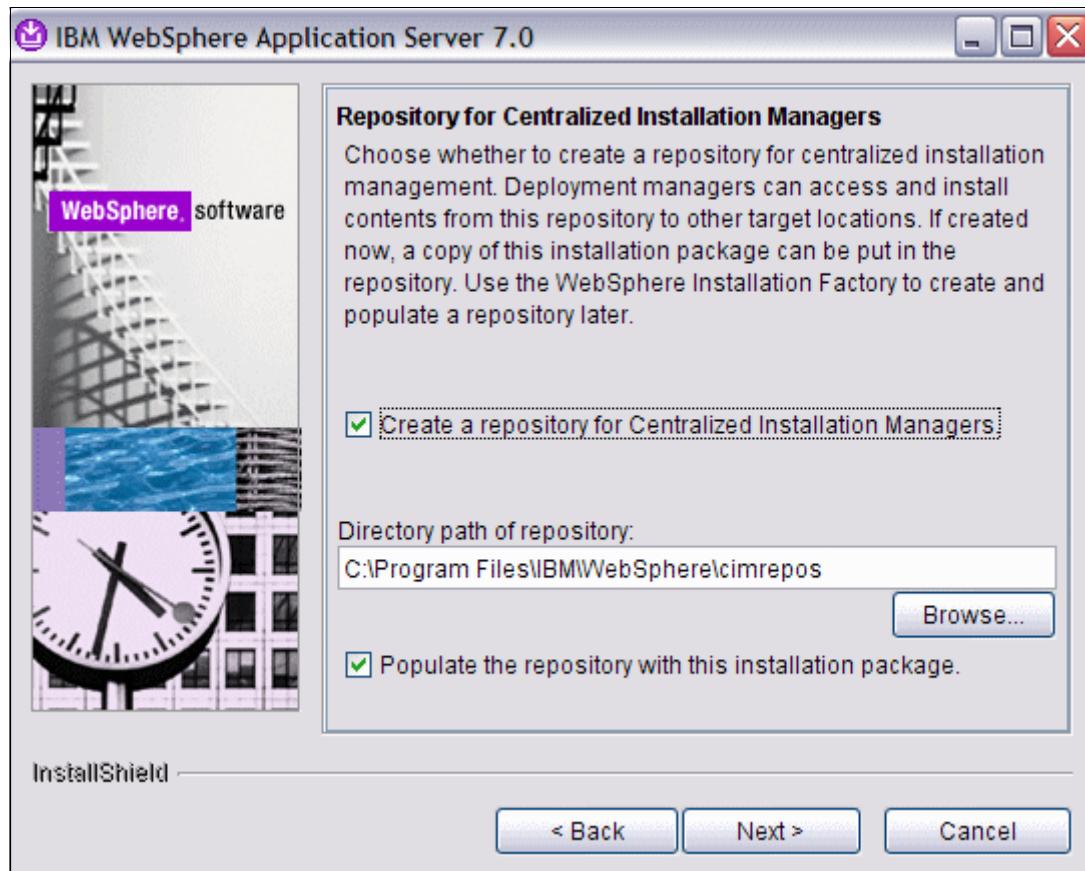


Figure 4-1 Installation Wizard

If you do not check the boxes to create and populate the repository during the initial install, Installation Factory can be used to create and populate the repository at a later time.

For information about creating an options file and doing a silent install, refer to the IBM Information Center for IBM WebSphere Application Server Network Deployment 7.0:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.installation.nd.doc/info/ae/ae/tins_cim_package_add_i nstall.html

4.3 Loading additional product packages into the repository

To add additional installation packages to your Central Installation Management repository, you can use Installation Factory, which is included with WebSphere Application Server Network Deployment Version 7.0, on the product CD or from the product download site:

- ▶ IBM WebSphere Installation Factory on AIX, HP-UX, Linux, and Solaris operating systems can create installation packages for all supported platforms.
- ▶ IBM WebSphere Installation Factory on the Windows operating system can create installation packages for Windows and IBM i platforms.
- ▶ Installation Factory does not run on the i5/OS® platform.

The package type includes the WebSphere Application Server Deployment manager Version 7.0. The descriptor files and binary files for product packages are not available to download separately. The files are included during the product packaging, therefore included when the product is loaded into the CIM repository from the product CD or product download site.

4.3.1 Installing Installation Factory

You can download the IBM WebSphere Installation Factory image from the IBM Installation Factory for WebSphere Application Server website:

<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg24020213>

After it is downloaded, unpack the code to any directory for which you have the appropriate permissions. You can also install Installation Factory on your system using the code from the product media. Copy the Installation Factory on your operating system use the **setupif** command provided on the Installation Factory disc, or as part of the package download:

- ▶ UNIX: Run the **setupif.sh** command or **setupif.sh target_location**
- ▶ Windows: Run the **setupif.bat** command or **setupif.bat target_location**

This command copies the Installation Factory to *user_home*/InstallationFactory by default. You can specify the target location by using the target location parameter.

4.3.2 Package types

There are four types of installation packages:

Product installation

The descriptor and binary files are included in product packaging and are loaded at the time the product is loaded into the repository. The descriptors that are included during the product installation are:

- ▶ Maintenance for WebSphere Application Server Network Deployment 6.1 descriptor files provided by the product installation
- ▶ Maintenance for WebSphere Application Server Network Deployment 7.0
- ▶ Update Installer for WebSphere Application Server 7.0
- ▶ WebSphere Application Server Network Deployment 7.0

Maintenance tool

This package contains the Update Installer, the tool used to apply maintenance to WebSphere Application Server environment. Before using CIM to apply maintenance on remote systems you must download the latest level of the Update Installer. Fix packs must first be installed locally on the deployment manager system using Update Installer.

Refresh and fix packs

With this package type you can download binary files based on a specific platforms. When refresh or fix pack for IBM WebSphere Application Server is released it usually comes with a fix pack for Java SDK. CIM requires both fix packs in the repository and will install both fix packs to the selected targets.

Interim fixes

Specify the APAR number of the interim fix and click **Search** to display a list of binary files associated with the interim fix.

4.3.3 Adding product packages to the CIM repository

Before you can populate the CIM repository, ensure that you have write access to the directories you will be using. Repeat this process for all product packages you would like to add to the CIM repository.

1. Download the product images and expand the file (tar or zip) to a temporary directory or have access to the product CD. Start the IBM WebSphere Installation Factory console with the **ifgui** command.
 - UNIX: **cip_root/bin/ifgui.sh**
 - Windows: **cip_root\bin\ifgui.bat**

2. Select **Manage Repository for Centralized Installation Manager**. See Figure 4-2.

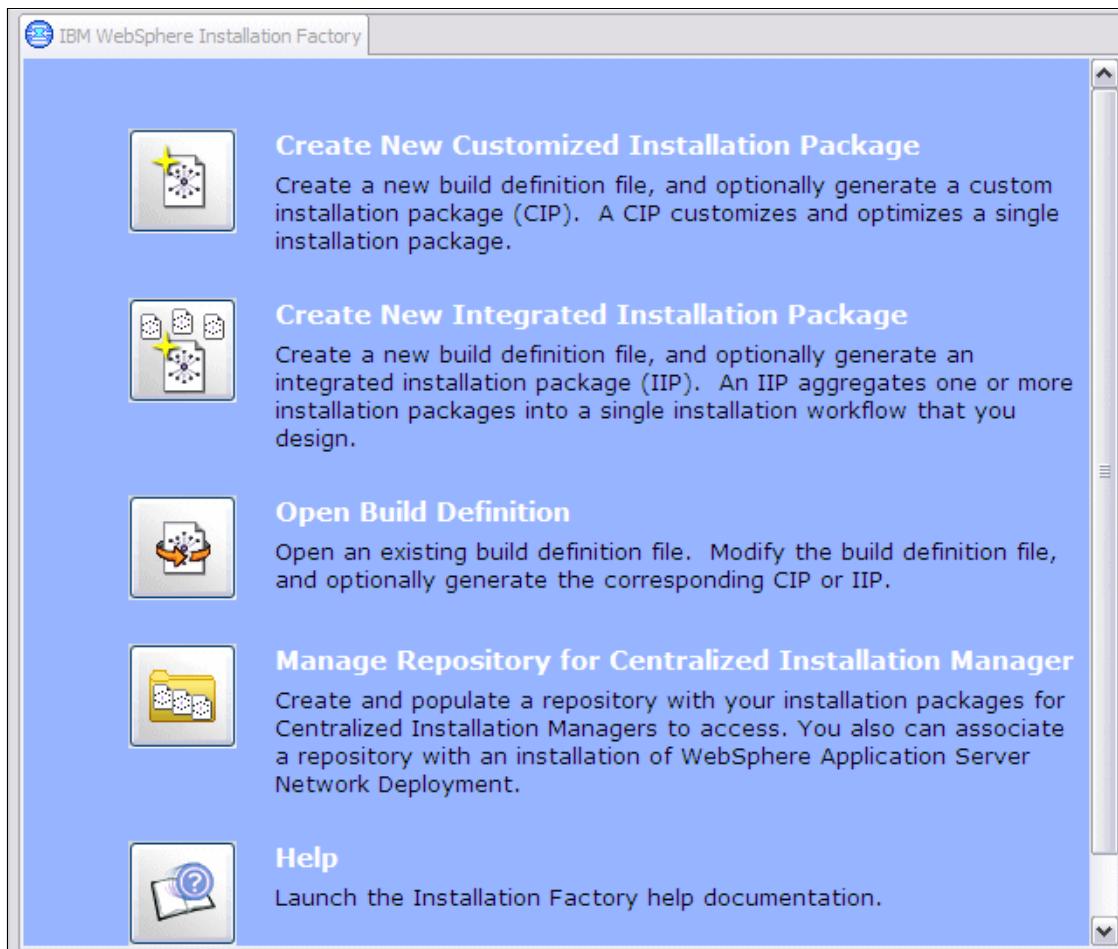


Figure 4-2 Installation Factory

3. Enter the directory path of your application server. The default would be C:\IBM\WebSphere\AppServ. Then click **Next**.

4. In Figure 4-3, enter the directory path to your CIM Repository and the directory that contains the product installation packages, to be loaded into the CIM repository. On this panel you can also calculate the available space in your repository and the space required by the installation package. After information is entered and required space is available, click **Next**.

Repository and Installation Package
Specify the repository and installation package location.

Specify the directory path for the repository. The repository will be associated with the WebSphere Application Server Network Deployment installation provided previously. If the repository will be shared with other installations, it is recommended that you create the repository outside the product installation directory. Specify the directory path to an installation package to add to the repository.

Repository

Directory path to repository:

Available free space:

Installation package

Directory path to installation package:

Installation package size:

Figure 4-3 Repository and Installation Locations

Next you receive a verification panel. Review the information, then select → **Finish**. When the product has been successfully added to your repository, a message will be displayed, successfully added to the repository. Select → **OK**.

4.3.4 Adding maintenance when the deployment manager is connected to the Internet

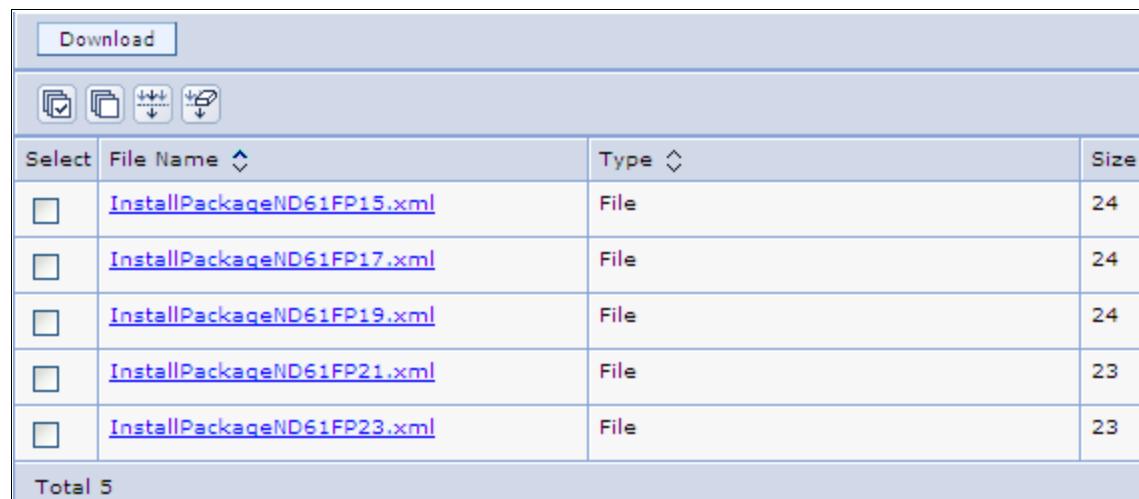
The latest level of the Update Installer is required to apply maintenance to remote targets. Use the following steps to download the latest level to your CIM Repository.

1. **System Administration → Centralized Installation Manager → Installation Packages → Select → Update Installer for WebSphere Application Server.**
2. Select one or more operating systems, then to proceed, select **Download**.
3. Review the summary, and select **Download** to start the download of the Update Installer.
4. On the Installation packages panel, after the download starts, you can monitor the status by selecting the **Refresh** button. If you receive errors, for more detailed information, refer to:
`<install_root>/profiles/<dmgr profile name>/logs/dmge/systemOut.log`

Downloading descriptors and associated binaries

Download additional installation packages and maintenance files to your CIM repository to install on your remote systems.

System Administration → Centralized Installation Manager → Installation Packages → Click add package. On the next panel, select one or more descriptor files from then select **Download to proceed** (Figure 4-4).



Select	File Name	Type	Size
<input type="checkbox"/>	InstallPackageND61FP15.xml	File	24
<input type="checkbox"/>	InstallPackageND61FP17.xml	File	24
<input type="checkbox"/>	InstallPackageND61FP19.xml	File	24
<input type="checkbox"/>	InstallPackageND61FP21.xml	File	23
<input type="checkbox"/>	InstallPackageND61FP23.xml	File	23
Total 5			

Figure 4-4 Descriptor files

You can monitor the progress of the download by selecting the **Download Status** button.

Note: You only need to download descriptors for WebSphere Application Server Network Deployment Version 6.1 fix packs using the method described in this section if you plan to use CIM to install 6.1 fix packs on your Version 6.1 nodes in your cell. The descriptors for WebSphere Application Server Network Deployment Version 7.0 fix packs are installed on the deployment manager machine when you install the particular 7.0 fix pack on your deployment manager using the WebSphere Update Installer.

Downloading the binaries for refresh and fix packs

The CIM supports the installation of Network Deployment Version 6.1 Fix Packs on remote nodes that are within the Network Deployment cell. This configuration is known as a mixed-version cell, where the deployment manager node is at Version 7.0 or higher and the other nodes within the cell are either at the same level as the deployment manager node or at the Version 6.1 level. CIM does not support maintenance levels below Version 6.1.

CIM currently has definitions for Network Deployment Version 6.1 Fix Pack 15 and 17. When newer Network Deployment Version 6.1 Fix Packs become available, CIM will have definitions for those as well. The content of these CIM defined Network Deployment Version 6.1 Fix Packs include the following individual fix packs for the distributed platforms and Windows:

- ▶ WebSphere Application Server fix pack
- ▶ Java Software Developer Kit (SDK) fix pack
- ▶ WebSphere Application Server Feature Pack for Web Services fix pack
- ▶ WebSphere Application Server Feature Pack for EJB 3.0 fix pack

After the descriptor for the required Network Deployment Version 6.1 Fix Pack has been downloaded using the method described here, you then proceed to download the *.pak files for that fix pack to the CIM repository. For Network Deployment Version 7.0 Fix Packs, the descriptor for those are installed on the deployment manager when you install the 7.0 Fix Pack on your deployment manager. You do not have to download the descriptor.

CIM determines whether either of the two Feature Pack fix packs are required and only sends the necessary ones to the target nodes for installation. Because both Network Deployment Version 6.1 Fix Pack 15 and 17 specify that a mandatory Interim Fix, PK53084, must be installed on the target if the Feature Pack for Web Services is installed, CIM also performs a check before allowing the installation of Fix Pack 15 and 17 to proceed.

CIM uses the Update Installer for WebSphere Application Server Version 7.0 to install and uninstall the CIM-defined Network Deployment Version 6.1 Fix Packs.

To download the binary files for a refresh pack, fix pack, or maintenance tool package type, which includes the Update Installer, complete the following steps:

1. To start downloading files, select **System Administration** → **Centralized Installation Manager** → **Installation Packages**, as shown in Figure 4-5. Click the package name in the table.

Installation packages		
Use this page to add or remove installation packages in this cell.		
<input checked="" type="checkbox"/> Preferences		
	<input type="button" value="Add Packages"/> <input type="button" value="Remove Packages"/>	
Select	Package Name	Version
	Maintenance for WebSphere Application Server Network Deployment C:\WebSphere\AppServer\properties\cim\InstallPackageND61Maintenance.xml	6.1
	Maintenance for WebSphere Application Server Network Deployment C:\WebSphere\AppServer\properties\cim\InstallPackageND70Maintenance.xml	7.0
	Update Installer for WebSphere Application Server C:\WebSphere\AppServer\properties\cim\InstallPackageUpdi70X.xml	7.0
	WebSphere Application Server Network Deployment C:\WebSphere\AppServer\properties\cim\InstallPackageND70X.xml	7.0
<input type="checkbox"/>	WebSphere Application Server Network Deployment Fix Pack C:\WebSphere\cimrepos\descriptors\InstallPackageND61FP15.xml	6.1.0.15
<input type="checkbox"/>	WebSphere Application Server Network Deployment Fix Pack C:\WebSphere\cimrepos\descriptors\InstallPackageND61FP17.xml	6.1.0.17
<input type="checkbox"/>	WebSphere Application Server Network Deployment Fix Pack C:\WebSphere\AppServer\properties\cim\InstallPackageND70FP3.xml	7.0.0.3
<input type="checkbox"/>	WebSphere Application Server Network Deployment Fix Pack C:\WebSphere\cimrepos\descriptors\InstallPackageND61FP19.xml	6.1.0.19
<input type="checkbox"/>	WebSphere Application Server Network Deployment Fix Pack C:\WebSphere\cimrepos\descriptors\InstallPackageND61FP21.xml	6.1.0.21

Figure 4-5 Installation Packages

2. Figure 4-6 shows the next panel. Select one or more platforms, then select **Download** to proceed. Select **Download** on the confirmation page to start downloading the binaries.



The screenshot shows a table titled 'Platforms' with two columns: 'Platform' and 'Download Status'. The 'Platform' column lists various operating systems, and the 'Download Status' column indicates whether the download is 'Completed' or 'Absent'. There are also 'Select' checkboxes and icons for sorting and filtering.

Select	Platform	Download Status
<input type="checkbox"/>	AIX 32 bit	Completed
<input type="checkbox"/>	AIX 64 bit	Absent
<input type="checkbox"/>	HP-UX	Absent
<input type="checkbox"/>	HP-UX 64 bit	Absent
<input type="checkbox"/>	Linux for AMD 64 bit	Absent
<input type="checkbox"/>	Linux for Intel	Completed
<input type="checkbox"/>	Linux for i/p Series	Absent
<input type="checkbox"/>	Linux for i/p Series 64 bit	Absent
<input type="checkbox"/>	Linux for zSeries	Absent
<input type="checkbox"/>	Linux for zSeries 64 bit	Absent
<input type="checkbox"/>	Solaris for Sun Sparc 32 bit	Absent
<input type="checkbox"/>	Solaris for Sun Sparc 64 bit	Absent

Figure 4-6 Platforms

- When all required files have been downloaded, the status column will display Complete. If one or more files are missing, the download status column displays an Incomplete status. In this case, you can try to download again. If your status is Incomplete, check for error messages in the **profile_root/logs/dmgr/SystemOut.log** file where **profile_root** is the profile location of the deployment manager.

Complete the following steps to download to download the binary files for an interim fix package type, for WebSphere Application Server Network Deployment Version 6.1 or 7.0:

- To download a specific APAR, select **System Administration** → **Centralized Installation Manager** → **Installation Packages**. Click on the name of the package file in the table, a new page will be displayed.
- Select **Add files** to go to the **Download Files** page.

3. Type the APAR number, then select **Search** (Figure 4-7).
4. Select the APAR from the list provided, then select **Download**, verify the information, and to proceed, click **Download**.

The screenshot shows a web-based interface titled "Installation packages". At the top, there's a search bar with "APAR:" and "PK75887" entered, and a "Search" button. Below the search bar, there are two expandable sections: "Download Options" and "Preferences". Under "Download Options", there is a "Download" button and a set of icons for file operations. A table below lists two files: "7.0.0.1-WS-WAS-IFPK75887.pak" and "readme.txt", both categorized as "File". The table has columns for "Select", "File Name", and "Type". At the bottom of the table, it says "Total 2".

Select	File Name	Type
<input type="checkbox"/>	7.0.0.1-WS-WAS-IFPK75887.pak	File
	readme.txt	File

Figure 4-7 Specific APAR download

4.3.5 When the deployment manager is not connected to the Internet

To use the download functions of your CIM, you must have an Internet connection. If your deployment manager system does not have Internet access, the files will have to be manually downloaded and transferred to your CIM repository.

Before you can copy the downloaded files into the repository, ensure that you have set up the directory structure described in 4.1.4, “Repository directory structure” on page 209.

To obtain the FTP site to manually download the required file, select **Administration console** → **System Administrator** → **Centralized Installation manager** → **Installation Packages**.

The FTP URL format is:

`ftp://ftp.software.ibm.com/software/websphere/appserv/support/cim/cim70
_yyyymmdd`

If the deployment manager does not have Internet access, an error message is displayed that the FTP URL is not known. Write down the FTP URLs, because they will be required on the system that has Internet access:

- ▶ Descriptor files: Select **Add packages**. To determine the location of the FTP server, expand the Download Options. This gives you the FTP URL location used by CIM for downloading the Descriptor files. Only descriptors for WebSphere Application Server Network Deployment Version 6.1 Fix Packs are stored in this FTP location. The descriptors for WebSphere Application Server Network Deployment Version 7.0 Fix Packs are installed on the deployment manager machine when you install the particular 7.0 fix pack on your deployment manager using the WebSphere Update Installer.
- ▶ Update Installer files: From **System Administrator** → **Centralized Installation manager** → **Installation Packages**, click the name **Update Installer for WebSphere Application Server**. Expand download options. This gives you the FTP URL location used by CIM for downloading the Update Installer for the various operating systems.
- ▶ Fix packs: From **System Administrator** → **Centralized Installation manager** → **Installation Packages**, click the name representing the particular fix pack on the table (such as WebSphere Application Server Network Deployment Fix Pack 7.0.0.1), then expand download options. This gives you the FTP URL location used by CIM for downloading cumulative fix packs and individual fixes. Choose the FTP URL for the type of fix you will be downloading.
- ▶ Interim Fixes: From **System Administrator** → **Centralized Installation manager** → **Installation Packages**, click the name **Maintenance for WebSphere Application Server Network Deployment 6.1** or **Maintenance for WebSphere Application Server Network Deployment 7.0**, then click **Add Files**.

With the FTP URL, you can now go to any system with Internet access and download the required files. After you have the files downloaded, copy them to the correct directory structure in the CIM repository.

Another option for obtaining the latest Update Installer, fix packs, and interim fixes would be to set up an FTP gateway on a system that has Internet access. Refer to the IBM InfoCenter for the steps to set up an FTP Gateway, at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.installation.nd.doc/info/ae/ae/tins_cim_files_manual_add.html

4.4 Using CIM to manage your environment

An installation target is a remote system on which selected software packages can be installed or uninstalled using CIM. Also, CIM allows you to edit the configuration of existing installation targets.

By default, all of the systems containing nodes that are defined in the cell are displayed as installation targets. You can add additional installation targets that are located outside of the cell. Other tasks that you can complete to further manage your installation targets include removing installation targets, editing the configuration of installation targets, and installing a Secure Shell (SSH) public key on installation targets.

4.4.1 Adding additional installation targets outside of the cell

Select **System Administrator** → **Centralized Installation Manager** → **Installation Targets** → **Add Installation Target**.

Enter the Host name, User name, Password of the target system that you would like to add, and the Platform Type (Figure 4-8). It is important to specify the domain-qualified hostname in the Host name: field rather than a short hostname. Especially important if you will be installing WebSphere Application Server on the remote target, the value specified in this field will be used in the configuration of the node.

Select **OK**. You should now see the target in the list of target systems.

[Installation targets](#) > New

Use this page to view or change the configuration of an installation target.

Configuration

General Properties

* Host name:

User name:

Password:

* Platform type:

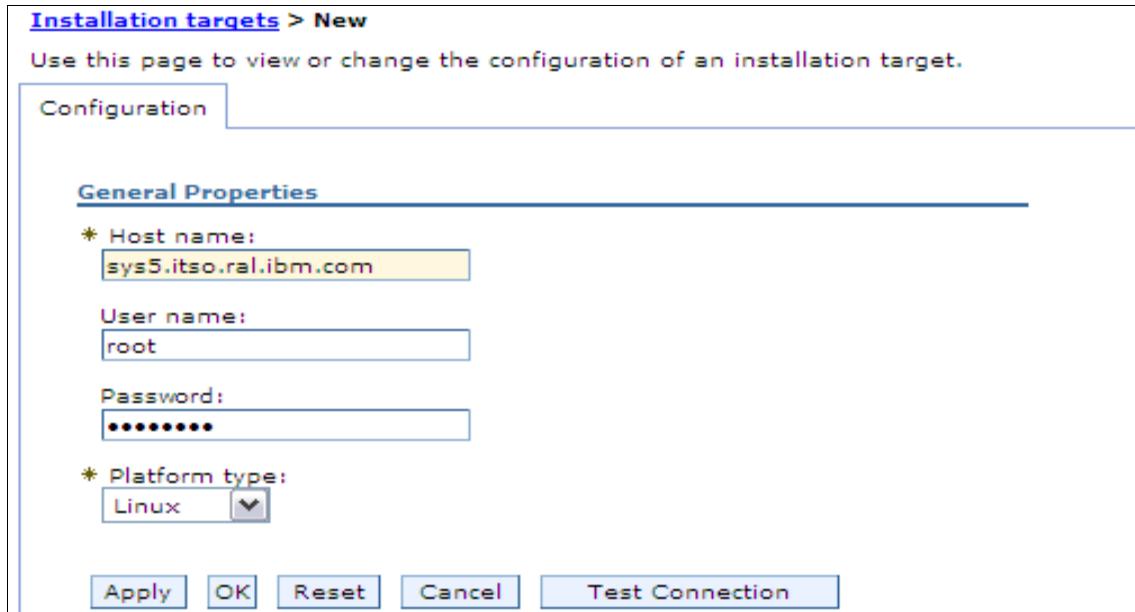


Figure 4-8 Installing Installation Targets

After the target has been added, you can select the **Test** button to test the connection using the Administrator ID and password you provided.

4.4.2 Installing a Secure Shell (SSH) public key

To install a Secure Shell (SSH) public key on specific installation targets, select one or more targets from the table, then select **Install SSH Public Key** (Figure 4-9). Enter userid and password → Click **Next**.

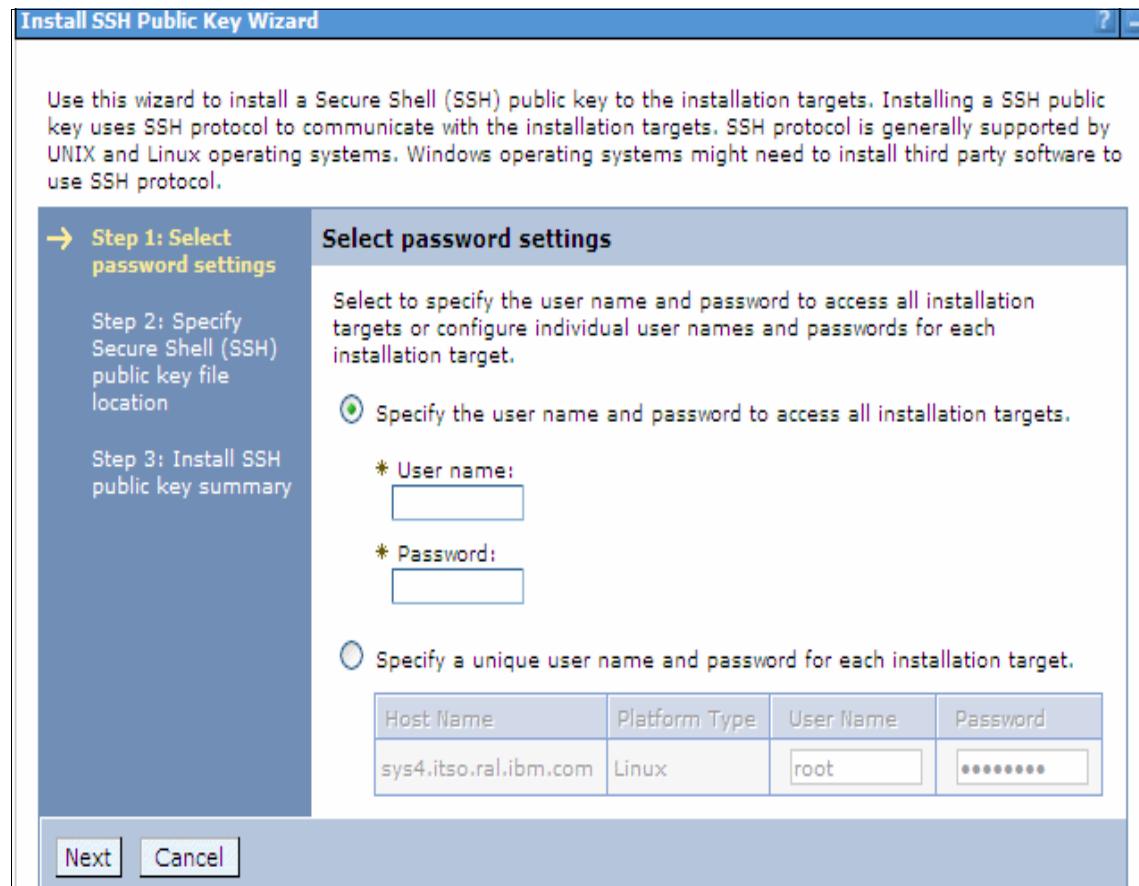


Figure 4-9 Target userid and password

You will be prompted for the specific SSH public key location (Figure 4-10). Enter your file location, then click **Next** to continue.

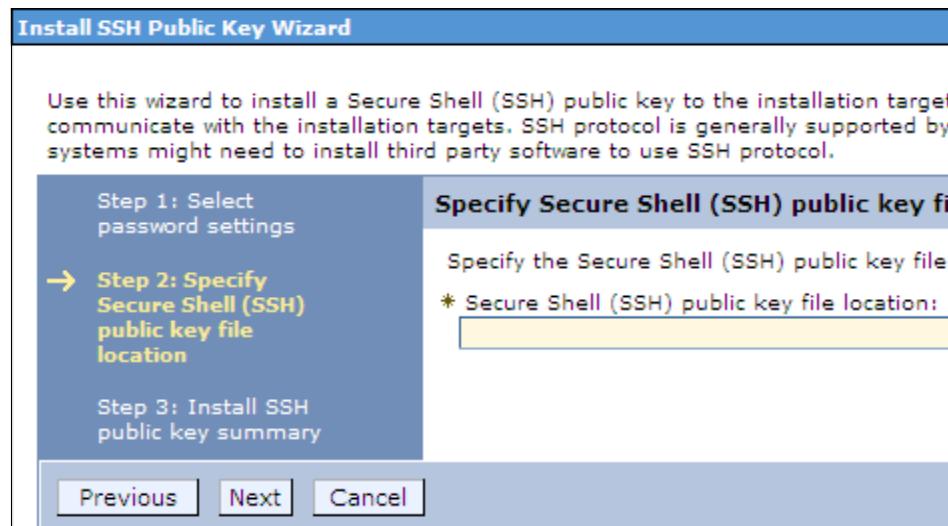


Figure 4-10 SSH Public Key

As a result, the wizard is then launched to complete the SSH public key installation process. For more information about accessing your remote workstations by using the SSH public/private key pair authentication method, refer to “Installing a secure shell public key to access remote targets” on page 207.

4.4.3 Removing installation target systems

To remove existing installation targets, select one or more targets from the table, then select **Remove Installation Target** (Figure 4-11). The confirmation page then lists each selected installation target. Select **Remove** to complete the action, and to return to the Installation targets page.



Figure 4-11 Removing target systems

You will receive a confirmation page where you can confirm the removal or cancel the request.

4.4.4 Installing packages to the target systems

The CIM relies heavily on remote node information maintained locally on the deployment manager node. This remote node information (namely the node-metadata.properties file) for each node is refreshed every time the node agent on the remote node starts and provides CIM with up-to-date information regarding the WebSphere products and versions that are installed on the target nodes.

One example of how the node-metadata.properties information is used by CIM is in the filtering of nodes that might be selected for the installation of an interim fix.

Assume you have downloaded an interim fix for the Feature Pack for Web Services to the CIM repository to be installed on remote node. CIM looks at the information contained within the interim fix and determines that the fix is only applicable for nodes that have the Feature Pack for Web Services Version 6.1.0.9 or higher installed. CIM then checks the node-metadata.properties of all the nodes within the cell to determine which of the remote nodes meet the requirement for this interim fix.

This process allows the cell administrator to see which nodes are potential candidates for this update and then initiate the installation of the interim fix on one or all the candidate nodes. Because of the availability of the node-metadata.properties on the deployment manager node, you could use CIM to perform this filtering without accessing the target nodes. The node agent process that runs on each node ensures that the node-metadata.properties files of the nodes on the deployment manager are kept up-to-date.

For this reason, if you apply maintenance to the node or install new WebSphere products (such as the Feature Pack for Web Services) outside of CIM on the remote node, you must restart the node agent process after the installation to get the deployment manager copy of the node-metadata.properties of the node up to date.

4.4.5 Product installation

To Install a package, select **System Administrator** → **Centralized Installation Manager** → **Available Installations**

Select the package type: **Product Install**, when selecting a product install you will be required to select **Optional features**. You can choose from the following features (Figure 4-12):

- ▶ Install the sample applications for learning and demonstration environments. The samples are not recommended for installation on production environments.
- ▶ Install all the non-English language files for using the administrative console from machines with non-English locales. If you do not select this option, then only the English language pack is installed.
- ▶ Install the non-English language files that support the application server runtime environment such as the wsadmin tool and logging. If you do not select this option, then only the English language pack is installed.

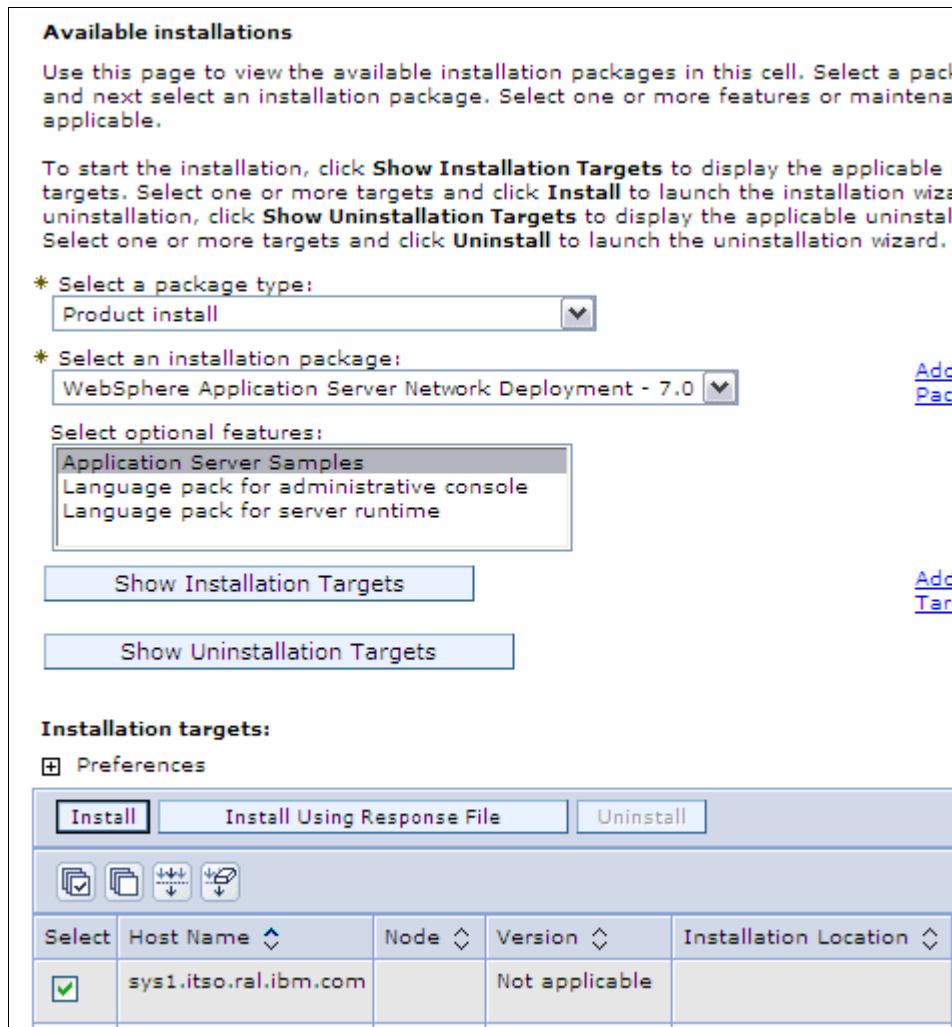


Figure 4-12 Product Install

Follow these steps:

1. Select **Install** → **Accept License Agreement** → **Next**.
2. Select **Authentication method** to access the target. You can choose a user name and password or the use of Secure Shell (SSH) Public/Private key authentication. Depending on the authentication method that you choose, the following panel will correspond. If you choose to authenticate using the user name and password method, you can provide a common user name and password to access all of the installation targets, or you can configure unique user names and passwords for each target.

3. In the next panel, you provide the installation directory and a working directory, then click **Next**.
4. The next two panels give you the option to disable prerequisite checking and to accept limitations to allow installing as a non-root user.
5. By default, CIM will use 64-bit installation binaries on 64-bit operating systems. The next panel allows you to override this. Then click **Next**.
6. Review the Summary panel, then select **Finish** to start the installation. Refer to Figure 4-13.

Go to **System Administrator** → **Centralized Installation Manager** → **Installation Progress** to watch the progress of the installation. When the installation is complete, go to **System Administrator** → **Centralized Installation Manager** → **Installation History** for details of the installation.

Use this wizard to install packages to one or more installation targets.																									
<p>Step 1: Accept software license agreement</p> <p>Step 2: Select authentication method</p> <p>Step 3: Select authentication settings</p> <p>Step 4: Specify location</p> <p>Step 5: Select command parameter 1</p> <p>Step 6: Select command parameter 2</p> <p>Step 7: Select command parameter 3</p> <p>→ Step 8: Summary</p>	<p>Summary</p> <p>The following information is a summary of your selections. Click Previous to change any settings.</p> <p>Summary of installation options</p> <table border="1"> <thead> <tr> <th>Options</th><th>Values</th></tr> </thead> <tbody> <tr> <td>Operation</td><td>Install</td></tr> <tr> <td>Package</td><td>WebSphere Application Server Network</td></tr> <tr> <td>Feature</td><td>Application Server Samples</td></tr> <tr> <td>Software license agreement</td><td>I accept the terms in the license agreement.</td></tr> <tr> <td>Authentication method</td><td>Use user name and password.</td></tr> <tr> <td>Authentication settings</td><td>Specify a unique user name and password for each installation target.</td></tr> <tr> <td>Command parameter 1</td><td>Disable operating system prerequisite checks: true</td></tr> <tr> <td>Command parameter 2</td><td>Accepts the limitation to allow installing as a non-root user: true</td></tr> <tr> <td>Command parameter 3</td><td>Use 32-bit installation binaries on 64-bit systems: false</td></tr> <tr> <td>Attention</td><td>A managed WebSphere Application Server node name and a profile name of "Customer" will be created on each of the remote targets in the current deployment manager cell. After deployment completes successfully, use the deployment manager to create a server or a cluster of servers with the same names.</td></tr> <tr> <td>Selected targets</td><td>testmsw</td></tr> </tbody> </table>	Options	Values	Operation	Install	Package	WebSphere Application Server Network	Feature	Application Server Samples	Software license agreement	I accept the terms in the license agreement.	Authentication method	Use user name and password.	Authentication settings	Specify a unique user name and password for each installation target.	Command parameter 1	Disable operating system prerequisite checks: true	Command parameter 2	Accepts the limitation to allow installing as a non-root user: true	Command parameter 3	Use 32-bit installation binaries on 64-bit systems: false	Attention	A managed WebSphere Application Server node name and a profile name of "Customer" will be created on each of the remote targets in the current deployment manager cell. After deployment completes successfully, use the deployment manager to create a server or a cluster of servers with the same names.	Selected targets	testmsw
Options	Values																								
Operation	Install																								
Package	WebSphere Application Server Network																								
Feature	Application Server Samples																								
Software license agreement	I accept the terms in the license agreement.																								
Authentication method	Use user name and password.																								
Authentication settings	Specify a unique user name and password for each installation target.																								
Command parameter 1	Disable operating system prerequisite checks: true																								
Command parameter 2	Accepts the limitation to allow installing as a non-root user: true																								
Command parameter 3	Use 32-bit installation binaries on 64-bit systems: false																								
Attention	A managed WebSphere Application Server node name and a profile name of "Customer" will be created on each of the remote targets in the current deployment manager cell. After deployment completes successfully, use the deployment manager to create a server or a cluster of servers with the same names.																								
Selected targets	testmsw																								

Figure 4-13 Summary of Install

4.4.6 Installing maintenance to target systems

Before you can install a Version 7.0 fix pack on target systems, you must install it locally first on the deployment manager host using the Update Installer for WebSphere Software. CIM cannot be used to install maintenance to the Deployment Manager. All node agents in the cell must be running on the target systems. If the node agents are not running, it is up to the administrator to make sure that all server processes have been stopped.

Note: You must also download and add the latest version of the Update Installer for WebSphere Application Server v7.0 to the CIM repository for the required platforms before attempting to install the fix pack or interim fixes. Notice that there is no need to explicitly install the Update Installer on the targets first before initiating the installation of the fix packs or interim fixes, because CIM will automatically install the latest version of the Update Installer on the target if needed.

Attention: Fix packs that include updates to the Software Development Kit (SDK) might overwrite unrestricted policy files. Back up unrestricted policy files before you apply a fix pack and reapply these files after the fix pack is applied.

For information about installing CIM-defined Network Deployment Version 6.1 Fix Packs, refer to the IBM Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.installation.nd.doc/info/ae/ae/tins_cim_files_manual_add.html

Using CIM to install refresh or fix packs

Follow these steps:

1. Select **System Administrator** → **Centralized Installation Manager** → **Available Installations**. For package type, select refresh pack, fix pack, or maintenance tool.
2. Next, from the drop-down list of available installation packages, choose the installation package that contains the refresh pack or fix pack that you want to install on your remote systems. These are the packages that you previously downloaded to your CIM repository.
3. Select **Show installation targets** to get a list of target systems available for install. Select your target systems. To continue, select **Install** (Figure 4-14).

Available installations

Available installations

Use this page to view the available installation packages in this cell. Select a package type first, and then select an installation package. Select one or more features or maintenance packs if applicable.

To start the installation, click **Show Installation Targets** to display the applicable installation targets for this package, select one or more targets and click **Install** to launch the installation wizard. To start the uninstallation, click **Show Uninstallation Targets** to display the applicable uninstallation targets. Select one or more targets and click **Uninstall** to launch the uninstallation wizard.

* Select a package type:
Refresh pack, fix pack, or maintenance tool

* Select an installation package:
WebSphere Application Server Network Deployment Fix Pack - 7.0.0.3

Add or Remove
Add Installation

Show Installation Targets Show Uninstallation Targets

Installation targets:

Preferences

Install Install Using Response File Uninstall

Select	Host Name	Node	Version	Installation Location	Platform	Processor
<input type="checkbox"/>	sys1.itso.ral.ibm.com	sys1Node01	ND 7.0.0.1	C:/WebSphere/AppServer	Windows	x86
<input checked="" type="checkbox"/>	sys3.itso.ral.ibm.com	Nodesys3	ND 7.0.0.1	C:/WebSphere/AppServer	Windows	x86
<input checked="" type="checkbox"/>	sys4.itso.ral.ibm.com	Nodesys4	ND 7.0.0.1	C:/WebSphere/AppServer	Windows	x86

Figure 4-14 Maintenance

4. The next panel is the license agreement. Review the agreement, select **Agree**, then to proceed, click **Next**.
5. On the next panel, you can specify the authentication method you want to use, as well as your user name and password or Secure Shell (SSH). Choose your method, then click **Next** to proceed.
6. The next panel displayed depends on the type of authentication you choose. You will receive a panel prompting to enter a userid and password or a panel prompting the location of the SSH private key file and keystore password.
7. Next verify the installation and the working location of the installation targets. The installation location is the remote location of each installation target in which the package is to be installed. The working location specifies the directory on the remote target where the files are sent before the package is installed in the specified location. Make sure that you have enough disk space in both the installation location and the working location. Click **Next** to continue.
8. A summary panel will be displayed. Review the information entered, then to start the installation, select **Finish** (Figure 4-15).

Note: Any interim fixes that you previously installed on the remote targets are uninstalled by the Update Installer prior to installing the refresh pack or fix pack. If the refresh pack or fix pack does not include the official fixes that were included in the removed interim fixes, you must reinstall the interim fixes after you install the refresh pack or fix pack.

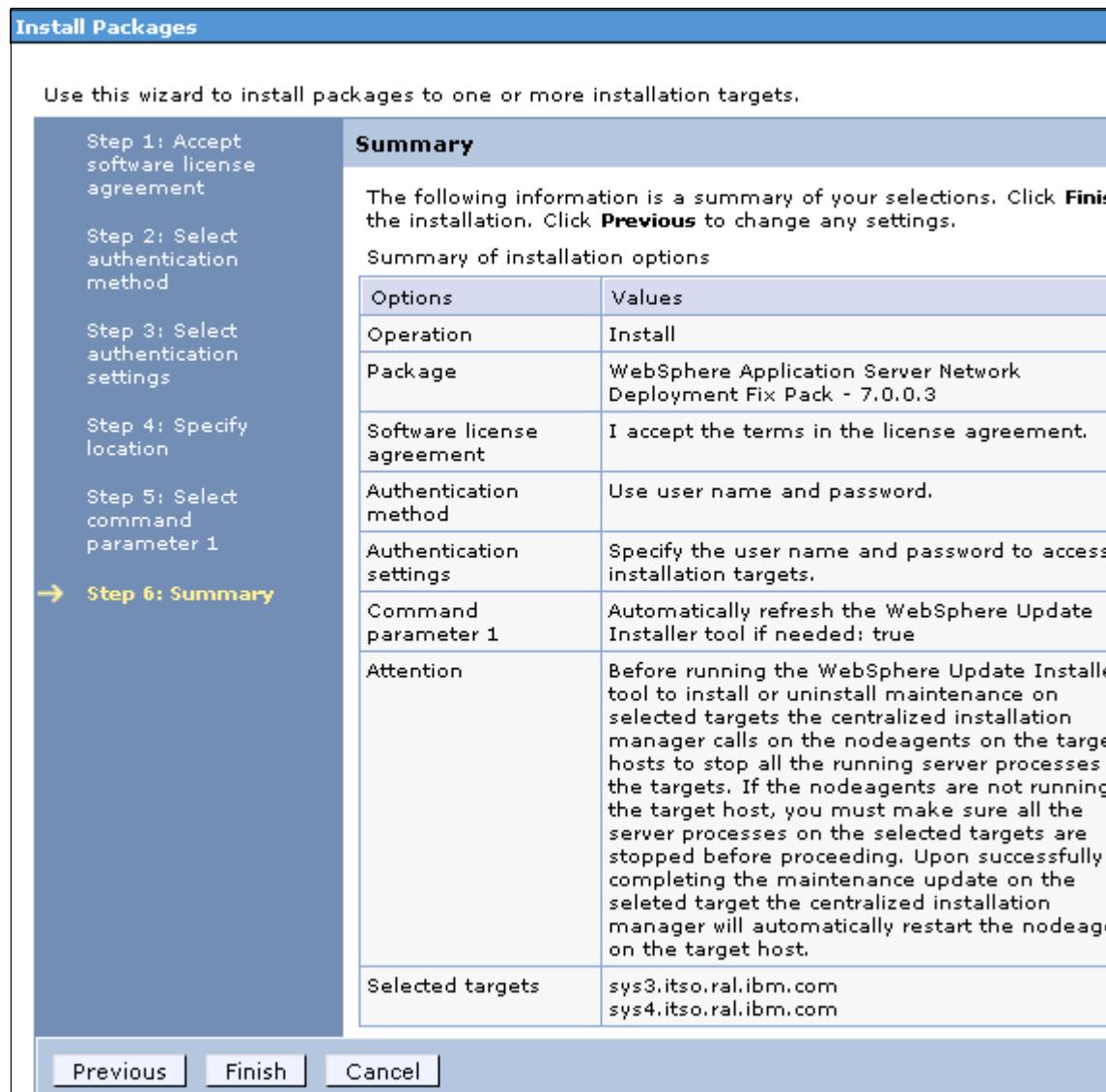


Figure 4-15 Maintenance Summary

You can go to **System Administrator** → **Centralized Installation Manager** → **Installation Progress** to check the progress of the installation. When complete, go to **System Administrator** → **Centralized Installation Manager** → **Installation History** for details of the installation.

Using CIM to install interim fixes

Before you can install an interim fix, you must download IBM Update Installer for WebSphere Application Server Version 7.0 and the binary files for one or more interim fixes. You do not need to install the Update Installer after you download. CIM automatically installs Update Installer before installing any refresh packs, fix packs, or interim fixes if the target does not have the Update Installer already installed.

The following steps show how to install the recommended interim fixes for WebSphere Application Server Network Deployment Version 6.1 or 7.0:

1. Select **System Administrator** → **Centralized Installation Manager** → **Available Installations**. On the drop menu under Select a package type, select **Interim fix**.
2. On the drop-down menu under Select an Installation package, select either **Maintenance for WebSphere Application Server Network Deployment 7.0 or 6.1**. If the interim fixes have been previously downloaded to the CIM repository, they will be displayed under Select one or more maintenance packs. Select the maintenance pack you want to install on your target systems, as displayed in Figure 4-16.

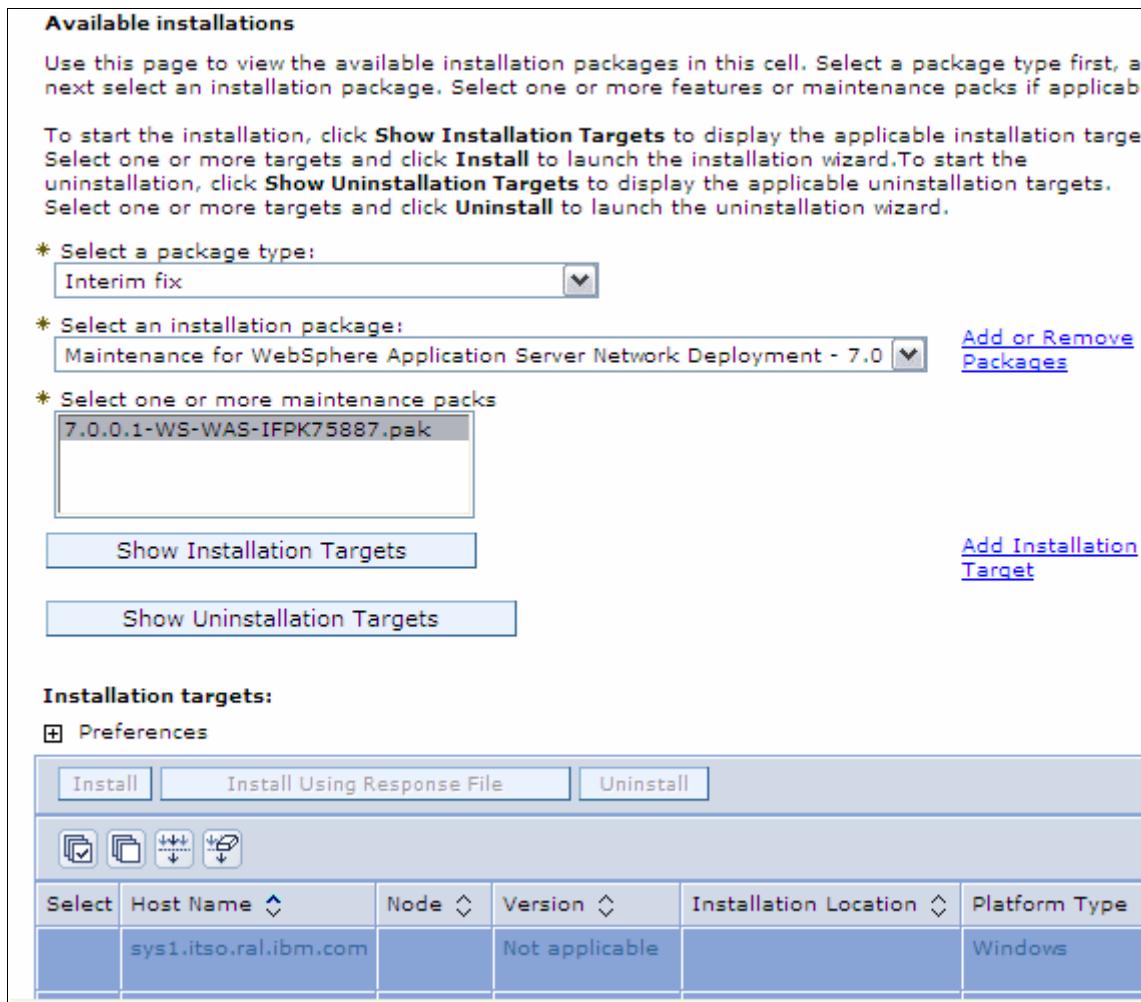


Figure 4-16 Interim fix

3. Click the **Show Installation Targets** button. A list of your target systems will be displayed. From this list, select your targets, then select **Install**.
4. Click **View License Agreement** to read the agreement and accept the terms. Click **Next** to continue.
5. The next panel is the authentication method you'd like to use, user name and password or Secure Shell (SSH). Choose your method, then click **Next** to proceed.

6. The next panel displayed depends on the type of authentication you choose. You will receive a panel prompting to enter a userid and password or a panel prompting the location of the SSH private key file and keystore password.
7. Next, verify the installation and the working location of the installation targets. The installation location is the remote location of each installation target in which the package is to be installed. The working location specifies the directory on the remote target where the files are sent before the package is installed in the specified location. Make sure you have enough disk space in both the installation location and the working location. Click **Next** to continue.
8. Verify the Summary Panel and select **Finish** to start the installation request. To check the status of your request, select **Installations in Progress** on the administrative console. After the installation is completed, click Installation history in the administrative console to review the log files for each of the installation requests that you submit. From the Installation History panel, the administrator can click **View Details** to display a panel with additional details on the results. Links to logs on the remote targets are included. However, those logs can be moved, replaced, or deleted by other users or administrator, if they are not viewed immediately after an installation operation.

If you attempt to install an interim fix without having a copy of the IBM Update Installer for WebSphere Software in your CIM repository, you will receive the following message:

The installation binary files required for the `install_package_name` or its dependent package Update Installer for WebSphere Application Server for `workstation_platform` do not exist.

4.4.7 Uninstalling packages

CIM can be used to uninstall previously installed packages on your installation targets. The tasks available for uninstall will depend on the environment you installed:

1. Select **System Administrator** → **Centralized Installation Manager** → **Available Installations**. On this panel select the Package type you would like to uninstall. then select the Installation package to be uninstalled. Select **Show Uninstallation Targets** → **Target system** → **Uninstall**.
2. You will be prompted for an authentication method. Enter either userid and password or location of SSH key file. Next, verify the Installation target directory. Review the summary page (Figure 4-17), then select **Finish** to proceed with the uninstall.

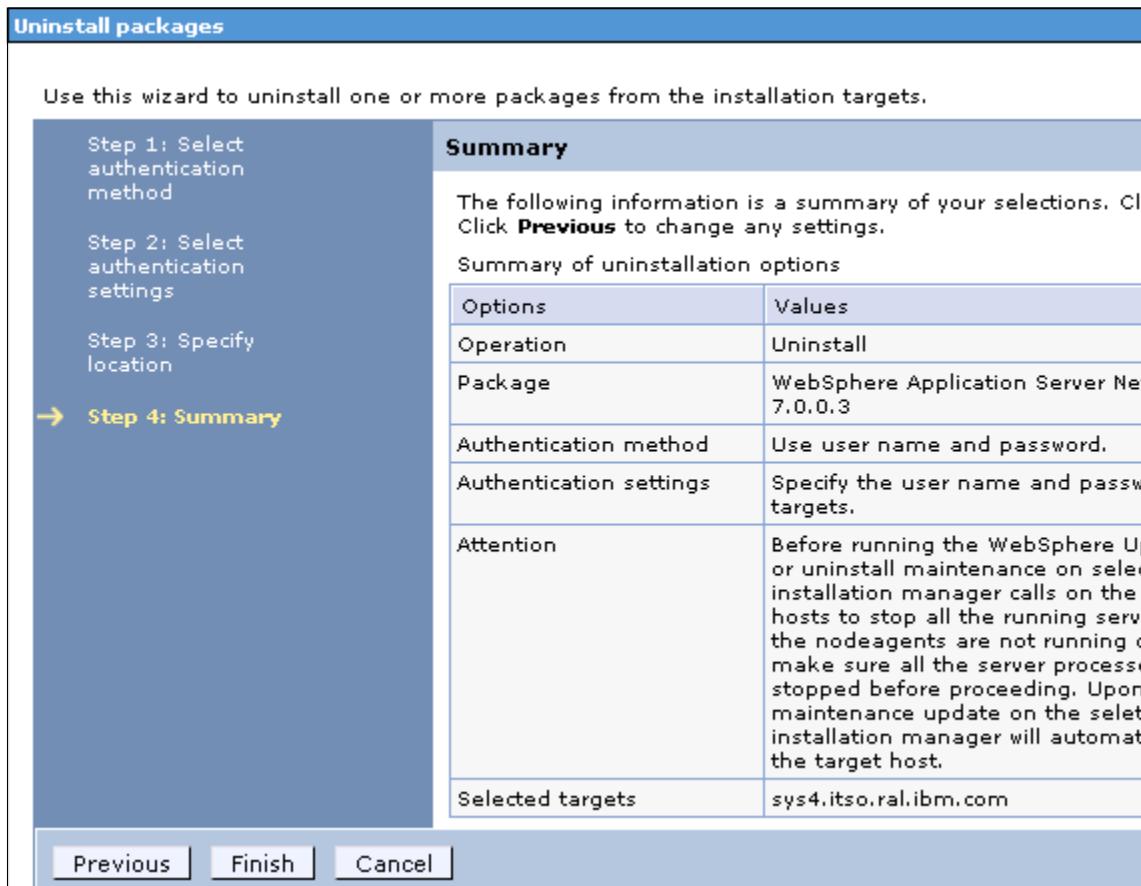


Figure 4-17 Uninstall Summary

Refer to the installation history for the results of the uninstall.

4.4.8 CIM AdminTask Commands

You can use the CIM AdminTask commands with Jacl or Jython scripting language. These commands and parameters can be used to install, uninstall, and manage various software packages and maintenance files in your CIM environment. Here is a list of AdminTask commands available for use:

- ▶ installWASExtension
- ▶ installSoftware
- ▶ installWithResponseFile
- ▶ installMaintenance
- ▶ listPackagesForInstall
- ▶ listFeaturesForInstall
- ▶ showPackageInfo
- ▶ showLicenseAgreement
- ▶ getManagedNodesOnHostByInstallLoc
- ▶ listManagedNodesOnHost
- ▶ testConnectionToHost
- ▶ testConnectionToHostUsingSSHKey
- ▶ installSSHPublicKeyOnHost
- ▶ listKeyInstallationRecords
- ▶ updateKeyInstallationRecords
- ▶ listPendingRequests
- ▶ listInProgressRequests
- ▶ listRequestsForTarget
- ▶ showLatestInstallStatus
- ▶ showLatestUninstallStatus
- ▶ uninstallSoftware
- ▶ uninstallMaintenance

Refer to the Information Center for detailed information about these commands.

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



Administration consoles and commands

WebSphere application server properties are stored in the configuration repository as XML files. It is not a good idea to manually edit any of the configuration files because this bypasses validation of any changes and could lead to synchronization-related problems. Rather, WebSphere application server provides administrative tools that help you administer the environment. These tools manage modifications to the files in the repository.

In this chapter we introduce the administrative consoles and command line administration. Scripting is discussed Chapter 8, “Administration with scripting” on page 439

We cover the following topics:

- ▶ “Introducing the WebSphere administrative consoles” on page 240
- ▶ “Securing the console” on page 266
- ▶ “Job manager console” on page 272
- ▶ “Using command line tools” on page 285

5.1 Introducing the WebSphere administrative consoles

The WebSphere administrative consoles are graphical, Web-based tools that you use to configure and manage the resources within the scope of the console. With the introduction of the flexible management topologies, there are multiple administrative consoles available in a WebSphere solution:

- ▶ Administrative console hosted by an application server or deployment manager:

This console is used to manage an entire WebSphere cell. It supports the full range of product administrative activities, such as creating and managing resources and applications, viewing product messages, and so on.

In a stand-alone server environment, the administrative console is located on the application server and can be used to configure and manage the resources of that server only.

In a distributed server environment, the administrative console is located in the deployment manager server, dmgr. In this case, the administrative console provides centralized administration of multiple nodes. Configuration changes are made to the master repository and pushed to the local repositories on the nodes by the deployment manager.

- ▶ Administrative agent console:

An administrative agent hosts the administrative console for application server nodes that are registered to it.

When you access the URL for the console, you can select the node type to manage. After your selection is made, you will be directed to the appropriate console where you can log in:

- Administrative console for the administrative agent:

This console allows you to manage the administrative agent, including security settings, and registering nodes that it controls with the job manager.

- Console for an application server:

This console is the administrative console for the application server.

- ▶ Job manager administrative console (referred to as the job manager console):

The job manager console provides the interface to manage the job manager itself, including security settings and mail resources. Its primary function, though, is to allow you to submit jobs for processing on the nodes that are registered to it.

5.1.1 Starting and accessing the consoles

The way that you access the administrative console is the same whether you have a stand-alone server environment or a distributed server environment. However, the location and how you start the necessary processes will vary.

Finding the URL for the console:

Each server process that hosts the administrative console has two admin ports that are used to access the console. These ports are referred to as:

- ▶ WC_adminhost
- ▶ WC_adminhost_secure (for SSL communication)

These ports are assigned at profile creation. If you do not know what the port number is for the console, you can look in the following location:

profile_home/properties/portdef.props

You can always use the following URL to access the console:

`http://<hostname>:WC_adminhost/ibm/console`

If administrative security is enabled, you will automatically be redirected to the secure port.

Administrative console in a stand-alone server environment

In a single application server installation, the console is hosted on the application server, so you must start the server in order to reach the console.

To access the administrative console, do the following steps:

1. Make sure that application server, server1, is running by using this command:

`serverStatus.sh -all`

2. If the status of server1 is not STARTED, start it with the following command:

`startServer.sh server_name`

3. Open a Web browser to the URL of the administrative console. For example:

`http://<hostname>:9060/ibm/console`

<hostname> is your host name for the machine running the application server.

Administrative console in a distributed environment

If you are working with a deployment manager and its managed nodes, the console is hosted on the deployment manager. You must start it in order to use the console. To access the administrative console, do the following steps:

1. Make sure that deployment manager, dmgr, is running by using this command:

```
serverStatus.sh -all
```

2. If the dmgr status is not STARTED, start it with the following command:

```
startManager.sh
```

3. Open a Web browser to the URL of the administrative console. For example:

```
http://<hostname>:9060/admin
```

<hostname> is your host name for the machine running the deployment manager.

Job manager console

To access the job manager administrative console, do the following steps:

1. Make sure that job manager process (jobmgr) is running by using this command:

```
serverStatus.sh -all
```

2. If the status of jobmgr is not STARTED, start it with the following command:

```
startServer.sh jobmgr
```

3. Open a Web browser to the URL of the administrative console. For example:

```
http://<hostname>:9960/ibm/console
```

Administrative agent console

1. Make sure that administrative agent process (adminagent) is running by using this command:

```
serverStatus.sh -all
```

2. If the status of adminagent process is not STARTED, start it with the following command:

```
startServer.sh adminagent
```

3. Open a Web browser to the URL of the administrative console. For example:

`http://<hostname>:9060/ibm/console`

If you have nodes registered with the administrative agent, you will be prompted to select which node to you would like to administer which includes the administrative agent.

4. Log in to the selected console.

5.1.2 Logging in to a console

The user ID specified during login is used to track configuration changes made by the user. This allows you to recover from unsaved session changes made under the same user ID, for example, when a session times out or the user closes the Web browser without saving. The user ID for login depends on whether WebSphere administrative security is enabled.

Note: You cannot logon to two instances of administrative consoles that are running on the same machine from a single browser type. For example, if you use Firefox to log in to the deployment manager administrative console, you cannot also login to a job manager running on the same machine.

There is a limitation that cookies are unique per domain rather than a combination of domain and port. Therefore, the cookies that control the session and authentication data in the first browser tab or window get overwritten when logging into the other console in a new browser tab or window. However, you should be able to log into two consoles simultaneously from two completely different browsers for example, Firefox and Internet Explorer.

- ▶ If WebSphere administrative security is not enabled:

You can enter any user ID, valid or not, to log in to the administrative console. The user ID is used to track changes to the configuration, but is not authenticated. You can also simply leave the User ID field blank and click the **Log In** button.

Note: Logging in without an ID is not a good idea if you have multiple administrators.

- ▶ If WebSphere administrative security is enabled:
You must enter a valid user ID and password that has been assigned an administrative security role.
If you enter a user ID that is already in session, you will receive the message Another user is currently logged in with the same User ID and you will be prompted to do one of the following actions:
 - Force the existing user ID out of session. You will be allowed to recover changes that were made in the other user's session.
 - Specify a different user ID.

Note: You can also get this message if a previous session ended without a logout. For example, if the user closed a Web browser during a session and did not log out first or if the session timed out.

Recovering from an interrupted session

Until you save the configuration changes you make during a session, the changes do not become effective. If a session is closed without saving the configuration changes made during the session, these changes are remembered and you are given the chance to pick up where you left off.

When unsaved changes for the user ID exist during login, you are prompted to do one of the following actions:

- ▶ Work with the master configuration:
Selecting this option specifies that you want to use the last saved administrative configuration. Changes made to the user's session since the last saving of the administrative configuration will be lost.
- ▶ Recover changes made in a prior session:
Selecting this option specifies that you want to use the same administrative configuration last used for the user's session. It recovers all changes made by the user since the last saving of the administrative configuration for the user's session.

As you work with the configuration, the original configuration file and the new configuration file are stored in a folder at:

`<profile_home>/wstemp`

After you save the changes, these files are removed from the wstemp folder.

5.1.3 Changing the administrative console session timeout

You might want to change the session timeout for the administrative console application. The session timeout is the time it takes for the console session to time out after a period of idleness. The default is 15 minutes. To change the session timeout value, see the following page in the Information Center:

- ▶ Changing the console session expiration:

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

The console application name is isclite. This is true for all administrative consoles (job manager, administrative agent, deployment manager, or standalone application server console).

5.1.4 The graphical interface

The WebSphere administrative consoles have the same layout pattern. In each console, you can find the following main areas:

- ▶ Banner
- ▶ Navigation tree
- ▶ Workspace, including the messages and help display areas.

Each area can be resized as desired. The difference in the console types will be in the Navigation tree. The options that you find there will vary depending on the console type.

Figure 5-1 uses the administrative console hosted on a deployment manager to illustrate the console layout.

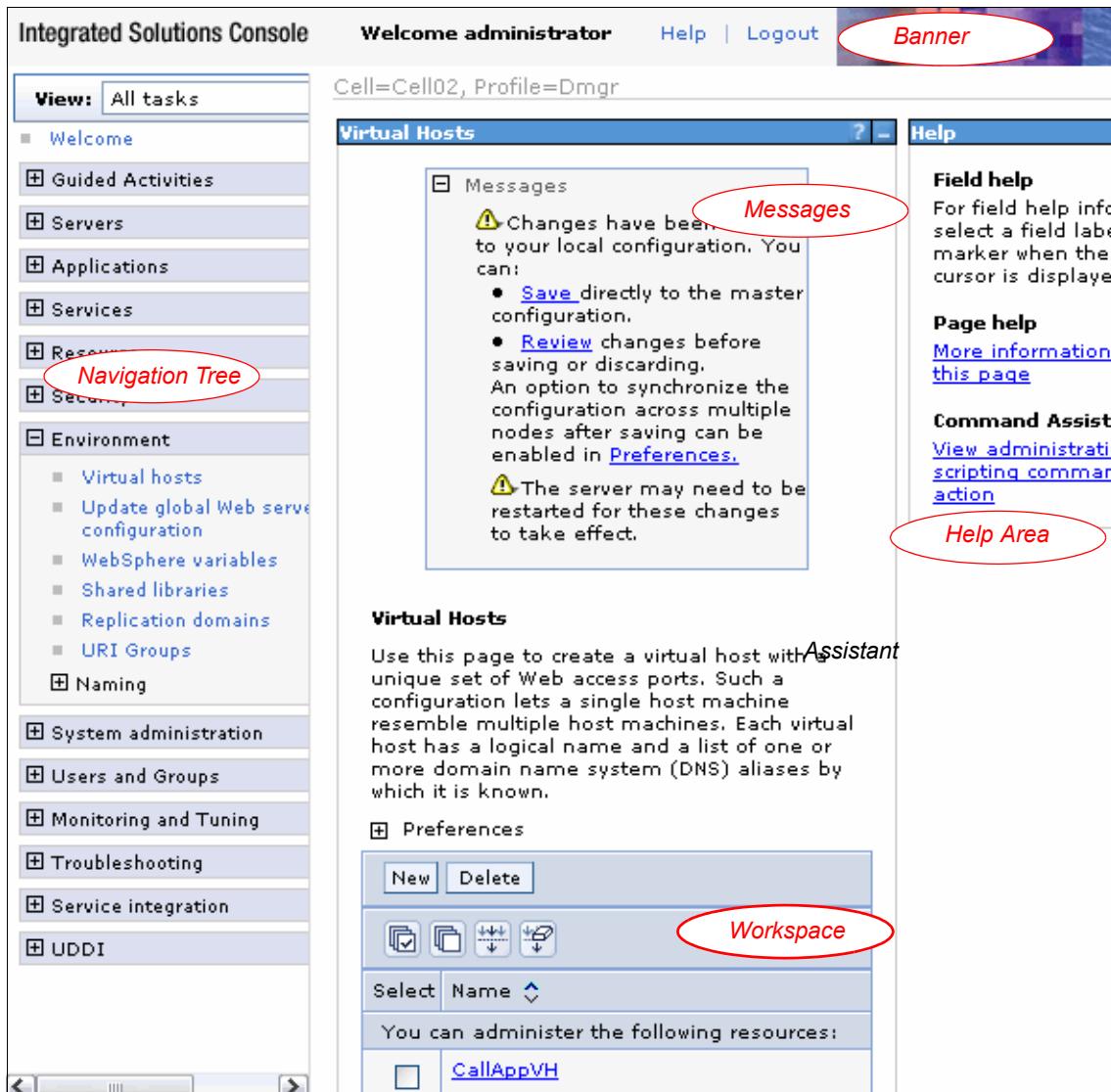


Figure 5-1 The administrative console graphical interface

Banner

The banner is the horizontal bar near the top of the console. The banner provides the following actions:

- ▶ Logout logs you out of the administrative console session and displays the Login page. If you have changed the administrative configuration since last saving the configuration to the master repository, the Save page displays before returning you to the Login page. Click **Save** to save the changes, **Discard** to return to the administrative console, or **Logout** to exit the session without saving changes.
- ▶ **Help** opens a new Web browser with detailed online help for the administrative console. This is not part of the Information Center.

Console identity (new in V7)

The banner can be customized to show a unique identifier for the console. This can be helpful in cases where administrators log on to multiple administrative consoles. Glancing at the banner lets you know which system you are logged on to. You can add a Console ID from the administrative console (Figure 5-2).

To customize the banner, navigate to **System environment** → **Console Identity**. Select **Custom** and enter the identity string. Save the changes, and log out of the console, then back in. This console identity will be displayed to all users that log in to that console application.

In an administrative agent configuration, the changes are applied to the administrative agent and all of its registered application servers, regardless of where the changes were actually saved.

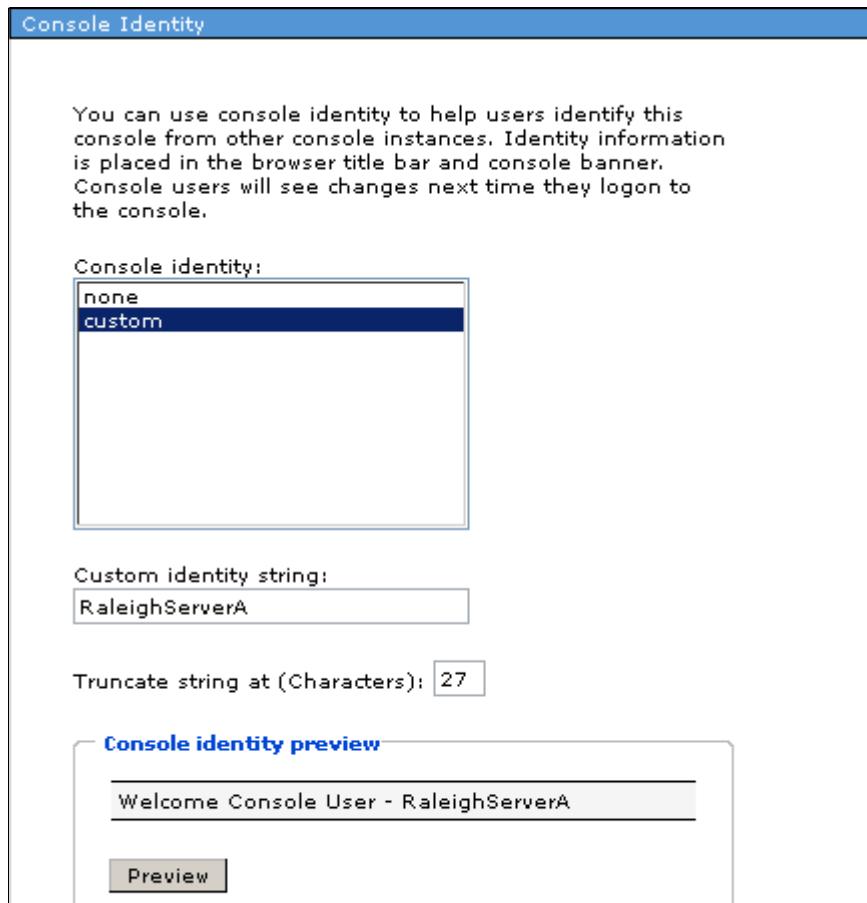


Figure 5-2 Changing the console identity

After you log back in, you will see the new console identity in the banner (Figure 5-3).



Figure 5-3 Banner with a customized console identity

Navigation tree

The navigation tree on the left side of the console offers links for you to view, select, and manage components.

Clicking a + beside a tree folder or item expands the tree for the folder or item. Clicking a - collapses the tree for the folder or item. Double-clicking an item toggles its state between expanded and collapsed.

The content displayed on the right side of the console, the *workspace*, depends on the folder or item selected in the tree view.

Guided activities

The navigation tree includes a category called “Guided Activities”. This section contains step-by-step assistance for performing some common tasks. These activities can be accomplished by performing each task separately, but using the Guided Activities option provides additional assistance.

Workspace

The workspace, on the right side of the console in Figure 5-1 on page 246, allows you to work with your administrative configuration after selecting an item from the console navigation tree.

When you click a folder in the tree view, the workspace lists information about instances of that folder type, the collection page. For example, selecting **Servers** → **Server Types** → **WebSphere application servers** shows all the application servers configured in this cell. Selecting an item, an application server in this example, displays the detail page for that item. The detail page can contain multiple tabs. For example, you might have a Runtime tab for displaying the runtime status of the item, and a Configuration tab for viewing and changing the configuration of the displayed item.

Messages

When you perform administrative actions, messages are shown at the top of the workspace to display the progress and results. These messages are limited in nature so if an action fails, review the JVM process logs for more detailed information.

When configuration changes have been made, the message area will contain links that you can click to review or save the changes.

Breadcrumb trail

As you navigate into multiple levels of a configuration page, a breadcrumb trail is displayed at the top of the workspace. It indicates how you reached the current page and provides links that allow you to go back to previous pages easily without starting the navigation trail over (Figure 5-4).

The screenshot shows a user interface for managing installed applications. At the top, there's a header bar with the title 'Application servers'. Below it, a breadcrumb trail is displayed: 'Application servers > server40a1 > Installed applications'. This breadcrumb trail is highlighted with a red oval. The main content area has a sub-header 'Installed applications' with the instruction 'Use this page to manage installed applications. A single application can be deployed'. There's a 'Preferences' button followed by a toolbar with icons for copy, paste, move up, move down, and delete. A search bar allows filtering by 'Name' and 'Application Status'. Below this, a section titled 'You can administer the following resources:' lists two items: 'ClassloaderExample' and 'ClassloaderExampleV2', each with a checkbox and a help icon. A summary at the bottom indicates 'Total 2' items.

Figure 5-4 Breadcrumb trail

Help area

As you are working in the administrative console, help is available in multiple ways. As you hover the mouse over a field, help text will be displayed for that field. In addition, most pages will have a **More information about this page** link in the Help area. Clicking the link will open the online help in a separate browser. And finally, many pages will have a **View administrative scripting command for last action** link. Clicking this link will display an equivalent scripting command for the action you just performed.

Setting console preferences

The look of the administrative console can be altered by setting console preferences. The preference you see will vary slightly depending on the console type. For example, the preference to synchronize changes with nodes is only applicable to a console on a deployment manager. See Figure 5-5.

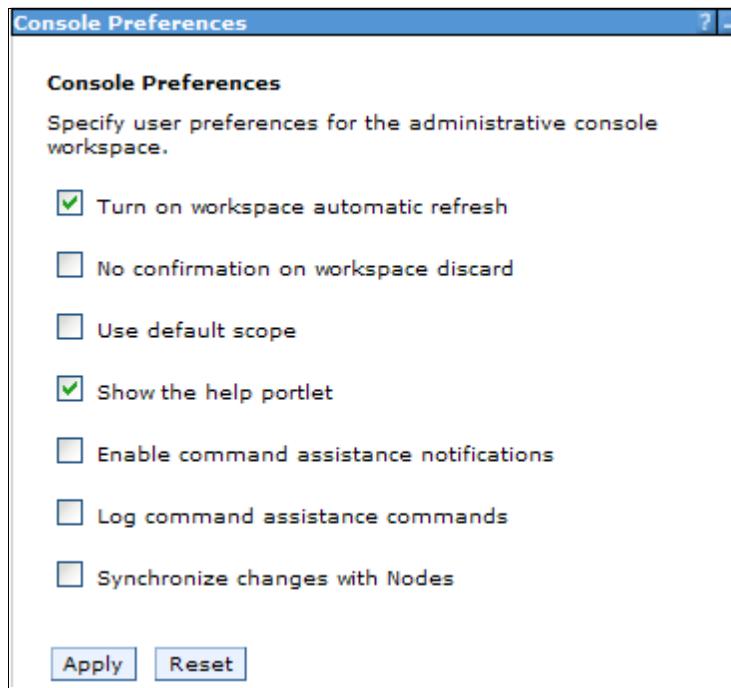


Figure 5-5 Administrative console preferences

To set console preferences, select **System Administration Console Preferences** in the navigation tree. You have the following options:

- ▶ **Turn on WorkSpace Auto-Refresh** specifies that the view automatically refreshes after a configuration change. If it is not selected, you must re-access the page to see the changes.
- ▶ **No Confirmation on Workspace Discard** specifies that a confirmation window be displayed if you elect to discard the workspace. For example, if you have unsaved changes and log out of the console, you will be asked whether you want to save or discard the changes. If this option is not selected and you elect to discard your changes, you will be asked to confirm the discard action.
- ▶ **Use default scope** (administrative console node) sets the default scope to the node of the administration console. If you do not enable this setting, the default is all scopes.
- ▶ **Show the help portlet** displays the help portlet at right top.

- ▶ **Enable command assistance notifications** allows you to send JMX notifications that contain command data. These notifications can be monitored in a Rational Application Developer workspace, providing assistance in creating scripts.
 - ▶ **Log command assistance commands** specifies whether to log all the command assistance wsadmin data for the current user.

When you select this option, script commands matching actions you take in the console are logged to the following location:
profile_root/logs/AssistanceJythonCommands_<user_name>.log
 - ▶ **Synchronize changes with Nodes** synchronizes changes that are saved to the deployment manager profile with all the nodes that are running.
- Click the boxes to select which preferences you want to enable and click **Apply**.

5.1.5 Finding an item in the console

To work with items in the console, do the following steps:

1. Select the associated task from the navigation tree. For example, to display JDBC providers that have been defined, select **Resources** → **JDBC** → **JDBC Providers**. See Figure 5-6.
2. Certain resources are defined at a scope level. If applicable, select the scope from the drop-down menu.
3. Set the preferences to specify how you would like information to be displayed on the page.

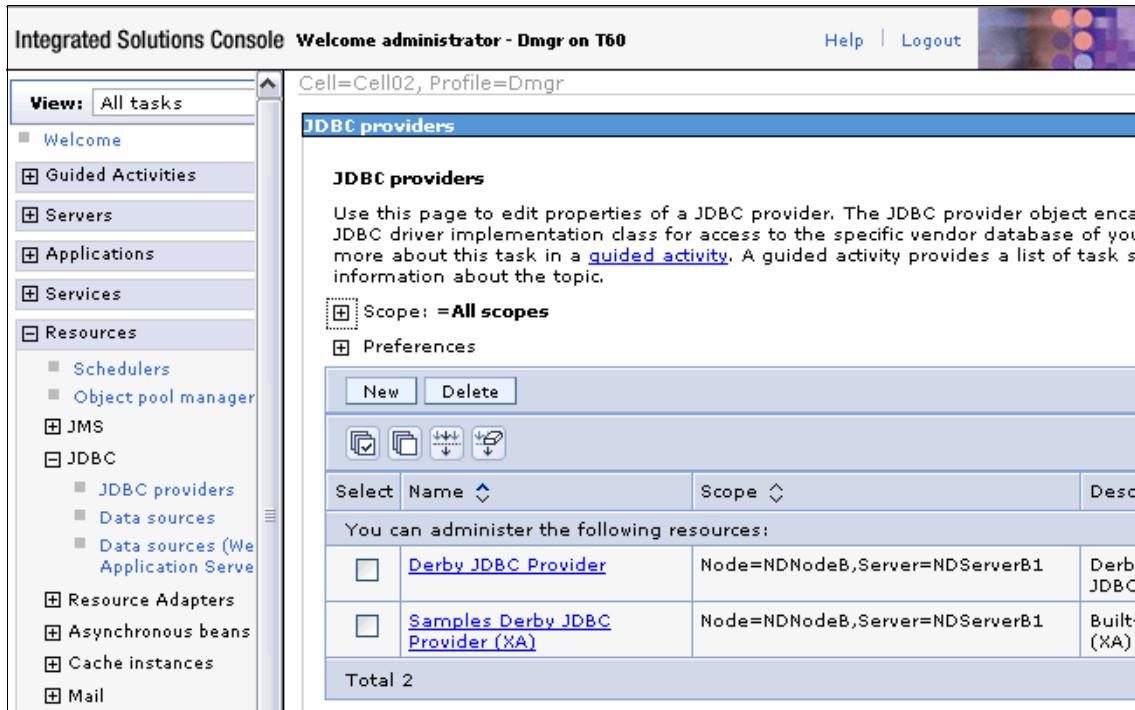


Figure 5-6 Working with the administrative console

Selecting a scope

The scope level determines which applications or application servers will see and use that configuration. The scope setting is available for all resource types, WebSphere variables, shared libraries, and name space bindings.

Scope levels

Configuration information is defined at the following levels: cell, cluster, node, server, and application. Here, we list these scopes in overriding sequence. Because you see application scope first, anything defined at this scope overrides any conflicting configuration you might find in the higher level scopes:

1. Resources and variables scoped at the *application* level apply only to that application. Resources and variables are scoped at the application level by defining them in an enhanced EAR. They cannot be created from the WebSphere administrative tools, but can be viewed and modified (in the administrative console, navigate to the details page for the enterprise application and select **Application scoped resources** in the References section).

2. Resources scoped at the *server* level apply only to that server. If a node and server combination is specified, the scope is set to that server. Shared libraries configured in an enhance EAR are automatically scoped at the server level.
3. Resources scoped at the *node* level apply to all servers on the node.
4. Resources scoped at the *cluster* level apply to all application servers in the cluster. New cluster members automatically have access to resources scoped at this level. If you do not have any clusters defined, you will not see this option.
5. Resources scoped at the *cell* level apply to all nodes and servers in the cell.

Stand-alone application servers: Although the concept of cells and nodes is more relevant in a managed server environment, scope is also set when working with stand-alone application servers. Because there is only one cell, node, and application server, and no clusters, simply let the scope default to the node level.

Configuration information is stored in the repository directory that corresponds to the scope. For example, if you scope a resource at the node level, the configuration information for that resource is in:

```
<profile_home>/config/cells/cell_name/nodes/<node>/resources.xml
```

If you scoped that same resource at the cell level, the configuration information for that resource is in:

```
<profile_home>/config/cells/cell_name/resources.xml
```

Setting scope levels in the console

Collection pages that contain items that require a scope level to be identified provide two different options for defining the scope. Setting the scope level both sets the level for any resources you create and limits what is displayed in the collection page.

Selecting the **Show scope selection drop-down list with the all scopes** option provides a drop-down box with all scopes that you can select from, including the “All scopes” option (Figure 5-7). Selecting a scope from the drop-down list changes the scope automatically.

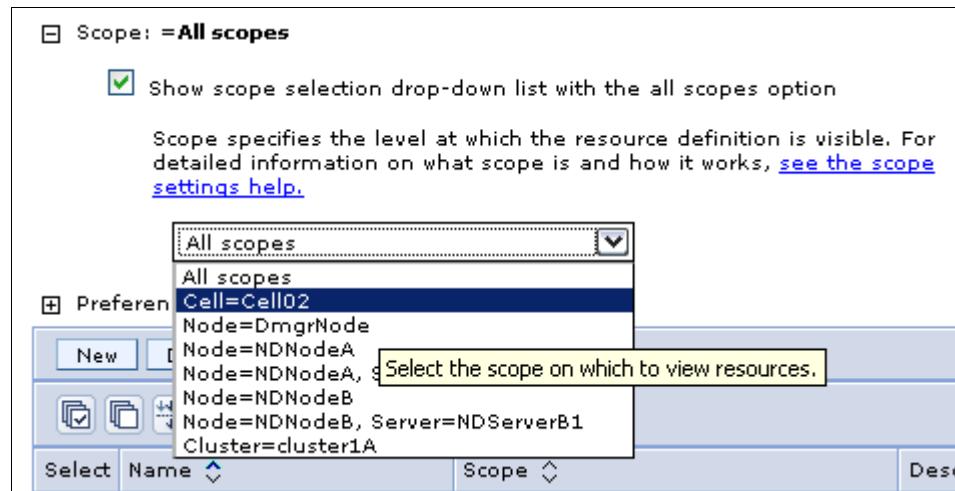


Figure 5-7 Scopes selected using a drop-down list

(New in V7) The second option for setting the scope is to de-select the **Show scope selection drop-down list with the all scopes**. Instead of a drop-down, you have fields for each scope level where you can browse a list of applicable entries at that scope level. Click **Apply** to complete the selection (Figure 5-8).

The scope is set to the lowest level entry you select (a red arrow to the left of the field indicates the current scope). To move to a higher scope, simply clear the lower field. For example, if you select a server as the scope level, and want to change the scope to the node level, clear the server field and click **Apply**.

This option is useful in cells that contain a large number of nodes, servers, or clusters. In those situations, the drop-down list can be difficult to navigate. However, note that the option to view all scopes is not available.

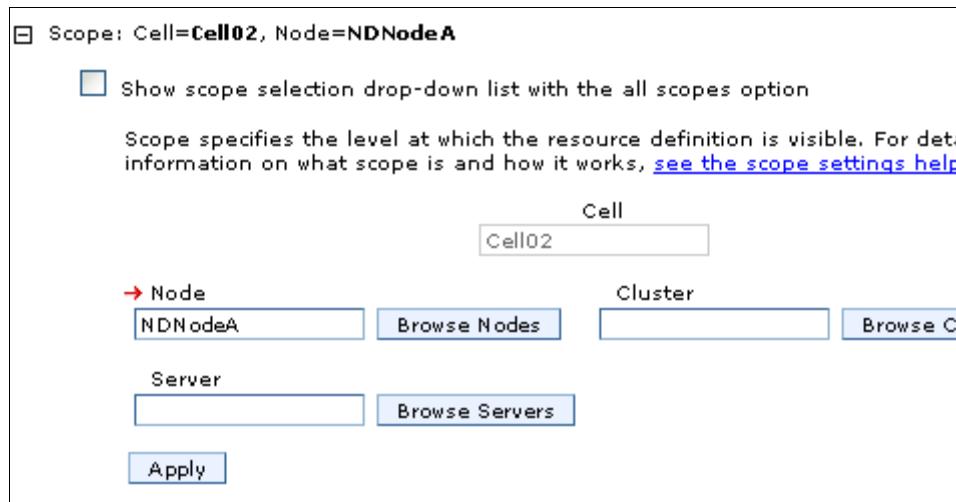


Figure 5-8 Selecting the scope with individual fields

Set preferences for viewing the console page

After selecting a task and a scope, the administrative console page shows a collection table with all the objects created at that particular scope. You can change the list of items you see in this table by using the filter and preference settings.

The preference settings available will vary by the type of item you are displaying. Many of these settings are new in V7. A list of the preference settings and their use is available in the Information Center:

- ▶ Administrative console preference settings:

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

Figure 5-9 shows the preference settings you would see when displaying a list of JDBC providers.

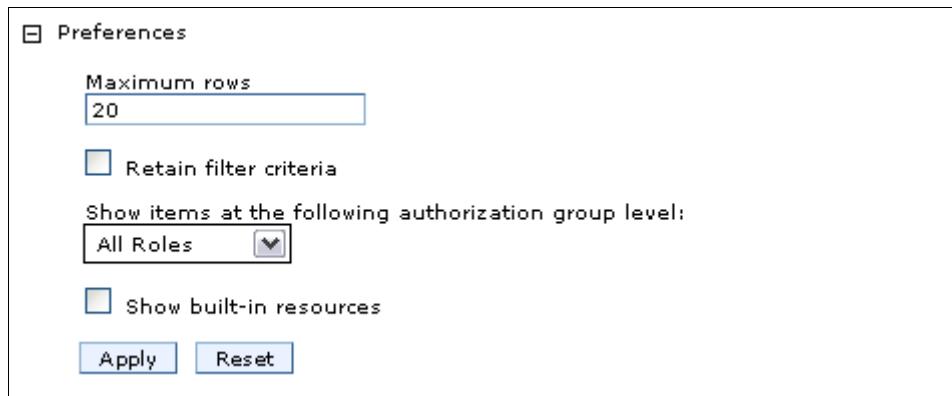


Figure 5-9 Filter and preference settings

The filter options can be displayed or set by clicking the Show Filter Function icon  at the top of the table. See Figure 5-10.

Figure 5-10 Setting filters and preferences

When you click the icon, a new area appears at the top of the table allowing you to enter filter criteria. To filter entries, do the following steps:

1. Select the column to filter on. For example, in Figure 5-10, the display table has three columns to choose from. Your options vary depending on the type of item you are filtering.

2. Enter the filter criteria. The filter criteria is case sensitive and wild cards can be used. In our example, to see only providers with names starting with "S", select the Name column to filter on and enter S* as the filter.
3. Click **Go**.
4. After you have set the filter, click the **Show Filter** icon again to remove the filter criteria from view. You still have a visual indication that the filter is set at the top of the table.

Setting the filter is temporary and only lasts for as long as you are in that collection. To keep the filter active for that collection, check the **Retain filter criteria** box in the Preferences section and click **Apply**. To clear the filter criteria, click the  icon.

For more help on using the filtering feature, see:

- ▶ Administrative console buttons:

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

5.1.6 Updating existing items

To edit the properties of an existing item, complete these tasks:

1. Select the category and type in the navigation tree. For example, select **Servers → Server Types → WebSphere application servers**.
2. A list of the items of that type in the scope specified will be listed in a collection table in the workspace area. Click an item in the table. This opens a detail page for the item.
3. In some cases, you see a Configuration tab and a Runtime tab on this page. In others, you only see a Configuration tab.

Updates are done under the Configuration tab. Specify new properties or edit the properties already configured for that item. The configurable properties will depend on the type of item selected.

For example, if you select a WebSphere Application Server cluster, this opens a detail page resembling Figure 5-11.

The screenshot shows a configuration interface for a WebSphere Application Server cluster named 'cluster1A'. The interface is divided into several sections:

- General Properties**: Contains fields for 'Cluster name' (set to 'cluster1A'), 'Bounding node group name' (set to '(none)'), and checkboxes for 'Prefer local' (checked) and 'Enable failover of transaction log recovery' (unchecked).
- Cluster messaging**: A section containing a single link: [Messaging engines](#).
- Additional Properties**: A section containing four links:
 - [Cluster members](#)
 - [Backup cluster](#)
 - [Endpoint listeners](#)
 - [Security domain](#)

At the bottom of the page are four buttons: 'Apply', 'OK', 'Reset', and 'Cancel'.

Figure 5-11 Editing an application server cluster properties

The detail page provides fields for configuring or viewing the more common settings and links to configuration pages for additional settings.

4. Click **OK** to save your changes to the workspace and exit the page. Click **Apply** to save the changes without exiting. The changes are still temporary. They are only saved to the workspace, not to the master configuration. This still needs to be done.

5. As soon as you save changes to your workspace, you will see a message in the Messages area reminding you that you have unsaved changes. See Figure 5-12.

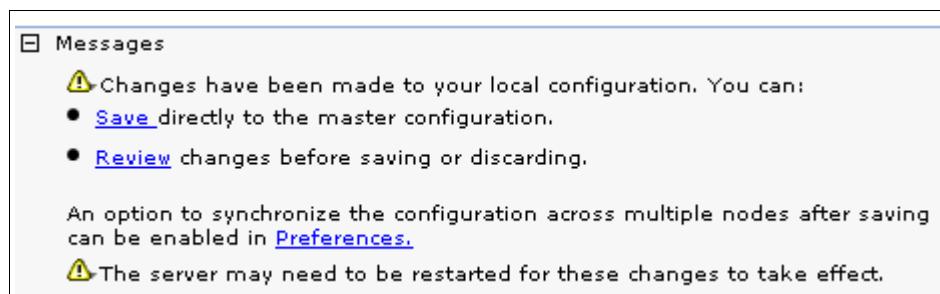


Figure 5-12 Save changes to the master repository

At intervals during your configuration work and at the end, you should save the changes to the master configuration. You can do this by clicking **Save** in the Messages area, or by selecting **System administration Save Changes to Master Repository** in the navigation tree.

To discard changes, use the same options. These options simply display the changes you have made and give you the opportunity to save or discard.

5.1.7 Adding new items

To create new instances of most item types (Figure 5-13), complete these tasks:

1. Select the category and type in the navigation tree.
2. Select the **Scope**. (To create a new item, you cannot select the **All** option for scope.)
3. Click the **New** button above the collection table in the workspace.

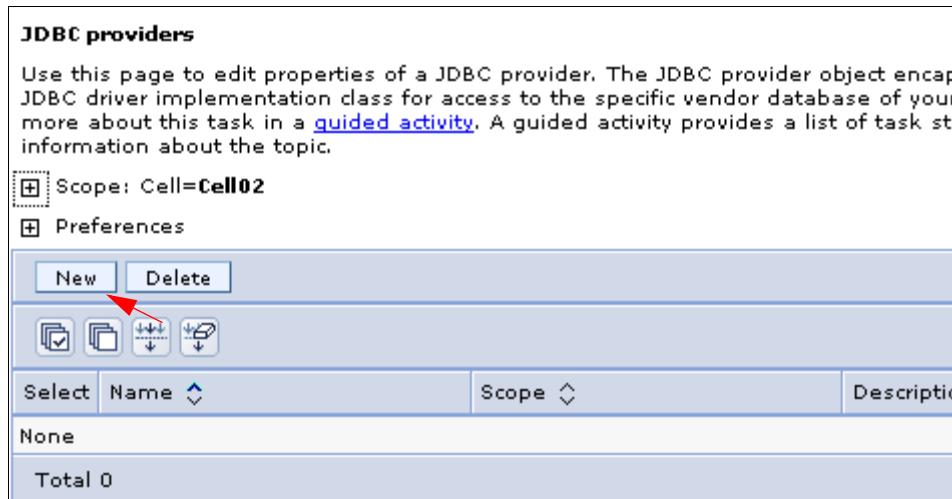


Figure 5-13 Create a new item

When you click **New** to add an item, one of two things will happen, depending on the type of item you are creating. A wizard will start to guide you through the definitions, or a new details page will open allowing you to fill in the basic details. In the latter case, enter the required information and click **Apply**. This will usually activate additional links to detail pages required to complete the configuration.

Note: In the configuration pages, you can click **Apply** or **OK** to store your changes in the workspace. If you click **OK**, you will exit the configuration page. If you click **Apply**, you will remain in the configuration page. As you are becoming familiar with the configuration pages, we suggest that you always click **Apply** first. If there are additional properties to configure, you will not see them if you click **OK** and leave the page.

4. Click **Save** in the task bar or in the Messages area when you are finished.

5.1.8 Removing items

To remove an item (Figure 5-14), complete these tasks:

1. Find the item.
2. Select the item in the collection table by checking the box next to it.
3. Click **Delete**.
4. If asked whether you want to delete it, click **OK**.
5. Click **Save** in the Messages area when you are finished.

For example, to delete an existing JDBC provider, select **Resources** → **JDBC** → **JDBC Providers**. Check the provider you want to remove and click **Delete**.

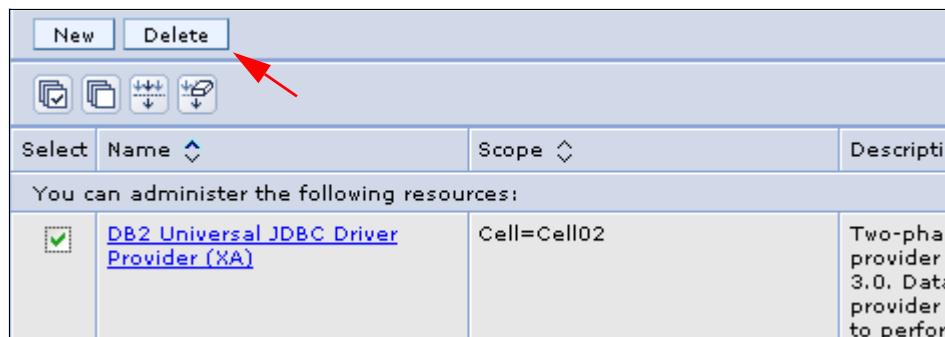


Figure 5-14 Deleting an item

5.1.9 Starting and stopping items

To start or stop an item using the console:

1. Select the category and type in the navigation tree.
2. Select the item in the collection table by checking the box next to it.
3. Click **Start** or **Stop**. The collection table shows the status of the item. See Figure 5-15.

For example, to start an application server in a distributed server environment, select **Servers** → **Server Types** → **WebSphere application servers**. Place a check mark in the check box beside the application server you want and click **Start**.

Application servers

Use this page to view a list of the application servers in your environment and the status of each server. You can also use this page to change the status of a specific application server.

[Preferences]

New Delete Templates... Start Stop Restart ImmediateStop Terminate

Select Name ▲ Clear filter value Host Name ▲ Version ▲ Cluster Name ▲ Status

You can administer the following resources:

	Name	Host Name	Version	Cluster Name	Status
<input type="checkbox"/>	NDServerB1	NDNodeB	t60	ND 7.0.0.1	
<input checked="" type="checkbox"/>	ServerA1	NDNodeA	t60	ND 7.0.0.1	

Figure 5-15 Starting and stopping items

Not all items can be started and stopped from the console. For example, the deployment manager and nodes must be started independently from the console. Also, there can be multiple options for starting and stopping an item (restart, stop immediate, etc.).

5.1.10 Using variables

WebSphere variables are name and value pairs used to represent variables in the configuration files. This makes it easier to manage a large configuration.

To set a WebSphere variable:

1. Click **Environment** → **WebSphere Variables**. See Figure 5-16.

WebSphere Variables

Use this page to define substitution variables. Variables specify a level of indirection for some system defined values, such as file system root directories. Variables have a scope level, which is either server node, cluster, or cell. Values at one scope level can differ from values at other levels. When a variable has conflicting scope values, the more granular scope value overrides values at greater scope levels. Therefore, server variables override node variables, which override cluster variables, which override cell variables.

Scope: Cell=Cell02

Preferences

New	Delete

Select	Name	Value	Scope
You can administer the following resources:			
<input type="checkbox"/>	DB2UNIVERSAL JDBC DRIVER_NATIVEPATH	C:\SQLLIB_Client\java	Cell=Cell02
<input type="checkbox"/>	DB2UNIVERSAL JDBC DRIVER_PATH	C:\SQLLIB_Client\java	Cell=Cell02

Figure 5-16 WebSphere variables

2. To add a new variable, click **New**, or click a variable name to update its properties.
3. Enter a name and value and click **Apply**. See Figure 5-17.

General Properties

* Name

Value

Description
 The directory that contains the DB2 JDBC Driver.
 [scrollbar]

Apply OK Reset Cancel

Figure 5-17 New WebSphere variable

5.1.11 Saving work

As you work with the configuration, your changes are saved to temporary workspace storage. For the configuration changes to take effect, they must be saved to the master configuration. If you have a distributed server environment, a second step is required to *synchronize*, or send, the configuration to the nodes. Consider the following possibilities.

If you work on a page, and click **Apply** or **OK**, the changes are saved in the workspace under your user ID. This allows you to recover changes under the same user ID if you exit the session without saving.

You need to save changes to the master repository to make them permanent. You have several options to save:

- ▶ Use the Save window in the Messages area. If this is displayed, it is the quickest method.
- ▶ Selecting **System administration Save Changes to Master Repository**.
- ▶ When you log in, if you logged out without saving the changes, you will be given the option to save the changes.

The Save window presents you with the following options:

- ▶ Save
- ▶ Discard: This reverses any changes made during the working session and reverts to the master configuration.
- ▶ Cancel: This does not reverse changes made during the working session. It just cancels the action of saving to the master repository for now.
- ▶ Synchronize changes with nodes: This distributes the new configuration to the nodes in a distributed server environment.

Before deciding whether you want to save or discard changes, you can see the changed items by expanding **Total changed documents** in the Save window.

Important: All the changes made during a session are cumulative. Therefore, when you decide to save changes to the master repository, all changes are committed. There is no way to be selective about what changes are saved to the master repository.

5.1.12 Getting help

Help is available to you in several different ways:

- ▶ Click **Help** on the administrative console banner. This opens a new Web browser with online help for the administrative console. It is structured by administrative tasks. See Figure 5-18.



Figure 5-18 Online help

- ▶ With the option **Show the help portlet** enabled, you can see the Help window in the workspace. Click **More information about this page**. This will open the help system to a topic-specific page.
- ▶ The Information Center can be viewed online or downloaded from:
<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.home.doc/welcome.html>

5.2 Securing the console

WebSphere Application Server provides the ability to secure the administrative consoles so only authenticated users can use them. Note that enabling administrative security does not enable application security.

Before enabling any type of security for a production system, you should be very familiar with WebSphere security and have a plan for securing your WebSphere environment. Security encompasses many components, including administrative security, application security, infrastructure security, and specialized resource security options. This section only provides an overview of administrative security.

The first decision you have to make is to select the user registry you will use. If you enable security when you create a profile for distributed systems, a file-based registry is automatically created and populated with one administrative user ID. On z/OS platforms, you will have the option of using the file-based registry or the z/OS system's SAF-compliant security database.

While a file-based user registry is not a good choice for securing applications, you can federate additional registries to the existing file-based registry to manage users and groups for application security.

If you are using a registry other than the WebSphere Application Server federated user registry, you must create at least one user ID to be used for the WebSphere administrator.

And while you might have heard about security domains that have been introduced in WebSphere Application Server V7, these domains are used for application security (not administrative security).

Before implementing security in a production environment, be sure to consult the *WebSphere Application Server V7 Security Guide*, SG24-7660.

5.2.1 Enabling security after profile creation

You can enable administrative security after profile creation through the administrative console by navigating to **Security** → **Global security**. Doing this allows you more flexibility in specifying security options. You will need to complete the configuration items for authentication, authorization, and realm (user registry). You will also need to populate the chosen user registry with at least one user ID to be used as an administrator ID (Figure 5-19).

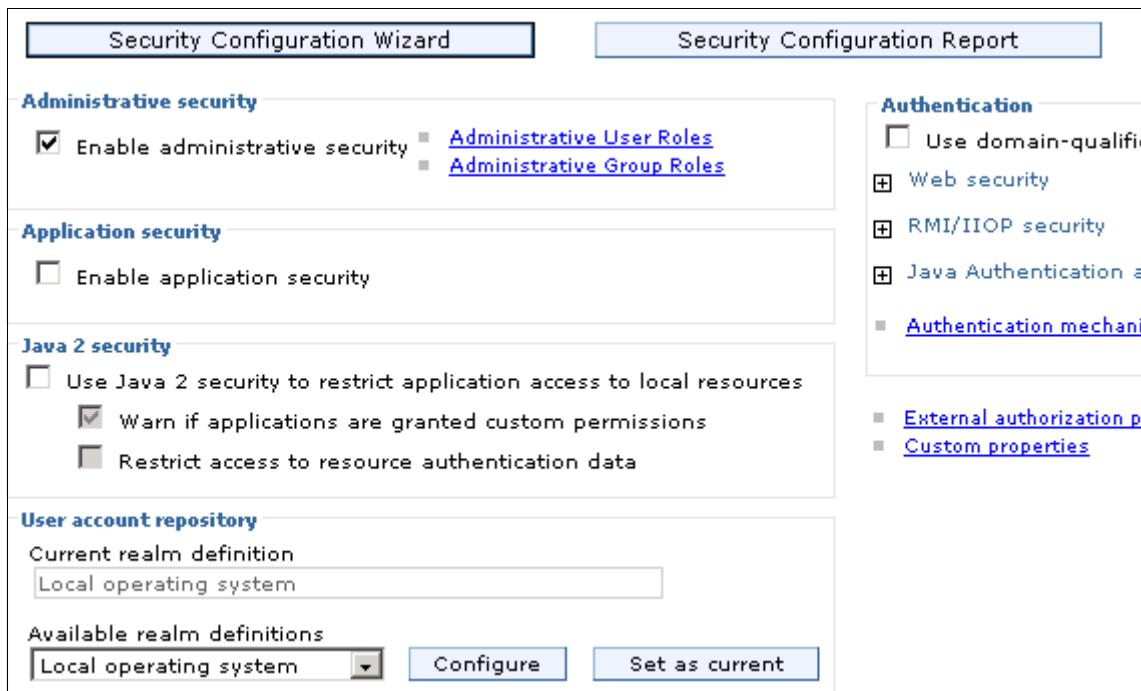


Figure 5-19 Enabling administrative security

Attention: Be aware that when you check the box to enable administrative security, the application security and Java 2 security check boxes are enabled automatically. If you are not prepared to use Java 2 or application security at this time, be sure to uncheck the boxes.

Tip: If you enable administrative security, and then find you cannot login, you can disable the security manually by editing the security.xml profile. This allows you to go back through the security configuration to see where things went wrong.

1. Open the security.xml file at *dmgr_profile_home/config/cells/cell_name*
2. Edit the second line, changing enabled="true" to enabled="false".

```
<security:Security xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:orb.securityprotocol="http://www.ibm.com/websphere/appserver/schemas/5.0/orb.securityprotocol.xmi"
  xmlns:security="http://www.ibm.com/websphere/appserver/schemas/5.0/security.xmi" xmi:id="Security_1" useLocalSecurityServer="true"
  useDomainQualifiedUserNames="false" enabled="false"
  cacheTimeout="600" issuePermissionWarning="false"
  activeProtocol="BOTH" enforceJava2Security="false"
  enforceFineGrainedJCASecurity="false" appEnabled="false"
  dynamicallyUpdateSSLConfig="true" allowBasicAuth="true"
  activeAuthMechanism="LTPA_1"
  activeUserRegistry="WIMUserRegistry_1"
  defaultSSLSettings="SSLConfig_1">
```

After saving the configuration, you must restart the application server in a stand-alone server environment or the deployment manager in a distributed server environment.

The next time you log in to the administrative console, you must authenticate with the user ID that was identified as having an administrative role. Entering commands from a command window will also prompt you for a user ID and password. You can add additional administrative users and assign authorization levels from the administrative console.

5.2.2 Administrative security roles

Administrative security is based on identifying users or groups that are defined in the active user registry and assigning roles to each of those users. When you log in to the administrative console or issue administrative commands, you must use a valid administrator user ID and password. The role of the user ID determines the administrative actions the user can perform.

Fine-grained administrative security (new in V7):

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

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

The following administrative security roles are available:

- ▶ Administrator:
The administrator role has operator permissions, configurator permissions, and the permission required to access sensitive data, including server password, Lightweight Third Party Authentication (LTPA) password and keys, and so on.
- ▶ Configurator:
The configurator role has monitor permissions and can change the WebSphere Application Server configuration.
- ▶ Operator:
The operator role has monitor permissions and can change the runtime state. For example, the operator can start or stop services.
- ▶ Monitor:
The monitor role has the least permissions. This role primarily confines the user to viewing the WebSphere Application Server configuration and current state.
- ▶ Deployer:
The deployer role is only available for **wsadmin** users, not for administrative console users. Users granted this role can perform both configuration actions and runtime operations on applications.
- ▶ AdminSecurityManager:
The AdminSecurityManager role is only available for **wsadmin** users, not for administrative console users. When using **wsadmin**, users granted this role can map users to administrative roles. When fine grained administrative security is used, users granted this role can manage authorization groups.

► **Iscadmins:**

The iscadmins role has administrator privileges for managing users and groups from within the administrative console only.

Assigning administrative roles to users and groups

If you are using a file-based repository, you can add users and groups through the console by navigating to **Users and groups Manage Users** or **Users and groups Manage Groups**. Otherwise, the users and groups must be added to the user registry using the tools provided by the registry product.

Role assignments for users and groups are managed through the administrative console. Navigate to **Users and groups Administrative User Roles** or **Users and groups Administrative Group Roles**. Use these panels to assign an administrative role to a user or group.

(New in V7) Fine-grained security

In releases prior to WebSphere Application Server version V7, users granted administrative roles could administer all of the resource instances in the cell. WebSphere Application Server is now more fine-grained, meaning that access can be granted to each user per resource instance. For example, users can be granted configurator access to a specific instance of a resource only (an application, an application server or a node). The administrative roles are now per resource instance rather than to the entire cell.

To achieve the instance-based security or fine-grained security, resources that require the same privileges are placed in a group called the administrative authorization group or authorization group. Users can be granted access to the authorization group by assigning to them the required administrative role.

You can define groups of resources that will be treated collectively by navigating to **Security Administrative Authorization Groups**. The resource instances which are added to an authorization group can be the following types:

- Cluster
- Node
- Servers, including application servers and Web servers
- Applications, including business level applications
- Node groups
- Assets

After the authorization group has been created, you can assign users or groups an administrative role for the authorization group.

Many administrative console pages have a preference setting that allows you to restrict the items that you can see to those that are valid for your authorization group level. The roles that you can choose from depend on the role of the user ID you are logged into the console with.

5.3 Job manager console

The job manager console has many of the basic options that you find in the administrative console, including global security settings, the option to add users and groups to the federated user repository, WebSphere variable settings, and others that are common to any administrative environment.

What is unique to the job manager console is the ability to submit jobs to nodes registered to it.

Figure 5-20 shows a job manager console. The option selected in the Navigation tree is **Jobs → Nodes**. You can see in this example, that one application server node has been registered from an admin agent. Two deployment manager nodes are registered as well.

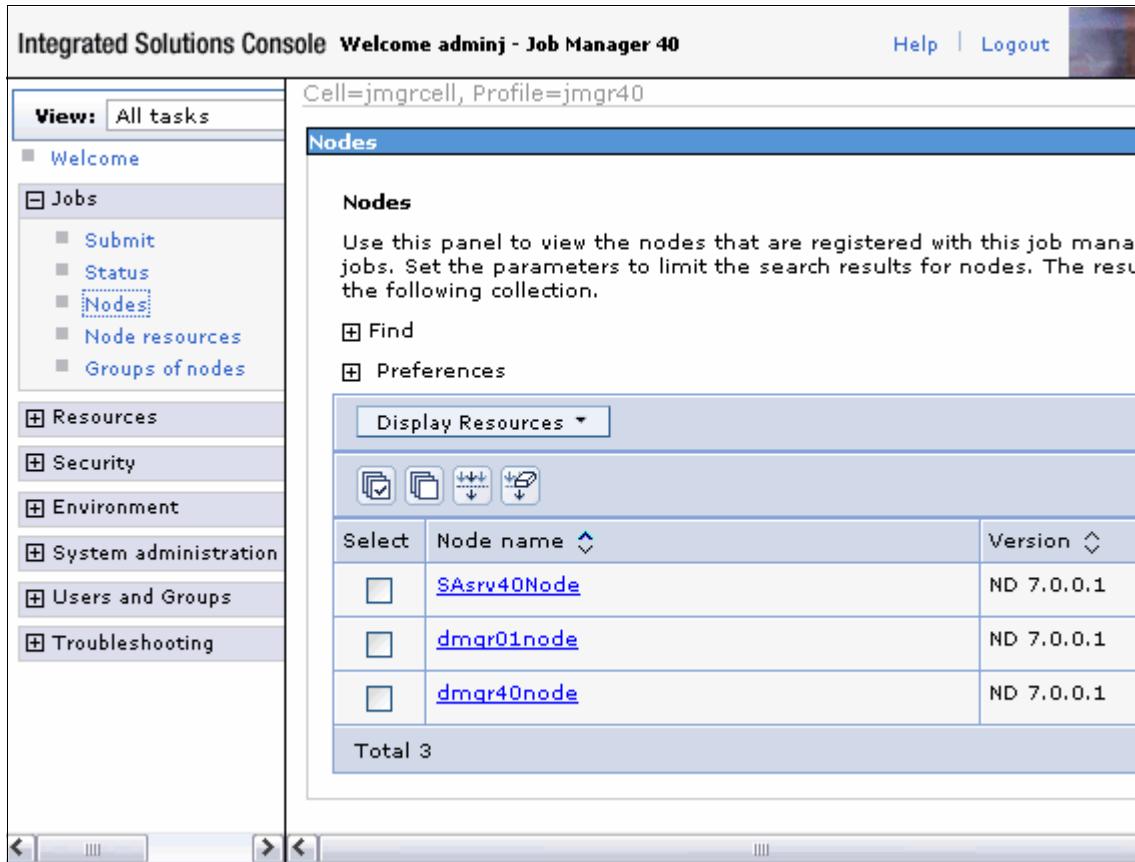


Figure 5-20 Job manager console - list of nodes

Groups of nodes: You can create “groups of nodes” that contain the nodes you will work with from the job manager (select **Jobs** → **Groups of nodes**). A group of nodes can be used as the target of administrative jobs.

When you submit a job, you can select one or more groups from a drop-down. The alternative is to type in the name of the node or use the Find feature to select each node. Using the Find feature takes several steps.

So, even if you do not plan to use multiple nodes as the target of a job, creating a group for each node allows you to easily select a node rather than typing it in or searching for it.

If you include multiple nodes in the group, beware that all the nodes have to have a common user ID and password. When you submit a job you only have one place where you can enter the user ID and password.

5.3.1 Submitting a job with the job manager

The job manager provides the following job types:

- ▶ Run a wsadmin script
- ▶ Manage applications
 - distributeFile
 - collectFile
 - removeFile
 - startApplication
 - stopApplication
 - installApplication
 - updateApplication
 - uninstallApplication
- ▶ Manage servers
 - createApplicationServer
 - deleteApplicationServer
 - createProxyServer
 - deleteProxyServer
 - createCluster
 - deleteCluster
 - createClusterMember
 - deleteClusterMember
 - configureProperties
- ▶ Manage the server runtime
 - startServer
 - stopServer
 - startCluster
 - stopCluster

Details on each of these job types can be found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rxml_7jobtypes.html

Follow these steps:

1. Start the job manager and log in to the job manager console:

`http://<job_manager_host>:9960/ibm/console`

2. To submit jobs, nodes must have already been registered with the job manager. To verify which nodes have been registered, expand **Jobs** in the navigation window, and select **Nodes**. If this is the first time using the job manager, you might not see all the nodes displayed. To refresh the view, enter * as the value for Node name and click **Find** (Figure 5-21).

The screenshot shows the 'Nodes' panel with the following interface elements:

- Find:** A section containing three dropdown menus for filtering by Node name, Job type, and Unique identifier, each set to '='. To the right of the first dropdown is a text input field containing '*'.
- Advanced find options:** A link to expand or collapse additional search criteria.
- Maximum results:** A text input field set to '50'.
- Buttons:** 'Find' and 'Reset' buttons at the bottom left, and a status message 'Retrieved 2 of' followed by a total count at the bottom right.
- Preferences:** A section with a 'Display Resources' dropdown menu and a toolbar with icons for selecting, deleting, and sorting.
- Table:** A grid displaying two registered nodes:

Select	Node name	Version
<input type="checkbox"/>	dmgr01node	ND 7.0.0.1
<input type="checkbox"/>	dmgr40node	ND 7.0.0.1
- Total:** A summary at the bottom stating 'Total 2'.

Figure 5-21 List of all nodes registered with job manager

3. Select **Jobs** → **Submit** to select the type of job to submit (Figure 5-22) and click **Next**.

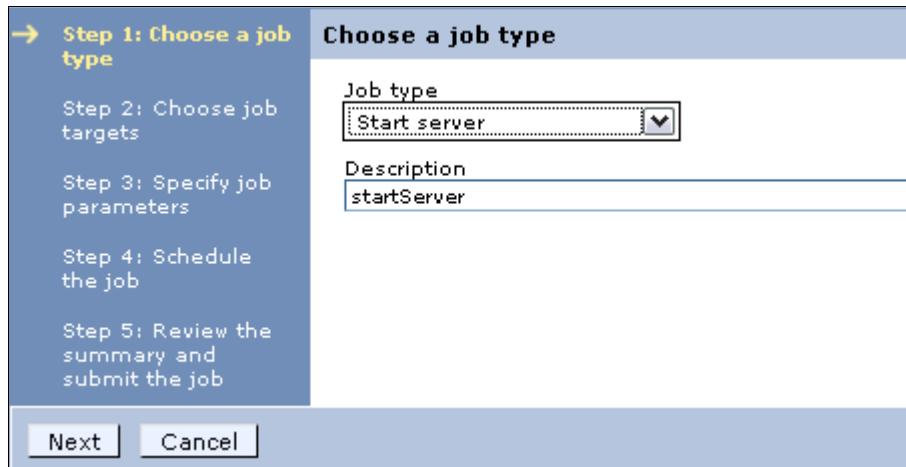


Figure 5-22 Select the job type

4. Select the node that you want to run the job on. You can select from a node group by using the drop-down next to the **Groups of nodes** field. Or you can select specific nodes.

Enter the user ID and password for the node that you will run the job against (Figure 5-23).

The screenshot shows a software interface for scheduling a job. On the left, a vertical sidebar lists five steps: Step 1: Choose a job type, Step 2: Choose job targets (which is currently selected), Step 3: Specify job parameters, Step 4: Schedule the job, and Step 5: Review the summary and submit the job. The main area is titled 'Choose job targets' and has a sub-section 'Job type: Start server'. It contains two radio button options: 'Groups of nodes' (unchecked) and 'Node names' (checked). Below this is a dropdown menu showing 'group40' with a dropdown arrow. To the right of the dropdown are 'Add' and 'Find...' buttons. A large rectangular list box is present, which is currently empty. At the bottom right of this list box is a 'Remove' button. Below the list box is a section titled 'Node authentication' with three input fields: 'User name' containing 'adminj', 'Password' containing '*****', and 'Confirm password' containing an empty field. At the very bottom of the dialog are three buttons: 'Previous', 'Next', and 'Cancel'.

Figure 5-23 Select the job target

To use a specific node, select **Node names** and either enter the node name and click **Add**, or click **Find**. Using the Find option opens a new panel where you can search and select nodes (Figure 5-24).

The simplest method of searching is to enter an “*” in the Node name field and click **Find**.

The list of nodes is shown in the Excluded nodes box. Select the nodes you want and use the arrow button to move them to the Chosen nodes box. You can hold the shift key down to select multiple nodes, or move them one at a time.

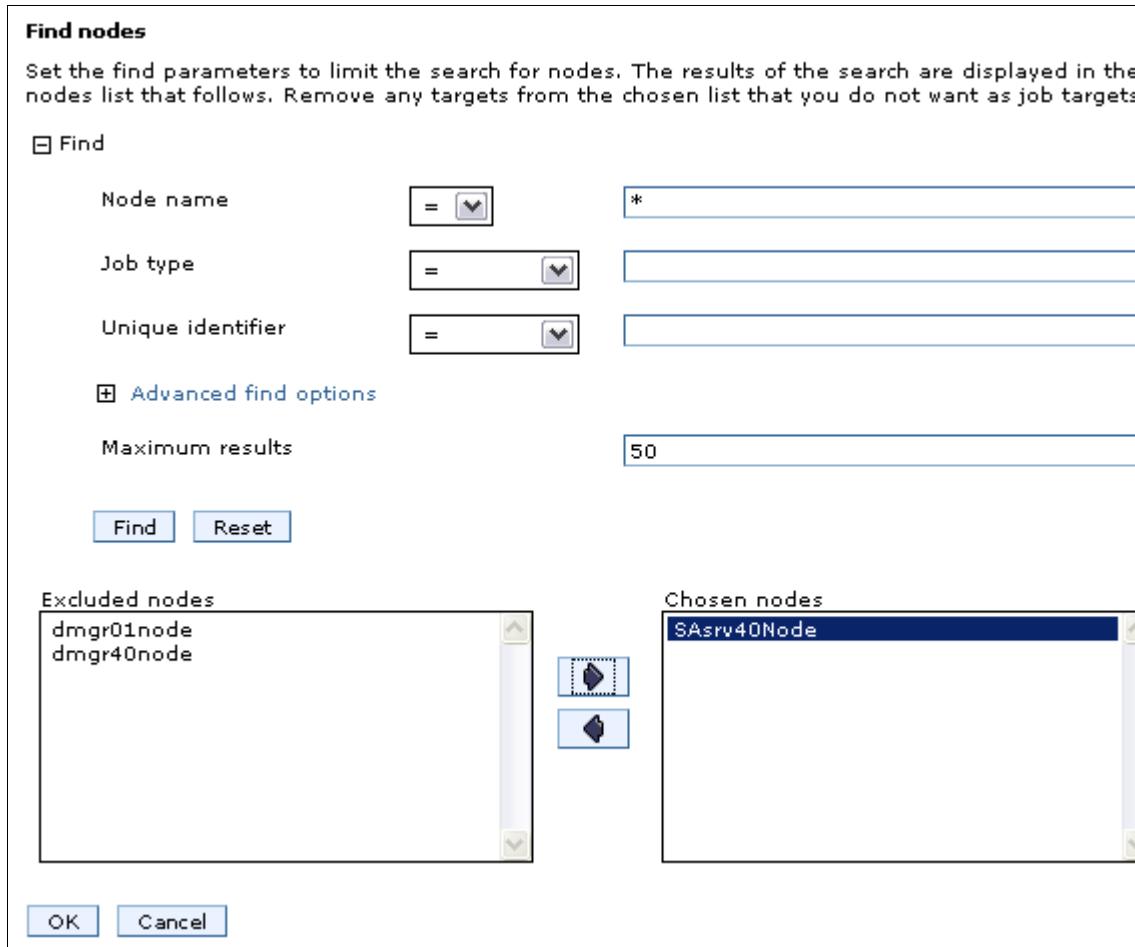


Figure 5-24 Search and select nodes

Click **OK**. This will return you to step 2 of the wizard with the node name filled in.

Click **Next** to continue the job submit process.

5. Specify the job parameters. These will vary widely depending on the type of job. The parameters provide the additional information the job will need to perform the task. For example, if you are running a job to start a server, you have selected the node in the previous step, but the server name must be entered as a parameter.

Click **Next**.

6. The next step contains fields that specify how and when the job should run, and if a notification via e-mail should be sent (Figure 5-25).

Step 1: Choose a job type Step 2: Choose job targets Step 3: Specify job parameters Step 4: Schedule the job Step 5: Review the summary and submit the job	<h3>Schedule the job</h3> <p>Job type: Start server</p> <p>Notification</p> <p>E-mail addresses <input type="text"/></p> <p>Initial Availability</p> <p>Specify when this job is first available.</p> <p><input checked="" type="radio"/> Make the job available now. <input type="radio"/> Schedule availability</p> <p>Date (MM/dd/yyyy) <input type="text"/> / <input type="text"/> / <input type="text"/> Time (HH:mm:ss) <input type="text"/> : <input type="text"/> : <input type="text"/></p> <p>Expiration</p> <p>Specify when this job is no longer available.</p> <p><input checked="" type="radio"/> Use default expiration - 1 days. <input type="radio"/> Expire the job based on a date</p> <p>Date (MM/dd/yyyy) <input type="text"/> / <input type="text"/> / <input type="text"/> Time (HH:mm:ss) <input type="text"/> : <input type="text"/> : <input type="text"/></p> <p><input type="radio"/> Expire the job based on a duration</p> <p>Expire after <input type="text"/> minutes <input type="button" value="▼"/></p> <p>Job Availability Interval</p> <p>Jobs can run repeatedly based on an interval. Specify the interval that the job is available.</p> <p>Availability interval <input type="button" value="Run once"/> <input type="button" value="▼"/></p> <p style="text-align: center;">Previous Next Cancel</p>
---	--

Figure 5-25 Specify job scheduling information

- Notification: The e-mail address specified will receive a notification when the job is finished. In order to use this field, you must configure a mail provider and mail session.
- Initial availability: You can make the job available now (it will run immediately after you have finished with the job submission process), or you can specify a date and time it will be available.
- Expiration: Specify an expiration date for the job.
- Job availability interval: This field allows you to repeat job submission at intervals. Depending on the selection, you will have an additional field displayed that allows you to choose the days, start and stop time, and so on (Figure 5-26).



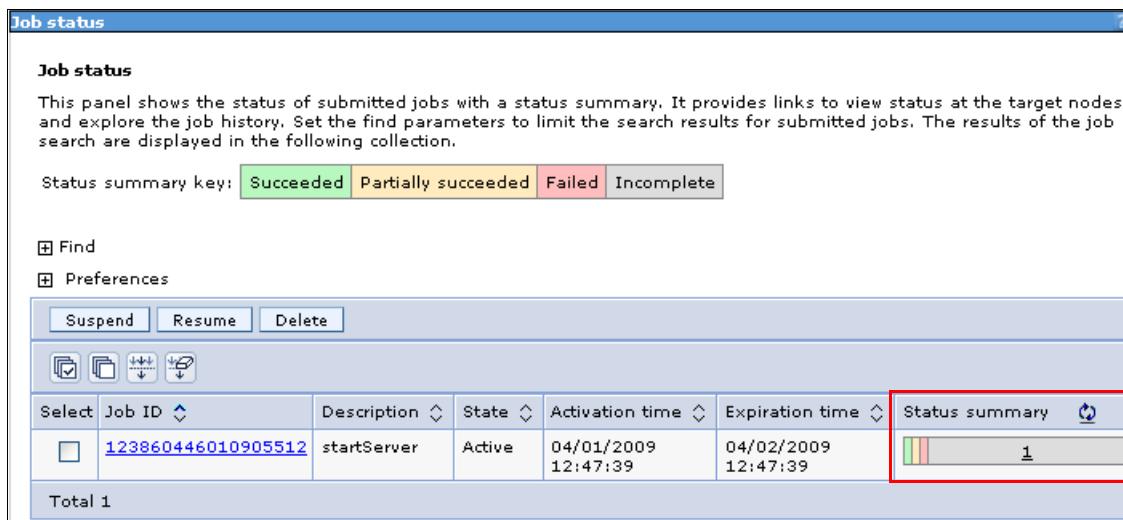
Figure 5-26 Job interval options

If you select **Make this job available now** and **Run once**, the job runs right away and the Expiration settings have no meaning. The alternative is to set an Initial availability, Expiration date or duration, and select an interval that the job will run at.

7. Review the summary and submit the job.

When a job is submitted from the job manager, the job details are saved in a database local to the job manager. The endpoint (deployment manager or administrative agent) pings the job manager at a predefined interval and fetches jobs that are to be executed. If the job submitted is a wsadmin job, the wsadmin script is executed. Otherwise, a corresponding job handler will execute the necessary admin code.

8. The Job status panel allows you to monitor the results. Use the Refresh icon in the Status summary column () to update the status. The color in the Status summary field will indicate the success or failure of the job (Figure 5-27).



The screenshot shows the 'Job status' interface. At the top, there's a 'Job status' header and a brief description of the panel's function. Below that is a 'Status summary key' section with four colored buttons: green (Succeeded), light blue (Partially succeeded), red (Failed), and grey (Incomplete). There are also 'Find' and 'Preferences' buttons. A toolbar with icons for Suspend, Resume, and Delete follows. The main area is a table with columns: Select, Job ID, Description, State, Activation time, Expiration time, and Status summary. The first row in the table has a 'Select' checkbox checked, a 'Job ID' of 123860446010905512, a 'Description' of startServer, a 'State' of Active, and activation/expiration times of 04/01/2009 12:47:39. The 'Status summary' column for this row is highlighted with a red border and contains a green bar with the number '1'. The table footer shows a total of 1 item.

Select	Job ID	Description	State	Activation time	Expiration time	Status summary
<input checked="" type="checkbox"/>	123860446010905512	startServer	Active	04/01/2009 12:47:39	04/02/2009 12:47:39	 1

Figure 5-27 Status summary

- Click the Job ID to see more information about the job (Figure 5-28).

The screenshot shows the 'Job status' page with the following details:

- Job status > 123860446010905512**: Shows the job status at each node. Set the find parameters to limit the search results for submitted jobs.
- General Properties** section:
 - Job ID: 123860446010905512
 - Description: startServer
 - Activation time: 04/01/2009 12:47:39
 - Expiration time: 04/02/2009 12:47:39
- Node names** table:

Node names	Status
SAsrv40Node	Succeeded
- Back** button

Figure 5-28 Job status information

Job status is always sent back to the job manager. Clicking the message in the Status column (Succeeded in this case) shows you additional information.

In the event of an error, you will see any messages produced by the job. Additional messages might be available in the logs for the server where the administrative action was to take place (Figure 5-29).

Job status		
Job status > 123860446010905512 > SAsrv40Node		
A detailed job history can be retrieved based on time.		
Find		
Time stamp	Status	Message
2009-04-01T12:47:49-0400	Distributed	
2009-04-01T12:47:51-0400	In progress	
2009-04-01T12:49:24-0400	Succeeded	CWWWSY0328I: Server sasrv40 was started on node SAsrv40Node
Previous records	Next records	Back

Figure 5-29 Job output

When you execute configuration type job (for example, create server) from the job manager to a deployment manager, the configuration will be saved if the job is successful. A job that submits a wsadmin script, however, will not save the configuration (the wsadmin script needs to do that).

Executing a job to a deployment manager does not cause node synchronization to occur. Synchronization will happen at the next automatic synchronization interval, or a wsadmin script can be submitted to synchronize.

5.3.2 Distributing files using the job manager

Some job types require that files be transferred to the node where the job will be run. The Distribute file job type can be used to transfer these files.

This is normally necessary in the following circumstances:

- ▶ When you want to run a wsadmin script on the node. The script must be distributed to the node before you can use the Run wsadmin script job.
- ▶ When you want to install or update an application. The EAR file must be distributed to the node before you can use the Install application or Update application jobs.

The following steps illustrate how to distribute a file to a node for use in later jobs. This example distributes a wsadmin script file to an admin agent:

1. The file to be distributed from the job manager must be in the /config/temp/JobManager directory of the job manager profile.

Create the `jobmgr_profile_root/config/temp/JobManager` directory and copy the file into it.

If you are developing a script or application in Rational Application Developer, you can export the file directly to the directory.

2. The distribute file job stores the file into the downloadedContent directory of the administrative agent or deployment manager profile. The destination parameter is relative to the downloadedContent directory. You must create this directory on the admin agent or deployment manager:
 - `adminagent_profile_home/downloadedContent`
 - `dmgr_profile_root/downloadedContent`
3. In the Job manager console, select the **Job → Submit** menu. This will launch the Job properties wizard.
 - a. Select **Distribute file** as the job type and click **Next**.
 - b. Enter the script file location on the job manager and the location to store the script file on the target node.

In this example, the `applInstall.py` script was stored in the following location:

`jobmgr_profile_root/config/temp/JobManager/applInstall.py`

On the admin agent it will be stored as:

`adminagent_profile_home/downloadedContent/applInstall.py`

The arguments are entered as shown in Figure 5-30.

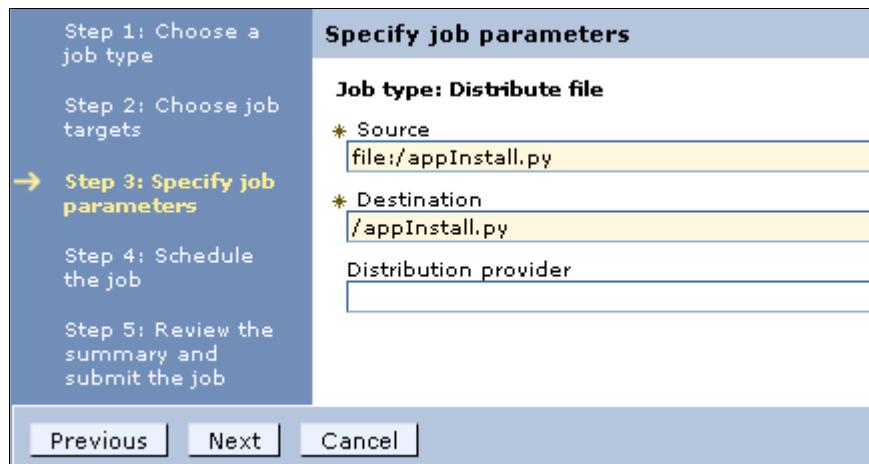


Figure 5-30 File distribution parameters

Click **Next**.

- c. Take the defaults for the job schedule. The defaults will execute the distribute file job once. Click **Next**.
- d. Click **Finish**. Monitor the status of the job and ensure it completes successfully.

5.4 Using command line tools

WebSphere Application Server provides commands that can be run from a command line. These commands can be used for many administrative tasks, for example, to start, view, or stop a WebSphere process. Many commands have an equivalent GUI interface, either created specifically for the command, through an administrative console, or through a First Steps console. However, it is often convenient to simply enter these commands manually from a command line.

Examples of commands are:

- ▶ `startServer`
- ▶ `stopServer`
- ▶ `serverStatus`
- ▶ `manageprofiles`
- ▶ `addNode`

5.4.1 Command location

Command line tools must be run on the system where the process you are entering the command for resides. For the most part, the commands exist in two places:

- ▶ *install_root/bin*
Commands entered from this location will operate against the default profile unless you use the `-profileName` parameter to specify the profile.
- ▶ *profile_root/bin*
Commands entered from this location will operate against the profile defined in *profile_root*.

5.4.2 Key usage parameters

The commands are consistent across platforms, though how you enter them, case sensitivity, and the extension will vary.

Note: Parameter values that specify a server name, a node name or a cell name are always case sensitive regardless of operating system.

There are several commonly used parameters that are valid for every command you should be aware of.

- ▶ **-profileName** specifies the profile the command is to run against.
- ▶ **-username** specifies the user ID with the administrative privileges required to execute the command
- ▶ **-password** specifies the password for the user ID specified in -username
- ▶ **-help** will display the usage requirements and a list of parameters for the command.

5.4.3 Entering commands

In this section, we show how to enter commands on the various operating systems.

Commands in the book: In the remainder of this book, the commands used were entered on Windows operating systems in our test labs and reflect that format. If you are using another platform, note that you need to adjust the examples for your platform.

Windows operating systems

Commands in Windows operating systems have an extension of .bat. It is not necessary to use the extension. Commands are not case sensitive, though parameters and names are case sensitive.

To use a command:

1. Open a Command Prompt window.
2. Change to the directory where the command is.

For example:

C:\Program Files\IBM\WebSphere\AppServer\profiles\profile_name

3. Enter the command. For example:

```
serverStatus -all -username admin -password admin
```

Note: When running command line tools on the Microsoft® Windows Vista operating system and Windows Server 2008: On the Windows Vista operating system and Windows Server 2008, you can install WebSphere Application Server as either Administrator or non-Administrator. When it is installed as Administrator, certain operations (such as those involving Windows Services) require Administrator privileges.

In order to ensure that WebSphere Application Server command line tools have sufficient privileges, run them as Administrator. When you run these command line tools from a Command Prompt, run them from a Command Prompt window that is launched by performing the following actions:

1. Right-click a Command Prompt shortcut.
2. Click **Run As Administrator**.
3. When you open the Command Prompt window as Administrator, an operating-system dialog appears that asks you if you want to continue. Click **Continue** to proceed.

If you are using a Windows Server Core installation of Windows Server 2008, any WebSphere Application Server commands that require a graphical interface are not supported, because a Windows Server Core system does not have a graphical user interface. Therefore, commands such as pmt.bat or ifgui.bat are not supported on that type of Windows Server 2008 installation.

UNIX operating systems

Commands in UNIX operating systems have an extension of .sh and are case sensitive.

To use a command:

1. Open a Command Prompt window.
2. Change to the directory where the command is.

For example, for root users the directory would be:

- (AIX) /usr/IBM/WebSphere/AppServer/profiles/*profile_name*/bin
- (HP, Linux, Solaris)
/opt/IBM/WebSphere/AppServer/profiles/*profile_name*/bin

For non-root users: *user_home*/IBM/WebSphere/AppServer/profiles/bin

3. Enter the command. For example:

```
serverStatus.sh -all -username admin -password admin
```

i5/OS operating systems

For an i5/OS system, proceed as follows:

1. From the i5/OS command line, start a Qshell session by issuing the **STRQSH CL** command.
2. Change to the directory where the command is.

For example:

```
cd /QIBM/UserData/WebSphere/AppServer/V7/ND/profiles/profilename/bin
```

3. Enter the command. For example:

```
serverStatus -all -username admin -password admin
```

z/OS operating systems

You can manage application servers on an z/OS system from a UNIX System Services environment as follows:

1. Enter **uss** (to switch to the UNIX System Services environment)
2. Change to the directory where the command is.

On z/OS, this will be always be *app_server_root*/profiles/default because only the profile name “default” is used in WebSphere Application Server for z/OS.

3. Enter the command. For example:

```
startServer.sh server1 -user username - password password
```



Administration of WebSphere processes

In this chapter, we provide information about basic administration tasks. The focus of this chapter is on managing WebSphere processes, including the deployment manager, nodes and node agents, application servers, and application server clusters.

We cover the following topics:

- ▶ “Working with the deployment manager” on page 290
- ▶ “Starting and stopping an administrative agent” on page 296
- ▶ “Starting and stopping the job manager” on page 297
- ▶ “Working with application servers” on page 297
- ▶ “Working with nodes in a distributed environment” on page 325
- ▶ “Working with clusters” on page 337
- ▶ “Working with virtual hosts” on page 350
- ▶ “Managing your configuration files” on page 353
- ▶ “Managing applications” on page 357

6.1 Working with the deployment manager

In this section we provide information about how to manage the deployment manager and introduce you to the configuration settings associated with it.

6.1.1 Deployment manager configuration settings

A deployment manager is built by creating a deployment manager profile. After it is built, there is usually not much that you need to do. However, it is good to note that there are settings that you can modify from the administration tools.

To view the deployment manager from the administrative console, select **System Administration Deployment manager**. You have two pages available, the Runtime page and the Configuration page. Figure 6-1 shows the Configuration page.

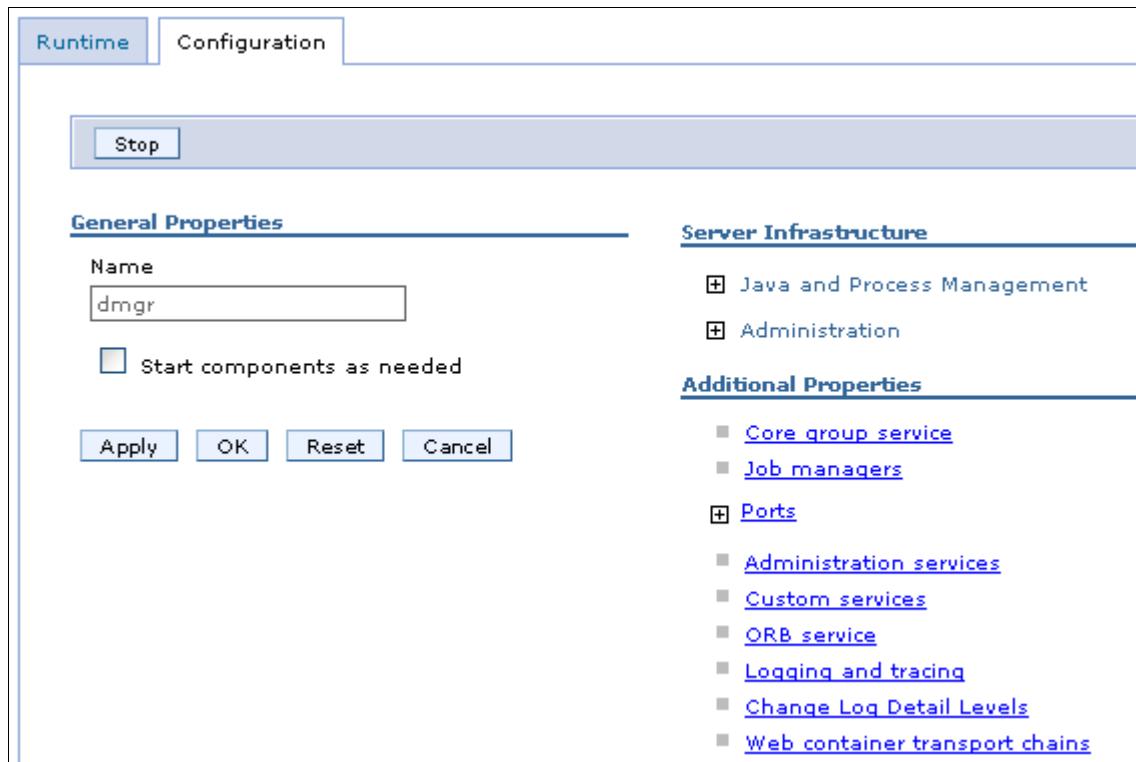


Figure 6-1 Deployment manager configuration

Configuration tab

Two options on this panel to note are:

- ▶ **Job managers (New in v7)**

Click this link to view work with job managers. You can view the job managers this deployment manager is registered to and you can register or unregister the deployment manager with a job manager.

- ▶ **Ports**

Select this link to view and manage the ports used by the deployment manager. This is useful for diagnostics, for example, when the deployment manager does not start because of a port conflict with another process. It is also useful for finding the SOAP connector port required when federating a custom profile.

Deployment manager Runtime tab

In addition to the Configuration page, the administrative console contains a Runtime page for the deployment manager. To view the Runtime page, select **System Administration Deployment manager** and click the **Runtime** tab at the top of the page. Figure 6-2 shows the Runtime tab.

The screenshot shows the 'Deployment manager' runtime page. The top navigation bar includes the title 'Deployment manager' and a sub-header 'Deployment manager'. Below the sub-header is a navigation bar with tabs: 'Runtime' (selected) and 'Configuration'. The main content area is organized into several sections:

- General Properties:** Contains fields for Process ID (5928), Cell name (Cell01), Node name (Dmgr02Node), and State (Started).
- Troubleshooting:** Contains a section for 'Diagnostic Provider service' with links to 'Tests', 'State Data', and 'Configuration Data'.
- Additional Properties:** Contains a link to 'Product Information'.

At the bottom left of the content area is a 'Back' button.

Figure 6-2 Deployment manager runtime page

The fact that the state is Started does not mean much, because you would not be able to access the administrative console otherwise. Items on this panel to note are:

► **Diagnostic Provider service**

These options allow you to query components for current configuration data, state data, and to run a self-diagnostic test routine. You would most likely use these options at the request of IBM support.

► **Product information (New in v7)**

Selecting this link opens a new page that provides information about the level of code running on the deployment manager system and provides links for more detailed information, including the installation history for the product and maintenance.

The product information is stored as XML files in *install_root/properties/version/dtd* folder and can be viewed without the administrative console.

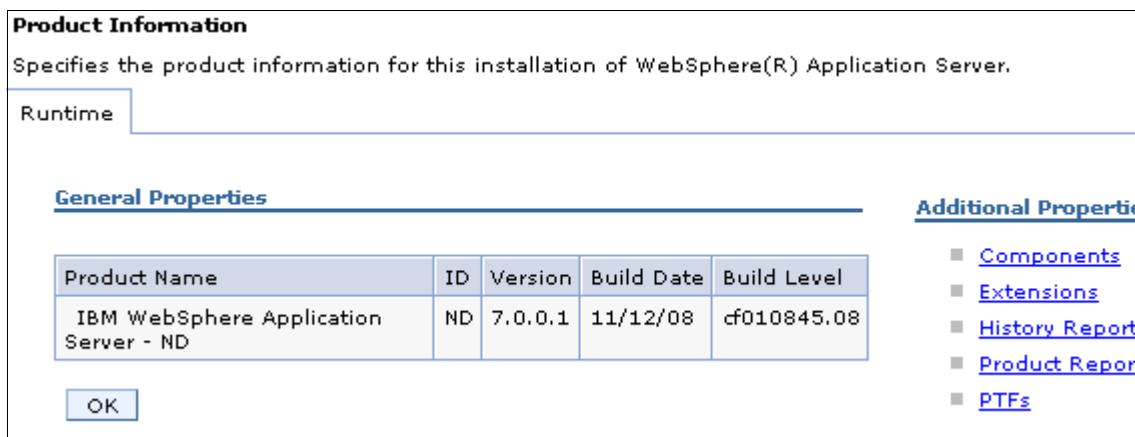


Figure 6-3 Product information

6.1.2 Starting and stopping the deployment manager

The deployment manager must be started and stopped with commands. The administrative console is not available unless it is running.

Starting the deployment manager with startManager

The `startManager` command is used to start the deployment manager on distributed systems, as shown in Example 6-1.

Example 6-1 startManager command

```
C:\WebSphereV7\AppServer\profiles\dmgr01\bin>startManager
ADMU7701I: Because dmgr is registered to run as a Windows Service, the
request to start this server will be completed by starting the
associated Windows Service.
```

```
ADMU0116I: Tool information is being logged in file
```

```
C:\WebSphereV7\AppServer\profiles\dmgr01\logs\dmgr\startServer.log
ADMU0128I: Starting tool with the Dmgr01 profile
ADMU3100I: Reading configuration for server: dmgr
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server dmgr open for e-business; process id is 1536
```

Run this command from the deployment manager *profile_root/bin* directory. If you run it from the *install_root/bin* directory, use the *-profileName* parameter to ensure that the command is run against the deployment manager profile.

Syntax of startManager

The syntax of the startManager command is:

```
startManager.bat(sh) [options]
```

The options are shown in Example 6-2.

Example 6-2 startManager options

```
Usage: startManager [options]
      options: -nowait
                -quiet
                -logfile <filename>
                -replacelog
                -trace
                -script [<script filename>] [-background]
                -timeout <seconds>
                -statusport <portnumber>
                -profileName <profile>
                -recovery
                -help
```

All arguments are optional. For more information about these options, see:

- ▶ *startManager command*

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

Starting the deployment manager on z/OS (START command)

On z/OS, the deployment manager can be started using a JCL start procedure. The exact command can be found in the BBOCCINS instruction member of the JCL generated to create the profile.

For example:

```
START WPDCR,JOBNAM=WPDMGR,ENV=WPCELL.WPDMNODE.WPDMGR
```

Where:

- ▶ WPDCR is the JCL start procedure.
- ▶ WPDMGR is the Job name.
- ▶ ENV is the concatenation of the cell short name, node short name, and server short name.

Starting the deployment manager will start the following components:

- ▶ A daemon. In our example, named WPDEMN. There will be one daemon per cell per MVS image. One of the functions of the daemon server is to provide the “location name service” for the cell. All daemons in the cell are fully aware of all the objects in the cell and use the same port values.
- ▶ A controller region. In our example, named WPDMGR. The controller region serves many functions, including acting as the endpoint for communications.
- ▶ A servant region. In our example, named WPDMGRS. The servant region contains the JVM where the applications are run.
- ▶ If you are using messaging, you will also see a control region adjunct server start.

Stopping the deployment manager

The deployment manager is stopped with the **stopManager** command, as shown in Example 6-3.

Example 6-3 stopManager command

```
C:\WebSphereV7\AppServer\profiles\dmgr01\bin>stopManager
```

ADMU7702I: Because dmgr is registered to run as a Windows Service, the request to stop this server will be completed by stopping the associated Windows Service.

ADMU0116I: Tool information is being logged in file

```
C:\WebSphere\AppServer\profiles\dmgr01\logs\dmgr\stopServer.log
```

ADMU0128I: Starting tool with the Dmgr01 profile

ADMU3100I: Reading configuration for server: dmgr

ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server dmgr stop completed.

Syntax of *stopManager*

The syntax of the *stopManager* command is:

```
stopManager.bat(sh) [options]
```

The options are shown in Example 6-2.

Example 6-4 startManager options

```
Usage: stopManager [options]
      options: -nowait
                -quiet
                -logfile <filename>
                -replacelog
                -trace
                -timeout <seconds>
                -statusport <portnumber>
                -conntype <connector type>
                -port <portnumber>
                -username <username>
                -password <password>
                -profileName <profile>
                -help
```

All arguments are optional. See:

- ▶ *stopManager command*

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

Stopping the deployment manager on z/OS (STOP command)

To stop the deployment manager with a STOP command, use the following format:

```
STOP dmgr_job
```

For example:

```
STOP WPDMGR
```

Stopping the daemon server will stop all servers for that cell, and all the servers on that daemon instance's MVS image will be stopped in an order fashion.

For example:

```
STOP WPDEMN
```

Windows start menu and services

On a Windows system, you have the option of starting and stopping the deployment manager using the Start menu. For example:

- ▶ **Start → All Programs → IBM WebSphere → Application Server Network Deployment V7.0 → Profiles → *profile_root* → Start the deployment manager**

Also, on a Windows system, you have the option of registering the deployment manager as a Windows service. In order to have it registered, you must select this option when you create the deployment manager profile or register it later using the **WASService** command.

For more information about the **WASService** command, see:

- ▶ *WASService command*

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

If the deployment manager is registered as a Windows service, all other options for starting the dmgr process are unchanged from the administrator's point of view, however the command used will start or stop the process through the service. In addition, you have the option to allow the service to be started automatically when the operating system starts.

6.2 Starting and stopping an administrative agent

An administrative agent process is managed in the same manner as an application server process. The process name is **adminagent**.

To view the status of the administrative agent process, use the following command:

```
serverStatus -adminagent
```

To start an administrative agent:

```
startServer adminagent
```

To stop an administrative agent:

```
stopServer adminagent
```

6.3 Starting and stopping the job manager

A job manager process is managed in the same manner as an application server process. The process name is **jobmgr**.

To view the status of the **jobmgr**, use the following command:

```
serverStatus -jobmgr
```

To start a job manager:

```
startServer jobmgr
```

To stop a job manager:

```
stopServer jobmgr
```

6.4 Working with application servers

This section covers the following topics:

- ▶ Creating an application server
- ▶ Viewing the status of an application server
- ▶ Starting an application server
- ▶ Stopping an application server
- ▶ Viewing runtime attributes of an application server
- ▶ Customizing application servers

Terminology—application server types:

- ▶ A *stand-alone application server* is created with an application server profile and is not federated to a cell or registered with an administrative agent. A stand-alone application server hosts its own administrative services and operates independently from other WebSphere processes. It cannot participate in application server clusters. Each stand-alone application server has its own node.

This option is available on all WebSphere Application Server packages, but is the only option in the Base and Express environments.

- ▶ An *unfederated application server (**New in V7**)* is an application server that resides on a node managed from an administrative agent. Unfederated application servers have the characteristics of a stand-alone application server, in that they cannot be used in a cluster. However, multiple application servers can exist on one node.

This option is available on all WebSphere Application Server packages.

- ▶ A *managed application server* is one that resides on a node that is managed from a deployment manager. A managed server can either be an application server that was created using an application server profile and subsequently federated to the cell, or it can be created directly from the deployment manager's administrative console.

Managed application servers can be clustered for high-availability and workload balancing.

This is only possible with the Network Deployment package.

6.4.1 Creating an application server

The process to create an application server depends on your WebSphere Application Server package.

Stand-alone application servers

Stand-alone application servers are created by creating an application server profile. This results in a profile that defines one stand-alone application server. This application server hosts the sample applications and the administrative console application.

(New in V7) In previous versions, a standalone server was always named server1. With V7 you have the opportunity to give the server a different name during profile creation.

For information about creating an application server profile, see:

- ▶ 2.3.3, “Creating an application server profile” on page 62.

Unfederated application server

(New in V7) An administrative agent can monitor and control multiple unfederated application servers on one or more nodes.

Unfederated application servers can be created in multiple ways:

- ▶ The first server on a node to be managed by an administrative agent must be created with a stand-alone profile and then registered with the administrative agent.

The registration process disables the administrative console on the server and makes a console for the application server node available on the administrative agent process.

See 2.3.10, “Registering nodes to an administrative agent” on page 95 for more information.

- ▶ Additional unfederated application servers on that node are created from the administrative agent. See “Creating an application server from the administrative console” on page 301.
- ▶ When you use the administrative agent console to register the application server node to a job manager, additional application servers can be created on the node by submitting a job from the job manager console.

What about wsadmin?

The administrative service remains active in unfederated application servers that are registered to an administrative agent. You can connect to either the application server or the administrative agent to run wsadmin commands, but all admin operations performed by connecting to the application server are forwarded to the agent. So you should ideally connect to the agent to avoid that extra hop.

Managed application servers

In a distributed server environment, you create an application server from the deployment manager administrative console. See “Creating an application server from the administrative console” on page 301.

If you are creating an application server with the intention of adding it to a cluster, use the **Servers → Clusters → WebSphere application server cluster** option. See 6.6, “Working with clusters” on page 337.

Application server options

You need to consider certain options as you create an application server. The method by which you select these options varies depending on how you are creating the server, but the values are the same.

Templates

An application server is created based on a template that defines the configuration settings. Four template options are provided:

- ▶ default: Standard production server. You get this option if you do not specify a template for a server on a distributed system.
 - ▶ defaultZOS: This option is available only on z/OS platforms and is the only option until you create new templates.
- DeveloperServer: The DeveloperServer template is used when setting up a server for development use. This template will configure a JVM for a quick start-up by disabling bytecode verification, and performing JIT compilations with a lower optimization level. This option should not be used on a production server, where long run throughput is more important than early server startup.
- ▶ Custom template: You can create templates based on existing application servers. (see “(Optional) Creating an application server template” on page 306).

Ports

Each server process uses a set of ports that must be unique on the system. When you create an application server, you have the following options:

- ▶ Use the default ports:

Use this selection if you will only have one application server on the system or if this is the first application server created and port selection is not an issue.
- ▶ Have a set of ports selected that are unique to the WebSphere system installation:

This selection ensures that no two WebSphere processes in the installation will have the same port assigned. It does not guarantee that ports will be selected that are not in use by non-WebSphere processes, or by WebSphere processes installed as a separate installation.
- ▶ Specify the ports:

This option is best if you have a convention for port assignment on your system that ensures unique ports are used by all processes, both WebSphere and non-WebSphere.

This option is only available when you create a new profile using the Advanced option or the `manageprofiles` command.

If you find that you have a port conflict, you can change the ports after the application server is created.

z/OS settings

Next we describe the various z/OS settings:

- ▶ Long name:

The long name of an application server is the name used in scripting and the administrative console. Long names can be up to 50 characters long and include mixed-case alphabetic characters, numeric characters, and the following special characters: ! ^ () _ - . { } []

- ▶ Short name:

Short names are specific to the z/OS implementation of WebSphere Application Server and are the principal names by which cells, nodes, servers, and clusters are known to z/OS.

- ▶ Specific short name (z/OS):

The short name is also used as the JOBNAME for the server. If you do not specify a value for the short name field, the short name defaults to BBOSnnn, where nnn is the first free number in the cell that can be used to create a unique short name. Make sure that you set up a RACF SERVER class profile that includes this short name.

- ▶ Generic short name (z/OS):

Nodes that have not been clustered have a server generic short name, also called a *cluster transition name*. When a cluster is created from an existing application server, the server's generic short name becomes the cluster name.

No two servers on the same z/OS system should have the same server generic short name unless they are in the same cluster.

If you do not specify a value for the generic short name field, the generic short name defaults to BBOCnnn, where nnn is the first free number in the cell that can be used to create a unique generic short name.

- ▶ Bit mode (z/OS):

The default setting is that the application server runs in 64-bit mode, however, you can elect to run in 31-bit mode. Note that 31-bit mode is deprecated.

Creating an application server from the administrative console

To create an application server from the administrative console:

1. Open the deployment manager administrative console.
2. Select **Servers** → **Server types** → **WebSphere application server**.

3. Click **New**.
4. Select the node for the new server and enter a name for the new server (Figure 6-4).

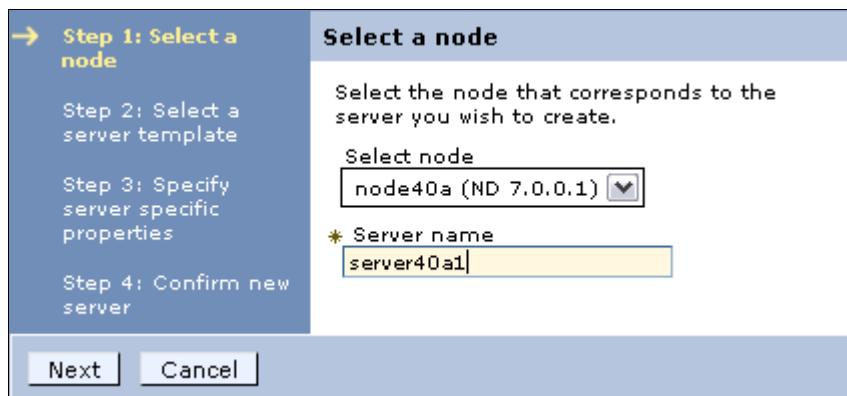


Figure 6-4 Create an application server: Step 2

Click **Next**.

5. Select a template to use by clicking the appropriate radio button (Figure 6-5).

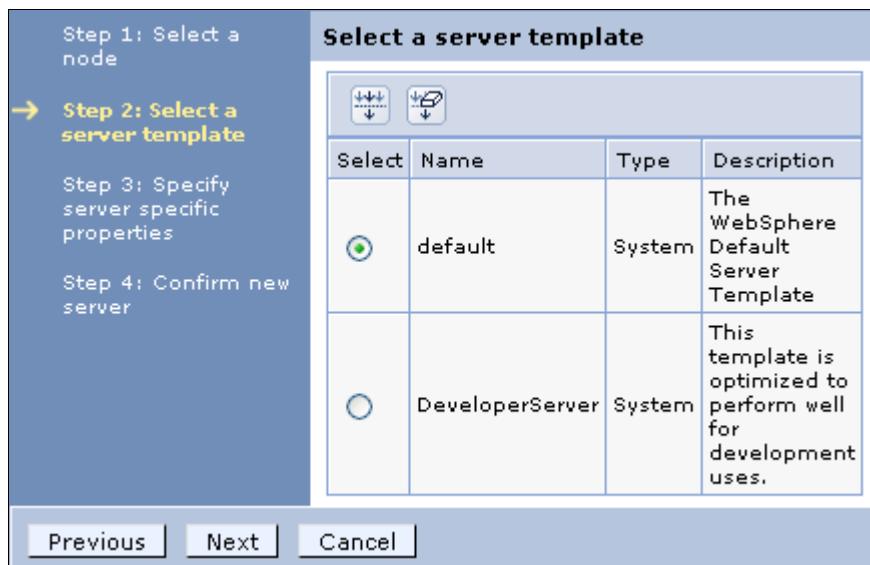


Figure 6-5 Create an application server: Step 2

On z/OS systems, there is one system defined template called defaultZOS.

Click **Next**.

6. The options you see on the next window vary depending on the platform.

For distributed platforms, you see a panel as shown in Figure 6-6.

Check the **Generate Unique Ports** box to have unique ports generated for this server. Deselecting this option will generate the default set of ports.

If you have multiple core groups defined, you have the option to select the core group.

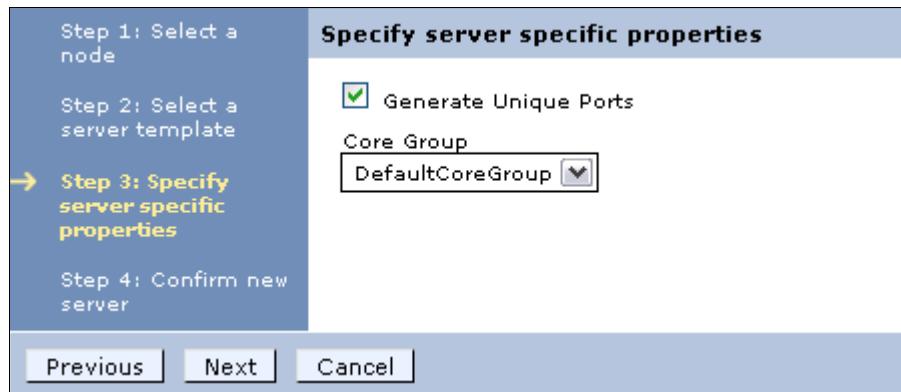


Figure 6-6 Create an application server: Step 3 for distributed systems

For z/OS systems, you see a panel as shown in Figure 6-7.

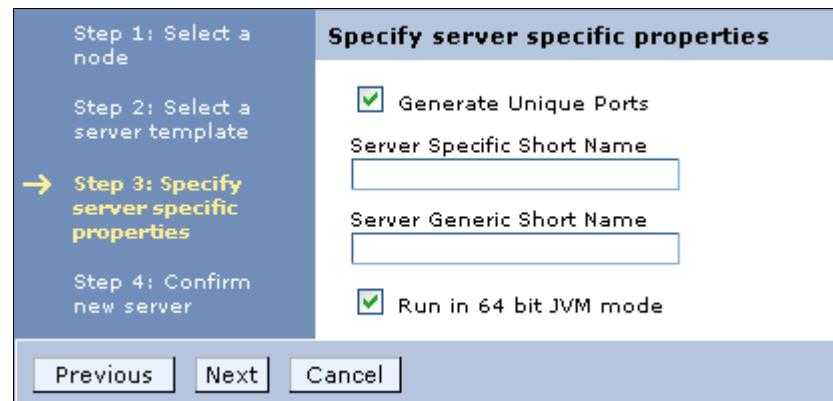


Figure 6-7 Create an application server: Step 3 for z/OS

The server specific short name specifies the short name for the server. This is also used as the job name (for example, BBOS002). The generic short name is the short name that is converted to a cluster short name if the server is later used in a cluster.

Click **Next**.

7. A summary window is presented with the options you chose. Click **Finish** to create the server.
8. In the messages box, click **Save** to save the changes to the master repository.
9. Review and update the virtual host settings (see “Update the virtual host settings” on page 306)
10. Regenerate the Web server plug-in and propagate it to the Web server.

Managing Web server plug-ins is discussed in the Information Center. See the topic, *Communicating with Web server*, at:

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

Note: If you are creating an application server on a Windows operating system, this process does not give you the option of registering the new server as a Windows service. You can do this later with the **WASService** command (see 6.10, “Enabling process restart on failure” on page 375)

Creating an application server from the job manager

To create an application server from the job manager, make the following selections as you step through the process to submit the job:

1. Step 1: Select the **Create application server** job type.
2. Step 2: Select the job target:
 - If you are creating an application server on an unfederated node, select the application server node.
 - If you are creating a new managed server, select the deployment manager node.

Enter the user ID and password with administrative authority on the target node.

3. Step 3: Specify the job parameters.

At minimum:

- Specify the name of the new application server.
- If the target is a deployment manager, enter the name of the node that the server will be created on.

Step 1: Choose a job type

Step 2: Choose job targets

Step 3: Specify job parameters

Step 4: Schedule the job

Step 5: Review the summary and submit the job

Specify job parameters

Job type: Create application server

* Server name

Node name

Additional job parameters.

Server template

Template name

Template location

Port control

Generate unique ports

Platform specific

Specific short name

Generic short name

Bit mode

Previous **Next** **Cancel**

Figure 6-8 Specify the options for the new server

The optional settings allow you to add platform specific settings, specify a different template, and specify the setting that determines if ports unique to the installation are generated.

- The template name field will default to the production server template for the operating system on which the application server will run. You only need to specify this setting if you want to use a custom template or the DeveloperServer template.
- Generate unique ports:
 - Specify true to generate ports unique to that installation (the default)
 - Specify false to use the default port numbers.

4. Review and update the virtual host settings (see “Update the virtual host settings” on page 306)
5. Regenerate the Web server plug-in and propagate it to the Web server.

Managing Web server plug-ins is discussed in the Information Center.

See the topic, *Communicating with Web servers*, at:

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

Update the virtual host settings

When you install applications, you associate a virtual host with each Web module.

When you create a new application server, the default_host virtual host is set as the default virtual host for Web modules installed on the server. You can change this default in the Web container settings for the application server or simply select a new virtual host when you install the applications.

If the application will only be accessed via a Web server, and the virtual host that you will use has been set up with the Web server port in the list of host aliases, then no action is necessary.

However, if application clients will access the Web container directly, or if you will be installing SIP applications on this server, you need to ensure that the relevant ports generated for this application server are added in the host alias list. See 6.7, “Working with virtual hosts” on page 350 for more information.

(Optional) Creating an application server template

To create an application server template based on an existing server:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. Click **Templates...** at the top of the server list.
3. Click **New**.
4. Select a server from the list to build the template from and click **OK**.
5. Enter a name and description for the template and click **OK**.
6. Save your configuration.

The new template will be in the list of templates and will be available to select, the next time you create an application server.

6.4.2 Viewing the status of an application server

There are multiple ways to check the status of an application server.

- ▶ Use the `serverStatus` command on the system where the application server is running.
- ▶ In a distributed environment, you can view the status from the administrative console. The node for the application server must be active for the deployment manager to know the status of a server on that node.
- ▶ From the job manager
- ▶ If the server is registered as a Windows service, you can check the status of the service.

Using the administrative console

To check the status of a managed server using the deployment manager's administrative console, the node agent must be started. To use the administrative console, do the following steps:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. The servers are listed. The last column to the right contains an icon to indicate the status of each server. Figure 6-9 shows the icons and the corresponding status.

	Started	The server is running.
	Partially stopped	The server is in the process of changing from a started state to a stopped state.
	Stopped	The server is not running.
	Unknown	The server status cannot be determined.

Figure 6-9 Status icons

Note: If the server status is Unknown, the node agent on the node in which the application server is installed is not active. The server cannot be managed from the administrative console unless its node agent is active.

Using the `serverStatus` command

The syntax of the `serverStatus` command in Example 6-5.

Example 6-5 serverStatus options

```
Usage: serverStatus <server name | -all>
[-logfile <filename>]
[-replacelog]
```

```
[-trace]
[-username <username>]
[-password <password>]
[-profileName <profile>]
[-help]
```

The first argument is mandatory. The argument is either the name of the server for which status is desired, or the `-all` keyword, which requests status for all servers defined on the node.

If you have administrative security enabled, you will need to enter a user ID and password of an administrator ID. If you do not include it in the command, you will be prompted for it.

For example, to view the status of a server, type:

```
cd profile_home/bin
serverStatus.sh server_name -username adminID -password adminpw
```

To check the status of all servers on the node, type:

```
cd profile_home/bin
serverStatus.sh -all -username adminID -password adminpw
```

For more information about this command, see:

- ▶ *serverStatus command*

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

Example 6-6 shows an example of using the `serverStatus` command.

Example 6-6 serverStatus example - Windows operating system

```
C:\WebSphereV7\AppServer\profiles\node40a\bin>serverstatus -all
-username admin -password adminpw
ADMU0116I: Tool information is being logged in file

C:\WebSphereV7\AppServer\profiles\node40a\logs\serverStatus.log
ADMU0128I: Starting tool with the node40a profile
ADMU0503I: Retrieving server status for all servers
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: nodeagent
ADMU0506I: Server name: server40a1
ADMU0508I: The Node Agent "nodeagent" is STARTED
ADMU0509I: The Application Server "server40a1" cannot be reached. It
appears to be stopped.
```

From the job manager console

To display the servers and their status from the job manager console:

1. Select **Jobs → nodes**.
2. Click the box to the left of the node name. In the **Display resources** drop-down box, select **Server** (Figure 6-10).

Display Resources ▾		
Select	Node name ▾	Version ▾
<input type="checkbox"/>	SAsrv40Node	ND 7.0.0.1
<input type="checkbox"/>	dmqr01node	ND 7.0.0.1
<input checked="" type="checkbox"/>	dmqr40node	ND 7.0.0.1

Total 3

Figure 6-10 Display the servers on a node

3. This action displays the list of servers in the node. Click the name of the server (Figure 6-11).

Resources ▾	Number ▾	Node name ▾
node/dmqr40node/server/dmqr	1	dmqr40node
node/node40a/server/AS1	1	dmqr40node
node/node40b/server/AS2	1	dmqr40node
Total 3		

Figure 6-11 Display the servers on a node

The status of the server is displayed (Figure 6-12).

Resource ID		
Resource ID	Node name	Status
node/node40a/server/AS1	dmgr40node	Stopped
Total 1		

Figure 6-12 Display the servers on a node

6.4.3 Starting an application server

How you start an application server depends largely on personal preference and on whether the application server is stand-alone or managed. This section discusses how to start individual application servers. You can also start all application servers in a cluster by starting the cluster (see 6.6.3, “Managing clusters” on page 349).

Using the administrative console to start a managed server

Tip: Before managing a server in a distributed server environment using the administrative console, the node agent for the server’s node must be running. To check the status of the node:

1. Select **System Administration Node Agents**.
2. The status of the node agent is in the far right column. If it is not started, you must start it (see 6.5.1, “Starting and stopping nodes” on page 326).

From the administrative console, do the following steps:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. Check the box to the left of each server you want to start.
3. Click **Start**.
4. Verify the results in the Server status feedback window.

If there are any errors, check the log files for the application server process. The default location for the logs is:

- ▶ *profile_home/logs/server_name/SystemOut.log*
- ▶ *profile_home/logs/server_name/startServer.log*

On z/OS, check the output in application server job log.

Tip: By default, all the applications on a server start when the application server starts. To prevent an application from starting, see 6.9.6, “Preventing an enterprise application from starting on a server” on page 368.

Using the `startServer` command

The syntax of the `startServer` command is shown in Example 6-7.

Example 6-7 startServer options

```
Usage: startServer <server> [options]
      options: -nowait
                -quiet
                -logfile <filename>
                -replacelog
                -trace
                -script [<script filename>] [-background]
                -timeout <seconds>
                -statusport <portnumber>
                -profileName <profile>
                -recovery
                -help
```

<server> is the name of the server to be started. The first argument is mandatory and case sensitive.

See the following article for more information about this command.

- ▶ *startServer command*

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

startServer example

Example 6-8 shows an example of using the `startServer` command. Note that the user ID and password are not required to start the server.

Example 6-8 startServer example

```
C:\WebSphereV7\AppServer\profiles\node40a\bin>startserver server40a1
```

```
ADMU0116I: Tool information is being logged in file
```

```
C:\WebSphereV7\AppServer\profiles\node40a\logs\server40a1\startServer
.log
```

```
ADMU0128I: Starting tool with the node40a profile
```

```
ADMU3100I: Reading configuration for server: server40a1
```

ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server40a1 open for e-business; process id is 3928

Starting a server from the job manager

To start an application server from the job manager, make the following selections to submit the job:

1. Step 1: Select the **Start server** job type.
 2. Step2: Select the job target.
 - If you are starting an application server on an unfederated node, select the application server node.
 - If you are starting a new managed server, select the deployment manager node.
- Enter the user ID and password with administrative authority on the target node.
3. Step 3: Specify the job parameters.
 - Specify the name of the new application server.
 - If the target is a deployment manager, enter the name of the node that the server will be created on.

6.4.4 Stopping an application server

This section shows multiple methods for stopping a server.

Using the administrative console to stop a managed server

Note: These directions assume that the node agent for the application server is running.

From the administrative console, you have the following options to stop an application server:

- ▶ The **Stop** button quiesces the application server and stops it. In-flight requests are allowed to complete.
- ▶ The **Immediate Stop** button stops the server, but bypasses the normal server quiesce process, which enables in-flight requests to complete before shutting down the entire server process. This shutdown mode is faster than the normal server stop processing, but some application clients can receive exceptions.

- ▶ The **Terminate** button deletes the application server process. Use this if immediate stop fails to stop the server.
- ▶ The **Restart** button stops, then starts the server.

From the administrative console, do these steps to stop an application server:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. Check the box to the left of each server you want to stop.
3. Click the appropriate stop option.

If there are any errors, check the log files for the application server process:

- ▶ *profile_home/logs/server_name/SystemOut.log*
- ▶ *profile_home/logs/server_name/stopServer.log*

On z/OS, check the output in application server job log.

Restarting all servers on a node

If you want to stop, and then restart, all the application servers on a node, you can do the following from the administrative console:

1. Select **System Administration Node Agents**.
2. Check the box to the left of the node agent.
3. Click **Restart all Servers on the Node**.

Stopping all servers in a cluster

If you want to stop all the servers in a cluster, you can do the following from the administrative console:

1. Select **Servers** → **Clusters** → **WebSphere application server clusters**.
2. Check the box to the left of the cluster.
3. Click **Stop** or **Immediate Stop**.

Restarting all servers in a cluster

If you want to stop, and then restart, all the servers in a cluster, you can do the following from the administrative console:

1. Select **Servers** → **Clusters** → **WebSphere application server clusters**.
2. Check the box to the left of the cluster.
3. Click **Ripplestart**.

Using the stopServer command

The syntax of the **stopServer** command is shown in:

Example 6-9 stopServer command

Usage: `stopServer <server> [options]`

```
options: -nowait  
        -quiet  
        -logfile <filename>  
        -replacelog  
        -trace  
        -timeout <seconds>  
        -statusport <portnumber>  
        -conntype <connector type>  
        -port <portnumber>  
        -username <username>  
        -password <password>  
        -profileName <profile>  
        -help
```

<server> is the name of the server to be started. The first argument is mandatory and is case sensitive.

See the following article for a list of available options.

► *stopServer command*

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

Table 6-10 shows an example of the **stopServer** command.

Example 6-10 stopServer command example

```
C:\WebSphereV7\AppServer\profiles\node40a\bin>stopserver server40a1  
-username admin -password adminpw
```

ADMU0116I: Tool information is being logged in file

```
C:\WebSphereV7\AppServer\profiles\node40a\logs\server40a1\stopServer.  
log
```

ADMU0128I: Starting tool with the node40a profile

ADMU3100I: Reading configuration for server: server40a1

ADMU3201I: Server stop request issued. Waiting for stop status.

ADMU4000I: Server server40a1 stop completed.

Note: If you attempt to stop a server and for some reason it does not stop, you can use the operating system commands to stop the Java process for the server.

6.4.5 Viewing runtime attributes of an application server

To view runtime attributes, do the following steps:

1. Select **Servers** → **Server Types** → **WebSphere application servers** to display the list of servers.
2. Click the server name to access the detail page.
3. If the server is running, you will see both a Configuration tab and Runtime tab. If it is not running, you will see only a Configuration tab. Click the **Runtime** tab. Figure 6-13 shows the Runtime tab and the information that it provides.

The screenshot shows the 'Application servers > server40a1' configuration page. At the top, there are two tabs: 'Runtime' (selected) and 'Configuration'. The main content area is divided into several sections:

- General Properties**:
 - Process ID: 4836
 - Cell name: Cell40
 - Node name: node40a
 - State: Started
- Server messaging**:
 - Messaging engines
- Troubleshooting**:
 - Diagnostic Provider service
- Additional Properties**:
 - Transaction Service
 - Performance Monitoring Infrastructure (PMI)
 - Product Information

A 'Back' button is located at the bottom left of the main content area.

Figure 6-13 Application server Runtime tab

4. From the Runtime tab, you have:
 - Access to a list of messaging engines that run on this application server. There will be one messaging engine for each bus that the server is a member of. You can start and stop the messaging engine from this panel.
 - Access to the Diagnostic Provider service, allowing you to query current configuration data, state, and to initiate diagnostic tests.
 - Access to the Transaction Service properties settings. You can change the timeout settings while the server is running, but not the transaction log directory setting. Changes made in this panel are active until the server is stopped.

To make these settings permanent and to configure additional Transaction Service settings, click the Configuration tab while you have this page open. This takes you directly to the Transaction Service settings.

You can also view or act on transactions in the following states by clicking **Review** to the right of the state. This action is not normally necessary, but in an exceptional situation, it might be useful:

- Manual transactions:

These transactions await administrative completion. For each transaction, the local or global ID is displayed. You can display each transaction resource and its associated resource manager. You can choose also to commit or rollback transactions in this state.

- Retry transactions:

These are transactions with some resources being retried. For each transaction, the local or global ID is displayed, and whether the transaction is committing or rolling back. You can display each transaction resource and its associated resource manager. You can choose also to finish, or abandon retrying, transactions in this state.

- Heuristic transactions:

These are transactions that have completed heuristically. For each transaction, the local or global ID and the heuristic outcome is displayed. You can display each transaction resource and its associated resource manager. You can also choose to clear the transaction from the list.

- Imported prepared transactions:

Transactions that have been imported and prepared but not yet committed. For each transaction, the local or global ID is displayed. You can display each transaction resource and its associated resource manager. You can also choose to commit or rollback transactions in this state.

- Performance Monitoring Service settings allow you to change the instrumentation levels while the server is running or make permanent changes to the configuration for that server.
- Product Information gives you access to extensive information about the product installation and Fix Pack information.

6.4.6 Customizing application servers

When you create a new application server, it inherits most of its configuration settings from the specified template server. To view or modify these settings, select **Servers** → **Server Types** → **WebSphere application servers**. A list of

application servers defined in the cell appears in the workspace. Click the name of the application server to make a modification.

This section gives you a quick overview of the types of settings that you can customize. See Figure 6-14 (not all settings are shown due to the size of the configuration window).

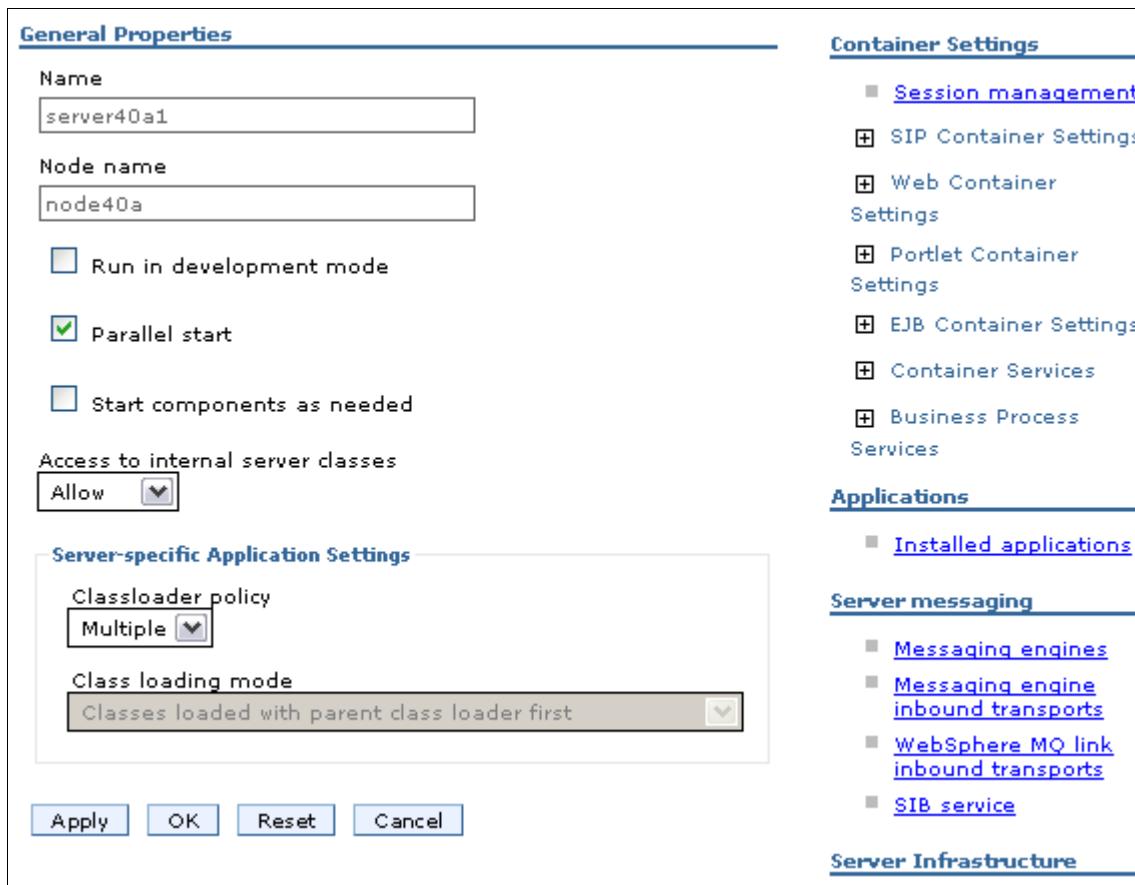


Figure 6-14 Application server configuration

General properties

The general properties consist of a few items that you can see immediately:

- ▶ Run in development mode: Enable this option to streamline the startup time of an application server. Do not enable this setting on production servers.
- ▶ Parallel start: Select this field to start the server components, services, and applications on multiple threads. This might shorten the startup time.

The order in which the applications start depends on the weights you assigned to each of them. Applications that have the same weight are started in parallel.

To set the weight of an application, in the administrative console, click **Applications** → **Application Types** → **WebSphere enterprise applications** → *application_name* → **Startup behavior**, and then specify an appropriate value in the **Startup order** field.

- ▶ Start components as needed: (**New in V7**) Select this field to start the application server components as they are needed. This might shorten the start time.
- ▶ Access to internal server classes: Specifies whether the applications can access many of the server implementation classes.
- ▶ Application classloader policy and class loading mode: These settings allow you to define an application server-specific classloader policy and class loading mode. Class loaders are discussed in Chapter 13, “Understanding class loaders” on page 677.

Container settings

Each application server has containers that run specific application components. This section in the configuration page for the server provides links to pages where you can modify the settings for the containers.

Tip: Modifying container settings is not something you would normally do on a daily basis. The most commonly used settings in these sections are discussed throughout this book in the appropriate topics. For now, it is enough to know that each container has settings that allow you to modify its configuration and how to find those settings.

SIP container settings

Session Initiation Protocol (SIP) support extends the application server to allow it to run SIP applications written to the JSR 116 specification. SIP is used to establish, modify, and terminate multimedia IP sessions including IP telephony, presence, and instant messaging. If you have SIP applications, review these settings.

Web container settings

The Web container serves application requests for servlets and JSPs. The Web container settings allow you to specify the default virtual host, enable servlet caching, specify session manager settings such as persistence and tuning parameters, and HTTP transport properties. See Figure 6-15.

(New in V7) The Asynchronous Request Dispatcher (ARD) enables servlets and JSPs to make standard include calls concurrently on separate threads. Selecting this link allows you to enable ARD and configure related settings.

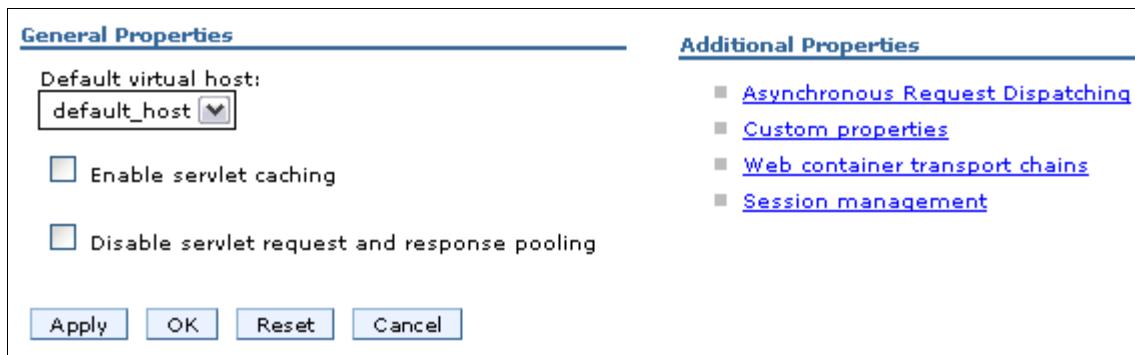


Figure 6-15 Web container settings

Portlet container services

The portlet container is the runtime environment for portlets using the JSR 168 Portlet Specification. Portlets based on this JSR 168 Portlet Specification are referred to as standard portlets. You can use these settings to enable portlet fragment caching to save the output of portlets to the dynamic cache.

EJB container properties

These properties allow you configure the services provided by the EJB container. This includes setting the passivation directory path, EJB cache and timer service settings, pool cleanup interval, a default data source JNDI name, and to enable stateful session bean failover using memory-to-memory replication. See Figure 6-16.

General Properties		Additional Properties
<p>* Passivation directory <input type="text" value="\${USER_INSTALL_ROOT}/temp"/></p> <p>Inactive pool cleanup interval <input type="text" value="30000"/> milliseconds</p> <p>Default data source JNDI name <input type="text" value="(none)"/></p> <p><input type="checkbox"/> Enable stateful session bean failover using memory-to-memory replication (No replication domains have been defined.)</p>		<ul style="list-style-type: none"> EJB cache settings EJB timer service settings
<input type="button" value="Apply"/> <input type="button" value="OK"/> <input type="button" value="Reset"/> <input type="button" value="Cancel"/>		

Figure 6-16 EJB container settings

Container services

The following settings are available under the container services section:

- ▶ Application profiling service: Application profiling is a WebSphere extension that used along with access intents allows you to define strategies to dynamically control concurrency, prefetch, and read-ahead. The container services settings allows you to enable this service and to set the compatibility mode for J2EE 1.3 applications.
- ▶ Transaction service: The transaction service properties allow you to specify settings for the transaction service, as well as manage active transaction locks. The settings include the directory location for the transaction service on the application server to store log files for recovery, the total transaction lifetime timeout, and client inactivity timeout.

When the application server is running, a Runtime tab is available in the Transaction Service properties workspace. From here, you can manage running transactions and modify timeout settings at runtime.

- ▶ Dynamic cache service: This page allows you to specify settings for the dynamic cache service of this server.
- ▶ (**New in V7**) The Java Persistence API (JPA) default settings: JPA provides a mechanism for managing persistence and object-relational mapping and functions for the EJB 3.0 specifications. This page allows you to configure default settings for JPA. JPA settings in an application will override these settings.
- ▶ Compensation service: The compensation service supports server-level configuration for compensation enablement and logging. This service is not

started automatically. If you will be running applications that require this service you must enable it here.

- ▶ Internationalization service: This service enables you to configure and manage an internationalization context for an application for which components are distributed across the enterprise. This section of the configuration panel allows you to enable this service. It is not enabled by default.
- ▶ Object pool service: The object pool service manages object pool resources that are used by the application server. This section of the configuration panel allows you to disable this service (it is enabled by default).
- ▶ Startup beans service: Startup beans are session beans that run business logic through the invocation of start and stop methods when applications start and stop. This section of the configuration panel allows you to enable this service (it is disabled by default).
- ▶ ORB service settings: These settings allow you to specify settings for the Object Request Broker service. These include request timeout, thread settings, and connection cache minimum and maximum.

Business process services

The business process settings allow you to manage the following features

- ▶ Activity session service
- ▶ Work area partition service
- ▶ Work area service

Applications

Use the **Installed Applications** link to view the applications installed on this server. This will display the collection of applications as links to the configuration page for each application.

Server messaging

The server messaging settings provide configuration settings and information for the messaging services.

Server infrastructure

The server infrastructure settings include settings for Java and process management and administration services:

- ▶ **Java and Process Management:**
 - Class loader: Create and configure class loader instances. Class loaders are discussed in Chapter 13, “Understanding class loaders” on page 677.
 - Process definition: Define runtime properties, such as the program to run, arguments to run the program, and the working directory. Within the process definitions, you will find the JVM definitions, such as the initial and maximum heap sizes, debug options, the process classpath, or different runtime options, such as profiler support and heap size.
 - Process execution: Include settings such as the process priority, or the user and group that should be used to run the process. These settings are not applicable on the Windows platform.
 - Monitoring policy: Determine how the node agent will monitor the application server. It includes ping intervals, timeouts, and an initial state setting. These can be used to ensure that the server is started when the node starts and will be restarted in the event of a failure.
- ▶ **Administration:**
 - Custom properties: Specifies additional custom properties for this component.
 - Administration services: This group of settings allows you to specify various settings for administration facility for this server, such as administrative communication protocol settings and timeouts. (These settings are not something you would normally be concerned with.)
 - Server components: Creates an additional runtime components that are configurable.
 - Custom Services: Creates custom service classes that run within this server and their configuration properties.
If you plan to extend the administration services by adding custom MBeans, see the topic, *Extending WebSphere Application Server Administrative System with custom MBeans*, in the Information Center.

Performance

These settings allow you to specify settings for the Performance Monitoring Infrastructure (PMI) and the Performance and Diagnostic Advisor framework.

Communications

The following communications settings are available:

- ▶ Ports:

These settings contain the basic port definitions for the server.

Tip: You might not ever need to manually change these ports. It is likely, however, that you will want to view these.

For example, if you use the **dumpNameSpace** command, you can specify the bootstrap port of the process to dump the name space from. When you federate a node, you will need to know the SOAP connector port of the node or deployment manager. And the inbound communications ports are essential for accessing applications and the administrative console.

Clicking **Details** takes you to a page that has links for the configurable port settings.

Some port settings will be defined to use the channel framework. These will have an associated transport chain that represents the network protocol stack. A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP or HTTP. Network ports can be shared among all of the channels within a chain. The Channel Framework function automatically distributes a request arriving on that port to the correct I/O protocol channel for processing.

Port Name	Port	Details
BOOTSTRAP_ADDRESS	9813	
SOAP_CONNECTOR_ADDRESS	8880	
ORB_LISTENER_ADDRESS	9100	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9401	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9403	
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9402	
WC_adminhost	9060	
WC_defaulthost	9080	
DCS_UNICAST_ADDRESS	9353	
WC_adminhost_secure	9043	
WC_defaulthost_secure	9443	
SIP_DEFAULTHOST	5060	
SIP_DEFAULTHOST_SECURE	5061	
SIB_ENDPOINT_ADDRESS	7276	
SIB_ENDPOINT_SECURE_ADDRESS	7286	
SIB_MQ_ENDPOINT_ADDRESS	5558	
SIB_MQ_ENDPOINT_SECURE_ADDRESS	5578	
IPC_CONNECTOR_ADDRESS	9637	

Figure 6-17 Viewing application server ports

- ▶ Message listener service:

The message listener service provides support for WebSphere Application Server V5 message-driven beans applications.

Security

Security settings for the application server allow you to set specific settings at the server level. Security settings are covered in *WebSphere Application Security V7 Security Handbook*, SG24-7660.

Troubleshooting

These settings include those for logging and tracing. For information about troubleshooting and using these settings, see *WebSphere Application Server V6: Diagnostic Data*, REDP-4085.

Additional properties

The following settings are defined under the additional properties section:

- ▶ Class loader viewer service: This is used to enable or disable the service that keeps track of classes that are loaded.
- ▶ Core group service: These settings are related to high availability.
- ▶ Endpoint listeners: An endpoint listener receives requests from service requester applications within a specific application server or cluster.
- ▶ Debugging service: On this page, you can specify settings for the debugging service, to be used in conjunction with a workspace debugging client application, for example, the Application Server Toolkit.
- ▶ Thread pool: The thread pool specifies the possible maximum number of concurrently running threads in the Web container. As one thread is needed for every client request, this directly relates to the number of active clients that can possibly access the Web container on this application server at any given time. A timeout value can be specified for the application server to remove threads from the pool based on a timed period of inactivity.

Finally, an option for creating threads beyond the maximum pool size is available. Be careful when using this option. It can have the unexpected effect of allowing the Web container to create more threads than the JVM might be able to process, creating a resource shortage and bringing the application server to a halt.

- ▶ (*New in V7*) Reliable messaging state: This link allows you to view and manage the WS-ReliableMessaging runtime state.
- ▶ Web server plug-in properties: This is used to change the HTTP plug-in configuration without having to stop the server and start it again.

6.5 Working with nodes in a distributed environment

Managing nodes is a concept specific to a Network Deployment environment. Nodes are managed by the deployment manager through a process known as a *node agent* that resides on each node. In order to manage a node in a Network Deployment environment, the node must be defined and the node agent on each WebSphere Application Server node must be started.

Nodes are created when you create a profile. Nodes are added to a cell through federation (see 2.3.7, “Federating nodes to a cell” on page 82).

6.5.1 Starting and stopping nodes

A node consists of the node agent and the servers. There are several ways to start and stop a node and node agent, or stop them individually. Before using any of these methods, be sure to note whether it affects the entire node, including servers, or just the node agent.

Starting a node agent

When a node agent is stopped, the deployment manager has no way to communicate with it. Therefore, the node agent has to be started with the **startNode** command run from on the profile node system.

startNode command

The command syntax is shown in Example 6-11.

Example 6-11 startNode command

```
Usage: startNode [options]
      options: -nowait
                -quiet
                -logfile <filename>
                -replacelog
                -trace
                -script [<script filename>] [-background]
                -timeout <seconds>
                -statusport <portnumber>
                -profileName <profile>
                -recovery
                -help
```

For information about the use of the **startNode** command, see:

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

See Example 6-12 for an example of the **startNode** command. Note that a user ID and password is not required.

Example 6-12 startNode command

```
C:\WebSphereV7\AppServer\profiles\node40c\bin>startnode
ADMU0116I: Tool information is being logged in file

C:\WebSphereV7\AppServer\profiles\node40c\logs\nodeagent\startServer.
log
ADMU0128I: Starting tool with the node40c profile
```

ADMU3100I: Reading configuration for server: nodeagent
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server nodeagent open for e-business; process id is 7480

Starting a node on z/OS using the START command

To start a node agent on z/OS using the START command, use the following format:

```
START nodeagent_procname,JOBNAM=server_shortname,  
ENV=cell_shortname.node_shortname.server_shortname
```

For example:

```
START WPACRA,JOBNAM=WPAGNTA,ENV=WPCELL.WPNODEA.WPAGNTA
```

Stopping a node agent

To stop the node agent and leave the servers running, do the following actions, depending on your preferred method.

From the administrative console, do the following steps:

1. From the administrative console, select **System Administration Node Agents**.
2. Check the box beside the node agent for the server and click **Stop**.

Or, from a command prompt, use the **stopNode** command.

Note: After you stop the node agent, the deployment manager has no way to communicate with the servers on that node. The servers might be up and running, but the administrative console is not able to determine their status.

stopNode command

The command syntax is shown in Example 6-13.

Example 6-13 The stopNode command

```
Usage: stopNode [options]  
       options: -nowait  
                 -stopservers [-saveNodeState]  
                 -quiet  
                 -logfile <filename>  
                 -replacetolog  
                 -trace  
                 -timeout <seconds>  
                 -statusport <portnumber>
```

```
-conntype <connector type>
-port <portnumber>
-username <username>
-password <password>
-profileName <profile>
-help
```

For information about the **stopNode** command and its options, see:

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

See Example 6-14 for an example and sample output of the **stopNode** command.

Example 6-14 stopNode command

```
C:\WebSphereV7\AppServer\profiles\node40c\bin>stopnode -username admin
-password adminpw
ADMU0116I: Tool information is being logged in file

C:\WebSphereV7\AppServer\profiles\node40c\logs\nodeagent\stopServer.log
ADMU0128I: Starting tool with the node40c profile
ADMU3100I: Reading configuration for server: nodeagent
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server nodeagent stop
completed.
```

Stopping a node on z/OS using the STOP command

To stop a node agent on z/OS, you can use the following command:

STOP nodeagent_JOBNAME

For example:

STOP WPAGNTA

Stopping a node (the node agent and servers)

You can use the administrative console to stop a node and its servers with one action. Follow these steps:

1. From the administrative console, select **System Administration Nodes**.
2. Check the box beside the node and click **Stop**.

Restarting a node agent

You can restart a running node agent from the administrative console by doing the following steps from the administrative console:

1. Select **System Administration Node Agents**.
2. Check the box beside the node agent for the server and click **Restart**.

6.5.2 Node agent synchronization

During a synchronization operation, a node agent checks with the deployment manager to see if any configuration documents that apply to the node have been updated. New or updated documents are copied to the node repository, and deleted documents are removed from the node repository.

Automatic synchronization

Automatic configuration synchronization between the node and the deployment manager is enabled by default. You can configure the interval between synchronizations in the administrative console by doing the following steps:

1. Expand **System Administration Node Agents** in the administrative console.
2. Select the node agent process on the appropriate server to open the Properties page.
3. In the Additional Properties section, click **File Synchronization Service**.
4. Configure the synchronization interval. By default, the synchronization interval is set to one minute.

Tip: Increase the synchronization interval in a production environment to reduce the overhead.

Note that a separate setting exists as an administrative console preference. The **Synchronize changes with Nodes** option, when selected, indicates that any time a change is saved to the console, it is automatically synchronized out to the running nodes. To set this preference, select **System administration** → **Console Preferences**.

Forced synchronization

Synchronization can be forced by selecting **System Administration Nodes**. Select a node and click **Synchronize** or **Full Synchronization**.

- **Synchronize** performs an immediate synchronization on the selected node. This type of synchronization is optimized for performance and only synchronizes changed files. If there are issues with manually edited files, this might not result in a complete synchronization.

- The Full Synchronization option disregards optimization and ensures that the node and cell configuration are identical.

Using the syncNode command

The **syncNode** command can be used from the node to force the synchronization of a node's local configuration repository with the master repository on the deployment manager node.

Tip: The **syncNode** command is normally only used in exception situations. To use the **syncNode** command, the node agent must be stopped. You can use the **-stopservers** and **-restart** options on the **syncNode** command to stop the node agent and application servers, and then restart the node agent.

The syntax of the **syncNode** command is shown in Example 6-15.

Example 6-15 syncNode command

```
Usage: syncNode dmgr_host [dmgr_port] [-conntype <type>] [-stopservers]
[-restart] [-quiet] [-nowait] [-logfile <filename>] [-replacelog]
[-trace] [-username <username>] [-password <password>]
[-localusername <localusername>] [-localpassword <localpassword>]
[-profileName <profile>] [-help]
```

You can find information about these options at:

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

Example 6-16 shows an example of using the **syncNode** command. The command is executed from the node. The **-stopservers** and **-restart** options are used to stop all the servers on the node, including the node agent, then restart the node agent after the synchronization.

Example 6-16 syncNode usage examples

```
C:\WebSphereV7\AppServer\profiles\node40c\bin>syncNode T60 8882 -stopservers -re start -username admin -password adminpw
ADMU0116I: Tool information is being logged in file
          C:\WebSphereV7\AppServer\profiles\node40c\logs\syncNode.log
ADMU0128I: Starting tool with the node40c profile
ADMU0401I: Begin syncNode operation for node node40c with Deployment
Manager T60: 8882
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: nodeagent
ADMU0506I: Server name: server40ca
ADMU2010I: Stopping all server processes for node node40c
```

ADMU0512I: Server server40ca cannot be reached. It appears to be stopped.
ADMU0512I: Server nodeagent cannot be reached. It appears to be stopped.
ADMU0016I: Synchronizing configuration between node and cell.
ADMU0018I: Launching Node Agent process for node: node40c
ADMU0020I: Reading configuration for Node Agent process: nodeagent
ADMU0022I: Node Agent launched. Waiting for initialization status.
ADMU0030I: Node Agent initialization completed successfully. Process id is: 4924
ADMU0402I: The configuration for node node40c has been synchronized with Deployment Manager T60: 8882

6.5.3 Removing a node from a cell

There are two ways of removing a node from a network distributed administration cell.

Note: When a node is removed, it is restored to its original configuration.

Using the administrative console

From the administrative console, do the following steps:

1. Select **System Administration Nodes**.
2. Place a check mark in the check box beside the node you want to remove and click **Remove Node**.

This method runs the **removeNode** command in the background.

Using the removeNode command

The **removeNode** command detaches a node from a cell and returns it to a stand-alone configuration.

The -removeNode command syntax is shown in (Example 6-17).

Example 6-17 removeNode command

```
Usage: removeNode [-force] [-quiet] [-nowait] [-logfile <filename>]
[-replaceLog] [-trace] [-username <username>] [-password <password>]
[-profileName <profile>] [-help]
```

For more information, see:

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

To use the command:

1. Change the directory to the *profile_root/bin* directory.
2. Run **removeNode**. All parameters are optional for this command.

In a distributed environment on z/OS, the **removeNode.sh** command is in the *install_root/bin* directory. You will need to specify the *-profileName* parameter to specify the profile for the node you want to remove.

The command performs the following operations:

1. Connects to the deployment manager process to read the configuration data.
2. Stops all of the running server processes of the node, including the node agent process.
3. Removes servers in the node from clusters.
4. Restores the original stand-alone node configuration. This original configuration was backed up when the node was originally added to the cell.
5. Removes the node's configuration from the master repository of the cell. The local copy of the repository held on each node will get updated at the next synchronization point for each node agent. Although the complete set of configuration files are not pushed out to other nodes, some directories and files are pushed out to all nodes.
6. Removes installed applications from application servers in the cell that are part of the node being removed.
7. Copies the original application server cell configuration into the active configuration.

The command provides the *-force* option to force the local node's configuration to be decoupled from the cell even if the deployment manager cannot be contacted. However, if this situation occurs, the cell's master repository will then have to be separately updated to reflect the node's removal, for example, through manual editing of the master repository configuration files.

Example

Table 6-18 shows an example of using the **removeNode** command.

Example 6-18 removeNode example

```
C:\WebSphereV7\AppServer\bin>removeNode -profileName node40c -username admin -password admin
```

ADMU0116I: Tool information is being logged in file
C:\WebSphereV7\AppServer\profiles\node40c\logs\removeNode.log
ADMU0128I: Starting tool with the node40c profile
ADMU2001I: Begin removal of node: node40c
ADMU0009I: Successfully connected to Deployment Manager Server:
t60:8882
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: nodeagent
ADMU0506I: Server name: server40ca
ADMU2010I: Stopping all server processes for node node40c
ADMU0512I: Server server40ca cannot be reached. It appears to be
stopped.
ADMU0510I: Server nodeagent is now STOPPED
ADMU2021I: Removing all servers on this node from all clusters in the
cell.
ADMU2014I: Restoring original configuration.
ADMU2017I: The local original configuration has been restored.

ADMU0306I: Note:
ADMU2031I: Any applications that were uploaded to the Cell40 cell
configuration during addNode using the -includeapps option are not
uninstalled by removeNode.
ADMU0307I: You might want to:
ADMU2032I: Use wsadmin or the Administrative Console to uninstall any
such applications from the Deployment Manager.

ADMU0306I: Note:
ADMU2033I: Any buses that were uploaded to the Cell40 cell
configuration during addNode using the -includebuses option are not
uninstalled by removeNode.

ADMU0307I: You might want to:
ADMU2034I: Use wsadmin or the Administrative Console to uninstall any
such buses from the Deployment Manager.
ADMU2024I: Removal of node node40c is complete.

6.5.4 Renaming a node

The **renameNode** command allows you to modify the node name of a federated server.

renameNode command

The command syntax is shown in Example 6-19.

Example 6-19 renameNode command syntax

Usage: `renameNode dmgr_host dmgr_port node_name [-nodeshortname <name>] [-conntype <type>] [-logfile <filename>] [-trace] [-username <username>] [-password <password>] [-help]`

For more information about the `renameNode` command, see:

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

To run the command, do the following steps:

1. Change to the `profile_root/bin` directory of the deployment manager.
2. Run the **renameNode** command.

The command:

1. Connects to the deployment manager.
2. Stops all servers.
3. Changes the node configuration on the deployment manager.
4. Synchronizes the node.

6.5.5 Node groups

In a distributed environment, you can have nodes in a cell with different capabilities. However, there are restrictions on how the nodes can coexist.

Node groups are created to group nodes of similar capability together to allow validation during system administration processes. Effectively, this means that a node group establishes a boundary from which servers can be selected for a cluster. Nodes on distributed platforms and nodes on the IBM i platform can be members of the same node group, however, they cannot be members of a node group that contains a node on a z/OS platform.

Note: Do not confuse node groups with “groups of nodes” in the job manager. These are two different concepts.

A default node group called DefaultNodeGroup is automatically created for you when the deployment manager is created, based on the deployment manager platform. New nodes on similar platforms are automatically added to the default group. A node must belong to at least one node group, but can belong to more than one.

As long as you have nodes in a cell with similar platforms, you do not need to do anything with node groups. New nodes are automatically added to the node group. However, before adding a node on a platform that does not have the same capabilities as the deployment manager platform, you will need to create the new node group.

Working with node groups

You can display the default node group and its members by selecting **System Administration Node Groups**. See Figure 6-18.

The screenshot shows a software interface for managing node groups. At the top, there are 'New' and 'Delete' buttons. Below them is a toolbar with icons for creating, deleting, and modifying files. The main area has three columns: 'Select', 'Name', 'Members', and 'Description'. The 'Name' column is sorted by clicking the arrow. The table displays one row for the 'DefaultNodeGroup'. The 'Name' column shows 'DefaultNodeGroup', the 'Members' column shows '4', and the 'Description' column shows 'WebSphere Default Node Group.' At the bottom left, it says 'Total 1'.

Select	Name	Members	Description
You can administer the following resources:			
	DefaultNodeGroup	4	WebSphere Default Node Group.
Total 1			

Figure 6-18 Display a list of node groups

- ▶ To create a new node group, click **New**. The only thing that you need to enter is the name of the new node group. Click **OK**.
- ▶ To delete a node group, check the box to the left of the node group name and select **Delete**.
- ▶ To display a node group, click the node group name. For example, in Figure 6-19, we have displayed the DefaultNodeGroup.

General Properties

Name
DefaultNodeGroup

Members
2

Description
WebSphere Default Node Group.

Additional Properties

- [Custom properties](#)
- [Node group members](#)

Apply OK Reset Cancel

Figure 6-19 Node group properties

- To add a node to a node group, display the node group and click **Node group members** in the Additional Properties section (Figure 6-19). When the list appears, click **Add**. You will be able to select from a list of nodes (Figure 6-20).

Node groups > DefaultNodeGroup > Node group members

Use this page to view and configure the nodes in this node group.

[+] Preferences

		Add	Remove
		<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>
Select	Name	▲	
You can administer the following resources:			
<input type="checkbox"/>	dmgr40node		
<input type="checkbox"/>	node40a		
<input type="checkbox"/>	node40b		
<input type="checkbox"/>	node40c		

Figure 6-20 Displaying node group members

6.6 Working with clusters

This section discusses creating, configuring, and managing application server clusters using the administrative console.

Clusters consist of one or more application servers that run the same applications. The configuration of each server can be unique.

Before creating a cluster, consider the number of servers you want to add to the cluster, the nodes that they will be created on, and how the workload should be distributed across the servers.

Work is distributed across the servers in the cluster based on weights assigned to each application server. If all cluster members have identical weights, work is distributed among the cluster members equally. Servers with higher weight values are given more work. An example formula for determining routing preference is as follows:

```
% routed to Server1 = weight1 / (weight_1 + weight_2 + ...+ weight_n)
```

In the formula, n represents the number of cluster members in the cluster. Consider the capacity of the system that hosts the application server.

6.6.1 Creating application server clusters

When you create a cluster, you have the option to create an empty cluster (no servers) or to create the cluster with one or more servers. The first application server added to the cluster acts as a template for subsequent servers. You can create the first server during the cluster creation process or you can convert an existing application server. The rest of the servers must be new and can be created when you create the cluster or added later.

Tip: When creating a cluster, it is possible to select the template of an existing application server for the cluster without adding that application server into the new cluster. If you need to change the attributes of the servers in your cluster after the cluster has been created, you must change each server individually. For this reason, consider creating an application server with the server properties that you want as a standard in the cluster first, then use that server as a template or as the first server in the cluster.

Cluster and cluster member options

When you create a new cluster, you have the following options to consider:

- ▶ Prefer local:

This setting indicates that a request to an EJB should be routed to an EJB on the local node if available. This is the default setting and generally will result in better performance.

- ▶ Configure HTTP session memory-to-memory replication (create a replication domain):

WebSphere Application Server supports session replication to another WebSphere Application Server instance. In this mode, sessions can replicate to one or more WebSphere Application Server instances to address HTTP Session single point of failure.

When you create a cluster, you can elect whether to create a replication domain for the cluster. The replication domain is given the same name as the cluster and is configured with the default settings for a replication domain. When the default settings are in effect, a single replica is created for each piece of data and encryption is disabled. Also, the Web container for each cluster member is configured for memory-to-memory replication.

For more information about replication domains, see 12.7, “Persistent session management” on page 628.

When you create a new cluster member, you have the following options to consider:

- ▶ Basis for first cluster member:

You can add application servers to the cluster when you create the cluster or later.

The first cluster member can be a new application server or you can convert an existing application server so that it becomes the first cluster member.

Subsequent application servers in the cluster must be created new. The first application server in the cluster acts as a template for the subsequent servers.

The options you have depend on the how you create the cluster.

When you use the job manager, you have the option to convert an existing server to use as the first cluster member, or create an empty cluster and run additional jobs to add cluster members.

When you use the deployment manager, you can convert an existing server, create one or more new servers, or create an empty cluster.

Note: The option to use an existing application server does not appear in the deployment manager administrative console if you create an empty cluster, then add a member later. If you want to convert an existing application server as the first server, specify that option when you create the cluster or use the job manager to create the cluster member.

Tip: To remove a server from a cluster, you must delete the server. Take this into consideration when you are determining whether to convert an existing server to a cluster.

► Server weight for each cluster member:

The weight value controls the amount of work that is directed to the application server. If the weight value for this server is greater than the weight values that are assigned to other servers in the cluster, then this server receives a larger share of the workload. The weight value represents a relative proportion of the workload that is assigned to a particular application server. The value can range from 0 to 20.

Member weight: Specify the relative weight of this server in the cluster. Values are from 0 to 20. 0 indicates that work is to be routed to this server only in the event that no other servers are available.

On z/OS, weight is used to balance some of the workload types, but others are balanced by the z/OS system:

- For HTTP requests, weights are used to distribute HTTP traffic between the Web server plug-in and the controller handling the clustered application server. Assign a higher weight value to the application server that should receive the HTTP traffic.
- For Web services calls, information is transferred from a servant in one application server to a controller in another application server. The application server that receives the call has the highest weight value.
- Weight has no affect on Internet Inter-ORB Protocol (IIOP) requests. IIOP requests are distributed to the correct application server using the sysplex distributor.

Using the deployment manager administrative console

To create a new cluster:

1. Select **Servers** → **Clusters** → **WebSphere application server clusters**.
2. Click **New**.
3. Enter the information for the new cluster (see Figure 6-21):
 - Enter a cluster name of your choice.
 - On z/OS, you will also be asked for the short name for the cluster.

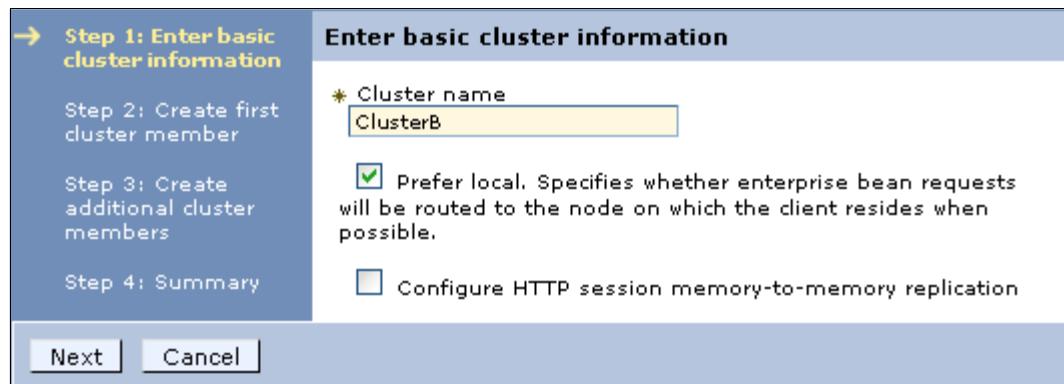


Figure 6-21 Creating a new cluster

4. Create first cluster member: The first cluster member determines the server settings for the cluster members (Figure 6-22).

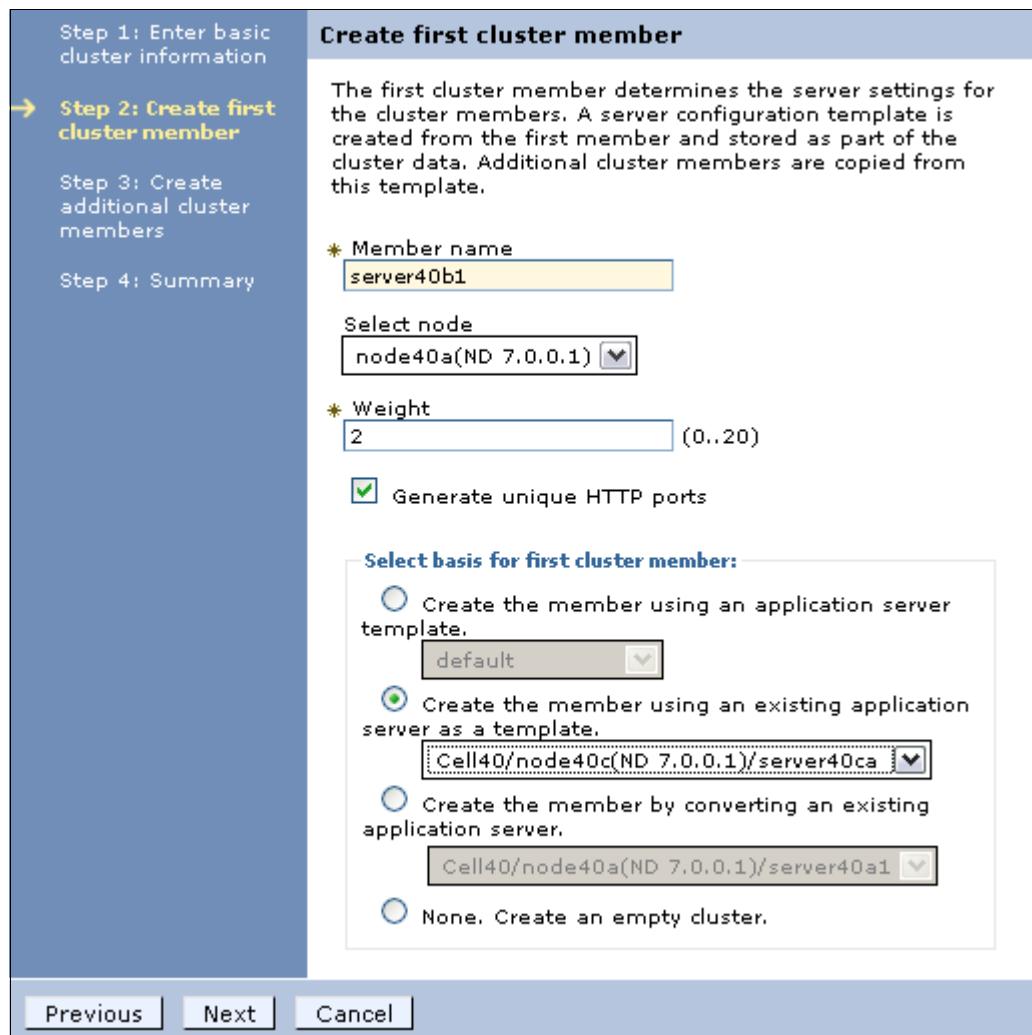


Figure 6-22 First cluster member

- Member Name: Type a name of the new server to be added to the cluster. On z/OS, you will also be asked for the short name for the server.
- Select Node: Specifies the node on which this new cluster member is created.

- Server weight: Assign the weight for this server.
- Generate unique HTTP ports: Generates unique port numbers for every transport that is defined in the source server, so that the resulting server that is created will not have transports that conflict with the original server or any other servers defined on the same node.
- Select basis for first cluster member:
 - If you select **Create the member using an application server template**, the settings for the new application server are identical to the settings of the application server template you select from the list of available templates.
 - If you select **Create the member using an existing application server as a template**, the settings for the new application server are identical to the settings of the application server you select from the list of existing application servers. However, applications that are installed on the template server are not installed on the new servers.
 - If you select **Create the member by converting an existing application server**, the application server you select from the list of available application servers becomes a member of this cluster.
Applications that are installed on the existing server are automatically installed on new members of the cluster.
Note that the only way to remove a server from a cluster is to delete it, and when you delete the cluster, all servers in the cluster are deleted. Consider this before selecting this option.
- If you select **None. Create an empty cluster**, a new cluster is created, but it does not contain any cluster members.

Click **Next**.

5. Create additional cluster members: Use this page to create additional members for a cluster. You can add a member to a cluster when you create the cluster or after you create the cluster. A copy of the first cluster member that you create is stored as part of the cluster data and becomes the template for all additional cluster members that you create.

To add a member, enter a new server name, select the node, and click **Add Member**. The new member will be added to the list. See Figure 6-23.

Step 1: Enter basic cluster information

Step 2: Create first cluster member

→ **Step 3: Create additional cluster members**

Step 4: Summary

Create additional cluster members

Enter information about this new cluster member, and click Add Member to add this cluster member to the member list. A server configuration template is created from the first member, and stored as part of the cluster data. Additional cluster members are copied from this template.

* Member name
server40b2

Select node
node40b(ND 7.0.0.1)

* Weight
2 (0..20)

Generate unique HTTP ports

Add Member

Use the Edit function to edit the properties of a cluster member that is already included in this list. Use the Delete function to remove a cluster member from this list. You are not allowed to edit or remove the first cluster member or an already existing cluster member.

Select	Member name	Nodes	Version	Weight
<input checked="" type="checkbox"/>	server40b1	node40a	ND 7.0.0.1	2
Total 1				

Previous | Next | Cancel |

Figure 6-23 Additional cluster members

6. When all the servers have been entered, click **Next**.
7. A summary page shows you what will be created.
8. Click **Finish** to create the cluster and new servers.
9. Save the configuration.

Adding additional servers to the cluster

To add a server using the administrative console:

1. Select **Servers** → **Clusters** → **WebSphere application server clusters**.
2. Click the cluster name.
3. Under the Additional Properties sections, click **Cluster members**.
4. Click **New**. This will open the same configuration panel presented when you created the cluster (Figure 6-23 on page 343).
5. Enter the name of the new server to create, select the node, and select the options to use.
6. Click **Next**, then click **Finish**.

Using cluster member templates

When you created your cluster's servers, a server template was created for the cluster by copying the first cluster member's configuration. This template is then used when you create additional servers for that cluster. This is important to understand, because you might not get the results you expect when working with clusters.

To view a cluster's member templates using the administrative console:

1. Select **Servers** → **Clusters** → **WebSphere application server clusters**.
2. Click the cluster name.
3. Under the Additional Properties sections, click **Cluster members**.
4. Click **Templates...**

This will open a configuration panel that lists the templates in this cluster. Typically, you will only have one template, but you will have additional templates if the cluster includes servers that are at different versions of WebSphere Application Server (for example, versions 7 and 6.1).

From this configuration panel, you can view and modify the server attributes of the template. If you modify the attributes, it is important to understand existing cluster members will not be affected. The template is only used for creating new cluster members.

To modify the attributes of a cluster's member using the administrative console:

1. Select **Servers** → **Clusters** → **WebSphere application server clusters**.
2. Click the cluster name.
3. Under the Additional Properties sections, click **Cluster members**.
4. Click the cluster member. This will open the server configuration panel where you can make your change.

If you want to make the same change to multiple cluster members, you must repeat these steps. You should also modify the same attributes in the cluster member template, because new cluster members will be created based on the template. If you do not change the cluster's template(s), additional cluster members will not match the existing members.

If you want to modify the same server attribute(s) across all of a cluster's members and you have several cluster members, one way to accomplish this using the administrative console:

1. Navigate to the cluster's templates (see before).
2. Click the cluster template and make your desired changes.
3. Save your changes.
4. Delete all of the members in the cluster.
5. Recreate the members.

The new members will be created from the updated cluster member template, and all of the cluster members will have the same configuration.

Using the job manager

When you create an application server cluster from a job manager, you can either create an empty cluster and run subsequent jobs to add cluster members, or you can convert an existing application server as your first cluster member. If you want to create a cluster with one or more new application servers in one step, use the administrative console instead of the job manager.

Building a cluster using the job manager is done in two major steps:

1. Create the cluster.
2. Create the cluster members.

Create the cluster

To create an application server cluster from the job manager, make the following selections as you step through the process to submit the job:

1. Step 1: Select the **Create cluster** job type.
2. Step 2: Select the deployment manager as the job target.

Enter the user ID and password with administrative authority on the deployment manager.

3. Step 3: Specify the job parameters.
 - Specify the name of the new cluster.

Step 1: Choose a job type

Step 2: Choose job targets

→ **Step 3: Specify job parameters**

Step 4: Schedule the job

Step 5: Review the summary and submit the job

Specify job parameters

Job type: Create cluster

* Cluster name
Cluster40A

Prefer local

Cluster type

Short name

Additional job parameters.

Replication domain

Create domain
false

Convert server

Server node

Server name

Member weight

Node group

Replication entry

Previous **Next** **Cancel**

The screenshot shows a configuration interface for creating a new cluster. On the left, a vertical sidebar lists five steps: Step 1 (Choose a job type), Step 2 (Choose job targets), Step 3 (Specify job parameters - highlighted with a blue arrow), Step 4 (Schedule the job), and Step 5 (Review the summary and submit the job). The main panel is titled 'Specify job parameters' and contains sections for 'Job type: Create cluster' and 'Replication domain'. Under 'Job type: Create cluster', there is a required field 'Cluster name' with the value 'Cluster40A'. Other fields include 'Prefer local', 'Cluster type', 'Short name', and a checkbox for 'Additional job parameters' which is unchecked. The 'Replication domain' section includes fields for 'Create domain' (set to 'false'), 'Server node', 'Server name', 'Member weight', 'Node group', and 'Replication entry'. At the bottom of the panel are 'Previous', 'Next', and 'Cancel' buttons.

Figure 6-24 Specify the options for the new server

Optionally:

- Prefer local: true (the default) or false
- Short name: cluster short name on z/OS platforms
- Cluster type: The options are:
 - APPLICATION_SERVER (the default)
 - PROXY_SERVER
 - ONDEMAND_ROUTER

Leave this field blank to create an application server cluster.

- Create replication domain: true or false (the default)
- Convert server settings: If you want to use an existing server as the first member of the cluster, complete the server node and server name fields. The other fields are optional.

If you specify the cluster name, and take all other defaults, you will create an empty cluster. When you create an empty cluster, it does not appear in the deployment manager console until you submit a job to add a member to it.

Create the cluster members

To create new application server cluster members from the job manager, make the following selections as you step through the process to submit the job:

1. Step 1: Select the **Create cluster member** job type.
2. Step 2: Select the deployment manager as the job target.
Enter the user ID and password with administrative authority on the deployment manager.
3. Step 3: Specify the job parameters.
 - Specify the name of the cluster.
 - Specify the node where the cluster member will be created.
 - Specify the name for the new cluster member.

Step 1: Choose a job type

Step 2: Choose job targets

→ Step 3: Specify job parameters

Step 4: Schedule the job

Step 5: Review the summary and submit the job

Specify job parameters

Job type: Create cluster member

* Cluster name: Cluster40A

* Member node: node40a

* Member name: C140ASrv1

Member weight:

Member UUID:

Generate unique ports:

Replicator entry:

Short name:

Additional job parameters.

Figure 6-25 Specify the options for the new server

Optionally:

- Member weight: Specify the relative weight of this server in the cluster. Values are from 0 to 20. 0 indicates that work is to be routed to this server only in the event that no other servers are available.
- Member UUID
- Generate unique ports (true or false):
- Replicator entry (true or false): Specifying true will add a replicator entry for this server in the cluster replication domain. The default is false.
- Short name: The short name for the server on z/OS.

If this is the first server in the cluster, or if you want to specify a different node or core group, expand the additional job parameters section. It contains settings to specify the template information and the option to specify a node and core group other than the default.

6.6.2 Viewing cluster topology

The administrative console provides a graphical view of the existing clusters and their members. To see the view, do the following steps:

1. Select **Servers Cluster Topology**.
2. Expand each category. See Figure 6-26.

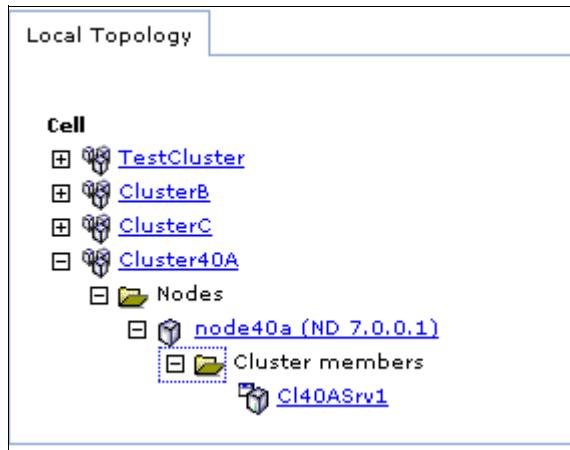


Figure 6-26 Cluster topology view

3. Selecting a server will take you to the configuration window for the application server.

6.6.3 Managing clusters

Application servers within a cluster can be managed as independent servers. A second option is to manage all the servers in the cluster as a single entity.

Using the administrative console

To display and manage the application server clusters:

1. Select **Servers → Clusters → WebSphere application server clusters**.
2. Check each cluster you want to work with and select one of the following options:
 - Start: Use this option to start all servers in the cluster.
 - Stop: Use this option to stops all servers in the cluster. This allows the server to finish existing requests and allows failover to another member of the cluster.
 - Ripplestart: Use this option to stop, then start all servers in the cluster.

- ImmediateStop: Stop all servers immediately.
- Delete: Deletes the cluster and all servers in the cluster.

Using the job manager

The job manager provides several job types that help you manage clusters:

- ▶ Delete cluster
- ▶ Delete cluster member
- ▶ Start cluster: Use this option to start all servers in a cluster:
 - You can specify that a ripplestart be used (specify true or false). The default is that a ripplestart is not used. A ripplestart will stop, then restart each server.
 - You can also specify a timeout value. If the timeout expires and all servers have not started, the state of the cluster will be reported without waiting any longer for the servers to start.
- ▶ Stop cluster:
 - You can also specify a timeout value. If the timeout expires and all servers have not started, the state of the cluster will be reported without waiting any longer for the servers to start.

6.7 Working with virtual hosts

Note: For many users, creating virtual hosts is unnecessary because the default_host that is provided is sufficient.

For an example of defining and using a new virtual host, see 15.1.4, “Defining the ITSO Bank virtual host” on page 761.

A *virtual host* is a configuration enabling a single host machine to resemble multiple host machines. It consists of a host alias or aliases, which consist of a host name and a port number. If you specify an asterisk (*) as a host name, all host names and IP addresses that the Web server can receive will be mapped to that virtual host.

The following virtual hosts are defined during installation:

- ▶ The *default_host* virtual host is intended for access to user applications, either through the HTTP transport or through a Web server.

Host aliases in this virtual host generally include the ports required to access applications from the Web server and directly to the application server. For example, the `wc_defaulthost`, `wc_defaulthost_secure`, `sip_defaulthost`, and `sip_defaulthost_secure` ports for application servers and ports 80 and 443 for requests via the Web server.

- ▶ The *admin_host* virtual host is used for access to the WebSphere administrative console.

At installation time, it is configured to match requests on the `wc_adminhost` and `wc_adminhost_secure` ports for the stand-alone server or deployment manager.

- ▶ The *proxy_host* virtual host includes default port definitions, port 80 and 443, which are typically initialized as part of the proxy server initialization. Use this proxy host as appropriate with routing rules associated with the proxy server.

When you install an application, you associate a virtual host with each Web module in the application. By associating a virtual host with a Web module, requests that match the host aliases for the virtual host should be processed by servlets in this Web module. The Web server plug-in also checks the URI of the request against the URIs for the Web module to determine whether the Web module can handle them or not. You can view or modify the virtual host a Web module is assigned to by selecting **Applications** → **WebSphere enterprise applications** → **app_name** → [Web Module Properties] **Virtual hosts**.

A single virtual host can be associated with multiple Web modules as long as each application has unique URIs. If there are duplicate URIs among applications, different virtual hosts must be created and associated with each of the applications.

A default virtual host is associated with a Web container when you create the application server. To find the default virtual host select **Servers** → **Server Types** → **WebSphere application servers** and click the server name to open the configuration page. In the Container settings section, expand Web container settings and select **Web container**.

6.7.1 Creating and updating virtual hosts

By default, default_host is associated with all user application requests. The following examples show cases in which multiple virtual hosts should be created:

- ▶ Applications with conflicting URIs
- ▶ Special support for extra ports
- ▶ Provide independence of each virtual host for applications and servers

To create a new virtual host:

1. Select **Environment** → **Virtual hosts** and then click **New**.
2. Enter a name for the virtual host and click **Apply**.
Note that two links become active: Host Aliases and MIME Types.
3. Click **Host Aliases** in the Additional Properties pane.
4. Click **New**.
5. Enter values for the Host Name and Port fields and click **OK**.

The host aliases are not necessarily the same as the host name and port number of the WebSphere Application Servers. They are the host names and port numbers that the Web server plug-in is expecting to receive from the browser. The Web server plug-in will send the request to the application server using the host name and port number in the transport setting for that server. If the Web server is running on a separate machine from WebSphere, then the host aliases are for Web server machines.

Mapping HTTP requests to host aliases is case sensitive and the match must be alphabetically exact. Also, different port numbers are treated as different aliases.

For example, the request `http://www.myhost.com/myservlet` does *not* map to any of the following steps:

`http://myhost/myservlet`
`http://www.myhost.com/MyServlet`
`http://www.myhost.com:9876/myservlet`

If the Web server plug-in receives a request that does not match one of the virtual hosts, then an HTTP error will be returned to the user.

Simple wild cards can be used on the host aliases. A * can be used for the host name, the port or both. It means that any request will match this rule.

Note: If the virtual host is used in a cluster environment, all host aliases used by servers in the cluster should be registered in the virtual host.

6. Save your changes.

Host aliases can also be updated for virtual hosts through the administrative console.

1. Select **Environment** → **Virtual hosts**.
2. Click the virtual host name to open the configuration page.
3. Click **Host Aliases** in the Additional Properties pane.
4. Click **New**.
5. Enter values for the Host Name and Port fields and click **OK**.

Important: If you create, delete, or update virtual hosts, you need to regenerate the Web server plug-in.

6.8 Managing your configuration files

This section summarizes some of the most common system management tasks dealing with configuration files:

- ▶ Backing up a profile
- ▶ Restoring a profile
- ▶ Exporting and importing profiles

6.8.1 Backing up a profile

Use the **backupConfig** command to back up a profile. You can run this command from the *was_home/bin* directory and use the **-profileName** option to specify the profile to back up. Or you can execute the command from the *profile_home/bin* directory.

The command will zip the configuration file and store it in the current directory or a specified file name. The zip file can be restored using the **restoreConfig** command. By default, **backupConfig** will stop all servers in the configuration before performing the backup.

The syntax is shown in Example 6-20.

Example 6-20 backupConfig command

```
Usage: backupConfig [backup_file] [-nostop] [-quiet] [-logfile
<filename>] [-replacelog] [-trace] [-username <username>] [-password
<password>] [-profileName <profile>] [-help]
```

For information about the command options and usage, see:

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

The `backup_file` parameter specifies the file where the backup is to be written. If you do not specify a backup file name, a unique name is generated and the file is stored in the current directory. If you specify a backup file name in a directory other than the current directory, the specified directory must exist.

Example

Example 3-13 shows an example of backing up a deployment manager.

Example 6-21 backupConfig example

```
C:\WebSphereV7\AppServer\bin>backupConfig
c:\WebSphereV7\Backups\dmgr01Mar1509 -profileName Dmgr01 -logfile
c:\WebSphereV7\Backups\logs\dmgr01Mar1509
ADMU0116I: Tool information is being logged in file
          c:\WebSphereV7\Backups\logs\dmgr01Mar1509
ADMU0128I: Starting tool with the Dmgr01 profile
ADMU5001I: Backing up config directory
          C:\WebSphereV7\AppServer\profiles\dmgr01\config to file
          C:\WebSphereV7\Backups\dmgr01Mar1509
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: dmgr
ADMU2010I: Stopping all server processes for node Dmgr02Node
ADMU0512I: Server dmgr cannot be reached. It appears to be stopped.
.....
.....
.....
ADMU5002I: 908 files successfully backed up
```

6.8.2 Restoring a profile

Use the `restoreConfig` command to restore a profile configuration using an archive previously generated using `backupConfig`. If the configuration to be restored exists, the config directory is renamed to `config.old` (then `config.old_1`, etc.) before the restore begins. The command then restores the entire contents of the `profile_root/config` directory. By default, all servers on the node stop before the configuration restores so that a node synchronization does not occur during the restoration.

The syntax is shown in Example 6-22.

Example 6-22 restoreConfig command

```
Usage: restoreConfig backup_file [-location restore_location] [-quiet]
[-nostop] [-nowait] [-logfile <filename>] [-replacetlog] [-trace]
[-username <username>] [-password <password>] [-profileName <profile>]
[-help]
```

For information about the command, see:

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

Example

Example 3-14 shows an example of restoring an application server profile.

Example 6-23 restoreConfig example

```
C:\WebSphereV7\AppServer\bin>restoreConfig
c:\WebSphereV7\Backups\dmgr01Mar1509
-profileName dmgr01 -logfile
c:\WebSphereV7\Backups\logs\dmgr01Mar1509_r
ADMU0116I: Tool information is being logged in file
          c:\WebSphereV7\Backups\logs\dmgr01Mar1509_r
ADMU0128I: Starting tool with the Dmgr01 profile
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: dmgr
ADMU2010I: Stopping all server processes for node Dmgr02Node
ADMU0512I: Server dmgr cannot be reached. It appears to be stopped.
ADMU5502I: The directory
C:\WebSphereV7\AppServer\profiles\dmgr01\config
          already exists; renaming to
          C:\WebSphereV7\AppServer\profiles\dmgr01\config.old
ADMU5504I: Restore location successfully renamed
ADMU5505I: Restoring file c:\WebSphereV7\Backups\dmgr01Mar1509 to
location C:\WebSphereV7\AppServer\profiles\dmgr01\config
.....
.....
.....
ADMU5506I: 908 files successfully restored
ADMU6001I: Begin App Preparation -
ADMU6009I: Processing complete.
ADMU6002I: Begin Asset Preparation -
ADMU6009I: Processing complete.
```

6.8.3 Exporting and importing profiles

The target configuration of an archive export / import can be a specific server or an entire profile.

To use an archive, you would:

1. Export a WebSphere configuration. This creates a zip file with the configuration.
2. Unzip the files for browsing or update for use on other systems. For example, you might need to update resource references.
3. Send the configuration to the new system. An import can work with the zip file or with the expanded format.
4. Import the archive. The import process requires that you identify the object in the configuration you want to import and the target object in the existing configuration. The target can be the same object type as the archive or its parent:
 - If you import a server archive to a server configuration, the configurations are merged.
 - If you import a server archive to a node, the server is added to the node.

Server archives

The following command in `wsadmin` can be used to create an archive of a server:

```
$AdminTask exportServer {-archive <archive_location> -nodeName <node> -serverName <server>}
```

This process removes applications from the server that you specify, and breaks the relationship between the server that you specify and the core group of the server, cluster, or bus membership. If you export a single server of a cluster, the relation to the cluster is eliminated.

To import a server archive, use the following command:

```
$AdminTask importServer {-archive <archive_location> [-nodeInArchive <node>] [-serverInArchive <server>] [-nodeName <node>] [-serverName <server>]}
```

When you use the `importServer` command, you select a configuration object in the archive as the source and select a configuration object on the system as the target. The target object can match the source object or be its parent. If the source and target are the same, the configurations are merged.

Profile archives

You can create a configuration archive (CAR) file containing the configuration of a stand-alone application server profile for later restoration. A CAR file can be used to clone the original profile to another machine or system. CAR files can be bundled in a customized installation package for use with the Installation Factory feature. For more information about using the Installation Factory, refer to the information center.

You can only create an archive of an unfederated profile (standalone application server).

The following command in `wsadmin` can be used to create an archive of a profile:

```
$AdminTask exportWasprofile {-archive <archive_location>}
```

6.9 Managing applications

WebSphere Application Server V7 supports J2EE 1.3, J2EE 1.4, and Java EE 5, which we refer to as *enterprise applications*.

A new concept has been introduced with V7 that allows you to manage applications at a business level. A *business-level application* does not contain binary application files, but rather, identifies WebSphere and non-WebSphere artifacts that logically belong together to form a business-level application. Deployable EAR files and modules are assets for the business-level application. After being deployed, an application becomes a composition unit to the business-level application. Administration of the artifacts that belong to the business-level application is separate from the administration of the business level definition.

Both enterprise and business-level applications can be configured and managed using the following methods:

- ▶ Using the `wsadmin` tool:

The `wsadmin` scripting tool can be used to manage both business level and enterprise applications.

Using scripts to manage applications is more complicated than using the other methods. It requires skill in at least one of the supported scripting languages and a complete understanding of the WebSphere Application Server configuration. However, scripting can offer a greater degree of control and can be quite useful in situations where you are performing the same administrative tasks multiple times, or when the tasks are to be done by multiple administrators. With introduction of script libraries, many of the scripting tasks have become easier to perform.

Information about using `wsadmin` scripts is found in Chapter 8, “Administration with scripting” on page 439.

- ▶ Using the administrative console:

Using the administrative console is an easy way to install or update business level and enterprise applications. Wizards take you through the process and provide help information at each step. This is the method discussed in this section at a high level.

- ▶ Using the job manager:

The job manager can be used to install, update, uninstall, stop, and start enterprise applications. Currently, business-level applications must be managed by submitting `wsadmin` jobs.

- ▶ Shortcuts:

WebSphere Rapid Deployment tools provide a shortcut to installing, uninstalling, and updating applications. However, these tools do not support JEE 5 nor J2EE 1.2 specification levels. For information about using this feature, see the topic, *Rapid deployment of J2EE applications*, in the Information Center.

6.9.1 Managing enterprise applications: Administrative console

To view and manage applications using the administrative console, select **Applications Enterprise Applications**.

In the window, you see the list of installed applications and options for performing application management tasks. Select one or more applications by checking the box to the left of the application name, and then click an action to perform. The exception to this is the Install option, which installs a new application, and requires no existing application to be selected. See Figure 6-27.

Enterprise Applications

Use this page to manage installed applications. A single application can be deployed onto multiple servers.

Preferences

Action Buttons: Start, Stop, Install, Uninstall, Update, Rollout Update, Remove File, Export, Export DDL

Toolbar: Select, New, Open, Save, Print, Refresh, Filter, Sort

Select	Name	Application Status
<input type="checkbox"/>	DefaultApplication	
<input type="checkbox"/>	PlantsByWebSphere	
<input type="checkbox"/>	SamplesGallery	
<input type="checkbox"/>	iwtApp	
<input type="checkbox"/>	query	

Figure 6-27 Working with enterprise applications

The following list describes the actions you can choose on this window:

- ▶ **Start:**
Applications normally start when the server to which they are mapped starts. Exceptions to this include when the application has just been installed, and when the application has been stopped manually.
- ▶ **Stop:**
You can stop an application manually without affecting the rest of the application server processes. This is common when you are updating an application or want to make it unavailable to users.
- ▶ **Install:**
The install option takes you through the process of installing a new enterprise application EAR file.
- ▶ **Uninstall:**
Use this option to uninstall an application. This removes it from the application servers and from the configuration repository.
- ▶ **Update or Rollout Update:**
Applications can be updated in several ways. The update options include full application, single module, single file, and partial application.
- ▶ **Remove file:**
With this option, you can remove a single file from an application.
- ▶ **Export:**

- ▶ Use this option to export an EAR file of the application.
- ▶ Export DDL:
 - ▶ Use this option to export DDL files found in the application.
- ▶ Export File:
 - ▶ Use this option to export individual files found in the application.

6.9.2 Installing an enterprise application

To install an enterprise application into a WebSphere configuration, you must install its modules onto one or more application servers.

Business-level applications (BLAs): Enterprise applications are considered a composite unit of a business-level application (BLA) and will be can either be added to the BLA at installation time, or can be installed when the undeployed application assets are added to the BLA.

Adding a new cluster member: When an application server is added as a member to a server cluster, the modules installed on other members are also installed on the new member. You do not need to re-install or upgrade the application.

In this section, we give you an overview of application installation. Later in this book, we discuss application installation in detail.

- ▶ An example of installing an enterprise application is given in Chapter 15, “Deploying applications” on page 753.
- ▶ An example of installing a business-level application can be found in Chapter 14, “Packaging applications for deployment” on page 709.

Using business-level applications

When you install an application, it is added to a business-level application (BLA). This can be an existing BLA, or you can create a new BLA as part of the application install process. If you do not plan to use BLAs to manage your applications, allowing the BLA to be created during the application install process is fine.

However, if you plan to manage your application as part of a BLA, you should start the installation process by importing the application EAR file as an asset, creating a business-level application to hold the assets, and then installing the assets.

The steps that this entails are:

1. Import the application EAR file as an asset:
 - a. Open the WebSphere administrative console and select **Applications** → **Application Types**. Click the **Assets** link.
 - b. Click the **Import** button and complete the steps in the wizard.

The asset will be imported to the asset repository. The default location is *profile_root/installAssets/asset_name/BASE/*.

Repeat this step for any additional assets that will belong to the BLA.

Save to master configuration when done.

2. Create a business-level application:
 - a. Select **Applications** → **Application Types** → **Business-level applications**. Click the **New** button.
 - b. Enter the name for the BLA and click **Apply**.
 - c. On the Business-level applications page, click the **Add** button under the Deployed assets section. You can use this option to add an asset (the EAR file) or a shared library.
Select the **Add Asset** option.
 - d. On the next panel select EAR file and click **Continue**. You will now configure (install) this asset with the necessary deployment options. This follows the normal steps for installing an application to WebSphere Application Server (see “Installing an application using the administrative console” on page 361).
 - e. Proceed through the installation panels until the Summary page is shown, and click **Finish**. WebSphere now installs the application and it is now a composition unit, an asset which has been configured.

Note: At Step 1: Select installation options WebSphere generates a unique application name such as app1143018803114601914 for the application. You might want to change this to something more descriptive.

3. **Save** to the master configuration when done.

See 14.10.1, “Example: Creating a business-level application” on page 749 for an example.

Installing an application using the administrative console

There are two approaches to installing an enterprise application. The first approach is to install the application as part of the process of adding assets to a

BLA (as described in the previous section). The second approach is to install the application and select a BLA, or create a new BLA as part of the process. This section assumes that you are taking the latter approach.

In either case, the process is basically the same after you get into the application installation wizard.

Follow these steps for this task:

1. Select **Applications** **New Application** **New Enterprise Application**.
2. Specify the location of the EAR file to install, as shown in Figure 3-55 on page 149.

The EAR file that you are installing can be either on the client machine running the Web browser, or on any of the nodes in the cell (Figure 6-28).

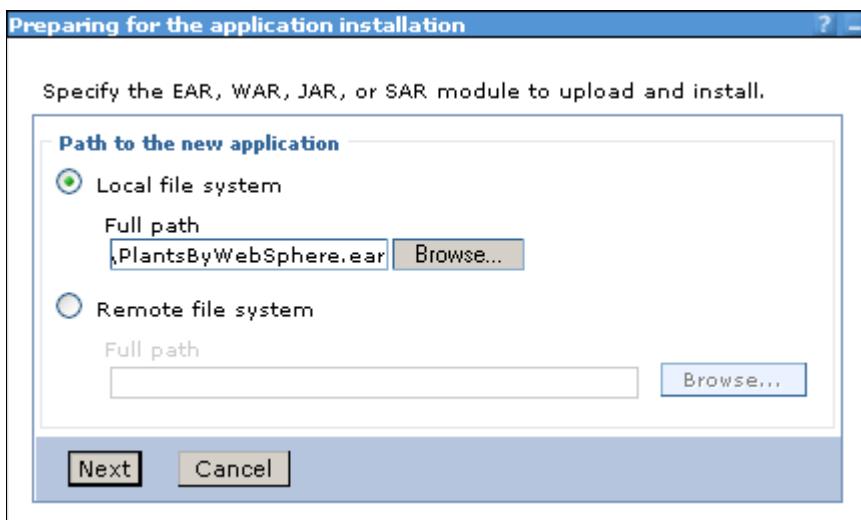


Figure 6-28 *Installing an enterprise application*

Click **Next**.

3. Select whether to use the fast path through the wizard or to display all the installation steps in the wizard.

The fast path will only display the steps in the wizard where decisions cannot be made through defaults.

If you expand the **Choose to generate default bindings and mapping** you can alter the bindings for the application you are deploying. If you select the **Generate Default Bindings** option, WebSphere Application Server completes any incomplete bindings in the application with default values, but it does not alter any existing bindings. Checking the **Override existing**

bindings allows you to specify a bindings file which contains new bindings (Figure 6-29).

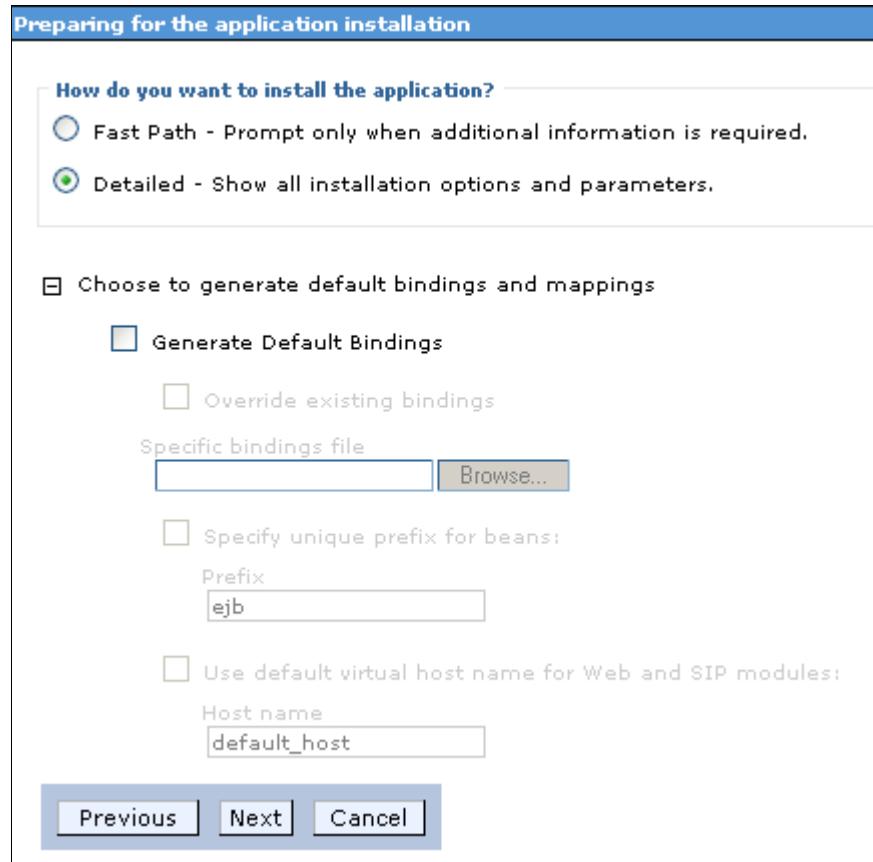


Figure 6-29 Select the path through the wizard and default binding options

Click **Next**.

4. The rest of the installation process is done in steps. The steps can vary, depending on the contents of the EAR file and whether you selected the fast path or detailed path through the wizard. The following steps are a typical sequence for the detailed path:
 - a. Provide options to perform the installation.

(New in V7) This panel also provides you the option of selecting a BLA or creating a new one. If you are installing the application as a result of adding an asset to a BLA, the correct BLA should be selected.

If you create a new BLA, it will have the same name as the application.

- b. Map modules to servers.
 - c. Provide JSP reloading options for Web modules
 - d. Map shared libraries
 - e. Map shared library relationships
 - f. Initialize parameters for servlets
 - g. Provide JNDI names for beans
 - h. (**New in V7**) Bind EJB business interfaces to a JNDI name.
 - i. Map EJB references to beans
 - j. Map resource references to resources
 - k. Map virtual hosts for Web modules
 - l. Map context roots for Web modules
 - m. Map security roles to users or groups
 - n. (**New in V7**) Specify metadata options. The settings on this page can be used to instruct a Java EE 5 enterprise bean (EJB) or Web module deployment descriptor to ignore annotations that specify deployment information.
 - o. Summary
5. Click **Finish** to install the application.
 6. Save the configuration and synchronize the changes to the nodes.
 7. Update the Web server plug-in and propagate it to the Web server.

Using the job manager

You can install enterprise applications from the job manager by selecting the following options to submit a job.

Note: When you use the job manager to install an application, the only options you can specify from the job manager console is the location to install. All other the options that you would see in the administrative console will default.

The first step is to make the application available to the system where the installation job will run.

1. Before installing an application, you must transfer the application binaries (EAR file) from the job manager to the node where the job will run using the distributeFile job (see 5.3.2, “Distributing files using the job manager” on page 283 to define the directories required for this task).

In Rational Application Developer, export the EAR file to:

jmgr_profile_root/config/temp/JobManager directory

2. In the Job manager console, select the **Job** → **Submit** menu. This will launch the Job properties wizard.
 - a. Select **Distribute file** as the job type and click **Next**.
 - b. Select the deployment manager or administrative agent as the target node.
 - c. Enter the EAR file location on the job manager and the location to store the EAR file on the node. The arguments are entered as shown in Figure 6-30.

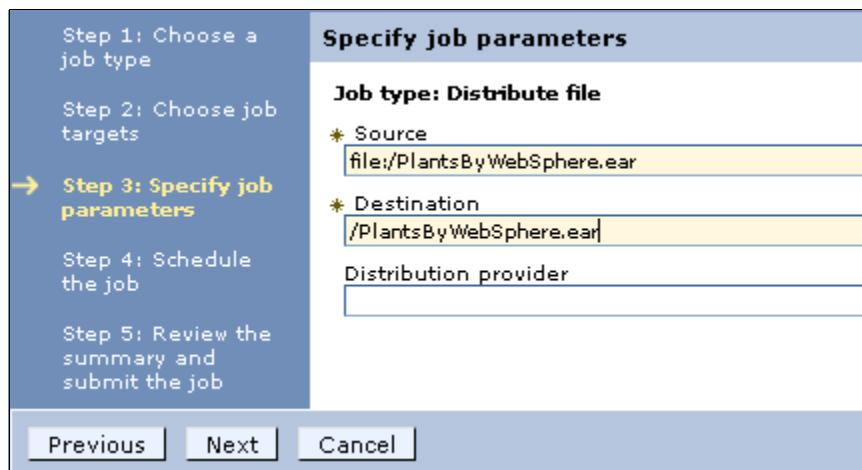


Figure 6-30 Specify the location of the source and target file

Using these arguments, the script file will be distributed to the following location:

dmgr_profile_root/downloadedContent/app_name

Click **Next**.

- d. Take the defaults for the job schedule. The defaults will execute the distribute file job once. Click **Next**.
- e. Click **Finish**. Monitor the status of the job and ensure it completes successfully.

To install the application from the job manager, make the following selections as you step through the process to submit the job:

1. Step 1: Select the **Install application** job type.
2. Step 2: Select the node where the job will run.

Enter the user ID and password with administrative authority on the target node.

3. Step 3: Specify the job parameters.

At minimum:

- Specify the name of the new application.
- If the target is a deployment manager, enter the name of the node and server or cluster that the application will be installed on (Figure 6-31).

The screenshot shows a software interface for specifying job parameters. On the left, a vertical navigation bar lists steps: Step 1: Choose a job type, Step 2: Choose job targets, Step 3: Specify job parameters (which is highlighted in yellow), Step 4: Schedule the job, and Step 5: Review the summary and submit the job. The main panel is titled 'Specify job parameters' and contains a section for 'Job type: Install application'. It includes fields for 'Application name' (containing 'PlantsByWebSphere'), 'Application location' (empty), 'Server name' (empty), 'Node name' (empty), and 'Cluster name' (containing 'ClusterB'). Each field has a 'Find...' button to its right. At the bottom of the panel are 'Previous', 'Next', and 'Cancel' buttons.

Figure 6-31 Specify the options for application install

6.9.3 Uninstalling an enterprise application

To uninstall a no longer needed enterprise application, use one of the following methods:

From the administrative console:

1. Select **Applications Application Types WebSphere enterprise applications**.
2. Check the application you want uninstall and click **Uninstall**.

From the job manager, use the **Uninstall application** job type.

Note: When you uninstall an application and it is the only composite unit in a business-level application, the BLA is also deleted.

6.9.4 Starting an enterprise application

An application starts automatically when the application server to which it is mapped starts. You only need to start an application immediately after installing it, or if you have manually stopped it.

Application startup: Starting an application server starts the applications mapped to that server. The order in which the applications start depends on the startup order you assigned to each them. The application with the lowest startup order value is started first. Applications that have the same order designation are started in no particular order. Enabling the parallel start option for the application server means to start applications with the same weight in parallel.

To view or change the application startup order, select **Applications Application Types WebSphere enterprise applications**. Open the configuration page for the application by clicking the application name and click **Startup behavior**.

An application can be started by following these steps from the administrative console:

1. Select **Applications Application Types WebSphere enterprise applications**.
2. Check the application you want and click **Start**.

An application can be started from the job manager console by selecting the **Start application** job type.

Note: In order to start an application, the application server that contains the application has to be started. If not, the application displays in the administrative console as unavailable and you are not able to start it.

6.9.5 Stopping an enterprise application

An application can be stopped using the administrative console.

From the administrative console, do the following steps:

1. Select **Applications Application Types WebSphere enterprise applications**.
2. Check the application you want to stop and click **Stop**.

Or, from the job manager console, use the **Stop application** job type.

6.9.6 Preventing an enterprise application from starting on a server

By default, an application will start when the server starts. The only way to prevent this occurrence is to disable the application from running on the server.

From the administrative console:

1. Select **Applications Application Types WebSphere enterprise applications**.
2. Click the application to open the configuration.
3. Select **Target specific application status** in the Detail Properties table.
4. Select the server for which you want to disable the application.
5. Click the **Disable Auto Start** button.
6. Save the configuration.

6.9.7 Viewing application details

The administrative console does not display the deployed servlets, JSPs, or EJBs directly on the console. However, you can use the console to display XML deployment descriptors for the enterprise application, Web modules, and EJB modules.

To view the application deployment descriptor for an application, do the following steps:

1. Select **Applications Application Types WebSphere enterprise applications**.
2. Click the application that you are interested in.
3. Under the Configuration tab, select **View Deployment Descriptor** under Detail Properties.

Figure 6-32 shows the deployment descriptor window for the PlantsByWebSphere enterprise application. The Configuration tab shows you the structure defined by the deployment descriptor:

- ▶ The name of the enterprise application
- ▶ The Web modules and their context roots
- ▶ The EJB modules and their associated JAR files
- ▶ The security roles associated with the enterprise application

The screenshot shows a software interface titled "Enterprise Applications". Under "Enterprise Applications", it lists "PlantsByWebSphere > Application servers > Deployment Descriptor". A sub-instruction says "Expand and collapse the application deployment descriptor data to view." Below are two buttons: "Expand All" and "Collapse All". The XML code for the deployment descriptor is displayed:

```
<application version="5"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/application_5.xsd" >
<display-name> PlantsByWebSphere </display-name>
<module id="WebModule_1" >
<web>
<web-uri> PlantsByWebSphere.war </web-uri>
<context-root> /PlantsByWebSphere </context-root>
</web>
</module>
<module id="WebModule_2" >
<web>
<web-uri> PlantsGallery.war </web-uri>
<context-root> /PlantsByWebSphere
<docs> </context-root>
</web>
</module>
<module id="EjbModule_1" >
<ejb> PlantsByWebSphereEJB.jar </ejb>
</module>
<security-role>
<description> Samples Administrator </description>
<role-name> SampAdmin </role-name>
</security-role>
</application>
```

Figure 6-32 Enterprise application deployment descriptor

Viewing EJB modules

To see the EJBs that are part of an enterprise application:

1. Select **Applications Application Types WebSphere enterprise applications**.
2. Click the application name.
3. Select **Manage Modules** under Modules Items.
4. Click the EJB module that you want to view (Figure 6-33).

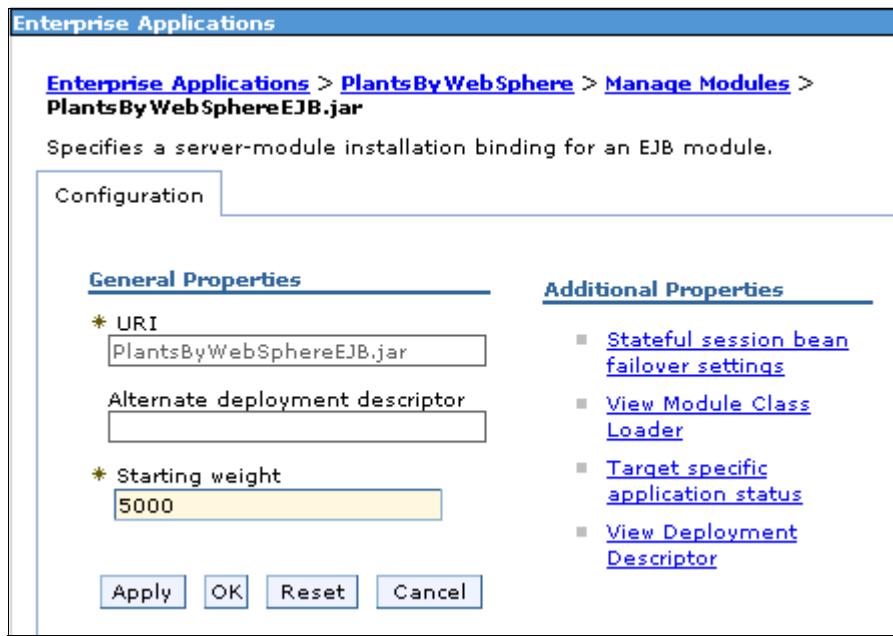


Figure 6-33 Viewing an EJB module configuration

5. Click **View Deployment Descriptor** under Additional Properties to see the EJB deployment descriptor.

Viewing Web modules

To see the servlets and JSPs that are part of an enterprise application:

1. Select **Applications Application Types WebSphere enterprise applications**
2. Click the application name.
3. Select **Manage Modules** under Modules.

4. Click the Web module you want to view (Figure 6-34).

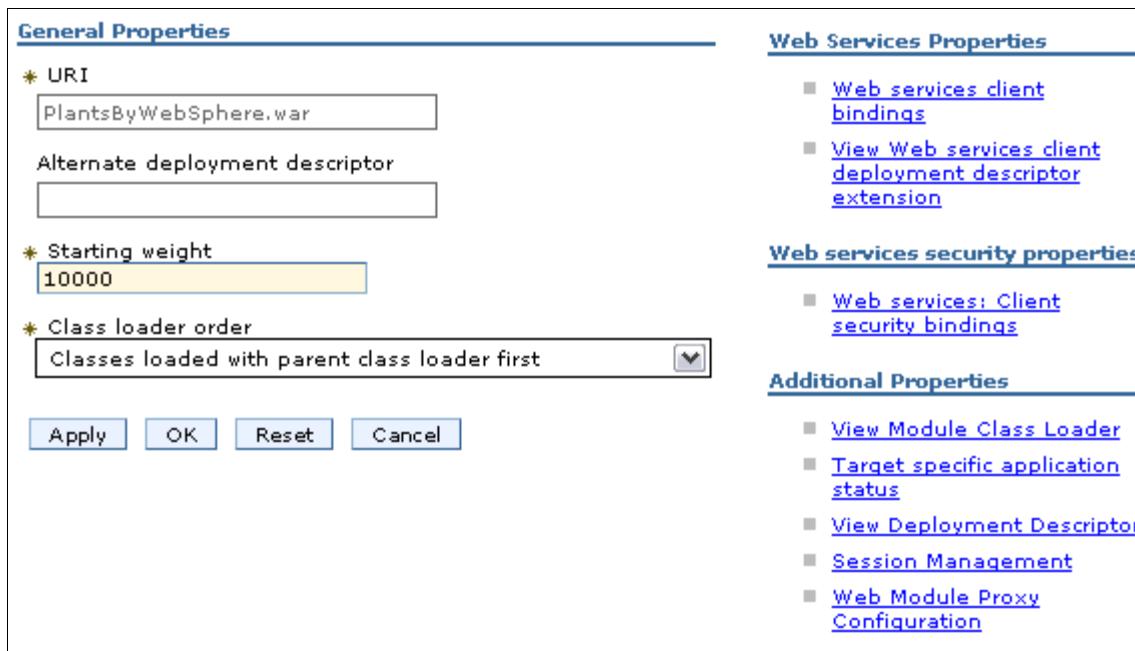


Figure 6-34 View a Web module

5. Click **View Deployment Descriptor** to see the details of the Web module content.

6.9.8 Finding a URL for a servlet or JSP

The URL for a servlet or JSP is the path used to access it from a browser. The URL is partly defined in the deployment descriptor provided in the EAR file and partly defined in the deployment descriptor for the Web module containing the servlet or JSP.

To find the URL for a servlet or JSP:

1. Find the context root of the Web module containing the servlet.
2. Find the URL for the servlet.
3. Find the virtual host where the Web module is installed.
4. Find the aliases by which the virtual host is known.
5. Combine the virtual host alias, context root, and URL pattern to form the URL request of the servlet/JSP.

For example, to look up the URL for the snoop servlet:

1. Find the context root of the Web module DefaultWebApplication of the DefaultApplication enterprise application. This Web module contains the snoop servlet.
 - a. Select **Applications Application Types WebSphere enterprise applications**.
 - b. Click the application that you are interested in, in our case, **DefaultApplication.ear**.
 - c. On the Configuration tab, select **Context Root for Web Modules** (Figure 6-36). You can see that:
 - i. There is only one Web module in this application, Default Web Application.
 - ii. The context root is “/”. We will use this later.

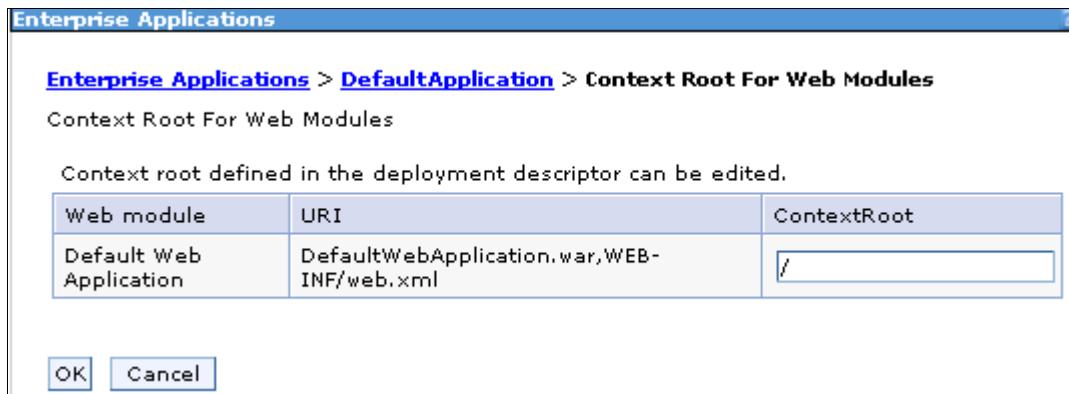


Figure 6-35 Context root for the Web modules in DefaultApplication

- d. Click **OK** to return to the DefaultApplication configuration.
2. Find the URL for the snoop servlet:
 - a. In the DefaultApplication configuration page, select **Manage Modules**.
 - b. Click the **Default Web Application** Web module to see the general properties.
 - c. Click **View Deployment Descriptor**.

This displays the Web module properties window, as shown in Figure 3-60. Note that the URL pattern for the snoop servlet starting from the Web module context root is “/snoop/*”. The Web module context root was “/”.

[Enterprise Applications](#) > [DefaultApplication](#) > [Manage Modules](#) >
[DefaultWebApplication.war](#) > **Deployment Descriptor**

Expand and collapse the application deployment descriptor data to view.

[Expand All](#)

[Collapse All](#)

```
<web-app id="WebApp_ID" >
    <display-name> Default Web Application</display-name>
    <description> This is the IBM WebSphere Application Server Default Web Application.</description>
    + <servlet id="Servlet_1" >
    + <servlet id="Servlet_2" >
    + <servlet id="Servlet_3" >
    - <servlet-mapping id="ServletMapping_1" >
        <servlet-name> Snoop Servlet</servlet-name>
        <url-pattern> /snoop/*</url-pattern>
    </servlet-mapping>
    - <servlet-mapping id="ServletMapping_2" >
        <servlet-name> Hello Pervasive Servlet</servlet-name>
        <url-pattern> /hello</url-pattern>
    </servlet-mapping>
    - <servlet-mapping id="ServletMapping_3" >
        <servlet-name> Hit Count Servlet</servlet-name>
        <url-pattern> /hitcount</url-pattern>
    </servlet-mapping>
    + <welcome-file-list id="WelcomeFileList_1" >
    + <error-page id="ErrorPage_1" >
    + <security-constraint id="SecurityConstraint_1" >
    + <security-role id="SecurityRole_1" >
    + <ejb-ref id="EjbRef_1" >
</web-app>
```

Figure 6-36 DefaultWebApplication Web module deployment descriptor

- d. Note that as you navigate through the windows, a navigation path is displayed below the Messages area. Click **DefaultApplication.ear** to return to the application configuration page.
3. Find the virtual host where the Web module is installed:
 - a. In the DefaultApplication configuration page, select **Virtual hosts** under Web Module Properties.

This will display all of the Web modules contained in the enterprise application, and the virtual hosts in which they have been installed. See Figure 6-37. Note that the Default Web Application Web module has been installed on the default_host virtual host.

Enterprise Applications > DefaultApplication > Virtual hosts		
Virtual hosts		
Specify the virtual host where you want to install the Web modules that are contained in your application. You can install Web modules on the same virtual host or disperse them among several hosts.		
<input checked="" type="checkbox"/>	Apply Multiple Mappings	
 		
Select	Web module	Virtual host
<input type="checkbox"/>	Default Web Application	<input type="button" value="default_host ▾"/>

Figure 6-37 List of virtual hosts

4. Find the host aliases for the default_host virtual host:
 - a. From the console navigation tree, select **Environment** → **Virtual Hosts**.
 - b. Click **default_host**.
 5. Select **Host Aliases** under Additional Properties:

This shows the list of aliases by which the default_host virtual host is known. See Figure 6-38.

Virtual Hosts > default_host > Host Aliases		
A list of one or more DNS aliases by which the virtual host is known.		
<input checked="" type="checkbox"/> Preferences		
Select	Host Name ▲	Port ▲
<input type="checkbox"/>	*	9080
<input type="checkbox"/>	*	80
<input type="checkbox"/>	*	9443
Total 3		

Figure 6-38 Default_host virtual host aliases

Note that the aliases are composed of a DNS host name and a port number. The host aliases for the default_host virtual host are *:80, *:9080 and *:9443, “*” meaning any host name.

6. Combine the virtual host alias, context root and URL pattern to form the URL request of the snoop servlet. Requests for the servlet with any of the following URLs will map to the default_host virtual host:
 - `http://hostname:80/snoop`
 - `http://hostname:9080/snoop`
 - `https://hostname:9443/snoop`

6.10 Enabling process restart on failure

WebSphere Application Server does not have either:

- ▶ A nanny process to monitor whether the AdminServer process is running, and restart it if the process has failed
- ▶ An AdminServer process to monitor whether each application server process is running, and restart it if the process has failed

Instead, WebSphere Application Server uses the native operating system functionality to restart a failed process. Refer to the following sections that discuss your operating system.

Windows

The administrator can choose to register one or more of the WebSphere Application Server processes on a machine as a Windows service during profile creation, or after by using the **WASService** command. With this command, Windows then automatically attempt to restart the service if it fails.

Syntax

Enter **WASService.exe** with no arguments to get a list the valid formats. See Example 6-24.

Example 6-24 WASService command format

```
Usage: WASService.exe (with no arguments displays this help)
      || -add <service name>
          -serverName <Server>
          -profilePath <Server's Profile Directory>
              [-wasHome <Websphere Install Directory>]
              [-configRoot <Config Repository Directory>]
              [-startArgs <additional start arguments>]
              [-stopArgs <additional stop arguments>]
```

```

[-userid <execution id> -password <password>]
[-logFile <service log file>]
[-logRoot <server's log directory>]
[-encodeParams]
[-restart <true | false>]
[-startType <automatic | manual | disabled>]
|| -remove <service name>
|| -start <service name> [optional startServer.bat
parameters]
|| -stop <service name> [optional stopServer.bat
parameters]
|| -status <service name>
|| -encodeParams <service name>

```

Be aware of the following considerations when using the **WASService** command:

- ▶ When adding a new service, the **-serverName** argument is mandatory. The **serverName** is the process name. If in doubt, use the **serverstatus -all** command to display the processes. For a deployment manager, the **serverName** is dmgr, for a node agent it is nodeagent, and for a server, it is the server name.
- ▶ The **-profilePath** argument is mandatory. It specifies the home directory for the profile.
- ▶ Use unique service names. The services are listed in the Windows Services control window as:

IBM WebSphere Application Server V7.0 - <service name>

The convention used by the Profile Management Tool is to use the node name as the service name for a node agent. For a deployment manager, it uses the node name of the deployment manager node concatenated with dmgr as the service name.

Examples

Example 6-25 shows using the **WASService** command to add a node agent as a Windows service.

Example 6-25 Registering a deployment manager as a Windows service

```
C:\WebSphereV7\AppServer\bin>WASService -add "node40b" -serverName
nodeagent -profilePath "C:\websphrev7\appserver\profiles\node40b"
-restart true
Adding Service: node40b
Config Root: C:\websphrev7\appserver\profiles\node40b\config
Server Name: nodeagent
Profile Path: C:\websphrev7\appserver\profiles\node40b
Was Home: C:\WebSphereV7\AppServer\
```

```
Start Args:  
Restart: 1  
IBM WebSphere Application Server V7.0 - node40b service successfully  
added.
```

Note that the service name added in Figure 6-39 will be IBM WebSphere Application Server V7.0, concatenated with the name you specified for the service name.

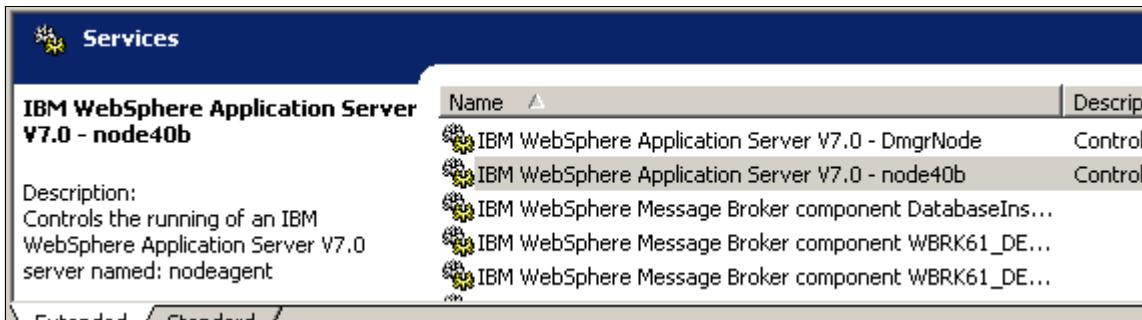


Figure 6-39 New service

If you remove the service using the **WASService -remove** command, specify only the latter portion of the name, as in Example 6-26.

Example 6-26 Removing a service

```
C:\WebSphereV7\AppServer\bin>WASService -remove "node40b"  
Remove Service: node40b  
Successfully removed service
```

UNIX and Linux

The administrator can choose to include entries in inittab for one or more of the WebSphere Application Server processes on a machine, as shown in Example 6-27. Each such process will then be automatically restarted if it has failed.

Example 6-27 Inittab contents for process restart

On deployment manager machine:

```
ws1:23:respawn:/usr/WebSphere/DeploymentManager/bin/startManager.sh
```

On node machine:

```
ws1:23:respawn:/usr/WebSphere/AppServer/bin/startNode.sh
```

```
ws2:23:respawn:/usr/WebSphere/AppServer/bin/startServer.sh nodename_server1
```

```
ws3:23:respawn:/usr/WebSphere/AppServer/bin/startServer.sh nodename_server2  
ws4:23:respawn:/usr/WebSphere/AppServer/bin/startServer.sh nodename_server2
```

Note: When setting the action for startServer.sh to respawn in /etc/inittab, be aware that init will always restart the process, even if you intended for it to remain stopped. As an alternative, you can use the rc.was script located in \${WAS_HOME}/bin, which allows you to limit the number of retries.

The best solution is to use a monitoring product that implements notification of outages and logic for automatic restart.

z/OS

WebSphere for z/OS takes advantage of the z/OS Automatic Restart Management (ARM) to recover application servers. Each application server running on a z/OS system (including servers you create for your business applications) are automatically registered with an ARM group. Each registration uses a special element type called SYSCB, which ARM treats as restart level 3, ensuring that RRS (It is a z/OS facility that provides two-phase sync point support across participating resource managers) restarts before any application server.

Note: If you have an application that is critical for your business, you need facilities to manage failures. z/OS provides rich automation interfaces, such as automatic restart management, that you can use to detect and recover from failures. The automatic restart management handles the restarting of servers when failures occur.

Some important things to consider when using automatic restart management:

- ▶ If you have automatic restart management (ARM) enabled on your system, you might want to disable ARM for the WebSphere Application Server for z/OS address spaces before you install and customize WebSphere Application Server for z/OS. During customization, job errors might cause unnecessary restarts of the WebSphere Application Server for z/OS address spaces. After installation and customization, consider enabling ARM.
- ▶ If you are ARM-enabled and you cancel or stop a server, it will restart in place using the **armrestart** command.
- ▶ It is a good idea to set up an ARM policy for your deployment manager and node agents. For more information about how to change the ARM policies, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.installation.zseries.doc/info/zseries/ae/cins_changearm.html

- ▶ If you start the location service daemon on a system that already has one, it will terminate.
- ▶ Every other server will come up on a dynamic port unless the configuration has a fixed port. Therefore, the fixed ports must be unique in a sysplex.
- ▶ If you issue STOP, CANCEL, or MODIFY commands against server instances, be aware of how automatic restart management behaves regarding WebSphere Application Server for z/OS server instances; Table 6-1 on page 379 depicts the behavior of ARM regarding WebSphere Application Server for z/OS server instances.

Table 6-1 Behavior of ARM and WebSphere Application Server for z/OS server instances

When you issue	ARM behavior
STOP address_space	It will not restart the address space.
CANCEL address_space	It will not restart the address space.
CANCEL address_space, ARMRESTART	It will restart the address space.
MODIFY address_space, CANCEL	It will not restart the address space.
MODIFY address_space, CANCEL,ARMRESTART	It will restart the address space.

For more information about how to activate the ARM, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.installation.zseries.doc/info/zseries/ae/tins_activearm.html

Let us say you have activated ARM and want to check the status of address spaces registered for automatic restart management; in order to get this information, you need to:

1. Initialize all servers.
2. Issue one or both of the commands shown in Example 6-28.

Example 6-28 Displaying the status of address spaces registered for automatic restart management

To display all registered address spaces (including the address spaces of server instances), issue the command:

d xcf,armstatus,detail

To display the status of a particular server instance, use the display command and identify the job name. For example, to display the status of the Daemon server instance (job BBODMN), issue the following command:

d xcf,armstatus,jobname=bbodmn,detail

For more information about how to use the **display** command, refer to:

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



WebSphere Application Server for z/OS

In this chapter, we concentrate on the features of WebSphere Application Server V7 for z/OS.

We cover the following topics:

- ▶ “WebSphere Application Server on z/OS Architecture” on page 382
- ▶ “WebSphere Application Server for z/OS operations” on page 392
- ▶ “Maintenance for the HFS” on page 401
- ▶ “Workload management” on page 405
- ▶ “What is new in V7 for z/OS” on page 414
- ▶ “Thread management using the workload profile” on page 429
- ▶ “Local connectivity to DB2” on page 432
- ▶ “Migrating to V7” on page 438

7.1 WebSphere Application Server on z/OS Architecture

In this section we show the added value that the implementation for WebSphere Application Server for z/OS offers as compared to the distributed versions.

7.1.1 Architecture of a single application server on z/OS

This section gives a conceptual view of an application server inside WebSphere Application Server for z/OS.

Overview

WebSphere Application Server on distributed platforms is based on a single process model. This means that the entire application server runs in a single process, which contains the Java Virtual Machine (JVM). If this process crashes for some reason, all applications that are deployed to this application server will be unavailable unless the application server is clustered.

With WebSphere Application Server for z/OS, a logical application server can consist of multiple JVMs, each executing in a different address space. These address spaces are called servant regions (SR), each containing one JVM. If a servant region abends, another servant region can take over the incoming requests in an multiple-servant environment.

In fact, each logical application server on z/OS has cluster capabilities through the use of multiple servants. These mini clusters benefit from cluster advantages such as availability and scalability without the overhead of a real cluster. This is a key differentiator against distributed platforms.

With regard to administration WebSphere Application Server for z/OS uses the same concepts as distributed environments to create and manage application servers. However, each application server is consists of multiple address spaces that represent a single logical application server:

- ▶ Control region (CR)
- ▶ Servant region (SR)
- ▶ Control region Adjunct

At minimum, one application server consists of one control region and one servant region. Additional servant regions can be added statically by defining a minimum amount of servant regions. Defining a maximum amount of servants, that is higher than the minimum amount allows the z/OS Workload Manager (WLM) to add more servants dynamically according to the demand of system resources. In practice the amount of servant regions is limited by the physical memory available on the system.

The main responsibility of the control region is to handle the incoming connections from the clients and dispatch the request to the WLM queues. Therefore the control region is equipped with its own JVM. Only IBM written code runs in this JVM, as opposed to the servant region, where the application code runs. There is only one control region per application server. In the unlikely case that the servant region abends, the incoming requests cannot be handled anymore. In this case availability can only be assured with a real clustered server. Because the control region only runs IBM authorized code, this case is a lot more unlikely than a crashing servant region.

Figure 7-1 illustrates these concepts.

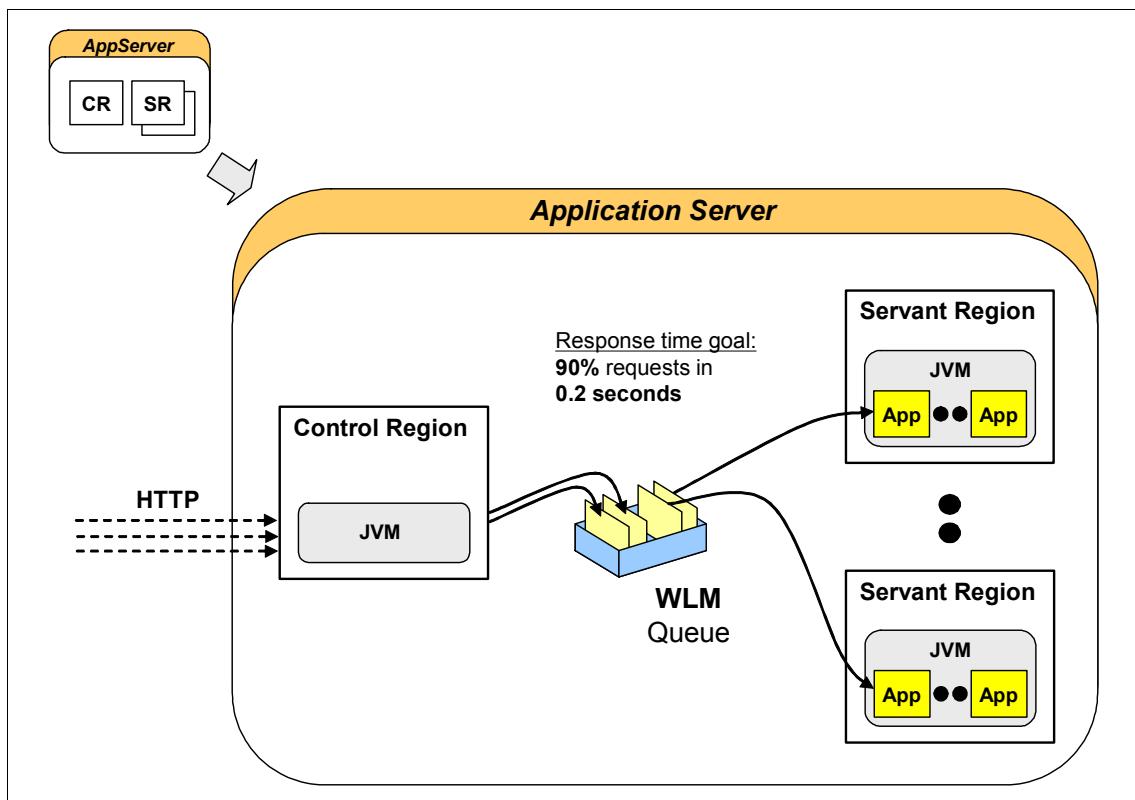


Figure 7-1 Architecture on a single application server

The z/OS Workload Manager (WLM) allows you to prioritize work requests on a transaction granularity, compared to server granularity on a distributed environments. Therefore, a service class will be assigned to each work request. For instance, you can define a service class in WLM that has the goal to complete 90% of the requests within 0.2 seconds. The WLM tries to achieve this goal with all available resources and if response times of user transactions do not meet the defined goals, the WLM starts additional servant regions to process the incoming work requests. In other words, the WLM gives you the opportunity to define Service Level Agreements in the form of service classes. Because the WLM is necessary for all subsystems within z/OS this is a huge advantage compared to distributed environments.

A WLM queue is used to queue work for further processing. Each WLM queue uses a first-in-first-out mechanism and represents a service class. Servant regions are bound to a certain priority and therefore take work from the queue with the priority they are bound to. In other words one servant region can only serve one service class, but one service class can be served by multiple servants. If additional service classes are defined, additional servant regions are required to process the workload.

All the possible profiles that can be instantiated on z/OS, are built using a control region and servant region:

- ▶ Application server
- ▶ Deployment manager
- ▶ Job manager
- ▶ Administrative agent

Although an application server is based on multiple components, it is still a single instance of a server from an application developer, system administrator or user perspective. This means that nearly all WebSphere variables are defined against the server, not each component of the server. However, some of the settings, such as heap sizes, have to be defined for each of component separately (control region, servant region and adjunct region) as shown in Figure 7-2. In the administrative console under **Application servers → appserver → Process definition**, you can define the heap size for each component.

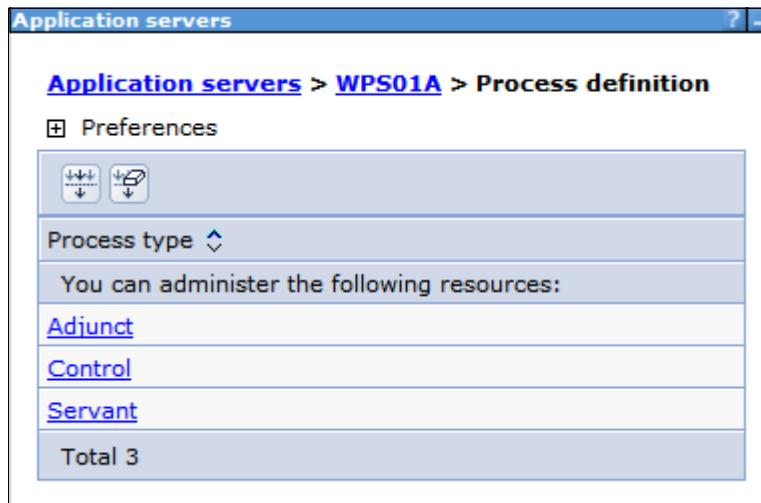


Figure 7-2 JVM settings of CR, SR, and Adjunct in the administrative console

Basically, the servant specific settings in the administrative console apply for all servants. For instance, if the servant minimum heap size is set to 256 megabyte in the administrative console, all servants will have a JVM with this minimum heap size. There is no possibility to differentiate between different servants.

Attention: This difference in platform settings also applies to `wsadmin` scripting. The major difference between scripting on WebSphere for z/OS versus distributed operating systems is that WebSphere for z/OS has multiple JVMs within one logical application server (control, servants, and adjunct). On distributed platforms, the `wsadmin` command to change the heap size of a particular application server is unique to the server JVM, whereas it is not on WebSphere for z/OS.

Control region

The control region is the face of the application server to the outside world. It is the only component that is reachable externally using standard protocols and port communication. For communication with the servant regions, where the application is executed, the control region is the end point for TCP transportation and switches to WLM queues.

Here are some facts to keep 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 the end point for communication with clients.

Control region adjunct

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

Servant region

The servant region is the component of an application server where the actual application runs and transactions are processed in a JVM. The EJB and Web container are in the servant region.

Through the use of multiple servants, it is possible to actually benefit from cluster advantages without the overhead of a real cluster. For additional robustness, 24x7 availability, and scalability, we recommend that you build an application server cluster that integrates these mini clusters. You still can use multiple servant regions for each cluster member.

Here are the most important things to note 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 utilize the multi-servant concept.
- ▶ The number of servant regions is transparent to the user and the application.
- ▶ Servant regions can dynamically be started by the WLM component, if response times of user transactions do not meet the defined goals. However, 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 utilize them all.

Changing the number of servants

The number of servants to be started initially and dynamically through WLM can be defined in the following path of the administrative console: **Application Servers** → **app_server** → **Java and Process Management** → **Server Instance**. See Figure 7-3.

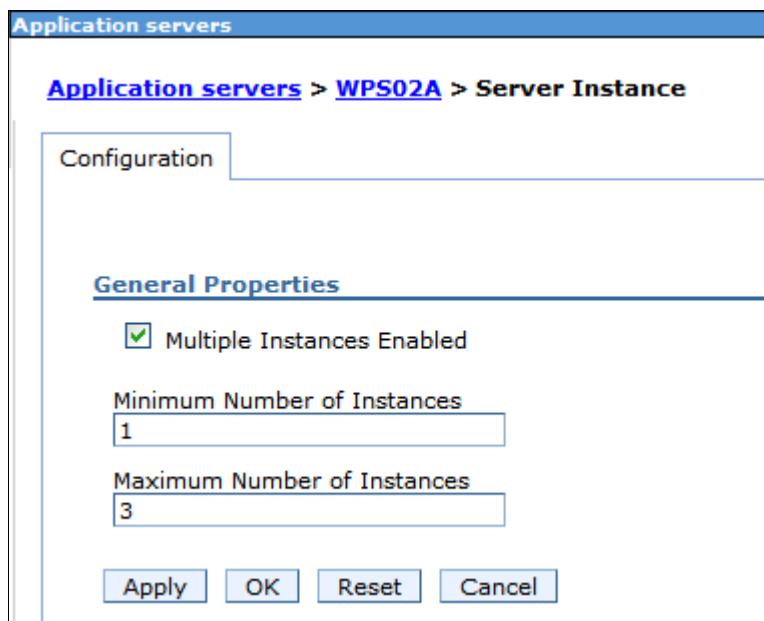


Figure 7-3 Setting the minimum and maximum amount of servants

In this panel, the minimum and maximum amount of servants can be defined. To use more than one servant region, the check box “**Multiple Instances Enabled**” must be checked.

The minimum amount of servants will be started initially during startup of the application server. If response times of user transactions do not meet the defined goals, the WLM can start additional servant regions. Therefore the maximum amount of servants defined must be higher than the minimum amount. If both numbers are the same, the WLM will not be able to start additional servant regions dynamically. Moreover, the WLM can only start additional servants dynamically until the maximum number of servants is achieved.

The default setting is minimum instances = 1 and maximum instances = 1 with “Multiple Instances Enable” unchecked. In this configuration the mini cluster will not be used.

The following example shows an SDSF panel with all active started tasks (STCs) of a complete cell, including a deployment manager, daemon, node agent and two application servers WPS01A and WPS02A. The jobname for all application server components (control region and servant region) starts with the same characters, but the servant region has an additional name suffix “S”. Another way to distinguish the components is to look at the procedure step (ProcStep) which is BBOPACR for the control region and BBOPASR for the servant region. Furthermore, server WPS02A consists of two servants, which have the same STC name WPS02AS.

Example 7-1 SDSF output showing the started tasks for a cell

SDSF DA SC04 SC04 PAG 0 CPU/L/Z 81/ 40/ 0 LINE 1-10 (10)									
COMMAND INPUT ==> SCROLL ==> PAGE									
NP	JOBNAME	StepName	ProcStep	JobID	Owner	C	Pos	DP	Real Paging SIO
	WPDEMN	WPDEMN	BBODAEMN	STC13150	WPDMGR	NS	FE	5332	0.00 0.00
	WPDPMGR	WPDPMGR	BBOPDCR	STC13145	WPDPMGR	NS	F6	91T	0.00 0.00
	WPDMGRS	WPDMGRS	BBOPDSR	STC13152	WPDMGRS	IN	F4	175T	0.00 0.00
	WPAGNTA	WPAGNTA	BBOPACR	STC13151	WPACR	NS	F6	62T	0.00 0.00
	WPS01A	WPS01A	BBOPACR	STC13154	WPACR	NS	F6	58T	0.00 0.00
	WPS01AS	WPS01AS	BBOPASR	STC13155	WPASR	IN	F4	98T	0.00 0.00
	WPS02A	WPS02A	BBOPACR	STC13333	WPACR	NS	F6	86T	0.00 0.00
	WPS02AS	WPS02AS	BBOPASR	STC13334	WPASR	IN	F4	87T	0.00 0.00
	WPS02AS	WPS02AS	BBOPASR	STC13335	WPASR	IN	F4	88T	0.00 0.00

As a guideline for problem determination, communication errors with the client should appear in the log of the control region, whereas application related errors should appear in the log of the servant region.

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. However, 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. In addition, 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 very low. Only WebSphere Application Server for z/OS product code is executed in this region. To make applications available, 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, as long as the system has enough resources available.
- ▶ You can set very detailed performance targets on a transactional level for the response time. The system will automatically adjust resources on a 7x24x365 base trying to make sure that these goals are kept.

Through the usage of multiple JVMs with smaller heap sizes the penalty a single transaction has to pay during garbage collection, will decrease and therefore the general throughput will increase.

7.1.2 Cell architecture of WebSphere Application Server for z/OS

This section describes the cell architecture of the WebSphere Application Server for z/OS.

Overview

A cell can span multiple z/OS images and in a heterogeneous cell; even a mixture of z/OS images and distributed platforms are possible. In contrast, a node is always dedicated to a certain z/OS image, because each node has its own file system.

The deployment manager is a specialized application server in a distributed environment. It has its own deployment manager node and hosts the administrative console (administrative console). As with an application server, the same concept of CR and SR applies to the deployment manager as well, but the deployment manager is limited to one servant region, as illustrated in Figure 7-4. The administrative console J2EE application, itsclite, runs in the JVM of the servant region.

In a standalone application environment, the administrative console is deployed into the application server.

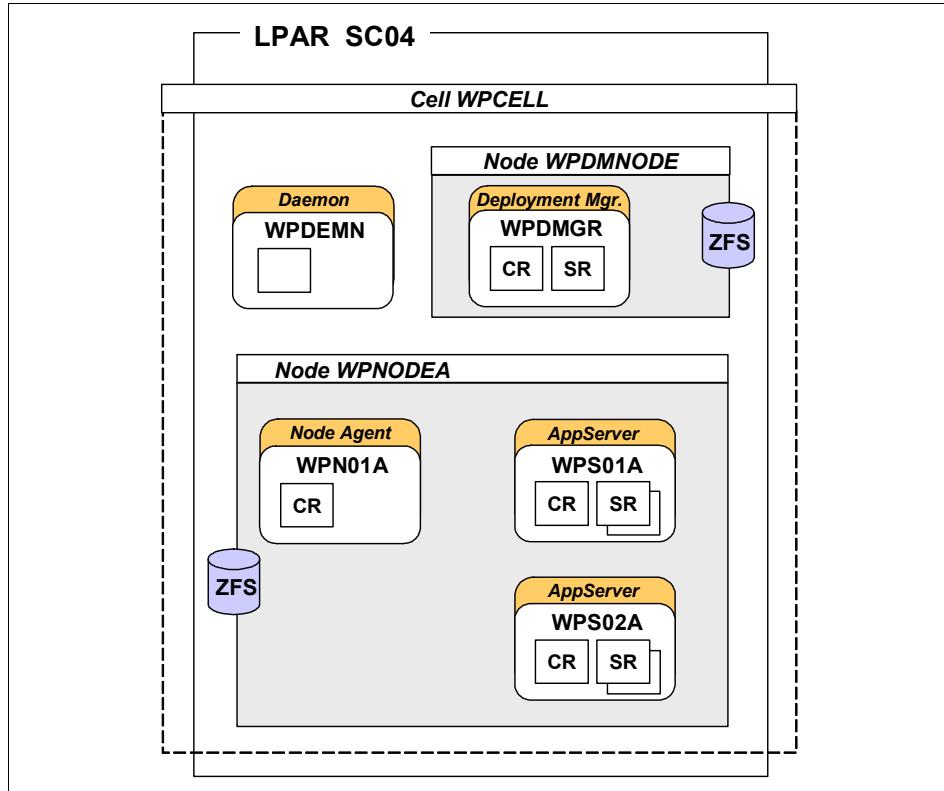


Figure 7-4 Architecture of a cell

WebSphere Application Server for z/OS introduces a WebSphere cell component exclusive to the z/OS platform: the location service daemon (LSD). In WebSphere Application Server for z/OS terminology the Location Service Agent is called daemon. It provides the location name service for external clients. For instance this service is used by clients to identify a target EJB within a cell. Another task of the daemon is to provide access to modules in storage for all servers within the cell on the same system.

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, one daemon will be created for each cell. The daemon is started automatically when the first control region for the cell on that z/OS image is started. If you terminate a daemon with a MVS STOP command, all the WebSphere Application Server components belonging to the cell on that z/OS image shut down. This is the easiest way to shut down all components of a cell on a specific system. The daemon is created as a part of the customization process. Moreover, within a WebSphere z/OS cell the daemon is the only address space that is not equipped with its own JVM.

In a distributed environment, one node agent will be created for each node. It is the responsibility of the node agent to synchronize the node configuration with the master repository of the deployment manager. Moreover, the node agent administers the application servers belonging to the same node. A node agent consists of one control region; it does not require a servant region.

In a WebSphere Application Server for z/OS distributed environment with one application server, there will be a minimum of six address spaces:

- ▶ Deployment manager control region
- ▶ Deployment manager servant region
- ▶ Location Service daemon (daemon)
- ▶ Application server control region
- ▶ Application server servant region
- ▶ Node agent
- ▶ (Optional) Application server control region adjunct

A stand-alone server installation would consist of at least three address spaces:

- ▶ Location Service daemon (daemon)
- ▶ Application server control region
- ▶ Application server servant (assumed that one servant is used)
- ▶ (Optional) Application server control region adjunct

Making the deployment manager mobile

In a cell that spans multiple z/OS systems, the deployment manager is located in only one of those systems. In the case of an outage of the system where the deployment manager is located, it would be helpful if the deployment manager can be restarted on other systems of the same sysplex.

The white paper, *Making the DMGR Mobile*, provides step-by-step instructions for implementing a mobile deployment manager using the capabilities of the sysplex distributor. This paper is available at:

<http://www-03.ibm.com/support/techdocs/atstrmstr.nsf/WebIndex/WP101140>

Introducing job manager and secure proxy server for z/OS

A job manager is a specialized base application server on z/OS, which can efficiently administrate multiple cells on z/OS. The Job Manager has been introduced with WebSphere Application Server for z/OS V7. The white paper, *Introducing The WebSphere V7 Job Manager for z/OS*, from the Washington Systems Center (WSC) provides step-by-step instructions to implement a job manager on z/OS. This paper is available at:

<http://www-03.ibm.com/support/techdocs/atstrmstr.nsf/WebIndex/WP101341>

In addition, the Secure Proxy Server for z/OS has been introduced with WebSphere Application Server for z/OS V7. The Secure Proxy Server is a special security hardened proxy server for use in a Demilitarized Zone (DMZ). To use the Secure Proxy Server on z/OS, a DMZ must be implemented on z/OS using a firewall.

The white paper, *Introducing the WebSphere V7 Secure Proxy Server for z/OS*, provides the information required to set up a Secure Proxy for z/OS. This paper is available at:

<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101423>

7.2 WebSphere Application Server for z/OS operations

In this section we discuss various aspects of operations in a WebSphere Application Server for z/OS environment that are unique to z/OS.

7.2.1 Structure of the configuration HFS

In WebSphere Application Server for z/OS, the product HFS is strictly separated from the configuration HFS. As the name indicates, the configuration HFS contains the complete configuration of one node, consisting of many XML configuration files. These files are encoded in ASCII. The configuration is mainly located in the *install_root/profiles/default* directory of the configuration HFS. Symbolic links in the configuration HFS, for instance in the *install_root/bin* directory, refer to files in the product HFS.

In contrast, the product HFS contains all necessary product files such as shell scripts, which can be shared across multiple nodes. The product HFS is maintained by System Modification Program/Extended (SMP/E).

During the customization process of a new node, you can either select an HFS or ZFS for the configuration file system. We recommend that you use ZFS for the configuration file system, because the performance of ZFS compared to HFS is significantly better and the possibilities to extend a ZFS file system are more flexible.

Attention: If you have a sysplex environment with multiple systems and you are using a shared USS file system, you need to be aware that the configuration file system is always mounted to the system where the corresponding started tasks are running. Otherwise this can have a major impact on performance, because the configuration file system always needs to be mounted in RDWR mode. If the configuration file system is mounted on a different system, the XCF traffic will increase dramatically in order to negotiate the writing permissions with the other systems.

The product file system can be shared across the sysplex. In order to avoid a performance impact, the product HFS / ZFS should be mounted in READ mode.

We recommend that you use one configuration dataset for each node, instead of using the same dataset for multiple nodes. There are several reasons for separating each node into a different dataset. Probably the most important one is, that the migration from one major version to another major version (for instance V6.1 to V7) is done by a node-by-node basis. For availability and performance reasons it is better to separate each node configuration by using different datasets.

Node synchronization

In a distributed environment the synchronization of nodes is important, especially if the security configuration has been changed.

The configuration file system of the deployment manager contains the master repository, which includes the XML configuration files of the complete cell. The master repository is located in <DMGR HOME>/profiles/default/config. In addition, each node has its own configuration file system where a local, reduced copy of the cell configuration and the complete copy of the node specific configuration files are stored. The master repository and configuration files for each node should always be in sync. For instance if important security changes are not propagated to the nodes, this can cause major communication problems between the deployment manager, node agents and application servers. It is the responsibility of the node agent to keep its own node file system in sync as illustrated in Figure 7-5.

By clicking “**Save**” after a change to an application server setting in the administrative console, the corresponding XML configuration files in the master repository will be changed first. In order to take effect, the adapted configuration files need to be copied to the configuration file system of the corresponding node. During the process of synchronization the node agent connects via TCPIP to the deployment manager and requests all configuration files that have changed since the last synchronization. The updated configuration files will be copied into the local configuration file system by the node agent.

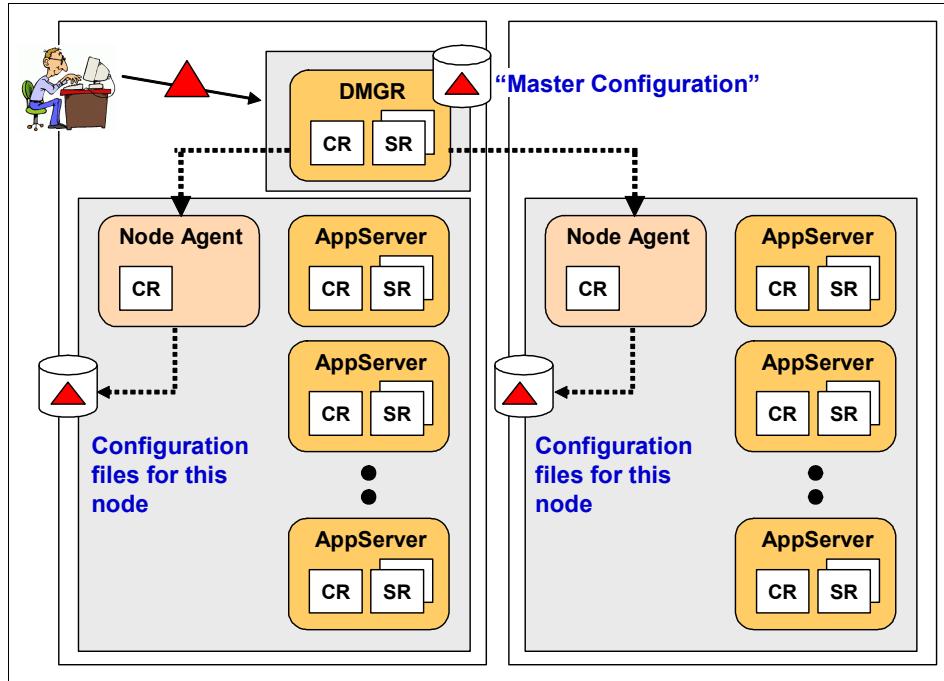


Figure 7-5 Node synchronization

Attention: In order to keep the consistency of the configuration file system, we recommend that you do not change any values directly in XML configuration files. A lot of settings require the change of multiple configuration files that are related to each other. If these dependencies are not taken into account, it is possible that the configuration settings are not in sync with each other anymore. Instead we recommend that you use the wsadmin shell with connection type “none”, which requires no running process and changes the configuration files directly while considering all dependencies.

7.2.2 Load module libraries in the HFS

In WebSphere Application Server V7, the load libraries are now included in the product HFS instead of separate MVS datasets. Consequently the load module libraries do not have to be added to the STEPLIB, LNKLST or LPA anymore. This simplifies the maintenance of WebSphere z/OS dramatically compared to previous versions because it eliminates the possibility of an accidental mismatch between the configuration HFS and the load module datasets. In previous versions, the configuration HFS always had to be in sync with the load module datasets.

Attention: In order for this function to work properly, the fixes of the following UNIX System Services APARs need to be applied:

- ▶ OA22093
- ▶ OA22094
- ▶ OA25489

These fixes provide an enhanced LOADHFS call to load the WebSphere modules directly from the HFS.

Within the configuration HFS, the load module libraries are now located in the directory *install_root/lib/modules* and begin the file name with “bbo”. In addition, the symbolic links in this directory point to <SMP/E_HOME>/lib/modules of the product HFS.

If you continue to use MVS datasets for the load modules, you can switch the configuration between using load modules in the file system and using load modules in a dataset with the **switchmodules** command, as described in the following Information Center article:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.installation.zseries.doc/info/zseries/ae/rins_switchmod.html

This major change also has implications for the start procedure JCLs, as described in the following section.

7.2.3 Changed start procedure JCLs with V7

In order to start a WebSphere Application Server component, such as a deployment manager, application server or node agent, different start procedures (JCLs) are used. As part of the customization process these procedures will be generated and copied into the PROCLIB. Some components such as the deployment manager and daemon have their own unique start procedure for each control region and servant region. It is common practice that all application servers belonging to the same node and the node agent share the same CR procedure and the same SR procedure as pointed out in Figure 7-6.

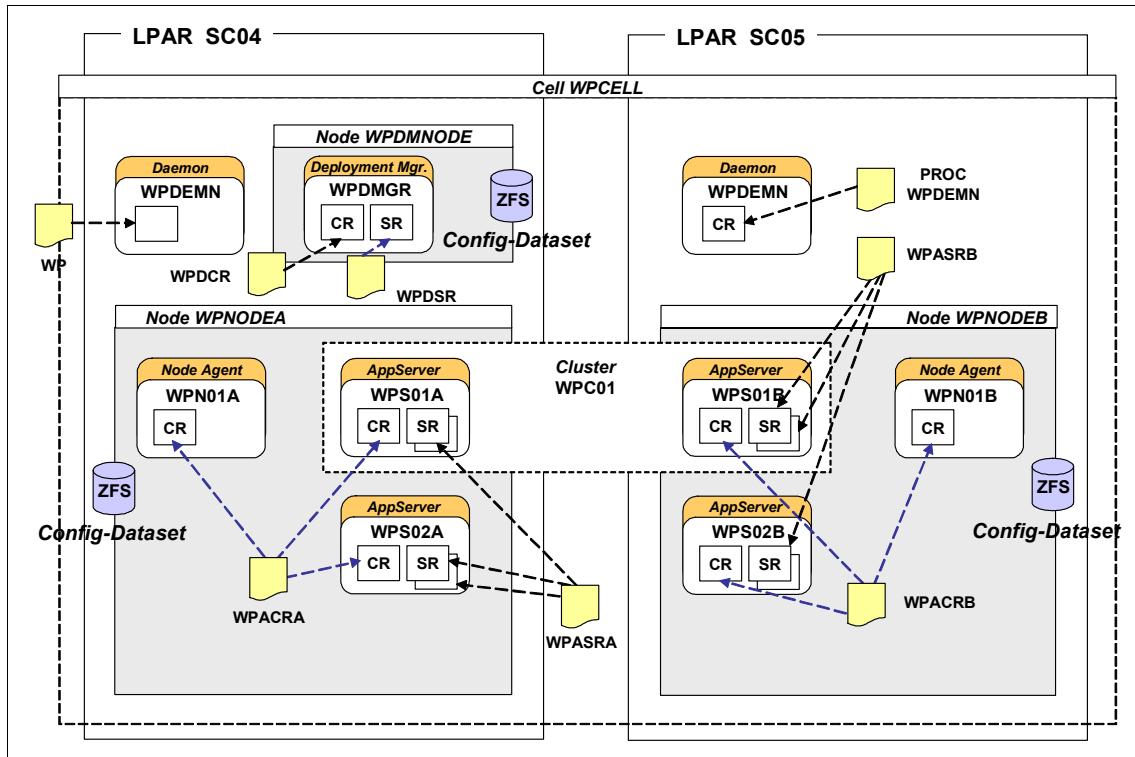


Figure 7-6 Reuse of PROCLIB members

The advantage of sharing the same procedures across all application servers of a node is that new application servers can be easily added using the administrative console without the need to define additional procedures. In addition, new start procedures usually require additional STARTED profiles in RACF.

The reason why the start procedures usually cannot be shared across multiple nodes is that the USS path to the WebSphere configuration file system is part of the procedure, as shown in Example 7-2. Theoretically, this can be realized by making use of additional variables for the configuration path. Consequently the MVS start command has to be extended for that new variable.

With WebSphere Application Server for z/OS V7, all load libraries are located in the USS file system instead of MVS datasets. Consequently, all start procedures have to be adapted in V7. Also, the customization job for the start procedures does not create the z-Member in PROCLIB anymore. In the past, this z-Member has been used for including STEPLIBs. Example 7-2 illustrates JCL where the STEPLIB entries for DB2 can be placed. They are marked in bold.

Example 7-2 Start procedure of application server CR

```
//WPACRA PROC ENV='',PARMS=' ',REC=N,AMODE=00
// SET ROOT='/wasconfig/wpcell1/wpnodea'
// SET FOUT='properties/service/logs/applyPTF.out'
// SET WSDIR='AppServer'
//*****
//** Test that OMVS can successfully launch a shell and return *
//*****
//TOMVS EXEC PGM=BPXBATCH,REGION=0M,
// PARM='SH exit 13'
//STDERR DD PATH='&ROOT./&ENV..HOME/&FOUT.',
// PATHOPTS=(OWRONLY,OCREATE,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//STDOUT DD PATH='&ROOT./&ENV..HOME/&FOUT.',
// PATHOPTS=(OWRONLY,OCREATE,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//*****
//** If the shell RC code is as expected (13) - proceed   *
//*****
//IFTST IF (RC = 13) THEN
//*****
//** Start the Multi-Product PTF Post-Installer          *
//*****
//APPLY EXEC PGM=BPXBATCH,REGION=0M,
// PARM='SH &ROOT./&ENV..HOME/bin/applyPTF.sh inline'
//STDERR DD PATH='&ROOT./&ENV..HOME/&FOUT.',
// PATHOPTS=(OWRONLY,OCREATE,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//STDOUT DD PATH='&ROOT./&ENV..HOME/&FOUT.',
// PATHOPTS=(OWRONLY,OCREATE,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//      IF (APPLY.RC <= 4) THEN
//*****
//** If the RC from the Post-Installer is LE 4 then start  *
//** the WebSphere Application Server                      *
//*****
//BBOPACR EXEC PGM=BPXBATA2,REGION=0M,TIME=MAXIMUM,MEMLIMIT=NOLIMIT,
// PARM='PGM &ROOT./&WSDIR./lib/bboctlm &AMODE. &PARMS. REC=&REC'
//STEPLIB DD DISP=SHR,DSN=DB8X8.SDSNEXIT
//      DD DISP=SHR,DSN=DB8X8.SDSNLOAD
//      DD DISP=SHR,DSN=DB8X8.SDSNL0D2
//STDENV DD PATH='&ROOT/&ENV/was.env'
//*
//** Output DDs
//*
//DEFALTDD DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//HRDCPYDD DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSOUT DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
```

```
//CEEDUMP DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE  
//SYSPRINT DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE  
//  
//      ENDIF  
//IFTSTEND ENDIF  
//
```

The Addressing Mode (AMODE) is a JCL parameter that is 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; however, we recommend *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, for example, with AMODE=64, and the XML files reflect a 31-bit installation (or vice versa), then the server will not start.

Note: We recommend that you use the default value for the AMODE (AMODE=00) in the startup JCL for the WebSphere Application Server components. However, keep in mind that you need to verify your automation settings.

7.2.4 Starting and stopping an application server

In order to start an application server, it is only necessary to issue the MVS start command for the control region (CR). The corresponding servant regions (SR) will be started by the WLM automatically.

An application server including the CR and all SRs can be stopped by executing the MVS stop command for the control region. Basically the same concept applies to the deployment manager.

Stopping a daemon address space will shut down all address spaces belonging to the same cell and z/OS system.

In Figure 7-7 the start order of the deployment manager is illustrated. The control region needs to be started first. If there is no daemon address space running, the CR will start the daemon address space. If the deployment manager CR is initialized, the WLM will start the minimum amount of servants. After the SRs have been started successfully, the “open for e-business” notification will appear in the job output of the CR. Next, the node agent CR can be started with the corresponding MVS start command.

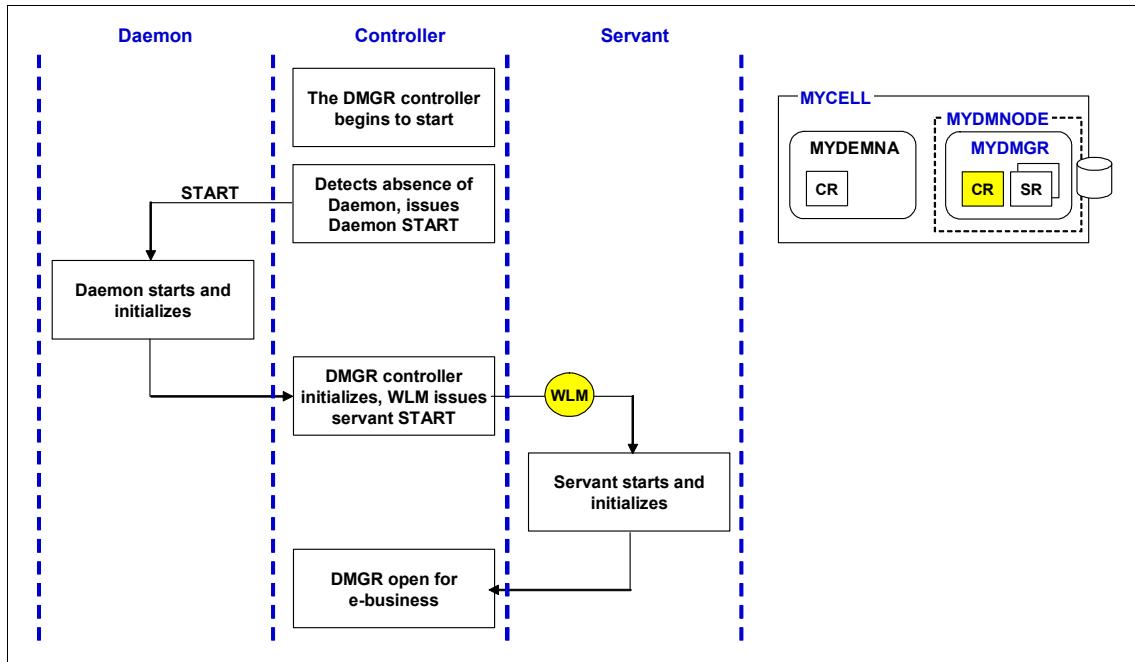


Figure 7-7 Address space start order

Start procedures can be shared between all application servers belonging to the same node. In order to determine which application server should be started, an ENV parameter needs to be specified in the MVS start command. This parameter is structured using the following qualifiers:

<cell_short_name>.<node_short_name>.<server_short_name>.

For most MVS people, this parameter seems to define an MVS dataset. But in this case, this parameter specifies a symbolic link that points to the configuration directory of the particular server. As illustrated in Figure 7-8, this symbolic link is located in the mount directory of the corresponding node. For every server belonging to the same node, a symbolic link will be created in this directory. During startup of a server, the was.env file in the server configuration directory is required, which contains the most important configurations for that server.

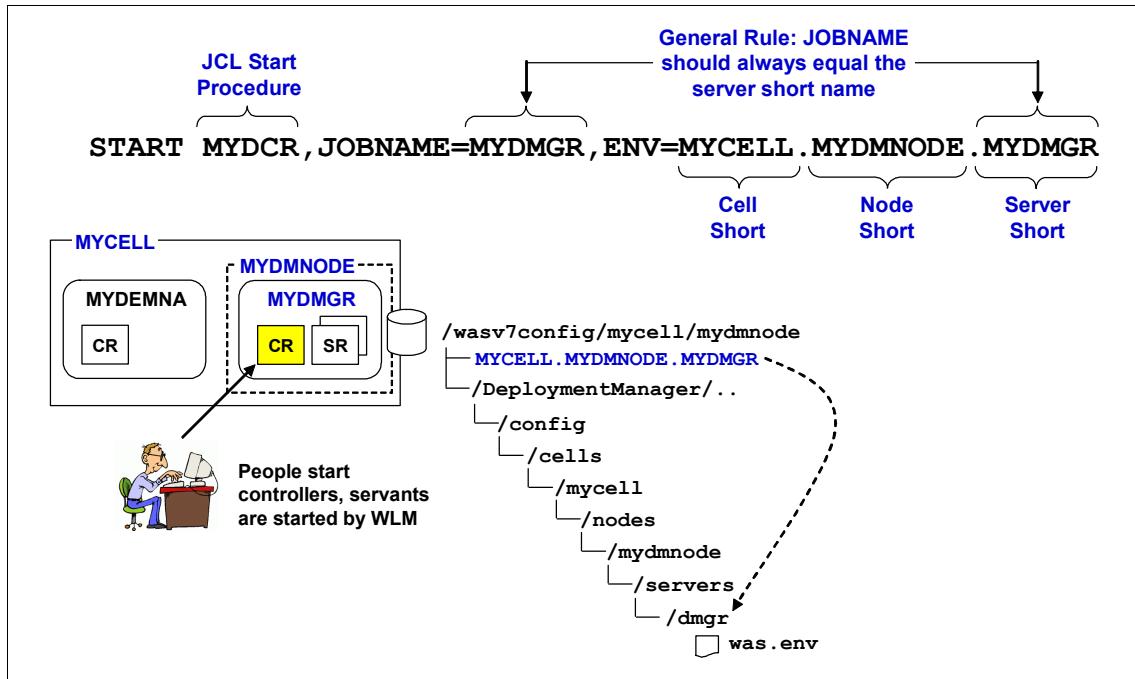


Figure 7-8 Start command for the deployment manager

Furthermore, this ENV parameter illustrates that WebSphere Application Server always uses the short names when facing the operating system. If a server short name is changed in the administrative console, the corresponding symbolic link will be recreated using the specified short name.

7.2.5 Logging and tracing

In order to enable a WebSphere trace or to change the trace level for a specific server, there are basically two options. As in WebSphere distributed environments, this can be done in the following path of the administrative console: **Logging and Tracing → server_name → Change Log Detail Levels**. Click the **Configuration** tab and add the WebSphere trace specification.

Another option is to use MVS modify commands to dynamically turn on WebSphere traces. Usually MVS modify commands are executed from the SYSLOG. For system programmers, this option is an efficient way to enable traces.

The WebSphere trace can be turned on dynamically from the MVS console using the following modify command:

```
F CR_short_name,TRACEJAVA='com.ibm.ws.security.*=all'
```

The following modify command resets the trace configuration to the original settings from the last startup:

```
F CR_short_name,TRACEINIT
```

After installation, the default log level for an application server is set to *=info, which means that all Java components are logged with trace level INFO. This can produce a massive trace overhead especially in a production environment. We recommend that you adjust the trace level of the application server in a production environment to SEVERE and in a development and test environment to WARNING for all components, in order to reduce the trace overhead. Basically it is not possible to distinguish between CR and SR for defining the log levels. These settings are always valid for the complete application server. The trace output will be written to the started tasks job outputs. These can be viewed for instance in SDSF. For application related problems the servant region should always be the starting point for investigation.

Customers sometimes require the job output to be directed to a file in the HFS file system. This can be done by modifying the corresponding start procedures as described in White paper "*Directing SYSPRINT Output to an HFS File in WebSphere for z/OS*". It is available on the following website:

<http://www-03.ibm.com/support/techdocs/atsmstr.nsf/WebIndex/TD101087>

Because of the additional overhead of writing the log into an HFS file, we recommend that you do *not* redirect the job output.

7.3 Maintenance for the HFS

One of the basic concepts of rolling maintenance through an environment is that maintenance can only be applied to the configuration file system on a node-by-node basis. In fact, maintenance can be applied on each node of the cell independently. If the complete cell should be updated to a new service level, the process of maintenance is repeated for each node.

Moreover, it is possible to have each node of a cell on a different maintenance level as long as the deployment manager has the highest maintenance level. For instance, you can have a NodeA with service level 7.0.0.1 and a NodeB with service level 7.0.0.3 as long as the deployment manager itself has the service level 7.0.0.3. Consequently, *a new service level should always be applied to the deployment manager first*.

7.3.1 The process of applying maintenance

Maintenance is applied first to the product datasets including the HFS dataset, using SMP/E. Each configuration HFS is linked to a dedicated product HFS.

During the startup of a deployment manager, node agent, or application server, the applyPTF.sh shell script of the control region will notice a difference in the service level of the configuration HFS compared to the product HFS. (This script runs automatically as part of the server startup and should not be executed manually). If new maintenance has been introduced in the product HFS, applyPTF.sh updates the configuration HFS for the node. This shell script then recreates certain symbolic links within the configuration HFS. As a result, the configuration HFS and the product HFS have the same service level applied.

In WebSphere Application Server for z/OS V7, the load module libraries are located in the HFS instead of MVS datasets, which simplifies the process of maintenance. In previous versions, the load module datasets always had to be in sync with the HFS file system.

7.3.2 The concept of intermediate symbolic links

A lot of customers have isolated their test, acceptance test, development and production environment by using separate cells for each environment. Each cell usually has its own set of product datasets, which need to be maintained by SMP/E. In most cases, customers have fewer maintenance levels in use than different environments. With the use of intermediate symbolic links, you can reduce the number of product datasets by providing product datasets for each service level instead of each environment. These product datasets can be shared by nodes from different environments.

Because the product HFS is only mounted in READ mode, there is no opportunity for other environments to corrupt the product HFS. Moreover, intermediate symbolic links allow you to change the product HFS for each node independently. For instance, NodeA in the test environment and NodeD in the development can share the same product HFS, which has applied a certain maintenance level.

Figure 7-9 illustrates the use of one product HFS without intermediate symbolic links. In this case all symbolic links within a configuration HFS point directly to the product HFS. If a new maintenance level is applied to the product HFS, all configuration HFSs will be updated automatically. There is no possibility to distinguish different service levels between those nodes.

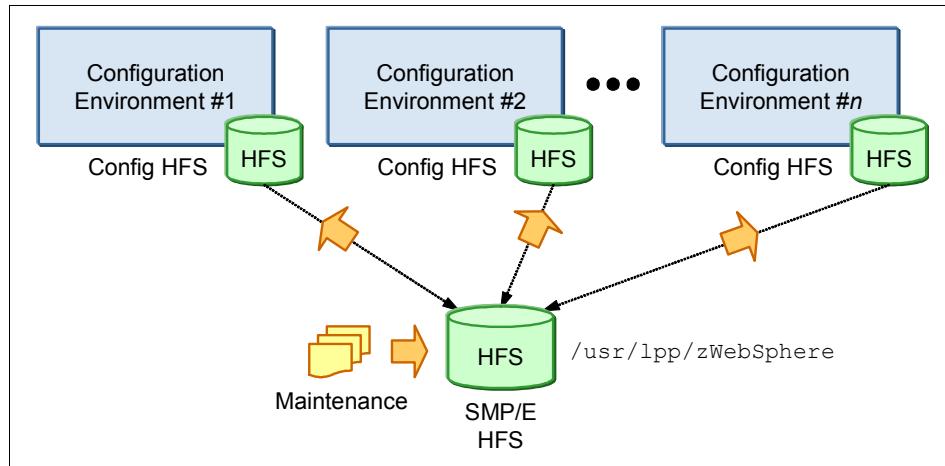


Figure 7-9 Applying maintenance without intermediate symbolic links

In contrast, intermediate symbolic links offer more flexibility for changing the maintenance level on node-basis. Instead of using a product HFS for each environment such as test, development or production, the concept of intermediate symbolic links accommodates a product HFS for each maintenance level, for instance one product HFS for version 7.0.0.1 and 7.0.0.3. An additional intermediate symbolic link is created, usually located in the mount directory of the node. All symbolic links within the configuration HFS point to the intermediate symbolic link, which refers to the product HFS.

As shown in Figure 7-10, the new maintenance level 700003 has been applied to a second product HFS. The intermediate symbolic link of node two is switched from the product HFS with version 700001 to the second product HFS with version 700003. During the next startup of a control region in node two, the configuration HFS will be updated to the new maintenance level 700003. Independently from node two, node one still points to version 700001.

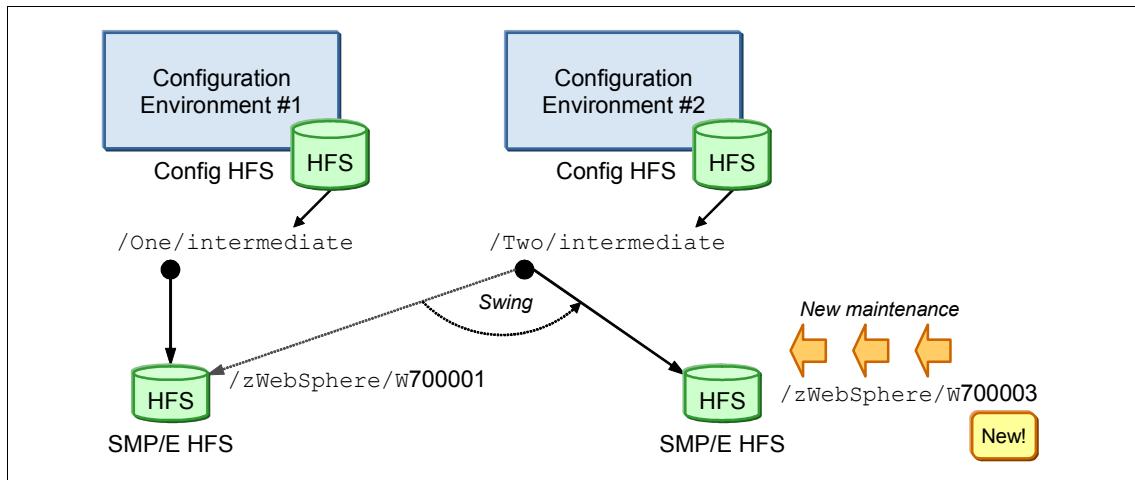


Figure 7-10 Applying maintenance with intermediate symbolic links

Although you might have only a few environments, we recommend that you use intermediate symbolic links to provide flexibility in changing maintenance levels on a node-by-node basis. During the customization of each WebSphere component in the PMT, you can specify an intermediate symbolic link as shown in Figure 7-11 on page 405. The check box, **Create intermediate symbolic link**, must be checked. The location of the intermediate symbolic link is usually in the mount point of the node as shown in this example.

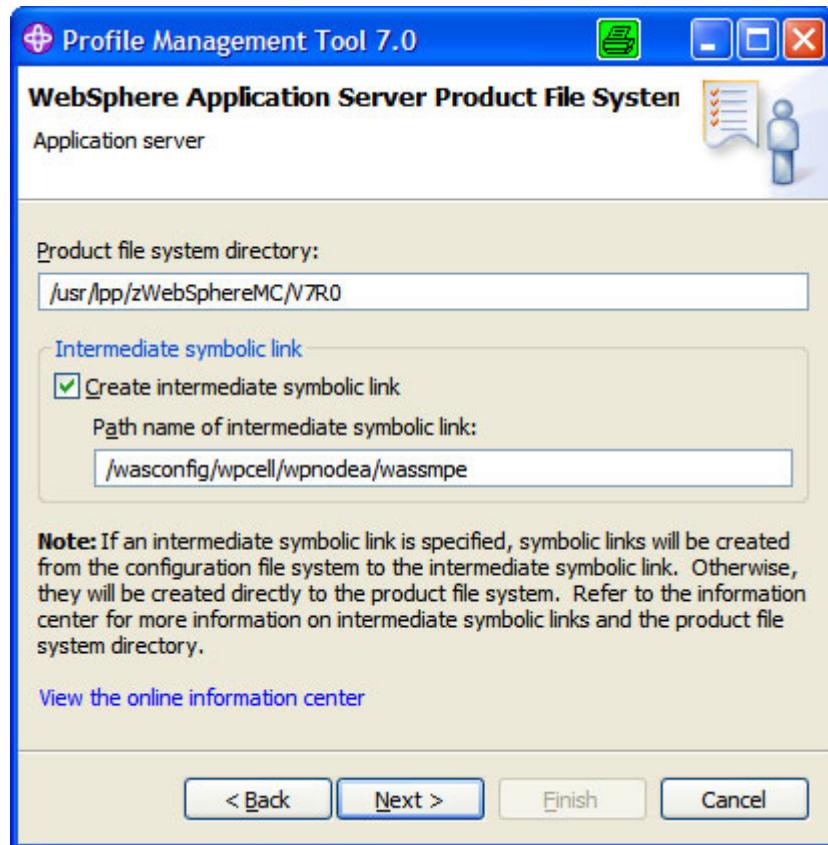


Figure 7-11 Create intermediate symbolic link during customization

More information about maintenance of WebSphere Application Server for z/OS is provided in the white paper, *WebSphere Application Server for z/OS - Planning for Test, Production and Maintenance*, which is available at the following website:

<http://www-03.ibm.com/support/techdocs/atstrmstr.nsf/WebIndex/WP100396>

7.4 Workload management

This section focuses on how WebSphere Application Server for z/OS exploits the WLM subsystem of z/OS.

7.4.1 Workload management overview

WebSphere Application Server for z/OS V7 can exploit the Workload Manager (WLM) subsystem of z/OS in the following ways:

- ▶ Workload classification:
 - Coarse grain workload management on a server basis.
- ▶ Transaction classification:
 - Fine grained workload management on transaction level.
- ▶ Servant activation:
 - Starts additional servant regions for application processing.

Before we go into more detail on the enhancements that the WLM offers, let us briefly explain the concepts of service classes, reporting classes, and enclaves.

Service class

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 0.2 seconds 90% 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.

In summary, a service class is the z/OS implementation of a service level agreement.

Enclaves

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 its 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.

In summary, an enclave is used to assign the user application a service class during runtime.

7.4.2 Workload classification

WebSphere Application Server for z/OS V7 and its prior versions are capable of classifying incoming work on a server basis. To do this, the control region of an application server checks to see which application server the request belongs to. It will then assign the request accordingly to the WLM queue. Each servant will process work for one service class at any point in time.

As seen in Figure 7-12 on page 407 incoming work is assigned a service class, based on information of the user- work request. The granularity is on application server level.

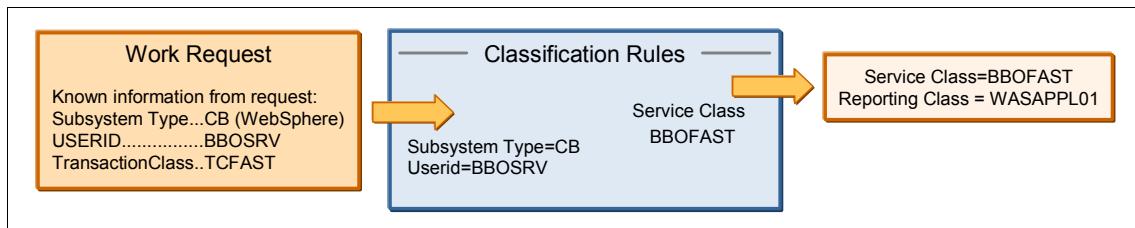


Figure 7-12 Workload classification for WebSphere Application Server for z/OS

7.4.3 Transaction classification

Transaction classification can be used to classify the transactions handled by your application. This technique can be used to prioritize special requests. A good example is a Web store that classifies its customers in gold and platinum terms, giving the platinum customers a better response time then the gold customers (see Figure 7-13).

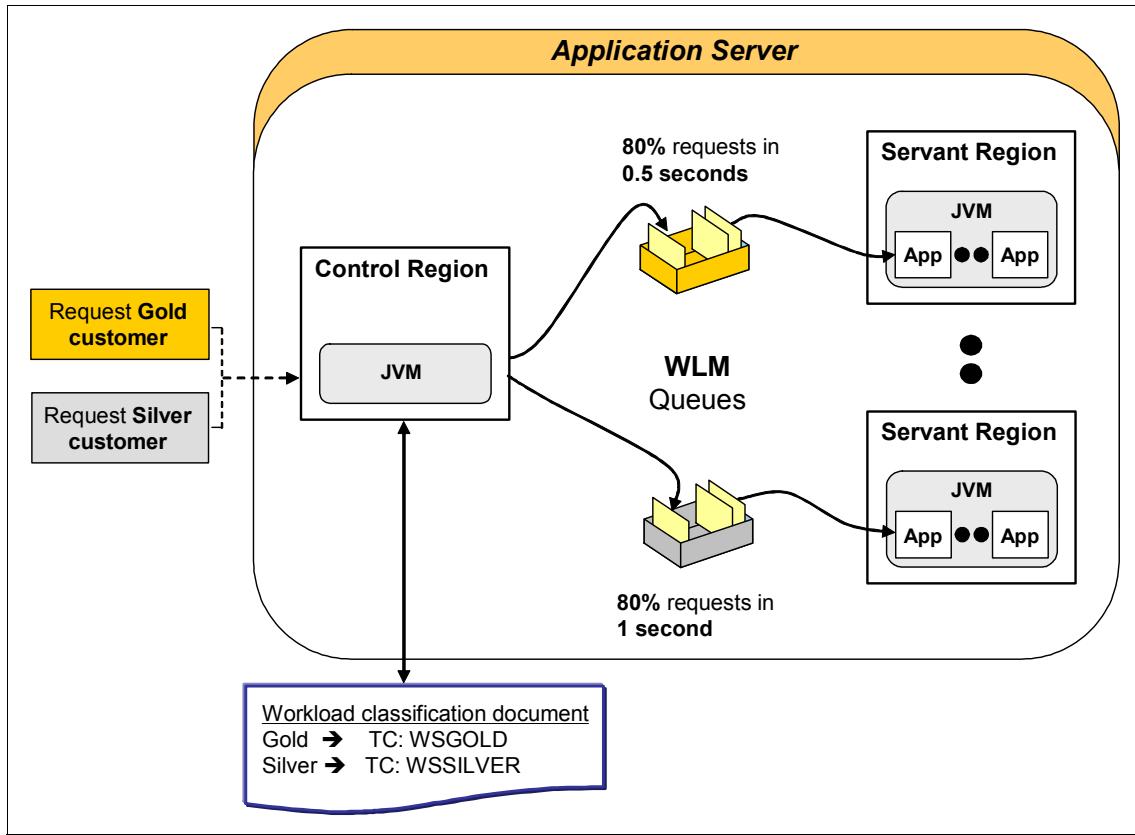


Figure 7-13 Transactional assignment of performance goals

A request that enters the system is assigned a *transaction class*, using the request details like utilized 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 file is an XML file that has to be populated with mapping rules.

Transaction classification can be used with the following protocols:

- ▶ Internal classification
- ▶ IIOP classification
- ▶ HTTP classification
- ▶ MDB classification
- ▶ SIP classification

In order to use transaction classification, you need to perform the following steps.

1. Talk with the application developers and the business functions to define what transactions need performance goals (Service Level Agreements).
2. Create a *workload classification document*.

In the following example, we have created a workload classification document for HTTP requests. The document includes transaction classes for multiple applications deployed to different application servers within the same cell.

The main advantage of the workload classification document is that the filter can be nested. For instance, the host name wtsc04.itso.ibm.com and port 12067 apply to the applications MyIVT, cachemonitor, and PlantsByWebSphere. In this case, we have not distinguished between different JSPs. By using the wildcard /PlantsByWebSphere/* all requests containing this context root are assigned to transaction class PLANTSWS.

Example 7-3 Example of a workload classification document for HTTP requests

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Classification SYSTEM "Classification.dtd" >
<Classification schema_version="1.0">
<InboundClassification type="http"
    schema_version="1.0"
    default_transaction_class="WPOTHER">
<http_classification_info transaction_class="WPOTHER"
    host="wtsc04.itso.ibm.com"
    port="12067"
    description="WPS01A HTTP">
<http_classification_info transaction_class="FRCAAPP"
    uri="/MyIVT/*"
    description = "FRCA Demo Application" />
<http_classification_info transaction_class="CACHEMON"
    uri="/cachemonitor/*"
    description="Cache Monitor" />
<http_classification_info transaction_class="PLANTSWS"
    uri="/PlantsByWebSphere/*"
    description="PlantsByWebSphere" />
<http_classification_info transaction_class="DAYTRADE"
    host="wtsc04.itso.ibm.com"
    port="12087"
    uri="/trade/*"
    description = "Trade Application" />
<http_classification_info transaction_class="DAYTRADE"
    host="wtsc04.itso.ibm.com"
    port="12088"
    uri="/daytrader/*"
    description = "Trade Application" />
```

```
<http_classification_info transaction_class="ADMINC"
                           host="wtsc04.itso.ibm.com"
                           port="12006"
                           uri="/ibm/console/*"
                           description = "AdminConsole HTTPS" />
</InboundClassification>
</Classification>
```

This final workload classification document needs to be copied to a path where the application server CR can access it. It is also necessary to create a DTD file in the same directory, which is referenced in the header of the workload classification document. Here is an example of that reference:

```
<!DOCTYPE Classification SYSTEM "Classification.dtd" >
```

You can use the DTD from the Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rrun_wlm_tclass_sample.html

3. Configure the server to use the classification document.
 - a. In order to activate the workload classification document, a new WebSphere variable needs to be defined with the path to the document. Select **Environment → WebSphere variables** in the administrative console.
 - b. Select the scope for the workload classification document. If the WebSphere variable is defined at the cell scope, the classification document is valid for the complete cell.
 - c. Click **New**.
 - d. Create a new variable named **wlm_classification_file**. The corresponding value should point to the complete USS path of the workload classification document.
 - e. Save and synchronize the changes with nodes.
 - f. In order for the classification file to take effect in a particular application server, it has to be restarted.
4. Modify the WLM settings to use transaction classes.

The same transaction classes specified in the workload classification document need to be defined in the Classification Rules of Subsystem CB in WLM. A classification rule maps a transaction class to a defined service class and reporting class. If you want to distinguish between multiple service classes within one application server, one servant region is required for each service class. Consequently the minimum and maximum amount of servant regions must be adjusted in order to handle the incoming workload.

According to the workload classification document in Example 7-3 on page 409, we have defined the transaction classes with the corresponding service classes and reporting classes in the following example. In this case the transaction classes should only apply to all started task names with the qualifier WP*. Therefore we defined a main rule (1) with the type CN and name WP*.

All transaction classes have been defined as sub rules (2) of the WP* filter. A transaction class is indicated by the classification type TC. The name of the transaction class must be equal to the transaction_class value in the workload classification document. In this example, a reporting class has been defined for each application, and all applications deployed on the same application server share the same service class.

For each report class, a separate workload activity report can be generated with the details about the CPU consumption. In this example, we have chosen service class WASHI, which has a response time goal of 80% requests in 0.5 seconds.

Example 7-4 Defining classification rules in WLM

```
Modify Rules for the Subsystem Type      Row 34 to 40 of 41
Command ==> _____ SCROLL ==>
CSR

Subsystem Type . : CB          Fold qualifier names? N (Y or N)
Description . . . _____

Action codes:   A=After     C=Copy      M=Move      I=Insert rule
                B=Before    D=Delete row R=Repeat    IS=Insert Sub-rule
                                         More ==>

-----Qualifier-----           -----Class-----
Action  Type    Name      Start      Service   Report
                               DEFAULTS: WASHI
                               _____
____ 1  CN     WP*      _____      WASHI    _____
____ 2  TC     PLANTSWS _____      WASHI    WPPLANTS
____ 2  TC     DAYTRADE _____      WASHI    WPDAYTRD
____ 2  TC     FRCAAPP _____      WASHI    WPFRCAAP
____ 2  TC     FRCAENC _____      WASHI    WPFRCA
____ 2  TC     CACHEMON _____      WASHI    WPCACHEM
____ 2  TC     ADMINC   _____      WASDM    WPADMINC
____ 2  TC     WPOTHER _____      WASDM    WPOTHER
```

The changes of the WLM configuration must be activated to take effect.

For detailed information about transaction classifications, 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:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0//index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rweb_classervers.html

5. To verify the defined workload classification, we have invoked several applications at the same time using a load generator and have monitored the classified enclaves in SDSF. All the active enclaves with the assigned service class and report class can be viewed in the SDSF.ENC panel.

Example 7-5 SDSF Display of Enclaves

SDSF ENCLAVE DISPLAY		SC04	ALL					
COMMAND INPUT ==>								
NP	NAME	SSType	Status	SrvClass	Per	PGN	RptClass	
	5400022F1B	CB	ACTIVE	WASDM	1		WPADMINC	
	5400022F45	CB	ACTIVE	WASHI	1		WPCACHEM	
	2C00022F38	CB	ACTIVE	WASHI	1		WPFRCAAP	
	3800022838	CB	ACTIVE	WASHI	1		WPFRCA	
	3000022F65	CB	ACTIVE	WASHI	1		WPPLANTS	
	2000000001	STC	INACTIVE	SYSSTC	1			
	2400000002	TCP	INACTIVE	SYSOTHER	1			

7.4.4 Servant activation

An application server can have multiple servant regions that process the user application. If the response time goals defined for the applications cannot be kept, WLM can start additional servant regions to process incoming, or queued up requests faster.

The minimum and maximum number of the servant regions can be defined as described in “Servant region” on page 386.

7.4.5 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.

The following recommendations apply:

- ▶ Control regions should be assigned a service class with high priority in the system, whether it is the SYSSTC service class or a high importance or velocity goal. This is because controllers perform 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 recommend that you use a high velocity goal.

In the following example, the WLM classification rules for all started tasks (STCs) that belong to a distributed environment have been defined, including a deployment manager, daemon, node agents and application servers. In order to use the STC name for classification, the type TN must be selected. The qualifier name specifies the STC name. In this example, wildcards have been used to define one classification rule that is valid for all application server CRs. We have assigned service classes with higher velocity goals to the application server CR and daemon. We have assigned service classes with lower velocity goals to the deployment manager, node agent, and application server SR.

Example 7-6 WLM classification of the CR and SR started tasks

```
Modify Rules for the Subsystem Type Row 16 to 22 of 106
Command ==> _____ SCROLL ==>
CSR

Subsystem Type . : STC      Fold qualifier names? N (Y or N)
Description . . . _____

Action codes: A=After      C=Copy      M=Move      I=Insert rule
              B=Before     D=Delete row  R=Repeat    IS=Insert Sub-rule
                           More ==>

-----Qualifier-----          -----Class-----
Action   Type     Name      Start      Service      Report
              DEFAULTS: STC_HIGH
-----          -----
____ 1 TN     WPDMGR   ____      STC        WPDMGR
____ 1 TN     WPDMGRS  ____      STC        WPDMGRS
____ 1 TN     WPDEMN   ____      STC_HIGH    WPDEMN
____ 1 TN     WPAGNT%  ____      STC        WPAGNT
____ 1 TN     WPS%%    ____      STC_HIGH    WPACR
____ 1 TN     WPS%%S   ____      STC        WPASR
```

Refer to the *Controller and Servant WLM classifications* article in the Information Center found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rweb_classservers.htm

- ▶ 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 need 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.

Attention: 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 (see Example 7-7).

Example 7-7 Service class definition for OMVS components

Subsystem Type . . :	OMVS	Fold qualifier names?	Y (Y or N)
Description . . .	OMVS subsystem rules mod for WAS		
<hr/>			
Action codes:	A=After	C=Copy	M=Move
	B=Before	D=Delete row	I=Insert rule
		R=Repeat	IS=Insert Sub-rule
			More ==>
-----Qualifier-----		-----Class-----	
Action	Type	Name	Start
			Service
			Report
		DEFAULTS:	OMVS
____	1	TN	FTPSEERVE
____	1	UI	OMVSKERN
____	1	TN	WSSRV*
			EBIZ_HI
			SYSSTC
			EBIZ_HI
			FTPSERV
			WAS70

7.5 What is new in V7 for z/OS

WebSphere Application Server V7 in general offers some new concepts, functions, and features. The following are new additions are specific to WebSphere Application Server V7.0 for z/OS:

- ▶ z/OS Fast Response Cache Accelerator (FRCA)
- ▶ Thread hang recovery

- ▶ SMF 120 subtype 9

In this section, we discuss each of these new features in more detail.

7.5.1 z/OS Fast Response Cache Accelerator (FRCA)

WebSphere Application Server for z/OS V7 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.

The high speed-cache can now 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 7-14 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.

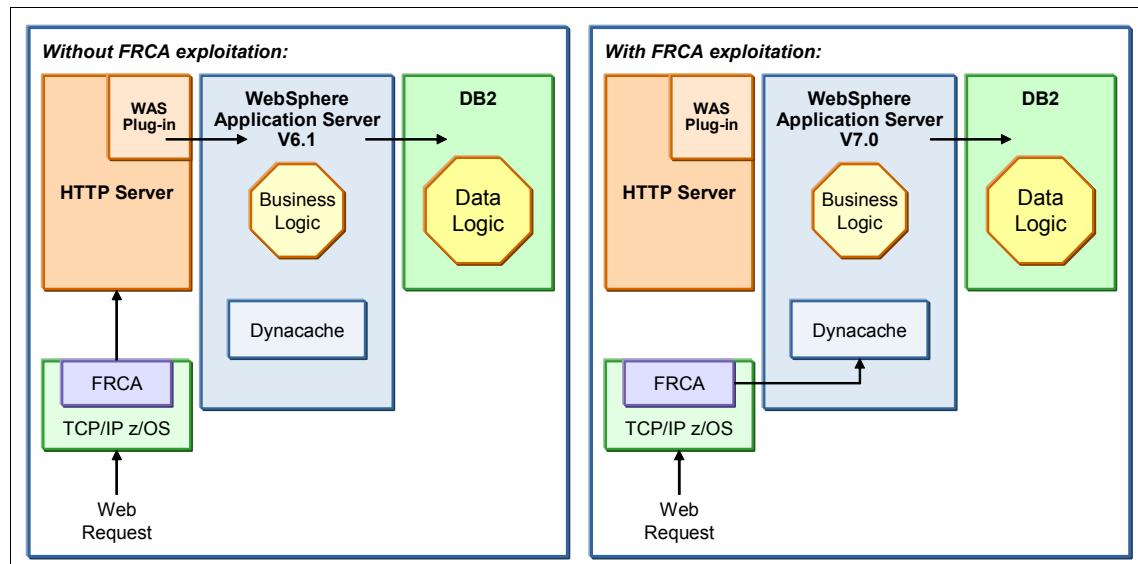


Figure 7-14 Overview of z/OS FRCA and WebSphere Application Server for z/OS

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 uses approximately 8% of the processor time that the same request consumed in a dynamic cache environment uses. These advantages come from its structure, because the FRCA cache can directly serve incoming TCP/IP requests.

Attention: This functionality needs z/OS 1.9 or higher to be used.

In order for FRCA to work properly, the fix for APAR PK72551 (UK42691) has to be applied to the Communications Server TCP/IP on z/OS Version 1.9. If this fix is not applied, the server will issue error message BBOO0347E or BBOO0348E. TCP/IP utilizes CSM storage to maintain the cache.

To use FRCA in an application server with 31-bit mode, the fix for APAR PK80838 must be applied. This fix is planned for inclusion in service level 7.0.0.3.

Restriction: Currently FRCA cache is only supported for non-SSL connections.

Configuring FRCA

FRCA support needs to be configured in the administrative 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 basis. To enable FRCA for a complete cell, it can be enabled on a server-by-server basis using wsadmin scripts:

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 10MB are used, modify user variable `protocol_http_large_data_response_buffer` to a size larger than the largest object to be cached.

Best-practice: We recommend that you configure the dynamic cache disk offload. This will prevent objects from being removed from the dynamic cache and hence being removed from the FRCA cache. Refer to the *Configuring dynamic cache disk offload* article in the information center for further information, found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/tdyn_diskoffload.html

Create external cache group

Follow these steps to create the cache group:

1. Click **Servers** → **Server Types** → **WebSphere application servers**.
2. Click on the server name that should benefit from the FRCA to open the configuration page.
3. Select **Container Services** → **Dynamic cache service**.
4. Select **External cache groups** under the Additional Properties section.
5. Click **New**.
6. Enter a name for the external cache group (frca in Figure 7-15), select **OK**, and save the changes to the master configuration. This name must be defined in the corresponding `cachespec.xml` file.

Application servers > WPS01A > Dynamic cache service > External cache group

Use this page to define sets of external caches that are controlled by WebSphere Application Server on Web servers, such as IBM Edge Server and IBM HTTP Server.

Preferences

	New	Delete	
Select	Name	Type	
You can administer the following resources:			
<input type="checkbox"/>	EsiInvalidator	Not shared	
<input type="checkbox"/>	frca	Not shared	

Figure 7-15 New external cache group

Add a member to the cache group

In this step, a new member has to be defined to the external cache group:

1. Open the new cache group by clicking on the name in the list of external cache groups (Figure 7-15).
2. Click on **External cache group members** in the Additional Properties section.
3. Select **New** to open the configuration page (Figure 7-16).

Application servers

[Application servers](#) > [WPS01A](#) > [Dynamic cache service](#) > [External cache group](#) > [frca](#) > [External cache group members](#) > 0

A single external cache that WebSphere Application Server controls.

Configuration

General Properties

Advanced Fast Path Architecture (AFPA)

Adapter bean name
com.ibm.ws.cache.servlet.Afpa

* Port
0

Fast Response Cache Accelerator

Enable fast response cache accelerator

Cache size
102400000 bytes

Max entry size
1000000 bytes

Stack name
TCPIP

Transaction class
FRCAENC

⚠ Disk offload caching not enabled. When fast response cache accelerator is enabled, it is highly recommended you also enable disk caching offload. You can enable it by selecting the 'Enable disk offload' option on the [Dynamic cache service](#) panel.

Figure 7-16 Enable FRCA cache

4. In the configuration tab, enter:
 - Select **Advanced Fast Path Architecture (AFPA)**.
 - Check the **Enable fast response cache accelerator** option.

- In the Port field, select 0 as the port number.

For FRCA configuration, the following fields will need to be set as well:

- **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 4K(4096) interval. The default is 102400000.
- **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 TCPIP 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 TCPIP 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.

Usually the TCPIP address space runs under a high WLM priority such as SYSSTC within z/OS. Because the FRCA cache is physically located in TCPIP address space, the FRCA processing runs under the same priority as the TCPIP address space, if no special CB transaction class for FRCA is defined with a lower service goal.

If no FRCA specific transaction class with a lower service goal is defined, the FRCA processing can theoretically cause the TCPIP address space to reject any additional connections. This can happen if especially large FRCA cached objects are invoked in a high frequency so that it consumes all available CPU resources. That is the reason why we recommend to define this additional transaction class.

Furthermore, a FRCA reporting class can be used to estimate the savings in CPU consumption compared to classical WebSphere workload without FRCA caching.

Currently it is not possible to distinguish between different transaction classes within the FRCA workload as opposed to the classical WebSphere workload, where a workload classification can be utilized.

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. Select **Servers** → **Server Types** → **WebSphere application servers** → *server_name* → **Web Container Settings** → **Web container transport chains** → *transport_chain*, and select the **Disable FRCA caching** option.

Update cachespec.xml

When updating the XML file, remember that the names used for the external cache in the XML file and the administrative console must match. Otherwise the cache cannot be used. A sample cachespec.xml is shown in Figure 7-17:

```
<property name="ExternalCache">frca</property>
```

An assembly tool is used to place the cachespec.xml file in the deployment module and to define the cacheable objects. You can also place a global cachespec.xml file in the application server properties directory.

For more information about the usage of a cachespec.xml refer to the WebSphere Application Server V7 Information Center, available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rdyn_cachespec.html

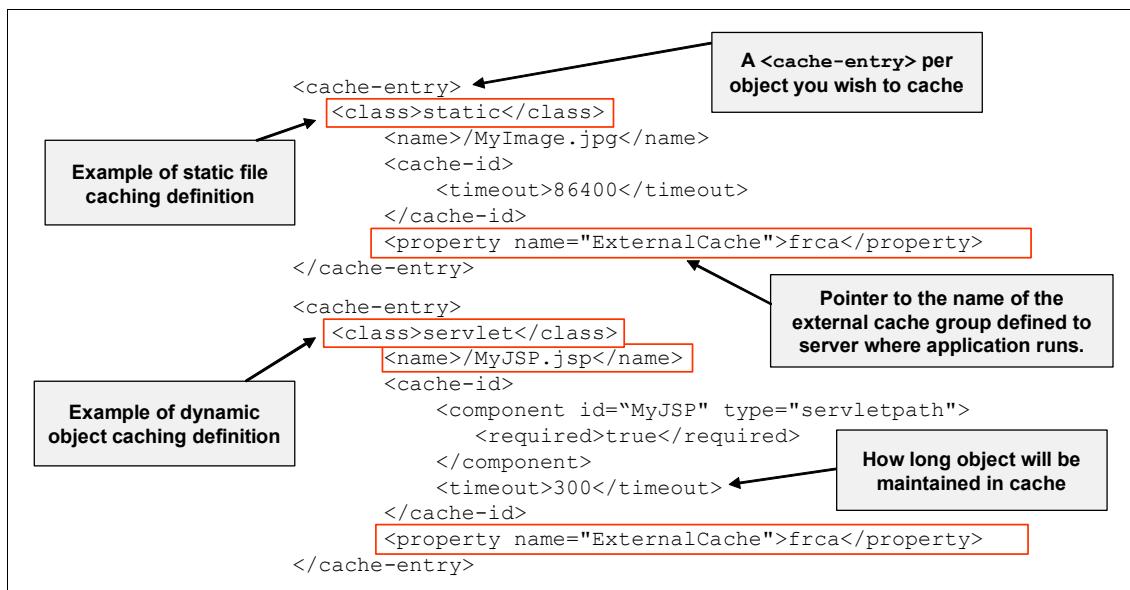


Figure 7-17 Sample `cachespec.xml` for usage with FRCA

[Optional] Enable FRCA logging

If you want to enable logging for FRCA, use the administrative console and execute the following steps:

1. Select **Servers** → **Server Types** → **WebSphere application servers** → **server_name**.
2. Under Troubleshooting, select **NCSA access and HTTP error logging**.
3. Select the **Enable logging service at server start-up** option.

You can modify the other settings or keep the defaults.

Large object caching

If objects larger than 10MB should be cached, then you need to set the custom property `protocol_http_large_data_response_buffer`. 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:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp>

Monitoring FRCA

You can use the WebSphere Application Server `modify display` command to display statistics and to list the contents of the FRCA cache. Refer to the *Configuring high-speed caching using FRCA with the WebSphere Application Server on z/OS* article in the Information Center for more detail:

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`
- ▶ `f <serverName>,display,frca,content`
- ▶ From z/OS console: `display tcpip,,netstat,cach`
- ▶ From TSO: `netstat cach`

To monitor the activity of object caching 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

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, then the message seen in Example 7-8 will be issued:

Example 7-8 FRCA access denied message

```
BB00nnnnE FRCA INITIALIZATION FAILED. SERVER NOT AUTHORIZED TO  
USE FRCA SERVICES. IOCTL RV=%d, RC=%d, RSN=%08X
```

In order to use FRCA, the following RACF definitions are required:

- ▶ First, a new SERVAUTH profile needs to be defined with the corresponding system name and TCP/IP procedure name:

```
RDEFINE SERVAUTH EZB.FRCAACCESS.<system_name>.<TCPIP_procname>  
UACC(NONE)
```

- ▶ Then, the application server control region must be given READ access to this SERVAUTH profile:

```
PERMIT EZB.FRCAACCESS.<system_name>.<TCPIP_procname> CLASS  
(SERVAUTH) ID (UID_CR) ACCESS (READ)
```

- ▶ Finally, the SERVAUTH class has to be refreshed, to activate the changes:

```
SETRPOTS RACLIST (SERVAUTH) REFRESH
```

7.5.2 Thread hang recovery

This section describes the new thread hang recovery option available on z/OS.

Overview

WebSphere Application Server for z/OS V7 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 IBM System z® Application Assist Processors (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. First, 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 with few processor resources per request. If such a transaction would suddenly consume a high amount of CPU due to an error, this situation would not be detected by prior versions unless the normal timeout occurs. Until the timeout occurs, there is a performance impact to the whole environment.

Pre-V7 technique

In releases prior to V7, when a request times out, the server assumes that the request is hung and will start to resolve 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. All active threads in that particular servant are lost. A multi-servant architecture in this case can prevent total outages, because new work requests can be dispatched to another servant. A dump of the servant can 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. Figure 7-18 illustrates a hanging thread in the servant region.

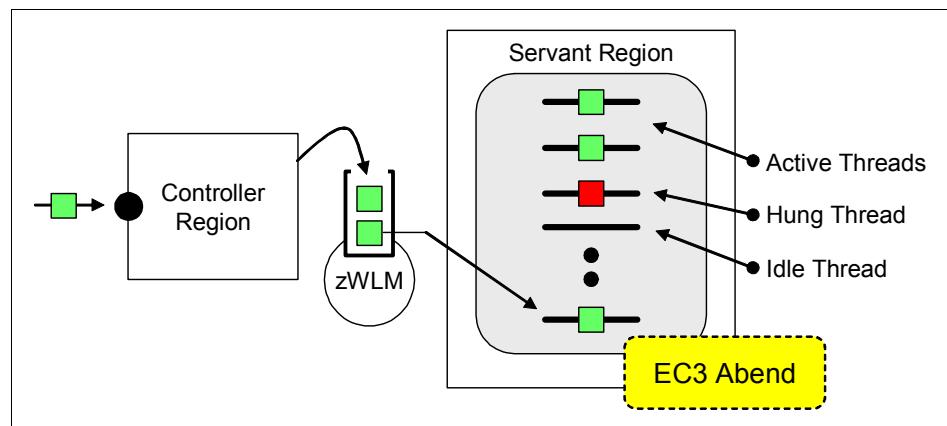


Figure 7-18 EC3 Abend caused by a hung thread prior to V7

- ▶ 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 that 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 if 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.

V7 technique

In V7, if a hung thread is detected, then the servant can try to interrupt the thread (shake it loose). 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 WebSphere Application Server V7. You do not have to implement code for the Interruptable Objects registry.

The results of this situation can be:

- ▶ The thread can be freed:

The user whose request is hung receives an exception. The administrator can define what dump action should be taken (none, svcdump, javacore or traceback).

- ▶ The thread cannot be freed:

If the thread cannot be freed, the system action depends on the administrator settings. Basically the options are:

- Abend the servant.
- Keep the servant up and running.
- Take a dump (defined by new variables, see Table 7-1, “Variables for hung thread related actions” on page 426)

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 on whether a servant should be abended or kept alive now depends on:

- ▶ How much CPU time is consumed by the thread? (Looping or really just hanging?)
- ▶ How many threads are already in a hung state, within this servant?
- ▶ Is the servant the last servant?

The recovery process of a hanging thread is illustrated in Figure 7-19.

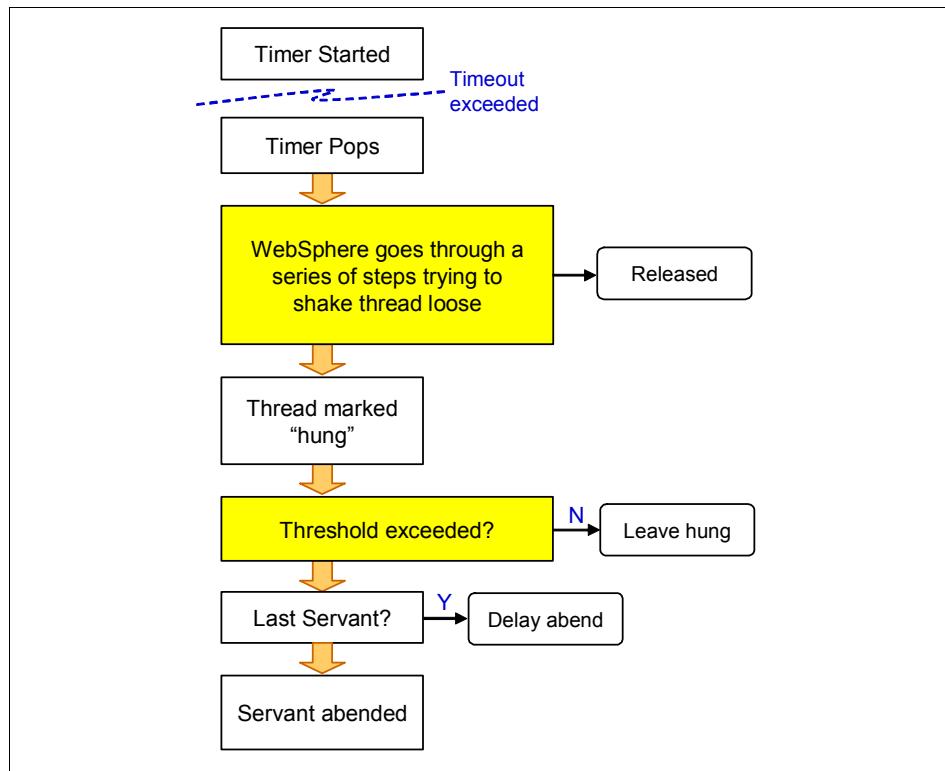


Figure 7-19 Hanging thread recovery process

With a new environment variable

`server_region_stalled_thread_threshold_percent`, a percentage of threads that can be marked as "hung" can be specified, before the control region terminates the servant.

New properties

V7 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 create the custom properties, log on to the administrative console and navigate to **Server → Server Types → WebSphere application servers → *server_name* → Server Infrastructure → Administration → Custom properties → New**. The variables are listed in Table 7-1.

Note: When modifying one of the new parameters, make sure to have these additional parameters configured (available in prior versions):

- ▶ control_region_timeout_delay
- ▶ control_region_timeout_save_last_servant

Table 7-1 Variables for hung thread related actions

Variable name	Values	Meaning
server_region_stalled_thread_threshold_percent	0 -100	This variable specifies the percentage of threads that can be marked as “hung” before the controller terminates the servant. Default value of 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 a stalled thread cannot be interrupted. <type> is the access method, which can be specified as http(s), iiop, mdb, or sip(s).
server_region_request_cputimeused_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 abandoned 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 might or might not get delivered to the rogue request.

The signal might 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 BB000327 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.

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>]
```

See Figure 7-20 for an illustration of this situation.

```
F <server>,DISPLAY,THREADS,ALL
One servant example:

BBOJ0111I: REQUEST ASID JW TO RE DISPATCH TIME
BBOJ0112I: fffffb35f 0176 N N N 2009/03/26 18:21:17.423648
BBOJ0112I: fffffb360 0176 Y N N 2009/03/26 18:21:15.569834
BBOJ0112I: fffffb361 0176 Y Y N 2009/03/26 18:21:12.790693
BBOJ0112I: fffffb362 0176 Y Y Y 2009/03/26 18:21:10.720461
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,THREADS,ALL

          *----- Healthy thread
          *----- In a Java Wait but not yet Timed Out
          *----- Java Wait and Timed Out, but ITI has
          *----- not yet marked it "hung"
          *----- JW, TO and retried -- Hung
```

In this example Threshold > 0 otherwise servant would be recycled because one thread is hung

Figure 7-20 Output of display thread modify command

The information is also available via a new `InterruptibleThreadInfrastructure` MBean.

More information about the thread hang recovery and dispatch timeouts is provided in the white paper, *WebSphere Application Server for z/OS V7 - Dispatch Timeout Improvements*, available at:

<http://www-03.ibm.com/support/techdocs/atstr.nsf/WebIndex/WP101374>

7.5.3 Systems Management Facility (SMF) Subtype 9

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

Since Version 4 of WebSphere Application Server, SMF record 120 has been used to log WebSphere usage 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, 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

In V7 a new subtype 9 record has been added. The name is *Request Activity Record* and 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. Additionally you can activate or deactivate this record dynamically.

Recommendation: Because the new SMF 120-9 subrecord consumes less processing time than the collection of multiple subrecords, we recommend the usage of the new subrecord. The old subrecords remain valid and can be used with WebSphere Application Server for z/OS V7.

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 7-2 shows the new variables, possible values and their meaning. A browser to display the contents of the new SMF records is provided.

Table 7-2 New SMF record 120-9 variables

Variable name	Values	Meaning
server_SMF_request_activity_enabled	0 1	Turn record off/ on
server_SMF_request_activity_CPU_detail	0 1	Processor usage details
server_SMF_request_activity_timestamps	0 1	Formatted timestamps
server_SMF_request_activity_security	0 1	Security information

Modify commands

The following modify commands can be used to activate/ 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 command:

F <server>,DISPLAY,SMF

For more information about the SMF enhancements refer to White paper *Understanding SMF Record Type 120, Subtype 9*, available at:

<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101342>

7.6 Thread management using the workload profile

The workload profile setting defines the number of available application threads for each servant. This setting applies only to worker threads which are managed by the application server.

To change the value via the administrative console, click **Servers** → **Server Types** → **WebSphere application servers** → **server_name** → **Container services** → **ORB service** → **z/OS additional settings**.

[Application servers](#) > [WPS02A](#) > [ORB service](#) > **z/OS additional settings**

Use this page to view and modify z/OS(R) additional settings for the Object Request Broker (ORB)

Configuration

General Properties

* ORB listener keep alive

seconds

* ORB SSL listener keep alive

seconds

* Workload manager timeout

seconds

Workload profile



Figure 7-21 Thread Management - Workload profile

The optimal number of parallel threads depends on the number of available CPU resources and on the workload characteristics of the application. We recommend that you use the Tivoli Performance Viewer in the administrative console to determine the number of parallel threads that are currently occupied by a representative production workload. In most customer cases, the IOBOUND profile is selected.

WebSphere Application Server for z/OS V7 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.

Table 7-3 explains all possible workload profile settings.

Table 7-3 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.
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	This option is new in V7! Specifies that the number of servant application threads is determined by the value that is specified for the servant_region_custom_thread_count server custom property. The minimum number of application threads that can be defined for this custom property is 1; the maximum number of application threads that can be specified is 100.

7.7 Local connectivity to DB2

If the DB2 resides on the same system as WebSphere Application Server, we recommend that you use the JDBC type 2 driver. This driver establishes a cross-memory connection to DB2, as opposed to a JDBC type 4 connection, which is always a TCP/IP connection with the corresponding overhead.

7.7.1 Prerequisites for implementing a JDBC type 2 driver

The following prerequisites must be fulfilled in order to get the JDBC type 2 driver to work:

1. Ensure that the following DB2 load libraries are in the STEPLIB or LNKLST concatenation of the application server control region and servant region:
 - a. SDSNEXIT: Depending on your individual DB2 setup, the SDSNEXIT module should be included.
 - b. SDSNLOAD
 - c. SDSNLOD2

See Example 7-2 on page 397.

2. When using the JDBC type 2 driver, WebSphere Application Server uses the DB2 subsystem identifier provided in the DSNHDECP load module. If another DB2 subsystem should be used, a db2.jcc.propertiesFile needs to be specified in the JVM custom properties, containing the subsystem's SSID. More information about this properties file, see:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/tdat_jdbcdb2cfg.html

3. The JDBC driver must first be bound with the DB2 packages that represent the SQL statements to be executed. The specific details of the bind utility and bind process are described by the README provided with the installed DB2 Universal JDBC Driver. Refer to this README for details on how to set up and perform the required binding. Example 7-9 shows the necessary OMVS commands for the binding process.

Example 7-9 JDBC Driver binding

```
export PATH=/usr/lpp/db2/d8xg/jcc3/bin:$PATH
export
CLASSPATH=/usr/lpp/db2/d8xg/jcc3/classes/db2jcc.jar:/usr/lpp/db2/d8xg/jcc3/classes/db2jcc_javax.jar:/usr/lpp/db2/d8xg/jcc3/classes/db2jcc_licence_cisuz.jar:/usr/lpp/db2/d8xg/jcc3/classes/sqlj.zip
```

```
java com.ibm.db2.jcc.DB2Binder -url  
jdbc:db2://<host_name>:<tcpport>/<location_name> -user <sysadm>  
-password xxxx
```

We recommend JDBC Driver V3.51 or higher, because the multi-row fetch feature has been introduced which can significantly improve the performance of a type 2 connection.

The JDBC Driver (JCC) V3.51+ driver is available to both DB2 for z/OS Version 8 and Version 9 customers. The initial deliveries of JCC 3.51 for DB2 z/OS were made under PK63584 for DB2 z/OS V8 and PK68428 for DB2 z/OS V9. These APARs are available for immediate customer install and include all of the following major JDBC type-2 for z/OS features/enhancements:

- ▶ Ability to concurrently connect to multiple, local DB2 subsystems
- ▶ Type 2 failover support using DB2 z/OS Group Attach Names
- ▶ Support for multiple row INSERT using when using JDBC batching
- ▶ Support for multiple row FETCH

The following command can be used to check the installed version of the JDBC Driver:

```
java -cp <path_to_jdbc_classes>/db2jcc.jar com.ibm.db2.jcc.DB2Jcc  
-configuration
```

WebSphere Application Server V7 is based on JDK 1.6, which includes the new JDBC 4.0 specification level. This specification is implemented in the JDBC Driver V4.3+ on z/OS. This version of the driver also supports the new multi-row fetch feature as with V3.51+ and is only delivered with DB2 for z/OS V9.

7.7.2 Creating a JDBC provider

To create a JDBC provider for type 2 connections:

1. From the WebSphere Application Server for z/OS administrative console, click **Resources** → **JDBC** → **JDBC Providers**.
2. Select the scope at which applications can use the JDBC provider. This scope becomes the scope of any data source that you associate with this provider. You can choose a cell, node, cluster, or server.
3. Click **New** to start the wizard for creating a new JDBC provider.
4. Specify the following parameters:
 - **Database type:** DB2
 - **Provider type:** DB2 Universal JDBC Provider
 - **Implementation type:** Connection pool data source

If you use the Connection pool data source implementation type with type 2 connectivity, WebSphere Application Server on z/OS uses Resource Recovery Services (RRS) to process both one-phase and two-phase transactions. In contrast, if the application does not require that connections support two-phase commit transactions, and you plan to use type 4 connectivity, choose Connection Pool Data Source.

Choose XA Data Source if you plan to use driver type 4, and your application requires connections that support two-phase commit transactions. Use only driver type 4 connectivity for the XA data source.

- Give a name to the new JDBC provider.

The screenshot shows the 'Create new JDBC provider' wizard. On the left, a sidebar lists steps: 'Step 1: Create new JDBC provider' (highlighted with a blue arrow), 'Step 2: Enter database class path information', and 'Step 3: Summary'. The main panel has a title 'Create new JDBC provider' and a descriptive text: 'Set the basic configuration values of a JDBC provider, which encapsulates the specific vendor JDBC driver implementation classes that are required to access the database. The wizard fills in the name and the description fields, but you can type different values.' It contains several input fields:

- Scope:** A dropdown menu showing 'cells:WPCell'.
- * Database type:** A dropdown menu showing 'DB2'.
- * Provider type:** A dropdown menu showing 'DB2 Universal JDBC Driver Provider'.
- * Implementation type:** A dropdown menu showing 'Connection pool data source'.
- * Name:** An input field containing 'DB2 Universal JDBC Driver Provider WPCell'.

Figure 7-22 Create a new JDBC provider: Step 1

Click **Next** to continue.

5. In the next panel the JDBC class paths need to be specified as shown in Figure 7-23 on page 435.

Here is an example of JDBC class paths:

Class path:

/usr/lpp/db2/d8xg/jcc3/classes

Native library path:

/usr/lpp/db2/d8xg/jcc3/lib

The native library path is the path, where the .so files are located.

Both paths DB2UNIVERSAL_JDBC_DRIVER_PATH and DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH will be saved as WebSphere environment variables with the same scope as the JDBC provider. The JDBC class paths can be adjusted afterwards by changing the value of these variables in **Environment** → **WebSphere variables** in the administrative console.

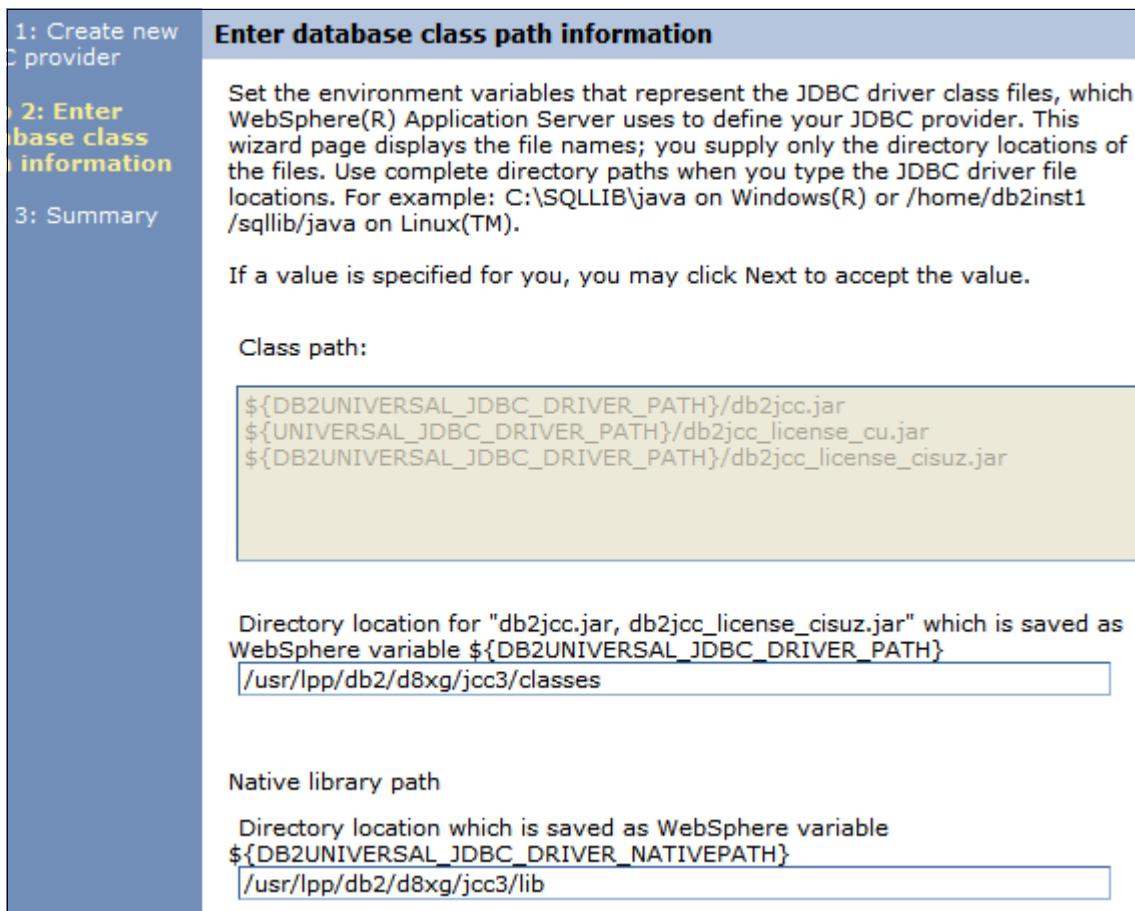


Figure 7-23 Database class path information

Click “**Next**” to continue.

6. Finally a summary of the JDBC provider configuration will be displayed. Click **Finish** to complete the wizard.

7.7.3 Defining a type 2 data source

The next step is to define the data source.

1. Click the JDBC provider, which has been just created.
2. In the right navigation bar of the JDBC provider configuration, click “**Data sources**” under Additional Properties.
3. Click the **New** button to create a new data source.

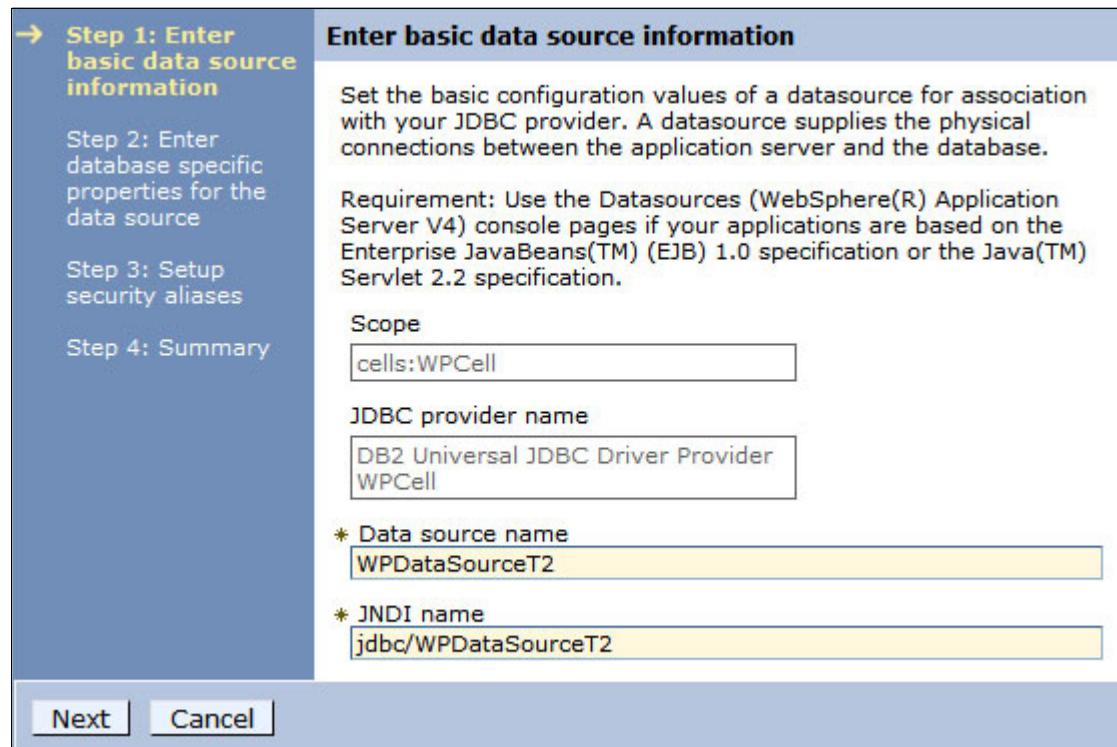


Figure 7-24 Create a new data source: Step 1

Specify the data source name and the corresponding JNDI name. Usually the prefix “**jdbc**” is used for data sources.

Click **Next** to continue.

4. In the next panel the database specific properties of the data source can be specified. Here is an example:
 - **Driver type:** 2
 - **Database name:** DB8X (database location name)
 - **Server name:** wtsc04.itso.ibm.com (host name)
 - **Port number:** 33760

A DISPLAY DDF command can be used to find the database location name, host name, and port number, as shown in Example 7-10.

Example 7-10 Display DDF command

```
-D8X1 DIS DDF
RESPONSE=SC04
DSNL080I -D8X1 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION          LUNAME        GENERICCLU
DSNL083I DB8X               DEIBMPIA.IPAADB21 -NONE
DSNL084I IPADDR             TCPPORT      RESPOR
DSNL085I 9.12.4.20          33760       33761
DSNL086I SQL    DOMAIN=wtsc04.itso.ibm.com
DSNL086I RESYNC DOMAIN=wtsc04.itso.ibm.com
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

For the JDBC type 2 driver, the server name and port number is not required, but it can be specified in order to easily switch between type 2 and typ4 driver.

Step 1: Enter basic data source information → Step 2: Enter database specific properties for the data source Step 3: Setup security aliases Step 4: Summary	<p>Enter database specific properties for the data source</p> <p>Set these database-specific properties, which are required by the database vendor JDBC driver to support the connections that are managed through the datasource.</p> <table border="1" style="width: 100%; border-collapse: collapse;"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>* Driver type</td><td><input type="button" value="2"/></td></tr><tr><td>* Database name</td><td>DB8X</td></tr><tr><td>Server name</td><td>wtsc04.itso.ibm.com</td></tr><tr><td>Port number</td><td>33760</td></tr></tbody></table> <p><input checked="" type="checkbox"/> Use this data source in container managed persistence (CMP)</p>	Name	Value	* Driver type	<input type="button" value="2"/>	* Database name	DB8X	Server name	wtsc04.itso.ibm.com	Port number	33760
Name	Value										
* Driver type	<input type="button" value="2"/>										
* Database name	DB8X										
Server name	wtsc04.itso.ibm.com										
Port number	33760										

Figure 7-25 Data source properties

Click “**Next**” to continue.

5. Finally a summary of the data source configuration will be displayed. Click “**Finish**” to complete the wizard.
6. Click **Save and synch changes with nodes**.

A restart of the corresponding application servers is necessary in order to use the JDBC type 2 driver.

7.8 Migrating to V7

If you plan to migrate from previous versions of WebSphere Application Server for z/OS to version 7.0, the white paper, *Migrating to WebSphere z/OS V7*, provides all necessary information to successfully migrate to V7. This document is available at:

<http://www-03.ibm.com/support/techdocs/atstrmstr.nsf/WebIndex/WP101329>

A lot of effort has been expended to improve the migration process over that found in previous versions.



Administration with scripting

The administrative console is sufficient for tasks that are non-repetitive, have a minimal number of administrative steps, and are relatively simple. For administration that requires many steps, which can be repetitive, and time consuming to configure, wsadmin combined with scripts is an ideal tool.

In this chapter we introduce the `wsadmin` scripting solution and describe how you can use it to perform basic tasks.

This chapter contains the following topics:

- ▶ “Overview of WebSphere scripting” on page 440
- ▶ “Launching wsadmin” on page 441
- ▶ “Command and script invocation” on page 445
- ▶ “wsadmin management objects” on page 447
- ▶ “Managing WebSphere using script libraries” on page 452
- ▶ “Assistance with scripting” on page 471
- ▶ “Example: Using scripts with the job manager” on page 476
- ▶ “Online resources” on page 485

8.1 Overview of WebSphere scripting

WebSphere Application Server provides a scripting interface based on the *Bean Scripting Framework (BSF)* called **wsadmin**. BSF is an open source project to implement an architecture for incorporating scripting into Java applications and applets. The BSF architecture works as an interface between Java applications and scripting languages. Using BSF allows scripting languages to do the following tasks:

- ▶ Look up a pre-registered bean and access a pre-declared bean
- ▶ Register a newly created bean
- ▶ Perform all bean operations
- ▶ Bind events to scripts in the scripting language

Because **wsadmin** uses BSF, it can make various Java objects available through language-specific interfaces to scripts. Figure 8-1 shows the major components involved in the **wsadmin** scripting solution.

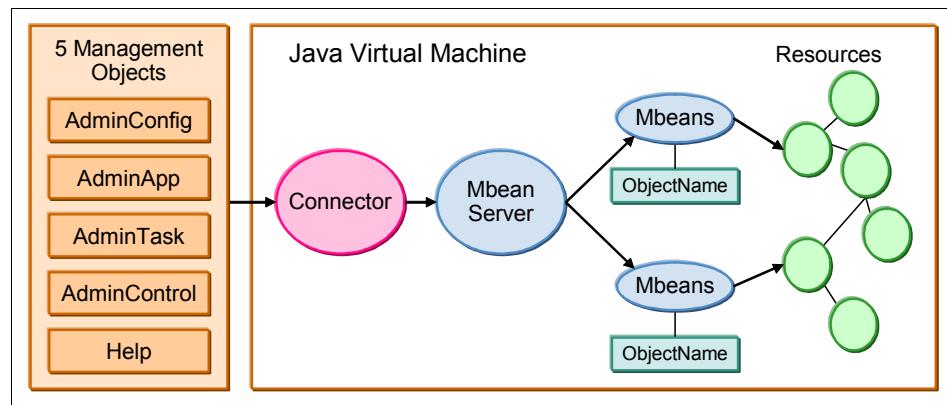


Figure 8-1 wsadmin scripting

8.1.1 Script programming languages

Two programming languages are used to write **wsadmin** scripts -- Jython and Jacl. WebSphere Application Server V7.0 represents the start of the deprecation process for the Jacl syntax. The script library and the command assistance on the administrative console only support Jython.

If you have existing Jacl scripts and want to start migrating to Jython, the Jacl-to-Jython conversion utility can be used to convert the scripts. This conversion assistant typically does 95-98% of a preliminary conversion. In most cases, the resulting conversion is syntactically and runtime equivalent.

However, we strongly recommend that you verify each line to ensure that the code functions as you originally intended. When Jacl and Jython language differences can result in lines of code that are difficult to convert automatically, the converted lines are flagged #?PROBLEM?. This provides assistance in finding areas that are most likely in need of manual conversion.

8.2 Launching wsadmin

The **wsadmin** command file resides in the bin directory of every profile. Start **wsadmin** from a command prompt with the command:

- ▶ (UNIX) *profile_root/bin/wsadmin.sh*
- ▶ (Windows) *profile_root\bin\wsadmin*

Note that the **wsadmin** command also exists in the bin directory of the *install_root* directory. If you start **wsadmin** from this location, you must be careful to specify the profile to work with in the command. If you do not specify the profile (or forget to specify it), the default profile will be chosen.

Example 8-1 illustrates how to start wsadmin. In this example, the wsadmin command is used to connect to the job manager. It is issued from the bin directory of the job manager profile, so the profile does not need to be specified. The -lang argument indicates Jython will be used (Jacl is the default).

Example 8-1 flexible management: wsadmin command-line

```
C:\WebSphereV7\AppServer\profiles\jmgr40\bin>wsadmin -lang jython
WASX7209I: Connected to process "jobmgr" on node jmgr40node using SOAP
connector
; The type of process is: JobManager
WASX7031I: For help, enter: "print Help.help()"
wsadmin>
```

To get syntax-related help, use **wsadmin -?** or **-help** (see Example 8-2).

Example 8-2 wsadmin syntax

```
wsadmin
  [ -h(elp) ]
  [ -? ]
  [ -c <command> ]
  [ -p <properties_file_name>]
  [ -profile <profile_script_name>]
  [ -f <script_file_name>]
  [ -javaoption java_option]
```

```

[ -lang language]
[ -wsadmin_classpath class path]
[ -profileName profile]
[ -conntype
    SOAP
        [-host host_name]
        [-port port_number]
        [-user userid]
        [-password password] |
    RMI
        [-host host_name]
        [-port port_number]
        [-user userid]
        [-password password] |
    JSR160RMI
        [-host host_name]
        [-port port_number]
        [-user userid]
        [-password password] |
    IPC
        [-ipchost host_name]
        [-port port_number]
        [-user userid]
        [-password password] |
    NONE
]
[ -jobid <jobid_string>]
[ -tracefile <trace_file>]
[ -appendtrace <true/false>]
[ script parameters ]

```

8.2.1 Scripting environment properties file

The properties that determine the scripting environment for `wsadmin` can be set using either the command line or a properties file. Modifying the properties file can be useful when you want to change a default setting, for example, changing the language from Jacl to Python.

Properties can be set in the following locations:

- ▶ The installation default properties file for the profile located in the following directory:
profile_root/properties/wsadmin.properties
- ▶ A user default properties file located in the Java user.home property.

- ▶ A customized properties file placed in the location pointed to by the WSADMIN_PROPERTIES environment variable.
- ▶ A customized properties file pointed to by using the **-p** argument to the **wsadmin** command.

When **wsadmin** is started, properties are loaded from these files in the order listed above. The properties file that is loaded last overrides the ones loaded earlier.

The properties are listed in Table 8-1.

Table 8-1 wsadmin properties

Property	Value
com.ibm.ws.scripting.connectionType	SOAP, RMI or NONE
com.ibm.scripting.port	TCP port of target system
com.ibm.scripting.host	Host name of target system
com.ibm.ws.scripting.defaultLang	Jython or Jacl
com.ibm.ws.scripting.echoparams	Determines whether parameters or arguments are output to STDOUT or to the wsadmin trace file
com.ibm.ws.scripting.traceFile	File for trace information
com.ibm.ws.scripting.validationOutput	Location of validation reports
com.ibm.ws.scripting.traceString	=com.ibm.*=all=enabled
com.ibm.ws.scripting.appendTrace	Appends to the end of the existing log file
com.ibm.ws.scripting.profiles	List of profiles to be run before running user commands, scripts, or an interactive shell
com.ibm.ws.scripting.emitWarningForCustomSecurityPolicy	Controls whether message WASX7207W is emitted when custom permissions are found
com.ibm.ws.scripting.tempdir	Stores temporary files when installing applications
com.ibm.ws.scripting.validationLevel	Level of validation to use when configuration changes are made from the scripting interface

Property	Value
com.ibm.ws.scripting.crossDocumentValidationEnabled	Determines whether the validation mechanism examines other documents when changes are made to one document
com.ibm.ws.scripting.classpath	List of paths to search for classes and resources

Some of the listed properties in the `wsadmin.properties` file are commented out by default. An example is `com.ibm.ws.scripting.traceString`. If you want to trace `wsadmin` execution, remove the comment sign # from the properties file.

Some of the properties contain default values. For example, `com.ibm.ws.scripting.connectionType` has a default value of SOAP. This means when a scripting process is invoked, a SOAP connector is used to communicate with the server. The `com.ibm.ws.scripting.defaultLang` property is set to Jacl.

Example: Specifying a properties file (-p)

Use the `-p` option to specify a customized properties file. Example 8-3 shows sample coding for invoking `wsadmin` to execute a script file using a specific properties file.

Example 8-3 specifying properties file on the command line

```
C:\WebSphereV7\AppServer\profiles\dmgr40\bin>wsadmin -p
c:\webspherev7\appserver
\profiles\dmgr40\properties\wsadmin_custom.properties
WASX7209I: Connected to process "dmgr" on node dmgr40node using SOAP
connector;
The type of process is: DeploymentManager
WASX7031I: For help, enter: "print Help.help()"
```

8.2.2 Script profile file

A script profile is a script that is invoked before the main script or before invoking `wsadmin` in interactive mode. The purpose of the script profile is to customize the environment in which script runs. For example, a script profile can be set for the Jacl scripting language that makes Jacl-specific variables or procedures available to the interactive session or main script.

The `-profile` command-line option can be used to specify a profile script. Several `-profile` options can be used on the command line and are invoked in the order given.

8.2.3 Connected versus local mode

The `wsadmin` command can operate in either connected or local mode. In connected mode, all operations are performed by method invocations on running JMX MBeans. In local mode, the application server (MBeans server) is not started and the `wsadmin` objects are limited to configuring the server by means of directly manipulating XML configuration documents.

When operating in local mode, be sure that you are operating on the correct profile, by either using the `-profileName` argument or starting `wsadmin` from the `profile/bin` directory.

When performing configuration changes in local mode in a distributed server environment, care should be taken to make configuration changes at the deployment manager level. Changes made directly to the node configuration will be lost at server startup or at configuration replication.

Use the `-conntype NONE` option to run in local mode.

8.3 Command and script invocation

The `wsadmin` commands can be invoked in three different ways. This section describes how to invoke the command.

Note: For simplicity, the examples in this chapter will assume that:

- ▶ `wsadmin` is executed from the `profile_root/bin` directory, so it is not necessary to specify the profile name, host, and port.
- ▶ Administrative security is disabled. In reality, you will need to specify the username and password when you invoke `wsadmin`.

Invoking a single command (-c)

The `-c` option is used to execute a single command using `wsadmin` in Example 8-4. In the example, we use the `AdminControl` object to query the node name of the WebSphere server process.

Example 8-4 Running a single command in wsadmin

```
C:\WebSphereV7\AppServer\profiles\jmgr40\bin>wsadmin -lang jython -c
AdminControl.getNode()
WASX7209I: Connected to process "jobmgr" on node jmgr40node using SOAP
connector
; The type of process is: JobManager
```

```
'jmgr40node'  
C:\WebSphereV7\AppServer\profiles\jmgr40\bin>
```

Running script files (-f)

The -f option is used to execute a script file. Example 8-5 shows a two-line Jython script named myScript.py. The script has a .py extension to reflect the Jython language syntax of the script. The extension plays no significance in **wsadmin**; the com.ibm.ws.scripting.defaultLang property or -lang parameter is used to determine the language used. If the property setting is not correct, use the -lang option to identify the scripting language, because the default is Jacl.

Example 8-5 Jython script

```
print "This is an example Jython script"  
print(""+ AdminControl.getNode()+"")
```

Example 8-6 shows how to execute the script.

Example 8-6 Running a Jython script in wsadmin

```
C:\WebSphereV7\AppServer\profiles\dmgr40\bin>wsadmin -f myScript.py  
-lang jython
```

```
WASX7209I: Connected to process "dmgr" on node dmgr40node using SOAP  
connector;  
The type of process is: DeploymentManager  
This is an example Jython script  
dmgr40node
```

Invoking commands interactively

The command execution environment can be run in interactive mode, so you can invoke multiple commands without having the overhead of starting and stopping the **wsadmin** environment for every single command. Run the **wsadmin** command without the command (-c) or script file (-f) options to start the interactive command execution environment, as shown in Example 8-7.

Example 8-7 starting the wsadmin interactive command execution environment

```
C:\WebSphereV7\AppServer\profiles\dmgr40\bin>wsadmin -lang jython  
WASX7209I: Connected to process "dmgr" on node dmgr40node using SOAP  
connector;  
The type of process is: DeploymentManager  
WASX7031I: For help, enter: "print Help.help()"
```

```
wsadmin>
```

From the wsadmin> prompt, the WebSphere administrative objects and built-in language objects can be invoked, as shown in Example 8-8. Simply type the commands at the wsadmin> prompt.

Example 8-8 interactive command invocation

```
wsadmin>AdminControl.getNode()  
'dmgr40node'  
wsadmin>
```

End the interactive execution environment by typing **quit** and pressing the Enter key.

8.4 wsadmin management objects

Five management objects are available for wsadmin, as shown in Figure 8-1. Prior to V7, the use of these objects in a wsadmin script could be quite complex and difficult to script. Command assist was available in the administrative console to provide the equivalent script commands for certain tasks performed in the console, but putting these commands to use in a script required some work.

New in V7: WebSphere Application Server V7.0 includes script libraries that can simplify the use of these objects. For that reason, in this chapter about the use of the management objects, refer to the Information Center. Note that the command assist feature has been expanded in the administrative console to include more tasks. Using the script libraries and command assist will be discussed more fully in later sections.

The wsadmin tool has the following five administrative objects that provide server configuration and management capabilities:

- ▶ Help
- ▶ AdminControl
- ▶ AdminConfig
- ▶ AdminApp
- ▶ AdminTask

8.4.1 Help

The Help object provides a quick way to get information about methods, operations, and attributes while using scripting.

For example, to get a list of the public methods available for the AdminControl object, enter the following command as shown:

```
wsadmin>print Help.AdminConfig()
```

To get a detailed description of a specific object method and the parameters it requires, invoke the help method of the target object with the method name as the option to the help method, as shown in Example 8-9.

Example 8-9 AdminConfig.help scripting

```
wsadmin>print AdminConfig.help("createClusterMember")
```

WASX7284I: Method: createClusterMember

Arguments: cluster id, node id, member attributes

Description: Creates a new Server object on the node specified by "node id." This Server is created as a new member of the existing cluster specified by "cluster id," and has attributes specified in "member attributes." One attribute is required: "memberName." The Server is created using the default template for Server objects, and has the name and specified by the "memberName" attribute.

attribute.

Method: createClusterMember

Arguments: cluster id, node id, member attributes, template id

Description: Creates a new Server object on the node specified by "node id." This Server is created as a new member of the existing cluster specified by "cluster id," and has attributes specified in "member attributes." One attribute is required: "memberName." The Server is created using the Server template specified by "template id," and has the name specified by the "memberName" attribute.

(New in V7) The AdminTask object supports a new help function that allows you to search for the specific command by using a wildcard character. For example, the command to get a list of the command names that start with "create" is shown in Example 8-10.

Example 8-10 AdminTask.help scripting

```
wsadmin>print AdminTask.help("-commands", "create*")
WASX8004I: Available admin commands:

createAllActivePolicy - Create a policy that automatically activates all group
members.
createApplicationServer - Command that creates a server
createApplicationServerTemplate - creates a server Template based on a server
configuration
createAuditEncryptionConfig - Configures audit record encryption.
createAuditEventFactory - Creates an entry in the audit.xml to reference the
configuration of a
    Factory interface.
createAuditFilter - Creates an entry in the audit.xml to reference an Audit
Specification. Enab
createAuditKeyStore - Creates a new Key Store.
createAuditNotification - Configures an audit notification.
createAuditNotificationMonitor - Configures an audit notification monitor.
createAuditSelfSignedCertificate - Create a new self-signed certificate and store it
in a keys
....
```

8.4.2 AdminControl

The AdminControl object is used for operational control. It communicates with MBeans that represent live objects running a WebSphere server process. It includes commands to query existing running objects and their attributes and invoke operations on the objects. In addition to the operational commands, the AdminControl object supports commands to query information about the connected server, convenient commands for client tracing, reconnecting to a server, and starting and stopping a server.

Note that because the AdminControl object operates on live MBeans, it cannot be used to start a deployment manager, node agent, or standalone application server.

8.4.3 AdminConfig

The AdminConfig object is used to manage the configuration information that is stored in the repository. This object communicates with the WebSphere Application Server configuration service component to make configuration inquiries and changes. You can use it to query existing configuration objects,

create configuration objects, modify existing objects, and remove configuration objects. In a distributed server environment, the AdminConfig commands are available only if a scripting client is connected to the deployment manager. When connected to a node agent or a managed application server, the AdminConfig commands will not be available because the configuration for these server processes are copies of the master configuration that resides in the deployment manager.

8.4.4 AdminApp

The AdminApp object can update application metadata, map virtual hosts to Web modules, and map servers to modules for applications already installed. Changes to an application, such as specifying a library for the application to use or setting session management configuration properties, are performed using the AdminConfig object.

8.4.5 AdminTask

The AdminTask object is used to access a set of task-oriented administrative commands that provide an alternative way to access the configuration commands and the running object management commands. The administrative commands run simple and complex commands. The administrative commands are discovered dynamically when the scripting client is started. The set of available administrative commands depends on the edition of WebSphere Application Server you install. You can use the AdminTask object commands to access these commands.

Two run modes are always available for each administrative command, namely the batch and interactive mode. When you use an administrative command in interactive mode, you go through a series of steps to collect your input interactively. This process provides users a text-based wizard and a similar user experience to the wizard in the administrative console. You can also use the help command to obtain help for any of the administrative commands and the AdminTask object.

New functions have been added to the AdminTask object to support V7 features, such as the certificate authority, business level applications, proxy management, and to support the use of properties file based configurations.

(New in V7) Properties file based configuration

Using complex scripts requires knowledge of Jacl or Jython scripting language, as well as the public MBean interfaces. The use of properties file based configuration provides a way to simplify administrative tasks using wsadmin.

To implement this type of configuration, five new commands have been added to the AdminTask object in the PropertiesBasedConfiguration command group:

- ▶ extractConfigProperties
- ▶ validateConfigProperties
- ▶ applyConfigProperties
- ▶ deleteConfigProperties
- ▶ createPropertiesFileTemplates

Properties file based configuration extracts configuration data to one file that is easy to read using any editor tool. The configuration attributes are provided as key/value pairs (Figure 8-2).

```
#  
# SubSection 1.0 # JDBCProvider attributes  
#  
ResourceType=JDBCProvider  
ImplementingResourceType=JDBCProvider  
ResourceId=Cell!=!{cellName}:JDBCProvider=ID#builtin_jdbcprovider  
#  
  
#  
#Properties  
#  
classpath=${DERBY_JDBC_DRIVER_PATH}/derby.jar  
name=Derby JDBC Provider (XA)  
implementationClassName=org.apache.derby.jdbc.EmbeddedXADataSource  
nativepath={}  
description=Built-in Derby JDBC Provider (XA)  
providerType=Derby JDBC Provider (XA) #readonly  
xa=true #boolean
```

1. Resource type and Identifier

2. Configuration Information

Figure 8-2 Example: properties for a JDBC provider

After the properties have been extracted, you make any necessary changes to the attributes and validate the changes before applying them to the server. It is also possible to create or delete configuration objects.

Samples of these commands are shown in Example 8-11.

Example 8-11 wsadmin commands for properties based configuration

```
AdminTask.extractConfigProperties('-configData Node=myNode  
-propertiesFileName myNodeProperties.props')
```

```
AdminTask.validateConfigProperties('-propertiesFileName  
myNodeProperties.props -reportFile report.txt')
```

```
AdminTask.applyConfigProperties('-propertiesFileName myPropFile.props  
-validate true')

AdminTask.deleteConfigProperties('-propertiesFileName  
myPropFile.props')

AdminTask.createPropertiesFileTemplates('-propertiesFileName  
serverTemplate.props -configType Server')
```

Because it is not possible to modify every configuration using properties file based configuration, we do *not* recommend that you use this tool for backup and recovery. BackupConfig and RestoreConfig found in the <was_home>/bin directory should still be used as the main backup and recovery tool.

8.5 Managing WebSphere using script libraries

(New in V7) Script libraries can be used to perform a higher level of wsadmin functions than can be done using a single wsadmin command. Only a single line from a library function is needed to perform complex functions. Each script is written in Jython, and is often referred to as “the Jython script”. The script libraries are categorized into six types and the types are subdivided (see Table 8-2).

Python script files are supplied for each Jython class file. The Python files can be read as a text files. Table 8-2 shows the scripts and the type of resource they manage. Each script has a set of procedures that perform specific functions.

Table 8-2 The types of the script library

TYPE	Python (Jython) script file
Application	AdminApplication AdminBLA
Resources	AdminJ2C AdminJDBC AdminJMS AdminResources
Security	AdminAuthorizations
Servers	AdminClusterManagement AdminServerManagement
System	AdminNodeGroupManagement AdminNodeManagement

TYPE	Python (Jython) script file
Application	AdminApplication AdminBLA
Utilities	AdminLibHelp AdminUtilities

Script libraries, their procedures, and syntax: More information about these script libraries can be found in the WebSphere Application Server Information Center as sub-topics under:

- ▶ Jython script files:

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

8.5.1 Invoking script libraries

The script libraries are located in the *install_root /scriptLibraries* directory. These libraries are loaded when wsadmin starts and are readily available from the wsadmin command prompt or to be used from the customized scripts. You can invoke the scripts in interactive mode or script mode.

Interactive mode

Interactive mode is suitable for simple tasks and testing. Using this mode allows you to get the results directly. To invoke a script interactively, start a wsadmin session and enter the script name, procedure, and arguments at the wsadmin> prompt (see Example 8-12).

Example 8-12 Using the Jython scripts in interactive mode

```
C:\WebSphere\AppServer\bin>wsadmin -lang jython
WASX7209I: Connected to process "dmgr" on node DmgrNode02 using SOAP
connector;
The type of process is: DeploymentManager
WASX7031I: For help, enter: "print Help.help()"
wsadmin>AdminServerManagement.checkIfServerExists("sys1Node01",
"Amember01")
-----
AdminServerManagement: Check if server exists
Node name:          sys1Node01
Server name:        Amember01
```

```
Usage: AdminServerManagement.checkIfServerExists("sys1Node01",
"Amember01")
```

```
'true'
wsadmin>
```

Script file mode (-f)

Using a script file with wsadmin is useful when you want to have daily tasks performed automatically or if you need to manage multiple servers.

To run in script mode, you select the script libraries to use and merge them into your own script file. Save the custom script as a Python file and run it from the command line.

Example 8-13 shows a Python file containing two script library commands.

Example 8-13 test.py script

```
# Writting by Jython
# Script file name : test.py

AdminServerManagement.checkIfServerExists("sys1Node01", "Amember21")
AdminServerManagement.createApplicationServer("sys1Node01",
"Amember21")
```

Example 8-14 shows how to invoke the script.

Example 8-14

```
C:\WebSphere\AppServer\bin>
C:\WebSphere\AppServer\bin>wsadmin.bat -lang jython -f C:\temp\test.py
```

Customizing scripts

This is an advanced use of the script mode. You can add customized code written in Python or Jython to your script file (the one that calls the script libraries).

Note: *DO NOT* modify the script libraries. If you need to customize the scripts, you can copy parts of the library code to other files and modify the copied code to improve it or to better suit your needs.

When customizing a script, it is best to do this in three steps:

1. Run each Jython script in interactive mode, verifying the syntax and the result.
2. Create the script file that will call the script libraries, combining all Jython scripts. Verify that the results are as you expect.
3. Add your additional `wsadmin` commands, written in Python, and verify that the customized script does the work that you intended.

8.5.2 Displaying help for script libraries

You can use AdminLibHelp script to display each script within a script library. For example, the following command invocation displays each script in the AdminApplication script library:

```
print AdminLibHelp.help("AdminApplication")
```

You can use the help script to display detailed descriptions, arguments, and usage information for a specific script. For example, the following command invocation displays detailed script information for the listApplications script in the AdminApplication script library:

```
print AdminApplication.help('listApplications')
```

Example 8-15 shows sample code for displaying help information for the createApplicationServer procedure in the AdminServerManagement script.

Example 8-15 Help information for a procedure in createApplicationServer

```
wsadmin>print AdminServerManagement.help('createApplicationServer')
WASL2004I: Procedure: createApplicationServer

    Arguments: nodeName, serverName, (Optional) templateName

    Description: Create a new application server

    Usage: AdminServerManagement.createApplicationServer( nodeName,
serverName, templateName)
wsadmin>
```

8.5.3 Application script library

The application scripts provide a set of procedures to manage and configure enterprise applications and business level applications. These scripts can be used individually, or combined in a custom script file to automate application installation, configuration, and management tasks.

The library that is located in the *install_root/scriptlibraries/application/V70* directory contains two scripts:

- ▶ AdminApplication, which provides procedures to manage enterprise applications
- ▶ AdminBLA, which provides procedures to manage business level applications

We discuss the AdminApplication script in the next section. For information about the AdminBLA script, see:

- ▶ Business-level application configuration scripts

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rxml_7libbla.html

AdminApplication script

The AdminApplication script contains procedures that allow you to manage enterprise applications.

The syntax of the AdminApplication script is:

`AdminApplication.procedure_name(arguments)`

The following list summarizes the script procedures available with this script. The name of the procedure gives you a good idea of the function it provides.

- ▶ Install and uninstall applications:
 - installAppWithAppNameOption
 - installAppWithDefaultBindingOption
 - installAppWithNodeAndServerOptions
 - installAppWithClusterOption
 - installAppModulesToSameServerWithMapModulesToServersOption
 - installAppModulesToDiffServersWithMapModulesToServersOption
 - installAppModulesToSameServerWithPatternMatching
 - installAppModulesToDiffServersWithPatternMatching
 - installAppModulesToMultiServersWithPatternMatching
 - installAppWithTargetOption
 - installAppWithDeployEjbOptions
 - installAppWithVariousTasksAndNonTasksOptions
 - installWarFile
 - uninstallApplication
- ▶ Update applications:
 - addSingleFileToAnAppWithUpdateCommand
 - addSingleModuleFileToAnAppWithUpdateCommand
 - addUpdateSingleModuleFileToAnAppWithUpdateCommand
 - addPartialAppToAnAppWithUpdateCommand

- deleteSingleFileToAnAppWithUpdateCommand
 - deleteSingleModuleFileToAnAppWithUpdateCommand
 - deletePartialAppToAnAppWithUpdateCommand
 - updateApplicationUsingDefaultMerge
 - updateApplicationWithUpdateIgnoreNewOption
 - updateApplicationWithUpdateIgnoreOldOption
 - updateEntireAppToAnAppWithUpdateCommand
 - updatePartialAppToAnAppWithUpdateCommand
 - updateSingleFileToAnAppWithUpdateCommand
 - updateSingleModuleFileToAnAppWithUpdateCommand
- ▶ Export applications:
- exportAnAppToFile
 - exportAllApplicationsToDir
 - exportAnAppDDLToDir
- ▶ Configure application deployment characteristics:
- configureStartingWeightForAnApplication
 - configureClassLoaderPolicyForAnApplication
 - configureClassLoaderLoadingModeForAnApplication
 - configureSessionManagementForAnApplication
 - configureApplicationLoading
 - configureLibraryReferenceForAnApplication
 - configureEJBModulesOfAnApplication
 - configureWebModulesOfAnApplication
 - configureConnectorModulesOfAnApplication
- ▶ Start and stop enterprise and business level applications:
- startApplicationOnSingleServer
 - startApplicationOnAllDeployedTargets
 - startApplicationOnCluster
 - stopApplicationOnSingleServer
 - stopApplicationOnAllDeployedTargets
 - stopApplicationOnCluster
- ▶ Query applications:
- checkIfAppExists
 - getAppDeployedNodes
 - getAppDeploymentTarget
 - getTaskInfoForAnApp
 - listApplications
 - listApplicationsWithTarget
 - listModulesInAnApp

Example: Installing an application

There are multiple procedures that can be used to install applications. When preparing to install an application, determine the options you will need and choose the procedure accordingly.

The most basic procedure is `installAppWithNodeAndServerOptions`. If the installation is successful, the message returns, installed application successfully.

The syntax to invoke this procedure is:

```
AdminApplication.installAppWithNodeAndServerOptions(app_name,  
app_location, node_name, app_server_Name)
```

Example 8-16 illustrates this procedure.

Example 8-16 Install application script library

```
wsadmin>AdminApplication.installAppWithNodeAndServerOptions("IBMUTC",  
"C:/IBMUTC.ear", "sys1Node01", "Amember01")  
-----  
AdminApplication:      Install application with -node and -server  
options  
Application name:     IBMUTC  
Ear file to deploy:   C:/IBMUTC.ear  
Node name:            sys1Node01  
Server name:          Amember01  
Usage: AdminApplication.installAppWithNodeAndServerOptions("IBMUTC",  
"C:/IBMUTC  
.ear", "sys1Node01", "Amember01")  
-----  
WASX7327I: Contents of was.policy file:grant codeBase  
"file:${application}" {permission java.security.AllPermission;  
};  
ADMA5016I: Installation of IBMUTC started.  
ADMA5058I: Application and module versions are validated with versions  
of deploy  
ment targets.  
  
CWSAD0040I: The application IBMUTC is configured in the Application  
Server repos  
itory.  
ADMA5113I: Activation plan created successfully.  
ADMA5011I: The cleanup of the temp directory for application IBMUTC is  
complete.
```

```
ADMA5013I: Application IBMUTC installed successfully.  
OK: installAppWithNodeAndServerOptions('IBMUTC', 'C:/IBMUTC.ear',  
'sys1Node01',  
'Amember01', 'false'):
```

Example: Starting an application

Multiple procedures are also available to start an application. To start the application, choose the most suitable script.

The startApplicationSingleServer procedure starts a single application on a single application server. The syntax is as follows:

```
AdminApplication.startApplicationOnSingleServer(app_name, node_name,  
app_server_name)
```

Example 8-17 illustrates this procedure.

Example 8-17 start application script library

```
wsadmin>AdminApplication.startApplicationOnSingleServer("IBMUTC", "sys1N  
ode01", "Amember01")  
-----  
AdminApplication:      Start an application on a single server  
Application name:    IBMUTC  
Node name:           sys1Node01  
Server name:         Amember01  
Usage: AdminApplication.startApplicationOnSingleServer("IBMUTC",  
"sys1Node01", "Amember01")  
-----  
OK: startApplicationOnSingleServer('IBMUTC', 'sys1Node01', 'Amember01',  
'false')  
:1
```

8.5.4 Resource script library

The Resource script library provides a set of scripts to manage WebSphere resources. The library provides script functions for J2C resources, JDBC providers, and JMS resources at the server scope. If you need to configure resources at the cell, node, or cluster level, it is possible to customize the scripts for this purpose.

The script library is located in the *install_root/scriptlibraries/resources/V70* directory. It contains the following scripts and procedures:

► AdminJ2C script:

The following procedures allow you to configure J2C resources:

- createJ2CActivationSpec
- createJ2CAdminObject
- createJ2CConnectionFactory
- installJ2CResourceAdapter

The following procedures allow you to query J2C resources:

- listAdminObjectInterfaces
- listConnectionFactoryInterfaces
- listJ2CActivationSpecs
- listJ2CAdminObjects
- listJ2CConnectionFactories
- listJ2CResourceAdapters
- listMessageListenerTypes

► AdminJDBC script:

The following procedures allow you to configure JDBC resources:

- createDataSource
- createDataSourceUsingTemplate
- createJDBCProvider
- createJDBCProviderUsingTemplate

The following procedures allow you to query JDBC resources:

- listDataSources
- listDataSourceTemplates
- listJDBCProviders
- listJDBCProviderTemplates

► AdminJMS script:

The following procedures allow you to configure JMS resources:

- createGenericJMSSConnectionFactory
- createGenericJMSSConnectionFactoryUsingTemplate
- createGenericJMSSDestination
- createGenericJMSSDestinationUsingTemplate
- createJMSProvider
- createJMSProviderUsingTemplate
- createWASQueue
- createWASQueueUsingTemplate
- createWASQueueConnectionFactory

- createWASQueueConnectionFactoryUsingTemplate
- createWASTopic
- createWASTopicUsingTemplate
- createWASTopicConnectionFactory
- createWASTopicConnectionFactoryUsingTemplate
- startListenerPort

The following procedures allow you to query JMS resources:

- listGenericJMSConnectionFactories
- listGenericJMSConnectionFactoryTemplates
- listGenericJMSDestinations
- listGenericJMSDestinationTemplates
- listJMSProviders
- listJMSPublisherTemplates
- listWASQueueConnectionFactoryTemplates
- listWASQueueTemplates
- listWASTopicConnectionFactoryTemplates
- listWASQueueConnectionFactories
- listWASQueues
- listWASTopicConnectionFactories
- listWASTopics
- listWASTopicTemplates

► **AdminResources:**

Use the following script procedures to configure your mail settings:

- createCompleteMailProvider
- createMailProvider
- createMailSession
- createProtocolProvider

Use the following script procedures to configure your resource environment settings:

- createCompleteResourceEnvProvider
- createResourceEnvEntries
- createResourceEnvProvider
- createResourceEnvProviderRef

Use the following script procedures to configure your URL provider settings:

- createCompleteURLProvider
- createURL

Use the following script procedures to configure additional Java Enterprise Edition (JEE) resources:

- createJAASAuthenticationAlias
- createLibraryRef

- createSharedLibrary
- createScheduler
- createWorkManager

Example: Listing JDBC resources

The listDataSources and listJDBCProviders procedures of the AdminJDBC script can be used to display a list of configuration IDs for the JDBC providers and data sources in your environment.

The syntax to use each procedure is:

- ▶ AdminJDBC.listDataSources(*ds_name*)
- ▶ AdminJDBC.listJDBCProviders(*jdbc_name*)

Example 8-18 shows the use of these procedures.

Example 8-18 list up JDBC resources

```
wsadmin>AdminJDBC.listDataSources("PLANTSDB")
-----
AdminJDBC:           listDataSources
Optional parameter:
DataSource name:    PLANTSDB
Usage: AdminJDBC.listDataSources("PLANTSDB")

-----
['PLANTSDB(cells/Cell02/nodes/sys1Node01/servers/server1|resources.xml#'
DataSource_1183122165968)']
wsadmin>
wsadmin>AdminJDBC.listJDBCProviders("Derby JDBC Provider")
-----
AdminJDBC:           listJDBCProviders
Optional parameter:
JDBC provider name: Derby JDBC Provider
Usage: AdminJDBC.listJDBCProvider("Derby JDBC Provider")

-----
["Derby JDBC
Provider(cells/Cell02/nodes/sys1Node01/servers/server1|resources.x
ml#JDBCProvider_1183122153343)"]
```

Example: Creating a J2C connection factory

The createJ2CConnectionFactory procedure in the AdminJ2C script creates a new J2C connection factory in the environment. The result is the configuration ID of the new J2C connection factory.

The arguments are the resource adapter, connection factory name, the connection factory interface, and the Java Naming and Directory Interface (JNDI) name arguments. The syntax is:

```
AdminJ2C.createJ2CConnectionFactory(resource_adapterID,  
                                     connfactory_name, connFactory_interface, jndi_name, attributes)
```

Example 8-19 shows sample coding for creating a J2C connection factory.

Example 8-19 createJ2CConnectionFactory

```
wsadmin>AdminJ2C.createJ2CActivationSpec("WebSphere MQ Resource  
Adapter(cells/Cel102/nodes/DmgrNode02/servers/dmgr|resources.xml#J2CRes  
ourceAdapter_1234298429000)", "WebSphere MQ Resource Adapter",  
"javax.jms.MessageListener", "jdbc/PlantsByWebSphereDataSourceNONJTA")  
-----  
AdminJ2C:                      createJ2CActivationSpec  
J2CResourceAdapter configID:    WebSphere MQ Resource  
Adapter(cells/Cel102/nodes/DmgrNode02/servers/dmgr|resources.xml#J2CRes  
ourceAdapter_1234298429000)  
J2CActivationSpec name:          WebSphere MQ Resource Adapter  
Message listener type:          javax.jms.MessageListener  
jndi name:                     jdbc/PlantsByWebSphereDataSourceNONJTA  
Optional attributes:  
  otherAttributesList []  
Usage: AdminJ2C.createJ2CActivationSpec("WebSphere MQ Resource  
Adapter(cells/Cel102/nodes/DmgrNode02/servers/dmgr|resources.xml#J2CRes  
ourceAdapter_1234298429000)", "WebSphere MQ Resource Adapter",  
"javax.jms.MessageListener", "jdbc/PlantsB  
yWebSphereDataSourceNONJTA")  
-----  
' "WebSphere MQ Resource  
Adapter(cells/Cel102/nodes/DmgrNode02/servers/dmgr|resou  
rces.xml#J2CActivationSpec_1236206121468)"'  
wsadmin>
```

8.5.5 Security script library

The security script library provides a script to manage the security. This library is located in the `install_root/scriptlibraries/security/V70` directory.

The AdminAuthorizations script has the following procedures:

- ▶ Configure authorization groups:
 - `addResourceToAuthorizationGroup`
 - `createAuthorizationGroup`
 - `mapGroupsToAdminRole`
 - `mapUsersToAdminRole`
- ▶ Remove users and groups from the security authorization settings:
 - `deleteAuthorizationGroup`
 - `removeGroupFromAllAdminRoles`
 - `removeGroupsFromAdminRole`
 - `removeResourceFromAuthorizationGroup`
 - `removeUserFromAllAdminRoles`
 - `removeUsersFromAdminRole`
- ▶ Query your security authorization group configuration:
 - `listAuthorizationGroups`
 - `listAuthorizationGroupsForUserID`
 - `listAuthorizationGroupsForGroupID`
 - `listAuthorizationGroupsOfResource`
 - `listUserIDsOfAuthorizationGroup`
 - `listGroupIDsOfAuthorizationGroup`
 - `listResourcesOfAuthorizationGroup`
 - `listResourcesForUserID`

Example: Listing the authorization groups

The `listAuthorizationGroups` procedure displays each authorization group in the security configuration. This script does not require arguments.

```
AdminAuthorizations.listAuthorizationGroups()
```

Example 8-20 shows sample coding for listing the authorization groups.

Example 8-20 Listing authorization groups

```
wsadmin>AdminAuthorizations.listAuthorizationGroups()
```

```
-----  
AdminAuthorizations: List authorization groups  
Usage: AdminAuthorizations.listAuthorizationGroups()  
-----
```

```
['sec_group1']
```

8.5.6 Server script library

The server script library provides a set of scripts and their procedures to manage the server and the cluster component.

This library is located in the *install_root/scriptlibraries/servers/V70* directory. The library contains the following scripts and procedures:

- ▶ AdminServerManagement script:

Start and stop servers:

- startAllServers
- startSingleServer
- stopAllServers
- stopSingleServer

Configure servers:

- createApplicationServer
- createAppServerTemplate
- createGenericServer
- createWebServer
- deleteServer
- deleteServerTemplate

Query the server configuration:

- checkIfServerExists
- checkIfServerTemplateExists
- getJavaHome
- getServerProcessType
- getServerPID
- help
- listJVMProperties
- listServers
- listServerTemplates
- listServerTypes
- queryingMBeans
- showServerInfo
- viewingProductInformation

Manage server settings:

- configureAdminService
- configureApplicationServerClassloader

- configureDynamicCache
 - configureEJBContainer
 - configureFileTransferService
 - configureListenerPortForMessageListenerService
 - configureMessageListenerService
 - configureStateManageable
 - configureCustomProperty
 - configureCustomService
 - configureEndPointsHost
 - configureJavaVirtualMachine
 - configureORBService
 - configureProcessDefinition
 - configureRuntimeTransactionService
 - configureThreadPool
 - configureTransactionService
 - setJVMProperties
 - setTraceSpecification
 - configureCookieForServer
 - configureHTTPTransportForWebContainer
 - configureSessionManagerForServer
 - configureWebContainer
 - configureJavaProcessLogs
 - configurePerformanceMonitoringService
 - configurePMIRequestMetrics
 - configureRASLoggingService
 - configureServerLogs
 - configureTraceService
- AdminClusterManagement script
- Start cluster processes:
- rippleStartAllClusters
 - rippleStartSingleCluster
 - startAllClusters
 - startSingleCluster
- Stop cluster processes:
- immediateStopAllRunningClusters
 - immediateStopSingleCluster
 - stopAllClusters
 - stopSingleCluster
- Configure clusters:
- createClusterMember
 - createClusterWithFirstMember
 - createClusterWithoutMember

- createFirstClusterMemberWithTemplate
- createFirstClusterMemberWithTemplateNameServer

Remove clusters and cluster members:

- deleteCluster
- deleteClusterMember

Query a cluster configuration:

- checkIfClusterExists
- checkIfClusterMemberExists
- help
- listClusters
- listClusterMembers

Example: Creating an application server

The CreateApplicationServer procedure of the AdminServerManagement script creates a new application server. The script requires the node to be running. The syntax is:

```
AdminServerManagement.createApplicationServer(node_name,  
server_name,Template)
```

Example 8-21 illustrates sample coding for creating an application server.

Example 8-21 Creating an application server

```
wsadmin>AdminServerManagement.createApplicationServer("sys1Node01", "Ame  
mber01", "default")  
-----  
AdminServerManagement: Create an application server on a given node  
Node name: sys1Node01  
New Server name: Amember01  
Optional parameter:  
Template name: default  
Usage: AdminServerManagement.createApplicationServer("sys1Node01",  
"Amember01", "default")  
-----  
'Amember01(cells/Cel102/nodes/sys1Node01/servers/Amember01|server.xml#Server_12  
35061945890)'
```

Example: Starting an application server

The StartAllServers procedure of the AdminServerManagement script starts all application servers on a node. StartSingleServer starts one server. The syntax for these two procedures is as follows:

- ▶ AdminServerManagement.startAllServers(*node_name*)
- ▶ AdminServerManagement.startSingleServer(*node_name*, *server_name*)

Example 8-22 shows sample coding for starting an application server.

Example 8-22 Starting the application server using a single script library

```
wsadmin>AdminServerManagement.startAllServers("sys1Node01")
```

```
-----  
AdminServerManagement: Start all servers  
Node name: sys1Node01  
Usage: AdminServerManagement.startAllServers("sys1Node01")
```

```
-----  
Start server: Amember01  
Start server: server1  
OK: startAllServers('sys1Node01', 'false'):1
```

```
wsadmin>AdminServerManagement.startSingleServer("sys1Node01", "Amember01")
```

```
-----  
AdminServerManagement: Start single server  
Node name: sys1Node01  
Server name: Amember01  
Usage: AdminServerManagement.startSingleServer("sys1Node01", "Amember01")
```

```
-----  
Start server: Amember01  
OK: startSingleServer('sys1Node01', 'Amember01', 'false'):1
```

Example: Stopping application servers

The StopAllServers procedure stops all application servers on a node. The StopSingleServer will stop one server. The syntax for these procedures is:

- ▶ AdminServerManagement.stopAllServers(*node_name*)
- ▶ AdminServerManagement.stopSingleServer(*node_name*, *server_name*)

Example 8-23 shows sample coding for stopping application servers.

Example 8-23 Stopping the application server using a single script library

```
wsadmin>AdminServerManagement.stopAllServers("sys1Node01")
```

```

-----
AdminServerManagement: Stop all servers
Node name: sys1Node01
Usage: AdminServerManagement.stopAllServers("sys1Node01")
-----
Stop server: Amember01
WASX7337I: Invoked stop for server "Amember01" Waiting for stop
completion.
Stop server: server1
WASX7337I: Invoked stop for server "server1" Waiting for stop
completion.
OK: stopAllServers('sys1Node01', 'false'):1

wsadmin>AdminServerManagement.stopSingleServer("sys1Node01",
"Amember01")
-----
AdminServerManagement: Stop single server
Node name: sys1Node01
Server name: Amember01
Usage: AdminServerManagement.stopSingleServer("sys1Node01",
"Amember01")
-----
Stop server: Amember01
WASX7337I: Invoked stop for server "Amember01" Waiting for stop
completion.
OK: stopSingleServer('sys1Node01', 'Amember01', 'false'):1

```

8.5.7 System management script library

The system management script library provides a set of scripts that manage nodes and node groups.

This library is located in the *install_root/scriptlibraries/system/V70* directory. It contains the following scripts and procedures:

- ▶ AdminNodeManagement
 - configureDiscoveryProtocolOnNode
 - doesNodeExist
 - isNodeRunning
 - listNodes
 - restartActiveNodes
 - restartNodeAgent
 - stopNode
 - stopNodeAgent

- syncActiveNodes
- syncNode
- ▶ AdminNodeGroupManagement
 - addNodeGroupMember
 - checkIfNodeExists
 - checkIfNodeGroupExists
 - createNodeGroup
 - createNodeGroupProperty
 - deleteNodeGroup
 - deleteNodeGroupMember
 - deleteNodeGroupProperty
 - help
 - listNodeGroups
 - listNodeGroupMembers
 - listNodeGroupProperties
 - modifyNodeGroup
 - modifyNodeGroupProperty

Example: Querying node group members

The listNodeGroupMembers procedure lists the name of each node that is configured within a specific node group. The syntax is:

```
AdminNodeGroupManagement.listNodeGroupMembers(node_group_name)
```

Example 8-24 shows sample coding for querying node group members.

Example 8-24 listing node group members using script library

```
wsadmin>AdminNodeGroupManagement.listNodeGroupMembers("DefaultNodeGroup")
-----
AdminNodeGroupManagement:      List nodes for a given node group
Optional parameter:
      Node group name:      DefaultNodeGroup
Usage:
AdminNodeGroupManagement.listNodeGroupMembers("DefaultNodeGroup")
-----
['DmgrNode02', 'sys1Node01']
```

Example: Synchronizing a node

The synActiveNodes and syncNode procedures propagate a configuration change. The syntax for these commands is:

- ▶ AdminNodeManagement.syncActiveNodes()

- ▶ AdminNodeManagement.syncNode(*node_name*)

Example 8-25 shows sample coding for synchronizing a node.

Example 8-25 sync the node using scriptlibrary

```
wsadmin>AdminNodeManagement.syncNode("sys1Node01")
```

```
-----  
AdminNodeManagement:      syncNode  
nodeName:                 sys1Node01  
Usage: AdminNodeManagement.syncNode("sys1Node01")  
-----
```

```
true  
1
```

8.6 Assistance with scripting

When you perform an action in the administrative console, you can use the command assistance feature to show the corresponding scripting commands. This allows you to capture and copy them for use in wsadmin scripts. You also have the option to send these as notifications to the Rational Application Developer V7.5, where you can use the Jython editor to build scripts.

8.6.1 Enabling command assistance

The command assistance feature in the administrative console was introduced in WebSphere Application Server V6.1 with limited scope in function. The command assistance feature has been broadened in V7.0.

When you perform an action in the administrative console, you can select the **View administrative scripting command for last action** option in the Help area of the panel to display the command equivalent. This command can be copied and pasted into a script or command window.

Two additional features can be enabled to assist in developing scripts.

To enable these features:

1. Go to **System administration** → **Console Preferences**.
2. Select the command assistance features you want to use (see Figure 8-3):
 - Enable command assistance notifications:

This option should be used in non-production environments to send notifications containing command assist data. Enablement of the notifications allows integration with Rational Application Developer.

- Log command assistance commands:

This option sends the commands to a log.

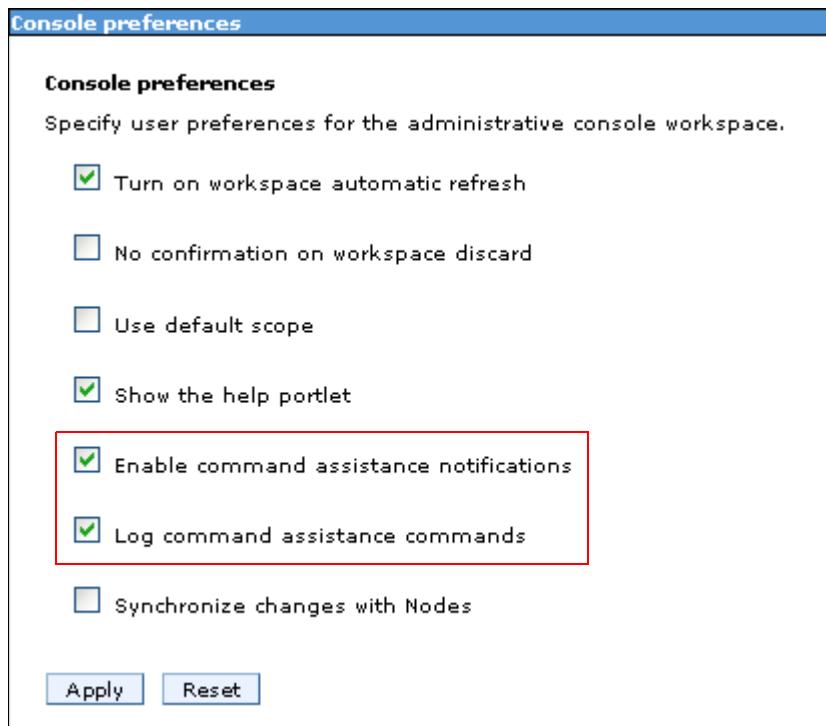


Figure 8-3 Administrative scripting command: mapping to actions

3. Click apply.

When you select the option to log commands, they are stored in the following location:

profile_root/logs/AssistanceJythonCommands_user_name.log

See Example 8-26 for some sample coding.

Example 8-26 log location

C:\WebSphere\AppServer\profiles\dmgr01\logs\dmgr
\commandAssistanceJythonCommands_wasadmin.log

The first line of each log entry consists of a timestamp and the location within the console where the command was generated. Below the timestamp, is the command information. Example 8-27 shows a sample of the log.

Example 8-27 the command assistance: log content

```
# [2/24/09 12:15:42:890 EST] Business-level applications > New
AdminTask.createEmptyBLA('[-name sample -description Sample ]')

# [2/24/09 12:15:42:906 EST] Business-level applications > New
AdminTask.listBLAs('[-blaID WebSphere:blaname=sample -includeDescription true
]')

# [2/24/09 12:15:42:906 EST] Business-level applications > New
AdminTask.listCompUnits('[-blaID WebSphere:blaname=sample -includeDescription
true -includeType true ]')

# [2/24/09 12:15:47:500 EST] Business-level applications > sample
AdminTask.listAssets('[-includeDescription true ]')
# [2/24/09 12:15:47:531 EST] Business-level applications > sample
AdminTask.listBLAs('[-includeDescription true ]')

# [2/24/09 12:15:50:531 EST] Business-level applications > sample > Add
AdminTask.addCompUnit('[-blaID WebSphere:blaname=sample -cuSourceID
WebSphere:blaname=IBMUTC ]')

# [2/24/09 12:15:53:562 EST] Business-level applications > sample > Add
AdminTask.addCompUnit('[-blaID WebSphere:blaname=sample -cuSourceID
WebSphere:blaname=IBMUTC -CUOptions [[WebSphere:blaname=sample
WebSphere:blaname=IBMUTC IBMUTC_0001 "" 1]]]')

# [2/24/09 12:15:57:625 EST] Adding composition unit to the business-level
application
AdminConfig.save()

# [2/24/09 12:15:57:890 EST] BLAManagement
AdminTask.listBLAs('[-includeDescription true ]')

# [2/24/09 12:16:01:421 EST] Business-level applications
AdminTask.startBLA('[-blaID WebSphere:blaname=sample ]')
```

8.6.2 Building script files using command assist

The command assist features provide several methods to build scripts. Commands can be copied from the Help area of the console, or from the log into Jython scripts.

The command assist notifications also provide an integration point with Rational Application Developer, which provides tools that allow you to monitor commands as they are created and to insert the monitored commands into a script.

Working with Jython scripts

To work with Jython scripts in Rational Application Developer, you create a Jython project and Jython script files in the project. This can be done in any perspective. When you open a new Jython script, it opens with the Jython editor.

The Jython editor in Rational Application Developer V7.5 is used to perform a variety of tasks, such as these:

- ▶ View the administrative console.
- ▶ Develop and edit Jython script files.
- ▶ Import existing Jython files for structured viewing.
- ▶ Set breakpoints for debugging your scripts.

The Jython editor has many text editing features, such as syntax highlighting, unlimited undo or redo, and automatic tab indentation.

When you tag a comment in a Jython script with "#TODO", the editor automatically creates a corresponding task as a reminder in the Tasks view. Then, if you open the task later, the editor automatically synchronizes to that TODO entry in the script source.

Other helpful features are content assist and tips, which provides a list of acceptable continuations depending on where the cursor is located in a Jython script file, or what you have just typed. The Jython editor is not integrated to a compiler. As a result, the Jython editor does not perform syntax verification on your scripts.

Using the command assist notifications

The command assistance in the administrative console sends JMX notifications containing command data. These notifications can be monitored from Rational Application Developer. This requires that you define the server producing the notifications as a server in the workspace.

To monitor the commands produced as actions are taken on the administrative console of the server, follow these steps:

1. In the Servers view, right-click the server and select **Administration** → **WebSphere administration command assist**. The WebSphere Administration Command view opens.
2. In the **Select Server to Monitor** list, specify the servers with a check mark that you want the tool to monitor as you interact with its administrative console. The Select Server to Monitor list is available in the toolbar of the WebSphere Administration Command view (see Figure 8-4).

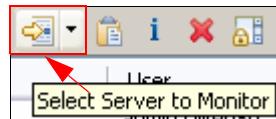


Figure 8-4 Select Server to Monitor icon

The server that you want to monitor needs to be started, started in profile or debug mode. The server is disabled for selection in the Select Server to Monitor list when the server is stopped.

As commands are generated by the console, they appear in the WebSphere Application Command Assist view. The commands are shown at WebSphere Administration Command view.

With the Jython script open in the Jython editor, and the monitored command data in the WebSphere Administration Command view, you can insert the commands directly into the script file. Place the cursor in the script file where you want the command to go. Then right-click the command and select **Insert**, as shown in Figure 8-5. Double-clicking the command also inserts it into the script.

The screenshot shows two windows. The top window is titled 'Admin Console' and contains a Jython script named 'create.py'. The script includes code to start a process and create a cluster with members. The bottom window is titled 'WebSphere Administration Command' and is a history of commands run via wsadmin. It lists several entries, each with a timestamp from 2/24/09 2:03 to 2:04.

Description	Command	Time
WASX7056I: Method: list	AdminConfig.list('com.ibm.ws.management.Admin...')	2/24/09 2:03 ...
Creates a new application ser...	AdminTask.createApplication(...)	2/24/09 2:04 ...
Creates a new member of an ...	AdminTask.createClusterMember(...)	2/24/09 2:04 ...
Creates a new member of an ...	AdminTask.createClusterMember(...)	2/24/09 2:04 ...
WASX7056I: Method: list	AdminConfig.list('com.ibm.ws.management.Admin...')	2/24/09 2:04 ...
WASX7124I: Method: save	AdminConfig.save()	2/24/09 2:04 ...

Figure 8-5 Jython editor running on automatically the command output

8.7 Example: Using scripts with the job manager

This section provides an example of how to use scripting to automate a WebSphere installation that uses a flexible management environment.

Most companies have routine tasks that occur in different phases of the software development life cycle. In an environment with multiple WebSphere Application Server installations, automation of these tasks can save a lot of time. The combination of wsadmin scripts, script libraries, and the automated management provided in a flexible management environment provides an automation solution.

8.7.1 Introduction

The scenario uses a simple WebSphere environment to illustrate how to automate tasks. The techniques here can be used in much more complex environments. This scenario contains the following steps:

1. Write a customized script to automate the tasks
2. Configure the job manager
3. Verify the results

Figure 8-6 shows the scenario environment. A single application server is configured on Node B. The deployment manager for the cell is registered to a job manager on Node A.

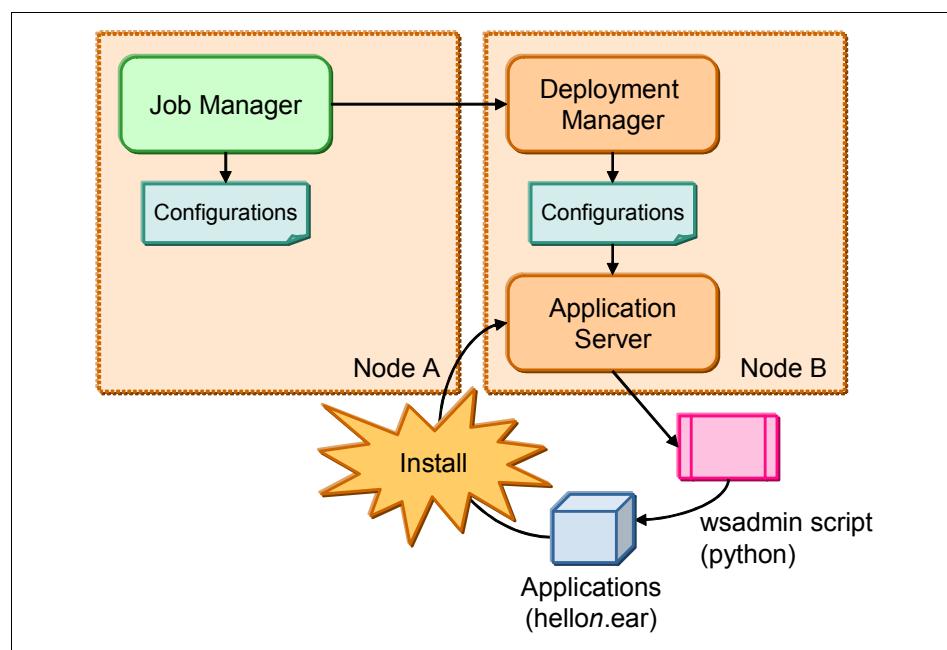


Figure 8-6 the environment details

Applications in this environment are installed frequently and the administrator needs a quick way to install these applications. Here is the plan:

1. A wsadmin script is prepared to install an application. The script will make use of the script libraries.
2. A job is scheduled to run the script at regular intervals.

3. The script checks a text file that lists the new applications to be installed.
The new application information stored in a text file includes the application name, application location, node name, and server name (see Figure 8-7).
4. If the job runs and finds that the application is not installed, the application is installed. If the application is already installed, it is uninstalled and then re-installed.
5. The text file is renamed *filename.txt.old* when the job is executed.
6. If the job executes and no text file exists, no actions are taken. It is only when you distribute a new text file and application that the job will perform the install.

To test, two applications are prepared; hello1.ear and hello2.ear.

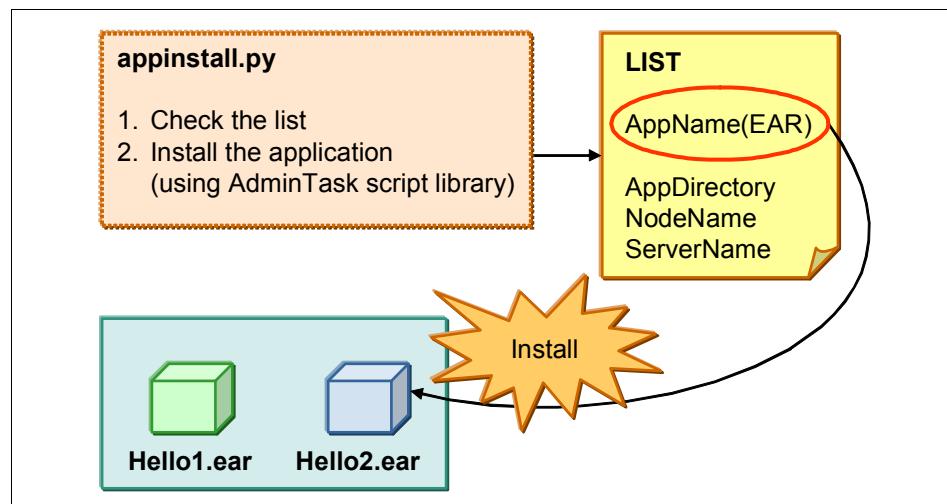


Figure 8-7 Scripting details

8.7.2 Creating the customized script

The AdminApplication script in the script libraries has been identified as having procedures that will accomplish the installation and update of applications.

In this scenario, two procedures from the script libraries were identified for use:

- ▶ AdminApplication.uninstallApplication

This procedure will be used to uninstall an application. The arguments specify the application name.

- ▶ AdminApplication.installAppWithNodeAndServerOptions:
This procedure will be used to install an application. This procedure is selected because there is only a single server in this environment.
In an environment with clustered application servers, the installAppWithClusterOption procedure can be used instead.

The script file is written in Python and is called applInstall.py (see Example 8-28).

Example 8-28 The applInstall.py script

```
#Check the list and install/update the applications.
#



import sys
import java
import os
import re
import time
from java.lang import *

dir = "C:/WebSphereV7/AppServer/profiles/dmgr40/downloadedContent/inputfile"
#
sep = System.getProperty("line.separator")
for fname in os.listdir(dir):
    print fname
    path = os.path.join(dir, fname)
    if (os.path.isfile(path)) and (not re.match(".*old$",path)) and (not
re.match("appInstall.py",fname)):
        print "procesing %s" % (path)
        inp = open(path, 'r')
        for line in inp.readlines():

            itemList = line[:-1].split(',')
            appName = itemList[0]
            earFile = itemList[1]
            nodeName= itemList[2]
            serverName = itemList[3]

            # application existence check
            print "application existence check"
            existAppList = AdminApp.list().split(sep)
            isExist = 0
            for existApp in existAppList:
                if(appName == existApp):
                    isExist = 1
```

```

        break
# acquire application manager
print "acquire application manager"

    appMgrID =
AdminControl.queryNames("type=ApplicationManager,node="+nodeName+",process="+serverNa
me+",*")

    # if exists, uninstall application
    print "app exists - uninstall"
    if( isExist == 1 ):
        appId = ""
        try:
            _excp_ = 0
            appID =
AdminControl.completeObjectName("type=Application,node="+nodeName+",Server="+serverNa
me+,name="+appName+",*")
            except:
                _type_, _value_, _tbck_ = sys.exc_info()
                _excp_ = 1
            #endTry
            # if running, stop application
            print "appID is %s" % (appID)
            if(len(appID) > 0):
                print "stopping %s" % (appName)
                stopped = AdminControl.invoke(appMgrID, "stopApplication", appName)
                sleep(1)
            # uninstall application
            print "Uninstalling %s" % (appName)
            AdminApplication.uninstallApplication(appName)

            # install application
            print "Installing %s" % (appName)
            AdminApplication.installAppWithNodeAndServerOptions(appName, earFile,
nodeName, serverName)
            print "Starting %s" % (appName)
            started = AdminControl.invoke(appMgrID, "startApplication", appName)
            sleep(1)

            inp.close()
            os.rename(fname, fname + ".old")
        #endif
    #endFor

```

The input file, hello.txt, is shown in Example 8-29.

Example 8-29 hello.txt

```
hello,/websphrev7/appserver/profiles/dmgr40/downloadedContent/app1/hello1.ear,node40a,server40a1
```

The sample script is first tested using wsadmin running in script mode. The result of the sample script is shown in Example 8-30.

Example 8-30 testing the sample script

```
C:\WebSphereV7\AppServer\profiles\dmgr40\bin>wsadmin -lang jython -f c:\websph  
ev7\appserver\profiles\dmgr40\downloadedContent\appinstall.py -username admin -p  
assword admin
```

```
WASX7209I: Connected to process "dmgr" on node dmgr40node using SOAP connector;  
The type of process is: DeploymentManager
```

```
hello.txt
```

```
procesing C:\WebSphereV7\AppServer\profiles\dmgr40\downloadedContent\inputfile\h  
ello.txt
```

```
application existence check
```

```
acquire application manager
```

```
app exists - uninstall
```

```
Installing hello
```

```
-----  
AdminApplication:      Install application with -node and -server options
```

```
Application name:      hello
```

```
Ear file to deploy:    /websphrev7/appserver/profiles/dmgr40/downloadedConten  
t/app1/hello1.ear
```

```
Node name:             node40a
```

```
Server name:           server40a1
```

```
Usage: AdminApplication.installAppWithNodeAndServerOptions("hello", "/webspher  
e7/appserver/profiles/dmgr40/downloadedContent/app1/hello1.ear", "node40a", "ser  
ver40a1")
```

```
ADMA5016I: Installation of hello started.
```

```
ADMA5058I: Application and module versions are validated with versions of deployment targets.
```

```
ADMA5005I: The application hello is configured in the WebSphere Application Server repository.
```

```
ADMA5053I: The library references for the installed optional package are created
```

```
.
```

```
ADMA5005I: The application hello is configured in the WebSphere Application Server repository.
```

ADMA5001I: The application binaries are saved in C:\WebSphereV7\AppServer\profiles\dmgr40\wstemp\Script120f8e64a06\workspace\cells\Cell40\applications\hello.ear\hello.ear
ADMA5005I: The application hello is configured in the WebSphere Application Server repository.
SECJ0400I: Successfully updated the application hello with the appContextIDForSecurity information.
ADMA5005I: The application hello is configured in the WebSphere Application Server repository.
CWSAD0040I: The application hello is configured in the Application Server repository.
ADMA5113I: Activation plan created successfully.
ADMA5011I: The cleanup of the temp directory for application hello is complete.
ADMA5013I: Application hello installed successfully.
OK: installAppWithNodeAndServerOptions('hello', '/websphrev7/appserver/profiles/dmgr40/downloadedContent/app1/hello1.ear', 'node40a1', 'server40a1', 'false'):

8.7.3 Submitting the job

This section describes how to use the job manager to execute the script:

1. Before running the `applInstall.py` script, you must transfer it, along with the `hello.txt` file and the `hello1.ear` file from the job manager to the deployment manager using the `distributeFile` job (see 5.3.2, “Distributing files using the job manager” on page 283 to define the directories required for this task).

In Rational Application Developer, export the `applInstall.py` script file and application EAR file to the `jmgr_profile_root/config/temp/JobManager` directory.

Manually copy the `hello.txt` file to the same directory.

2. In the Job manager console, select the **Job** → **Submit** menu. This will launch the Job properties wizard.

Use the **Distribute file** job to transfer each file. In each case, select the admin agent as the target node.

The source location refers to the file in the `jmgr_profile_root/config/temp/JobManager` directory.

The format is:

`file:/file_name`

The `hello.txt` file should be distributed to the following directory:

`dmgr_profile_root/downloadedContent/inputfile`

The appInstall.py file should be distributed to the following directory:

dmgr_profile_root/downloadedContent

The appInstall.py file should be distributed to the following directory:

dmgr_profile_root/downloadedContent/app1

3. The next step is to submit the wsadmin script for execution. Select the **Job → Submit** menu. This will launch the Job properties wizard.

- a. Select **Run wsadmin script** as the job type and enter a description.
Click **Next** (Figure 8-8).

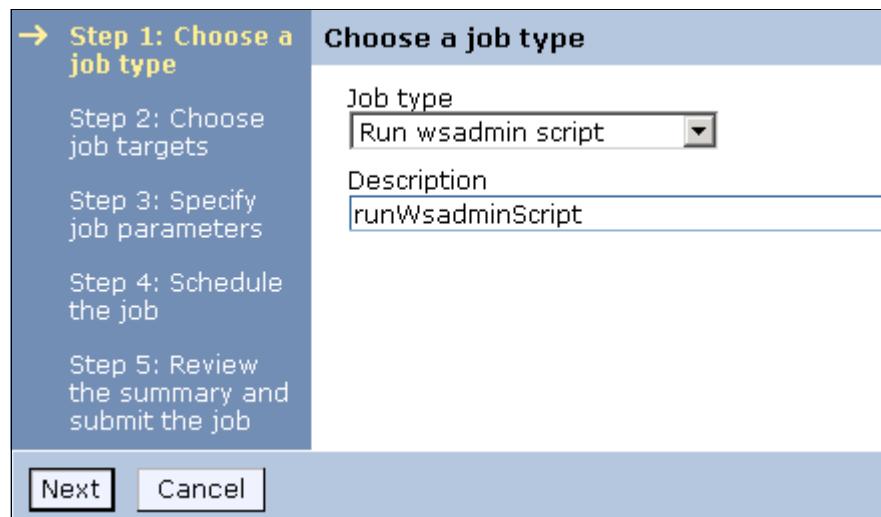


Figure 8-8 Step1: Choose the job type

- b. Select the job target. In this case, the deployment manager node is selected. Enter the user ID and password required for administration tasks on the deployment manager. Click **Next**.
- c. Specify the script location. This is the same location you used when you distributed the file. The current directory is *dmgr_profile_root*/downloadedContent directory. Then, click **Next** (Figure 8-9).

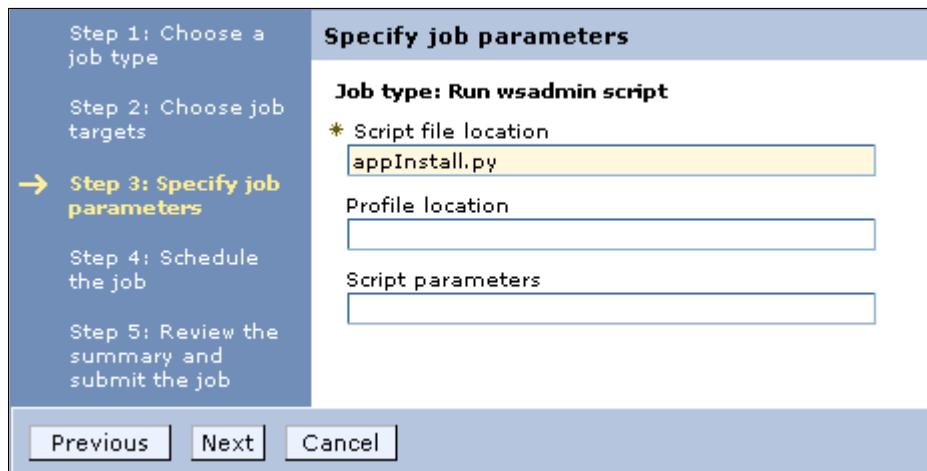


Figure 8-9 Step3: Specify job parameters

- d. The next step allows you to schedule the job to run once, or at regular intervals. It also allows you to define when the job is available for execution and when it expires.

In this example, the execution of the script from the job manager will be tested first using the Run once option in the Availability interval field. After the job has been tested, the job submit process can be repeated and an interval can be set so the job runs automatically.

Click **Next**.

- e. Review the summary and click **Finish**.
4. Monitor the job status to ensure it completes successfully.
5. If the job does not complete successfully, click the Job ID to see the messages produced.

8.7.4 Check the results

The results can be verified by displaying the new application in the administrative console, starting it, then accessing it from a Web browser:

1. Access the deployment manager console and expand **Applications** from the navigation tree and click **Application Types** → **WebSphere enterprise** application.
2. Verify that the new application is in the list.

8.8 Online resources

These websites and URLs are also relevant as further information sources:

- ▶ Command assistance simplifies administrative scripting in WebSphere Application Server:
http://www.ibm.com/developerworks/websphere/library/techarticles/0812_rhodes/0812_rhodes.html
- ▶ Sample Scripts for WebSphere Application Server Versions 5 and 6:
<http://www.ibm.com/developerworks/websphere/library/samples/SampleScripts.html>



Accessing databases from WebSphere

When an application or WebSphere component requires access to a database, that database must be defined to WebSphere as a data source. Two basic definitions are required:

- ▶ A JDBC provider definition defines an existing database provider, including the type of database access that it provides and the location of the database vendor code that provides the implementation.
- ▶ A data source definition defines which JDBC provider to use, the name and location of the database, and other connection properties.

In this chapter, we discuss the various considerations for accessing databases from WebSphere.

We cover the following topics:

- ▶ “JDBC resources” on page 488
- ▶ “Steps in defining access to a database” on page 493
- ▶ “Example: Connecting to an IBM DB2 database” on page 495
- ▶ “Example: Connecting to an Oracle database” on page 503
- ▶ “Example: Connecting to an SQL Server database” on page 510
- ▶ “Example: Connecting to an Informix Dynamic Server database” on page 516
- ▶ “Configuring connection pooling properties” on page 523

9.1 JDBC resources

The JDBC API provides a programming interface for data access of relational databases from the Java programming language. WebSphere Application Server V7 supports the following JDBC APIs:

- ▶ JDBC 4.0 (**New in V7**)
- ▶ JDBC 3.0
- ▶ JDBC 2.1 and Optional Package API (2.0)

In the following sections, we explain how to create and configure data source objects for use by JDBC applications. This method is the recommended method to connect to a database and the only method if you intend to use connection pooling and distributed transactions.

Note: DB2 for z/OS local JDBC Provider (RRS) Version 6.1 is not supported in WebSphere Application Server V7.0. If you use this provider, you need to migrate to IBM JCC Driver or DB2 Universal JDBC Driver.

The following database platforms are supported for JDBC:

- ▶ DB2
- ▶ Oracle
- ▶ Sybase
- ▶ Informix®
- ▶ SQL Server
- ▶ IBM Cloudscape and IBM Derby (test and development only)
- ▶ Third-party vendor JDBC data source using SQL99 standards

9.1.1 JDBC providers and data sources

A *data source* represents a real-world data source, such as a relational database. When a data source object is registered with a JNDI naming service, an application can retrieve it from the naming service and use it to make a connection to the data source that it represents.

Information about the data source and how to locate it, such as its name, the server on which it resides, its port number, and so on, is stored in the form of *properties* on the *DataSource* object. Storing this information in this manner makes an application more portable because it does not need to hard code a driver name, which often includes the name of a particular vendor. It also makes maintaining the code easier because if, for example, the data source is moved to a different server, all that needs to be done is to update the relevant property in the data source. None of the code using that data source needs to be touched.

After a data source is registered with an application server's JNDI name space, application programmers can use it to make a connection to the data source that it represents.

The connection usually is a *pooled connection*. In other words, when the application closes the connection, the connection is returned to a connection pool, rather than being destroyed.

Data source *classes* and JDBC *drivers* are implemented by the data source vendor. By configuring a JDBC provider, you provide information about the set of classes that are used to implement the data source and the database driver. Also, you provide the environment settings for the DataSource object. A driver can be written purely in the Java programming language or in a mixture of the Java programming language and the Java Native Interface (JNI) native methods.

In the next sections, we describe how to create and configure data source objects, as well as how to configure the connection pools used to serve connections from the data source.

9.1.2 WebSphere support for data sources

The programming model for accessing a data source is as follows:

1. An application retrieves a DataSource object from the JNDI naming space.
2. After the DataSource object is obtained, the application code calls the `getConnection()` request on the data source to get a Connection object. The connection is obtained from a pool of connections.
3. After the connection is acquired, the application sends SQL queries or updates to the database.

In addition to the data source support for Java EE 5, J2EE 1.3, and J2EE 1.4 applications, support is also provided for J2EE 1.2 data sources. The two types of support differ in how connections are handled. However, from an application point of view, they look the same.

Data source support

The primary data source support is intended for J2EE 1.3 and J2EE 1.4, and Java EE 5 applications. Connection pooling is provided by two components, a JCA Connection Manager, and a relational resource adapter. See Figure 9-1.

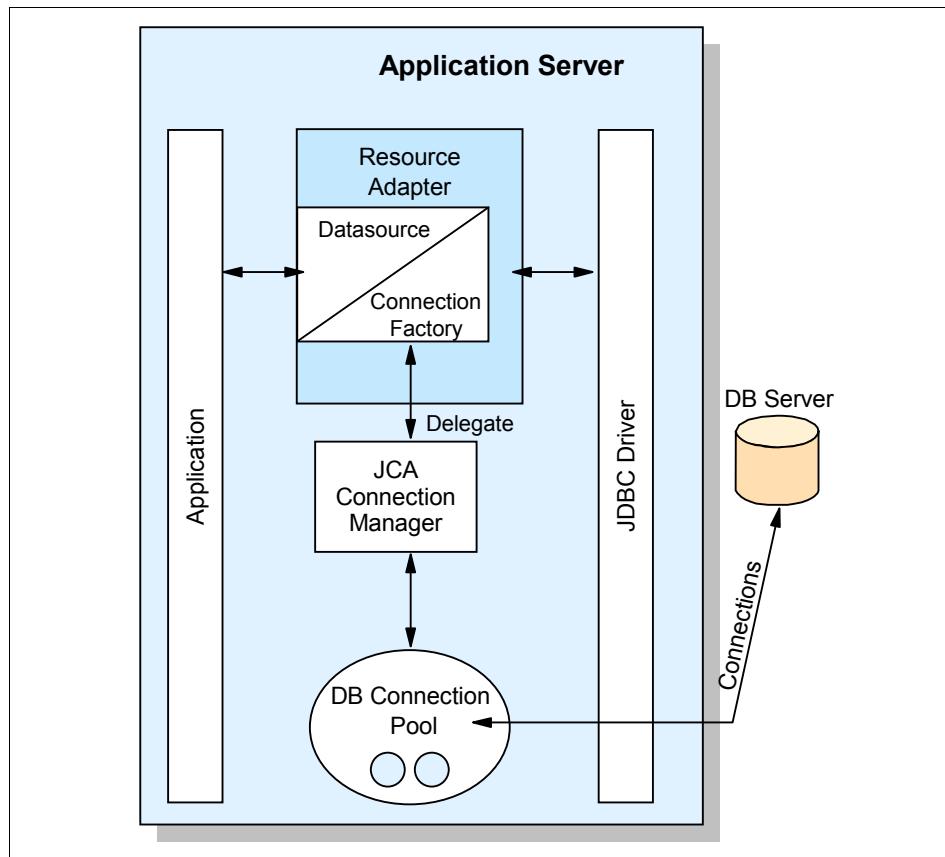


Figure 9-1 Resource adapter in J2EE connector architecture

The JCA Connection Manager provides connection pooling, local transactions, and security support.

The relational resource adapter provides JDBC wrappers and the JCA common client interface (CCI) implementation that allows BMP, JDBC applications, and CMP beans to access the database.

Figure 9-2 shows the relational resource adapter model.

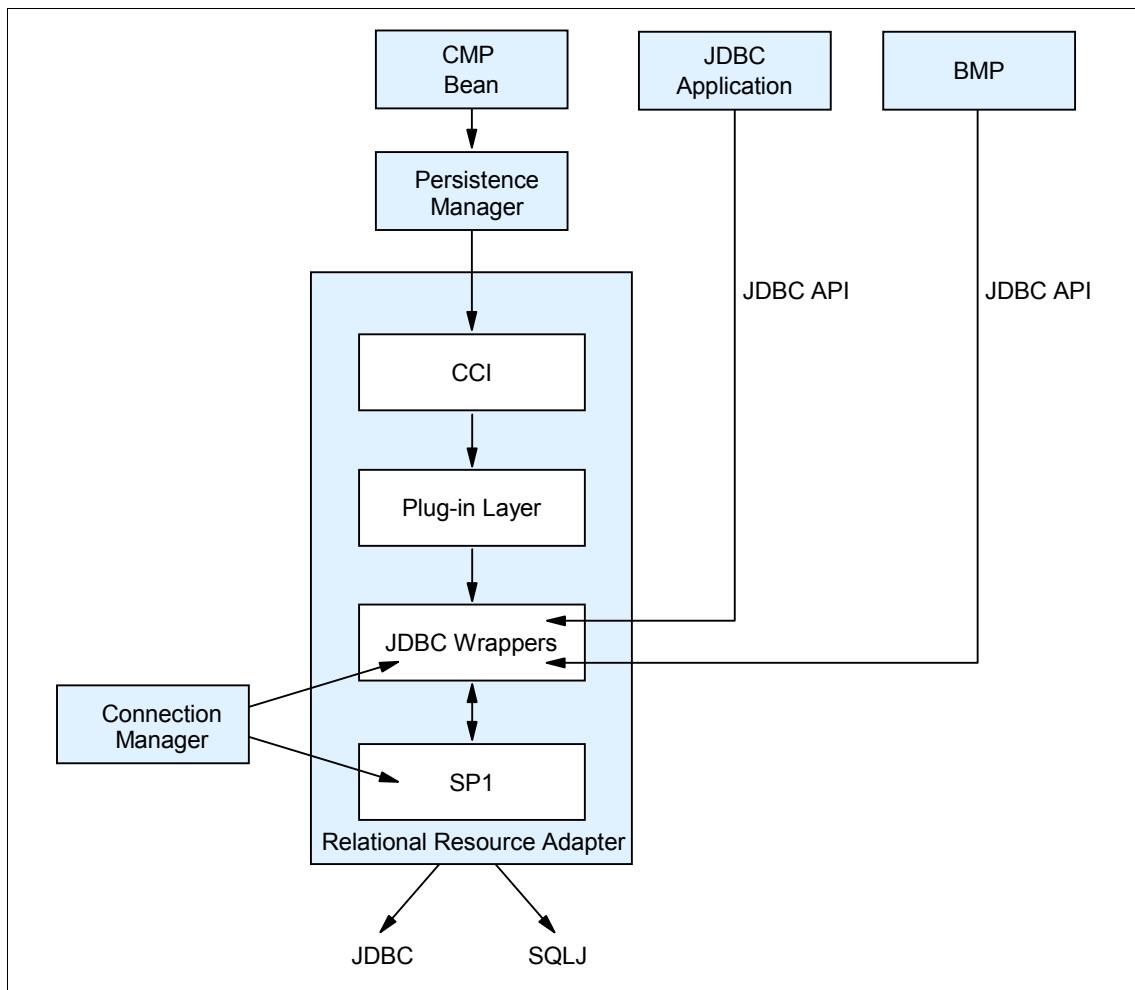


Figure 9-2 Persistence resource adapter model

WebSphere Application Server has a Persistence Resource Adapter that provides relational persistence services to EJB beans as well as providing database access to BMP and JDBC applications. The Persistence Resource Adapter has two components:

- ▶ The Persistence Manager, which supports the EJB CMP persistence model
- ▶ The Relational Resource Adapter

The Persistence Resource Adapter code is included in the following Java packages:

- ▶ The `com.ibm.ws.rsadapter.cci` package contains the CCI implementation and JDBC wrappers.
- ▶ The `com.ibm.ws.rsadapter.spi` package contains the service provider interface (SPI) implementation.
- ▶ The `com.ibm.ws.rsadapter.jdbc` package contains all the JDBC wrappers.
- ▶ The `com.ibm.websphere.rsadapter` package contains `DataStoreHelper`, `WSCallerHelper`, and `DataAccessFunctionSet`.

The Relational Resource Adapter is the Persistence Manager's vehicle to handle data access to and from the back-end store, providing relational persistence services to EJB beans. The implementation is based on the J2EE Connector Architecture (JCA) specification and implements the JCA CCI and SPI interfaces.

When an application uses a data source, the data source uses the JCA connector architecture to get to the relational database.

For an EJB, the sequence is as follows:

1. An EJB performs a JNDI lookup of a data source connection factory and issues a `getConnection()` request.
2. The connection factory delegates the request to a connection manager.
3. The connection manager looks for an instance of a connection pool in the application server. If no connection pool is available, then the manager uses the `ManagedConnectionFactory` to create a physical, or nonpooled, connection.

Version 4 data source

WebSphere Application Server V4 provided its own JDBC connection manager to handle connection pooling and JDBC access. This support is included with WebSphere Application Server V7.0 to provide support for J2EE 1.2 applications. If an application chooses to use a Version 4 data source, the application has the same connection behavior as in Version 4 of the application server.

Use the Version 4 data source for the following purposes:

- ▶ J2EE 1.2 applications
 - All EJB beans, JDBC applications, or Version 2.2 servlets must use the Version 4 data source.
- ▶ EJB 1.x modules with 1.1 deployment descriptor
 - All of these modules must use the Version 4 data source.

9.2 Steps in defining access to a database

The following steps are involved in creating a data source:

1. Verify that connection to the database server is supported by WebSphere Application Server. See:
<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27012369>
2. Ensure that the database has been created and can be accessed by the systems that will use it.
3. Ensure that the JDBC provider classes are available on the systems that will access the database. If you are not sure which classes are required, consult the documentation for the provider.
4. Create an authentication alias that contains the user ID and password that will be used to access the database.
5. Create a JDBC provider.

The JDBC provider gives the classpath of the data source implementation class and the supporting classes for database connectivity. This is vendor-specific.

The information center provides information about JDBC driver support and requirements. To determine if your provider is supported, refer to the JDBC Provider Summary article at:

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

New in V7 for DB2: The DB2 Using IBM JCC Driver is a one-phase commit JCC provider for DB2 that uses the IBM Data Server Driver for JDBC and SQLJ. The DB2 Using IBM JCC Driver is the next generation of the DB2 Universal JCC driver. Data sources that you create with this provider support only one-phase commit processing, unless you use the type 2 JDBC driver with the application server for z/OS. If you run the application server on z/OS with the type 2 driver, the driver uses RRS and supports two-phase commit processing. This driver provides some JDBC 4.0 capabilities.

6. Create a data source.

The JDBC data source encapsulates the database-specific connection settings. You can create many data sources that use the same JDBC provider.

7. Save the changes to the master repository and synchronize with the nodes involved.
8. Test the connection to the data source.
9. Review and adjust the connection pool settings (this should be done on a periodic basis).

9.2.1 Creating an authentication alias

The examples in this chapter assume that the database is password protected and that the user ID and password will be defined at run time.

To create a J2C authentication alias that contains the user ID and password that is required to access the database, follow these steps:

1. Select **Security** → **Global security**.
2. In the Authentication area, expand Java Authentication and Authorization Server, and click **J2C authentication data**.
3. Click **New**.
4. Enter an alias name, user ID, and password, as shown in Figure 9-3. The alias name will be used later when you create a resource to identify this as the authentication alias to use. The user ID and password must be valid for the database system and have authority to the database.

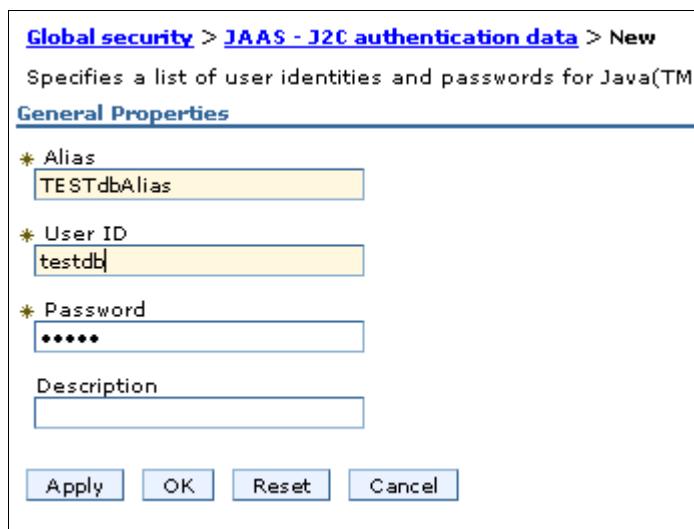


Figure 9-3 Define an authentication alias

5. Click **OK**.

9.3 Example: Connecting to an IBM DB2 database

In this section, we illustrate how to configure a JDBC provider using a DB2 provider as an example.

9.3.1 Creating the JDBC provider

To create a JDBC provider, complete the following steps from the administrative console:

1. Ensure that the implementation classes for the provider are available to the system. The class files will need to be located on each system where the application servers will run.
2. In the administrative console, expand **Resources** → **JDBC** from the navigation tree.
3. Click **JDBC Providers**.
4. Select the scope. (Although you can select **All scopes** to view all resources, you must select a specific scope to create a resource.)

Note: JDBC resources are created at a specific scope level. The data source scope level is inherited from the JDBC provider. For example, if we create a JDBC provider at the node level and then create a data source using that JDBC provider, the data source inherits:

- ▶ The JDBC provider settings, such as classpath, implementation class, and so on
- ▶ The JDBC provider scope level

In this example, if the scope were set to node-level, all application servers running on that node register the data source in their name space.

The administrative console now shows all the JDBC providers that are created at that scope level.

5. Select **New** to start the wizard and to create a new JDBC provider.

6. In step 1 of the wizard, define the type of provider you will use. See Figure 9-4.



Figure 9-4 Define a new JDBC provider: Window 1

Specify the following information

– Database type

Select the vendor-specific database type. If the database type you need is not in the list, select **User-defined**, and consult the vendor documentation for the specific properties that are required.

– Provider type

Select from a predefined list of supported provider types, based on the database type that you select.

- Implementation type

Select from the implementation types for the provider type that you selected.

- Name

Specify a Name for this driver.

Click **Next**.

7. The settings page for your JDBC database class path opens. Figure 9-5 shows the configuration page for a Universal JDBC Provider.

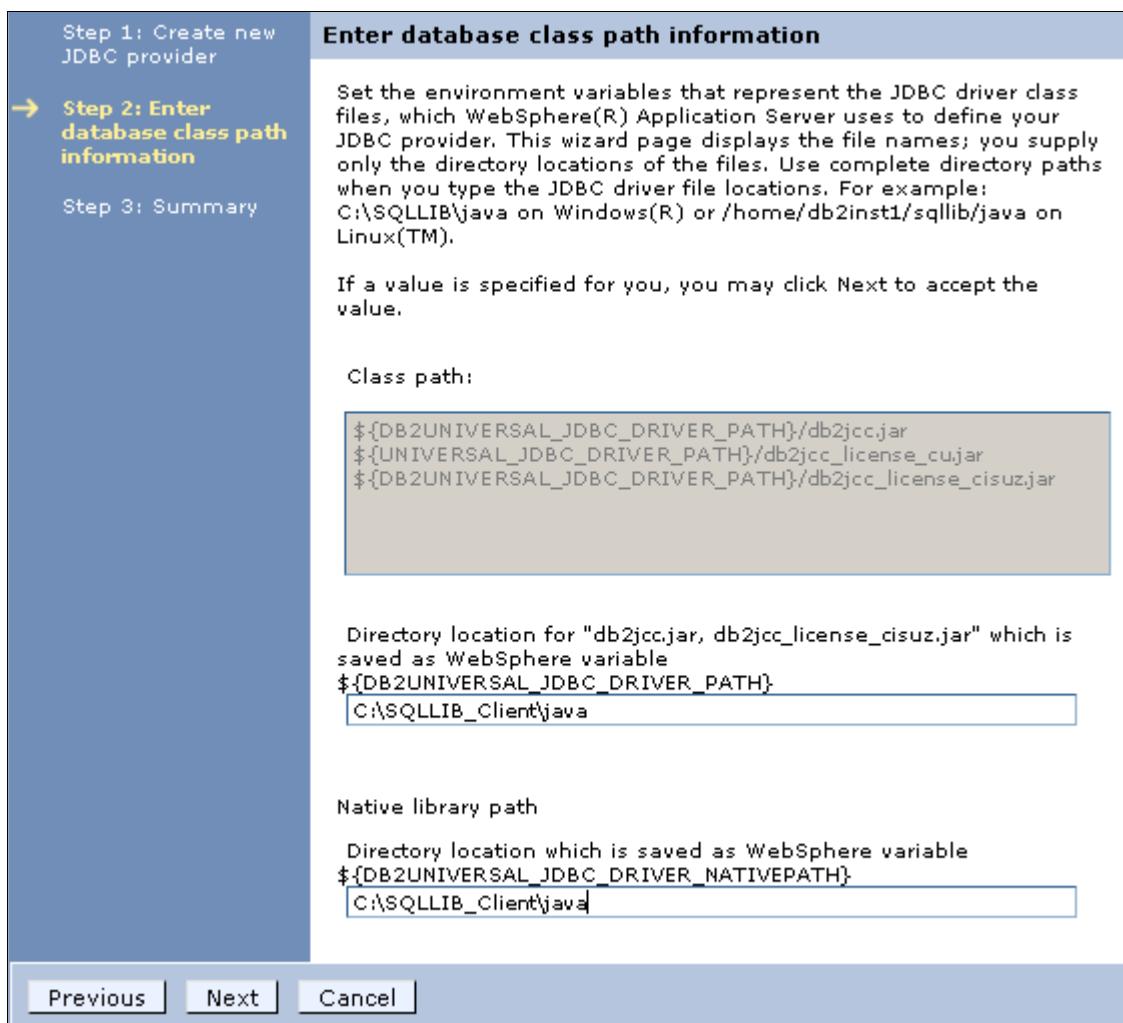


Figure 9-5 Define a new JDBC provider: Window 2

Enter the JDBC provider properties:

- Classpath

This field is a list of paths or JAR file names that together form the location for the resource provider classes. This field is pre-set using variable names that are specific to each type of provider. If you are creating a user-defined provider, specify the entries by pressing Enter between each entry.

The remaining properties are dependent upon the type of provider. They represent the variables that are used in the classpath and their value. If you enter a value for a variable on this panel, the corresponding variables are populated automatically with these values. Conversely, if the variables are already defined, these fields are populated with the variables.

You can view or modify the variables by selecting **Environment** → **WebSphere Variables** in the navigation menu.

Because this example is for DB2, the following fields are available:

- Library path

This field specifies the values for the global variable `UNIVERSAL_JDBC_DRIVER_PATH`, which indicates the classpath jar's location.

- Native Library Path

This field is an optional path to any native libraries. Entries are required if the JDBC provider chosen uses non-Java, or native, libraries. The global variable for this is `UNIVERSAL_JDBC_DRIVER_NATIVEPATH`.

8. After verifying the settings, click **Finish** to enable the links to create data sources under the Additional Properties section.

Tip: To make a data source available on multiple nodes using different directory structures, complete the following steps using the administrative console:

1. Define the JDBC provider and data source at the cell scope. Use WebSphere environment variables for the classpath and native path.
2. Define the variables at the node scope for each node to specify the driver location for the node.

For example, `${DRIVER_PATH}` can be used for the classpath in the provider definition. You can then define a variable called `${DRIVER_PATH}` at the cell scope to act as a default driver location. Then you can override that variable on any node by defining `${DRIVER_PATH}` at the node scope. The node-level definition takes precedence over the cell-level definition.

9.3.2 Creating the data source

Data sources are associated with a specific JDBC provider and can be viewed or created from the JDBC provider configuration page. You have two options when creating a data source, depending on the J2EE support of the application. Here we discuss creating or modifying data sources for Java EE5, J2EE 1.3, and J2EE 1.4 applications. For information about using data sources with J2EE 1.2 applications, see the topic, *Data sources (Version 4)* in the information center.

The administrative console provides a wizard that helps you create a data source. Keep in mind, however, that although the wizard provides a good way to establish connections quickly, it also establishes default-sized connection pool settings that you need to tune properly before production.

To create a data source, complete the following steps:

1. Expand **Resources** → **JDBC** in the navigation tree, and select **Data sources**.
2. Select the scope. Although you can select **All** to view all resources, you must select a specific scope to create a resource.

The scope determines which applications can use this data source. We recommend that you select the narrowest scope that is required, while also ensuring that the applications that require the resource can access it. For information about selecting a scope, see “Selecting a scope” on page 253.
3. Click **New** to create a new data source and to start a wizard (Figure 9-6).

The screenshot shows a wizard interface for creating a data source. On the left, a sidebar lists steps: Step 1: Enter basic data source information (highlighted in yellow), Step 2: Select JDBC provider, Step 3: Enter database specific properties for the data source, Step 4: Setup security aliases, and Step 5: Summary. The main panel title is "Enter basic data source information". It contains a descriptive text block and a requirement note. Below that are fields for "Scope", "Data source name", and "JNDI name", each with its respective value. At the bottom are "Next" and "Cancel" buttons.

→ Step 1: Enter basic data source information	Enter basic data source information
Step 2: Select JDBC provider	Set the basic configuration values of a datasource for association with your JDBC provider. A datasource supplies the physical connections between application server and the database.
Step 3: Enter database specific properties for the data source	Requirement: Use the Datasources (WebSphere(R) Application Server V6 console pages if your applications are based on the Enterprise JavaBeans (EJB) 1.0 specification or the Java(TM) Servlet 2.2 specification.
Step 4: Setup security aliases	Scope cells:Cell02:nodes:sys1Node01
Step 5: Summary	* Data source name TDSDB2
	* JNDI name jdbc/testdb
Next	Cancel

Figure 9-6 Data source general properties

Specify the following information:

- Data source name

This field is a name by which to administer the data source. Use a name that is suggestive of the database name or function.

- JNDI name

This field refers to the data source's name as registered in the application server's name space.

When installing an application that contains modules with JDBC resource references, the resources need to be bound to the JNDI name of the resources; for example, `jdbc/<database_name>`.

Click **Next**.

4. Now you need to specify database specific properties, as shown on the right of Figure 9-7. Click **Next**.

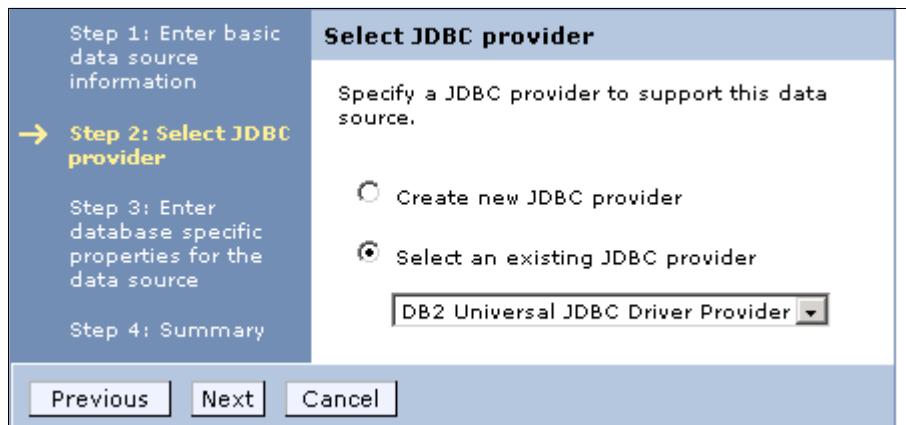


Figure 9-7 Select a JDBC provider

This window allows you to select a JDBC provider or to create a new one. If you create a new JDBC provider, you will be routed through the windows seen earlier in 9.3.1, “Creating the JDBC provider” on page 495. If you select an existing JDBC provider, continue with the next step here.

In this case, we select an existing JDBC provider and click **Next**.

The entries shown in Figure 9-8 are specific to the JDBC driver and data source type, which show the properties for the Universal data source.

Step 1: Enter basic data source information

Step 2: Select JDBC provider

→ **Step 3: Enter database specific properties for the data source**

Step 4: Setup security aliases

Step 5: Summary

Enter database specific properties for the data source

Set these database-specific properties, which are required by the database vendor JDBC driver to support the connections that are managed through the datasource.

Name	Value
* Driver type	4
* Database name	TESTDB
* Server name	T60
* Port number	50000

Use this data source in container managed persistence (CMP)

Previous | Next | Cancel

Figure 9-8 Database-specific properties

Specify the following information:

- Driver type
 - The type of JDBC Driver (2 or 4) used to access the database. To determine the best type of driver to use for your circumstances, consult the documentation for the specific driver that you use.
 - In general, however, use type 2 for databases on the same system as the application server and type 4 for remote databases.
- Database Name
 - The name of the database (or the cataloged alias).
- Server name and port
 - The database server name and its listening port (the default for DB2 is 50000).
- Container managed persistence (CMP)
 - This field specifies if the data source is to be used for container managed persistence of EJB beans.

Deep-dive: Selecting the “Use this data source in container managed persistence (CMP)” option causes a CMP connection factory that corresponds to this data source to be created for the relational resource adapter. The name of the connector factory that is created is <datasourcename>_CF and the connector factory is registered in JNDI under the entry eis/<jndi_name>_CMP.

To view the properties of the just created connection factory, select **Resources** → **Resource Adapters** → **Resource Adapters**. Enable the **Show built-in resources** check box in the preferences. Select **WebSphere Relational Resource Adapter** → **CMP Connection Factories**. Be sure to set the scope so that it is the same scope as that for the data source.

Click **Next**.

5. The next step allows you to select or define a J2C authentication alias for the database. The authentication alias simply contains the user ID and password required to access the database (Figure 9-9).

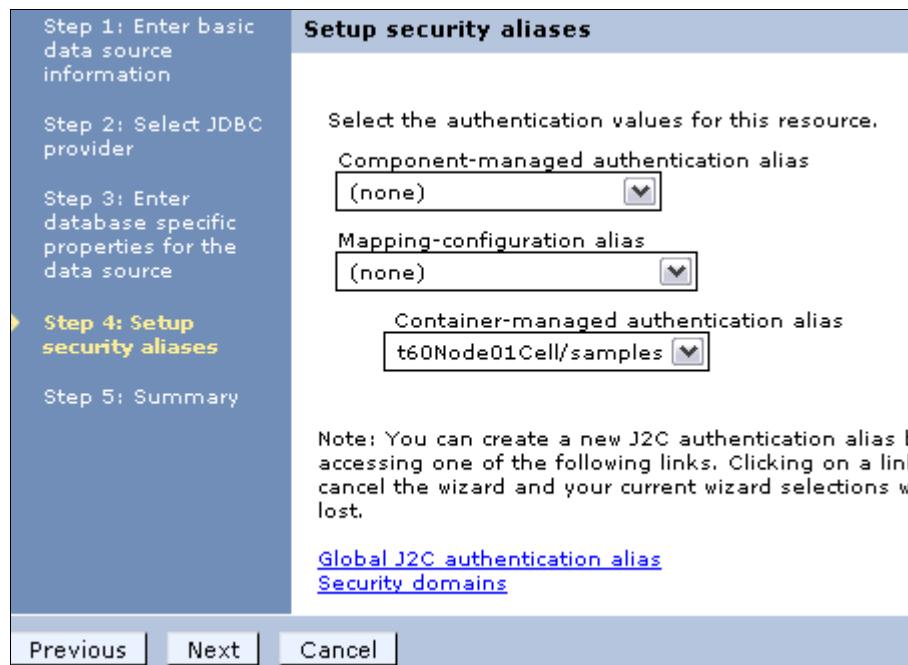


Figure 9-9 Specify the authentication alias

Click **Next**.

- 6. A summary of the options that you chose displays. Click **Next** to create the data source.

- The new data source is listed in the table of resources. You can test the new connection by checking the box to the left of the data source and clicking **Test Connection**. You can view or modify settings for the new data source by clicking the name in the resources list.

9.4 Example: Connecting to an Oracle database

This example illustrates a connection to an Oracle Express 10g database.

- Ensure that the implementation classes for the provider are available to the system. The class files need to be located on each system where the application servers will run.

9.4.1 Creating the JDBC provider

Follow these steps to create the JDBC provider:

1. In the administrative console, expand **Resources** → **JDBC** from the navigation tree.
2. Click **JDBC Providers**.
3. Select the scope. (Although you can select All scopes to view all resources, you must select a specific scope to create a resource.)
4. Select **New** to start the wizard and to create a new JDBC provider.

5. In step 1 of the wizard, define the type of provider that you will use. See Figure 9-10.

Create new JDBC provider

Set the basic configuration values of a JDBC provider, which encapsulates the specific vendor JDBC driver implementation classes that are required to access the database. The wizard fills in the name and the description fields, but you can type different values.

Scope
cells:sys2Cell01:clusters:TradeCluster

* **Database type**
Oracle

* **Provider type**
Oracle JDBC Driver

* **Implementation type**
XA data source

* **Name**
Oracle JDBC Driver (XA)

Description
Oracle JDBC Driver (XA)

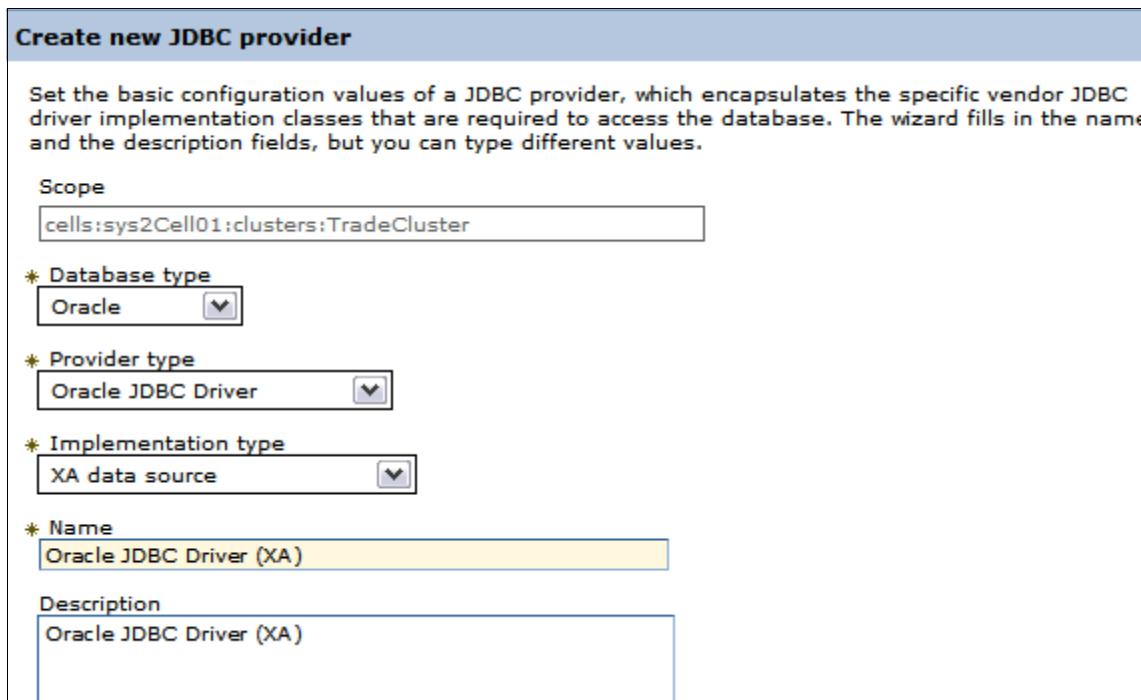


Figure 9-10 Define a new Oracle JDBC provider, Step 1

The database type is Oracle and provider type is Oracle JDBC driver.

Options of implementation type are XA data source or connection pool data source. XA data source types support two-phase commit transactions.

Click **Next**.

6. In the next panel (Figure 9-11), enter the directory location for the Oracle JDBC drivers.

In this example, the ojdbc6.jar is assumed by the wizard. However, the database requires the ojdbc14.jar driver. For now, we complete the wizard to define the driver, and then alter the driver name in the configuration page.

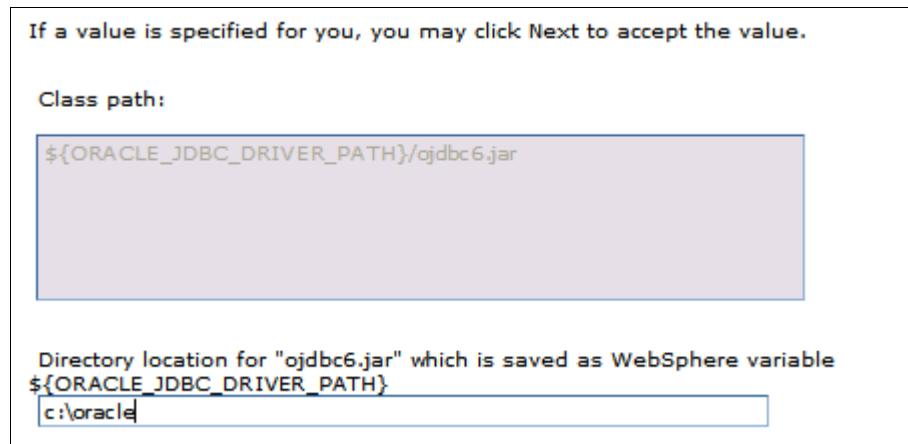


Figure 9-11 Define a new Oracle JDBC provider, Step 2

If you have predefined the ORACLE_JDBC_DRIVER_PATH variable, the driver location is already entered. If you enter a value here, it is saved in the variable.

Click **Next**.

7. Review the summary of the settings, and click **Finish**. The new JDBC provider displays in the list of providers.
8. Click the JDBC provider name to open the configuration page (Figure 9-12). Remember that the wizard assumes that the ojdbc6.jar driver is used.

Change the class path field to point to the ojdbc14.jar.

The implementation class name stays the same.

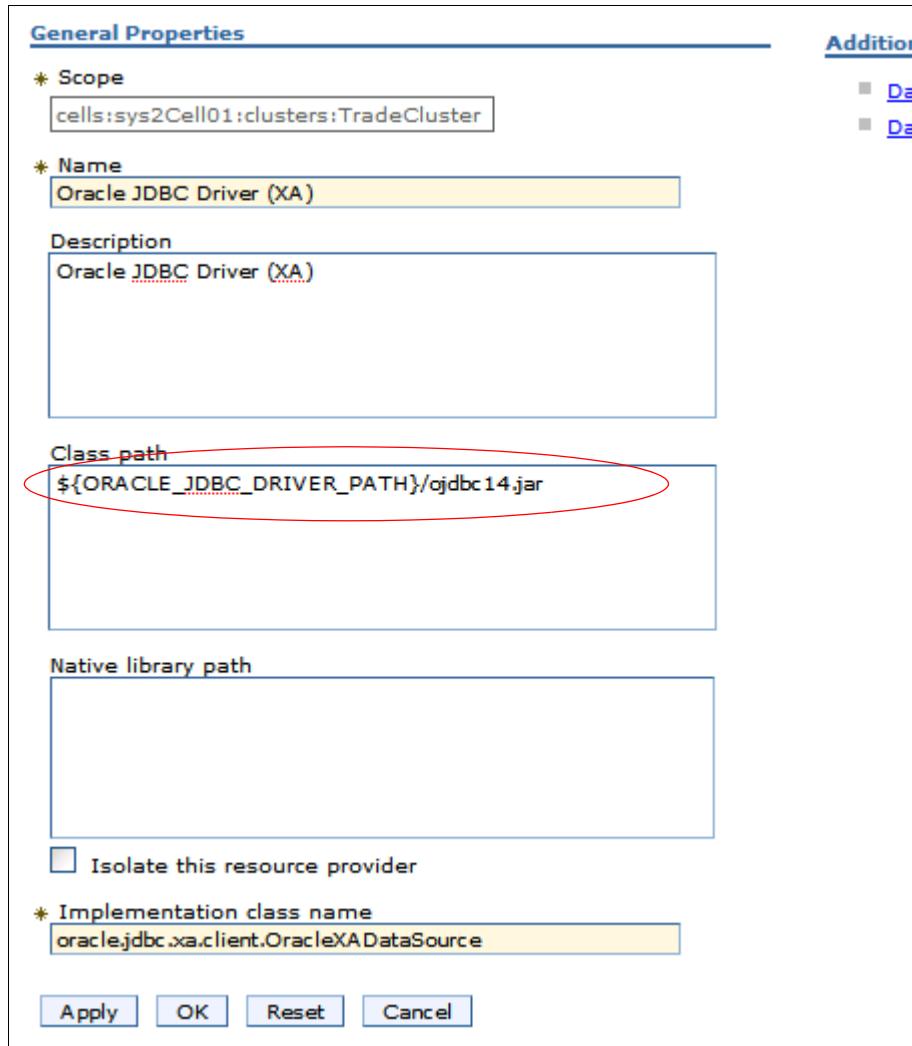


Figure 9-12 Configure the class path

Click **OK**.

9.4.2 Creating the data source

To create a data source, complete the following steps:

1. Expand **Resources** → **JDBC** in the navigation tree, and select **Data sources**.
2. Select the scope. Although you can select **All** to view all resources, you must select a specific scope to create a resource.
3. Click **New** to create a new data source and to start a wizard (Figure 9-13).

The screenshot shows a configuration dialog titled "Enter basic data source information". It contains instructions about setting basic configuration values for a datasource. A note specifies that requirement is to use Datasources (WebSphere(R) Application Server V4) as they are based on the Enterprise JavaBeans(TM) (EJB) 1.0 specification and the Servlet 2.2 specification. The "Scope" field is set to "cells:sys2Cell01:clusters:TradeCluster". The "Data source name" field is filled with "orac-trade-ds" and has an asterisk indicating it is required. The "JNDI name" field is filled with "jdbc/trade-1-ds" and also has an asterisk indicating it is required.

Figure 9-13 Create a data source, Step 1

Enter a name for the new data source. This is used for administrative purposes. Enter the JNDI name that will be used to access the data source and click **Next**.

4. In the next panel (Figure 9-14), select the Oracle JDBC driver and click **Next**.

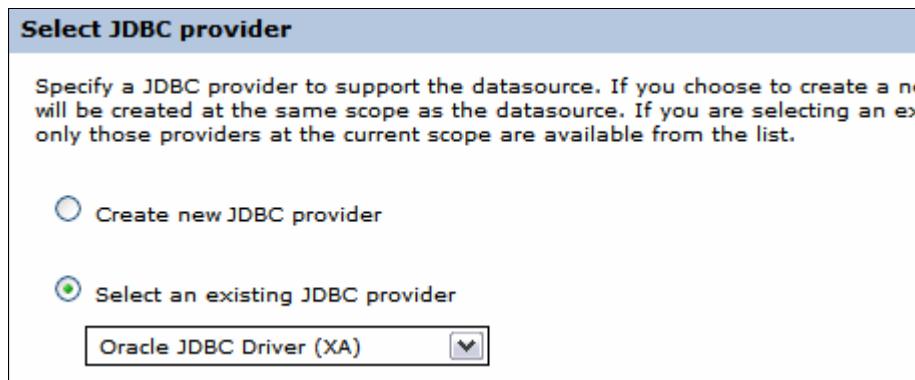


Figure 9-14 Create a data source, Step 2

5. Enter the properties for the database, as shown in Figure 9-15.

The screenshot shows a dialog box titled "Enter database specific properties for the data source". It contains a message: "Set these database-specific properties, which are required by the database vendor to support the connections that are managed through the datasource." Below the message is a table with two rows. The first row has columns "Name" and "Value", with "URL" in the Name column and "jdbc:oracle:thin:@sys2.itso.ral.com" in the Value column. The second row has columns "Name" and "Value", with "Data store helper class name" in the Name column and "Oracle10g data store helper" in the Value column. At the bottom of the dialog is a checked checkbox labeled "Use this data source in container managed persistence (CMP)".

Name	Value
* URL	jdbc:oracle:thin:@sys2.itso.ral.com
* Data store helper class name	Oracle10g data store helper

Figure 9-15 Create a data source, Step 3

Specify the following information:

- The URL for the connection to the XE database is in the following format:
`jdbc:oracle:thin:@host_name:port:service`
In this case:
`jdbc:oracle:thin:@sys2.itso.ral.ibm.com:1521:XE`
- Select the data store helper class name.

Click **Next**.

6. In the next panel (Figure 9-16), select the authentication alias that will provide the user ID and password required to access the database. Click **Next**.

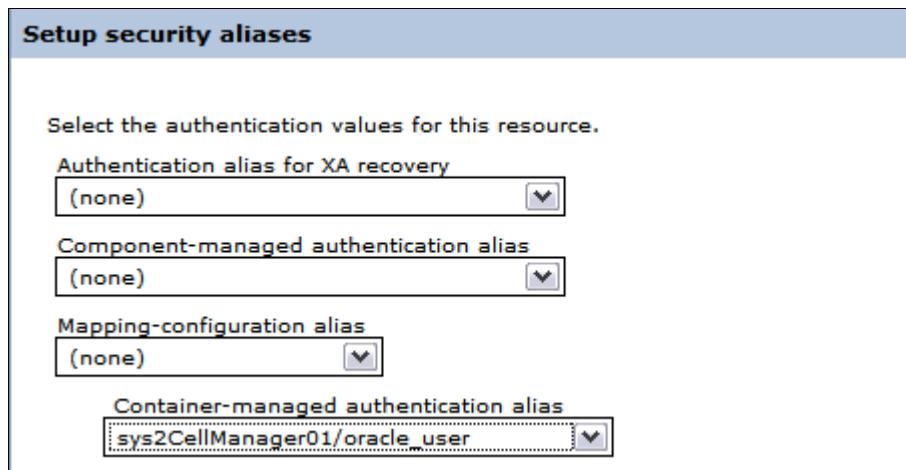


Figure 9-16 Create a data source, Step 4

7. Review the summary of your selections, and click **Finish**.
8. When the data source creation is complete, save the configuration and synchronize the changes with the nodes.
9. Test the new connection by selecting the new data source and clicking **Test connection** (Figure 9-17).



Figure 9-17 Test the connection

Tip: If you receive the following error, make sure that you have adjusted the JDBC provider to use the correct implementation JAR file (step 9):

```
DSRA8040I: Failed to connect to the DataSource. Encountered "":  
java.sql.SQLException: Invalid argument(s) in call  
DSRA0010E: SQL  
State = 99999, Error Code = 17,068
```

9.5 Example: Connecting to an SQL Server database

This example illustrates a connection to a Microsoft SQL Server Express 2005 database.

Ensure that the implementation classes for the provider are available to the system. The class files need to be located on each system where the application servers will run.

Although some JDBC drivers are bundled with WebSphere Application Server to facilitate quick connectivity, in general, JDBC drivers are provided by the database vendor. Information about the location and features of the JDBC provider is provided by the database vendor versus the WebSphere documentation.

9.5.1 Creating the JDBC provider

To create a JDBC provider:

1. In the administrative console, expand **Resources** → **JDBC** from the navigation tree.
2. Click **JDBC Providers**.
3. Select the scope. (Although you can select All scopes to view all resources, you must select a specific scope to create a resource.)
4. Select **New** to start the wizard to create a new JDBC provider.

5. In step 1 of the wizard, define the type of provider that you will use. See Figure 9-18.

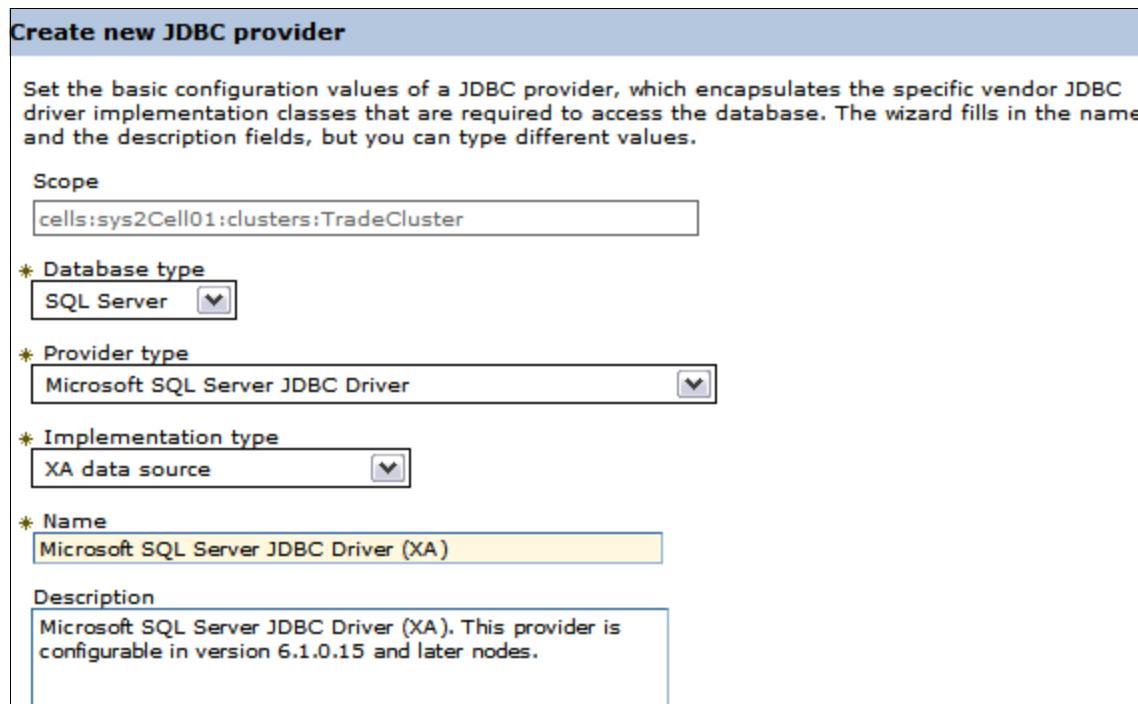


Figure 9-18 Define a new SQL Server JDBC provider, Window1

The database type is SQL Server and provider type is Microsoft SQL Server JDBC driver.

Options of implementation type are XA data source or connection pool data source. XA data source types support two-phase commit transactions.

Click **Next**.

6. In the next panel (Figure 9-19), enter the directory location for the SQL Server JDBC drivers.

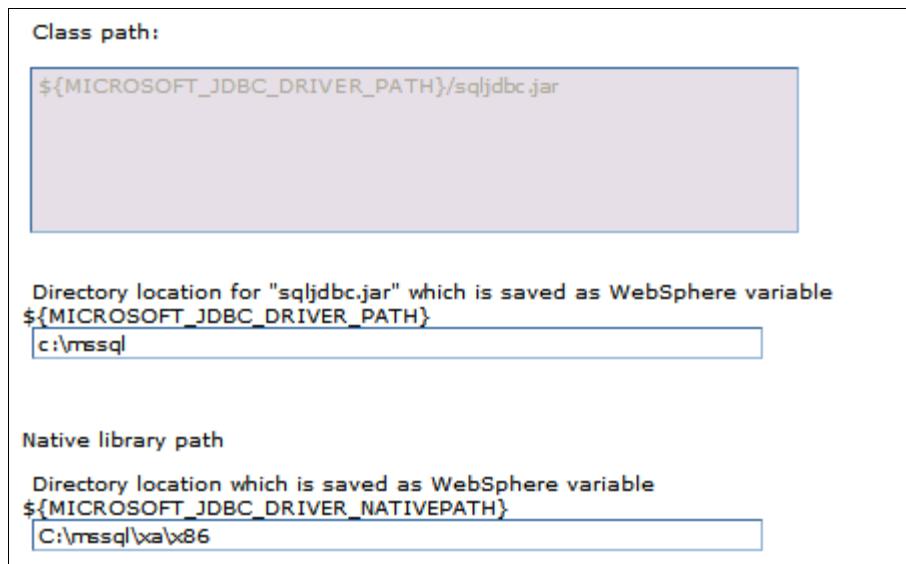


Figure 9-19 Define a new SQL Server JDBC provider, Window 2

If you have predefined the variables used here, the driver locations are entered already. If you enter a value here, it is saved in the appropriate variable.

Click **Next**.

7. Review the summary of the settings, and click **Finish**. The new JDBC provider displays in the list of providers.
8. Click the JDBC provider name to open the configuration page (Figure 9-20). Remember that the wizard assumes that the sqljdbc.jar driver is used in the class path.

Change the class path field to point to the sqljdbc4.jar.

The implementation class name stays the same.

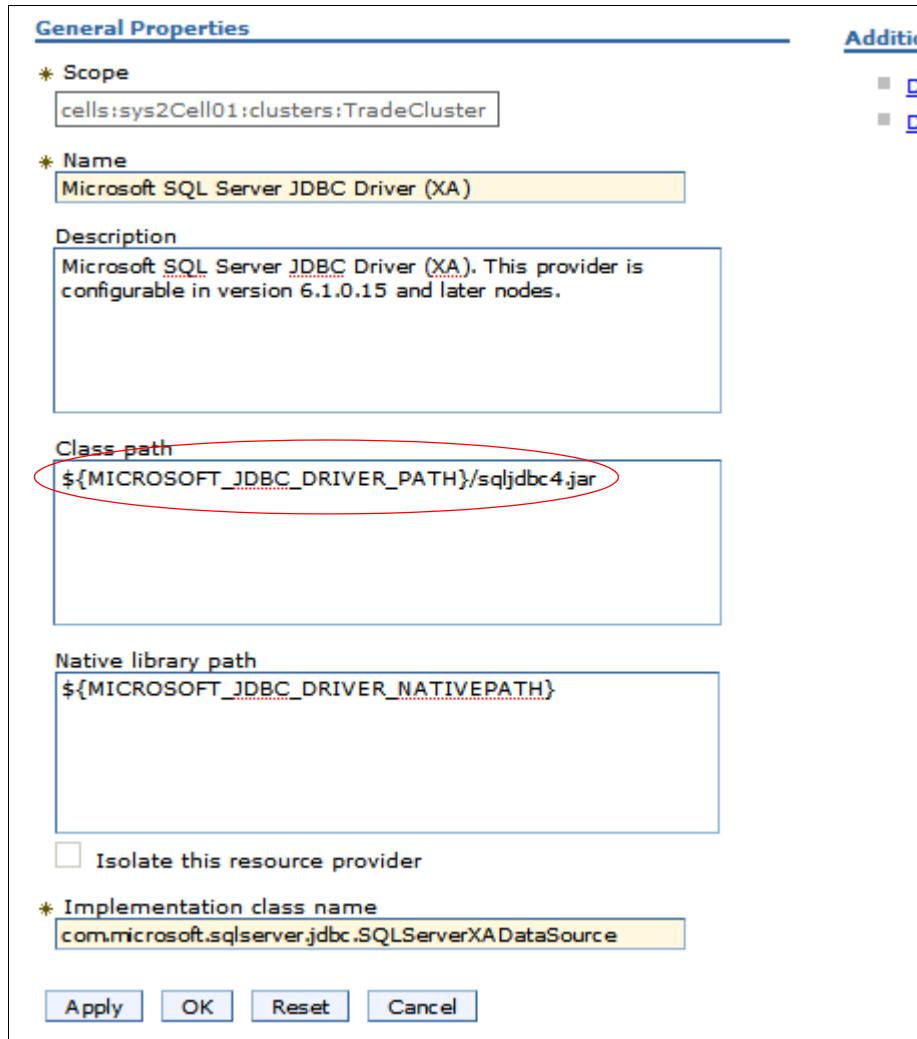


Figure 9-20 Configure the class path

Click **OK**.

9.5.2 Creating the data source

To create a data source, complete the following steps:

1. Expand **Resources** → **JDBC** in the navigation tree and select **Data sources**.
2. Select the scope. Although you can select **All** to view all resources, you must select a specific scope to create a resource.
3. Click **New** to create a new data source and to start a wizard (Figure 9-21).

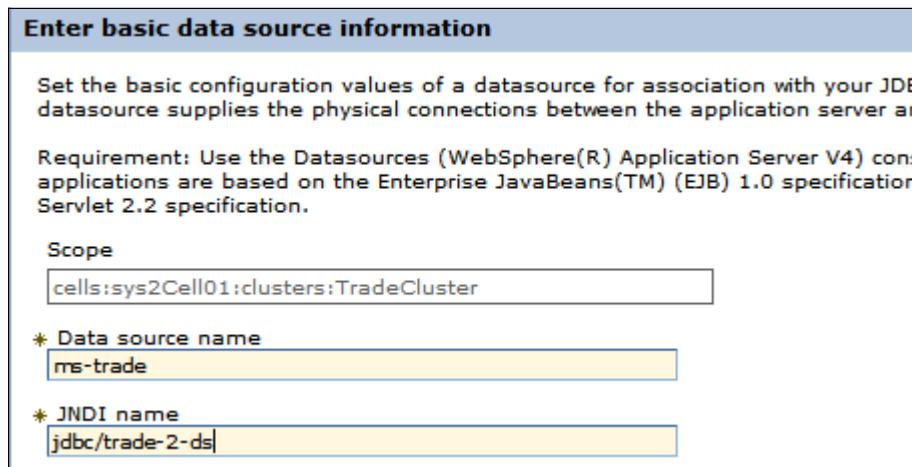


Figure 9-21 Create a data source, Step 1

Enter a name for the new data source. This name is used for administrative purposes. Enter the JNDI name that will be used to access the data source, and click **Next**.

4. In the next panel (Figure 9-22), select the Microsoft SQL Server JDBC driver, and click **Next**.

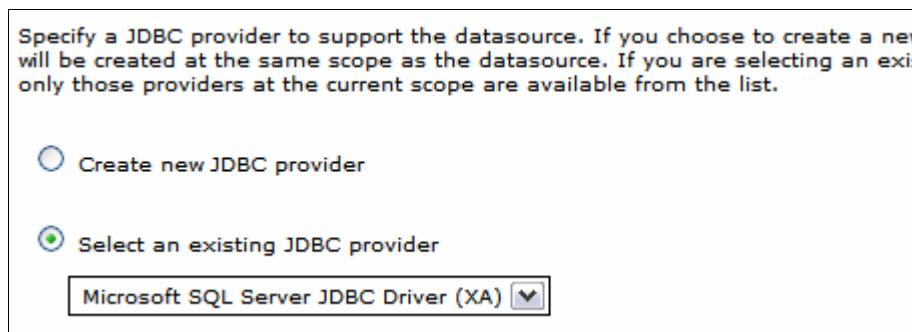


Figure 9-22 Create a data source, Step 2

5. Enter the properties for the database (Figure 9-23).

Enter database specific properties for the data source

Set these database-specific properties, which are required by the database vendor to support the connections that are managed through the datasource.

Name	Value
Database name	TRADE
Port number	1433
Server name	sys2.itso.ral.ibm.com

Use this data source in container managed persistence (CMP)

Figure 9-23 Create a data source, Step 3

Specify the following information:

- Enter the database name.
- Enter the port number the database server listens on.
- Enter the host name of the SQL Server installation.

Click **Next**.

6. In the next panel (Figure 9-24), select the authentication alias that provides the user ID and password that are required to access the database. Click **Next**.

Setup security aliases

Select the authentication values for this resource.

Authentication alias for XA recovery

Component-managed authentication alias

Mapping-configuration alias

Container-managed authentication alias

Figure 9-24 Create a data source, Step 4

7. Review the summary of your selections, and click **Finish**.
8. When the data source creation is complete, save the configuration, and synchronize the changes with the nodes.
9. Test the new connection by selecting the new data source and clicking **Test connection**.

9.6 Example: Connecting to an Informix Dynamic Server database

This example illustrates a connection to an Informix Dynamic Server (IDS) database using the Informix JDBC driver.

Before starting the configuration, ensure that the implementation classes for the provider are available to the system. The class files need to be located on each system where the application servers will run.

Also make sure that an authentication alias is created for the user ID that will be used to connect to the database. For more information, see “Creating an authentication alias” on page 494.

9.6.1 Creating the JDBC provider

Follow these steps to create the JDBC provider:

1. In the administrative console, expand **Resources** → **JDBC** from the navigation tree.
2. Click **JDBC Providers**.
3. Select the scope. (Although you can select **All scopes** to view all resources, you must select a specific scope to create a resource.)
4. Select **New** to start the wizard to create a new JDBC provider.

5. In step 1 of the wizard, define the type of provider that you will use. See Figure 9-25.

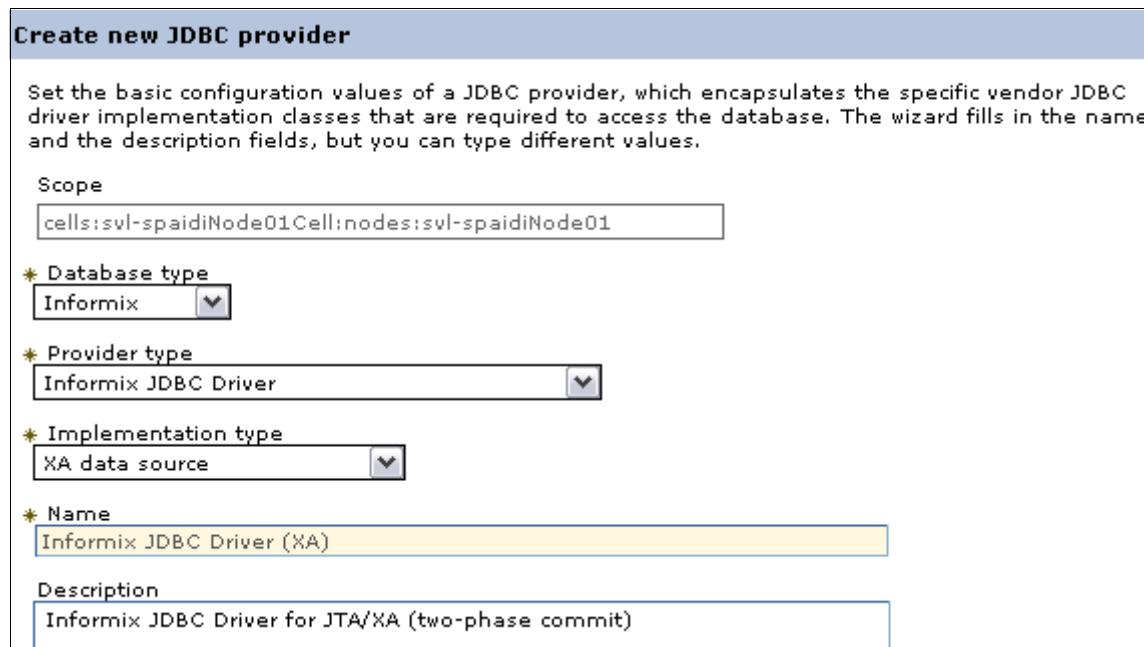


Figure 9-25 Define a new Informix JDBC provider, Step 1

The database type is Informix and provider type is Informix JDBC driver.

Options of implementation type are XA data source or connection pool data source. XA data source types support two-phase commit transactions.

Click **Next**.

6. In the next panel (Figure 9-26), enter the directory location for the Informix JDBC drivers.

In this example, the ifxjdbc.jar and ifxjdbcx.jar are assumed by the wizard.

Class path:
<pre> \${INFORMIX_JDBC_DRIVER_PATH}/ifxjdbc.jar \${INFORMIX_JDBC_DRIVER_PATH}/ifxjdbcx.jar</pre>
Directory location for "ifxjdbc.jar, ifxjdbcx.jar" which is saved as WebSphere variable <pre> \${INFORMIX_JDBC_DRIVER_PATH} C:\Program Files\IBM\Informix_JDBC_Driver\lib</pre>

Figure 9-26 Define a new Informix JDBC provider, Step 2

If you have predefined the INFORMIX_JDBC_DRIVER_PATH variable, the driver location is already entered. If you enter a value here, it is saved in the variable.

Click **Next**.

7. Review the summary of the settings, and click **Finish**. The new JDBC provider displays in the list of providers.
8. Click the JDBC provider name to open the configuration page (Figure 9-27). If you plan to use SQLJ for queries, change the class path field to add the ifxsqlj.jar file.

The implementation class name stays the same.

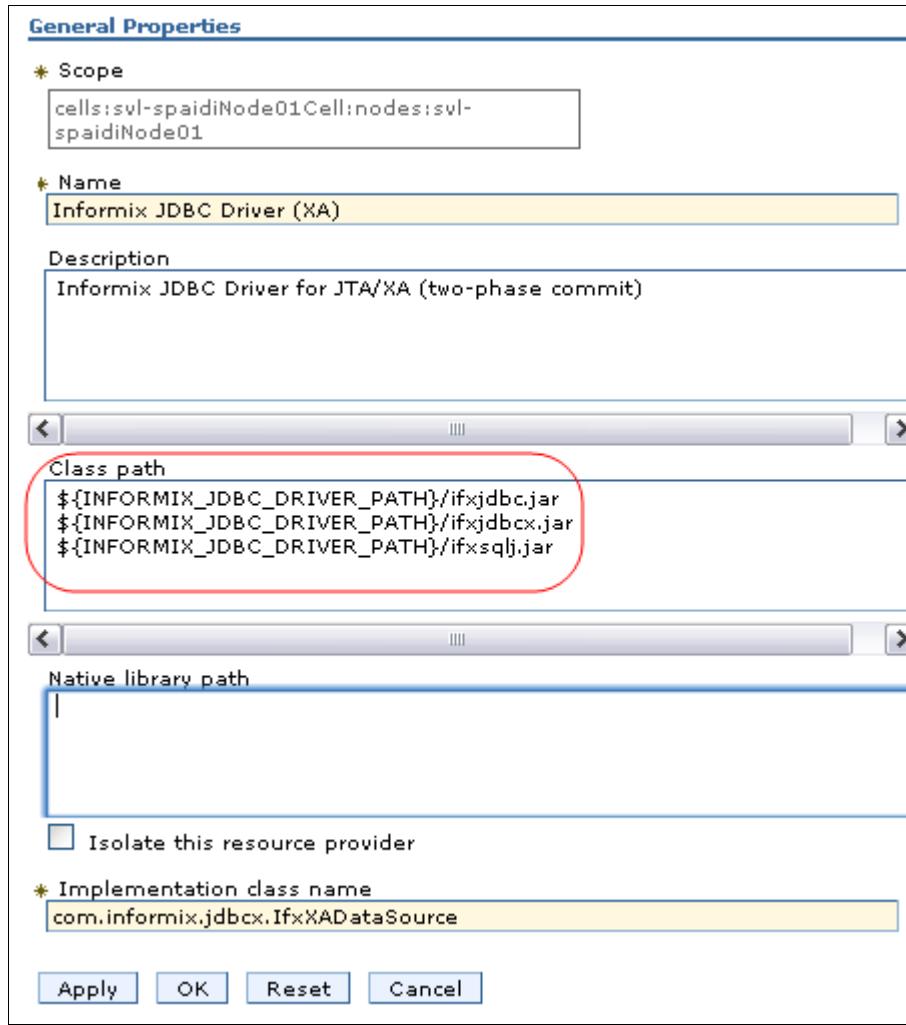


Figure 9-27 Configure the class path

Click **OK**.

9.6.2 Creating the data source

To create a data source, complete the following steps:

1. Expand **Resources** → **JDBC** in the navigation tree, and click **Data sources**.
2. Select the scope on the right. Although you can select **All scopes** to view all resources, you must select a specific scope to create a resource.
3. Click **New** to create a new data source and to start a wizard (Figure 9-28).

The screenshot shows the 'Enter basic data source information' wizard step 1. It has a header bar with the title. Below it is a descriptive text about JDBCs. A note states: 'Requirement: Use the Datasources (WebSphere(R) Application Server V4) configuration. Applications are based on the Enterprise JavaBeans(TM) (EJB) 1.0 specification 2.2 specification.' There are three input fields: 'Scope' containing 'cells:svl-spaidiNode01Cell:nodes:svl-spaidiNode01'; 'Data source name' containing 'stores'; and 'JNDI name' containing 'jdbc/stores'. The 'Data source name' and 'JNDI name' fields are marked with an asterisk (*) indicating they are required.

Figure 9-28 Create a data source, Step 1

Enter a name for the new data source. This name is used for administrative purposes. Enter the JNDI name that will be used to access the data source, and click **Next**.

4. In the next panel (Figure 9-29), select the Informix JDBC driver, and click **Next**.

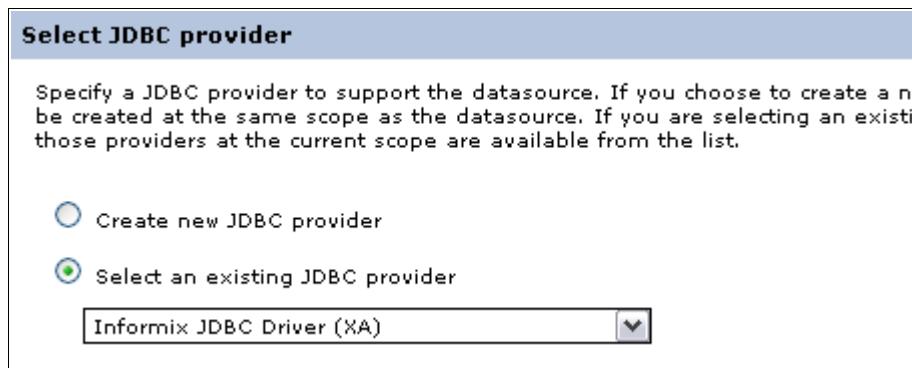


Figure 9-29 Create a data source, Step 2

5. Enter the properties for the database, as shown in Figure 9-30.

The screenshot shows a dialog box titled "Enter database specific properties for the data source". The instruction inside says: "Set these database-specific properties, which are required by the database vendor to manage the connections that are managed through the datasource." Below this, there is a table with five rows of properties:

Name	Value
* Informix lock mode wait	2
* Server name	ol_ids_1150_1
* Database name	stores
Port number	9088

Below the table is a checked checkbox labeled "Use this data source in container managed persistence (CMP)".

Figure 9-30 Create a data source, Step 3

Specify the following information:

- Enter the Informix lock mode wait. Default is 2.
- Enter the server name. This name is the INFORMIXSERVER value, that is the Informix instance name.
- Enter the database name.

- Enter the port number. This number is the olsocpt protocol port number. Check your SQLHOSTS file on UNIX, Linux, or Windows registry on Windows for the correct value to enter.
- Enter the ifxIFXHOST name. This name is host name or the IP Address of the host that is running your Informix instance.

Click **Next**.

6. In the next panel (Figure 9-31), select the authentication alias that provides the user ID and password required to access the database. In this step, we assume that an authentication alias is created already. Click **Next**.



Figure 9-31 Create a data source, Step 4

7. Review the summary of your selections, and click **Finish**.
8. When the data source creation is complete, save the configuration, and synchronize the changes with the nodes.

9. Test the new connection by selecting the new data source and clicking **Test connection** (Figure 9-32).



Figure 9-32 Test the connection

9.7 Configuring connection pooling properties

Performance of an application that connects to a database can be greatly affected by the availability of connections to the database and how those connections affect the performance of the database itself. There are no simple rules that tell you how to configure the connection pool properties. Your configuration is highly dependent on application, network, and database characteristics. You need to coordinate the values that you specify in WebSphere closely with the database administrator.

Remember to include all resources in capacity planning. If 10 applications all connect to a database using separate connection pools of 10 maximum connections, this means that there is a theoretical possibility of 100 concurrent connections to the database. Make sure that the database server has sufficient memory and processing capacity to support this requirement.

To access the connection pool properties:

1. Navigate to **Resources** → **JDBC** → **Data sources**, and click the data source name.
2. Select **Connection pool properties** in the Additional Properties section, which opens the panel shown in Figure 9-33.

General Properties

Scope: cells:Cell02:nodes:sys1Node01:servers:server1

- * **Connection timeout**: 180 seconds
- * **Maximum connections**: 10 connections
- * **Minimum connections**: 1 connections
- * **Reap time**: 180 seconds
- * **Unused timeout**: 1800 seconds
- * **Aged timeout**: 0 seconds

Purge policy: EntirePool

Additional Properties

- [Advanced connection pool properties](#)
- [Connection pool custom properties](#)

Buttons: Apply, OK, Reset, Cancel

Figure 9-33 Data source connection pool properties

Specify the following information:

– **Connection Timeout**

Specify the interval, in seconds, after which a connection request times out and a `ConnectionWaitTimeoutException` is thrown. This can occur when the pool is at its maximum (Max Connections) and all of the connections are in use by other applications for the duration of the wait.

For example, if Connection Timeout is set to 300 and the maximum number of connections is reached, the Pool Manager waits for 300 seconds for an available physical connection. If a physical connection is not available within this time, the Pool Manager throws a `ConnectionWaitTimeoutException`.

Tip: If Connection Timeout is set to 0, the pool manager waits as long as necessary until a connection is allocated.

– **Max Connections**

Specify the maximum number of physical connections that can be created in this pool.

These connections are the physical connections to the back-end database. After this number is reached, no new physical connections are

created and the requester waits until a physical connection that is currently in use is returned to the pool, or a `ConnectionWaitTimeoutException` is thrown.

For example, if `Max Connections` is set to `5`, and there are five physical connections in use, the Pool Manager waits for the amount of time specified in `Connection Timeout` for a physical connection to become free. If, after that time, there are still no free connections, the Pool Manager throws a `ConnectionWaitTimeoutException` to the application.

| – Min Connections

Specify the minimum number of physical connections to be maintained. Until this number is reached, the pool maintenance thread does not discard any physical connections. However, no attempt is made to bring the number of connections up to this number.

For example, if `Min Connections` is set to `3`, and one physical connection is created, that connection is not discarded by the `Unused Timeout` thread. By the same token, the thread does not automatically create two additional physical connections to reach the `Min Connections` setting.

| **Tip:** Set `Min Connections` to zero (0) if the following conditions are true:

- ▶ You have a firewall between the application server and database server.
- ▶ Your systems are not busy 24/7.

| – Reap Time

Specify the interval, in seconds, between runs of the pool maintenance thread.

For example, if `Reap Time` is set to `60`, the pool maintenance thread runs every `60` seconds. The `Reap Time` interval affects the accuracy of the `Unused Timeout` and `Aged Timeout` settings. The smaller the interval you set, the greater the accuracy. When the pool maintenance thread runs, it discards any connections that have been unused for longer than the time value specified in `Unused Timeout`, until it reaches the number of connections specified in `Min Connections`. The pool maintenance thread also discards any connections that remain active longer than the time value specified in `Aged Timeout`.

| **Tip:** If the pool maintenance thread is enabled, set the `Reap Time` value less than the values of `Unused Timeout` and `Aged Timeout`.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

| – **Unused Timeout**

Specify the interval in seconds after which an unused or idle connection is discarded.

Tips:

- ▶ Set the Unused Timeout value higher than the Reap Timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections not in use exceeds the Min Connections setting.
- ▶ Make sure that the database server's timeout for connections exceeds the Unused timeout property specified here. Long lived connections are normal and desirable for performance.

For example, if the unused timeout value is set to 120, and the pool maintenance thread is enabled (Reap Time is not 0), any physical connection that remains unused for two minutes is discarded. Note that accuracy of this timeout, as well as performance, is affected by the Reap Time value. See the Reap Time bullet for more information.

| – **Aged Timeout**

Specify the interval in seconds before a physical connection is discarded, regardless of recent usage activity.

Setting Aged Timeout to *0* allows active physical connections to remain in the pool indefinitely. For example, if the Aged Timeout value is set to 1200, and the Reap Time value is not 0, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that accuracy of this timeout, as well as performance, is affected by the Reap Time value. See Reap Time for more information.

Tip: Set the Aged Timeout value higher than the Reap Timeout value for optimal performance.

| – **Purge Policy**

Specify how to purge connections when a stale connection or fatal connection error is detected.

Valid values are EntirePool and FailingConnectionOnly. If you choose EntirePool, all physical connections in the pool are destroyed when a stale

connection is detected. If you choose FailingConnectionOnly, the pool attempts to destroy only the stale connection. The other connections remain in the pool. Final destruction of connections that are in use at the time of the error might be delayed. However, those connections are never returned to the pool.

Tip: Many applications do not handle a StaleConnectionException in the code. We recommend that you test to ensure that your applications handle them.

Selecting the **Advanced connection pool properties** link allows you to modify the additional connection pool properties. These properties require advanced knowledge of how connection pooling works and how your system performs. For information about these settings, see the *Connection pool advanced settings* topic in the information center.

9.7.1 WebSphere Application Server data source properties

You can set the properties that apply to the WebSphere Application Server connection, rather than to the database connection. To access the connection pool properties, navigate to **Resources** → **JDBC** → **Data sources**, and click the data source name. Select **WebSphere Application Server data source properties** in the Additional Properties section. See Figure 9-33 on page 524.

Clicking the link gives you the window shown in Figure 9-34.

The screenshot shows the 'General Properties' configuration window for a WebSphere data source. It includes sections for Statement cache size, Error detection model, Connection validation properties, and Validation options. The 'Log missing transaction context' checkbox is checked. The 'Use WebSphere Application Server exception mapping model' radio button is selected. The 'Validate new connections' checkbox is unchecked. The 'Number of retries' is set to 100 and the 'Retry interval' is 3 seconds. The 'Validate existing pooled connections' checkbox is unchecked. The 'Validation options' section contains a query: 'SELECT CURRENT SQLID FROM SYSIBM.SYSDUMMY1'.

General Properties	
Statement cache size	<input type="text" value="10"/> statements
<input type="checkbox"/> Enable multithreaded access detection	
<input type="checkbox"/> Enable database reauthentication	
<input checked="" type="checkbox"/> Log missing transaction context	
<input type="checkbox"/> Non-transactional data source	
Error detection model	
<input type="radio"/>	Use WebSphere Application Server exception checking model
<input checked="" type="radio"/>	Use WebSphere Application Server exception mapping model
Connection validation properties	
<input type="checkbox"/> Validate new connections	
Number of retries	<input type="text" value="100"/>
Retry interval	<input type="text" value="3"/> seconds
<input type="checkbox"/> Validate existing pooled connections	
Retry interval	<input type="text" value="0"/> seconds
Validation options	
Query	<input type="text" value="SELECT CURRENT SQLID FROM SYSIBM.SYSDUMMY1"/>

Figure 9-34 WebSphere data source custom properties

Specify the following information:

► Statement Cache Size

Specify the number of prepared statements that are cached per connection. A prepared statement is a precompiled SQL statement that is stored in a prepared statement object. This object is used to execute the given SQL statement multiple times. The WebSphere Application Server data source optimizes the processing of prepared statements.

In general, the more statements your application has, the larger the cache should be. For example, if the application has five SQL statements, set the statement cache size to 5, so that each connection has five statements.

Tip: This setting is vital to performance of the database and will most likely require tuning to suit the specific application. Our general experience is that the default is not high enough for best performance.

- ▶ Enable multi-threaded access detection

If you enable this feature, the application server detects the existence of access by multiple threads.

- ▶ Enable database reauthentication

Connection pool searches do not include the user name and password. If you enable this feature, a connection can still be retrieved from the pool, but you must extend the DataStoreHelper class to provide implementation of the `doConnectionSetupPerTransaction()` method where the reauthentication takes place.

Connection reauthentication can help improve performance by reducing the overhead of opening and closing connections, particularly for applications that always request connections with different user names and passwords.

- ▶ Log missing transaction context

Specifies whether the container issues an entry to the activity log when an application obtains a connection without a transaction context.

- ▶ Non-transactional data source

Setting the flag to true will cause the Application Server to never enlist the connections from the datasource in global or local transactions. Applications must explicitly call `setAutoCommit(false)` on the connection if they want to start a local transaction on the connection, and they must commit or rollback the transaction that they started. Note: this property should rarely be set to true, however the Java Persistence API (JPA) requires both JTA and non-JTA datasources.

- ▶ Error detection model

The error detection model has been expanded and the data source has a configuration option that you can use to select the exception mapping model or the exception checking model for error detection.

▶ Connection validation properties

There are two properties, and you can choose both. If you check the Validate new connections box, the application server tries to connect to database. If you select this property, you can specify how often, in seconds (interval).

If you check the Validate existing pooled connections box, the application server retries to make a connection. If you select this property, you can specify retry interval for the server reroute. The pretest SQL string is sent to the database to test the connection.

New in V7: Connection validation by SQL query is deprecated in WebSphere Application Server V7.0. You can use validation by the JDBC Driver instead. If you use the property of validation by JDBC driver, you need JDBC 4.0 or later. If you do not have JDBC 4.0, you have to update the JDBC driver first.

▶ Advanced DB2 features

– Optimize for get/use/close/connection pattern with heterogeneous pooling

If you check this property, the heterogeneous pooling feature allows you to extend the data source definition. You can specify retry interval for client reroute, how often to retry, alternate server name or names for the DB2 server, port number, JNDI name. Details are described in “Extended DB2 data source” on page 531.

– DB2 automatic client reroute options

(New in V7) Client reroute for DB2 allows you to provide an alternate server location, in case the connection to the database server fails. If you decide to use client reroute with the persistence option, the alternate server information will persist across Java Virtual Machines (JVMs). In the event of an application server crash, the alternate server information will not be lost when the application server is restored and attempts to connect to the database.

9.7.2 Extended DB2 data source

(New in V7) The DB2 Universal JDBC Driver and DB2 using IBM JCC Driver support extends a DB2 data source with what is known as heterogeneous pooling. The extended DB2 data source configures a WebSphere DB2 data source with a set of core data source properties. An application can define one or more non-core set of data source properties and associate each with a different resource-reference that points to the main WebSphere DB2 data source.

The benefit of using an extended DB2 data source is that it allows applications to share the same WebSphere connection pool even though each application can have its own set of data source properties.

The sharing leads to a reduction of the number of open connections, and it pushes to reduce resource consumption on both the client side (WebSphere) and the server side (database layer).

For more information, refer to the following article in the information center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/tdat_heteropool.html



Accessing EIS applications from WebSphere

The J2EE Connector architecture (JCA) defines a standard architecture for connecting the J2EE platform to heterogeneous Enterprise Information Systems (EIS), for example, ERP, mainframe transaction processing, database systems, and existing applications not written in the Java programming language. By defining a set of scalable, secure, and transactional mechanisms, the JCA enables the integration of EISs with application servers and enterprise applications. WebSphere Application Server V7.0 provides a complete implementation of the JCA 1.5 specification, including the features of the JCA 1.0 Specification.

In this chapter, we discuss the various considerations for accessing EIS applications from WebSphere.

We cover the following topics:

- ▶ “JCA resource adapters” on page 534
- ▶ “Resource adapters” on page 536
- ▶ “Configuring J2C connection factories” on page 541
- ▶ “Resource authentication” on page 544

10.1 JCA resource adapters

The JCA Resource Adapter is a system-level software driver supplied by EIS vendors or other third-party vendors. It provides the following functionality:

- ▶ Provides connectivity between J2EE components, such as an application server or an application client and an EIS.
- ▶ Plugs into an application server.
- ▶ Collaborates with the application server to provide important services, such as connection pooling, transaction, and security services.

JCA defines the following set of system-level contracts between an application server and EIS:

- A *connection management contract* lets an application server pool connect to an underlying EIS, and lets application components connect to an EIS. This leads to a scalable application environment that can support a large number of clients requiring access to EISs.
- A *transaction management contract* between the transaction manager and an EIS supports transactional access to EIS resource managers. This contract lets an application server use a transaction manager to manage transactions across multiple resource managers. This contract also supports transactions that are managed internally to an EIS resource manager without the necessity of involving an external transaction manager.
- A *security contract* enables a secure access to an EIS. This contract provides support for a secure application environment, reducing security threats to the EIS and protecting valuable information resources managed by the EIS.

The resource adapter implements the EIS-side of these system-level contracts.

- ▶ Implements the Common Client Interface (CCI) for EIS access.

The CCI defines a standard client API through which a J2EE component accesses the EIS. This simplifies writing code to connect to an EIS data store.

The resource adapter provides connectivity between the EIS, the application server, and the enterprise application via the CCI.

- ▶ Implements the standard Service Provider Interface (SPI).

The SPI integrates the transaction, security, and connection management facilities of an application server (JCA Connection Manager) with those of a transactional resource manager.

Multiple resource adapters (one resource adapter per type of EIS) are pluggable into an application server. This capability enables application components deployed on the application server to access the underlying EISs. This is shown in Figure 10-1.

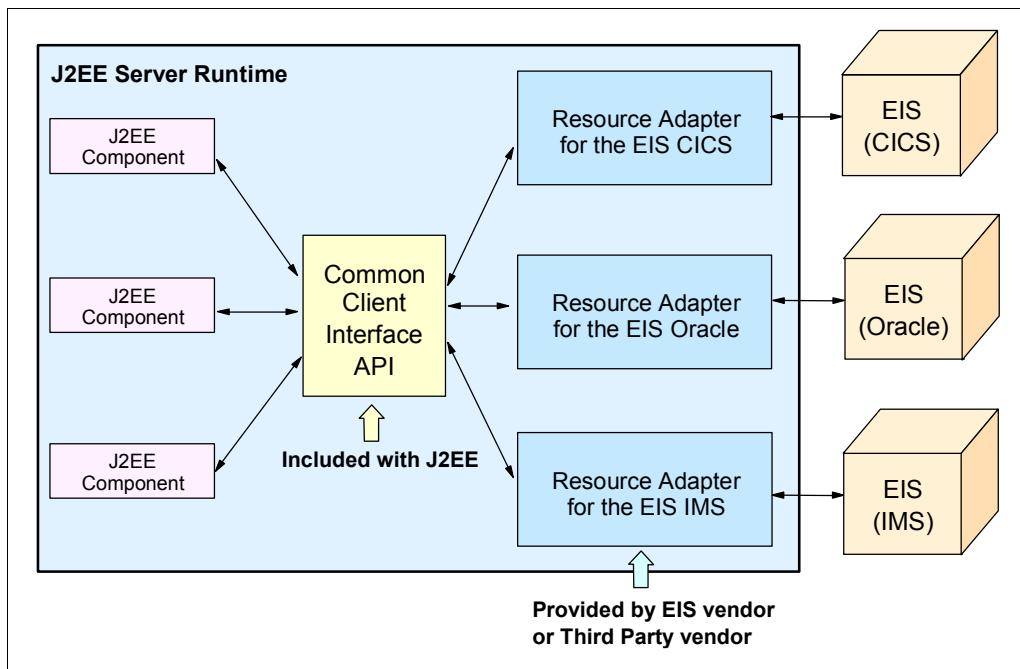


Figure 10-1 Common Client Interface API

10.1.1 WebSphere Application Server JCA support

In WebSphere Application Server, two types of objects are configured for JCA support:

- ▶ Resource adapters
- ▶ Connection factories

The role of the WebSphere administrator is to:

- ▶ Install and define the resource adapter.
- ▶ Define one or more connection factories associated with the resource adapter.

From the application point of view, the application using the resource adapter requests a connection from the connection factory through a JNDI lookup. The connection factory connects the application to the resource adapter.

10.2 Resource adapters

A WebSphere resource adapter administrative object represents the library that supplies implementation code for connecting applications to a specific EIS, such as CICS or SAP. Resource adapters are stored in a Resource Adapter Archive (RAR) file, which is a Java archive (JAR) file used to package a resource adapter for the connector architecture. The file has a standard file extension of .rar.

A RAR file can contain the following elements:

- ▶ EIS-supplied resource adapter implementation code in the form of JAR files or other executables, such as DLLs
- ▶ Utility classes
- ▶ Static documents, such as HTML files for developer documentation, not used for runtime
- ▶ J2C common client interfaces, such as cci.jar
- ▶ A mandatory deployment descriptor (ra.xml):

This deployment descriptor instructs the application server about how to use the resource adapter in an application server environment. The deployment descriptor contains information about the resource adapter, including security and transactional capabilities, and the ManagedConnectionFactory class name.

The RAR file or JCA resource adapter is provided by your EIS vendor.

New in V7: Registering the resource adapter with the high availability manager specifies that the high availability (HA) manager will manage the lifecycle of a JCA 1.5 resource adapter in a cluster. This ensures that applications using resource adapters for inbound communication remain highly available. To that end, appropriate use of the HA capability options enable you to set up an environment that will be able to implement failover for inbound activity when a server goes down.

WebSphere provides two JCA resource adapters:

- ▶ The WebSphere Relational Resource Adapter, used to connect to relational databases using JDBC
- ▶ The SIB JMS Resource Adapter, used to connect to the default messaging provider

Connection factory

The WebSphere connection factory administrative object represents the configuration of a specific connection to the EIS supported by the resource adapter. The connection factory can be thought of as a holder of a list of connection configuration properties.

Application components, such as CMP enterprise beans, have `cmpConnectionFactory` descriptors that refer to a specific connection factory, not to the resource adapter.

10.2.1 Installing and configuring resource adapters

To use a resource adapter, you need to install the resource adapter code and create connection factories that use the adapter. The resource adapter configuration is stored in the `resources.xml` file.

There are two ways to make a resource adapter (.rar file) available to applications. One way is to install the adapter into WebSphere Application Server. The other way is to install the adapter in the application (embedded adapter). For example, Rational Application Developer embeds resource adapters when you create a J2C application. This chapter describes installing the adapter into WebSphere Application Server.

To install an adapter, do the following steps:

1. From the administrative console, expand **Resources** from the navigation tree and click **Resource Adapters** and select a scope (Figure 10-2).

Note that you can see all the WebSphere built-in resources by selecting the **Show built-in resources** preference.

Resource adapters

Resource adapters

Use this page to manage resource adapters, which provide the fundamental interface for connecting applications to an Enterprise Information System (EIS). The WebSphere(R) Relational Resource Adapter is embedded within the product to provide access to relational databases. To access another type of EIS, use this page to install a standalone resource adapter archive (RAR) file. You can configure multiple resource adapters for each installed RAR file.

Scope: Cell=Cell02, Node=sys1Node01

Show scope selection drop-down list with the all scopes option

Scope specifies the level at which the resource definition is visible.
For detailed information on what scope is and how it works, [see the scope settings help](#).

Preferences

Maximum rows

Retain filter criteria

Show items at the following authorization group level:

Show built-in resources

Select	Name	Scope
<input type="checkbox"/>	WebSphere MQ Resource Adapter	Node=sys1Node01
Total 1		

Figure 10-2 JCA resource adapters

2. Click **Install RAR** to install a new resource adapter.

3. Enter the path to the RAR file supplied by your EIS vendor. It can reside locally, on the same machine as the browser, or on any of the nodes in your cell. See Figure 10-3.

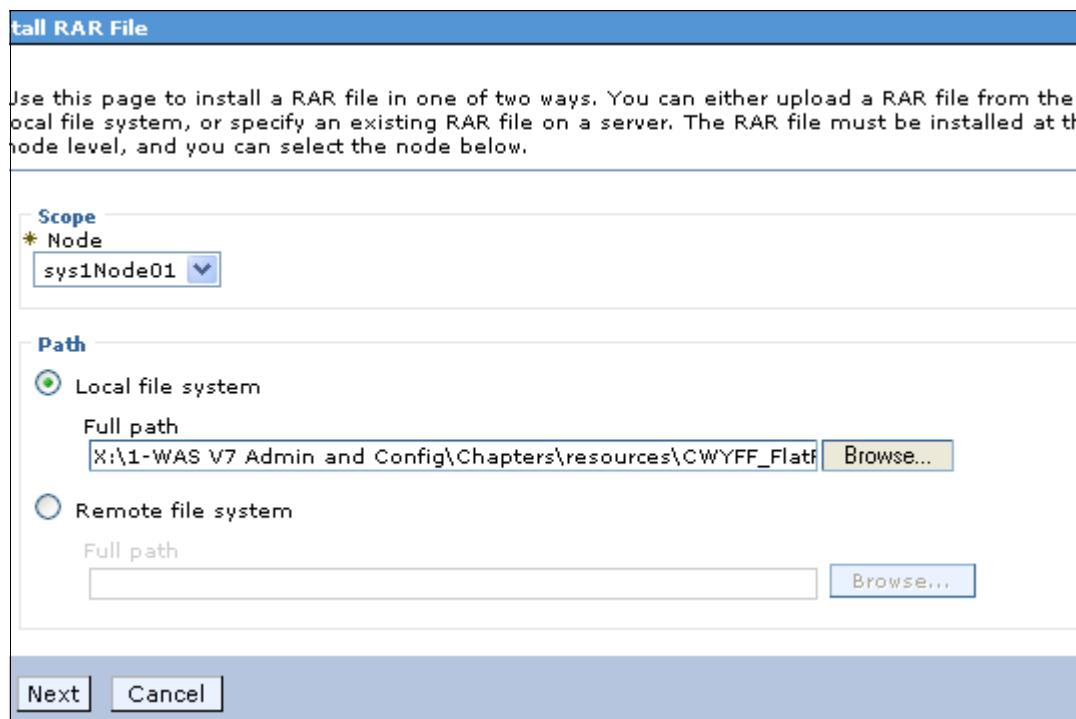


Figure 10-3 RAR file location

Select the node where you want to install the RAR file. You have to install the file on each node separately.

Click **Next**.

4. The Configuration page for the resource adapter selected is displayed. This is shown in Figure 10-4.

General Properties

* Scope
cells:Cell40:nodes:node40a

Name
IBM WebSphere Adapter for Flat Files

Description

Archive path
\${CONNECTOR_INSTALL_ROOT}

Class path

Native library path

Isolate this resource provider

Figure 10-4 JCA resource adapter properties

In this example, you do not have to configure any properties. The defaults combined with the information supplied in the RAR file provide all the information needed. However, you have the option of configuring the following properties:

– Name:

Create an administrative name for the resource adapter.

- Description:
Create an optional description of the resource adapter, for your administrative records.
 - Archive path:
This field is the path where the RAR file is installed. If this property is not specified, the archive will be extracted to the absolute path represented by the \${CONNECTOR_INSTALL_ROOT} variable. The default is *profile_root*/installedConnectors/*adapter_name.rar*.
 - Class path:
A list of paths or JAR file names that together form the location for the resource adapter classes. The resource adapter code base itself, the RAR file, is automatically added to the classpath.
 - Native path:
This is a list of paths that together form the location for the resource adapter native libraries (.dll, and .so files).
5. Click **OK**.
 6. Save the configuration and synchronize the nodes.

10.3 Configuring J2C connection factories

Note: The terms J2C and JCA both refer to J2EE Connector Architecture and they are used here interchangeably.

A J2C connection factory represents a set of connection configuration values. Application components such as EJBs have <resource-ref> descriptors that refer to the connection factory, not the resource adapter. The connection factory is just a holder of a list of connection configuration properties. In addition to the arbitrary set of configuration properties defined by the vendor of the resource adapter, there are several standard configuration properties that apply to the connection factory. These standard properties are used by the connection pool manager in the application server runtime and are not used by the vendor supplied resource adapter code.

To create a J2C connection factory, do the following steps:

1. Select **Resources** → **Resource Adapters** → **J2C connection factories**.
You will see a list of J2C connection factories at the selected scope.
2. Click **New** to create a new connection factory, or select an existing one to modify the connection factory properties.

The J2C Connection Factory Configuration page is shown in Figure 10-5.

General Properties		Additional Properties
* Scope cells:Cell40:nodes:node40a		Connection pool properties
* Provider IBM WebSphere Adapter for Flat Files		Advanced connection factory
* Name FFadapter		Custom properties
JNDI name eis/FFadapterSample		Related Items
Description		JAAS - J2C authentication
* Connection factory interface javax.resource.cci.ConnectionFactory		
Category		
Security settings Select the authentication values for this resource.		
Component-managed authentication alias (none)		
Mapping-configuration alias DefaultPrincipalMapping		
Container-managed authentication alias t60Node01Cell/samples		
Container-managed authentication		
Authentication preference None		

Figure 10-5 J2C connection factory properties

The general properties are:

- Name:

Type an administrative name for the J2C connection factory.

- JNDI name:

This field is the connection factory name to be registered in the application server's name space, including any naming sub context.

When installing an application that contains modules with J2C resource references, the resources defined by the deployment descriptor of the module need to be bound to the JNDI name of the resource.

As a convention, use the value of the Name property prefixed with eis/, for example,

eis/<ConnectionFactoryName>

- Description:

This is an optional description of the J2C connection factory, for your administrative records.

- Connection factory interface:

This field is the name of the connection factory interfaces supported by the resource adapter.

- Category:

Specify a category that you can use to classify or group the connection factory.

- Security settings:

You have multiple options when securing access to the J2C resource.

While component-managed might be faster in some instances, it is not the best solution for security. Container-managed authentication is the preferred method.

For more information, see 10.4, “Resource authentication” on page 544.

3. Click **Apply**. The links under the Additional Properties section for connection pool, advanced connection factory, and custom properties become active.

The connection pool properties can affect performance of your application. You should monitor and adjust these settings to maximize performance.

The advanced connection factory properties are shown in Figure 10-6.

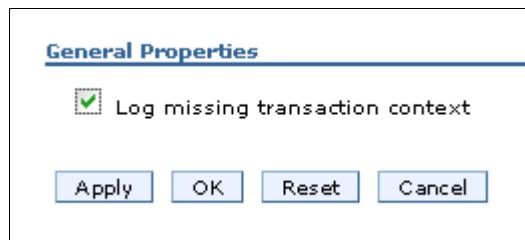


Figure 10-6 Advanced connection factory properties

The J2EE programming model indicates that connections should always have a transaction context. However, some applications do not have a context associated with them. The Log missing transaction context option tells the container to log the fact that there is a missing transaction context in the activity log when the connection is obtained.

10.4 Resource authentication

Resources often require you to perform authentication and authorization before an application can access them. You can configure the settings to determine how this is done in a number of ways. This section discusses the configuration settings and how to use them. However, before implementing any security, you should review the information in *WebSphere Application Server V7 Security Handbook*, SG24-7660.

The party responsible for the authentication and authorization is determined by the res-auth setting found in the Web and EJB deployment descriptors. There are two possible settings:

- ▶ **res-auth=Container:** WebSphere is responsible.
The authentication data is supplied by the application server.
- ▶ **res-auth=Application:** The application, or component, is responsible.
The authentication data is taken from the following elements, in order:
 - The user ID and password that are passed to the getConnection method. (This is not recommended for obvious reasons. This implies that the user ID and password are coded in the application).
 - The component-managed authentication alias in the connection factory or the data source
 - The custom properties user name and password in the data source

These settings can be configured during application assembly using Rational Application Developer in the EJB or Web deployment descriptor. They can also be set or overridden during application installation. See Table 10-1.

Table 10-1 Authentication settings

Authentication type	Setting at assembly Authorization type	Setting during installation Resource authorization
WebSphere managed: res-auth=Container	Container	Container
Application (component) managed: res-auth=Application	Per_Connection_Factory	Per application

10.4.1 Container-managed authentication

Container-managed authentication removes the requirement that the component programmatically supply the credentials for accessing the resource. Instead of calling the `getConnection()` method with a `ConnectionSpec` object, `getConnection()` is called with no arguments. The authentication credentials are then supplied by the Web container, application container, or the EJB container, depending on from where the resource is accessed. WebSphere Application Server supports the JAAS specification, so the credentials can be mapped from any of the configured JAAS authentication login modules, including any custom JAAS authentication login module.

The default selection for the JAAS application login module (in the `mapping-configuration-alias` field of the J2C connection factory), `DefaultPrincipleMapping`, maps the user ID and password using a pre-configured J2C authentication alias.

Container-managed authentication is the preferred method.

10.4.2 Component-managed authentication

In the case of component-managed authentication, the application component accessing the resource or adapter is responsible for programmatically supplying the credentials. WebSphere can also supply a default component-managed authentication alias if available. After obtaining the connection factory for the resource from JNDI, the application component creates a connection to the resource using the `create` method on the connection factory supplying the credentials. If no credentials are supplied when creating a connection and a component-managed authentication alias has been specified on the J2C connection factory, the credentials from the authentication alias will be used.

Assuming that the credentials are valid, future requests using the same connection will use the same credentials.

The application follows these basic steps:

1. Get the initial JNDI context.
2. Look up the connection factory for the resource adapter.
3. Create a ConnectionSpec object holding credentials.
4. Obtain a connection object from the connection factory by supplying the ConnectionSpec object.



Monitoring

Being able to measure and monitor system interactions helps IT in providing business continuity. Monitoring capabilities play a key role in successfully managing enterprise systems. In WebSphere Application Server, there are a number of tools that can contribute to an organization's monitoring strategy and provide insights into the performance of the application server.

In this chapter, we provide an introduction to these toolsets.

We cover the following topics:

- ▶ “Overview” on page 548
- ▶ “Enabling monitoring infrastructures” on page 552
- ▶ “Viewing the monitoring data” on page 567
- ▶ “Monitoring scenarios” on page 579
- ▶ “ITCAM for WebSphere” on page 593
- ▶ “Monitoring considerations summary” on page 604

11.1 Overview

IT environments are complex, involving many different servers working together to deliver the electronic functions of business. In a single user interaction, it is typical that information can be retrieved from many systems. Consider the very simple distributed WebSphere Application Server environment in Figure 11-1.

The stars in the figure highlight that even a simple Web application request can pass through a whole series of dependent servers in order to successfully complete a request. JEE is a component based architecture, requiring a request to interact and use ‘n’ number of these components to complete. Monitoring system components and their performance can become complex, yet is critical to understanding the overall performance of an application.

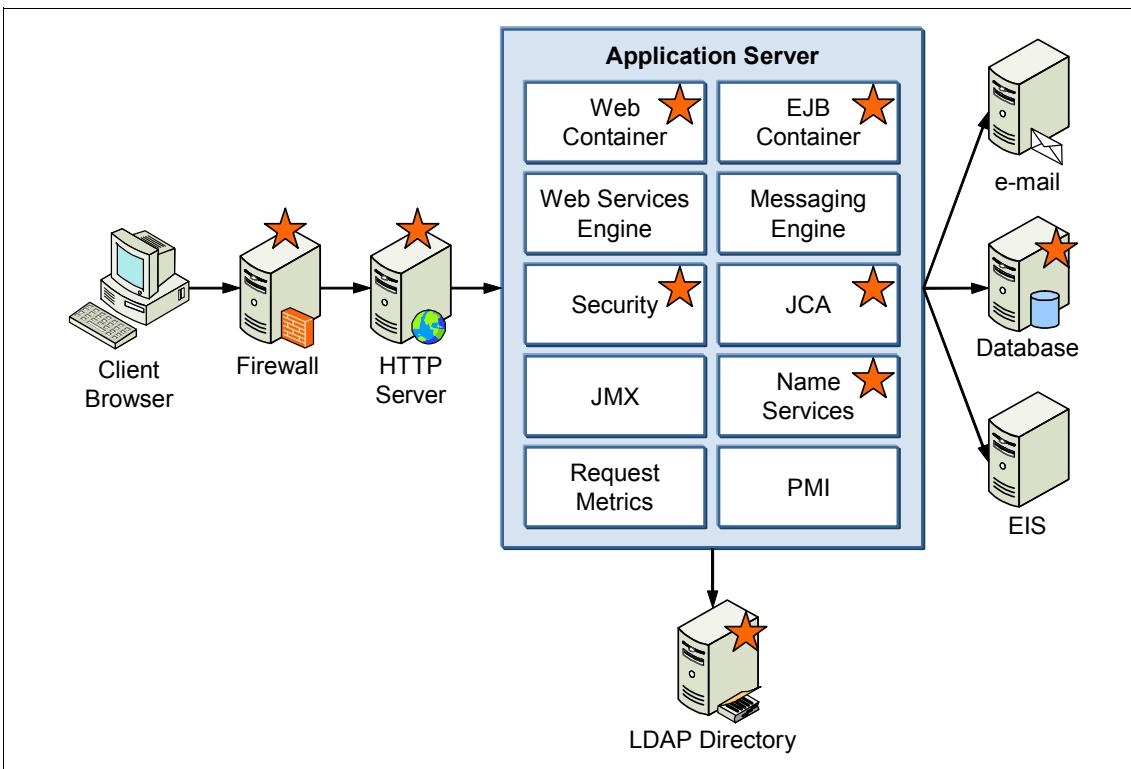


Figure 11-1 Simple Web system topology

Monitoring the systems contributes to overall systems management by:

- ▶ Establishing an understanding of the performance baseline and of what runtime behaviors constitute “normal” operations

- ▶ Measuring performance and identifying poorly performing systems and components
- ▶ Identifying service failures, and can assist in root cause identification

WebSphere Application Server monitoring tools rely primarily on information gathered from two core data infrastructures:

- ▶ Performance Monitoring Infrastructure (PMI), which is a collection of statistical agents scattered through out the application server that gather statistical data on the performance of the application server components.
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/cprf_pmidata.html
- ▶ Request metrics, which are primarily a set of timing agents that track a request as it navigates the components of the application server. A key differentiation of request metrics is that they are measured at the request level. The focus of a request metric is to record the time spent by individual requests in different components of the application and at the end of the request provide a record of where time was spent in the request.
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.base.doc/info/aes/ae/tprf_requestmetrics.html

11.1.1 Monitoring scenarios

This chapter demonstrates the use of the system monitoring infrastructures by using different monitoring scenarios, which are summarized in Figure 11-1.

Note: These scenarios cover several common uses of the monitoring tools, but it should be understood that many of the different types of data are not explicitly discussed. This chapter provides an introduction to the tool sets and a way for new administrators to get started, and serves as a reminder for experienced administrators about some of the tools that they might not have utilized in some time.

Table 11-1 Monitoring Scenario Summary

Monitoring Scenario	Chapter Reference
How is monitoring data activated and what are the monitoring choices? <ul style="list-style-type: none"> ▶ PMI data defaults ▶ Enabling request metrics 	<p>“Enabling monitoring infrastructures” on page 552</p> <ul style="list-style-type: none"> ▶ “PMI defaults and monitoring settings” on page 552 ▶ “Enable request metrics” on page 561

Monitoring Scenario	Chapter Reference
<p>What are the tools that I can use for understanding the collected data?</p> <ul style="list-style-type: none"> ▶ Tivoli Performance Viewer. 	11.3, “Viewing the monitoring data” on page 567
<p>Understanding how application(s) are interacting with a database. This scenario helps identify data that will help investigate:</p> <ul style="list-style-type: none"> ▶ Is the database responding fast enough? ▶ Is there enough connections to the database? ▶ Are the connections being returned to the pool? 	“Database interactions” on page 580
<p>Understanding JCA connection pool utilization. This scenario helps administrators understand:</p> <ul style="list-style-type: none"> ▶ What is the response like for the JCA connection pools. ▶ Are there enough connections to support the system interactions? ▶ Are the connections being returned to the pool? 	“Database interactions” on page 580 “JCA interactions” on page 581
<p>What about threading resources. Is there sufficient JMS, EJB, and Web threads allocated in the server thread pools? This scenario helps administrator understand:</p> <ul style="list-style-type: none"> ▶ How to monitor and improve system related throughput. ▶ Current limits on system concurrency. 	“Threading resources” on page 582
<p>Monitoring memory allocation and garbage collection. JVM memory tuning is vital to application server performance, and this scenario introduces administrators to tools that assist in making memory tuning choices.</p>	“JVM memory usage” on page 586
<p>Finding bottlenecks in response time, Services, EJB, Web and response times. Understanding of component response time helps in the identification of application bottlenecks and performance issues.</p>	“Request level details” on page 587
<p>Special features of ITCAM for WebSphere in WebSphere Application Server v7.0</p>	“ITCAM for WebSphere” on page 593

The monitoring infrastructure scenarios are demonstrated using the example environment shown in Figure 11-2.

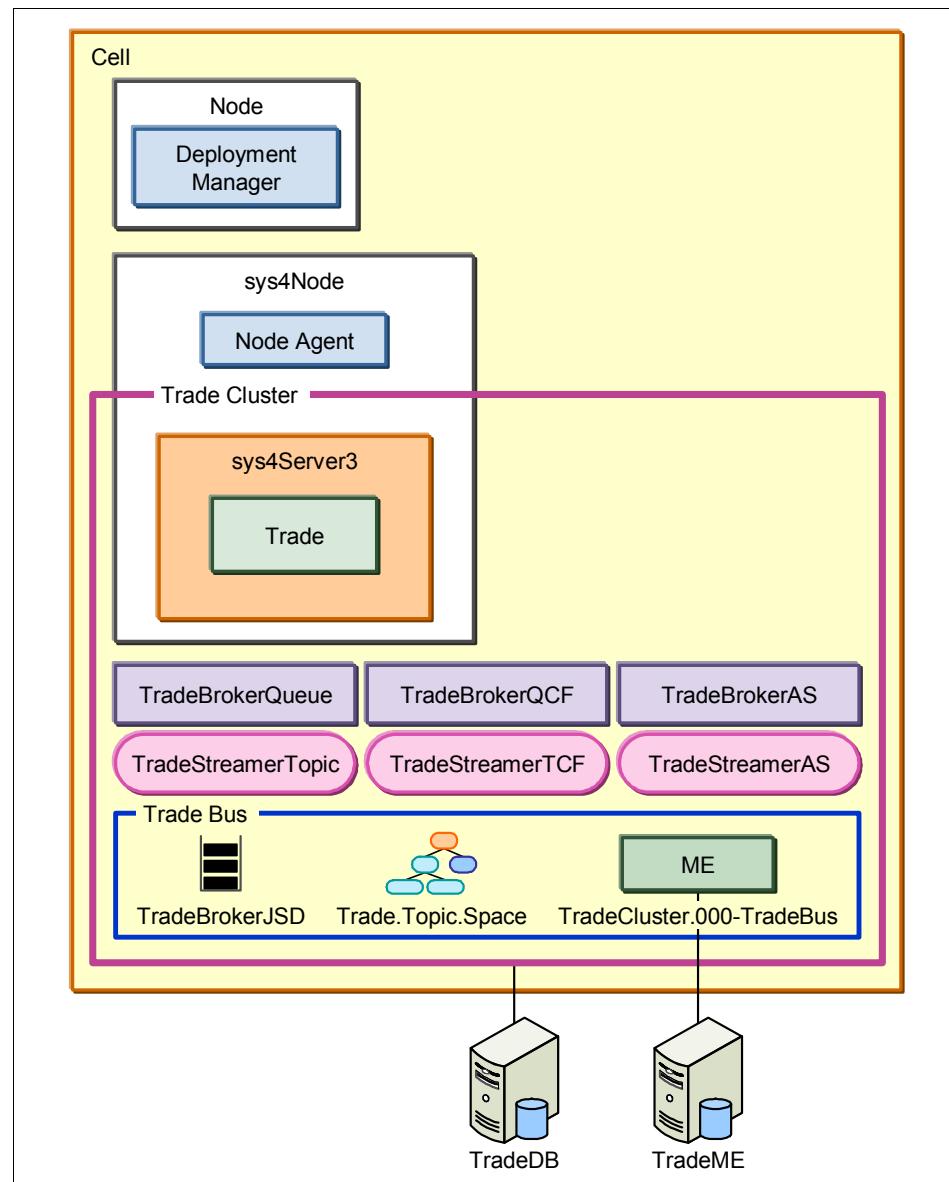


Figure 11-2 Example monitoring environment

Note: We did no special tuning in the test environment. We simply installed the Trade performance application into a base server configuration and changed the default memory.

11.2 Enabling monitoring infrastructures

This section shows you how to enable the PMI monitoring infrastructure and the request metrics that provide the performance data.

11.2.1 PMI defaults and monitoring settings

The enabling of PMI data is managed on a server-by-server basis. In the administrative console, do the following steps:

1. Navigate to the **Performance Monitoring Infrastructure (PMI)** menu item in the **Monitoring and Tuning** navigation menu.
2. Select the link for server for that you want to manage the PMI controls for. In this example, sys4Server3 is the server. Figure 11-3 shows the PMI configuration panel for the server.

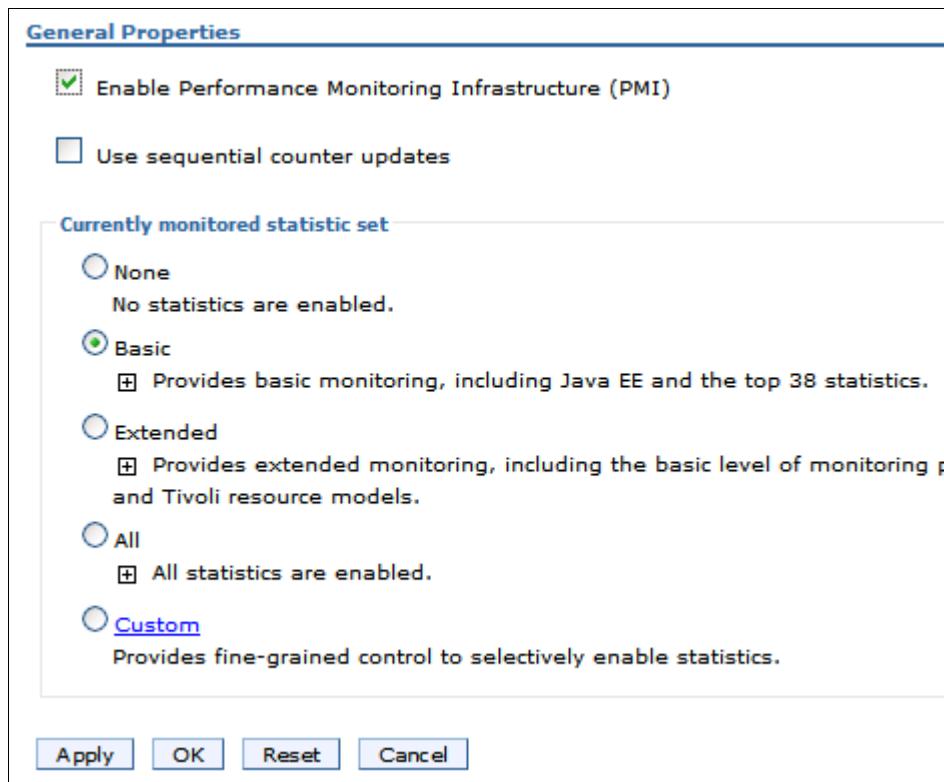


Figure 11-3 Default PMI settings

On this panel it is worthy to note that:

- PMI is enabled by default.
- The default statistical set is the basic set.

The PMI data can be changed at runtime using settings in the Runtime tab.

Note: Disabling and enabling of PMI data requires a server restart.

The enabling and disabling of PMI is not available on the Runtime tab. However, the monitoring level can be set to None in a server with PMI enabled using the Runtime tab.

Understanding the sets of PMI statistics

The PMI statistic sets represent a group of individual statistical agents. The types of statistics that PMI can collect are classified. Information about these classes can be found in the WebSphere Information Center on the *PMI data classification* page at:

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

Everyone working in system administration knows that every action executed in an environment has a cost. Monitoring is no different, and for PMI, the cost of monitoring is impacted primarily by two factors:

1. The amount of data that is monitored.
2. The overhead of individual performance metrics. Not all metrics have the same collection cost.

With PMI, there are multiple sets of statistics that can be enabled as shown in Figure 11-3 on page 553. These sets of statistics are:

- ▶ None
- ▶ Basic
- ▶ Extended
- ▶ All
- ▶ Custom

None and All are fairly self explanatory, so here we take a closer look at the options provided by Basic, Extended, and Custom.

Basic statistic set

The Basic statistic set is the default setting. The basic setting is configured with the intention of providing an overall understanding of application server health, including statistics as outlined in the JEE specification, as well as other common performance hotspots and key monitoring points for JEE applications. Later, we discuss how to determine the overhead and level of a statistic (see “Getting more information about statistics sets” on page 558).

Figure 11-4 shows the list of PMI counters that are active for the basic PMI data level. For details on each counter, refer to “Getting more information about statistics sets” on page 558.

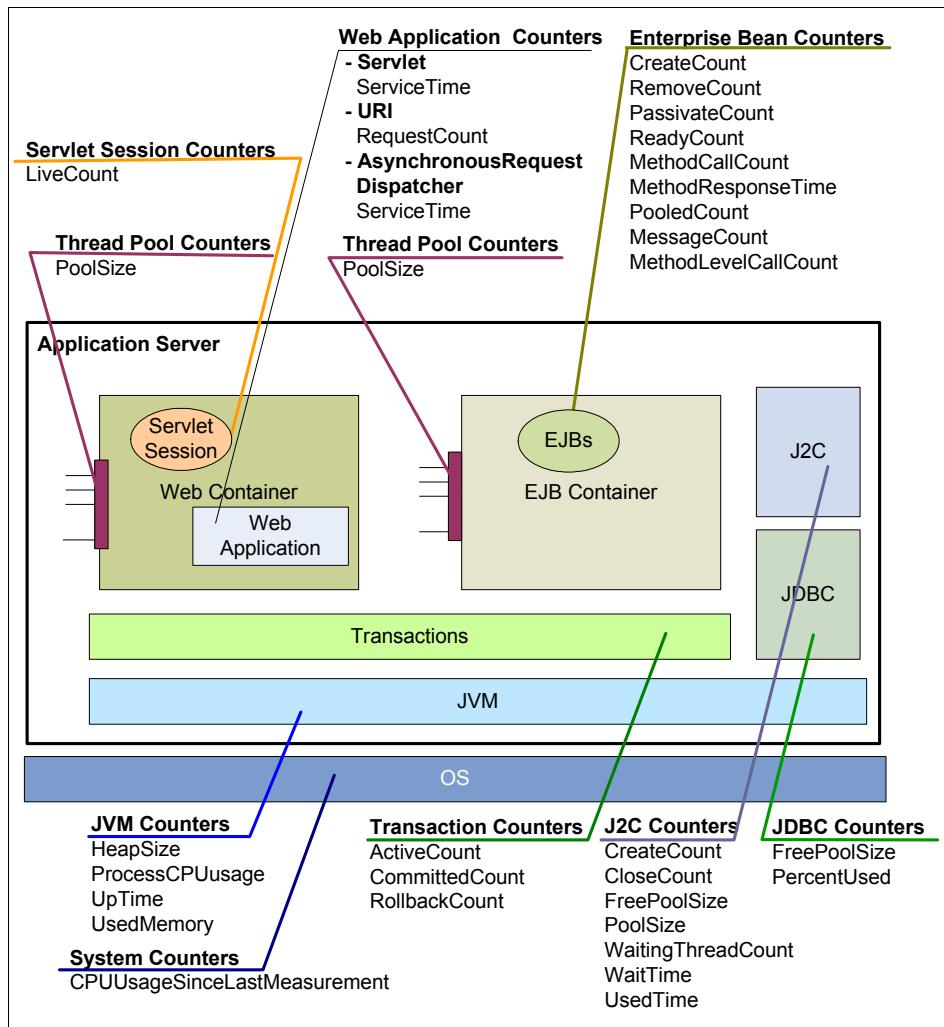


Figure 11-4 PMI basic counters

Extended statistic set

The extended PMI data set has the basic set as well as some additional statistics with a particular emphasis on statistics that look at the load on the server and the servers response to the load being applied. The statistical agents in the extended set might or might not apply to a JEE application depending on the individual application architecture and environment configurations.

Figure 11-5 shows the extended metrics that are additional to those of the basic configuration.

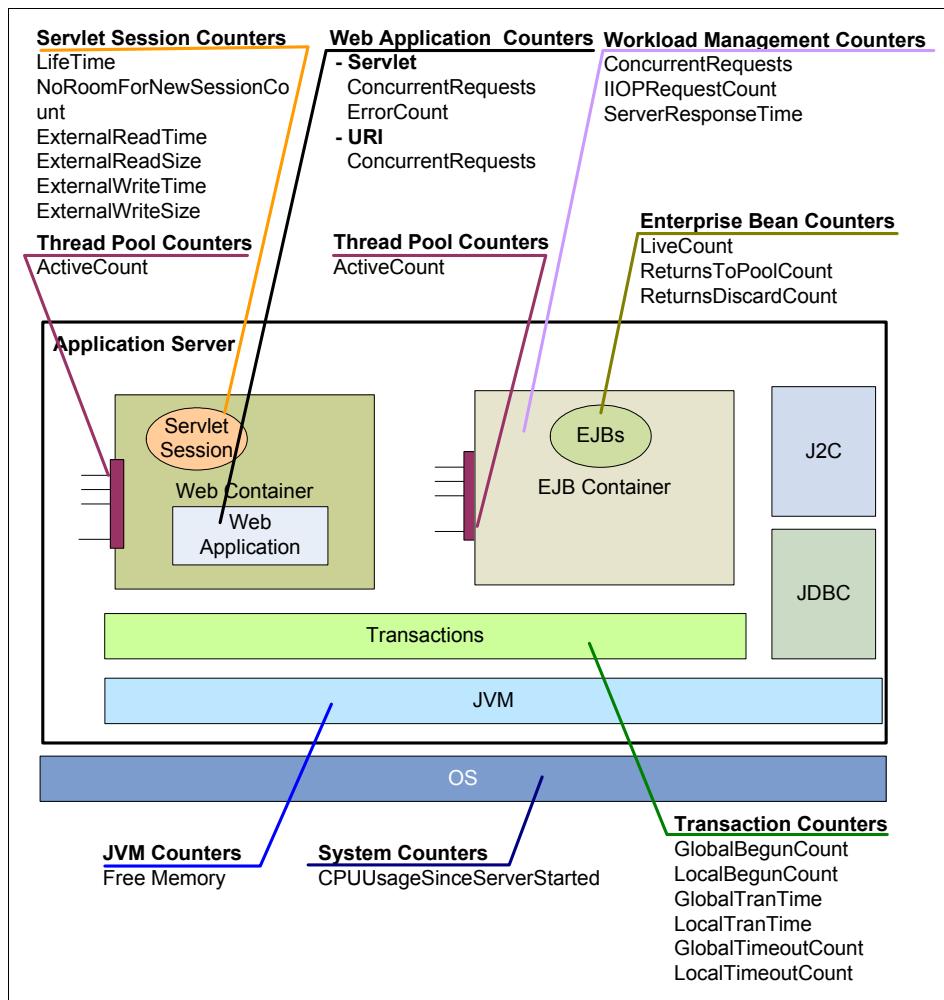


Figure 11-5 PMI extended counters

Custom statistic set

The Custom PMI data collection set allows the administrator to choose the counters that are most appropriate for the application(s) that are deployed on the server. Each counter is individually activated. This is the most powerful configuration but requires that the administrator spend some time reviewing the available statistical counters and also that the administrator understands the type of counter that is useful for the applications.

For example, consider the counters activated for `ServletSession` if the extended data set is selected. The counters are:

- ▶ `LiveCount`
- ▶ `LifeTime`
- ▶ `NoRoomForNewSessionCount`
- ▶ `ExternalReadTime`
- ▶ `ExternalReadSize`
- ▶ `ExternalWriteTime`
- ▶ `ExternalWriteSize`

The `NoRoomForNewSessionCount` counter only applies if the **Allow overflow** from the Web container session management was changed from its default value of true. This attribute is shown in Figure 11-6.

The screenshot shows the 'Session management' configuration page for 'sys4Server3'. The 'General Properties' section is visible, containing settings for session tracking mechanism, maximum in-memory session count, and the 'Allow overflow' checkbox. The 'Allow overflow' checkbox is checked, indicating it is enabled by default.

Session tracking mechanism:

- Enable SSL ID tracking
- [Enable cookies](#)
- Enable URL rewriting
- Enable protocol switch rewriting

Maximum in-memory session count: 1000 sessions

Allow overflow

Session timeout: [empty input field]

Figure 11-6 Allow overflow default is true

Similarly, the counters related to external session management only apply if session persistence is configured. Hence the activation of the extended session information does little for an application where the overflow option is not modified and session persistence is not configured.

With a custom metric approach, the administrator could choose to simply add LiveCount and LifeTime counters as well as perhaps choosing other metrics of interest, such as the TimeoutInvalidationCount, to measure how many sessions are being timed out rather than logged off.

Tip for using custom PMI settings: The counters used for the basic set can be customized to form a baseline for the custom counter activation. They should be supplemented with additional counters that are relevant for the application types that are being deployed.

Overhead of PMI

The actual overhead of each statistic level varies depending on the particular applications on the server and load that is being executed by the server. In the WebSphere Information Center, each counter has a documented qualitative overhead level to indicate the type of overhead it will incur (see “Getting more information about statistics sets” on page 558). This is not intended to prevent administrators from using counters with high overhead. It is important to remember that the overhead is a relative measurement, and the administrator needs to balance the need for the data versus the overhead incurred to enable a particular counter temporarily or for the long term.

Note: If running in a stand alone server configuration, it is important to remember that the data collection and statistical counters are all working in the same JVM as the applications. In this circumstance, the administrator should anticipate that the activation of additional PMI data can incur additional CPU and memory usage by the JVM.

The approximate overhead of the PMI statistic sets are as follows:

- ▶ Basic overhead up to 2%
- ▶ Extended overhead up to 3%
- ▶ All overhead of up to 6%
- ▶ Custom will depend on the counters enabled but it is reasonable to expect somewhere between 2%-6%

Getting more information about statistics sets

The WebSphere Information Center has extensive information to assist the administrator in understanding exactly which metrics are set for a particular level, and to appreciate the potential overheads of using the statistics.

If we consider the number of components that make up an application server as shown in Figure 11-7, it should be no surprise that there are many PMI counters available to help monitor the application server.

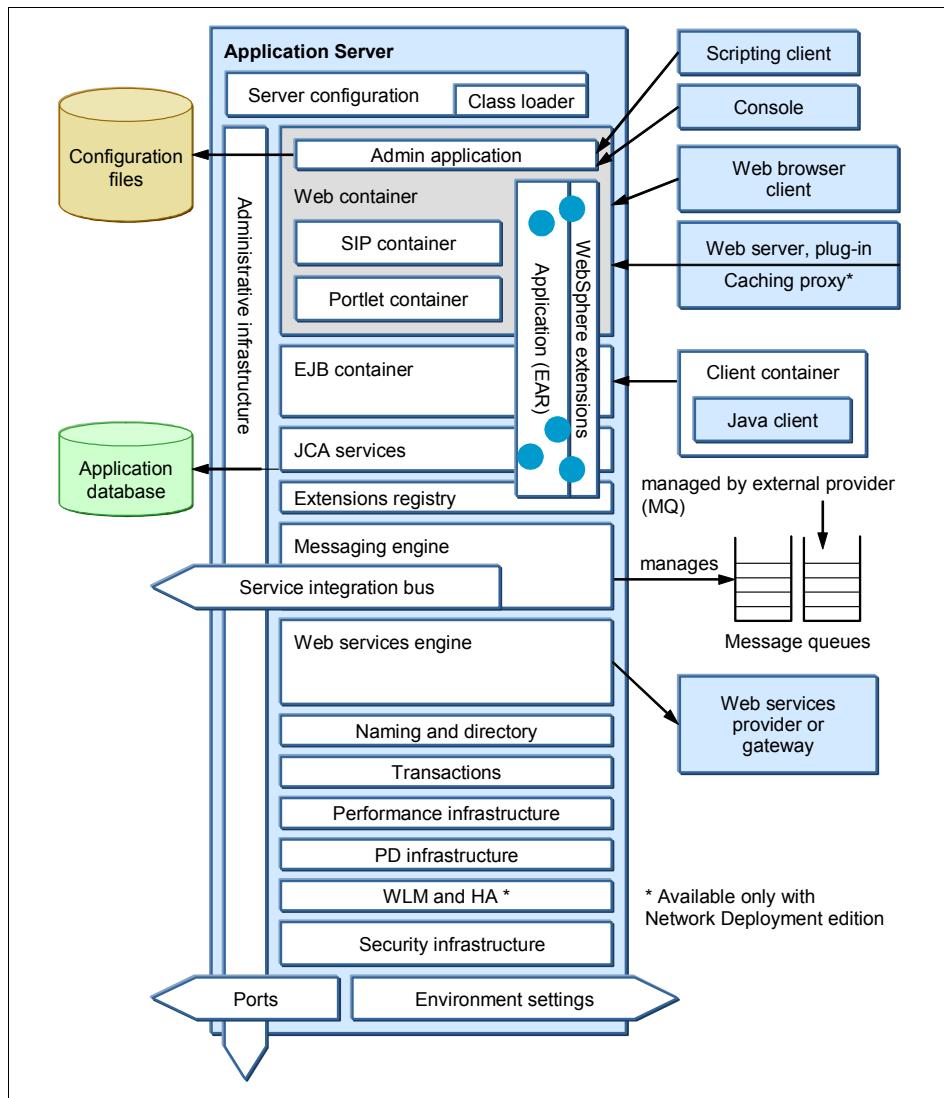
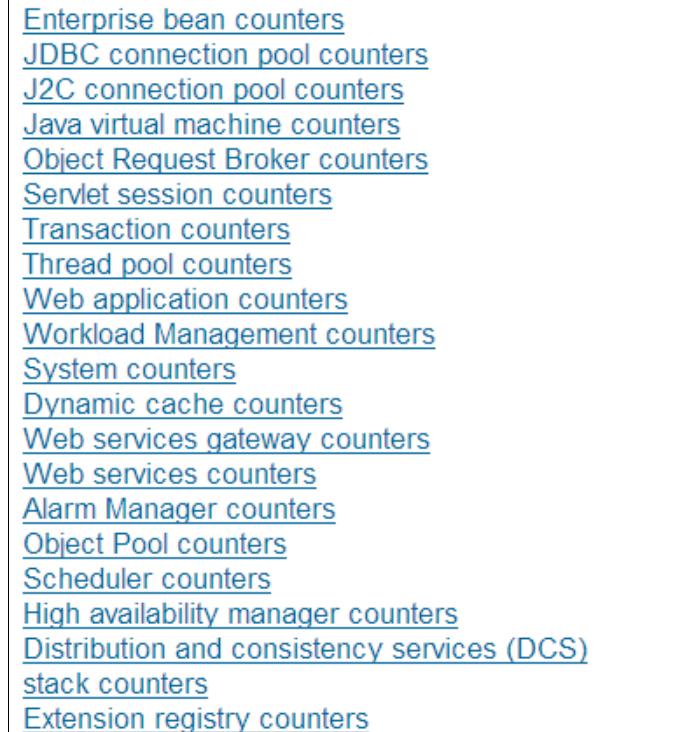


Figure 11-7 Application server components

The WebSphere Information Center provides a summary of PMI counters to help administrators understand the variety of counters that are available in each of the different counter classifications in the application server. A good place to start with gathering information is the article, *Enabling PMI data collection*, at:

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

Figure 11-8 shows the links found at the bottom of this article, taking you to a page with more information about each counter.



The following list contains links to various PMI counter types:

- [Enterprise bean counters](#)
- [JDBC connection pool counters](#)
- [J2C connection pool counters](#)
- [Java virtual machine counters](#)
- [Object Request Broker counters](#)
- [Servlet session counters](#)
- [Transaction counters](#)
- [Thread pool counters](#)
- [Web application counters](#)
- [Workload Management counters](#)
- [System counters](#)
- [Dynamic cache counters](#)
- [Web services gateway counters](#)
- [Web services counters](#)
- [Alarm Manager counters](#)
- [Object Pool counters](#)
- [Scheduler counters](#)
- [High availability manager counters](#)
- [Distribution and consistency services \(DCS\) stack counters](#)
- [Extension registry counters](#)

Figure 11-8 PMI counter types

From the links to the counter classifications, it is possible to navigate and view the individual counters that each counter classification contains.

The following topics in the information center can provide more information:

- ▶ General PMI data organization:
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/rprf_dataorg.html
- ▶ WebSphere Application Server supports the Eclipse framework for extensible applications. A key part of this framework is the implementation of the Extension registry. These counters are only relevant when referring to extensible applications. For more information, see:
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.base.iseries.doc/info/iseries/ae/cweb_extensions.html

- Service integration bus counters:
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/rprf_sibcounter.html

11.2.2 Enable request metrics

The enabling of request metrics is a cell wide configuration and when activated, it is activated for all servers in the cell. In the administrative console:

1. Navigate to the **Request Metrics** menu item in the **Monitoring and Tuning** navigation menu (Figure 11-9).

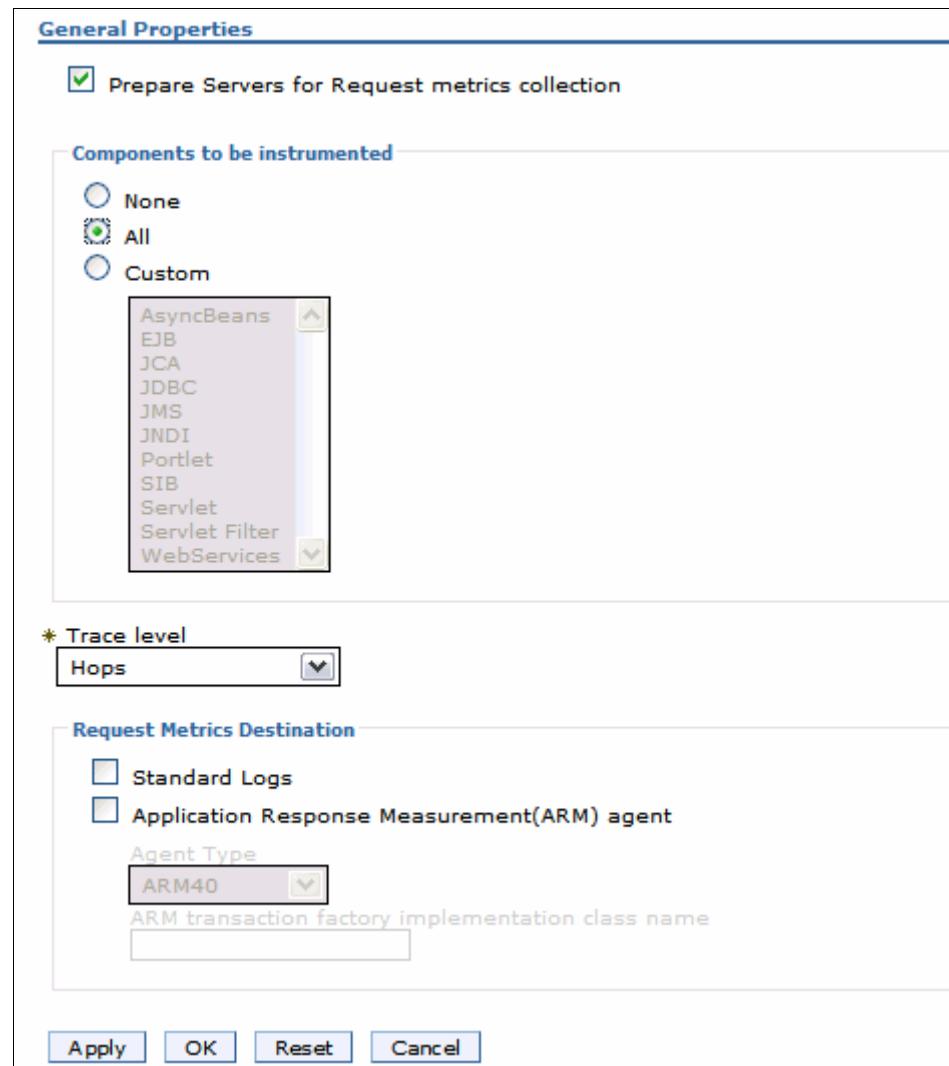


Figure 11-9 Request metrics panel

Request metrics are enabled by:

- a. Checking the **Prepare servers for request metrics collection**
- b. Choosing a monitoring level from the **Components to be instrumented** section of the panel.
- c. Choosing a trace level, and
- d. Choosing a destination from the **Request Metrics destination** section of the **Request Metrics** panel.

When configured, the servers must be restarted for request metrics to be enabled. The servers must also be stopped when disabling request metrics.

Understanding component instrumentation and trace levels

Trace levels and component instrumentation work together to determine if the request is instrumented. The component instrumentation levels are shown in Figure 11-10.

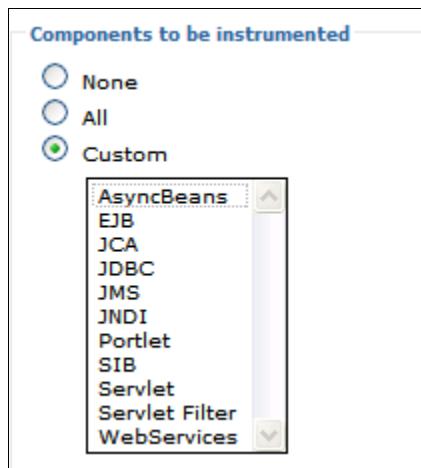


Figure 11-10 Components to be instrumented

If you select **All**, all components will be monitored based on trace level settings.

If you select **Custom**, you can select the components to be monitored. Data will be collected from the components if the trace level also calls for the capturing of data from this component.

Note: When a component is defined as an edge component, meaning the request enters or exits the application server through the component, then this component is instrumented even if it is not selected as part of the custom component listing.

Working in conjunction with the component instrumentation levels is the trace level (Figure 11-11).

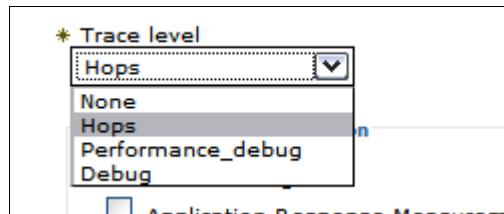


Figure 11-11 Request metric trace levels

The following trace levels are possible:

- ▶ None: No instrumentation is generated.
- ▶ Hops: Generates instrumentation information about process boundaries only. When this setting is selected, you see the data at the application server level, not the level of individual components such as enterprise beans or servlets.
- ▶ Performance_debug: Generates the data at Hops level and the first level of the intra-process servlet and Enterprise JavaBeans (EJB) call (for example, when an inbound servlet forwards to a servlet and an inbound EJB calls another EJB). Other intra-process calls like naming and service integration bus (SIB) are not enabled at this level.
- ▶ Debug: Provides detailed instrumentation data, including response times for all intra-process calls. Note that requests to servlet filters will only be instrumented at this level.

Note: Working with instrumentation and trace levels are further explored later in the chapter (see “Request level details” on page 587)

Important: Request metrics are checked starting with the HTTP plug-in for some Web related settings. The HTTP-plug-in configuration must be regenerated and propagated after enabling request metrics.

Using request metric filters

One final way that can be used to control the request metric instrumentation is to use request metric filters. Filters provide a way to specifically target flows and components to reduce the overhead of broad monitoring and to also make it easier to analyze the captured data by reducing the amount data that is captured.

It is important to understand, however, that filters are implemented as edge component filtering, not as intra-component processing, so an EJB filter will not be effective if the EJB is always invoked from a servlet. In this case it is the URI that needs filtering, not the EJB. Filters should be applied on edge components.

Filters are configured by selecting the **filters** link from the Additional properties section of the **Request metrics** panel. This navigates the administrator to the Request metric filter panel shown in Figure 11-12.

The screenshot shows the 'Request Metrics > Request Metrics Filter' panel. It displays a table of filter types and their enable status. The columns are 'Type' and 'Enable'. The rows include EJB, JMS, SOURCE IP, URI, and WEB SERVICES, all set to false. A summary at the bottom indicates a total of 5 filters.

Type	Enable
EJB	false
JMS	false
SOURCE IP	false
URI	false
WEB SERVICES	false
Total 5	

Figure 11-12 Request Metric Filter panel

Using filters, fine grained controls can be applied to the different edge types of EJB, JMS, IP address, URI, and Web services. The first step is to specify the filter. An example of each type can be seen by navigating the administration console for that type of filter. For example, selecting the URI link in Figure 11-12 takes you to the URI panel shown in Figure 11-13.

The screenshot shows the 'Uri' panel. It has two main sections: 'General Properties' and 'Additional Properties'. Under General Properties, there is a dropdown for 'Type' set to 'URI', an 'Enable' checkbox, and buttons for 'Apply', 'OK', 'Reset', and 'Cancel'. Under Additional Properties, there is a link for 'Filter Values'.

Figure 11-13 Uri panel

The enable check box must be checked for the filters to be enabled. Then the filters will be used along with the component and trace level settings to determine which components are instrumented.

Note: Enabling filters requires an application server restart.

Select the **Filter Values** link in the filter panel to add or edit filters. This is also where the default example filters are displayed (Figure 11-14).

Request Metrics > Request Metrics Filter > URI > Filter Values

Specifies the value of request metrics filter and enablement for the filter type.

Preferences

New	Delete
<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>

Select	Value 	Enable
<input type="checkbox"/>	/hitcount	false
<input type="checkbox"/>	/snoop	false

Total 2

Figure 11-14 Default filter values displayed in filter value panel

Each filter type has its own syntax that is appropriate for the type. For example, the EJB filter specifies a method class or package that sets the scope of the filtering. Figure 11-15 shows the example URI filer value supplied for EJB filters.

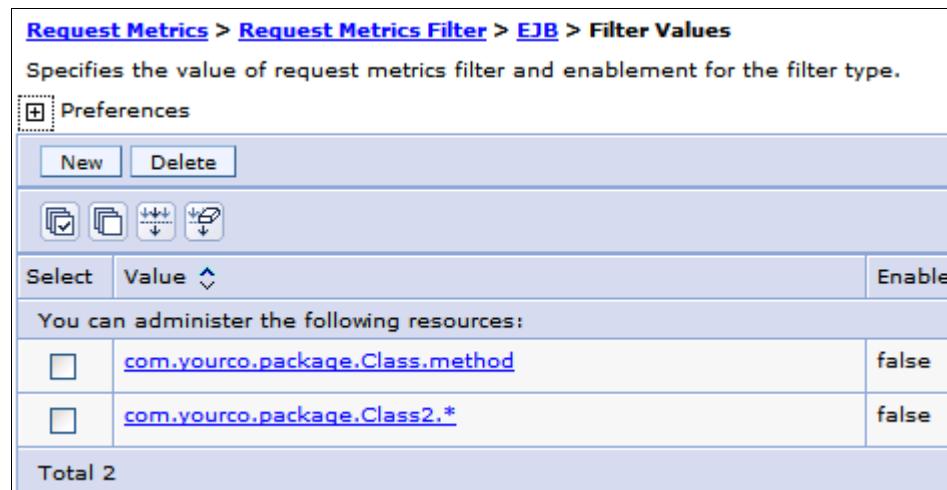


Figure 11-15 EJB default filter values

It is possible to use wildcard settings in the filters if desired, as shown in the second entry in Figure 11-15 on page 566.

Tip: To enable/disable a filter it requires the restarting of a server. It is important to plan the component levels and filters that an application might require to minimize the need to stop and restart servers.

Destination type considerations

The final consideration when configuring the request metrics is where the metrics will be gathered. There are two types of supported destinations,

- ▶ Data can be logged with standard logs. In this configuration the instrumented components are logged to the SystemOut.log file.
- ▶ Data can also be collated in an Application Response Measurement (ARM) data collector. In this case, the data is normally then moved to a monitoring system for analysis and display (for example, using IBM Tivoli Monitoring Transaction Performance).

Tip: When configuring ARM agents for use with the application server, follow the installation instructions provided with the specific agent. For more information about ARM agents, see:

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

Both logging types can be activated at once. Writing to standard logs is not recommended as a long term monitoring strategy because the overhead can be higher than is desirable.

Overhead of request metrics

The overhead of request metrics can vary significantly based on the components being monitored and the complexity of the request execution within the monitored components. There are no specific metrics on what the overhead is, but it is reasonable to assume that request metrics on every request and component might incur more overhead than is desired. We recommend that an organization consider and plan carefully the interactions that it wants to monitor, then measure the specific overhead associated with configuring request metrics for these components.

11.3 Viewing the monitoring data

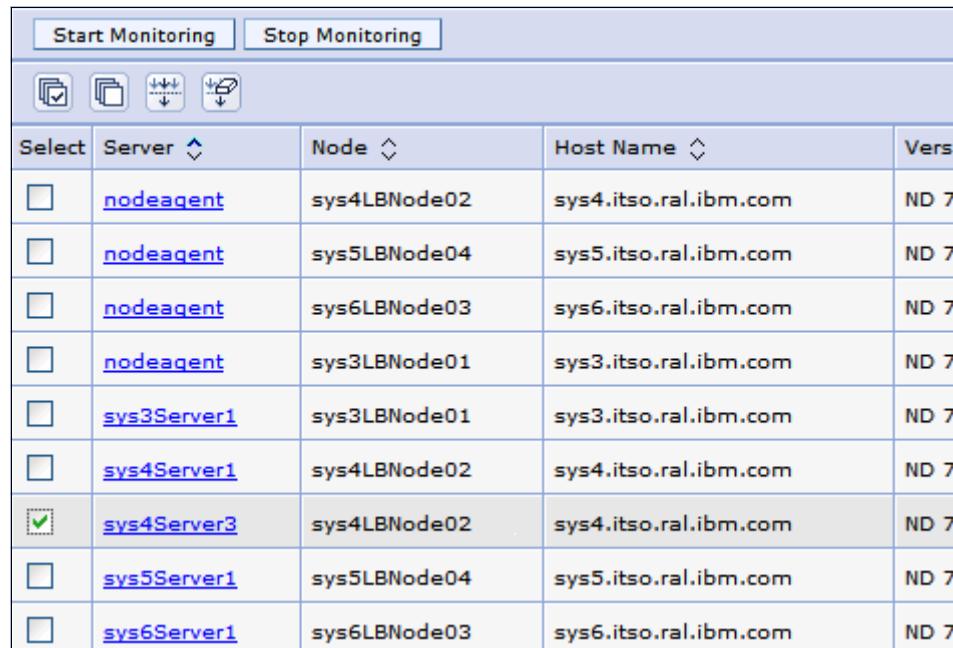
WebSphere Application Server provides an interface for viewing the monitored data. The interface is the Tivoli Performance Viewer (TPV) found in the administrative console.

Starting TPV monitoring and configuring settings

To work with the TPV from the administrative console, follow these steps:

1. Navigate to the Tivoli Performance Viewer panel, by selecting **Monitoring and Tuning** → **Performance Viewer** → **Current Activity**.
2. Select the check-box of server(s) that are to be monitored and click the **Start Monitoring** button (Figure 11-16).

Tip: If you are only starting monitoring on a single server, monitoring can be started by simply clicking the server link and navigating into the TPV viewer.



The screenshot shows a user interface for monitoring servers. At the top, there are two buttons: "Start Monitoring" and "Stop Monitoring". Below these are four small icons: a checkbox, a folder, an up arrow, and a down arrow. The main area is a table with columns: "Select", "Server", "Node", "Host Name", and "Version". There are nine rows of data. In the "Select" column, the first seven rows have empty checkboxes, while the eighth row has a checked checkbox. The "Server" column contains links: "nodeagent", "nodeagent", "nodeagent", "nodeagent", "sys3Server1", "sys4Server1", "sys4Server3", "sys5Server1", and "sys6Server1". The "Node" column lists nodes: "sys4LBNODE02", "sys5LBNODE04", "sys6LBNODE03", "sys3LBNODE01", "sys3LBNODE01", "sys4LBNODE02", "sys4LBNODE02", "sys5LBNODE04", and "sys6LBNODE03". The "Host Name" column lists host names: "sys4.itso.ral.ibm.com", "sys5.itso.ral.ibm.com", "sys6.itso.ral.ibm.com", "sys3.itso.ral.ibm.com", "sys3.itso.ral.ibm.com", "sys4.itso.ral.ibm.com", "sys4.itso.ral.ibm.com", "sys5.itso.ral.ibm.com", and "sys6.itso.ral.ibm.com". The "Version" column shows "ND 7" for all rows.

Select	Server	Node	Host Name	Version
<input type="checkbox"/>	nodeagent	sys4LBNODE02	sys4.itso.ral.ibm.com	ND 7
<input type="checkbox"/>	nodeagent	sys5LBNODE04	sys5.itso.ral.ibm.com	ND 7
<input type="checkbox"/>	nodeagent	sys6LBNODE03	sys6.itso.ral.ibm.com	ND 7
<input type="checkbox"/>	nodeagent	sys3LBNODE01	sys3.itso.ral.ibm.com	ND 7
<input type="checkbox"/>	sys3Server1	sys3LBNODE01	sys3.itso.ral.ibm.com	ND 7
<input type="checkbox"/>	sys4Server1	sys4LBNODE02	sys4.itso.ral.ibm.com	ND 7
<input checked="" type="checkbox"/>	sys4Server3	sys4LBNODE02	sys4.itso.ral.ibm.com	ND 7
<input type="checkbox"/>	sys5Server1	sys5LBNODE04	sys5.itso.ral.ibm.com	ND 7
<input type="checkbox"/>	sys6Server1	sys6LBNODE03	sys6.itso.ral.ibm.com	ND 7

Figure 11-16 Start Server monitoring

After monitoring is started, a message will be returned in the messages section of the panel, and the Status column of the server will be updated to Monitored.

3. The PMI data can only be observed one server at a time when using a single user session. Select the server name link to navigate to the Tivoli Performance Viewer panel (Figure 11-17).

The screenshot shows the Tivoli Performance Viewer interface. At the top, it says "Tivoli Performance Viewer > sys4Server3". Below that is a message: "Use this page to view and refresh performance data for the selected server, change user and log set modules." On the left, there's a navigation tree with "sys4Server3" expanded, showing "Advisor", "Settings", "Summary Reports", and "Performance Modules". There are "Refresh" and "View Module(s)" buttons. A "Deselect All" button is also present. On the right, there's a "Servlets Summary Report" section with a "Start Logging" button and a table of recent servlet activity:

Name	Application	Total R
/account.jsp	Trade#tradeWeb.war	251
/displayQuote.jsp	Trade#tradeWeb.war	3,463
/marketSummary.jsp	Trade#tradeWeb.war	425

Figure 11-17 Tivoli Performance Viewer

The default view for this panel shows the Servlet Summary Report panel indicating recent servlet activity, as well as the TPV tree navigation panel.

Note: The different ways that data can be viewed is discussed in “Exploring TPV data views” on page 572. But before exploring the data, let us first take a look at the settings menu to examine the User and Logging settings.

4. The user settings are reached by expanding the Settings category and selecting **User**. This panel helps you control how much data is retained and how often data samples are taken in the live system (Figure 11-18).

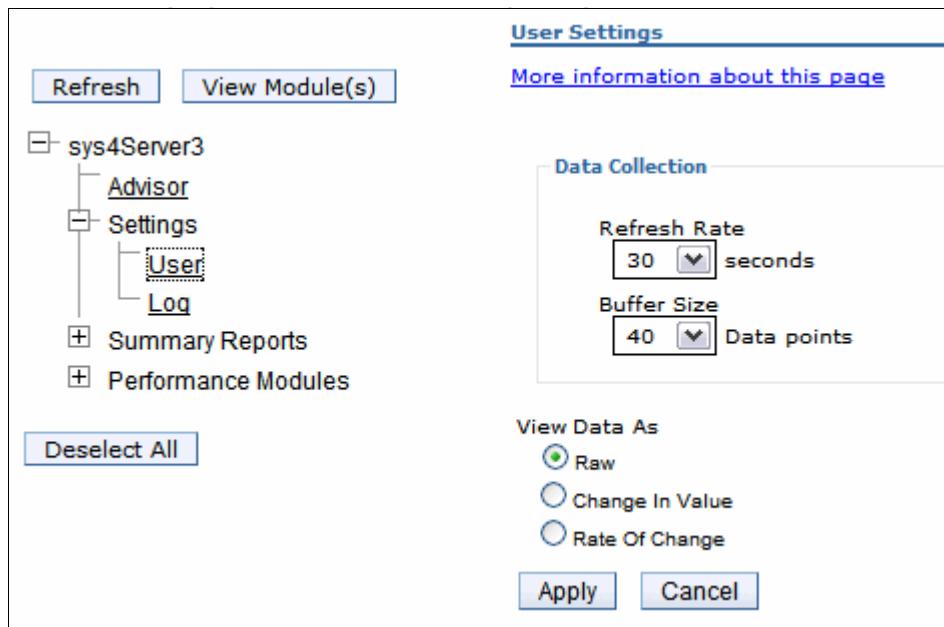


Figure 11-18 User settings

The user settings are very significant and can have a direct impact on the performance of the server. The two key configuration settings are in the Data Collection section of the panel:

- The refresh rate indicates the interval between data sampling. Higher frequency rates mean that the server will gather and report on statistics more frequently, adding load to the servers that are collecting data.
- The buffer size indicates the number of data points that are kept. More data points simply mean that the PMI data will require more memory.

For a standalone server this means that PMI data at high frequency and high buffer re-initiation will need more processing time and more memory.

In a distributed server model the load is shared, but some settings might still need to be tuned. The data is collected at the node level and stored in memory on the node agent. Thus if a node has many servers, the memory requirements of the node agent will need to be adjusted.

Also in a distributed server configuration, the data is viewed from the deployment manager. Thus, to process the data, the deployment manager should have adequate memory and CPU resources also. Consider also that more than one administrator might be observing data at once.

5. A powerful feature of TPV is the ability to record PMI data and then replay the data later in a different deployment manager as if it were in real time or with options to fast forward and rewind.

Logging is started by simply clicking the **Start Logging** button shown in the TPV viewing panel (Figure 11-17 on page 569). However, before clicking this button, the Log settings must be configured. The Log settings are found by selecting **Settings** → **Log** as shown in Figure 11-19.

The screenshot shows the 'Tivoli Performance Viewer > sys4Server3' interface. At the top, there are 'Refresh' and 'View Module(s)' buttons, and a link to 'More information about this page'. On the left, a tree view shows 'sys4Server3' expanded, with 'Advisors', 'Settings' (which has 'User' and 'Log' children), 'Summary Reports', and 'Performance Modules'. Below the tree is a 'Deselect All' button. On the right, under 'Logging Settings', there is a 'Logging Output' section with the following configuration:

* Duration	20	minutes
* Maximum File Size	5	MB
* Maximum Number of Historical Files	3	
* File Name	tpv	

Below this is a 'Log Output Format' dropdown set to 'XML'. At the bottom are 'Apply' and 'Cancel' buttons.

Figure 11-19 Log Settings

Examining the log settings that are available, the administrator is faced with several configuration choices:

- Duration:

The logging of PMI data has with it a certain amount of overhead (resulting from logging to a file, the buffering of data in memory, and disk usage for log storage...). It is not intended to be used as a long term production monitoring strategy. Thus when logging is enabled, it is configured to be disabled after a period of time.

PMI data logging used in short durations can be used to capture runtime characteristics that might need further investigation or sharing with development and troubleshooting specialists.

- Maximum file size and Maximum number of historical files:

The settings for maximum file size and the number kept that are appropriate for your environment will depend on two conditions:

- How much PMI data is enabled
- The data sampling frequency (as configured in the user settings).

- File name:

The server name and the time at which the log is started is appended to the file name specified to help users identify a log file

- Log output format:

The other configuration item of consequence is the format type. The default is XML but the binary format requires a smaller footprint on the disk. If logging larger amounts of data, the binary logging format might be more suitable.

Exploring TPV data views

Tivoli Performance Viewer has three primary types of data that can be viewed

- ▶ Summary Reports
- ▶ Performance Modules
- ▶ Advisors

Summary reports

The summary reports provide a general overview in a tabulated format of the current system performance. The reports that are available include:

- ▶ Servlets
- ▶ EJBs
- ▶ EJB Methods
- ▶ Connection Pool
- ▶ Thread Pool

Servlet and EJB summary reports can be useful for identifying the application resources that are most busy, and to the extent that averages can be used, to identify candidates that might warrant further investigation as to their performance. Together with request metrics, results from the summary reports can help identify possible candidates for request metric filters.

The information for Connection Pool and Thread Pool utilization, while useful, can also be easily determined by monitoring the metrics in the performance modules.

Note: For summary reports to be available, the PMI data must be reported at a sufficiently detailed level. For the basic PMI data level, only the Servlets and EJBs reports are available. Higher or custom PMI settings must be specified for the other reports.

Figure 11-20 shows an example of the Servlets report.

Start Logging				
Name	Application	Total Requests	Avg Resp Time (ms)	Total Time
/account.jsp	Trade#tradeWeb.war	1,916	0.113	217
/displayQuote.jsp	Trade#tradeWeb.war	23,547	42.717	1,005,859
/marketSummary.jsp	Trade#tradeWeb.war	2,969	47.34	140,553
/order.jsp	Trade#tradeWeb.war	836	0.074	62
/portfolio.jsp	Trade#tradeWeb.war	1,758	0.17	298
/quote.jsp	Trade#tradeWeb.war	4,740	212.837	1,008,849
/tradehome.jsp	Trade#tradeWeb.war	2,969	47.535	141,131
/welcome.jsp	Trade#tradeWeb.war	693	0.045	31
FilterProxyServlet	Trade#tradeWeb.war	3	0	0
TradeAppServlet	Trade#tradeWeb.war	12,852	191.795	2,464,948
TradeScenarioServlet	Trade#tradeWeb.war	11,203	219.079	2,454,339
rsp servlet	ibmasyncrsp.war	0	0	0
Total 12				

Figure 11-20 Servlets summary report

Figure 11-21 shows an example of the EJBs summary report.

Start Logging				
Name	Application	Method Calls	Avg Resp Time (ms)	Total
AccountEJB	Trade#tradeEJB.jar	25,269	4.11	103,889
AccountProfileEJB	Trade#tradeEJB.jar	34,051	13.867	472,153
HoldingEJB	Trade#tradeEJB.jar	26,717	13.48	360,111
KeyGenEJB	Trade#tradeEJB.jar	6	10.5	63
KeySequenceEJB	Trade#tradeEJB.jar	1,632	0.067	110
OrderEJB	Trade#tradeEJB.jar	18,281	1.262	23,063
QuoteEJB	Trade#tradeEJB.jar	1,056,786	1.114	1,177,104
TradeBrokerMDB	Trade#tradeEJB.jar	0	0	0
TradeEJB	Trade#tradeEJB.jar	51,367	51.667	2,653,449
TradeStreamerMDB	Trade#tradeEJB.jar	0	0	0
Total 10				

Figure 11-21 EJB summary

Tip: This summary data can be used to identify EJBs that get invoked very often. If the system is working correctly, the slowest bean on average might be worth investigating if the time exceeds expected SLAs, and so on.

It can also be useful to identify what work is not happening. In the preceding snapshot, there are two message-driven beans that have not been invoked. This, in itself, might be unusual and warrant investigation.

Performance modules

Performance modules provide a tracking mechanism for each of the PMI counters that are active. These counters are categorized under their different PMI data classifications. The data can be viewed as a table or graphically displayed using the embedded Adobe® SVG viewer in a graphical display.

The performance modules provide a powerful runtime view of the data as it is being recorded to allow the administrator to analyze the current system health.

The data that can be displayed is limited depending on the PMI level that has been configured. The administrator can select one or more metrics for the current PMI level as shown in Figure 11-22, and then select the **View Modules** button at the top of the panel.

The screenshot shows the 'Performance Modules' interface. On the left, a tree view lists various performance modules: DCS Statistics, ExtensionRegistryStats.name, SIB Service, SipContainerModule, Enterprise Beans, Dynamic Caching, JDBC Connection Pools (with DB2 Using IBM JCC Drive expanded, showing jdbc/trade-ds and jdbc/trade-me-ds selected), Derby JDBC Provider (XA), Microsoft SQL Server JDE, Oracle JDBC Driver (XA), HAManager, JCA Connection Pools, JVM Runtime (selected), Object Pool, ORB, pmiWebServiceModule, Servlet Session Manager, and Thread Pools (Default, HAManager.thread.pool, Message Listener, Object Request Broker, ProcessDiscovery).

On the right, the 'Servlets Summary Report' window is open. It contains a table with the following data:

Name	Application	Total Requests
/account.jsp	Trade#tradeWeb.war	5,267
/displayQuote.jsp	Trade#tradeWeb.war	67,475
/marketSummary.jsp	Trade#tradeWeb.war	7,846
/order.jsp	Trade#tradeWeb.war	2,373
/portfolio.jsp	Trade#tradeWeb.war	4,917
/quote.jsp	Trade#tradeWeb.war	13,481
/tradehome.jsp	Trade#tradeWeb.war	7,846
/welcome.jsp	Trade#tradeWeb.war	1,819
FilterProxyServlet	Trade#tradeWeb.war	3
TradeAppServlet	Trade#tradeWeb.war	35,518
TradeScenarioServlet	Trade#tradeWeb.war	31,023
rsp servlet	ibmasyncrsp.war	0

Total 12

Figure 11-22 Performance modules

After the performance modules are selected, the data panel provides a view of the data that is being collected. By default, the graphical view is used. Each most recent data point and the graph key for the different counters can be seen under the graph. The graph can then be customized to include or exclude counters from the chosen set of PMI data. Figure 11-23 shows an example of the PMI data displayed in the example environment.

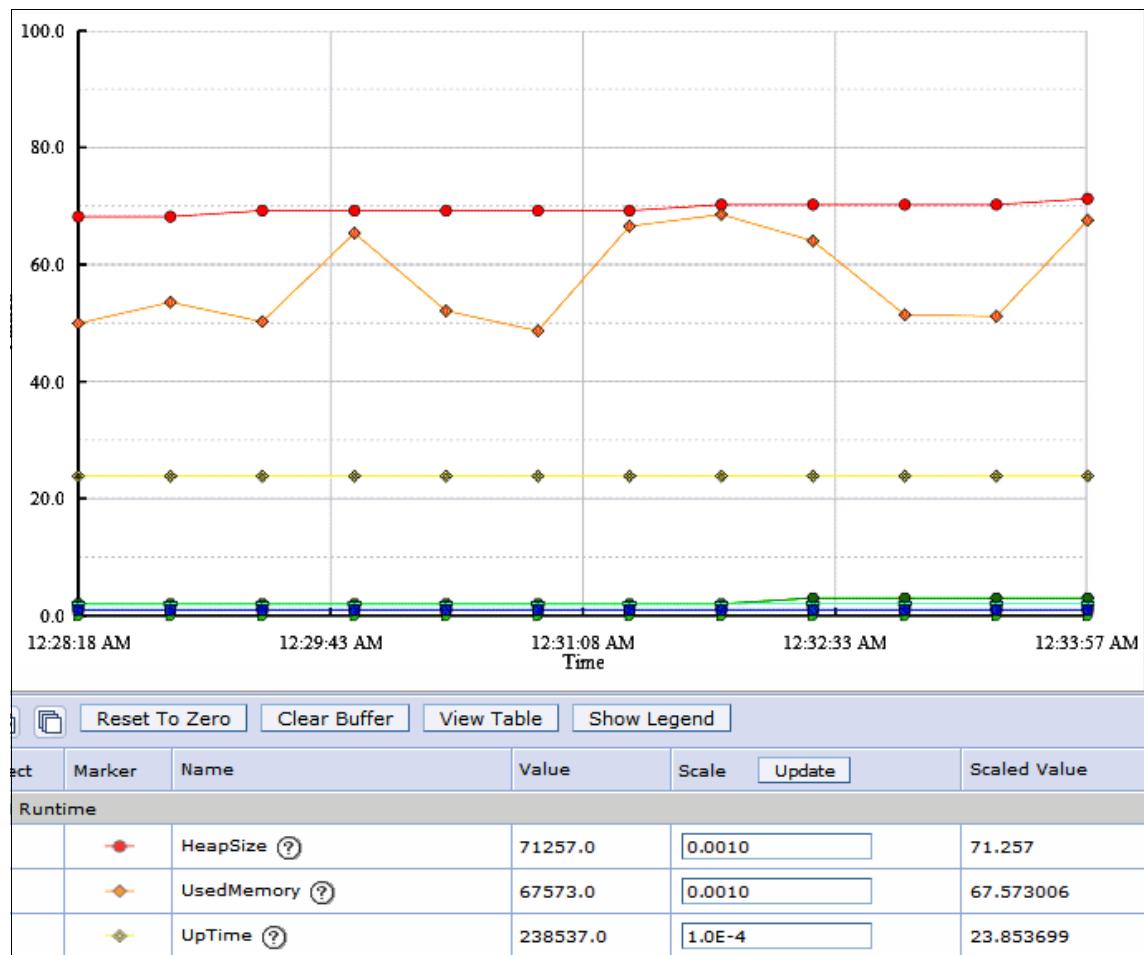


Figure 11-23 View modules, graphical data view

To view the data in table format, click the **View table** button. Scales can also be adjusted by changing the scale and clicking the **Update** button.

Performance advisors

The last of the TPV data sets contains the TPV performance advisors. These advisors analyze the data using rules that are pre-configured by IBM based on recommended practice and performance observations. The advisors provide tuning recommendations to help improve the performance.

The types of items that TPV will provide advice on include several well known performance hot spots:

- ▶ Object Request Broker service thread pools
- ▶ Web container thread pools
- ▶ Connection pool size
- ▶ Persisted session size and time
- ▶ Data source statement cache size
- ▶ Session cache size
- ▶ Dynamic cache size
- ▶ Java virtual machine heap size
- ▶ DB2 Performance Configuration wizard
- ▶ Connection use violations

Advisors are more of a tuning aid than a monitoring tool set and are not suitable for use in production environments. But advisors can be a useful aid in identifying well known performance hotspots in the current server configurations in testing.

Advisors are best used when:

- ▶ A reasonable load can be driven to the application server utilizing significant CPU (50+%).
- ▶ You want help in tuning a server while establishing initial performance benchmarks

For more information about the advisors, see:

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

To view the Advisors panel, click the **Advisor** link in the TPV menu panel. Figure 11-24 shows an example of what you will see.

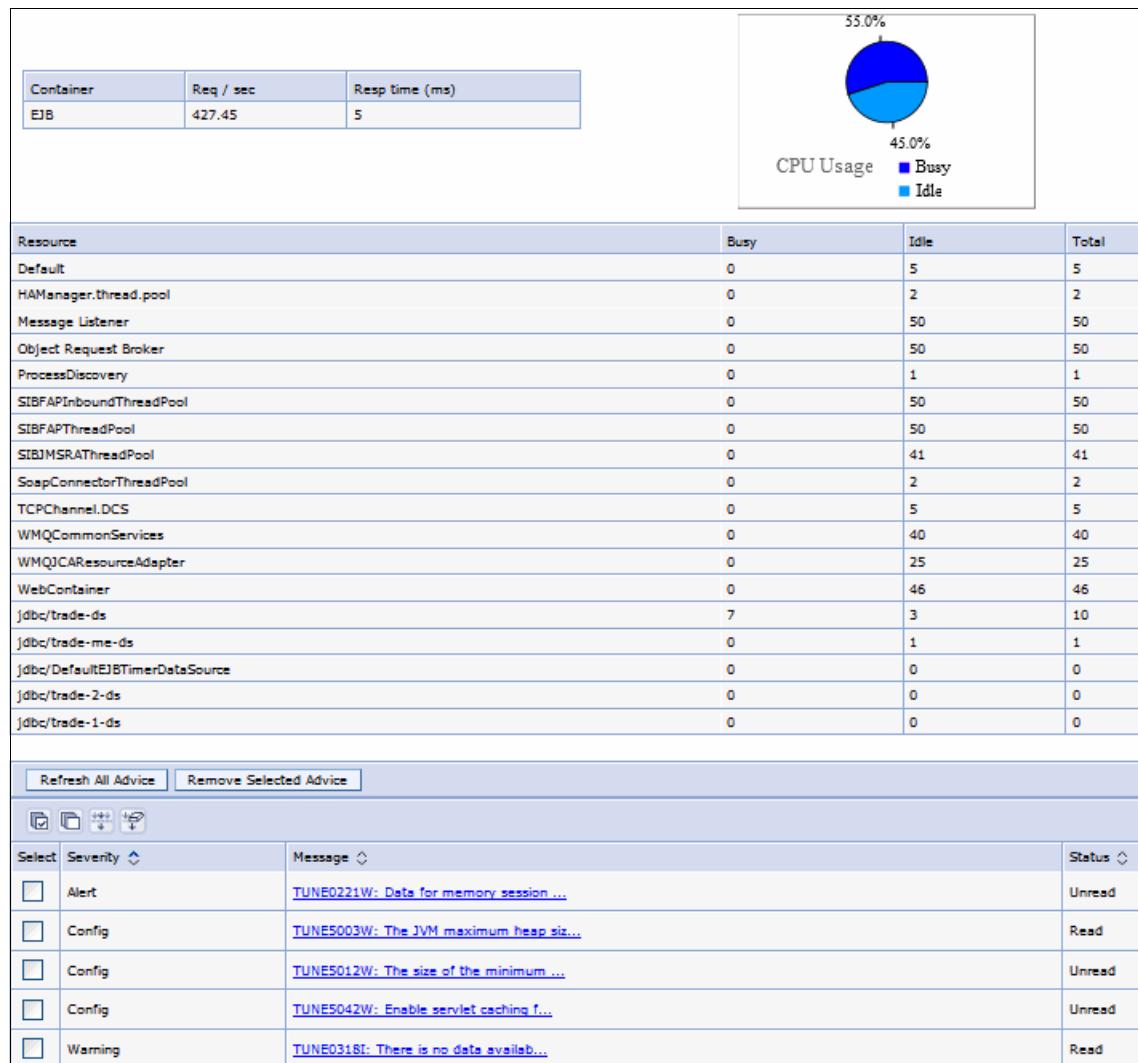


Figure 11-24 Advisors view

From within the advisors view, the different advice statements can be selected and further analyzed by the administrator. Figure 11-25 shows an example of an advice panel.

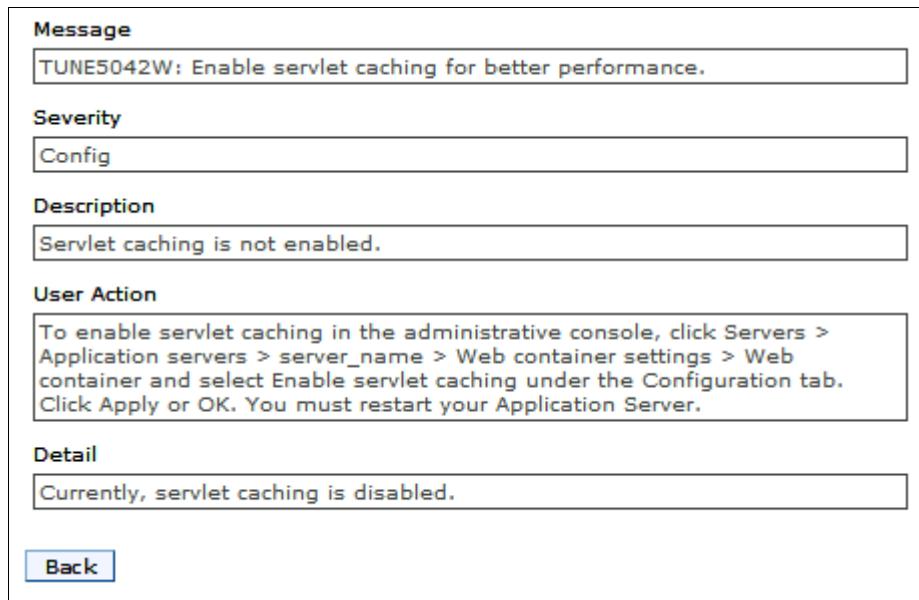


Figure 11-25 Advice panel

The Advice panel provides a clear description and recommendation to aid the administrator in quickly addressing well understood performance considerations.

Note: Advisors are not further used in this chapter. Our focus in this chapter is monitoring, and advisors really fall into a tuning toolset more than monitoring.

11.4 Monitoring scenarios

In this section, a variety of monitoring data will be used to provide examples and information about what that data tells an administrator.

Important: While the statistical data that is observed in PMI and request metrics provides a powerful understanding of what is occurring in the environment, it is very important to remember that the counters and statistics are mostly averages. You must always verify that the statistics make sense for your environment. For example, if an EJB seems to be having good response time, but it is throwing and catching an exception shortly after entering the bean, it would still appear to be having excellent response time. Response time alone is not enough to indicate a healthy system.

11.4.1 Database interactions

One of the hotspots for monitoring is interactions with external servers, especially interactions with databases. PMI provides a good set of metrics at even the basic level for these types of interactions.

Consider the data snapshot shown in Figure 11-26. This snapshot was taken in the example environment while the server was under load. The report is viewed by selecting **Monitoring and Tuning** → **Performance Viewer** → **Current Activity** → **Performance Modules** → **JDBC Connection Pools**. The monitoring level in effect was the Basic level.

Select	Marker	Name	Value	Scale	Update
jdbc/trade-ds					
<input checked="" type="checkbox"/>		CreateCount (?)	0.0	<input type="text" value="1.0"/>	
<input checked="" type="checkbox"/>		CloseCount (?)	0.0	<input type="text" value="1.0"/>	
<input checked="" type="checkbox"/>		PoolSize (?)	10.0	<input type="text" value="1.0"/>	
<input type="checkbox"/>		FreePoolSize (?)	1.0	<input type="text" value="1.0"/>	
<input type="checkbox"/>		WaitingThreadCount (?)	4.0	<input type="text" value="1.0"/>	
<input type="checkbox"/>		PercentUsed (?)	90.0	<input type="text" value="1.0"/>	
<input type="checkbox"/>		UseTime (?)	39.785263	<input type="text" value="1.0E20"/>	
<input type="checkbox"/>		WaitTime (?)	124.26131	<input type="text" value="1.0E20"/>	

Figure 11-26 Basic PMI for database

From a monitoring perspective, this provides several very useful statistics that can help the administrator understand if the database interaction is healthy. For measuring runtime health, the following counters are very beneficial:

- ▶ PercentUsed indicates if the database connections are being over utilized or under utilized.

Depending on the application and capacity of the database, it is typically not a good sign if the database is 100% utilized. If a connection pool becomes 100% utilized all of the time after the system has been tuned, either the database might need more capacity to support more connections, or some type of error is occurring, for example, the application might have a connection leak. Utilization is a good indicator that database connections need some attention.

- ▶ Similarly, it is not unreasonable to have waiting threads on the data source, because the resource is shared. From a monitoring perspective, it is a combination of the waiting thread count and the wait time that makes an interesting combination.

If the wait time and waiting thread count grow over time, this is an indication that the database might be responding more slowly than desired, or there are insufficient connections available to support the load. How long is too long a wait time depends on application service level agreements and whether wait times occur all the time or only on occasion under exceptional load.

- ▶ UseTime can help understand how much time is spent communicating with the database and can help to indicate if database response is degrading.

If the administrator believes there are response time errors with the database, request metrics can be useful in diagnosing which components the application is spending its time.

JCA interactions

JCA is the more generic form of the JDBC, and is used with standard adapters that comply with the JCA standard. As part of this, the adapter can have connection pooling the same as used with JDBC. Probably the most common JCA adapter other than JDBC is that of JMS connections, whether connecting to the service integration bus or connecting to an external JMS provider such as WebSphere MQ.

Hence, when working with JCA or JMS connections, the same indicators as used for the JDBC and as discussed in “Database interactions” on page 580 are relevant, with the exception of counters that are JDBC specific, of course. When monitoring, it is beneficial in particular to monitor the waiting threads and their wait time, and the current pool size.

11.4.2 Threading resources

Assuming that there is enough CPU and memory, while simplistic, it is reasonable to surmise that thread pools along with resource connections are a major factor in understanding the limits of throughput on the application server. The various thread pools in the application server control the entry points for requests into the system. If the pool is exhausted, then requests to the system are queued and have to wait. From a monitoring perspective, it is preferred that thread pools are not constantly exhausted and running at their maximum.

But before monitoring thread pools, the administrator needs to first be aware of what types of thread pools their application uses. The following application topologies provide some insight into what needs to be considered in understanding which thread pools might be important to application runtime throughput.

In the first example, consider a clustered application server environment where the deployed application has both Web and EJB components deployed together in the JVM (Figure 11-27). For performance reasons, WebSphere Application Server will use process affinity when invoking the EJB's, sending requests from the Web container to the EJB container in the same JVM.

In this scenario, threads are not swapped and the requests are executed on Web container threads. Even though EJBs are extensively used in the application, the ORB thread pool would not be used in this application topology. Web container threads in this topology control the concurrency of the application. This is the topology used by the sample application and is common to many JEE applications.

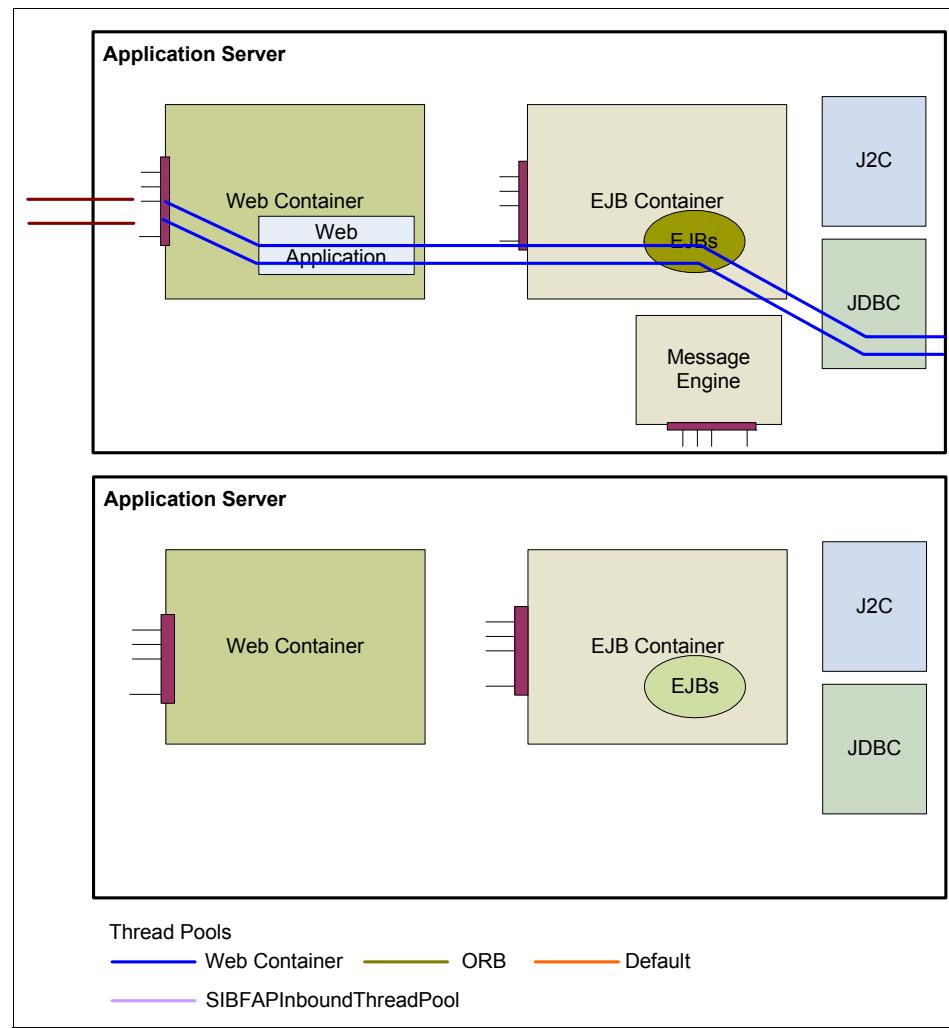


Figure 11-27 Web application

But what if the EJB components were deployed in a separate JVM (Figure 11-28)?

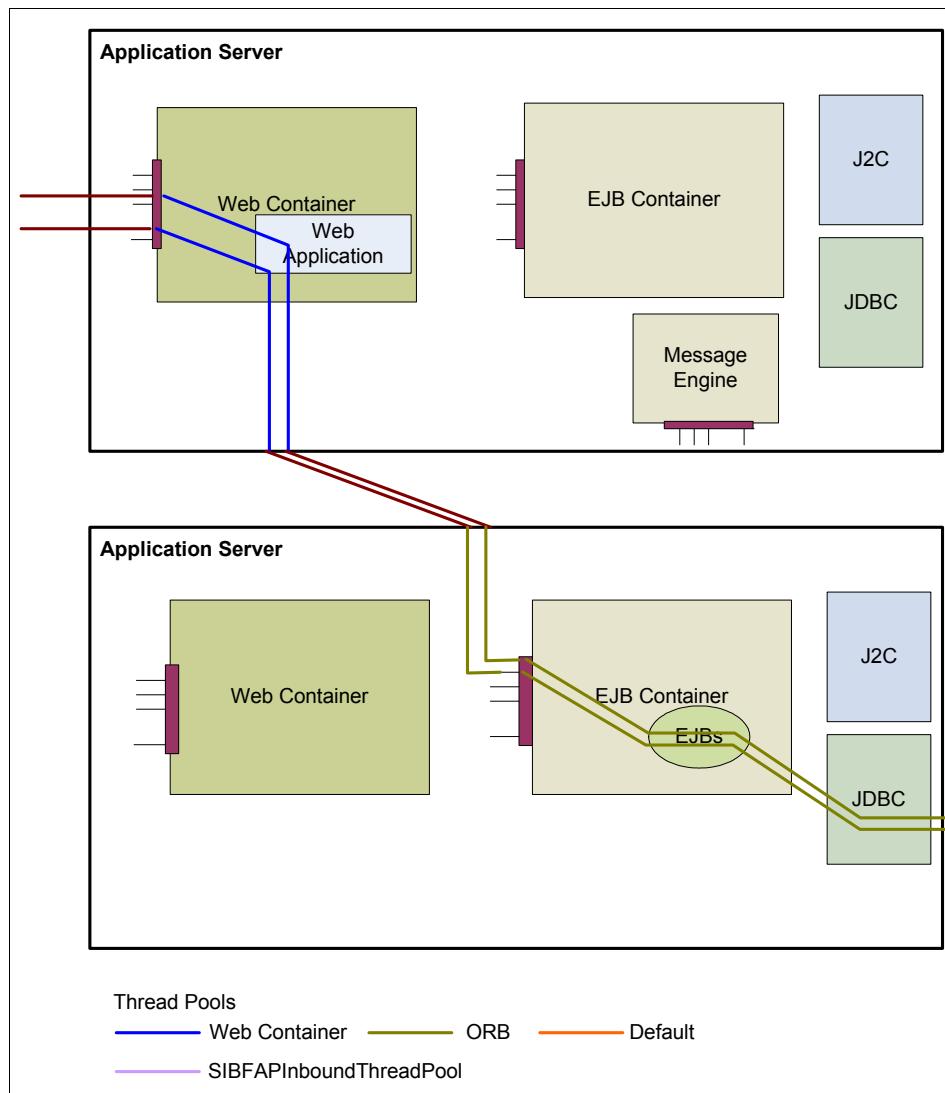


Figure 11-28 Distributed application model

In such a scenario, the ORB thread pool in each application server would become equally important as the Web container thread pool. Request throughput would now be controlled by the Web container, the ORB thread pool and other parameters such as memory and connections to external resources.

Different environment resources use different thread pools and a key consideration to understand as an administrator is what are the components that the deployed applications will interact with, and what is the likely environmental impact.

Tivoli performance viewer provides a number of resources that can assist with understanding ThreadPool utilization, and we recommend consulting the Thread Counters to learn more.

Figure 11-29 shows a graphic where the red line represents the Web container active thread pool count available when PMI is activated at the basic level. The report is viewed by selecting **Monitoring and Tuning** → **Performance Viewer** → **Current Activity** → **Performance Modules**. Expand the **Thread Pools** section and select **Web container**.

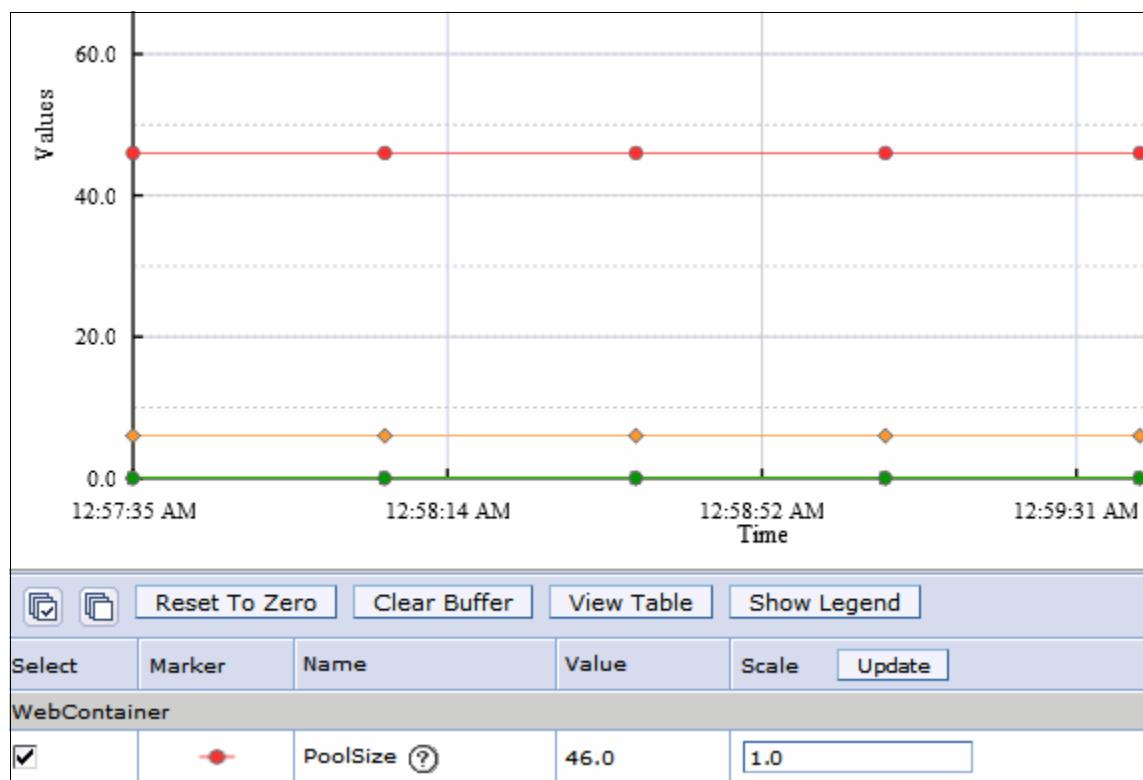


Figure 11-29 Thread Pool counters

11.4.3 JVM memory usage

Environmentally, the way the applications and application servers utilize memory is very important to the overall server performance. As a performance hotspot, it should be no surprise that the application server PMI data provides some basic level monitoring capabilities.

The JVM Used memory counter can be monitored and displayed in the TPV graph. Figure 11-30 shows an example of this data.

The report can be viewed by selecting **Monitoring and Tuning** → **Performance Viewer** → **Current Activity** → **Performance Modules**, then selecting **JVM Runtime**.

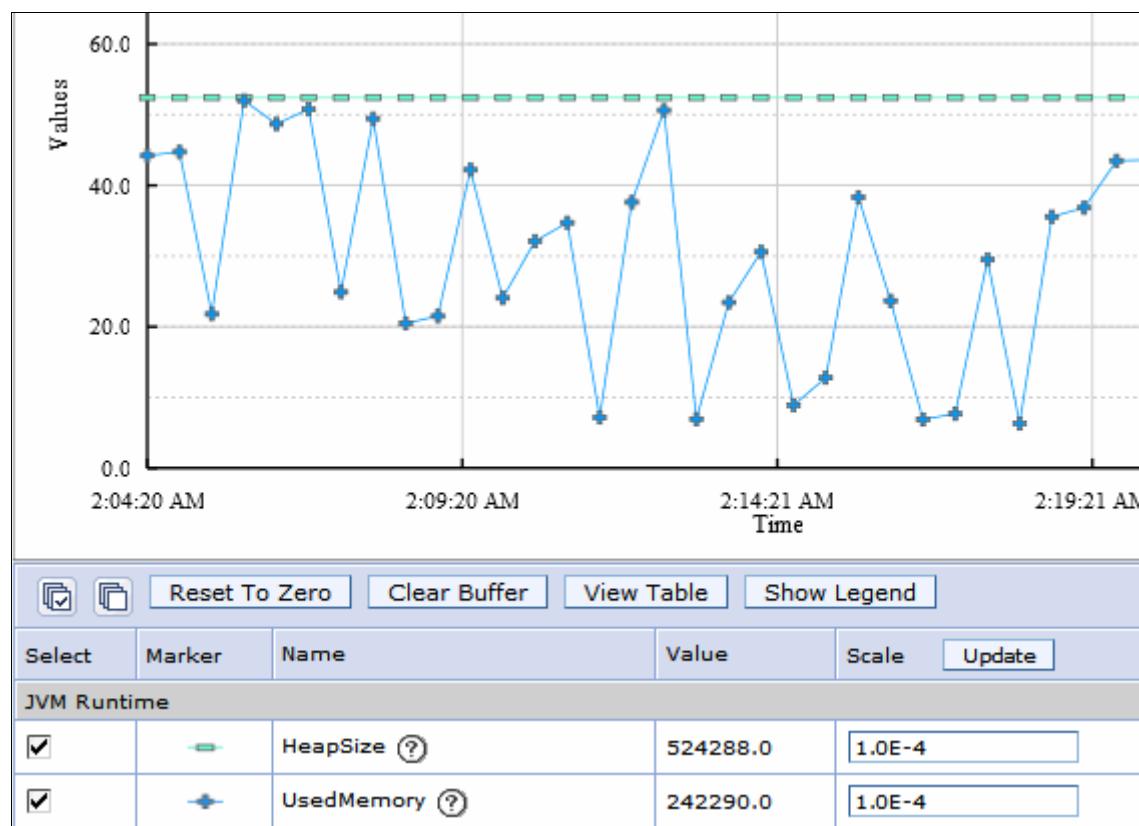


Figure 11-30 JVM memory usage

This graphical view is a nice way to visually check that memory is not constantly growing and that over time, garbage collection stabilizes it. It also gives a good indication of how frequently garbage collection might be occurring.

However, when tuning memory sizes, we recommend that verbose garbage collection and a combination of the application server support tools available in IBM Support Assistant are utilized for the analysis of the memory usage.

Note: TPV provides a good representation of what is occurring with the JVM memory, but the IBM Garbage Collection and Memory Visualizer tool and the IBM Pattern Modelling and Analysis for Java Garbage Collector tool provide more insightful help on memory related configurations to assist tuning memory. For example, you can be provided with advice on heap sizes and garbage collection algorithms based on the patterns observed in the garbage collection log.

These tools are available as add-ons to IBM Support Assistant. Use IBM Support Assistant setup wizards to download these and other Support tools. For more information, see:

- ▶ IBM Support Assistant, at:
<http://www-01.ibm.com/software/support/isa/>
- ▶ IBM Education Assistant module for The IBM Garbage Collection and Memory Visualizer, at:
http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.was_v7/was/7.0/ProblemDetermination/WASv7_GCMOverview/player.html

11.4.4 Request level details

When examining performance data, especially when tuning a server, it can be useful to have more detailed data. This is particularly true when first trying to pin down and explore bottlenecks. Request metrics can provide this lower level information, showing where time is spent in a request.

Manually creating an application server perceived response time graph (Figure 11-31) can help illustrate the response time of each component as the request is processed. The request metrics at the outer most or edge component are taken as well as all inner components, as each one handles the request. Each new component represents a narrower view excluding the work done by the components that proceed it.

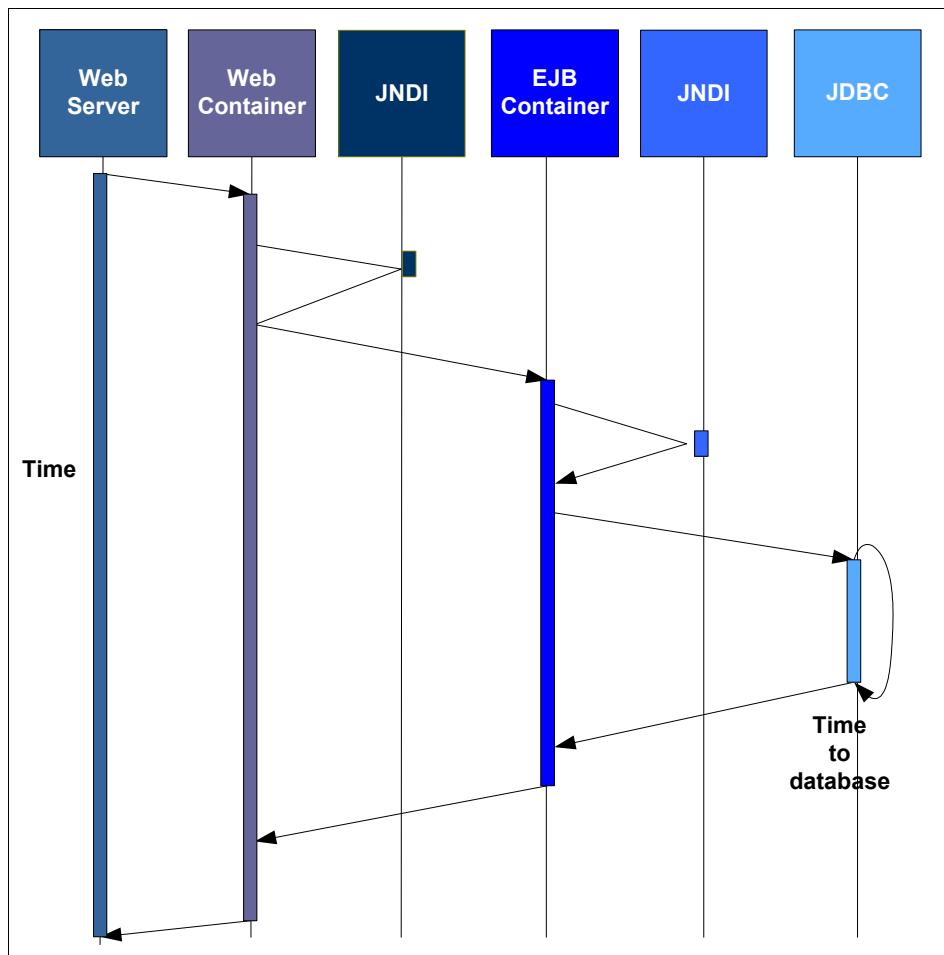


Figure 11-31 Time view of request metrics

Tip: The metrics, when printed to the standard logs, are printed from right to left with the innermost statistic reported first.

Request metrics are ideal for helping to identify the slowest and worst performing request types and can assist in identifying areas where performance can be improved. As a second phase, data from more detailed monitoring for the request type can be broken down to provide insight on where the bulk of time is being spent and provide insight into what areas of the code warrant investigation.

Unlike PMI, there is no built in tool for request metric analysis, thus for these examples, the data will be extracted from the logs. IBM has tooling available in the Tivoli monitoring suites but it is not part of the application server offering.

Example 11-1 shows the request metrics captured for a single request in the standard log file. In this example, request metrics are enabled with a trace level of hops and all components instrumented. As you can see, it is quite verbose even with a small amount of detail active.

If you do not have tools, a simple editor with good search and parsing capabilities can be used to filter the request based on **reqid** or other fields.

Example 11-1 Simplified PMI data example

```
[6/2/09 2:37:12:187 EDT] 00000703 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818051,event=1 type=JDBC  
detail=javax.resource.spi.ManagedConnectionFactory.matchManagedConnections(Set, Subject,  
ConnectionRequestInfo) elapsed=0  
[6/2/09 2:37:12:187 EDT] 00000703 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818052,event=1 type=JDBC  
detail=javax.resource.spi.ManagedConnection.getConnection(Subject, ConnectionRequestInfo)  
elapsed=0  
[6/2/09 2:37:12:187 EDT] 00000703 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818053,event=1 type=JDBC  
detail=javax.resource.spi.XAResource.start(Xid, int) elapsed=0  
[6/2/09 2:37:12:203 EDT] 00000703 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818054,event=1 type=JDBC  
detail=java.sql.PreparedStatement.executeQuery() elapsed=16  
...  
Lots of lines deleted for simplification  
...  
[6/2/09 2:37:12:234 EDT] 00000703 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818089,event=1 type=JDBC  
detail=java.sql.PreparedStatement.executeQuery() elapsed=16
```

```
[6/2/09 2:37:12:234 EDT] 00000703 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818090,event=1 type=JDBC  
detail=javax.resource.spi.XAResource.end(Xid, int) elapsed=0  
[6/2/09 2:37:12:234 EDT] 00000703 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818091,event=1 type=JDBC  
detail=javax.resource.spi.XAResource.commit(Xid, boolean) elapsed=0  
[6/2/09 2:37:12:234 EDT] 00000703 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818092,event=1 type=JDBC  
detail=javax.resource.spi.ManagedConnection.cleanup() elapsed=0  
[6/2/09 2:37:12:234 EDT] 00000703 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 type=URI  
detail=/trade/scenario elapsed=47
```

The timing data can manually be plotted as illustrated in Figure 11-32. Graphing complex applications might be an easy way to visualize where time is being spent. Plotting data manually can take some time. Interaction diagrams can be substituted and the timing plotted on the diagrams.

In this example, the request takes 47 milliseconds (not a long time). But it can be observed that the time for the prepared statement cache used about 70% of this time. Of note in this configuration is that only the edge metrics are captured. The edge being the JDBC data and the original servlet URI request.

Whatever representation is preferred the key understanding that is required is that request metrics provide a useful mechanisms for the analysis of where time is spent in requests that are being executed.

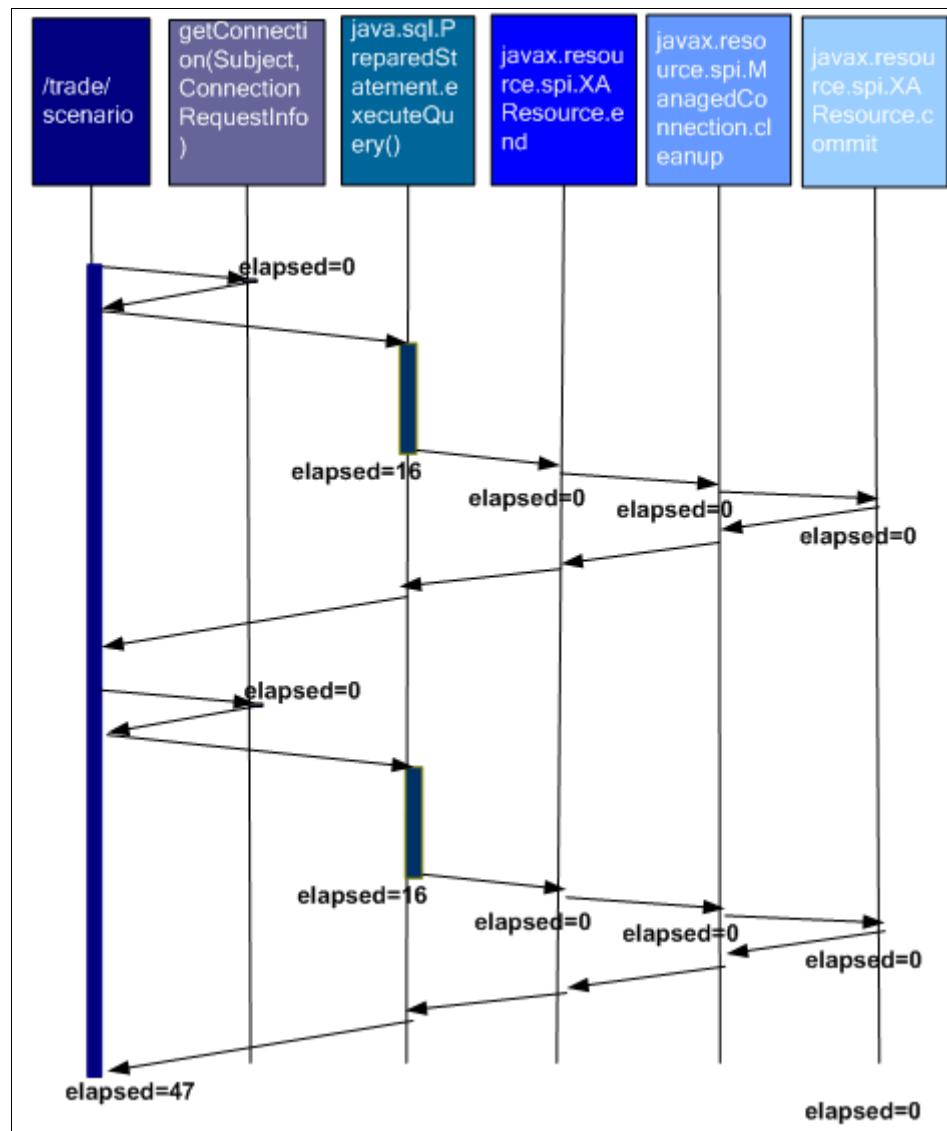


Figure 11-32 Hops time graph for trade application

As more detail is added, the picture becomes more complete. Instead of a trace level of hops, you can use trace level debug to gather more detail (Figure 11-33).

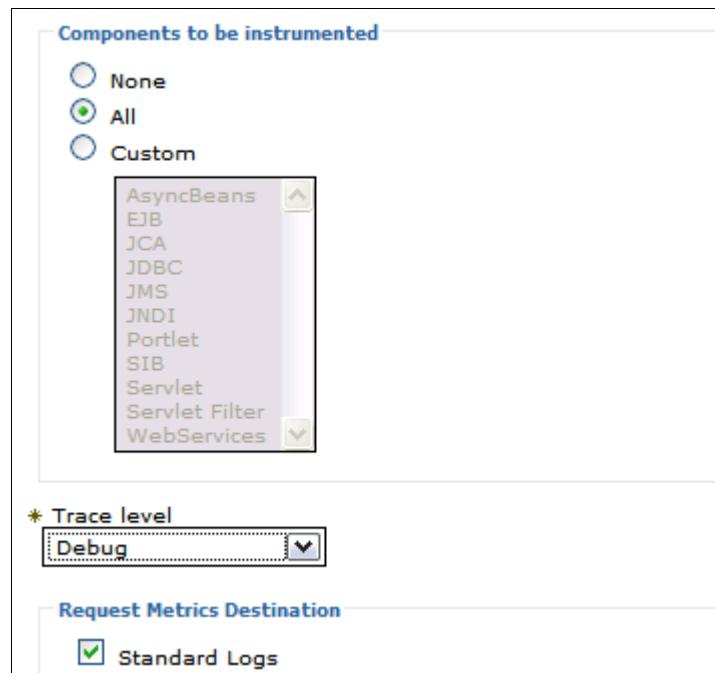


Figure 11-33 Request metrics configuration panel

Example 11-2 shows metrics captured at a trace level of debug.

Example 11-2 Additional component types now recorded

```
[6/2/09 3:43:59:312 EDT] 00000033 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=645,event=1 type=EJB  
detail=com.ibm.websphere.samples.trade.ejb.QuoteBean.findByPrimaryKeyForUpdate  
elapsed=0  
[6/2/09 3:43:59:312 EDT] 00000033 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=650,event=1 type=EJB  
detail=com.ibm.websphere.samples.trade.ejb.QuoteBean.getPrice elapsed=0  
[6/2/09 3:43:59:312 EDT] 00000033 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -  
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=651,event=1 type=EJB  
detail=com.ibm.websphere.samples.trade.ejb.QuoteBean.getPrice elapsed=0  
[6/2/09 3:43:59:328 EDT] 00000033 PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -
```

```
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=652,event=1 type=EJB
detail=com.ibm.websphere.samples.trade.ejb.QuoteBean.updatePrice elapsed=16
...
Lines deleted
...
[6/2/09 3:43:59:484 EDT] 00000033 PmiRmArmWrapp I PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=674,event=1 type=JDBC
detail=javax.resource.spi.XAResource.commit(Xid, boolean) elapsed=0
[6/2/09 3:43:59:484 EDT] 00000033 PmiRmArmWrapp I PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=675,event=1 type=JDBC
detail=javax.resource.spi.ManagedConnection.cleanup() elapsed=0
[6/2/09 3:43:59:484 EDT] 00000033 PmiRmArmWrapp I PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=567,event=1 -
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 type=EJB
detail=com.ibm.websphere.samples.trade.ejb.TradeBean.updateQuotePriceVolume
elapsed=172
```

For more information about these techniques, see:

- ▶ Why use request metrics? at:
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tprf_requestmetrics.html
- ▶ Example, at:
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rprf_example.html

11.5 ITCAM for WebSphere

IBM Tivoli Composite Application Manager (ITCAM) for WebSphere is shipped with WebSphere Application V7. After being installed, it is embedded in the application server. ITCAM for WebSphere provides a subset of the ITCAM for Web resources. The following sections show how to configure and use ITCAM for WebSphere.

11.5.1 Installing the data collector

The first step in enabling ITCAM is to install and configure a data collector for the monitored servers. ITCAM for WebSphere can be installed from the WebSphere Application Server V7 installation launchpad.

For information about the installation, refer to the product installation documentation:

http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.itcamwas_wr.doc_6.2/welcome.html

11.5.2 Configuring ITCAM for WebSphere metrics

The last step of the installation process contains a check box to start the configuration tool that configures the servers for data collection. The configuration tool can be started immediately after installation using this option, or you can start it after installation.

The tool is located in *itcam_install/config_dc/config_dc*.

The TPV settings for the servers can also be adjusted to include ITCAM data during the configuration.

Follow the instructions in the data configuration wizard:

1. Click **Next** on the data collector configuration panel (Figure 11-34).

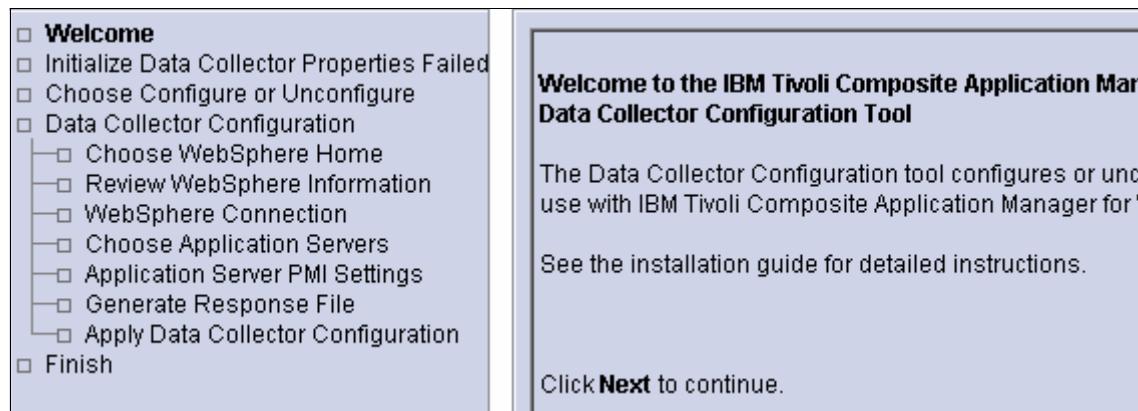


Figure 11-34 Collector configuration wizard

2. Select **Configure application servers for data collection** (Figure 11-35) and click **Next**.



Figure 11-35 Configure application servers for data collection

3. Select the node that the collector is being configured for and click **Next**. (Figure 11-36)

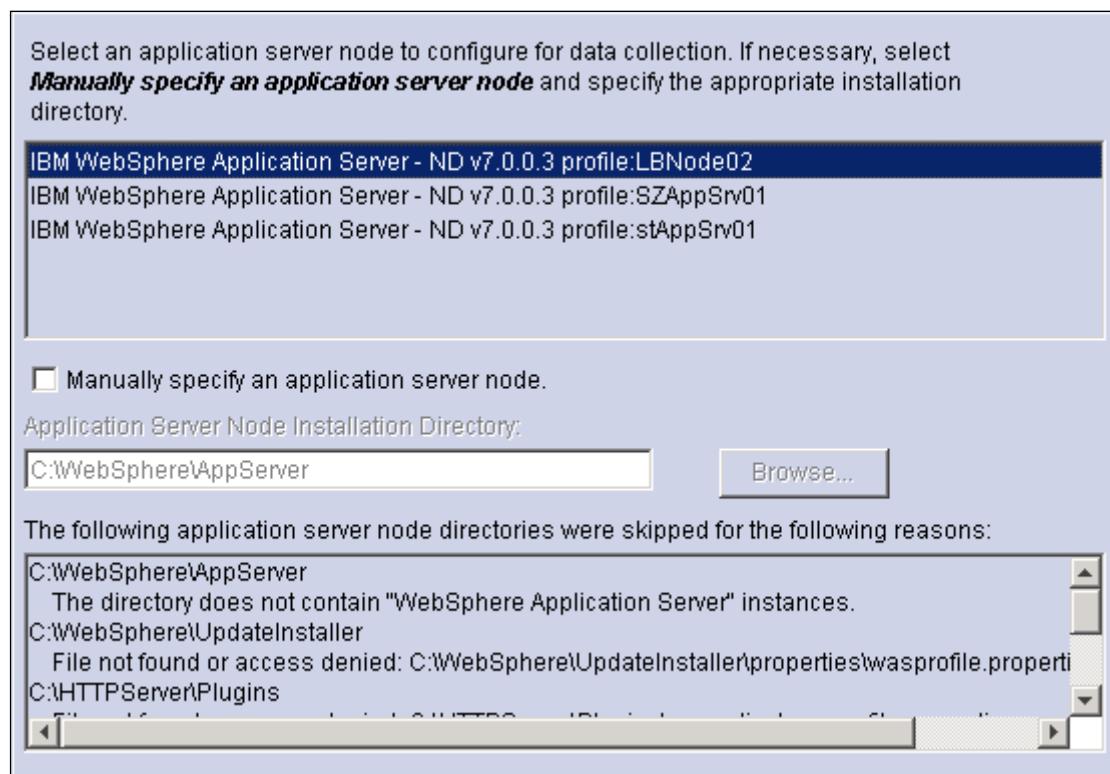


Figure 11-36 Data collector configuration

4. Click **Next** on the directory home panel.

5. Enter the administration port details (Figure 11-37). For Network Deployment, this needs to be the information for the deployment manager configuration. Click **Next**.

For a standalone application server, specify the appropriate host name (or IP address) and SOAP/RMI port. You must run the Data Collector Configuration Tool against each application server that will be monitored via IBM Tivoli Composite Application Manager for WebSphere Application Server. In addition, ensure that the target application server is running before using the Configuration Tool.

For a Network Deployment environment, specify the host name (or IP address) and SOAP/RMI port of the deployment manager. The deployment manager and node agent must be running.

WebSphere Global Security is enabled. Enter the WebSphere user name and password.

Host Name:	sys2.itso.ral.ibm.com
Connector Type:	SOAP
SOAP Connector Port:	8879
User Name:	admin1
Password:	*****
<input type="checkbox"/> Show Advanced Options	

Figure 11-37 Enter administration port details

6. Specify which servers will have data collection configured and click **Next** (Figure 11-38).

Select the application servers to configure for data collection.

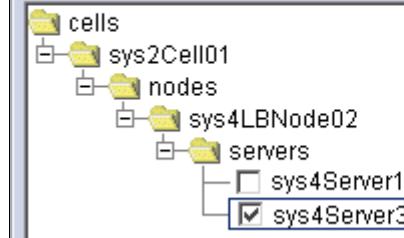


Figure 11-38 Choose servers to configure data collector for

7. As part of the data collection configuration, the PMI data settings can be modified to include ITCAM metrics, this can be done manually at another time or the administrator can allow the configuration tool to complete the modification (Figure 11-39). Click **Next**.

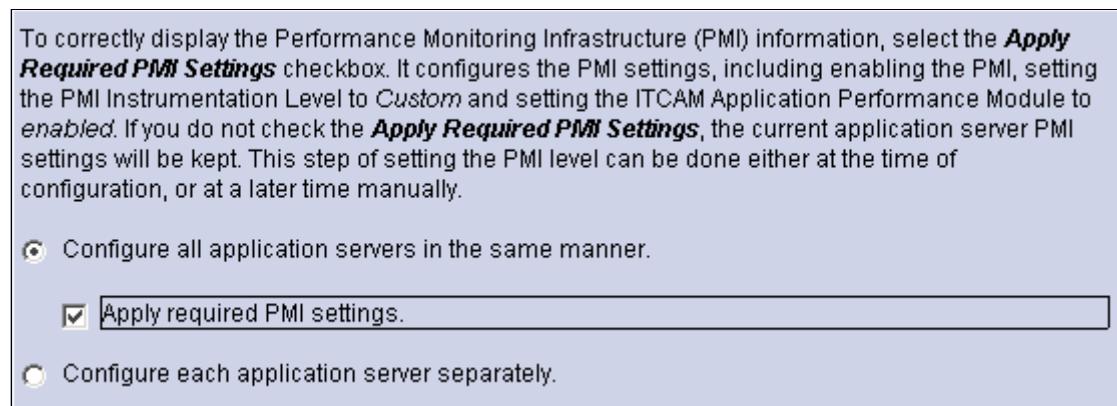


Figure 11-39 Modify PMI settings

8. Finalize the configuration steps (Figure 11-40) and click **Next**.

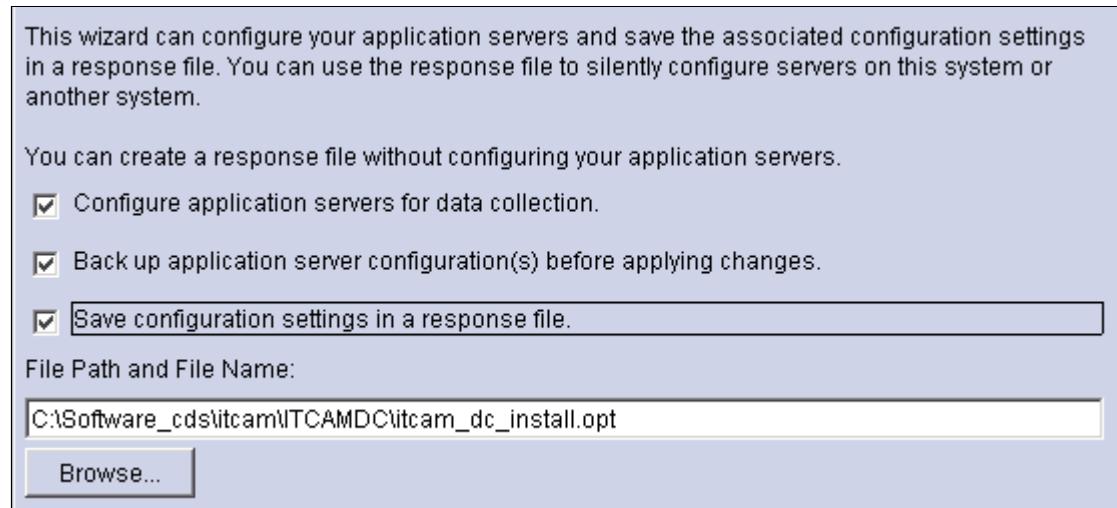


Figure 11-40 Configure the servers for data collection and create options file

9. Click **Finish**.

10. After the data collector configuration is complete, navigate in the WebSphere administration console to the Performance Monitoring Infrastructure panel for the configured server (Figure 11-41). A new **ITCAM for WAS** link is now in the Additional Properties section of the panel.

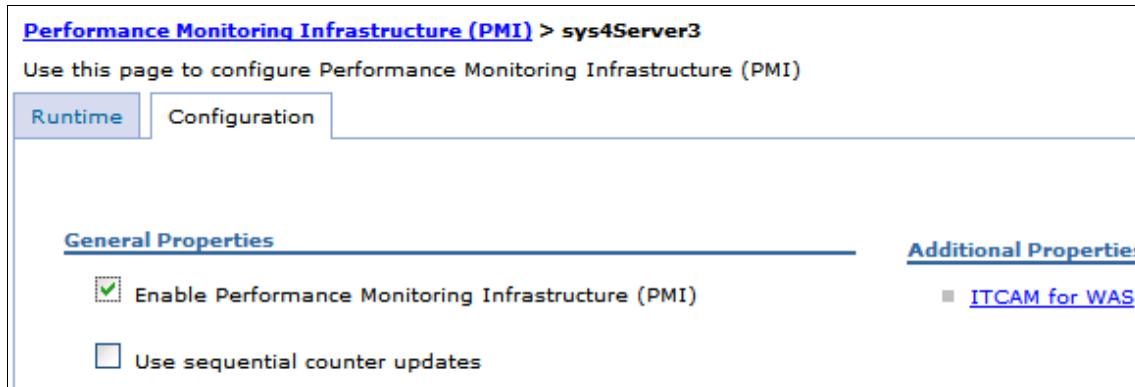


Figure 11-41 PMI configuration with ITCAM for WAS

11. Select the **ITCAM for WAS** link to open the ITCAM for WAS panel (Figure 11-42). This panel contains the setting that enables the data collector.

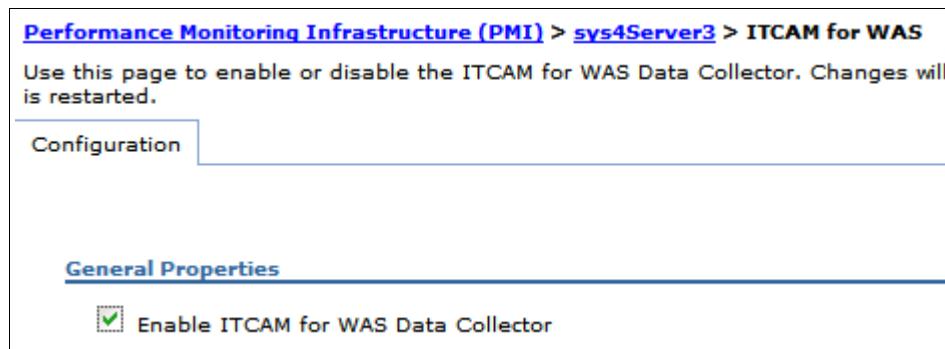


Figure 11-42 ITCAM for WAS panel

Note: This setting will only be enabled if you choose to apply the required PMI settings as shown in Figure 11-36 on page 595. Otherwise you will need to enable the ITCAM for WAS data collector and restart the server.

The PMI settings must also be configured at custom level.

12. Restart the servers for which the data collector was configured to make the metrics become active.

11.5.3 Viewing ITCAM for WebSphere data

This section assumes that the data collector has been installed and the ITCAM for WAS data collector is enabled as shown in Figure 11-42 on page 598.

1. Navigate to the Performance Monitoring Infrastructure panel for the server (Figure 11-43). Confirm that the **Custom** statistic set is selected.

The screenshot shows the 'Performance Monitoring Infrastructure (PMI) > sys4Server3' configuration panel. The 'Runtime' tab is active. In the 'General Properties' section, the checkbox for 'Enable Performance Monitoring Infrastructure (PMI)' is checked. Below it, there are two checkboxes: 'Use sequential counter updates' (unchecked) and 'Enable Performance Monitoring Infrastructure (PMI)' (checked). In the 'Currently monitored statistic set' section, the 'Custom' option is selected, which is highlighted with a green circle and underlined. Other options include 'None', 'Basic', 'Extended', and 'All'. To the right, under 'Additional Properties', there is a link labeled 'ITCAM for WA'.

Figure 11-43 PMI Configuration panel

2. Click the **ITCAM for WAS** link and navigate to open the configuration panel (Figure 11-44).

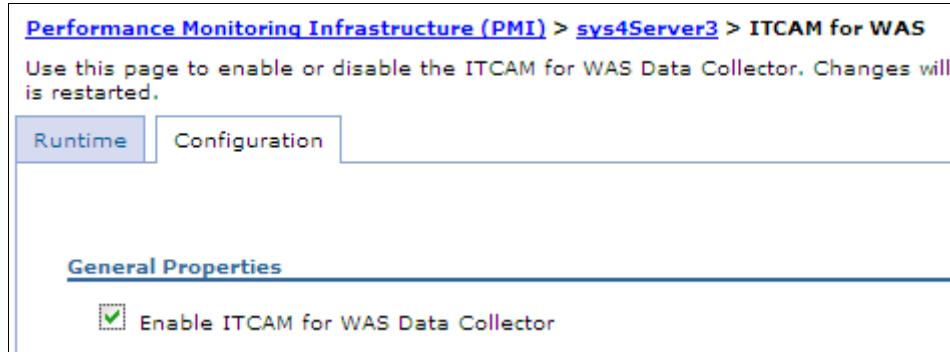


Figure 11-44 ITCAM for WAS configuration tab

3. Click the **Runtime** tab (Figure 11-45).



Figure 11-45 ITCAM for WAS runtime tab

4. Click the **Start Monitoring** button.

Notes:

You must start ITCAM for WAS monitoring on a server each server restart. It is not preserved in the configuration.

After monitoring is started, the **Stop Monitoring** button is displayed instead. If you decide later to stop the ITCAM for WAS monitoring then return to this panel and click the button.

5. Click the server link (sys4Server3) in the navigation trail at the top of the panel to return to the PMI configuration panel.

- Click the **Custom** link to navigate into the custom monitoring panel. In this panel select the **Runtime** tab. Click **ITCAM Application Performance** to show the ITCAM counters (Figure 11-46).

Note: ITCAM counters are only visible in the Runtime tab.

[Performance Monitoring Infrastructure \(PMI\)](#) > [sys4Server3](#) > Custom monitoring level

Use this page to configure Performance Monitoring Infrastructure (PMI)

Runtime Configuration

sys4Server3

- + DCS Statistics
- + ExtensionRegistryStats.name
- + SIB Service
- + SipContainerModule
- + Enterprise Beans
- + Dynamic Caching
- + JDBC Connection Pools
- + ITCAM Application Performance
- + HAManager
- + JCA Connection Pools
- + JVM Runtime
- + Object Pool
- + ORB
- + pmiWebServiceModule
- + Servlet Session Manager
- + Thread Pools
- + Transaction Manager
- + Web Applications
- + Web services
- + Workload Management
- + Web services Gateway

Enable Disable

Select	Counter	Type
<input type="checkbox"/>	90%CPUUsage	CountStat
<input type="checkbox"/>	90%ResponseTime	CountStat
<input type="checkbox"/>	AverageCPUUsage	AverageStat
<input type="checkbox"/>	AverageResponseTime	AverageStat
<input type="checkbox"/>	LastMinuteAverageCPUUsage	AverageStat
<input type="checkbox"/>	LastMinuteAverageResponseTime	AverageStat
<input type="checkbox"/>	MaximumCPUUsage	CountStat
<input type="checkbox"/>	MaximumResponseTime	CountStat
<input type="checkbox"/>	MinimumCPUUsage	CountStat
<input type="checkbox"/>	MinimumResponseTime	CountStat
<input type="checkbox"/>	RequestCount	CountStat

Total 11

Figure 11-46 ITCAM Application performance counters

7. The next step is to choose which ITCAM for WAS counters are needed. The counter panel provides additional description information and the current status of the counter.

Choose the counters that are desired and click **Enable**.

Note that some counters can only be activated by the system. This means that they will stay in disabled mode, even if you try to enable them. This does not mean the counters are broken. In Figure 11-47, you can see that some counters remain in the Disabled state, even though all the counters were selected to be enabled.

ITCAM for WAS Counter Selection				
Select	Counter	Type	Description	Status
<input type="checkbox"/>	90%CPUUsage	CountStatistic	This metric collects the CPU usage and then calculates the 90% median CPU usage.	Disabled
<input type="checkbox"/>	90%ResponseTime	CountStatistic	This metric collects the response time and then calculates the 90% median response time.	Disabled
<input type="checkbox"/>	AverageCPUUsage	AverageStatistic	This metric collects the CPU usage and then calculates the average CPU usage in milliseconds.	Enabled
<input type="checkbox"/>	AverageResponseTime	AverageStatistic	This metric collects the response time and then calculates the average response in milliseconds.	Enabled
<input type="checkbox"/>	LastMinuteAverageCPUUsage	AverageStatistic	This metric collects the CPU usage for the last minute and then calculates the average CPU usage in milliseconds.	Enabled
<input type="checkbox"/>	LastMinuteAverageResponseTime	AverageStatistic	This metric collects the response time for the last minute and then calculates the average response in milliseconds.	Enabled
<input type="checkbox"/>	MaximumCPUUsage	CountStatistic	This metric collects the CPU usage and then calculates the maximum CPU usage in milliseconds.	Disabled
<input type="checkbox"/>	MaximumResponseTime	CountStatistic	This metric collects the response time and then calculates the maximum response in milliseconds.	Disabled
<input type="checkbox"/>	MinimumCPUUsage	CountStatistic	This metric collects the CPU usage and then calculates the minimum CPU usage in milliseconds.	Disabled
<input type="checkbox"/>	MinimumResponseTime	CountStatistic	This metric collects the response time and then calculates the minimum response in milliseconds.	Disabled
<input type="checkbox"/>	RequestCount	CountStatistic	The total number of requests.	Enabled
Total 11				

Figure 11-47 ITCAM counters enabled

- After enabling the counters, navigate to the Current Activity Tivoli Performance Viewer panel for the server that is being monitored (Figure 11-48). Open the Performance Modules tree view and then select the **ITCAM Application Performance** menu item.

Servlets Summary Report

[More information about this page](#)

Name	Application	Total Requests
TradeAppServlet	Trade#tradeWeb.war	0
TradeScenarioServlet	Trade#tradeWeb.war	0
rsp servlet	ibmasyncrsp.war	0
Total 3		

Start Logging

sys4Server3

- sys4Server3
 - + Advisor
 - + Settings
 - + Summary Reports
 - + Performance Modules
 - + DCS Statistics
 - ExtensionRegistryStats.name
 - + SIB Service
 - + SipContainerModule
 - + Enterprise Beans
 - + Dynamic Caching
 - + JDBC Connection Pools
 - + HAManager
 - + JCA Connection Pools
 - JVM Runtime
 - + Object Pool
 - + ORB
 - + pmiWebServiceModule
 - + Servlet Session Manager
 - + Thread Pools
 - Transaction Manager
 - + Web Applications
 - + Web services
 - + Workload Management
 - + Web services Gateway
 - + ITCAM Application Performance

Figure 11-48 Select the ITCAM Application performance menu item

9. Click the **View Module(s)** button to view the data (Figure 11-49).

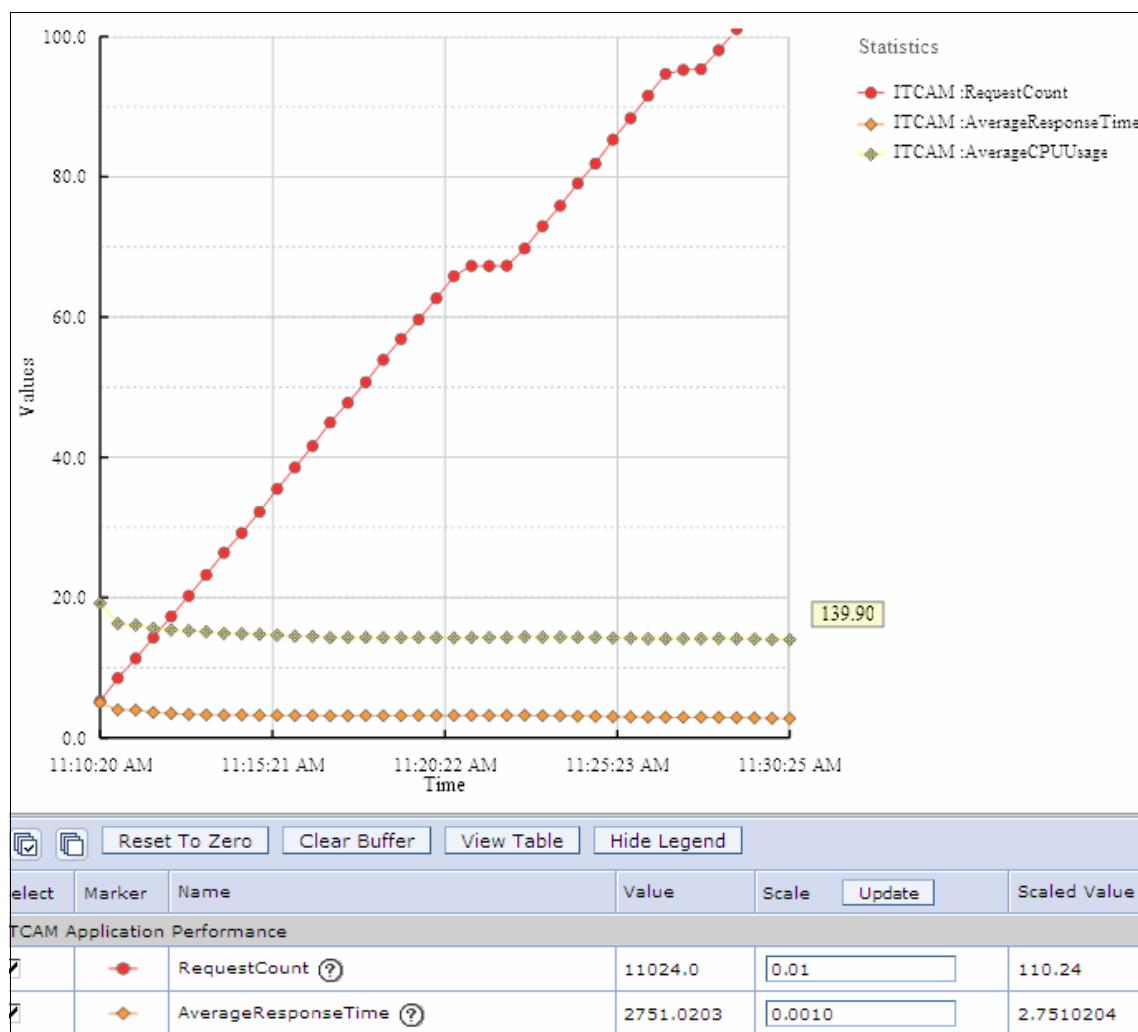


Figure 11-49 ITCAM counters graph

11.6 Monitoring considerations summary

In this section we provide some final thoughts and advice on monitoring.

11.6.1 Other tools and considerations

There are a number of other sources of information an administrator can use in relation to monitoring the health of an application server.

Such resources include logging, garbage collection, and operating system views:

- ▶ Logging:

WebSphere Application Server provides a great many logging options and the most significant environment incidents are logged automatically. To support the analysis of logging activity, the IBM Support Assistant can be used to launch the IBM Tivoli Log Analyzer. This tool supports the downloading of symptom catalogs of known issues that go into the logs. Further, application developers can build symptom catalogs and write logs from their applications to add to the manageability of their applications.

Figure 11-50 shows the IBM Tivoli Log analyzer.

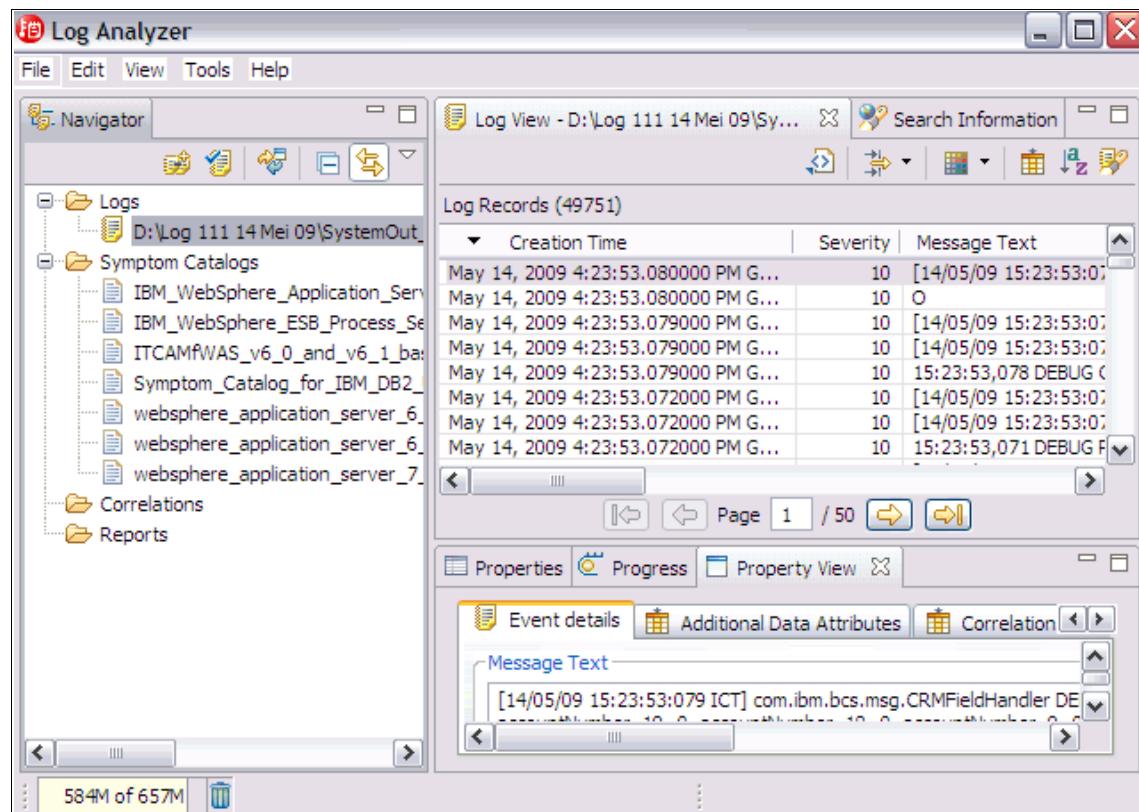


Figure 11-50 IBM Tivoli log analyzer

- ▶ Verbose garbage collection:

Verbose garbage collection was mentioned earlier in the chapter. Again, we recommend that, assuming there is appropriate disk space, a monitoring strategy would include verbose garbage collection. To enable verbose garbage collection, navigate to the process definition for the JVM of the server (Figure 11-51) and check the **Verbose garbage collection** box.

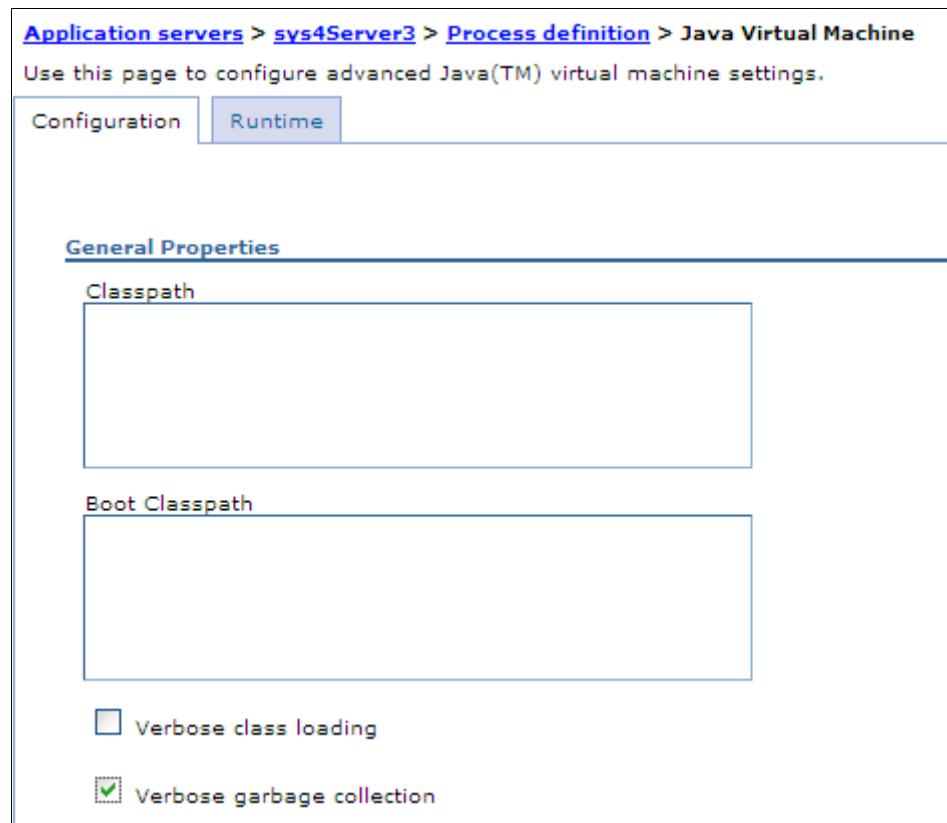


Figure 11-51 Enable verbose garbage collection

Verbose garbage collection can also be enabled at runtime (Figure 11-52).

The screenshot shows a web-based administration interface for a Java Virtual Machine. At the top, the navigation path is: Application servers > sys4Server3 > Process definition > Java Virtual Machine. A sub-instruction reads: Use this page to configure advanced Java(TM) virtual machine settings. Below this, there are two tabs: Configuration (which is highlighted in blue) and Runtime. Under the Configuration tab, there is a section titled General Properties. Within this section, there is a checkbox labeled Verbose garbage collection, which has a green checkmark indicating it is selected.

Figure 11-52 Enable verbose garbage collection at runtime

- ▶ Operating system views:
Most operating systems have tools for the monitoring of memory, CPU utilization, and disk I/O. These should be monitored and controlled.
- ▶ The monitoring tools available with WebSphere Application Server are useful in any size enterprise. However, in larger enterprises, a more complete system monitoring solution (such as those provided in the advanced ITCAM product suites) are more suitable for improved centralized monitoring. Tools that can monitor systems end-to-end are very beneficial.

11.6.2 Summary of monitoring tips

The following summary lists a simple set of recommendations on how to establish a useful monitoring environment.

- ▶ Take time and understand the applications that are being deployed into the environment. Use this knowledge to plan for the kinds of metrics that will be beneficial in understanding application performance.
- ▶ Activate monitoring at the planned level in all testing environments especially when benchmarking. While vital to your ability to understand an environment, monitoring is not free and uses CPU, memory and other resources. Monitoring can impact capacity planning.
- ▶ Use monitoring to understand normal operations and gain an appreciation of what is normal for your systems.
- ▶ Check for differences in your systems after all release changes, especially if full performance testing and benchmarks have not been re-established.
- ▶ Use the basic metrics of PMI as a good starting set of metrics, but customize them to your needs.
- ▶ Monitoring WebSphere Application Server alone is not enough. Application server monitoring needs to be part of an overall monitoring strategy.



Part 2

Working with applications



Session management

Session support allows a Web application developer to maintain state information across multiple user visits to the application. In this chapter, we discuss HTTP session support in WebSphere Application Server V7 and how to configure it. We also discuss the support for stateful session bean failover.

We cover the following topics:

- ▶ “HTTP session management” on page 612
- ▶ “Session manager configuration” on page 612
- ▶ “Session identifiers” on page 615
- ▶ “Local sessions” on page 620
- ▶ “General properties for session management” on page 622
- ▶ “Session affinity” on page 624
- ▶ “Persistent session management” on page 628
- ▶ “Invalidating sessions” on page 659
- ▶ “Session security” on page 661
- ▶ “Session performance considerations” on page 663
- ▶ “Stateful session bean failover” on page 670

12.1 HTTP session management

In many Web applications, users collect data dynamically as they move through the site based on a series of selections on pages they visit. Where the user goes next, and what the application displays as the user's next page, or next choice, depends on what the user has chosen previously from the site. For example, if the user clicks the checkout button on a site, the next page must contain the user's shopping selections.

In order for this to happen, a Web application needs a mechanism to hold the user's state information over a period of time. However, HTTP does not recognize or maintain a user's state. HTTP treats each user request as a discrete, independent interaction.

The Java servlet specification provides a mechanism for servlet applications to maintain a user's state information. This mechanism, known as a *session*, addresses some of the problems of more traditional strategies, such as a pure cookie solution. It allows a Web application developer to maintain all user state information at the host, while passing minimal information back to the user through cookies, or another technique known as *URL rewriting*.

(New in V7) In WebSphere Application Server V7 session tracking, using SSL ID is deprecated. Use cookies or URL rewriting instead.

12.2 Session manager configuration

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.
- ▶ Application:
Configuration at this level is applied to all Web modules within the application.
- ▶ Web module:
Configuration at this level is applied only to that Web module.

12.2.1 Session management properties

With one exception, the session management properties that you can set are the same at each configuration level:

- ▶ *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.
- ▶ *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* lets you specify the maximum number of sessions to keep in memory and whether to allow this number to be exceeded, or overflow.
- ▶ *Session timeout* specifies the amount of time to allow a session to remain idle before invalidation.
- ▶ *Security integration* specifies that the user ID be associated with the HTTP session.
- ▶ *Serialize session access* determines if concurrent session access in a given server is allowed.
- ▶ *Distributed environment settings* determines how to persist sessions (memory-to-memory replication or a database) and set tuning properties. Memory-to-memory persistence is only available in a Network Deployment distributed server environment.

12.2.2 Accessing session management properties

You can access all session management configuration settings using the administrative console.

Application server session management properties

To access session management properties at the application server level, from the administrative console, do the following steps:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. Click the application server.
3. In the Container Settings section of the Configuration tab, click **Session management**.

Application session management properties

To access session management properties at the application level, from the administrative console, do the following steps:

1. Click **Applications** → **Application Types** → **WebSphere enterprise applications**.
2. Click the application name to open its configuration page.
3. In the Web Module Properties section of the Configuration tab, click **Session management**.

Web module session management properties

To access session management properties at the Web module level, from the administrative console, do the following steps:

1. Click **Applications** → **Application Types** → **WebSphere enterprise applications**.
2. Click the application.
3. In the Modules section of the Configuration tab, click **Manage Modules**.
4. Click the Web module.
5. In the Additional Properties section, click **Session Management**.

12.3 Session identifiers

WebSphere session support keeps information about the user's session on the server. WebSphere passes the user an identifier known as a *session ID*, which correlates an incoming user request with a session object maintained on the server.

Note: The example session IDs provided in this chapter are for illustrative purposes only and are *not* guaranteed to be absolutely consistent in value, format, and length.

12.3.1 Choosing a session tracking mechanism

WebSphere supports three approaches to tracking sessions:

- ▶ SSL session identifiers (deprecated)
- ▶ Cookies
- ▶ URL rewriting

It is possible to select all three options for a Web application. If you do this:

- ▶ SSL session identifiers are used in preference to cookie and URL rewriting.
- ▶ Cookies are used in preference to URL rewriting.

Note: If SSL session ID tracking is selected, we recommend 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.

To set or change the session mechanism type, do the following steps:

1. Open the session management properties for the application server, enterprise application, or Web module.
2. Select the session tracking mechanism (Figure 12-1).

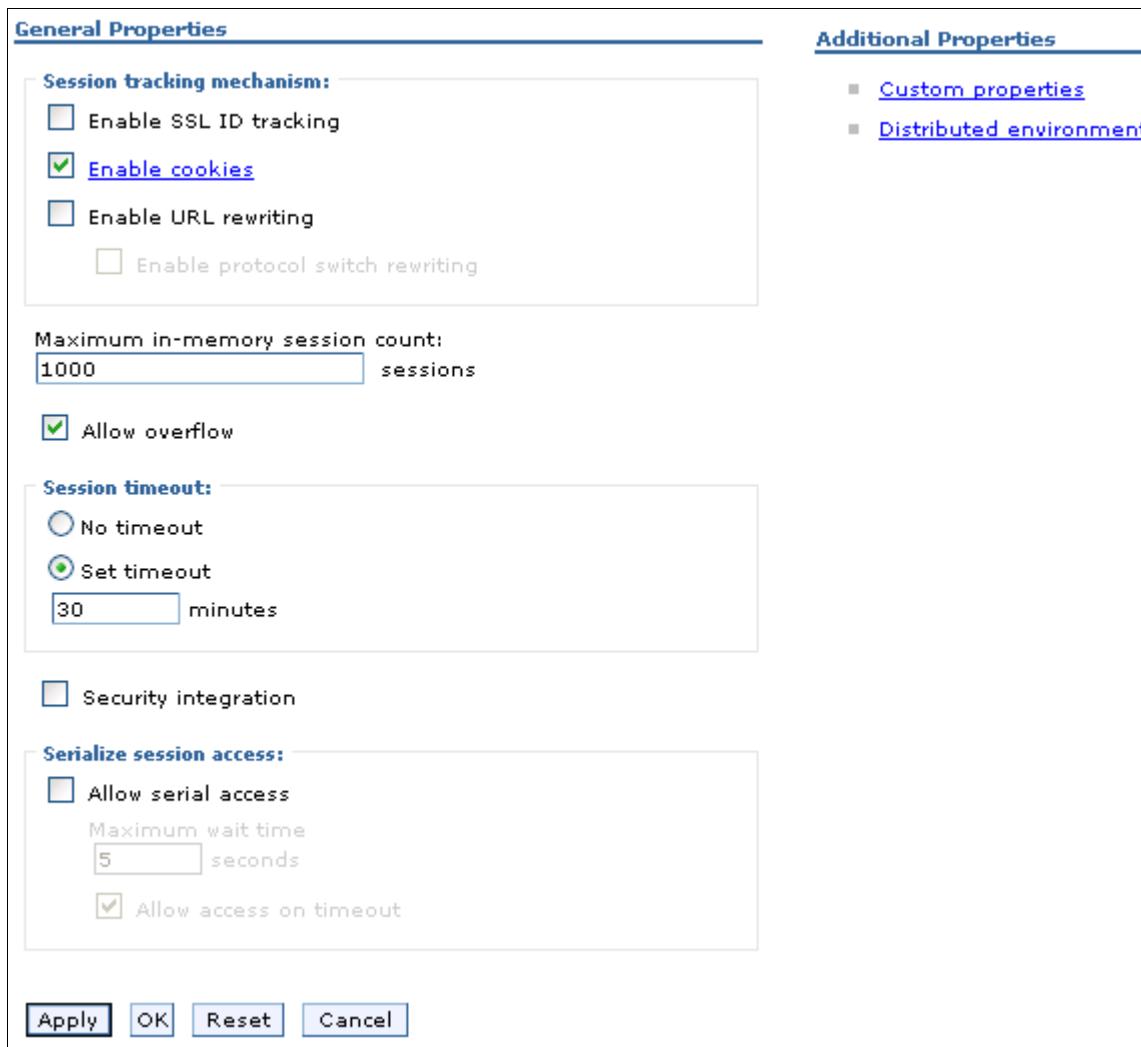


Figure 12-1 Selecting a session tracking mechanism window

3. Click **OK**.
4. Save and synchronize the configuration changes.
5. Restart the application server or the cluster.

12.3.2 Cookies

Many sites choose cookie support to pass the user's identifier between WebSphere and the user. WebSphere Application Server session support generates a unique session ID for each user, and returns this ID to the user's browser with a cookie. The default name for the session management cookie is JSESSIONID. See Figure 12-2.

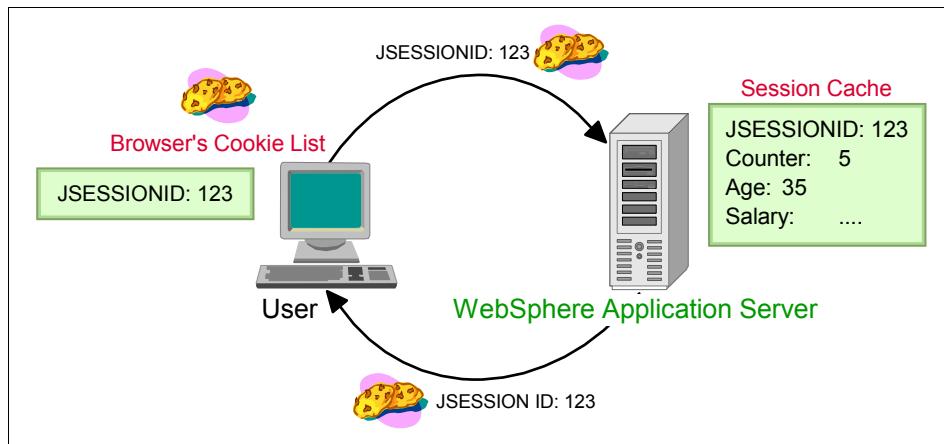


Figure 12-2 Cookie overview

A cookie consists of information embedded as part of the headers in the HTML stream passed between the server and the browser. The browser holds the cookie and returns it to the server whenever the user makes a subsequent request. By default, WebSphere defines its cookies so they are destroyed if the browser is closed. This cookie holds a session identifier. The remainder of the user's session information resides at the server.

The Web application developer uses the HTTP request object's standard interface to obtain the session:

```
HttpSession session = request.getSession(true);
```

WebSphere places the user's session identifier in the outbound cookie whenever the servlet completes its execution, and the HTML response stream returns to the end user. Again, neither the cookie or the session ID within it require any direct manipulation by the Web application. The Web application only sees the contents of the session.

Be aware that some users, either by choice or mandate, disable cookies from within their browser.

Cookie settings

To configure session management using cookies, do the following steps from the administrative console:

1. Open the Session Manager window at your preferred level.
2. Click the box for **Enable Cookies** as the session tracking mechanism. See Figure 12-1 on page 616.
3. If you would like to view or change the cookies settings, select the **Enable Cookies** hot link. The following cookie settings are available:

- Cookie name:

The cookie name for session management should be unique. The default cookie name is JSESSIONID. However, this value can be configured for flexibility.

- Restrict cookies to HTTPS sessions:

Enabling this feature restricts the exchange of cookies only to HTTPS sessions. If it is enabled, the session cookie's body includes the secure indicator field.

- Cookie domain:

This value dictates to the browser whether or not to send a cookie to particular servers. For example, if you specify a particular domain, the browser will only send back session cookies to hosts in that domain. The default value in the session manager restricts cookies to the host that sent them.

Note: The Lightweight Third Party Authentication (LTPA) token/cookie that is sent back to the browser is scoped by a single DNS domain specified when security is configured. This means that *all* application servers in an *entire* WebSphere Application Server domain must share the same DNS domain for security purposes.

- Cookie path:

The paths on the server to which the browser will send the session tracking cookie. Specify any string representing a path on the server. Use the slash (/) to indicate the root directory.

Specifying a value restricts the paths to which the cookie will be sent. By restricting paths, you can keep the cookie from being sent to certain URLs on the server. If you specify the root directory, the cookie will be sent no matter which path on the given server is accessed.

- Cookie maximum age:

The amount of time that the cookie will live in the client browser. There are two choices:

- Expire at the end of the current browser session
- Expire at a configurable maximum age

If you choose the maximum age option, specify the age in seconds.

4. Click **OK** to exit the page and change your settings.
5. Click **OK** to exit the session management settings.
6. Save and synchronize your configuration changes.
7. Restart the application server or the cluster.

12.3.3 URL rewriting

WebSphere also supports URL rewriting for session ID tracking. While session management using SSL IDs or cookies is transparent to the Web application, URL rewriting requires the developer to use special encoding APIs, and to set up the site page flow to avoid losing the encoded information.

URL rewriting works by storing the session identifier 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. This is an example of a Web page link with URL encoding:

```
<a href="/store/catalog;jsessionid=DA32242SSGE2">
```

When the user clicks this link to move to the /store/catalog page, the session identifier is passed in the request as a parameter.

URL rewriting requires explicit action by the Web application developer. If the servlet returns HTML directly to the requester, without using a JavaServer Page, the servlet calls the API, as shown in Example 12-1, to encode the returning content.

Example 12-1 URL encoding from a servlet

```
out.println("<a href=\"");
out.println(response.encodeURL ("/store/catalog"));
out.println("\>catalog</a>");
```

Even pages using redirection, a common practice, particularly with servlet or JSP combinations, must encode the session ID as part of the redirect, as shown in Example 12-2.

Example 12-2 URL encoding with redirection

```
response.sendRedirect(response.encodeRedirectURL("http://myhost/store/catalog"));
```

When JavaServer Pages (JSPs) use URL rewriting, the JSP calls a similar interface to encode the session ID:

```
<% response.encodeURL ("/store/catalog"); %>
```

URL rewriting configuration

When you select URL rewriting, an additional configuration option, Enable protocol switch rewriting, is available. This option defines whether the session ID, added to a URL as part of URL encoding, should be included in the new URL if a switch from HTTP to HTTPS or from HTTPS to HTTP is required. For example, if a servlet is accessed over HTTP and that servlet is doing encoding of HTTPS URLs, URL encoding will be performed only when protocol switch rewriting is enabled, and vice versa.

Considerations for using URL rewriting

The fact that the servlet or JSP developer has to write extra code is a major drawback over the other available session tracking mechanisms.

URL rewriting limits the flow of site pages exclusively to dynamically generated pages, such as pages generated by servlets or JSPs. WebSphere inserts the session ID into dynamic pages, but cannot insert the user's session ID into static pages, .htm, or .html.

Therefore, after the application creates the user's session data, the user must visit dynamically generated pages exclusively until they finish with the portion of the site requiring sessions. URL rewriting forces the site designer to plan the user's flow in the site to avoid losing their session ID.

12.4 Local sessions

Many Web applications use the simplest form of session management: the in-memory, local session cache. The local session cache keeps session information in memory and local to the machine and WebSphere Application Server where the session information was first created.

Local session management does not share user session information with other clustered machines. Users only obtain their session information if they return to the application server. Most importantly, local session management lacks a persistent store for the sessions it manages. A server failure takes down not only the WebSphere instances running on the server, but also destroys any sessions managed by those instances.

WebSphere allows the administrator to define a limit on the number of sessions held in the in-memory cache from the administrative console settings on the session manager. This prevents the sessions from acquiring too much memory in the Java VM associated with the application server.

The session manager also allows the administrator to permit an unlimited number of sessions in memory. If the administrator enables the **Allow overflow** setting on the session manager, the session manager permits two in-memory caches for session objects. The first cache contains only enough entries to accommodate the session limit defined to the session manager, 1000 by default. The second cache, known as the overflow cache, holds any sessions the first cache cannot accommodate, and is limited in size only by available memory. The session manager builds the first cache for optimized retrieval, while a regular, un-optimized hash table contains the overflow cache.

For best performance, define a primary cache of sufficient size to hold the normal working set of sessions for a given Web application server.

Important: If you enable overflow, the session manager permits an unlimited number of sessions in memory. Without limits, the session caches might consume all available memory in the WebSphere instance's heap, leaving no room to execute Web applications. For example, here are two scenarios under which this could occur:

- ▶ The site receives greater traffic than anticipated, generating a large number of sessions held in memory.
- ▶ A malicious attack occurs against the site where a user deliberately manipulates their browser so the application creates a new session repeatedly for the same user.

If you choose to enable session overflow, the state of the session cache should be monitored closely.

Note: Each Web application will have its own base, or primary, in-memory session cache, and with overflow allowed, its own overflow, or secondary, in-memory session cache.

12.5 General properties for session management

The session management settings allow the administrator to tune a number of parameters that are important for both local or persistent sessions (see Figure 12-1 on page 616). Next we describe the settings:

- ▶ Maximum in-memory session count:

This field specifies the maximum number of sessions to maintain in memory. The meaning differs depending on whether you are using local or persistent sessions. For local sessions, this value specifies the number of sessions in the base session table. Select **Allow overflow** to specify whether to limit sessions to this number for the entire session manager, or to allow additional sessions to be stored in secondary tables. Before setting this value, see 12.4, “Local sessions” on page 620.

For persistent sessions, this value specifies the size of the general cache. This value determines how many sessions will be cached before the session manager reverts to reading a session from the database automatically. Session manager uses a least recently used (LRU) algorithm to maintain the sessions in the cache.

This value holds when you use local sessions, persistent sessions with caching, or persistent sessions with manual updates. The manual update cache keeps the last n time stamps representing the last access times, where n is the maximum in-memory session count value.

- ▶ Allow overflow:

Choosing this option specifies whether to allow the number of sessions in memory to exceed the value specified in the maximum in-memory session count field. If **Allow overflow** is not checked, then WebSphere limits the number of sessions held in memory to this value.

For local sessions, if this maximum is exceeded and Allow overflow is not checked, then sessions created thereafter will be dummy sessions and will not be stored in the session manager. Before setting this value, see 12.4, “Local sessions” on page 620.

As shown in Example 12-3, the IBM HttpSession extension can be used to react if sessions exceed the maximum number of sessions specified when overflow is disabled.

Example 12-3 Using IBMSession to react to session overflow

```
com.ibm.websphere.servlet.session.IBMSession sess =  
    (com.ibm.websphere.servlet.session.IBMSession) req.getSession(true);  
if(sess.isOverFlow()) {  
    //Direct to a error page...  
}
```

Note: Allowing an unlimited amount of sessions can potentially exhaust system memory and even allow for system sabotage. Someone could write a malicious program that continually hits your site, creating sessions, but ignoring any cookies or encoded URLs and never utilizing the same session from one HTTP request to the next.

- ▶ Session timeout:

If you select **Set timeout**, when a session is not accessed for this many minutes it can be removed from the in-memory cache and, if persistent sessions are used, from the persistent store. This is important for performance tuning. It directly influences the amount of memory consumed by the JVM in order to cache the session information.

Note: For performance reasons, the session manager invalidation process runs at regular intervals to invalidate any invalid sessions. This interval is determined internally based on the Session timeout interval specified in the Session manager properties. For the default timeout value of 30 minutes, the invalidation process interval is around 300 seconds. In this case, it could take up to 5 minutes (300 seconds) beyond the timeout threshold of 30 minutes for a particular session to become invalidated.

The value of this setting is used as a default when the session timeout is not specified in a Web module's deployment descriptor.

If you select **No timeout**, a session will be never removed from the memory unless explicit invalidation has been performed by the servlet. This can cause a memory leak when the user closes the window without logging out from the system. This option might be useful when sessions should be kept for a while until explicit invalidation has been done, when an employee leaves the company, for example. To use this option, make sure that enough memory or space in a persistent store is kept to accommodate all sessions.

- ▶ Security integration:

When security integration is enabled, the session manager associates the identity of users with their HTTP sessions. See 12.9, “Session security” on page 661 for more information.

Note: Do not enable this property if the application server contains a Web application that has form-based login configured as the authentication method and the local operating system is the authentication mechanism. It will cause authorization failures when users try to use the Web application.

- ▶ Serialize session access:

This option is available to provide serialized access to the session in a given JVM. This ensures thread-safe access when the session is accessed by multiple threads. No special code is necessary for using this option. This option is not recommended when framesets are used heavily because it can affect performance.

An optional property, Maximum wait time, can be set to specify the maximum amount of time a servlet request waits on an HTTP session before continuing execution. The default is 5 seconds.

If you set the Allow access on timeout option, multiple servlet requests that have timed out concurrently will execute normally. If it is false, servlet execution aborts.

12.6 Session affinity

In a clustered environment, any HTTP requests associated with an HTTP session must be routed to the same Web application in the same JVM. This ensures that all of the HTTP requests are processed with a consistent view of the user's HTTP session. The exception to this rule is when the cluster member fails or has to be shut down.

WebSphere assures that session affinity is maintained in the following way: Each server ID is appended to the session ID. When an HTTP session is created, its ID is passed back to the browser as part of a cookie or URL encoding. When the browser makes further requests, the cookie or URL encoding will be sent back to the Web server. The Web server plug-in examines the HTTP session ID in the cookie or URL encoding, extracts the unique ID of the cluster member handling the session, and forwards the request.

This situation can be seen in Figure 12-3, where the session ID from the HTTP header, `request.getHeader("Cookie")`, is displayed along with the session ID from `session.getId()`. The application server ID is appended to the session ID from the HTTP header. The first four characters of HTTP header session ID are the cache identifier that determines the validity of cache entries.

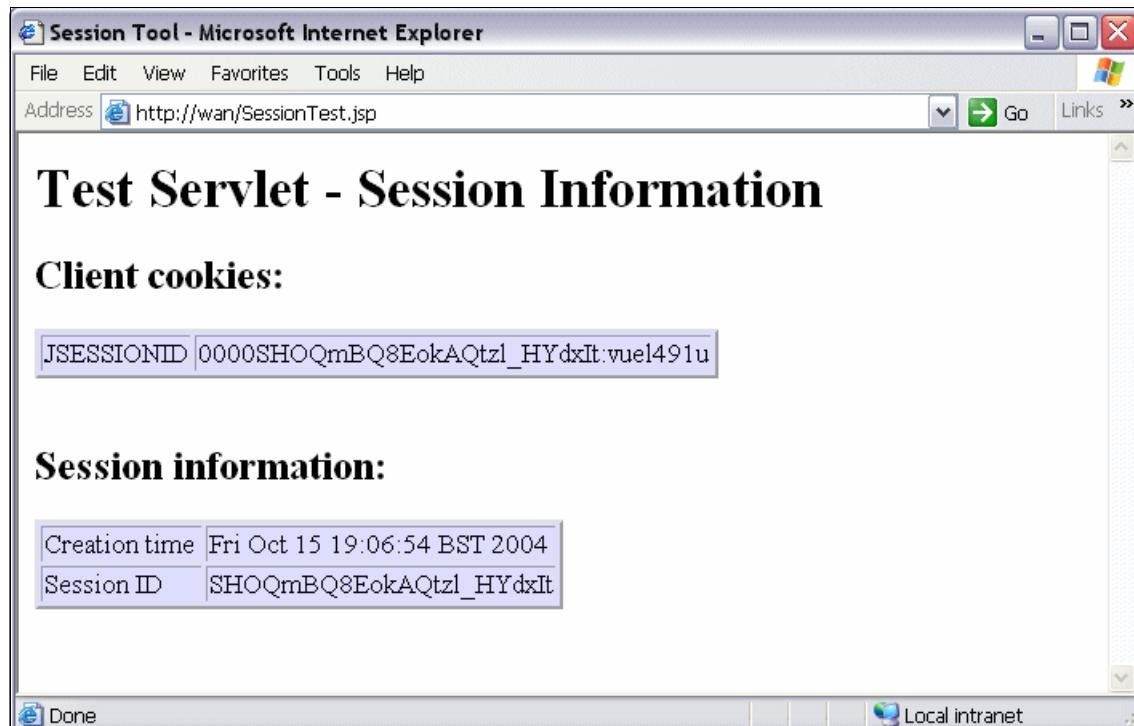


Figure 12-3 Session ID containing the server ID and cache ID

The JSESSIONID cookie can be divided into these parts: cache ID, session ID, separator, clone ID, and partition ID. JSESSION ID will include a partition ID instead of a clone ID when memory-to-memory replication in peer-to-peer mode is selected. Typically, the partition ID is a long numeric number.

Table 12-1 shows their mappings based on the example in Figure 12-3. A clone ID is an ID of a cluster member.

Table 12-1 Cookie mapping

content	value in the example
Cache ID	0000
Session ID	SHOQmBQ8EokAQtzl_HYdxlt
separator	:
Clone ID	vuel491u

The application server ID can be seen in the Web server plug-in configuration file, `plug-in-cfg.xml` file, as shown in Example 12-4.

Example 12-4 Server ID from plugin-cfg.xml file

```
<?xml version="1.0" encoding="ISO-8859-1"?><!--HTTP server plugin config file for the cell ITS0Cell generated on 2004.10.15 at 07:21:03 PM BST-->
<Config>
.....
<ServerCluster Name="MyCluster">
    <Server CloneID="vuel491u" LoadBalanceWeight="2" Name="NodeA_server1">
        <Transport Hostname="wan" Port="9080" Protocol="http"/>
        <Transport Hostname="wan" Port="9443" Protocol="https">
    .....
</Config>
```

Note: Session affinity can still be broken if the cluster member handling the request fails. To avoid losing session data, use persistent session management. In persistent sessions mode, cache ID and server ID will change in the cookie when there is a failover or when the session is read from the persistent store, so do not rely on the value of the session cookie remaining the same for a given session.

12.6.1 Session affinity and failover

Server clusters provide a solution for failure of an application server. Sessions created by cluster members in the server cluster share a common persistent session store. Therefore, any cluster member in the server cluster has the ability to see any user's session saved to persistent storage.

If one of the cluster members fails, the user can continue to use session information from another cluster member in the server cluster. This is known as failover. Failover works regardless of whether the nodes reside on the same machine or several machines. Only a single cluster member can control and access a given session at a time. See Figure 12-4.

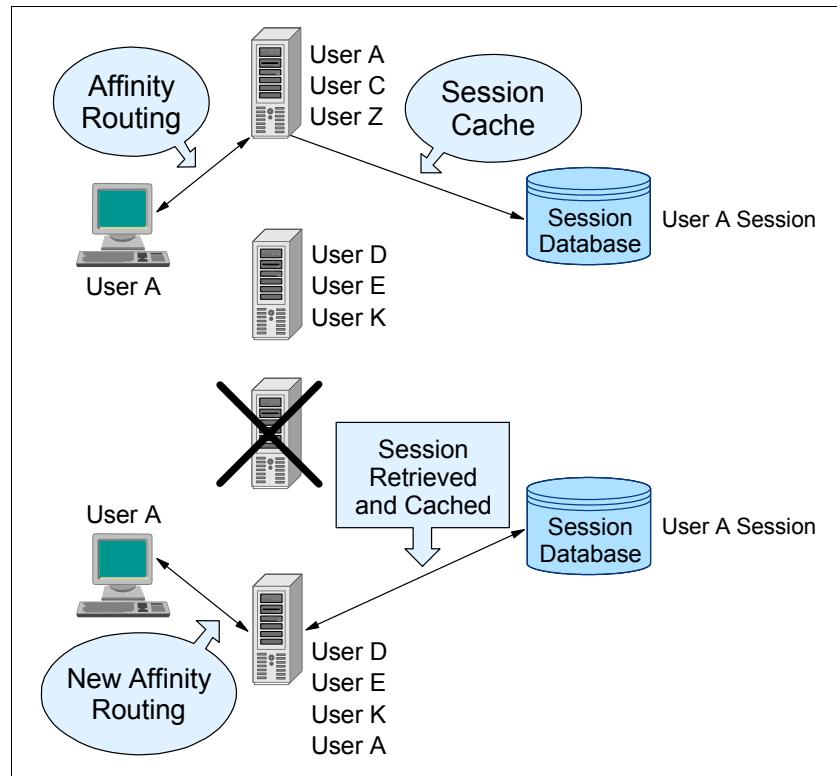


Figure 12-4 Session affinity and failover

After a failure, WebSphere redirects the user to another cluster member, and the user's session affinity switches to this replacement cluster member. After the initial read from the persistent store, the replacement cluster member places the user's session object in the in-memory cache, assuming that the cache has space available for additional entries.

The Web server plug-in maintains a cluster member list and picks the cluster member next in the list to avoid the breaking of session affinity. From then on, requests for that session go to the selected cluster member. The requests for the session go back to the failed cluster member when the failed cluster member restarts.

WebSphere provides session affinity on a best-effort basis. There are narrow windows where session affinity fails. These windows are as follows:

- ▶ When a cluster member is recovering from a crash, a window exists where concurrent requests for the same session could end up in different cluster members. The reason for this is that the Web server is multi-processed and each process separately maintains its own retry timer value and list of available cluster members. The end result is that requests being processed by different processes might end up being sent to more than one cluster member after at least one process has determined that the failed cluster member is running again.

To avoid or limit exposure in this scenario, if your cluster members are expected to crash very seldom and are expected to recover fairly quickly, consider setting the retry timeout to a small value. This narrows the window during which multiple requests being handled by different processes get routed to multiple cluster members.

- ▶ A server overload can cause requests belonging to the same session to go to different cluster members. This can occur even if all the cluster members are running. For each cluster member, there is a backlog queue where an entry is made for each request sent by the Web server plug-in waiting to be picked up by a worker thread in the servlet engine. If the depth of this queue is exceeded, the Web server plug-in starts receiving responses that the cluster member is not available. This failure is handled in the same way by the Web server plug-in as an actual JVM crash. Here are some examples of when this can happen:
 - The servlet engine does not have an appropriate number of threads to handle the user load.
 - The servlet engine threads take a long time to process the requests. Reasons for this include: applications taking a long time to execute, resources being used by applications taking a long time, and so on.

12.7 Persistent session management

By default, WebSphere places session objects in memory. However, the administrator has the option of enabling persistent session management, which instructs WebSphere to place session objects in a persistent store.

Administrators should enable persistent session management when:

- ▶ The user's session data must be recovered by another cluster member after a cluster member in a cluster fails or is shut down.
- ▶ The user's session data is too valuable to lose through unexpected failure at the WebSphere node.

- ▶ The administrator desires better control of the session cache memory footprint. By sending cache overflow to a persistent session store, the administrator controls the number of sessions allowed in memory at any given time.

There are two ways to configure session persistence as shown in Figure 12-5:

- ▶ Database persistence, supported for the Web container only
- ▶ Memory-to-memory session state replication using the data replication service available in distributed server environments

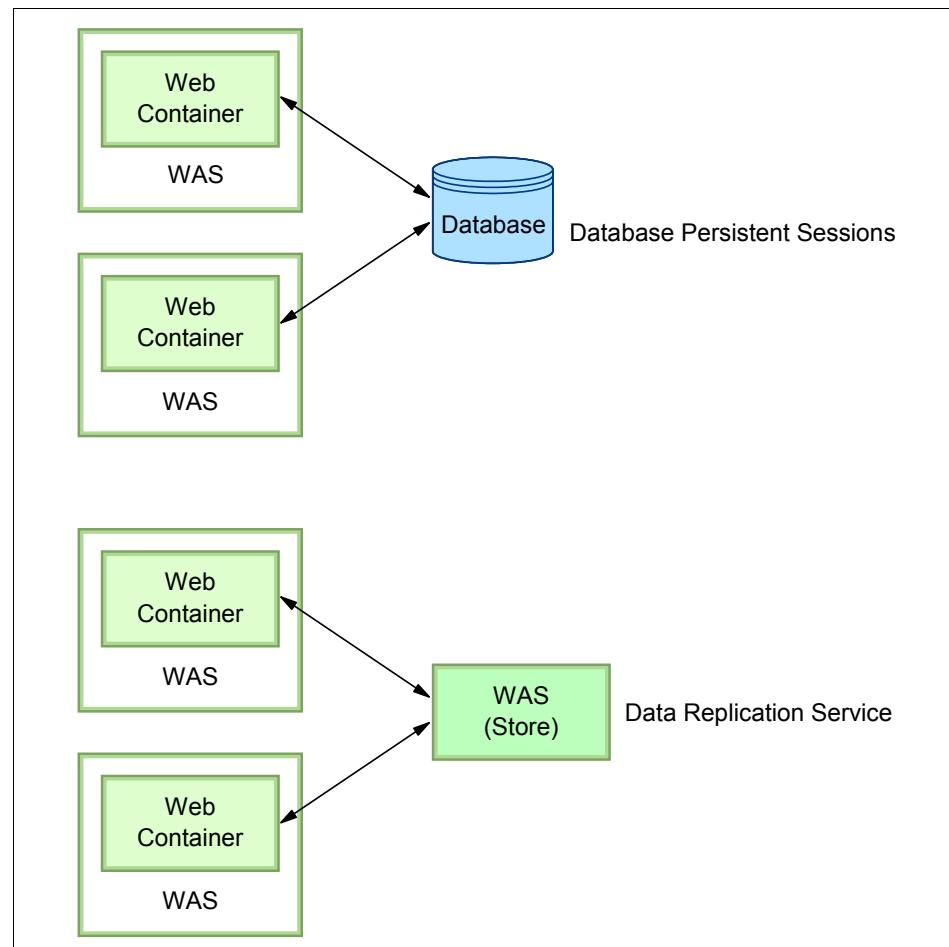


Figure 12-5 Persistent session options

All information stored in a persistent session store must be serialized. As a result, all of the objects held by a session must implement `java.io.Serializable` if the session needs to be stored in a persistent session store.

In general, consider making all objects held by a session serialized, even if immediate plans do not call for the use of persistent session management. If the website grows, and persistent session management becomes necessary, the transition between local and persistent management occurs transparently to the application if the sessions only hold serialized objects. If not, a switch to persistent session management requires coding changes to make the session contents serialized.

Persistent session management does not impact the session API, and Web applications require no API changes to support persistent session management. However, as mentioned previously, applications storing unserializable objects in their sessions require modification before switching to persistent session management.

If you use database persistence, using multi-row sessions becomes important if the size of the session object exceeds the size for a row, as permitted by the WebSphere session manager. If the administrator requests multi-row session support, the WebSphere session manager breaks the session data across multiple rows as needed. This allows WebSphere to support large session objects. Also, this provides a more efficient mechanism for storing and retrieving session contents under certain circumstances. See 12.7.5, “Single and multi-row schemas (database persistence)” on page 654 for information about this feature.

Using a cache lets the session manager maintain a cache of most recently used sessions in memory. Retrieving a user session from the cache eliminates a more expensive retrieval from the persistent store. The session manager uses a *least recently used* scheme for removing objects from the cache. Session data is stored to the persistent store based on your selections for write frequency and write option.

12.7.1 Enabling database persistence

We assume in this section that the following tasks have already been completed before enabling database persistence:

1. Create a session database.
2. (z/OS DB2) Create a table for the session data. Name the table SESSIONS. If you choose to use another name, update the Web container custom property SessionTableName value to the new table name. Grant ALL authority for the server region user ID to the table. An example of creating the table can be found in the Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tprs_db2tzos.html

3. In distributed environments, the session table will be created automatically for you when you define the data source for the database as the session management table; however, if you want to use a page (row) size greater than 4 KB, you will need to create the tablespace manually. An example of creating the tablespace can be found in the Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tprs_db2t.html

4. Create a JDBC provider and data source for the database.

The data source should be non-XA enabled. and must be a non-JTA enabled data source.

The JNDI name will be used to specify the database for persistence (in this example, jdbc/Sessions).

A summary of the data source selections for a DB2 database on z/OS is shown at the end of the wizard (Figure 12-6).

Options	Values
Scope	cells:WTCCell
Data source name	SessionDb
JNDI name	jdbc/Sessions
JDBC provider name	DB2 Universal JDBC Driver Provider
Description	One-phase commit DB2 JCC provider that supports JTA. Data sources that use this provider support only 1-phase commit processing, unless you use driver type 2 with application server for z/OS. If you use the application for z/OS, driver type 2 uses RRS and supports 2-phase processing.
Class path	\${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar \${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar \${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cus.jar
	/pp/db2v8/UK39204/jcc/classes
Native path	
Implementation class name	com.ibm.db2.jcc.DB2ConnectionPoolDataSource
Driver type	2
Database name	DB8X
Server name	wtsc04.itso.ibm.com
Port number	33760
Use this data source in container managed persistence (CMP)	true
Component-managed authentication alias	WTdmNode/jdbc/db8x
Mapping-configuration alias	(none)
Container-managed authentication alias	WTdmNode/jdbc/db8x

Figure 12-6 Data source creation summary

Note: The following example illustrates the steps to enable database persistence at the application server level. Session management settings can also be performed at the enterprise application level and the Web application level.

To enable database persistence, repeat the following steps for each application server:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. Select the server.
3. Click **Session management** under the Container Settings section.
4. Click **Distributed environment settings**.
5. Select **Database** and click **Database**. See Figure 12-7.

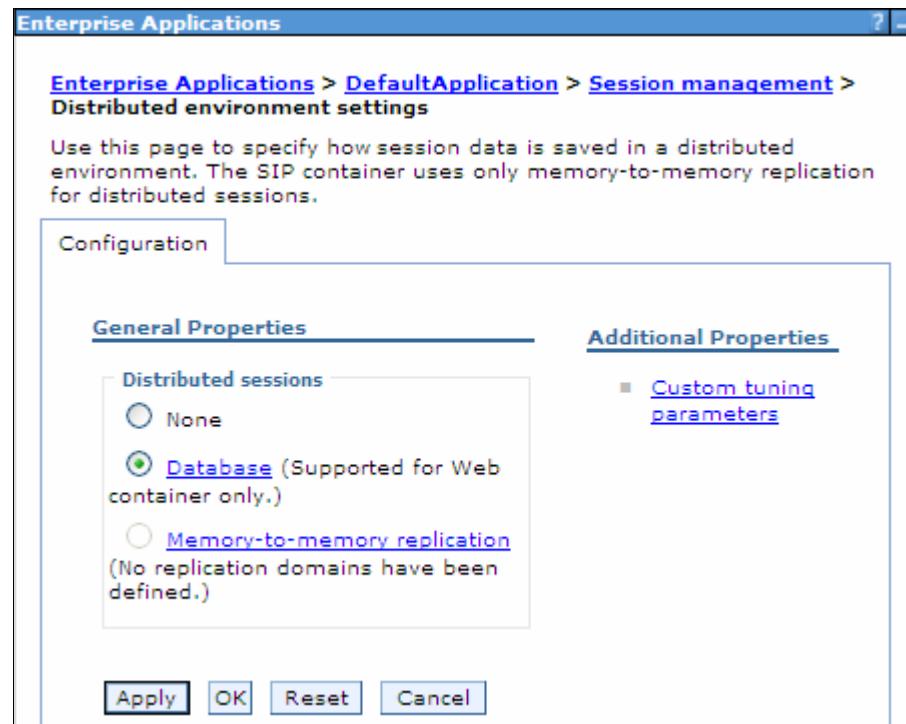


Figure 12-7 Distributed Environment Setting (database)

6. Enter the database information:
 - **(z/OS)** Enter the data source JNDI name (Figure 12-8).

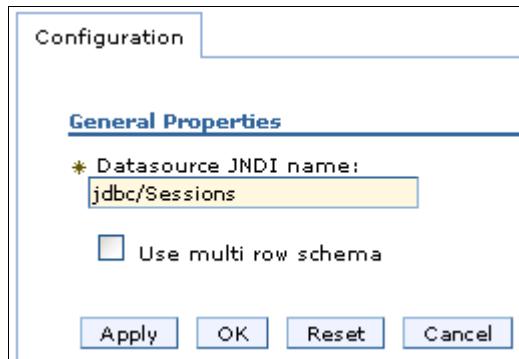


Figure 12-8 Data Source name

Click **OK**.

– **In distributed platforms:**

Enter the information required to access the database (Figure 12-9).

If you are using DB2 and you anticipate requiring row sizes greater than 4 KB, select the appropriate value from the DB2 row size menu. If the DB2 row size is other than 4 KB, you are required to enter the name of tablespace. See “Larger DB2 page sizes and database persistence” on page 653.

If you intend to use a multi-row schema, select **Use Multi row schema**. See 12.7.5, “Single and multi-row schemas (database persistence)” on page 654 for more information.

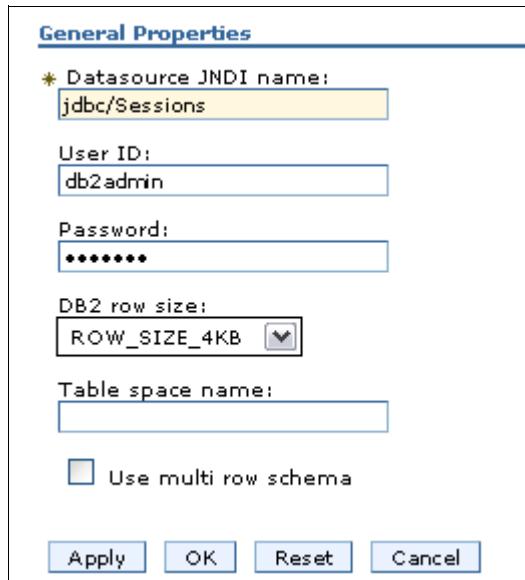


Figure 12-9 Database settings for session persistence

Click **OK**.

7. Click **OK** to accept the changes to the distribute session configuration.
8. Repeat this for each server in the cluster and save the configuration changes, synchronize them with the servers, and restart the application servers.

12.7.2 Memory-to-memory replication

Memory-to-memory replication uses the data replication service to replicate data across many application servers in a cluster without using a database. Using this method, sessions are stored in the memory of an application server, providing the same functionality as a database for session persistence. Separate threads handle this functionality within an existing application server process.

The data replication service is an internal WebSphere Application Server component. In addition to its use by the session manager, it is also used to replicate dynamic cache data and stateful session beans across many application servers in a cluster.

Using memory-to-memory replication eliminates the overhead and cost of setting up and maintaining a real-time production database. It also eliminates the single point of failure that can occur with a database. Session information between application servers is encrypted.

Note: Memory-to-memory replication requires the HA manager to be active. HA manager is active by default, but can be disabled. For more information, see *When to use a high availability manager* at:

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

Data replication service modes

The memory-to-memory replication function is accomplished by the creation of a data replication service instance in an application server that communicates to other data replication service instances in remote application servers.

You can set up a replication service instance to run in three possible modes:

- ▶ Server mode:
In this mode, a server only stores backup copies of other application server sessions. It does not send copies of sessions created in that particular server.
- ▶ Client mode:
In this mode, a server only broadcasts or sends copies of the sessions it owns. It does not receive backup copies of sessions from other servers.
- ▶ Both mode:
In this mode, the server simultaneously sends copies of the sessions it owns, and acts as a backup table for sessions owned by other application servers.

You can select the replication mode of server, client, or both when configuring the session management facility for memory-to-memory replication. The default is both modes.

With respect to mode, these are the primary examples of memory-to-memory replication configuration:

- ▶ Peer-to-peer replication
- ▶ Client/server replication

Although the administrative console allows flexibility and additional possibilities for memory-to-memory replication configuration, only these configurations are officially supported.

There is a single replica in a cluster by default. You can modify the number of replicas through the replication domain.

Peer-to-peer topology

Figure 12-10 shows an example of peer-to-peer topology. Each application server stores sessions in its own memory. It also stores sessions to and retrieves sessions from other application servers. Each application server acts as a client by retrieving sessions from other application servers. Each application server acts as a server by providing sessions to other application servers.

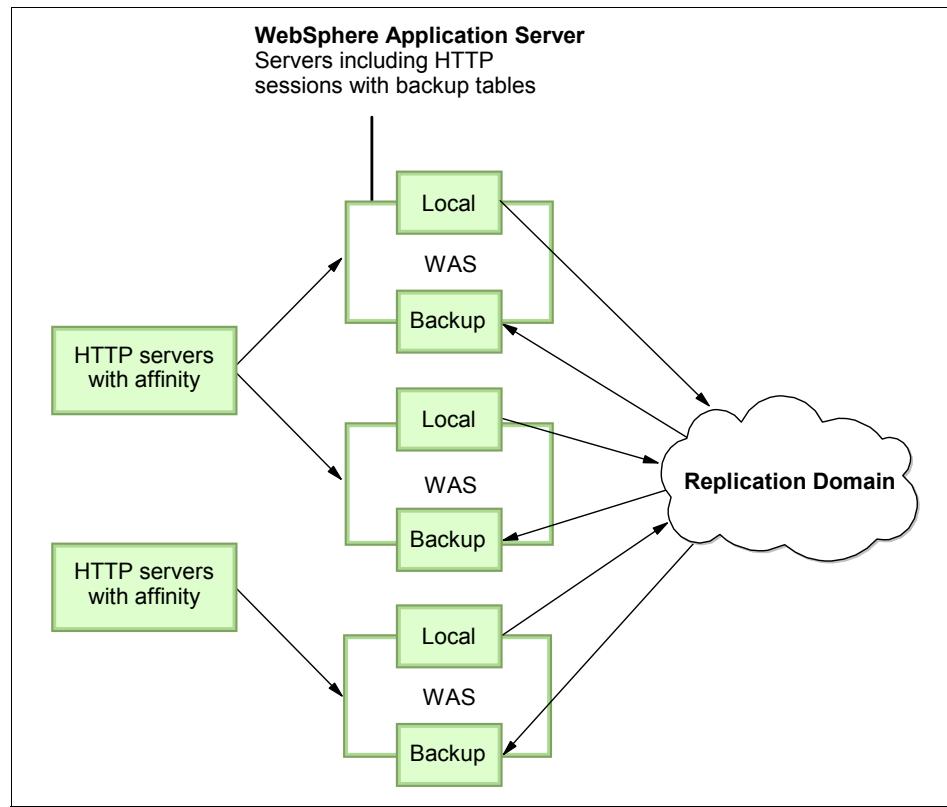


Figure 12-10 Example of peer-to-peer topology

The basic peer-to-peer (both mode) topology is the default configuration and has a single replica. However, you can also add additional replicas by configuring the replication domain.

In this basic peer-to-peer topology, each application server can:

- ▶ Host the Web application leveraging the HTTP session
- ▶ Send changes to the HTTP session that it owns
- ▶ Receive backup copies of the HTTP session from all of the other servers in the cluster

This configuration represents the most consolidated topology, where the various system parts are collocated and requires the fewest server processes. When using this configuration, the most stable implementation is achieved when each node has equal capabilities (CPU, memory, and so on), and each handles the same amount of work.

The advantage of this topology is that no additional processes and products are required to avoid a single point of failure. This reduces the time and cost required to configure and maintain additional processes or products.

One of the disadvantages of this topology is that it can consume large amounts of memory in networks with many users, because each server has a copy of all sessions. For example, assuming that a single session consumes 10 KB and one million users have logged into the system, each application server consumes 10 GB of memory in order to keep all sessions in its own memory. Another disadvantage is that every change to a session must be replicated to all application servers. This can cause a performance impact.

Client/server topology

Figure 12-11 shows an example of client/server topology. In this setup, application servers act as either a replication client or a server. Those that act as replication servers store sessions in their own memory and provide session information to clients. They are dedicated replication servers that just store sessions but do not respond to the users' requests. Client application servers send session information to the replication servers and retrieve sessions from the servers. They respond to user requests and store only the sessions of the users with whom they interact.

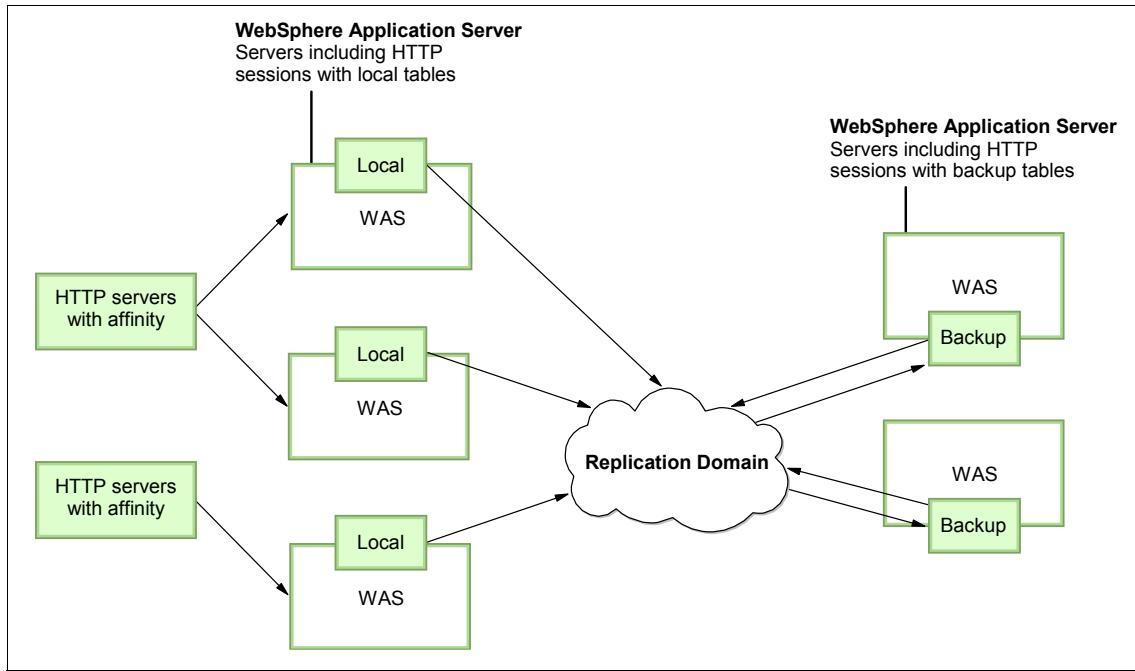


Figure 12-11 Example of client/server topology

The advantage of this topology is that it clearly distinguishes the role of client and server. Only replication servers keep all sessions in their memory and only the clients interact with users. This reduces the consumption of memory on each application server and reduces the performance impact, because session information is only sent to the servers.

You can recycle a backup server without affecting the servers running the application. When there are two or more backups, failure recovery is possible. Conversely, you can recycle an application server without losing the backup data.

When running Web applications on lower-end hardware, you can choose to have one or two more powerful computers that have the capacity to run a couple of session managers in replication server mode, allowing you to reduce the load on the Web application hardware.

One of the disadvantages of this topology is that additional application servers have to be configured and maintained over and above those that interact with users. We recommend that you have multiple replication servers configured to avoid a single point of failure.

Replication domain

The memory-to-memory replication function is accomplished by the creation of a data replication service instance in an application server that communicates to other data replication service instances in remote application servers. You must configure this data replication service instance as a part of a replication domain.

Data replication service instances on disparate application servers that replicate to one another must be configured as a part of the same domain. You must configure all session managers connected to a replication domain to have the same topology. If one session manager instance in a domain is configured to use the client/server topology, then the rest of the session manager instances in that domain must be a combination of servers configured as Client only and Server only.

If one session manager instance is configured to use the peer-to-peer topology, then all session manager instances must be configured as both client and server. For example, a server-only data replication service instance and a both client and server data replication service instance cannot exist in the same replication domain. Multiple data replication service instances that exist on the same application server due to session manager memory-to-memory configuration at various levels that are configured to be part of the same domain must have the same mode.

You should create a separate replication domain for each consumer. For example, create one replication domain for session manager and another replication domain for dynamic cache. The only situation where you should configure one replication domain is when you configure session manager replication and stateful session bean failover. Using one replication domain in this case ensures that the backup state information of HTTP sessions and stateful session beans are on the same application servers.

New in V7: A replication domain created with WebSphere Application Server V6.1 is referred to as a *multi-broker domain*. This type of replication domain consists of replicator entries. This is deprecated in WebSphere Application Server V7 and supported only for backward compatibility. Multi-broker replication domains do not communicate with each other, so migrate any multi-broker replication domains to the new data replication domains. You cannot create a multi-broker domain or replicator entries in the administrative console of WebSphere Application Server V7.

Enabling memory-to-memory replication

We assume in this section that the following tasks have already been completed before enabling data for the replication service:

1. You have created a cluster consisting of at least two application servers.
In this example, we are working with a cluster called MyCluster. It has two servers, server1 and server2.
2. You have installed applications to the cluster.

Note: This example illustrates setting up the replication domain and replicators after the cluster has been created. You also have the option of creating the replication domain and the replicator in the first server in the cluster when you create the cluster.

To enable memory-to-memory replication, do the following steps:

1. Create a replication domain to define the set of replicator processes that communicate with each other.
 - a. Select **Environment** → **Replication domains**. Click **New**. See Figure 12-12, and enter information in the fields.

General Properties

* Name
MyClusterRepDomain

* Request timeout
5 seconds

Encryption

Encryption type
none

Regenerate encryption key

Number of replicas

Single replica

Entire Domain

Specify

Number of replicas

Apply OK Reset Cancel

The screenshot displays a configuration dialog titled "General Properties". It includes fields for "Name" (set to "MyClusterRepDomain") and "Request timeout" (set to "5 seconds"). An "Encryption" section contains a dropdown menu set to "none" and a button labeled "Regenerate encryption key". A "Number of replicas" section has three radio button options: "Single replica" (selected), "Entire Domain", and "Specify". Below this is a text input field for "Number of replicas". At the bottom are four buttons: "Apply", "OK", "Reset", and "Cancel".

Figure 12-12 Create a replication domain

– Name:

At a minimum, you need to enter a name for the replication domain. The name must be unique within the cell. In this example, we used MyClusterRepDomain as the name, and defaults are used for the other properties.

– Encryption:

Encrypted transmission achieves better security but can impact performance. If DES or TRIPLE_DES is specified, a key for data transmission is generated. We recommend that you generate a key by clicking the **Regenerate encryption key** button periodically to enhance security.

- Number of replicas:

A single replica allows you to replicate a session to only one other server. This is the default. When you choose this option, a session manager picks another session manager connected to the same replication domain with which to replicate the HTTP session during session creation. All updates to the session are replicated to that single server. This option is set at the replication domain level.

When this option is set, every session manager connected to this replication domain creates a single backup copy of the HTTP session state information on a backup server.

Alternatively, you can replicate to every application server that is configured as a consumer of the replication domain with the **Entire Domain** option or to a specified number of replicas within the domain.

- b. Click **Apply**.
 - c. Click **OK**.
 - d. Save the configuration changes.
2. Configure the cluster members.
Repeat the following steps for each application server:
 - a. Select **Servers** → **Server Types** → **WebSphere application servers**.
 - b. Click the application server name. In this example, **server1** and **server2** are selected as application servers respectively.
 - c. Click **Session management** in the Container settings section.
 - d. Click **Distributed environment settings**.
 - e. Click on **Memory-to-memory replication** (Figure 12-13).

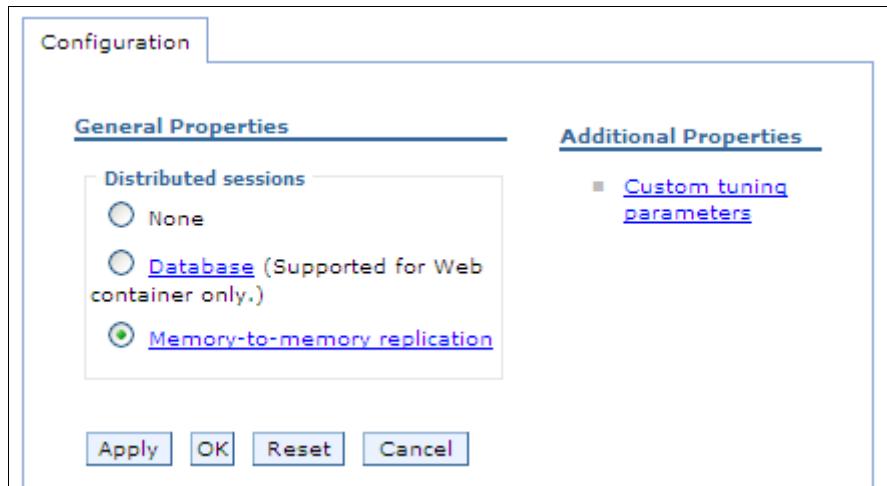


Figure 12-13 Distributed environment settings

- f. Choose a replicator domain. See Figure 12-14.

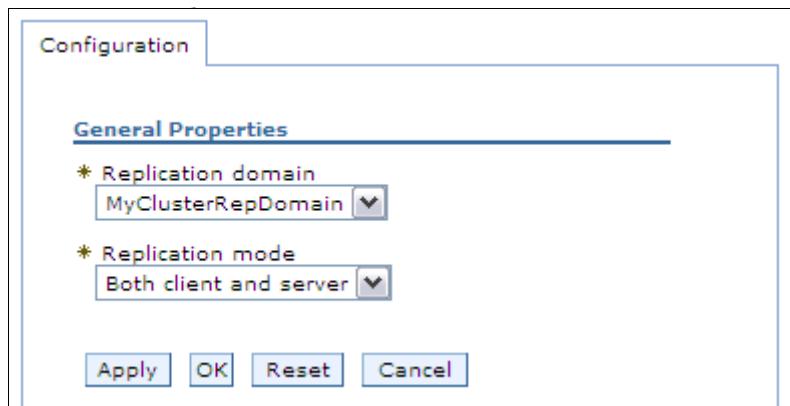


Figure 12-14 Data replication service settings

Select the replication topology by specifying the replication mode. Selecting **Both client and server** identifies this as a peer-to-peer topology. In a client/server topology, select **Client only** for application servers that will be responding to user requests. Select **Server only** for those that will be used as replication servers.

- g. Click **OK**.
3. Repeat the previous steps for the rest of the application servers in the cluster.

4. Save the configuration and restart the cluster. You can restart the cluster by selecting **Servers** → **Clusters** → **WebSphere application server clusters**. Check the cluster, and click **Stop**. After the messages indicate the cluster has stopped, click **Start**.

HTTP session replication in the z/OS controller

WebSphere Application Server V7 for z/OS can store replicated HTTP session data in the controller and replicate data to other WebSphere Application Servers. HTTP session data stored in a controller is retrievable by any of the servants of that controller. HTTP session affinity is still associated to a particular servant; however, if that servant should fail, any of the other servants can retrieve the HTTP session data stored in the controller and establish a new affinity.

The capability of storing HTTP sessions in the controller can also be enabled in unmanaged application servers on z/OS. When this capability is enabled, servants store the HTTP session data in the controller for retrieval when a servant fails which is similar to managed servers. HTTP session data stored in the controller of an unmanaged application server is not retrievable by other application servers and is not replicated to other application servers.

To store HTTP session data in the controller in an unmanaged application server:

1. Select to **Servers** → **Server Types** → **WebSphere application servers** → ***server_name***.
2. Under Server Infrastructure, click **Java and Process Management** → **Process Definition** → **Servant** → **Java Virtual Machine** → **Custom Properties**.
3. In the name field put `HttpSessionEnableUnmanagedServerReplication` and true on value field. (Figure 12-15).

[Application servers](#) > [WTS01A](#) > [Process definition](#) > [Servant](#) > [Java Virtual Machine](#) > [Custom properties](#)

Use this page to specify an arbitrary name and value pair. The value that is specified for the name and value pair is a string that can set internal system configuration properties.

Preferences

New	Delete
<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>

Select	Name	Value	Description
<input type="checkbox"/>	HttpSessionEnableUnmanagedServerReplication	true	Enable HTTP session data in the controller

Figure 12-15 Enable HTTP session in the controller

4. Save and restart the application server.

12.7.3 Session management tuning

Performance tuning for session management persistence consists of defining the following conditions:

- ▶ How often session data is written (write frequency settings).
- ▶ How much data is written (write contents settings).
- ▶ When the invalid sessions are cleaned up (session cleanup settings).

These settings are set in the Custom tuning parameters found under the Additional properties section for session management settings. Several combinations of these settings are predefined and available for selection, or you can customize them. These options are available on **Servers** → **Server Types** → **WebSphere application servers** → **server_name** → **Session management** → **Distributed environment settings** → **Custom Tuning parameters**.

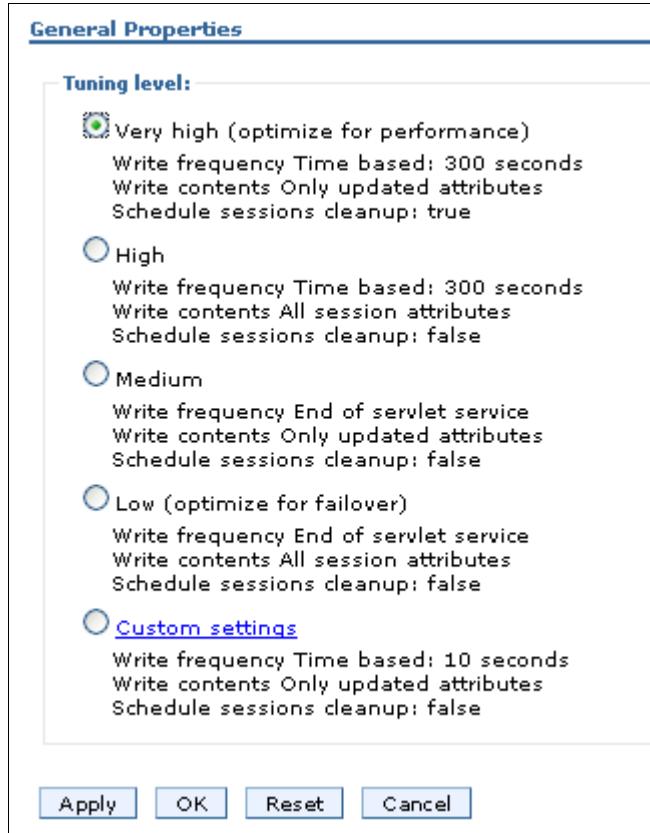


Figure 12-16 Distributed session management tuning levels

Custom settings

You can customize the settings by selecting the **custom settings option**. The settings can be seen in Figure 12-17.

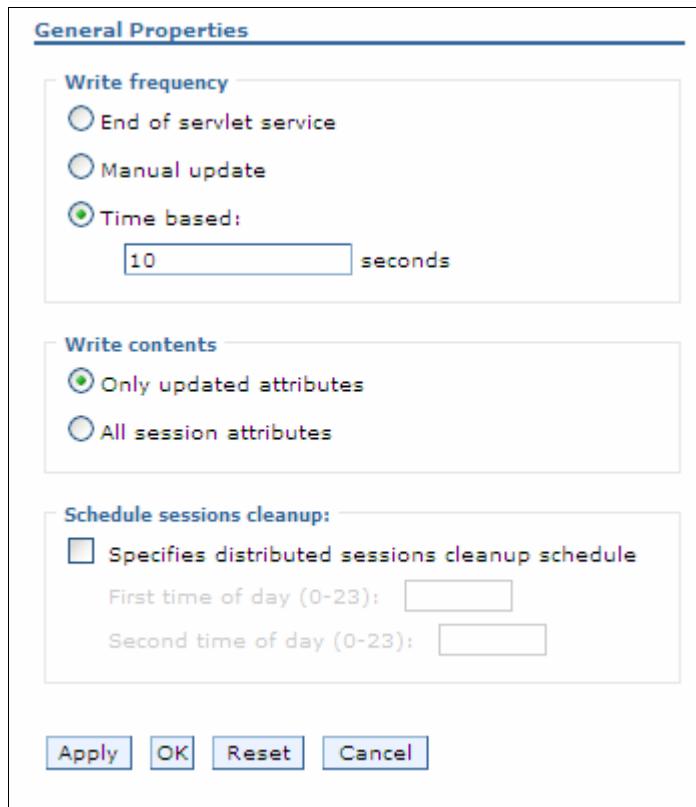


Figure 12-17 Session management tuning parameters

The following sections go into more detail on these custom settings.

Writing frequency settings

You can select from three different settings that determine how often session data is written to the persistent data store:

- ▶ **End of servlet service:**
If the session data has changed, it will be written to the persistent store after the servlet finishes processing an HTTP request.
- ▶ **Manual update:**
The session data will be written to the persistent store when the sync() method is called on the IBMSSession object.
- ▶ **Time-based:**
The session data will be written to the persistent store based on the specified write interval value.

Note: The **last access time** attribute is always updated each time the session is accessed by the servlet or JSP, whether the session is changed or not. This is done to make sure the session does not time out:

- ▶ If you choose the end of servlet service option, each servlet or JSP access will result in a corresponding persistent store update of the last access time.
- ▶ If you select the manual update option, the update of the last access time in persistent store occurs on sync() call or at later time.
- ▶ If you use time-based updates, the changes are accumulated and written in a single transaction. This can significantly reduce the amount of I/O to the persistent store.

See 12.10.2, “Reducing persistent store I/O” on page 666 for options to change this database update behavior.

Consider an example where the Web browser accesses the application once every five seconds:

- ▶ In End of servlet service mode, the session would be written out every five seconds.
- ▶ In Manual update mode, the session would be written out whenever the servlet issues `IBMSession.sync()`. It is the responsibility of the servlet writer to use the `IBMSession` interface instead of the `HttpSession` Interface and the servlets/JSPs must be updated to issue the sync().
- ▶ In Time-based mode, the servlet or JSP need not use the `IBMSession` class nor issue `IBMSession.sync()`. If the write interval is set to 120 seconds, then the session data is written out at most every 120 seconds.

End of servlet service

When the write frequency is set to the end of servlet service option, WebSphere writes the session data to the persistent store at the completion of the `HttpServlet.service()` method call. The write content settings determine output.

Manual update

In manual update mode, the session manager only sends changes to the persistent data store if the application explicitly requests a save of the session information.

Note: Manual updates use an IBM extension to `HttpSession` that is not part of the Servlet 2.5 API.

Manual update mode requires an application developer to use the `IBMSession` class for managing sessions. When the application invokes the `sync()` method, the session manager writes the modified session data and last access time to the persistent store. The session data written to the persistent store is controlled by the write contents option selected.

If the servlet or JSP terminates without invoking the `sync()` method, the session manager saves the contents of the session object into the session cache (if caching is enabled), but does not update the modified session data in the session database. The session manager will only update the last access time in the persistent store asynchronously, at later time. Example 12-5 shows how the `IBMSession` class can be used to manually update the persistent store.

Example 12-5 Using `IBMSession` for manual update of the persistent store

```
public void service (HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
{
    // Use the IBMSession to hold the session information
    // We need the IBMSession object because it has the manual update
    // method sync()
    com.ibm.websphere.servlet.session.IBMSession session =
        (com.ibm.websphere.servlet.session.IBMSession)req.getSession(true);

    Integer value = 1;

    //Update the in-memory session stored in the cache
    session.putValue("MyManualCount.COUNTER", value);

    //The servlet saves the session to the persistent store
    session.sync();
}
```

This interface gives the Web application developer additional control over when and if session objects go to the persistent data store. If the application does not invoke the `sync()` method, and manual update mode is specified, the session updates go only to the local session cache, not the persistent data store. Web developers use this interface to reduce unnecessary writes to the session database, and thereby to improve overall application performance.

All servlets in the Web application server must perform their own session management in manual update mode.

Time-based writes to the session database

Using the time-based write option will write session data to the persistent store at a defined write interval. The reasons for implementing time-based write lies in the changes introduced with the Servlet 2.2 API. The Servlet 2.2 specification introduced two key concepts:

- ▶ It limits the scope of a session to a single Web application.
- ▶ It both explicitly prohibits concurrent access to an HttpSession from separate Web applications, and allows for concurrent access within a given JVM.

Because of these changes, WebSphere provides the session affinity mechanism that assures an HTTP request is routed to the Web application handling its HttpSession. This assurance still holds in a WLM environment when using persistent HttpSession. This means that the necessity to immediately write the session data to the persistent store can now be relaxed somewhat in these environments, as well as non-clustered environments, because the persistent store is used now only for failover and session cache full scenarios.

With this in mind, it is now possible to gain potential performance improvements by reducing the frequency of persistent store writes.

Note: Time-based writes requires session affinity for session data integrity.

The following details apply to time-based writes:

- ▶ The expiration of the write interval does not necessitate a write to the persistent store unless the session has been touched (`getAttribute/setAttribute/removeAttribute` was called since the last write).
- ▶ If a session write interval has expired and the session has only been retrieved (`request.getSession()` was called since the last write), then the last access time will be written to the persistent store regardless of the write contents setting.
- ▶ If a session write interval has expired and the session properties have been either accessed or modified since the last write, then the session properties will be written in addition to the last access time. Which session properties get written is dependent on the write contents settings.
- ▶ Time-based write allows the servlet or JSP to issue `IBMSession.sync()` to force the write of session data to the database.
- ▶ If the time between session servlet requests for a particular session is greater than the write interval, then the session effectively gets written after each service method invocation.

- ▶ The session cache should be large enough to hold all of the active sessions. Failure to do this will result in extra persistent store writes, because the receipt of a new session request can result in writing out the oldest cached session to the persistent store. To put it another way, if the session manager has to remove the least recently used HttpSession from the cache during a full cache scenario, the session manager will write that HttpSession using the Write contents settings upon removal from the cache.
- ▶ The session invalidation time must be at least twice the write interval to ensure that a session does not inadvertently get invalidated prior to getting written to the persistent store.
- ▶ A newly created session will always be written to the persistent store at the end of the service method.

Writing content settings

These options control what is written. See 12.7.6, “Contents written to the persistent store using a database” on page 656 before selecting one of the options, because there are several factors to decide. The options available are:

- ▶ Only updated attributes are written to the persistent store.
- ▶ All session attributes are written to the persistent store.

Session cleanup settings

WebSphere allows the administrator to defer (to off-hours) the clearing of invalidated sessions from the persistent store. *Invalidated sessions* are sessions that are no longer in use and timed out. For more information, see 12.8, “Invalidate sessions” on page 659.

Select **Specifies distributed sessions cleanup schedule** to enable this option.

The cleanup can be done either once or twice a day. The following fields are available:

- ▶ First time of day (0-23) is the first hour, during which the invalidated persistent sessions will be cleared from the persistent store. This value must be a positive integer between 0 and 23.
- ▶ Second time of day (0-23) is the second hour, during which the invalidated persistent sessions will be cleared from the persistent store. This value must be a positive integer between 0 and 23.

Consider using scheduled invalidation for intranet-style applications that have a somewhat fixed number of users wanting the same HTTP session for the whole business day.

12.7.4 Larger DB2 page sizes and database persistence

WebSphere supports 4 KB, 8 KB, 16 KB, and 32 KB page sizes, and can have larger varchar for bit data columns of 7 KB, 15 KB, or 31 KB. Using this performance feature, we see faster persistence for HttpSession of sizes of 7 KB to 31 KB in the single-row case, or attribute sizes of 4 KB to 31 KB in the multi-row case.

Enabling the use of greater than 4K page size involves dropping any existing table created with a 4 KB buffer pool and tablespace. This also applies if you subsequently change between 4 KB, 8 KB, 16 KB, or 32 KB.

To use a page size other than the default 4 KB, do the following steps:

1. If the SESSIONS table already exists, drop it from the DB2 database:

```
DB2 connect to session  
DB2 drop table sessions
```

2. Create a new DB2 buffer pool and tablespace, specifying the same page size (8 KB, 16 KB, or 32 KB) for both, and assign the new buffer pool to this tablespace. Example 12-6 shows simple steps for creating an 8 KB page.

Example 12-6 Creating an 8K page size

```
DB2 connect to session  
DB2 CREATE BUFFERPOOL sessionBP SIZE 1000 PAGESIZE 8K  
DB2 connect reset  
DB2 connect to session  
DB2 CREATE TABLESPACE sessionTS PAGESIZE 8K MANAGED BY SYSTEM USING  
('D:\DB2\NODE000\SQL00005\sessionTS.0') BUFFERPOOL sessionBP  
DB2 connect reset
```

Refer to the DB2 product documentation for details.

3. Configure the correct tablespace name and page size, and DB2 row size, in the session management database configuration. See Figure 12-9 on page 635.

Restart WebSphere. On startup, the session manager creates a new SESSIONS table based on the page size and tablespace name specified.

12.7.5 Single and multi-row schemas (database persistence)

When using the single-row schema, each user session maps to a single database row. This is WebSphere's default configuration for persistent session management. With this setup, there are hard limits to the amount of user-defined, application-specific data that WebSphere Application Server can access.

When using the multi-row schema, each user session maps to multiple database rows, with each session attribute mapping to a database row.

In addition to allowing larger session records, using a multi-row schema can yield performance benefits, as discussed in 12.10.3, “Multirow persistent sessions: Database persistence” on page 666.

Switching from single-row to multi-row schema

To switch from single-row to multi-row schema for sessions, do the following steps:

1. Modify the session manager properties to switch from single to multi-row schema. Select the **Use multi row schema** on **Servers → Server Types → WebSphere application servers → server_name → Session management → Distributed environment settings → Database settings**.
2. Manually drop the database table or delete all the rows in the session database table. To drop the table:
 - a. Determine which data source the session manager is using. This is set in the session management distributed settings window. See 12.7.1, “Enabling database persistence” on page 631.
 - b. Look up the database name in the data source settings. Find the JDBC provider, then the data source. The database name is in the custom settings.
 - c. Use the database facilities to connect to the database and drop it.
3. Restart the application server or cluster.

Design considerations

Consider configuring direct, single-row usage to one database and multi-row usage to another database while you verify which option suits your application's specific needs. You can do this by switching the data source used, then monitoring the performance. Table 12-2 provides an overview.

Table 12-2 Single versus multi-row schemas

Programming issue	Application scenario
Reasons to use single-row	<ul style="list-style-type: none">▶ You can read/write all values with just one record read/write.▶ This takes up less space in a database, because you are guaranteed that each session is only one record long.
Reasons <i>not</i> to use single-row	<ul style="list-style-type: none">▶ There is a 2 MB limit of stored data per session. The sum of sizes of all session attributes is limited to 2 MB.
Reasons to use multi-row	<ul style="list-style-type: none">▶ The application can store an unlimited amount of data. You are limited only by the size of the database and a 2 MB-per-record limit. The size of each session attribute can be 2 MB.▶ The application can read individual fields instead of the whole record. When large amounts of data are stored in the session but only small amounts are specifically accessed during a given servlet's processing of an HTTP request, multi-row sessions can improve performance by avoiding unneeded Java object serialization.
Reasons <i>not</i> to use multi-row	<ul style="list-style-type: none">▶ If data is small in size, you probably do not want the extra overhead of multiple row reads when everything could be stored in one row.

In the case of multi-row usage, design your application data objects so they do not have references to each other. This is to prevent circular references.

For example, suppose you are storing two objects (A and B) in the session using `HttpSession.put(..)`, and A contains a reference to B. In the multi-row case, because objects are stored in different rows of the database, when objects A and B are retrieved later, the object graph between A and B is different from that stored. A and B behave as independent objects.

12.7.6 Contents written to the persistent store using a database

WebSphere supports two modes for writing session contents to the persistent store:

- ▶ Only updated attributes:

Write only the HttpSession properties that have been updated via `setAttribute()` and `removeAttribute()`.

- ▶ All session attributes:

Write all the HttpSession properties to the database.

These settings are found in the tuning parameters for distributed environment settings (select **custom settings**).

When a new session is initially created with either of the above two options, the entire session is written, including any Java objects bound to the session. When using database persistence, the behavior for subsequent servlet or JSP requests for this session varies depending on whether the single-row or multi-row database mode is in use:

- ▶ In single-row mode, choose from the following options:

- Only updated attributes:

If any session attribute has been updated, through `setAttribute` or `removeAttribute`, then all of the objects bound to the session will be written to the database.

- All session attributes:

All bound session attributes will be written to the database.

- ▶ In multi-row mode, choose from the following options:

- Only updated attributes

Only the session attributes that were specified via `setAttribute` or `removeAttribute` will be written to the database.

- All session attributes

All of the session attributes that reside in the cache will be written to the database. If the session has never left the cache, then this should contain all of the session attributes.

By using the All session attributes mode, servlets and JSPs can change Java objects that are attributes of the HttpSession without having to call `setAttribute()` on the HttpSession for that Java object in order for the changes to be reflected in the database.

Using the All session attributes mode provides some flexibility to the application programmer and protects against possible side effects of moving from local sessions to persistent sessions.

However, using All session attributes mode can potentially increase activity and be a performance drain. Individual customers will have to evaluate the pros and cons for their installation. It should be noted that the combination of All session attributes mode with time-based write could greatly reduce the performance penalty and essentially give you the best of both worlds.

As shown in Example 12-7 and Example 12-8, the initial session creation contains a setAttribute, but subsequent requests for that session do not need to use setAttribute.

Example 12-7 Initial servlet

```
HttpSession sess = request.getSession(true);
myClass myObject = new myClass();
myObject.someInt = 1;
sess.setAttribute("myObject", myObject); // Bind object to the session
```

Example 12-8 Subsequent servlet

```
HttpSession sess = request.getSession(false);
myObject = sess.getAttribute("myObject"); // get bound session object
myObject.someInt++; // change the session object
// setAttribute() not needed with write "All session attributes" specified
```

Example 12-9 and Example 12-10 show setAttribute is still required even though the write all session attributes option is enabled.

Example 12-9 Initial servlet

```
HttpSession sess = request.getSession(true);
String myString = new String("Initial Binding of Session Object");
sess.setAttribute("myString", myString); // Bind object to the session
```

Example 12-10 Subsequent servlet

```
HttpSession sess = request.getSession(false);
String myString = sess.getAttribute("myString"); // get bound session object
...
myString = new String("A totally new String"); // get a new String object
sess.setAttribute("myString", myString); // Need to bind the object to the session since a NEW Object is used
```

HttpSession set/getAttribute action summary

Table 12-3 summarizes the action of the HttpSession setAttribute and removeAttribute methods for various combinations of the row type, write contents, and write frequency session persistence options.

Table 12-3 Write contents versus write frequency

Row type	Write contents	Write frequency	Action for setAttribute	Action for removeAttribute
Single-row	Only updated attributes	End of servlet service / sync() call with Manual update	If any of the session data has changed, then write all of this session's data from cache. ¹	If any of the session data has changed, then write all of this session's data from cache. ¹
Single-row	Only updated attributes	Time-based	If any of the session data has changed, then write all of this session's data from cache. ¹	If any of the session data has changed, then write all of this session's data from cache. ¹
	All session attributes	End of servlet service / sync() call with Manual update	Always write all of this session's data from cache.	Always write all of this session's data from cache.
	All session attributes	Time-based	Always write all of this session's data from cache.	Always write all of this session's data from cache.
Multi-row	Only updated attributes	End of servlet service / sync() call with Manual update	Write only thread-specific data that has changed.	Delete only thread-specific data that has been removed.
	Only updated attributes	Time-based	Write thread-specific data that has changed for <i>all</i> threads using this session.	Delete thread-specific data that has been removed for <i>all</i> threads using this session.
	All session attributes	End of servlet service / sync() call with Manual update	Write all session data from cache.	Delete thread-specific data that has been removed for <i>all</i> threads using this session.
	All session attributes	Time-based	Write all session data from cache.	Delete thread-specific data that has been removed for <i>all</i> threads using this session.

Row type	Write contents	Write frequency	Action for setAttribute	Action for removeAttribute
¹ When a session is written to the database while using single-row mode, <i>all</i> of the session data is written. Therefore, no database deletes are necessary for properties removed with removeAttribute(), because the write of the entire session does not include removed properties.				

Multi-row mode has the notion of thread-specific data. *Thread-specific data* is defined as session data that was added or removed while executing under this thread. If you use End of servlet service or Manual update modes and enable **Only updated attributes**, then only the thread-specific data is written to the database.

12.8 Invalidating sessions

This section discusses invalidating sessions when the user no longer needs the session object, for example, when the user has logged off a site. Invalidating a session removes it from the session cache, as well as from the persistent store.

WebSphere offers the following methods for invalidating session objects:

- ▶ Programmatically, you can use the invalidate() method on the session object. If the session object is accessed by multiple threads in a Web application, be sure that none of the threads still have references to the session object.
- ▶ An invalidator thread scans for timed-out sessions every n seconds, where n is configurable from the administrative console. The session timeout setting is in the general properties of the session management settings.
- ▶ For persistent sessions, the administrator can specify times when the scan runs. This feature has the following benefits when used with persistent session:
 - Persistent store scans can be scheduled during periods that normally have low demand. This avoids slowing down online applications due to contention in the persistent store.
 - When this setting is used with the End of servlet service write frequency option, WebSphere does not have to write the last access time with every HTTP request. The reason is that WebSphere does not have to synchronize the invalidator thread's deletion with the HTTP request access.

You can find the schedule sessions cleanup setting in the Session management settings under the Custom tuning parameters for distributed environments (select **custom settings**).

If you are going to use session cleanup, be aware of these considerations:

- HttpSession timeouts are not enforced. Instead, all invalidation processing is handled at the configured invalidation times.
- With listeners, described in 12.8.1, “Session listeners”, processing is potentially delayed by this configuration. Session cleanup scheduling is not recommended if listeners are used.

12.8.1 Session listeners

Listener classes are defined to listen for state changes of a session and its attributes. This allows greater control over interactions with sessions, leading programmers to monitor creation, deletion, and modification of sessions.

Programmers can perform initialization tasks when a session is created, or clean up tasks when a session is removed. It is also possible to perform some specific tasks for the attribute when an attribute is added, deleted, or modified.

The following the Listener interfaces are used to monitor the events associated with the HttpSession object:

- ▶ `javax.servlet.http.HttpSessionListener`:
Use this interface to monitor creation and deletion, including session timeout, of a session.
- ▶ `javax.servlet.http.HttpSessionAttributeListener`:
Use this interface to monitor changes of session attributes, such as add, delete, and replace.
- ▶ `javax.servlet.http.HttpSessionActivationListener`:
This interface monitors activation and passivation of sessions. This interface is useful to monitor if the session exists, whether in memory or not, when persistent session is used.

Table 12-4 is a summary of the interfaces and methods.

Table 12-4 Listener interfaces and their methods

Target	Event	Interface	Method
session	create	HttpSessionListener	sessionCreated()
	destroy	HttpSessionListener	sessionDestroyed()
	activate	HttpSessionActivationListener	sessionDidActivate()
	passivate	HttpSessionActivationListener	sessionWillPassivate()

Target	Event	Interface	Method
attribute	add	HttpSessionAttributeListener	attributeAdded()
	remove	HttpSessionAttributeListener	attributeRemoved()
	replace	HttpSessionAttributeListener	attributeReplaced()

For more information, see *Java Platform Enterprise Edition, v 5.0 API Specifications* at:

<http://java.sun.com/javaee/5/docs/api/>

12.9 Session security

WebSphere Application Server V7 maintains the security of individual sessions. When session manager integration with WebSphere security is enabled, the session manager checks the user ID of the HTTP request against the user ID of the session held within WebSphere. This check is done as part of the processing of the `request.getSession()` function. If the check fails, WebSphere throws an `com.ibm.websphere.servlet.session.UnauthorizedSessionRequestException` exception. If it succeeds, the session data is returned to the calling servlet or JSP.

Session security checking works with the standard `HttpSession`. The identity or user name of a session can be accessed through the `com.ibm.websphere.servlet.session.IBMSession` interface. An unauthenticated identity is denoted by the user name *anonymous*.

The session manager uses WebSphere's security infrastructure to determine the authenticated identity associated with a client HTTP request that either retrieves or creates a session.

Security integration rules for HTTP sessions

Session management security has the following rules:

- ▶ Sessions in unsecured pages are treated as accesses by the anonymous user.
- ▶ Sessions created in unsecured pages are created under the identity of that anonymous user.
- ▶ Sessions in secured pages are treated as accesses by the authenticated user.

- ▶ Sessions created in secured pages are created under the identity of the authenticated user. They can only be accessed in other secured pages by the same user. To protect these sessions from use by unauthorized users, they cannot be accessed from an unsecure page. Do not mix access to secure and unsecure pages.

Table 12-5 lists possible scenarios when security integration is enabled, where outcomes depend on whether the HTTP request was authenticated and whether a valid session ID and user name was passed to the session manager.

Table 12-5 HTTP session security

Request session ID/ user name.	Unauthenticated HTTP request is used to retrieve the session.	Authenticated HTTP request is used to retrieve the session. The user ID in the HTTP request is FRED.
No session ID was passed in for this request, or the ID is for a session that is no longer valid.	A new session is created. The user name is anonymous.	A new session is created. The user name is FRED.
A valid session ID is received. The current session user name is anonymous.	The session is returned.	The session is returned. The session manager changes the user name to FRED.
A valid session ID is received. The current session user name is FRED.	The session is not returned. UnauthorizedSession-RequestException is thrown. ¹	The session is returned.
A valid session ID is received. The current session user name is BOB.	The session is not returned. UnauthorizedSession-RequestException is thrown. ¹	The session is not returned. UnauthorizedSession-RequestException is thrown. ¹
¹ com.ibm.websphere.servlet.session.UnauthorizedSessionRequestException is thrown to the servlet or JSP.		

See 12.5, “General properties for session management” on page 622 for more information about the security integration setting.

12.10 Session performance considerations

This section includes guidance for developing and administering scalable, high-performance Web applications using WebSphere Application Server V7 session support.

12.10.1 Session size

Large session objects pose several problems for a Web application. If the site uses session caching, large sessions reduce the memory available in the WebSphere instance for other tasks, such as application execution.

For example, assume that a given application stores 1 MB of information for each user session object. If 100 users arrive over the course of 30 minutes, and assume the session timeout remains at 30 minutes, the application server instance must allocate 100 MB just to accommodate the newly arrived users in the session cache:

$$1 \text{ MB for each user session} * 100 \text{ users} = 100 \text{ MB}$$

Note this number does not include previously allocated sessions that have not timed out yet. The memory required by the session cache could be considerably higher than 100 MB.

Web developers and administrators have several options for improving the performance of session management:

- ▶ Reduce the size of the session object.
- ▶ Reduce the size of the session cache.
- ▶ Add additional application servers.
- ▶ Invalidate unneeded sessions.
- ▶ Increase the memory available.
- ▶ Reduce the session timeout interval.

Reducing session object size

Web developers must consider carefully the information kept by the session object:

- ▶ Remove information easily obtained or easily derived to help keep the session object small.
- ▶ Remove unnecessary, unneeded, or obsolete data from the session.
- ▶ Consider whether it would be better to keep a certain piece of data in an application database rather than in the HTTP session. This gives the developer full control over when the data is fetched or stored and how it is combined with other application data. Web developers can leverage the power of SQL if the data is in an application database.

Reducing object size becomes particularly important when persistent sessions are used. Serializing a large amount of data and writing it to the persistent store requires significant WebSphere performance overhead. Even if the option to write only updated attributes is enabled, if the session object contains large Java objects or collections of objects that are updated regularly, there is a significant performance penalty in persisting these objects. This penalty can be reduced by using time-based writes.

Notes: In general, you can obtain the best performance with session objects that are less than 2 KB in size. When the session object exceeds 4-5 KB, you can expect a significant decrease in performance.

Even if session persistence is not an issue, minimizing the session object size will help to protect your Web application from scale-up disasters as user numbers increase. Large session objects will require more and more JVM memory, leaving no room to run servlets.

See 12.7.4, “Larger DB2 page sizes and database persistence” on page 653 to learn how WebSphere can provide faster persistence of larger session objects when using DB2.

Session cache size

The session manager allows administrators to change the session cache size to alter the cache’s memory footprint. By default, the session cache holds 1000 session objects. By lowering the number of session objects in the cache, the administrator reduces the memory required by the cache.

However, if the user’s session is not in the cache, WebSphere must retrieve it from either the overflow cache (local caching), or the session store (for persistent sessions). If the session manager must retrieve persistent sessions frequently, the retrievals can impact overall application performance.

WebSphere maintains overflowed local sessions in memory, as discussed in 12.4, “Local sessions” on page 620. Local session management with cache overflow enabled allows an unlimited number of sessions in memory. To limit the cache footprint to the number of entries specified in session manager, use persistent session management, or disable the overflow.

Note: When using local session management without specifying the Allow overflow property, a full cache will result in the loss of user session objects.

Creating additional application servers

WebSphere also gives the administrator the option of creating additional application servers. Creating additional instances spreads the demand for memory across more JVMs, thus reducing the memory burden on any particular instance. Depending on the memory and CPU capacity of the machines involved, the administrator can add additional instances within the same machine. Alternatively, the administrator can add additional machines to form a hardware cluster, and spread the instances across this cluster.

Note: When configuring a cluster, session affinity routing provides the most efficient strategy for user distribution within the cluster, even with session persistence enabled. With cluster members, the Web server plug-in provides affinity routing among cluster member instances.

Invalidating unneeded sessions

If the user no longer needs the session object, for example, when the user has logged out of the site, it should be invalidated. Invalidating a session removes it from the session cache, as well as from the session database. For more information, see 12.8, “Invalidating sessions” on page 659.

Increasing available memory

WebSphere allows the administrator to increase an application server’s heap size. By default, WebSphere allocates 256 MB as the maximum heap size. Increasing this value allows the instance to obtain more memory from the system, and thus hold a larger session cache.

A practical limit exists, however, for an instance heap size. The machine memory containing the instance needs to support the heap size requested. Also, if the heap size grows too large, the length of the garbage collection cycle with the JVM might impact overall application performance. This impact has been reduced with the introduction of multi-threaded garbage collection.

Session timeout interval

By default, each user receives a 30 minute interval between requests before the session manager invalidates the user’s session. Not every site requires a session timeout interval this generous. By reducing this interval to match the requirements of the average site user, the session manager purges the session from the cache and the persistent store, if enabled, more quickly.

Avoid setting this parameter too low and frustrating users. In some cases where the persistent store contains a large number of entries, frequent execution of the timeout scanner reduces overall performance. In cases where the persistent store contains many session entries, avoid setting the session timeout so low

that it triggers frequent, expensive scans of the persistent store for timed-out sessions. Alternatively, the administrator should consider schedule-based invalidation where scans for invalid object can be deferred to a time that normally has low demand. See 12.8, “Invalidating sessions” on page 659.

12.10.2 Reducing persistent store I/O

From a performance point of view, the Web developer's considerations are the following:

- ▶ Optimize the use of the HttpSession within a servlet. Only store the minimum amount of data required in HttpSession. Data that does not have to be recovered after a cluster member fails or is shut down can be best kept elsewhere, such as in a hash table. Recall that HttpSession is intended to be used as a *temporary* store for state information between browser invocations.
- ▶ Specify session=false in the JSP directive for JSPs that do not need to access the session object.
- ▶ Use time-based write frequency mode. This greatly reduces the amount of I/O, because the persistent store updates are deferred up to a configurable number of seconds. Using this mode, all of the outstanding updates for a Web application are written periodically based on the configured write interval.
- ▶ Use the Schedule sessions cleanup option. When using the End of servlet service write frequency mode, WebSphere does not have to write out the last access time with every HTTP request. This is because WebSphere does not have to synchronize the invalidator thread's deletion with the HTTP request's access.

12.10.3 Multirow persistent sessions: Database persistence

When a session contains multiple objects accessed by different servlets or JSPs in the same Web application, multi-row session support provides a mechanism for improving performance. Multi-row session support stores session data in the persistent session database by Web application and value. Table 12-6 shows a simplified representation of a multi-row database table.

Table 12-6 Simplified multi-row session representation

Session ID	Web application	Property	Small value	Large value
DA32242SSGE2	ShoeStore	ShoeStore.First.Name	Alice	
DA32242SSGE2	ShoeStore	ShoeStore.Last.Name	Smith	

Session ID	Web application	Property	Small value	Large value
DA32242SSGE2	ShoeStore	ShoeStore.Big.String		A big string....

In this example, if the user visits the ShoeStore application, and the servlet involved needs the user's first name, the servlet retrieves this information through the session API. The session manager brings into the session cache only the value requested. The ShoeStore.Big.String item remains in the persistent session database until the servlet requests it. This saves time by reducing both the data retrieved and the serialization overhead for data the application does not use.

After the session manager retrieves the items from the persistent session database, these items remain in the in-memory session cache. The cache accumulates the values from the persistent session database over time as the various servlets within the Web application request them. With WebSphere's session affinity routing, the user returns to this same cached session instance repeatedly. This reduces the number of reads against the persistent session database, and gives the Web application better performance.

How session data is written to the persistent session database has been made configurable in WebSphere. For information about session updates using single and multi-row session support, see 12.7.5, "Single and multi-row schemas (database persistence)" on page 654. Also see 12.7.6, "Contents written to the persistent store using a database" on page 656.

Even with multi-row session support, Web applications perform best if the overall contents of the session objects remain small. Large values in session objects require more time to retrieve from the persistent session database, generate more network traffic in transit, and occupy more space in the session cache after retrieval.

Multi-row session support provides a good compromise for Web applications requiring larger sessions. However, single-row persistent session management remains the best choice for Web applications with small session objects. Single-row persistent session management requires less storage in the database, and requires fewer database interactions to retrieve a session's contents (all of the values in the session are written or read in one operation). This keeps the session object's memory footprint small, as well as reducing the network traffic between WebSphere and the persistent session database.

Note: Avoid circular references within sessions if using multi-row session support. The multi-row session support does not preserve circular references in retrieved sessions.

12.10.4 Managing your session database connection pool

When using persistent session management, the session manager interacts with the defined database through a WebSphere Application Server data source.

Each data source controls a set of database connections known as a connection pool. By default, the data source opens a pool of no more than 10 connections. The maximum pool size represents the number of simultaneous accesses to the persistent session database available to the session manager.

For high-volume websites, the default settings for the persistent session data source might not be sufficient. If the number of concurrent session database accesses exceeds the connection pool size; the data source queues the excess requests until a connection becomes available. Data source queuing can impact the overall performance of the Web application (sometimes dramatically).

For best performance, the overhead of the connection pool used by the session manager needs to be balanced against the time that a client can spend waiting for an occupied connection to become available for use. By definition, a connection pool is a *shared* resource, so in general the best performance is realized typically with a connection pool that has significantly fewer connections than the number of simultaneous users.

A large connection pool does not necessarily improve application performance. Each connection represents memory overhead. A large pool decreases the memory available for WebSphere to execute applications. Also, if database connections are limited because of database licensing issues, the administrator must share a limited number of connections among other Web applications requiring database access as well. This is one area where performance tuning tests are required to determine the optimal setting for a given application.

As discussed above, session affinity routing combined with session caching reduces database read activity for session persistence. Likewise, manual update write frequency, time-based write frequency, and multi-row persistent session management reduce unnecessary writes to the persistent database. Incorporating these techniques can also reduce the size of the connection pool required to support session persistence for a given Web application.

Prepared statement caching is a connection pooling mechanism that can be used to further improve session database response times. A cache of previously prepared statements is available on a connection. When a new prepared statement is requested on a connection, the cached prepared statement is returned, if available. This caching reduces the number of costly prepared statements created, which improves response times. In general, base the prepared statement cache size on the following considerations:

- ▶ The smaller of:
 - Number of concurrent users
 - Connection pool size
- ▶ The number of different prepared statements

With 50 concurrent users, a connection pool size of 10, and each user using two statements, a select and an insert, the prepared statement cache size should be at least $10 \times 2 = 20$ statements. Check the Information Center for more details at:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/jdbcdatasorprops.html>

12.10.5 Session database tuning

While the session manager implementation in WebSphere provides for a number of parameters that can be tuned to improve performance of applications that utilize HTTP sessions, maximizing performance requires tuning the underlying session persistence table. WebSphere provides a first step by creating an index for the sessions table when creating the table. The index is composed of the session ID, the property ID for multi-row sessions, and the Web application name.

While most database managers provide a great deal of capability in tuning at the table or tablespace level, creating a separate database or instance provides the most flexibility in tuning. Proper tuning of the instance and database can improve performance by 5% or more over that which can be achieved by simply tuning the table or tablespace.

While the specifics vary, depending on the database and operating system, in general, tune and configure the database as appropriate for a database that experiences a great deal of I/O. The database administrator (DBA) should monitor and tune the database buffer pools, database log size, and write frequency. Additionally, maximizing performance requires striping the database or instance across multiple disk drives and disk controllers, and utilizing any hardware or OS buffering available to reduce disk contention.

12.11 Stateful session bean failover

Stateful session bean utilizes the functions of the data replication service and workload management.

Each EJB container provides a method for stateful session beans to fail over to other servers. This enables you to specify whether failover occurs for the stateful session beans at the EJB module level or container level. You can also override the parent object's stateful session bean replication settings from the module level.

12.11.1 Enabling stateful session bean failover

Depending on the requirement, you might not want to enable failover for every single stateful session bean installed in the EJB container. You can set or override the EJB container settings at either the application or EJB module level. You can either enable or disable failover at each of these levels. For example, consider the following situations:

- ▶ You want to enable failover for all applications except for a single application. To do this, you enable failover at the EJB container level and override the setting at the application level to disable failover on the single application.
- ▶ You want to enable failover for a single, installed application. To do this, disable failover at the EJB container level and then override the setting at the application level to enable failover on the single application.
- ▶ You want to enable failover for all applications except for a single module of an application. To do this, enable failover at the EJB container level, then override the setting at the module application level to disable failover on the single module.
- ▶ You want to enable failover for a single, installed EJB module. To do this, disable failover at the EJB container level and then override the setting at the EJB module level to enable failover on the single EJB module.

EJB container stateful session bean failover properties

To access stateful session bean failover properties at the EJB container level from the administrative console:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. Click the application server.
3. In the Container Settings section of the Configuration tab, click **EJB Container Settings** → **EJB container**.
4. In the General Properties section, check **Enable stateful session bean failover using memory-to-memory replication**.

This check box is disabled until you define a replication domain. This selection has a hyperlink to help you configure the replication settings. If no replication domains are configured, the link takes you to a window where you can create one. If at least one domain is configured, the link takes you to a window where you can select the replication settings to be used by the EJB container. See Figure 12-18.

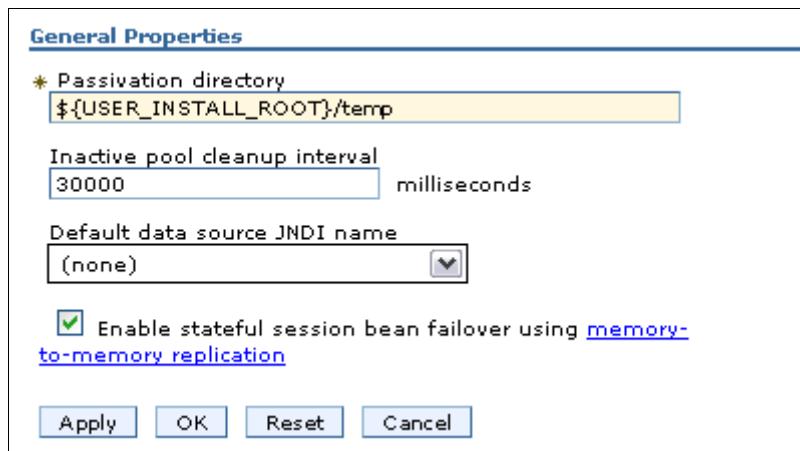


Figure 12-18 Stateful session bean failover settings at the container level

EJB module stateful session bean failover properties

To access stateful session bean failover properties at the EJB module level from the administrative console:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**.
2. Click the application.
3. In the Enterprise Java Bean Properties section of the Configuration tab, click **Stateful session bean failover settings**. See Figure 12-19.

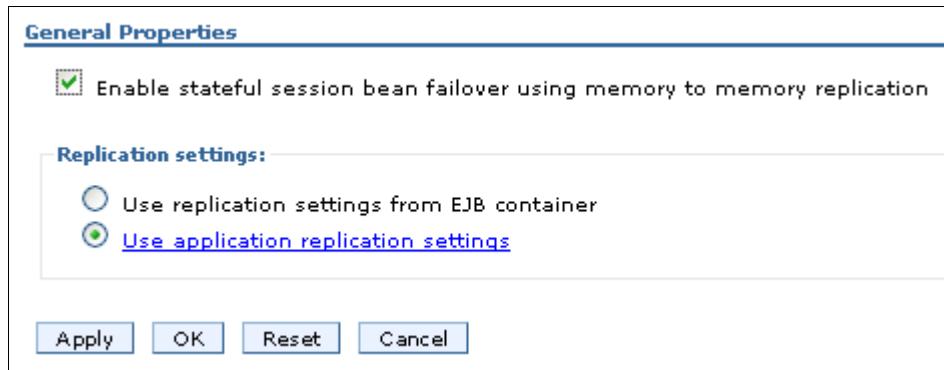


Figure 12-19 Stateful session bean failover settings at the module level

- Enable stateful session bean failover:

Check this check box to enable stateful session bean failover. If you want to disable the failover, clear this check box.

- Use replication settings from EJB container:

If you select this option, any replication settings defined for this application are ignored.

Important: If you use this radio button, then you must configure memory to memory replication at the EJB container level. Otherwise, the settings on this window are ignored by the EJB container during server startup and the EJB container will log a message indicating that stateful session bean failover is not enabled for this application.

- Use application replication settings:

If you select this option, you override the EJB container settings. This button is disabled until you define a replication domain. This selection has a hyperlink to help you configure the replication settings. If no replication domains are configured, the link takes you to a window to create one. If at least one domain is configured, the link takes you to a window where you can select the replication settings to be used by the application.

4. Select your choice of replication settings from:

- Use replication settings from EJB container
- Use application replication settings using memory-to-memory replication

5. Select **OK**.

On WebSphere Application Server V7 for z/OS, the stateful session bean failover among servants can be enabled. Failover only occurs between the servants of a given unmanaged server. If an unmanaged z/OS server has only one servant, then enabling failover has no effect. An unmanaged z/OS server that has failover enabled does not fail over to another unmanaged z/OS server. To enable this feature, consult the instructions in the Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tejb_sfsbfzos.html

12.11.2 Stateful session bean failover consideration

In the following sections we present a few considerations when using the stateful session bean failover feature.

Stateful session bean activation policy with failover enabled

At application assembly, you can specify an activation policy to use for stateful session beans. It is important to consider that the only time the EJB container prepares for failover by replicating the stateful session bean data using DRS, is when the stateful session bean is passivated. If you configure the bean with an activate once policy, the bean is essentially never passivated. If you configure the activate at transaction boundary policy, the bean is passivated whenever the transaction that the bean is enlisted in completes.

For stateful session bean failover to be useful, the **activate at transaction boundary policy** is required. Rather than forcing you to edit the deployment descriptor of every stateful session bean and reinstall the bean, the EJB container simply ignores the configured activation policy for the bean when you enable failover. The container automatically uses the activate at transaction boundary policy.

Container or bean managed units of work

The relevant units of work in this case are transactions and activity sections. WebSphere Application Server supports stateful session bean failover for:

- ▶ Container managed transactions (CMT)
- ▶ Bean managed transactions (BMT)
- ▶ Container managed activity sessions (CMAS)
- ▶ Bean managed activity sessions (BMAS)

In the container-managed cases, preparation for failover only occurs if a request for an enterprise bean method invocation fails to connect to the server. Failover does not take place if the server fails after a request is sent to it and has been acknowledged.

When a failure occurs in the middle of a request or unit of work, WLM cannot safely fail over to another server without some compensation code being executed by the application. When that happens, the application receives a Common Object Request Broker Architecture (CORBA) exception and minor code telling it that transparent failover could not occur because the failure happened during execution of a unit of work.

The application should be written to check for the CORBA exception and minor code, and compensate for the failure. After the compensation code executes, the application can retry the requests and, if a path exists to a backup server, WLM routes the new request to a new primary server for the stateful session bean.

The same is true for bean-managed units of work, transactions, or activity sessions. However, bean managed work introduces a new possibility that needs to be considered.

For bean managed units of work, the failover process is not always able to detect that a BMT or BMAS started by a stateful session bean method has not completed. Thus, it is possible that failover to a new server can occur despite the unit of work failing during the middle of a transaction or session. Because the unit of work is implicitly rolled back, WLM behaves as though it is safe to transparently fail over to another server, when in fact some compensation code might be required. When this happens, the EJB container detects this on the new server and initiates an exception. This exception occurs under the following scenario:

1. A method of a stateful session bean using bean-managed transaction or activity session calls begin on a `UserTransaction` it obtained from the `SessionContext`. The method does some work in the started unit of work, but does not complete the transaction or session before returning to the caller of the method.

2. During post invocation of the method started in step 1, the EJB container suspends the work started by the method. This is the action required by EJB specification for bean managed units of work when the bean is a stateful session bean.
3. The client starts several other methods on the stateful session bean. Each invocation causes the EJB container to resume the suspended transaction or activity session, dispatch the method invocation, and then suspend the work again before returning to the caller.
4. The client calls a method on the stateful session bean that completes the transaction or session started in step 1.

This scenario depicts a *sticky* bean-managed unit of work. The transaction or activity session sticks around for more than a single stateful session bean method. If an application uses a sticky BMT or BMAS, and the server fails after a sticky unit of work completes and before another sticky unit of work starts, failover is successful. However, if the server fails before a sticky transaction or activity session completes, the failover is not successful. Instead, when the failover process routes the stateful session bean request to a new server, the EJB container detects that the failure occurred during an active, sticky transaction or activity session. At that time, the EJB container initiates an exception.

Essentially, this means that failover for both container-managed and bean-managed units of work is not successful if the transaction or activity session is still active. The only real difference is the exception that occurs.

Application design considerations

Consider the following recommendations when designing applications that use the stateful session bean failover process:

- ▶ To avoid the possibility described in the section above, you are encouraged to write your application to configure stateful session beans to use container-managed transactions (CMT) rather than bean-managed transactions (BMT).
- ▶ If you want immediate failover, and your application creates either an HTTP session or a stateful session bean that stores a reference to another stateful session bean, then the administrator must ensure the HTTP session and stateful session bean are configured to use the same replication domain.
- ▶ Do not use a local and a remote reference to the same stateful session bean. The Java EE 5 specification has additional requirements for Http Sessions that require the Http Session state objects to be able to contain local references to EJBs.

Normally a stateful session bean instance with a given primary key can only exist on a single server at any given moment in time. Failover might cause the bean to be moved from one server to another, but it never exists on more than one server at a time. However, there are some unlikely scenarios that can result in the same bean instance, the same primary key, existing on more than one server concurrently. When that happens, each copy of the bean is unaware of the other, and no synchronization occurs between the two instances to ensure they have the same state data. Thus, your application receives unpredictable results.

Note: To avoid this situation, you must remember that with failover enabled, your application should never get both a local (EJBLocalObject) and remote (EJBObject) reference to the same stateful session bean instance.



Understanding class loaders

Understanding how the Java and WebSphere class loaders work is critical to packaging and deploying Java EE5 applications. Failure to set up the class loaders properly most likely results in a cascade of the infamous class loading exceptions (such as `ClassNotFoundException`) when trying to start your application.

In this chapter we explain class loaders and how to customize the behavior of the WebSphere class loaders to suit your particular application's requirements. The chapter concludes with an example designed to illustrate these concepts.

We cover the following topics:

- ▶ “A brief introduction to Java class loaders” on page 678
- ▶ “WebSphere class loader overview” on page 682
- ▶ “Configuring WebSphere for class loaders” on page 686
- ▶ “Class loader viewer” on page 693
- ▶ “Learning class loaders by example” on page 694

13.1 A brief introduction to Java class loaders

Class loaders enable the Java virtual machine (JVM) to load classes. Given the name of a class, the class loader locates the definition of this class. Each Java class must be loaded by a class loader.

When you start a JVM, you use three class loaders: the bootstrap class loader, the extensions class loader, and the application class loader:

- ▶ The *bootstrap* class loader is responsible for loading only the core Java libraries in the `Java_home/jre/lib` directory. This class loader, which is part of the core JVM, is written in native code.
- ▶ The *extensions* class loader is responsible for loading the code in the extensions directories (`Java_home/jre/lib/ext` or any other directory specified by the `java.ext.dirs` system property). This class loader is implemented by the `sun.misc.Launcher$ExtClassLoader` class.
- ▶ The *application class loader* is responsible for loading code that is found on `java.class.path`, which ultimately maps to the system `CLASSPATH` variable. This class loader is implemented by the `sun.misc.Launcher$AppClassLoader` class.

The parent-delegation model is a key concept to understand when dealing with class loaders. It states that a class loader delegates class loading to its parent before trying to load the class itself. The parent class loader can be either another custom class loader or the bootstrap class loader. But what is very important is that a class loader can only delegate requests to its parent class loader, never to its child class loaders (it can go up the hierarchy but never down).

The extensions class loader is the parent for the application class loader. The bootstrap class loader is the parent for the extensions class loader. The class loaders hierarchy is shown in Figure 13-1 on page 679.

If the application class loader needs to load a class, it first delegates to the extensions class loader, which, in turn, delegates to the bootstrap class loader. If the parent class loader cannot load the class, the child class loader tries to find the class in its own repository. In this manner, a class loader is responsible only for loading classes that its ancestors cannot load.

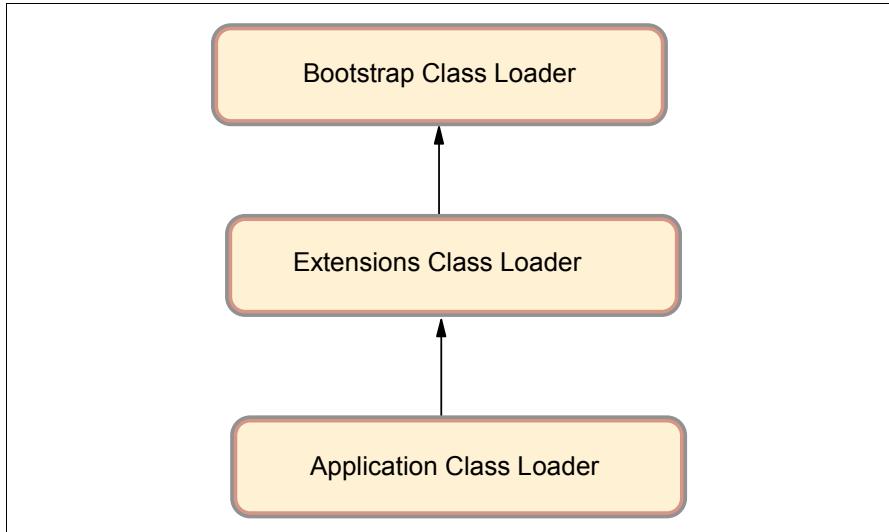


Figure 13-1 Java class loaders hierarchy

This behavior can lead to some interesting problems if a class is loaded from a class loader that is not on a leaf node in the class loader tree. Consider Example 13-1. A class called WhichClassLoader1 loads a class called WhichClassLoader2, in turn invoking a class called WhichClassLoader3.

Example 13-1 WhichClassLoader1 and WhichClassLoader2 source code

```

public class WhichClassLoader1 {

    public static void main(String[] args) throws
javax.naming.NamingException {
    // Get classpath values
    String bootClassPath = System.getProperty("sun.boot.class.path");
    String extClassPath = System.getProperty("java.ext.dirs");
    String appClassPath = System.getProperty("java.class.path");

    // Print them out
    System.out.println("Bootstrap classpath  =" + bootClassPath +
"\n");
    System.out.println("Extensions classpath  =" + extClassPath +
"\n");
    System.out.println("Application classpath=" + appClassPath +
"\n");

    // Load classes
    Object obj = new Object();
}

```

```

WhichClassLoader1 wcl1 = new WhichClassLoader1();
WhichClassLoader2 wcl2 = new WhichClassLoader2();

// Who loaded what?
System.out.println("Object was loaded by "
    + obj.getClass().getClassLoader());
System.out.println("WCL1 was loaded by "
    + wcl1.getClass().getClassLoader());
System.out.println("WCL2 was loaded by "
    + wcl2.getClass().getClassLoader());

    wcl2.getTheClass();
}
}
=====
===
public class WhichClassLoader2 {

// This method is invoked from WhichClassLoader1
public void getTheClass() {
    WhichClassLoader3 wcl3 = new WhichClassLoader3();
    System.out.println("WCL3 was loaded by "
        + wcl3.getClass().getClassLoader());
}
}

```

If all WhichClassLoaderX classes are put on the application class path, the three classes are loaded by the application class loader, and this sample runs just fine. Now suppose that you package the WhichClassLoader2.class file in a JAR file that you store under *Java_home/jre/lib/ext* directory. You can see the output in Example 13-2.

Example 13-2 NoClassDefFoundError exception trace

Bootstrap classpath

```
=C:\WebSphere\AppServer\java\jre\lib\vm.jar;C:\WebSphere\AppServer\java
\jre\lib\core.jar;C:\WebSphere\AppServer\java\jre\lib\charsets.jar;C:\W
ebSphere\AppServer\java\jre\lib\graphics.jar;C:\WebSphere\AppServer\jav
a\jre\lib\security.jar;C:\WebSphere\AppServer\java\jre\lib\ibmpkcs.jar;
C:\WebSphere\AppServer\java\jre\lib\ibmorb.jar;C:\WebSphere\AppServer\j
ava\jre\lib\ibmcfw.jar;C:\WebSphere\AppServer\java\jre\lib\ibmorbapi.ja
r;C:\WebSphere\AppServer\java\jre\lib\ibmjcefw.jar;C:\WebSphere\AppServ
er\java\jre\lib\ibmjgssprovider.jar;C:\WebSphere\AppServer\java\jre\lib
\ibmjsseprovider2.jar;C:\WebSphere\AppServer\java\jre\lib\ibmjaaslm.jar
;C:\WebSphere\AppServer\java\jre\lib\ibmjaasactivelm.jar;C:\WebSphere\A
ppServer\java\jre\lib\ibmcertpathprovider.jar;C:\WebSphere\AppServer\ja
```

```
va\jre\lib\server.jar;C:\WebSphere\AppServer\java\jre\lib\xml.jar  
Extensions classpath =C:\WebSphere\AppServer\java\jre\lib\ext  
Application classpath=.  
  
Exception in thread "main" java.lang.NoClassDefFoundError:  
WhichClassLoader3  
    at java.lang.J9VMInternals.verifyImpl(Native Method)  
    at java.lang.J9VMInternals.verify(J9VMInternals.java:59)  
    at java.lang.J9VMInternals.initialize(J9VMInternals.java:120)  
    at WhichClassLoader1.main(WhichClassLoader1.java:17)
```

As you can see, the program fails with a `NoClassDefFoundError` exception, which might sound strange because `WhichClassLoader3` is on the application class path. The problem is that it is *now* on the wrong class path.

What happened was that the `WhichClassLoader2` class was loaded by the extensions class loader. In fact, the application class loader delegated the load of the `WhichClassLoader2` class to the extensions class loader, which in turn delegated the request to the bootstrap class loader. Because the bootstrap class loader could not find the class, the class loading control was returned to the extensions class loader. The extensions class loader found the class on its class path and loaded it.

Now, when a class has been loaded by a class loader, any new classes that the class needs reuse the same class loader to load them (or goes up the hierarchy according to the parent-delegation model). So when the `WhichClassLoader2` class needed to access the `WhichClassLoader3` class, it is the extensions class loader that first gets the request to load it. The extensions class loader first delegates the request to the Bootstrap class path, which cannot find the class, and then tries to load it itself but does not find it either because `WhichClassLoader3` is not on the extensions class path but on the application classpath. And because the extensions class loader cannot delegate the request to the application class loader (a delegate request can only go up the hierarchy, never down), a `NoClassDefFoundError` exception is thrown.

Note: Remember that developers very often also load property files through the class loader mechanism using the following syntax:

```
Properties p = new Properties();  
p.load(MyClass.class.getClassLoader().getResourceAsStream("myApp.properties"));
```

This means, if the class `MyClass` is loaded by the extensions class loader and the `myApp.properties` file is only seen by the application class loader, the loading of the property file fails.

13.2 WebSphere class loader overview

Note: Keep in mind when reading the following discussion that each JVM has its own set of class loaders. In a WebSphere environment hosting multiple application servers (JVMs), this means the class loaders for the JVMs are completely separate even if they are running on the same physical machine.

Also note that the JVM uses class loaders called the extensions and application class loaders. The WebSphere run time also uses class loaders, called *extensions*, and application class loaders. However, despite their names, these class loaders are not the same as the JVM class loaders.

WebSphere provides several custom delegated class loaders, as shown in Figure 13-2.

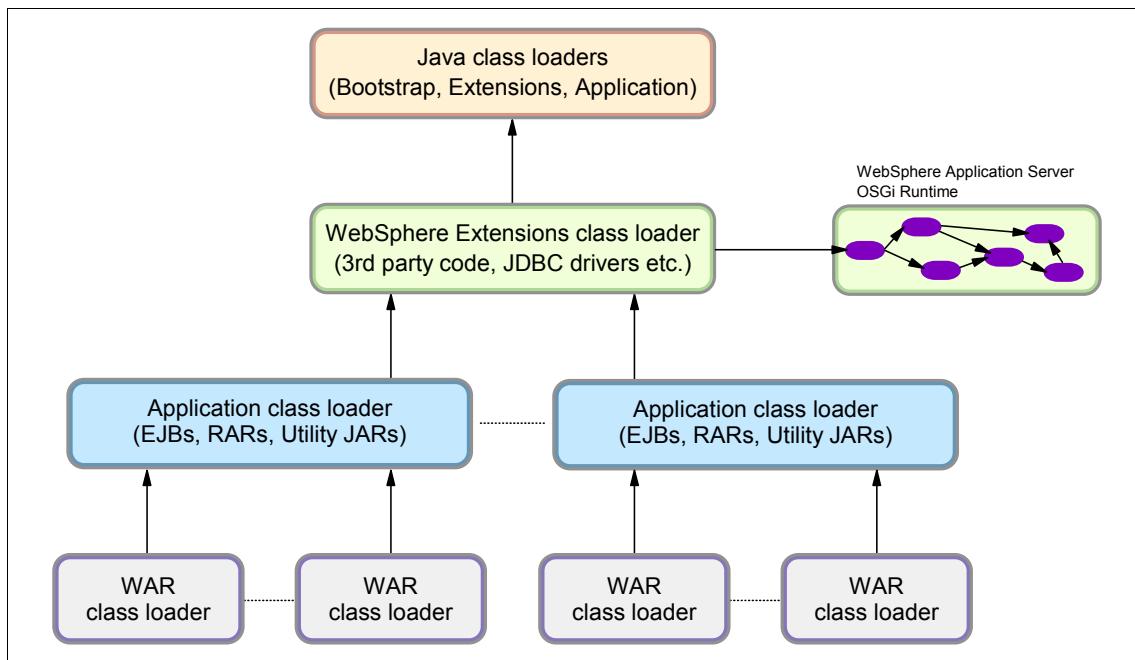


Figure 13-2 WebSphere class loaders hierarchy

The top box represents the Java (bootstrap, extensions, and application) class loaders. WebSphere loads just enough here to get itself bootstrapped and initialize the WebSphere extensions class loader.

13.2.1 WebSphere extensions class loader

The WebSphere extensions class loader is where WebSphere itself is loaded. In versions of WebSphere prior to V6.1, the runtime was loaded by this single class loader. However, beginning with V6.1, WebSphere is packaged as a set of OSGi bundles. Each OSGi bundle is loaded separately by its own class loader. This network of OSGi class loaders is then connected to the extensions class loader and the rest of the class loader hierarchy through an OSGi gateway class loader.

Despite this architectural change in the internals of how WebSphere loads its own classes, there is no behavioral change as far as your applications are concerned. They still have the same visibility, and the same class loading options still exist for your application.

Prior to V6.1, the WebSphere runtime classes files were stored in the classes, lib, lib\ext, and installedChannels directories in the *install_root* directory. Because of the OSGi packaging, these directories no longer exist and the runtime classes are now stored in the *install_root\plugins* directory.

The class path used by the extensions class loader is retrieved from the ws.ext.dirs system property, which is initially derived from the WAS_EXT_DIRS environment variable set in the setupCmdLine script file. The default value of ws.ext.dirs is displayed in Example 13-3.

Example 13-3 Default value of ws.ext.dirs

```
SET  
WAS_EXT_DIRS=%JAVA_HOME%\lib;%WAS_HOME%\classes;%WAS_HOME%\lib;%WAS_HOME%\installedChannels;%WAS_HOME%\lib\ext;%WAS_HOME%\web\help;%ITP_LOC%\plugins\com.ibm.etools.ejbdeploy\runtime
```

Each directory listed in the ws.ext.dirs environment variable is added to the WebSphere extensions class loaders class path and every JAR file and ZIP file in the directory is added to the class path.

While the classes and installedChannels directories no longer exist in the *install_root* directory, the setupCmdLine script still adds them to the extensions class path. This means that if you have added your own JAR files to one of these directories in previous releases, you could create this directory and add your JAR files to it and they would still be loaded by the extensions class loader. However, this is not recommended and you should really try to migrate away from such a setup.

On the other hand, if you have developed Java applications that rely on the WebSphere JAR files that were in the *install_root\lib* directory prior to V6.1, you will need to modify your application to retain compatibility. WebSphere

Application Server provides two thin client libraries designed specifically for such applications: one administrative client library and one Web services client library. These thin client libraries can be found in the `install_root\runtimes` directory:

- ▶ `com.ibm.ws.admin.client_7.0.0.jar`
- ▶ `com.ibm.ws.webservices.thinclient_7.0.0.jar`

These libraries provide everything your application might need for connecting to and working with WebSphere. WebSphere Application Server V7 gives you the ability to restrict access to internal WebSphere classes so that your applications do not make unsupported calls to WebSphere classes not published in the official WebSphere Application Server API. This setting is an application server setting called **Access to internal server classes**.

The default setting is **Allow**, meaning that your applications can make unrestricted calls to non-public internal WebSphere classes. This function is *not* recommended and might be prohibited in future releases. Therefore, as an administrator, it is a good idea to switch this setting to **Restrict** to see if your applications still work. If they depend on non-public WebSphere internal classes, you will receive a `ClassNotFoundException`, and in that case you can switch back to `Allow`. Your developers should then try to migrate their applications so that they do not make unsupported calls to the WebSphere internal classes in order to retain compatibility with future WebSphere Application Server releases.

13.2.2 Application and Web module class loaders

Java EE 5 applications consist of five primary elements: Web modules, EJB modules, application client modules, resource adapters (RAR files), and utility JARs. Utility JARs contain code used by both EJBs and servlets. Utility frameworks such as log4j are good examples of a utility JAR.

EJB modules, utility JARs, resource adapter files, and shared libraries associated with an application are always grouped together into the same class loader. This class loader is called the application class loader. Depending on the class loader policy, this class loader can be shared by multiple applications (EARs), or be unique for each application, which is the default.

By default, Web modules receive their own class loader, a WAR class loader, to load the contents of the WEB-INF/classes and WEB-INF/lib directories. You can modify the default behavior by changing the application's WAR class loader policy. This policy setting can be found in the administrative console by selecting **Applications → WebSphere enterprise applications → *application_name* → Class loading and update detection → WAR class loader policy**. See Figure 13-3.

[Enterprise Applications](#) > [ClassloaderTestV1](#) > Class loader

Use this page to configure the reloading of classes when application files are updated.

Configuration

General Properties

Class reloading options

Override class reloading settings for Web and EJB modules
 true

Polling interval for updated files
 Seconds

Class loader order

Classes loaded with parent class loader first
 Classes loaded with local class loader first (parent last)

WAR class loader policy

Class loader for each WAR file in application
 Single class loader for application



Figure 13-3 WAR class loader policy

The default is set to **Class loader for each WAR file in the application**. This setting is called **Module** in previous releases and in the application deployment descriptor as viewed in Rational Application Developer.

If the WAR class loader policy is set to **Single class loader for application**, the Web module contents are loaded by the application class loader in addition to the EJBs, RARs, utility JARs, and shared libraries. The application class loader is the parent of the WAR class loader. This setting is called **Application** in previous releases and in the application deployment descriptor as viewed in Rational Application Developer.

The application and the WAR class loaders are reloadable class loaders. They monitor changes in the application code to automatically reload modified classes. You can modify this behavior at deployment time.

13.2.3 Handling JNI code

Because a JVM only has a single address space, and native code can only be loaded once per address space, the JVM specification states that native code can only be loaded by one class loader in a JVM.

This might cause a problem if, for example, you have an application (EAR file) with two Web modules that both need to load the same native code through a Java Native Interface (JNI). Only the Web module that first loads the library will succeed.

To solve this problem, you can break out just the few lines of Java code that load the native code into a class on its own and place this file on WebSphere's application class loader (in a utility JAR). However, if you deploy multiple such applications (EAR files) to the same application server, you have to place the class file on the WebSphere extensions class loader instead to ensure that the native code is only loaded once per JVM.

If the native code is placed on a reloadable class loader (such as the application class loader or the WAR class loader), it is important that the native code can properly unload itself should the Java code have to reload. WebSphere has no control over the native code, and if it does not unload and load properly, the application might fail.

If one native library depends on another one, things become even more complicated. For more details, search for Dependent native library in the Information Center.

13.3 Configuring WebSphere for class loaders

In the previous topic, you learned about WebSphere class loaders and how they work together to load classes. There are settings in WebSphere Application Server that allow you to influence WebSphere class loader behavior. This section discusses these options.

13.3.1 Application server class loader policies

For each application server in the system, the class loader policy can be set to Single or Multiple. These settings can be found in the administrative console by selecting **Servers** → **Server Types** → **WebSphere application servers** → **server_name**. See Figure 13-4.

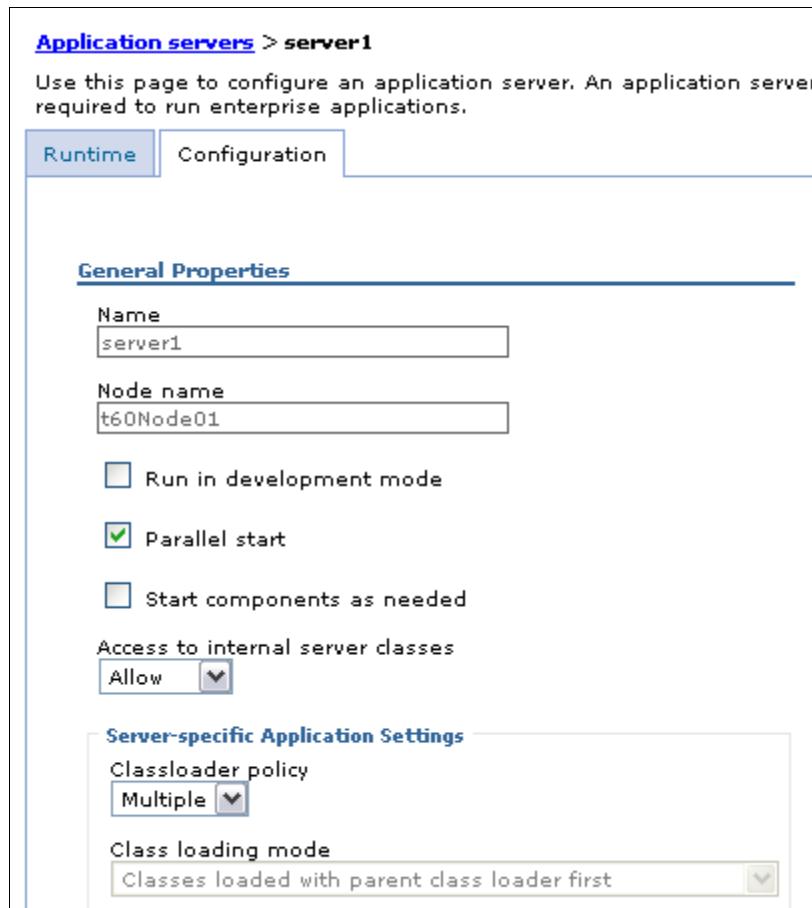


Figure 13-4 Application server classloader settings

When the application server class loader policy is set to Single, a single application class loader is used to load all EJBs, utility JARs, and shared libraries within the application server (JVM). If the WAR class loader policy then has been set to **Single class loader for application**, the Web module contents for this particular application are also loaded by this single class loader.

When the application server class loader policy is set to **Multiple**, the default, each application will receive its own class loader for loading EJBs, utility JARs, and shared libraries. Depending on whether the WAR class loader policy is set to **Class loader for each WAR file in application** or **Single class loader for application**, the Web module might or might not receive its own class loader.

Here is an example to illustrate. Suppose that you have two applications, Application1 and Application2, running in the same application server. Each application has one EJB module, one utility JAR, and two Web modules. If the application server has its class loader policy set to **Multiple** and the class loader policy for all the Web modules are set to **Class loader for each WAR file in application**, the result is as shown in Figure 13-5.

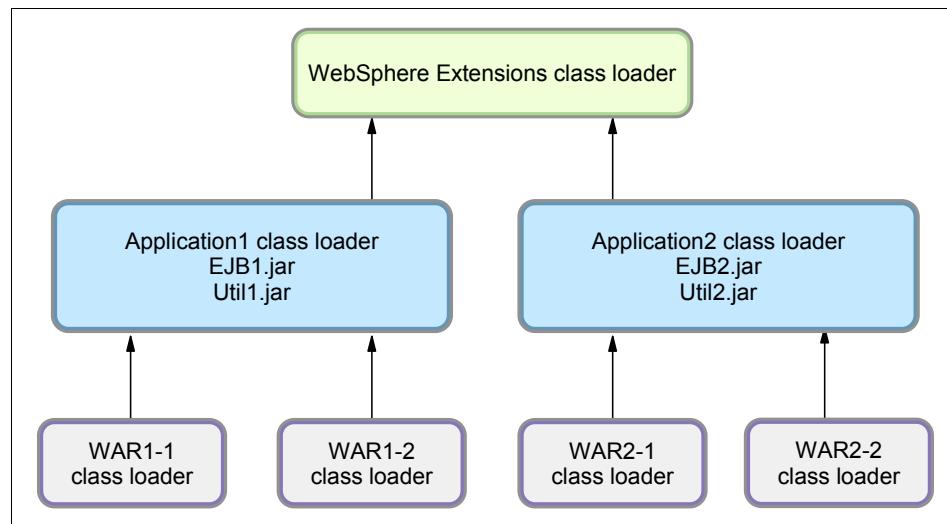


Figure 13-5 Class loader policies: Example 1

Each application is completely separated from the other and each Web module is also completely separated from the other one in the same application. WebSphere's default class loader policies results in total isolation between the applications and the modules.

If we now change the class loader policy for the WAR2-2 module to **Single class loader for application**, the result is shown in Figure 13-6.

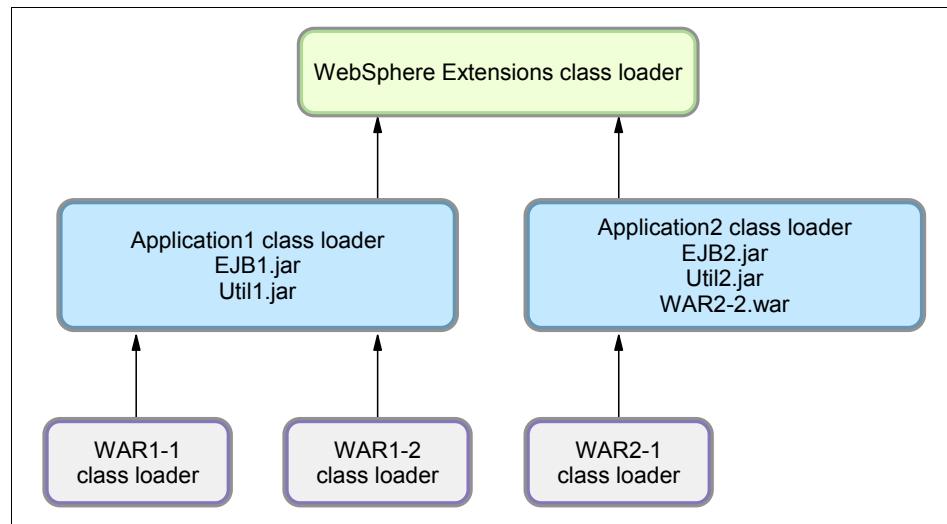


Figure 13-6 Class loader policies: Example 2

Web module WAR2-2 is loaded by Application2's class loader and classes, and for example, classes in Util2.jar are able to see classes in WAR2-2's /WEB-INF/classes and /WEB-INF/lib directories.

As a last example, if we change the class loader policy for the application server to **Single** and also change the class loader policy for WAR2-1 to **Single class loader for application**, the result is as shown in Figure 13-7.

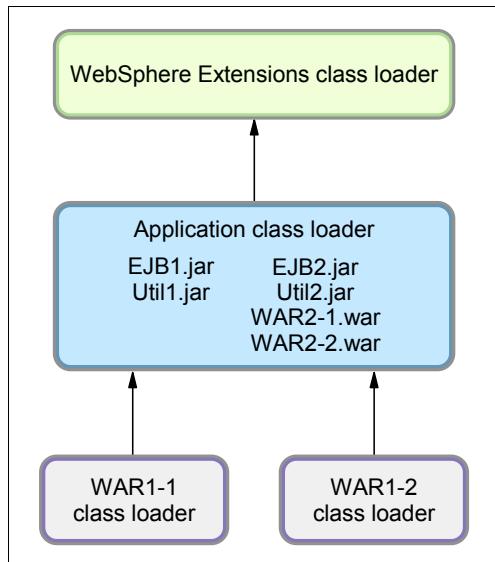


Figure 13-7 Class loader policies: Example 3

There is now only a single application class loader loading classes for both Application1 and Application2. Classes in Util1.jar can see classes in EJB2.jar, Util2.jar, WAR2-1.war and WAR2-2.war. The classes loaded by the application class loader still cannot, however, see the classes in the WAR1-1 and WAR1-2 modules, because a class loader can only find classes by going up the hierarchy, never down.

13.3.2 Class loading/delegation mode

WebSphere's application class loader and WAR class loader both have a setting called the class loader order (see Figure 13-4 on page 687). This setting determines whether the class loader order should follow the normal Java class loader delegation mechanism, as described in 13.1, “A brief introduction to Java class loaders” on page 678, or override it.

There are two possible options for the class loading mode:

- ▶ Classes loaded with parent class loader first
- ▶ Classes loaded with local class loader first (parent last)

In previous WebSphere releases, these settings were called PARENT_FIRST and PARENT_LAST, respectively.

The default value for class loading mode is **Classes loaded with parent class loader first**. This mode causes the class loader to first delegate the loading of classes to its parent class loader before attempting to load the class from its local class path. This is the default policy for standard Java class loaders.

If the class loading policy is set to **Classes loaded with local class loader first (parent last)**, the class loader attempts to load classes from its local class path before delegating the class loading to its parent. This policy allows an application class loader to override and provide its own version of a class that exists in the parent class loader.

Note: The administrative console is a bit confusing at this point. On the settings page for a Web module, the two options for class loader order are Classes loaded with parent class loader first and Classes loaded with local class loader first (parent last). However, in this context, the “local class loader” really refers to the WAR class loader, so the option **Classes loaded with local class loader first** should really be called Classes loaded with WAR class loader first.

Assume that you have an application, similar to Application1 in the previous examples, and it uses the popular log4j package to perform logging from both the EJB module and the two Web modules. Also assume that each module has its own, unique, log4j.properties file packaged into the module. You could configure log4j as a utility JAR so you would only have a single copy of it in your EAR file.

However, if you do that, you might be surprised to see that all modules, including the Web modules, load the log4j.properties file from the EJB module. The reason is that when a Web module initializes the log4j package, the log4j classes are loaded by the application class loader. Log4j is configured as a utility JAR. Log4j then looks for a log4j.properties file on its class path and finds it in the EJB module.

Even if you do not use log4j for logging from the EJB module and the EJB module does not, therefore, contain a log4j.properties file, log4j does not find the log4j.properties file in any of the Web modules anyway. The reason is that a class loader can only find classes by going up the hierarchy, never down.

To solve this problem, you can use one of the following approaches:

- ▶ Create a separate file, for example, Resource.jar, configure it as a utility JAR, move all log4j.properties files from the modules into this file, and make their names unique (like war1-1_log4j.properties, war1-2_log4j.properties, and ejb1_log4j.properties). When initializing log4j from each module, tell it to load the proper configuration file for the module instead of the default (log4j.properties).

- ▶ Keep the log4j.properties for the Web modules in their original place (/WEB-INF/classes), add log4j.jar to both Web modules (/WEB-INF/lib) and set the class loading mode for the Web modules to **Classes loaded with local class loader first (parent last)**. When initializing log4j from a Web module, it loads the log4j.jar from the module itself and log4j would find the log4j.properties on its local classpath, the Web module itself. When the EJB module initializes log4j, it loads from the application class loader and it finds the log4j.properties file on the same class path, the one in the EJB1.jar file.
- ▶ If possible, merge all log4j.properties files into one and place it on the Application class loader, in a Resource.jar file, for example).

Singles: The Singleton pattern is used to ensure that a class is instantiated only once. However, *once* only means *once for each class loader*. If you have a Singleton instantiated in two separate Web modules, two separate instances of this class are created—one for each WAR class loader. Thus, in a multi-class loader environment, take care when implementing Singletons.

13.3.3 Shared libraries

Shared libraries are files used by multiple applications. Examples of shared libraries are commonly used frameworks like Apache Struts or log4j. You use shared libraries typically to point to a set of JARs and associate those JARs to an application, a Web module, or the class loader of an application server. Shared libraries are especially useful when you have different versions of the same framework you want to associate to different applications.

Shared libraries are defined using the administration tools. They consist of a symbolic name, a Java class path, and a native path for loading JNI libraries. They can be defined at the cell, node, server, or cluster level. However, simply defining a library does not cause the library to be loaded. You must associate the library to an application, a Web module, or the class loader of an application server for the classes represented by the shared library to be loaded. Associating the library to the class loader of an application server makes the library available to all applications on the server.

Note: If you associate a shared library to an application, do not associate the same library to the class loader of an application server.

You can associate the shared library to an application in one of two ways:

- ▶ You can use the administrative console. The library is added using the **Shared libraries references** link under the References section for the enterprise application.
- ▶ You can use the manifest file of the application and the shared library. The shared library contains a manifest file that identifies it as an extension. The dependency to the library is declared in the application's manifest file by listing the library extension name in an extension list.

For more information about this method, search for installed optional packages in the Information Center.

Shared files are associated with the class loader of an application server using the administrative tools. The settings are found in the Server Infrastructure section. Expand the Java and Process Management. Select **Class loader** and then click the **New** button to define a new class loader. After you have defined a new class loader, you can modify it and, using the Shared library references link, you can associate it to the shared libraries you need.

See “Step 4: Sharing utility JARs using shared libraries” on page 702 for more details.

13.4 Class loader viewer

If the Class Loader Viewer Service is not enabled, the Class Loader Viewer only displays the hierarchy of class loaders and their classpaths, but not the classes actually loaded by each of the class loaders. This also means that the search capability of the Class Loader Viewer is lost.

To enable the Class Loader Viewer Service, select **Servers** → **Server Types** → **WebSphere application server** → *server_name* and then click the **Class Loader Viewer Service** under the **Additional Properties** link. Then select **Enable service at server startup**. You will need to restart the application server for the setting to take effect.

In the next section, we give you an example of how to work with the different class loader settings, then we also use the Class Loader Viewer to illustrate the different results.

13.5 Learning class loaders by example

We have now described all the different options for influencing class loader behavior. In this section, we take an example and use all the different options we have discussed to this point so that you can better evaluate the best solution for your applications.

We have created a very simple application, with one servlet and one EJB. Both call a class, VersionChecker, shown in Example 13-4. This class can print which class loader was used to load the class. The VersionChecker class also has an internal value that can be printed to check which version of the class we are using. This will be used later to demonstrate the use of multiple versions of the same utility JAR.

Example 13-4 VersionChecker class source code

```
package com.itso.classloaders;

public class VersionChecker {
    static final public String classVersion = "v1.0";

    public String getInfo() {
        return ("VersionChecker is " + classVersion +
            ". Loaded by " + this.getClass().getClassLoader());
    }

}
```

After being installed, the application can be invoked through <http://localhost:9080/ClassLoaderExampleWeb/ExampleServlet>. This invokes the ExampleServlet which calls VersionChecker and then displays the classloader.

The VersionCheckerV1.jar file contains the VersionChecker class file that returns Version number 1.0. For all the following tests, we have, unless otherwise noted, left the class loader policies and loading modes to their defaults. In other words, we have one class loader for the application and one for the WAR file. Both have their delegation modes set to **Classes loaded with parent class loader first**.

13.5.1 Step 1: Simple Web module packaging

Start with the following assumption: our utility class is only used by a servlet. We have placed the VersionCheckerV1.jar file under the WEB-INF/lib directory of the Web module.

Tip: You place JAR files that are used by a single Web module, or a JAR file that *only* this Web module should see, under WEB-INF/lib.

Example 13-5 shows the results of running the application in such a configuration.

Example 13-5 Class loader: Example 1

```
VersionChecker called from Servlet
VersionChecker is v1.0.
Loaded by
com.ibm.ws.classloader.CompoundClassLoader@18721872[war:ClassloaderExample/ClassloaderExampleWeb.war]
```

Local ClassPath:

```
C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExample.ear\ClassloaderExampleWeb.war\WEB-INF\classes;
C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExample.ear\ClassloaderExampleWeb.war\WEB-INF\lib\VersionCheckerV1.jar;
C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExample.ear\ClassloaderExampleWeb.war
```

Parent:

```
com.ibm.ws.classloader.CompoundClassLoader@7f277f27[app:ClassloaderExample]
```

Delegation Mode: PARENT_FIRST

There are a few things that we can learn from this trace:

1. The type of the WAR class loader is:

```
com.ibm.ws.classloader.CompoundClassLoader
```

2. It searches classes in the following order:

```
ClassloaderExampleWeb.war\WEB-INF\classes
ClassloaderExampleWeb.war\WEB-INF\lib\VersionCheckerV1.jar
ClassloaderExampleWeb.war
```

The WEB-INF/classes folder holds unpacked resources (such as servlet classes, plain Java classes, and property files), while the WEB-INF/lib holds resources packaged as JAR files. You can choose to package your Java code in JAR files and place them in the lib directory or you can put them unpacked in the classes directory. They will both be on the same classpath. Because our sample application was developed and exported from the Rational Application Developer, our servlet goes into the classes folder, because the Java classes are not packaged in a JAR file when exporting an application.

The root of the WAR file is the next place where you can put code or properties, but you really should not do that because that folder is the document root for the Web server (if the File Serving Servlet capabilities are enabled, which they are by default) so anything that is in that folder is accessible from a browser. According to the Java EE 5 specification, though, the WEB-INF folder is protected, which is why the classes and lib folders are under WEB-INF.

The class loader class path is dynamically built at application startup.

We can now also use the Class Loader Viewer to display the class loader. In the administrative console, select **Troubleshooting** → **Class Loader Viewer**. Then expand **server1** → **Applications** → **ClassloaderExample** → **Web modules** and click the **ClassloaderExampleWeb.war**, as shown in Figure 13-8.

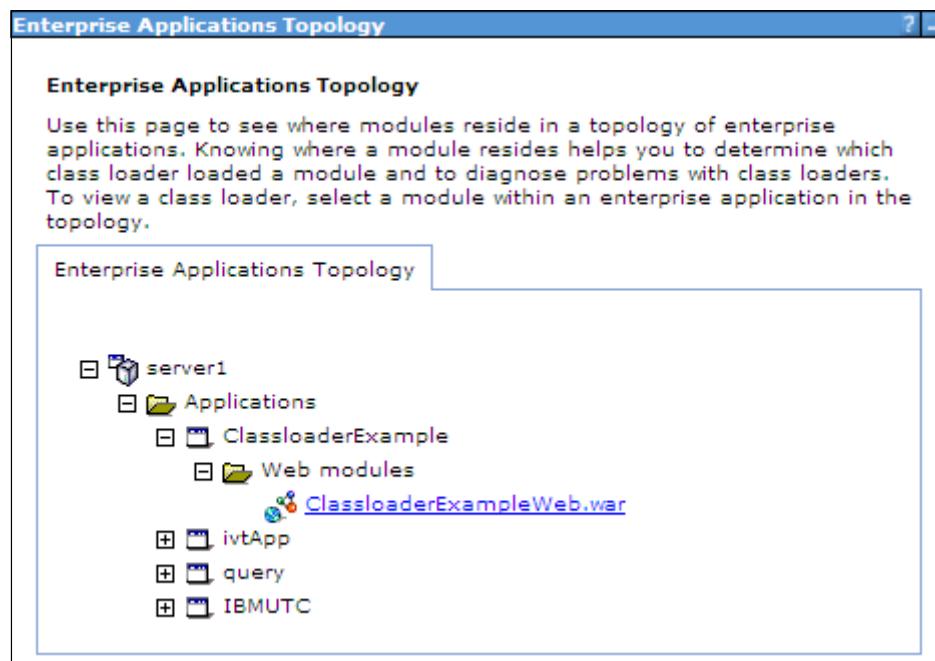


Figure 13-8 Class Loader Viewer showing applications tree

When the Web module is expanded, the Class Loader Viewer shows the hierarchy of class loaders all the way from the JDK Extensions and JDK application class loaders at the top to the WAR class loader at the bottom, called the compound class loader. See Figure 13-9.

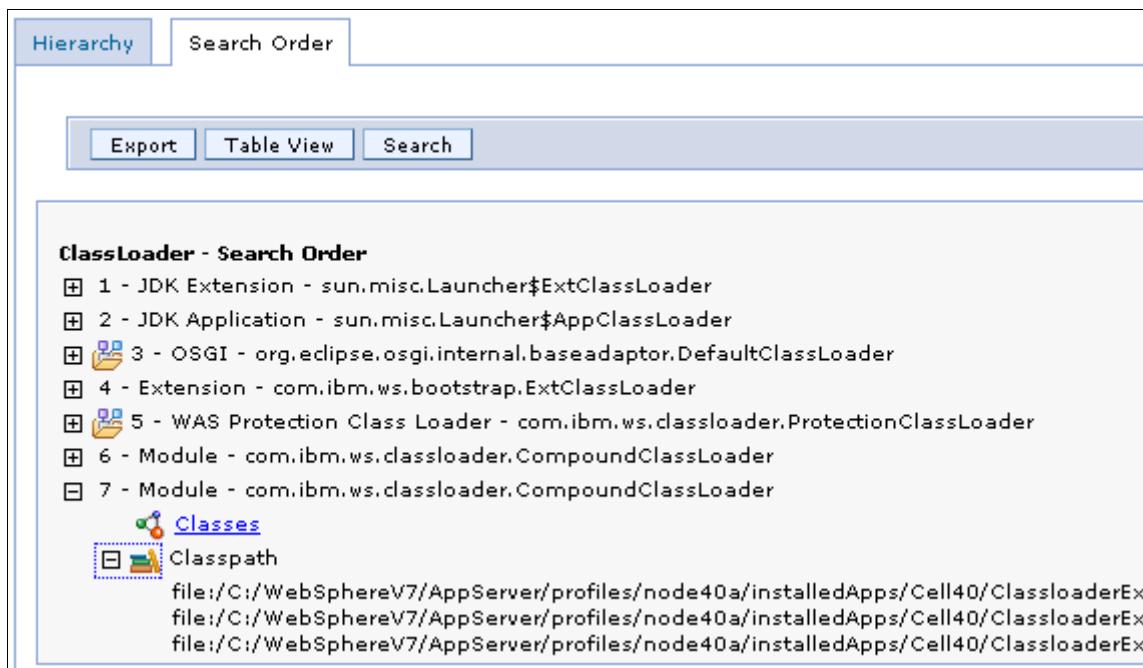


Figure 13-9 Class Loader Viewer showing class loader hierarchy

If you expand the classpath for com.ibm.ws.classloader.CompoundClassLoader, you see the same information as the VersionChecker class prints (see Example 13-5 on page 688).

Note: For the Class Loader Viewer to display the classes loaded, the Class Loader Viewer Service must be enabled as described in “Class loader viewer” on page 693.

The Class Loader Viewer also has a table view that displays all the class loaders and the classes loaded by each of them on a single page. The table view also displays the Delegation mode. True means that classes are loaded with parent class loader first, while false means that classes are loaded with local class loader first (parent last), or the WAR class loader in the case of a Web module. See Figure 13-10.

Module - com.ibm.ws.classloader.CompoundClassLoader	
Delegation	true
Classpath	<pre>file:/C:/WebSphereV7/AppServer/profiles/node40a/installedApps/Cell40 /ClassloaderExample.ear/ClassloaderExampleWeb.war/WEB-INF/classes</pre> <pre>file:/C:/WebSphereV7/AppServer/profiles/node40a/installedApps/Cell40 /ClassloaderExample.ear/ClassloaderExampleWeb.war/WEB-INF /lib/VersionCheckerV1.jar</pre> <pre>file:/C:/WebSphereV7/AppServer/profiles/node40a/installedApps/Cell40 /ClassloaderExample.ear/ClassloaderExampleWeb.war</pre>

Figure 13-10 Class Loader Viewer table view

As you can see, the WAR class loader has loaded our example servlet and the VersionChecker class, just as expected.

The Class Loader Viewer also has a search feature where you can search for classes, JAR files, folders, and so on. This can be particularly useful if you do not know which of the class loaders loaded a class you are interested in. The search feature is case sensitive but allows wild cards, so a search for *VersionChecker* finds our VersionChecker class.

13.5.2 Step 2: Adding an EJB module and utility jar

Next, we decided to add an EJB to our application that also depends on our VersionChecker JAR file. For this task, we added a VersionCheckerV2.jar file to the root of our EAR. The VersionChecker class in this JAR file returns Version 2.0. To make it available as a utility JAR on the extensions class loader, we added a reference to it in the EJB module's manifest file, as shown in Example 13-6.

Example 13-6 Updated MANIFEST.MF for EJB module

Manifest-Version: 1.0
 Class-Path: VersionCheckerV2.jar

The result is that we now have a Web module with a servlet in the WEB-INF/classes folder and the VersionCheckerV1.jar file in the WEB-INF/lib folder. We also have an EJB module that references the VersionCheckerV2.jar utility JAR in the root of the EAR. Which version of the VersionChecker class file would you expect the Web module to load? Version 1.0 from the WEB-INF/lib or version 2.0 from the utility JAR?

The test results are shown in Example 13-7.

Example 13-7 Class loader: Example 2

VersionChecker called from Servlet

VersionChecker is v2.0.

Loaded by

com.ibm.ws.classloader.CompoundClassLoader@404a404a[app:ClassloaderExampleV2]

Local ClassPath:

C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExampleV2.ear\ClassloaderExampleEJB.jar;C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExampleV2.ear\VersionCheckerV2.jar

Parent: com.ibm.ws.classloader.ProtectionClassLoader@a540a54

Delegation Mode: PARENT_FIRST

VersionChecker called from EJB

VersionChecker is v2.0.

Loaded by

com.ibm.ws.classloader.CompoundClassLoader@404a404a[app:ClassloaderExampleV2]

Local ClassPath:

C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExampleV2.ear\ClassloaderExampleEJB.jar;C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExampleV2.ear\VersionCheckerV2.jar

Parent: com.ibm.ws.classloader.ProtectionClassLoader@a540a54

Delegation Mode: PARENT_FIRST

As you can see, the VersionChecker is Version 2.0 when called both from the EJB module and the Web module. The reason is, of course, that the WAR class loader delegates the request to its parent class loader instead of loading it itself, so the Utility JAR is loaded by the same class loader regardless of whether it was called from the servlet or the EJB.

13.5.3 Step 3: Changing the WAR class loader delegation mode

What if we now wanted the Web module to use the VersionCheckerV1.jar file from the WEB-INF/lib folder? For that task, we would have to change the class loader delegation from parent first to parent last.

Set the delegation mode to PARENT_LAST, using the following steps:

1. Select the **WebSphere Enterprise Applications** entry in the navigation area.
2. Select the **ClassloaderExample** application.
3. Select **Manage modules** under the Modules section.
4. Select the **ClassloaderExampleWeb** module.
5. Change the Class loader order to **Classes loaded with local class loader first (parent_last)**. Remember, this entry should really be called Classes loaded with WAR class loader first, as noted in “Class loading/delegation mode” on page 690.
6. Click **OK**.
7. Save the configuration.
8. Restart the application.

The VersionCheckerV1 in WEB-INF/lib returns a class version of 1.0. You can see in Example 13-8 that this is the version now used by the WAR file.

Example 13-8 Class loader: Example 3

VersionChecker called from Servlet

VersionChecker is v1.0.

Loaded by

com.ibm.ws.classloader.CompoundClassLoader@1c421c42[war:ClassloaderExampleV2/ClassloaderExampleWeb.war]

Local ClassPath:

C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Ce1140\ClassloaderExampleV2.ear\ClassloaderExampleWeb.war\WEB-INF\classes;C:\WebSphere V7\AppServer\profiles\node40a\installedApps\Ce1140\ClassloaderExampleV2 .ear\ClassloaderExampleWeb.war\WEB-INF\lib\VersionCheckerV1.jar;C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Ce1140\ClassloaderExam pleV2.ear\ClassloaderExampleWeb.war

Parent:

com.ibm.ws.classloader.CompoundClassLoader@1a5e1a5e[app:ClassloaderExam pleV2]

Delegation Mode: PARENT_LAST

```
VersionChecker called from EJB
VersionChecker is v2.0.
Loaded by
com.ibm.ws.classloader.CompoundClassLoader@1a5e1a5e[app:ClassloaderExam
pleV2]

Local ClassPath:
C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell40\Classloa
derExampleV2.ear\ClassloaderExampleEJB.jar;C:\WebSphereV7\AppServer\pro
files\node40a\installedApps\Cell40\ClassloaderExampleV2.ear\VersionChec
kerV2.jar
Parent: com.ibm.ws.classloader.ProtectionClassLoader@a540a54

Delegation Mode: PARENT_FIRST
```

Tip: Use this technique to specify that a Web module should use a specific version of a library, such as Struts, or to override classes coming with the WebSphere runtime. Put the common version at the top of the hierarchy, and the specialized version in WEB-INF/lib.

The Java EE 5 specification does not provide a standard option to specify the delegation mode in the EAR file. However, by using a WebSphere Extended EAR file, you can specify this setting so you do not have to change it every time you redeploy your application.

If you use the search feature of the Class Loader Viewer to search for “*VersionChecker”, you would see the two entries in Example 13-9.

Example 13-9 Class Loader Viewer search feature

WAS Module Compound Class Loader (WAR class loader):
file: / C: / WebSphereV7 / AppServer / profiles / node40a /
installedApps / Cell40 / ClassloaderExampleV2.ear /
ClassloaderExampleWeb.war / WEB-INF / lib / VersionCheckerV1.jar

WAS Module Jar Class Loader (Application class loader):
file: / C: / WebSphereV7 / AppServer / profiles / node40a /
installedApps / Cell40 / ClassloaderExampleV2.ear /
VersionCheckerV2.jar

13.5.4 Step 4: Sharing utility JARs using shared libraries

In this situation, the VersionCheckerV2.jar file is used by a single application. What if you wanted to share it among multiple applications? Of course, you could package it within each EAR file. But changes to this utility JAR file require redeploying all applications again. To avoid this, you can externalize global utility JARs using a shared library.

Shared libraries can be defined at the cell, node, application server, and cluster levels. After you have defined a shared library, you must associate it to the class loader of an application server, an application, or an individual Web module. Depending on the target the shared library is assigned to, WebSphere will use the appropriate class loader to load the shared library.

You can define as many shared libraries as you want. You can also associate multiple shared libraries with an application, Web module, or application server.

Using shared libraries at the application level

To define a shared library named VersionCheckerV2_SharedLib and associate it to our ClassloaderTest application, do the following steps:

1. In the administrative console, select **Environment** → **Shared Libraries**.
2. Select the scope at which you want this shared library to be defined, such as Cell, and click **New**.
3. Specify the properties shown in Figure 13-11.

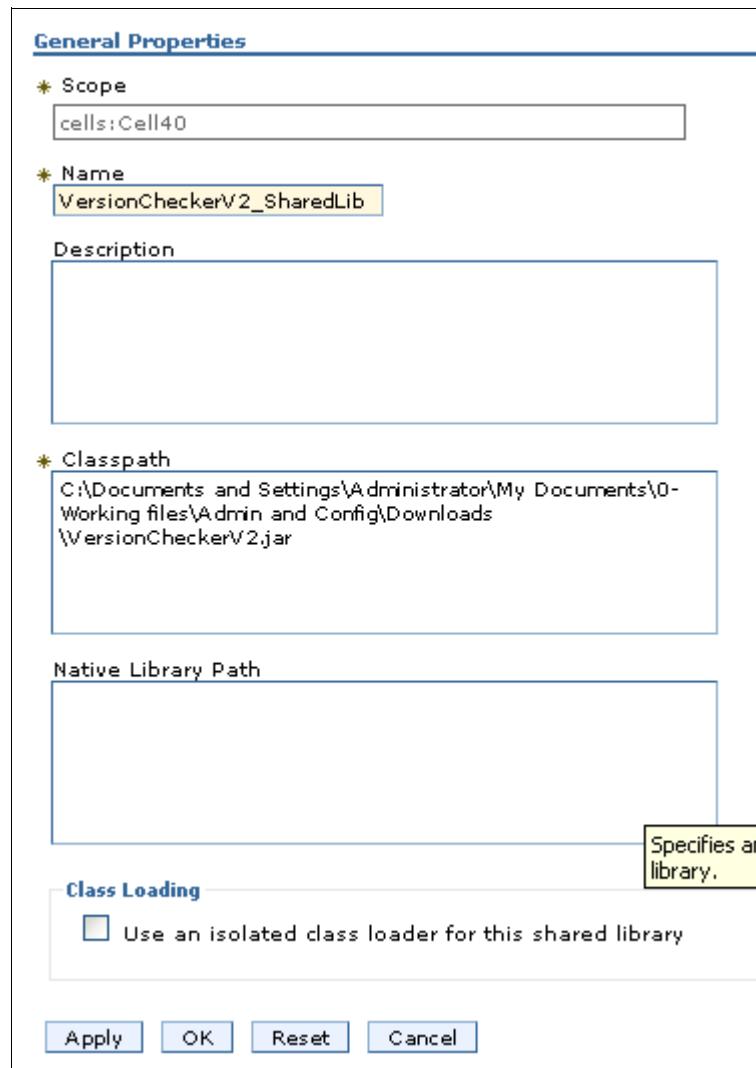


Figure 13-11 Shared library configuration

- Name: Enter VersionCheckerV2_SharedLib.
- Class path: Enter the list of entries on the class path. Press Enter between each entry. Note that if you need to provide an absolute path, we highly recommend that you use WebSphere variables, such as %FRAMEWORK_JARS%/VersionCheckerV2.jar. Make sure that you declare this variable at the same scope as the shared library for cell, node, server, or cluster.

- Native library path: Enter a list of DLLs and .so files for use by the JNI code.
 - (**NEW in V7**) If you want to have only one instance of a version of a class shared among applications, select **Use an isolated class loader for this shared library**.
4. Click **OK**.
 5. Select **Applications → Application Types → WebSphere enterprise applications**.
 6. Select the **ClassloadersExample** application.
 7. In References, select **Shared library references**.
 8. Select **ClassloaderExample** in the Application row.
 9. Click **Reference shared libraries**.
 10. Select the **VersionCheckerV2_SharedLib** and click the **>>** button to move it to the Selected column, as shown in Figure 13-12.

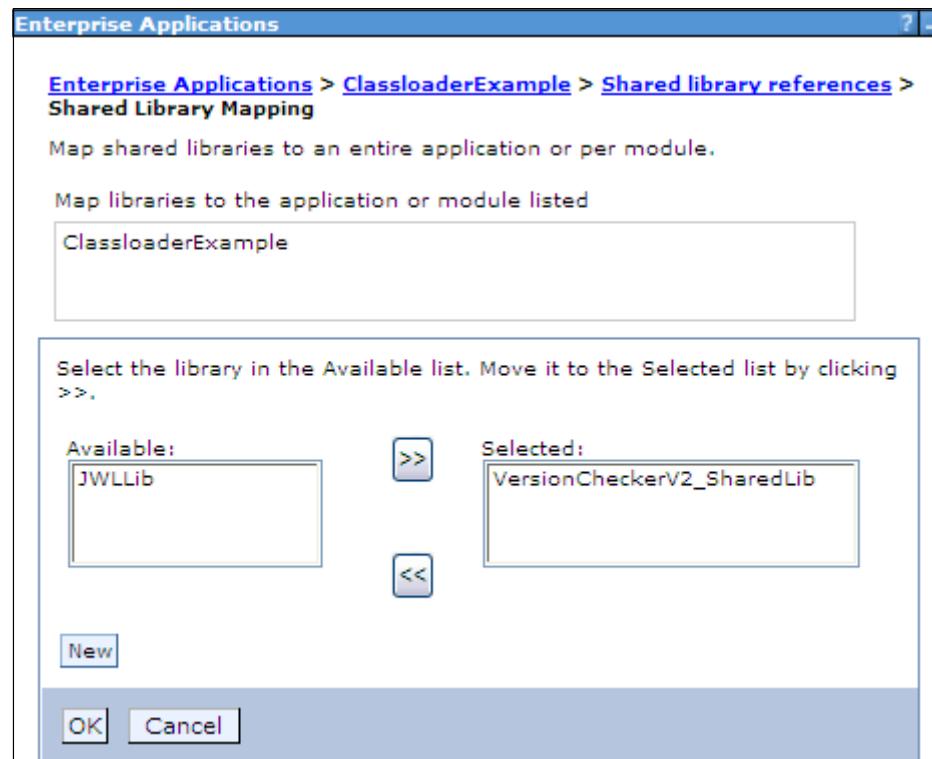


Figure 13-12 Assigning a shared library

11.Click **OK**.

12.The shared library configuration window for the ClassloaderExample application should now look like Figure 13-13.

The screenshot shows a configuration interface for 'Enterprise Applications'. The title bar says 'Enterprise Applications'. Below it, the path 'Enterprise Applications > ClassloaderExample > Shared library references' is displayed. A sub-section titled 'Shared Library Mapping for Modules' is shown. A note states: 'Specify shared libraries that the application or individual modules reference. These libraries must be defined in the configuration at the appropriate scope.' Below this is a table titled 'Reference shared libraries'. It has four columns: 'Select', 'Application', 'URI', and 'Shared Libraries'. There are two rows. The first row corresponds to the 'ClassloaderExample' application, with its URI being 'META-INF/application.xml' and its shared library being 'VersionCheckerV2_SharedLib'. The second row corresponds to the 'ClassloaderExampleWeb' module, with its URI being 'ClassloaderExampleWeb.war,WEB-INF/web.xml' and no shared library listed.

Select	Application	URI	Shared Libraries
<input type="checkbox"/>	ClassloaderExample	META-INF/application.xml	VersionCheckerV2_SharedLib
Select	Module	URI	Shared Libraries
<input type="checkbox"/>	ClassloaderExampleWeb	ClassloaderExampleWeb.war,WEB-INF/web.xml	

Figure 13-13 Shared library assigned to ClassloaderExample application

13.Click **OK** and save the configuration.

If we now remove the VersionCheckerV2.jar file from the root of the EAR file, and remove the reference to it from the EJB module's manifest file, and restart the application server, we see the results in Example 13-10. Remember the class loader order for the Web module is still **Classes loaded with local class loader first (parent last)**.

Example 13-10 Class loader: Example 5

VersionChecker called from Servlet

VersionChecker is v1.0.

Loaded by

`com.ibm.ws.classloader.CompoundClassLoader@405d405d [war:ClassloaderExampleV3/ClassloaderExampleWeb.war]`

Local ClassPath:

`C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExampleV3.ear\ClassloaderExampleWeb.war\WEB-INF\classes;C:\WebSphere V7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExampleV3 .ear\ClassloaderExampleWeb.war\WEB-INF\lib\VersionCheckerV1.jar;C:\WebS phereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExam pleV3.ear\ClassloaderExampleWeb.war Parent:`

`com.ibm.ws.classloader.CompoundClassLoader@3e793e79 [app:ClassloaderExam pleV3]`

Delegation Mode: PARENT_LAST

VersionChecker called from EJB

VersionChecker is v2.0.

Loaded by

com.ibm.ws.classloader.CompoundClassLoader@3e793e79 [app:ClassloaderExampleV3]

Local ClassPath:

C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassLoaderExampleV3.ear\ClassLoaderExampleEJB.jar;C:\Documents and Settings\Administrator\My Documents\0-Working files\Admin and Config\Downloads\VersionCheckerV2.jar Parent:
com.ibm.ws.classloader.ProtectionClassLoader@7d677d67

Delegation Mode: PARENT_FIRST

As expected, because of the delegation mode for the Web module, the VersionCheckerV1.jar file was loaded when the servlet needed the VersionChecker class. When the EJB needed the VersionChecker class, it was loaded from the shared library, which points to C:\henrik\VersionCheckerV2.jar.

If we would like the Web module to also use the shared library, we would just restore the class loader order to the default, **Classes loaded with parent class loader first**, for the Web module.

Using shared libraries at the application server level

A shared library can also be associated with an application server. All applications deployed on this server see the code listed on that shared library. To associate a shared library to an application server, you must first create an additional class loader for the application server, as follows:

1. Select an application server.
2. In the Server Infrastructure section, expand the **Java and Process Management**. Select **Class loader**.
3. Choose **New**, and select a class loader order for this class loader, Classes loaded with parent class loader first or Classes loaded with local class loader first (parent last). Click **OK**.
4. Click the class loader that is created.
5. Click **Shared library references**.

6. Click **Add**, and select the library you want to associate to this application server. Repeat this operation to associate multiple libraries to this class loader. For our example, we selected the **VersionCheckerV2_SharedLib** entry.
7. Click **OK**.
8. Save the configuration.
9. Restart the application server for the changes to take effect.

Because we have now attached the VersionCheckerV2 shared library to the class loader of the application server, we obtain the results in Example 13-11.

Example 13-11 Class loader: Example 6

VersionChecker called from Servlet
VersionChecker is v1.0.

Loaded by
com.ibm.ws.classloader.CompoundClassLoader@26a426a4 [war:ClassloaderExampleV3/ClassloaderExampleWeb.war]

Local ClassPath:
C:\WebSphereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExampleV3.ear\ClassloaderExampleWeb.war\WEB-INF\classes;C:\WebSphere V7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExampleV3 .ear\ClassloaderExampleWeb.war\WEB-INF\lib\VersionCheckerV1.jar;C:\WebS phereV7\AppServer\profiles\node40a\installedApps\Cell140\ClassloaderExam pleV3.ear\ClassloaderExampleWeb.war Parent:
com.ibm.ws.classloader.CompoundClassLoader@1f381f38 [app:ClassloaderExam pleV3]

Delegation Mode: **PARENT_LAST**

VersionChecker called from EJB
VersionChecker is v2.0.

Loaded by com.ibm.ws.classloader.ExtJarClassLoader@48964896 [server:0]

Local ClassPath: **C:\Documents and Settings\Administrator\My Documents\0-Working files\Admin and Config\Downloads\VersionCheckerV2.jar**
Parent: com.ibm.ws.classloader.ProtectionClassLoader@7d677d67
Delegation Mode: **PARENT_FIRST**

The new class loader we defined is called the ExtJarClassLoader and it has loaded the VersionCheckerV2.jar file when requested by the EJB module.

The WAR class loader still continues to load its own version due to the delegation mode.

Tip: For further details, tuning, tuning and troubleshooting, see the IBM JDK6 Diagnostics Guide, found in DeveloperWorks at:

<http://www.ibm.com/developerworks/java/jdk/diagnosis/60.html>



Packaging applications for deployment

In this chapter, we discuss packaging of Java Platform, Enterprise Edition 5 (JEE 5) applications using Enterprise JavaBeans 3.0 (EJB 3.0). We also provide a section about working with business-level applications, a new concept that is introduced in WebSphere Application Server v7.0.

We cover the packaging of the following JEE artifacts:

- ▶ Enterprise Archives (EAR)
- ▶ EJB 3.0 modules
- ▶ Web modules
- ▶ JPA persistence units
- ▶ Working with Enhanced EAR files

We also describe some of the IBM enhancements WebSphere Application Server v7.0 supports in addition to the JEE specification.

If you are working on a pre-JEE 5 application or are using EJB 2.1 or earlier modules, for details about the earlier versions, refer to *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304, which is available at:

<http://www.redbooks.ibm.com/abstracts/sg247304.html?Open>

14.1 JEE 5 EAR files

WebSphere Application Server v7.0 supports the JEE 5 and the EJB 3.0 specifications, which make use of annotations. Because developers annotate the source code with information about how it should be deployed, much of the packaging and deployment tasks that were necessary in previous versions of WebSphere Application Server are no longer relevant. Annotations also reduce the number of classes and interfaces the developer needs to manage within the project. With the introduction of JEE 5, developing, packaging, and deploying Java enterprise applications is simplified.

As with previous versions of the Java 2 Enterprise Edition (J2EE) specification JEE 5 applications are packaged in Enterprise Archive (EAR) files. An EAR file can contain Web Archive (WAR) files, EJB modules (packaged as EJB JARs), Resource Adapter Archive (RAR) files, Java Utility Projects (packaged as JAR files), and Application Client modules. Figure 14-1 shows a schematic overview of a JEE EAR file.

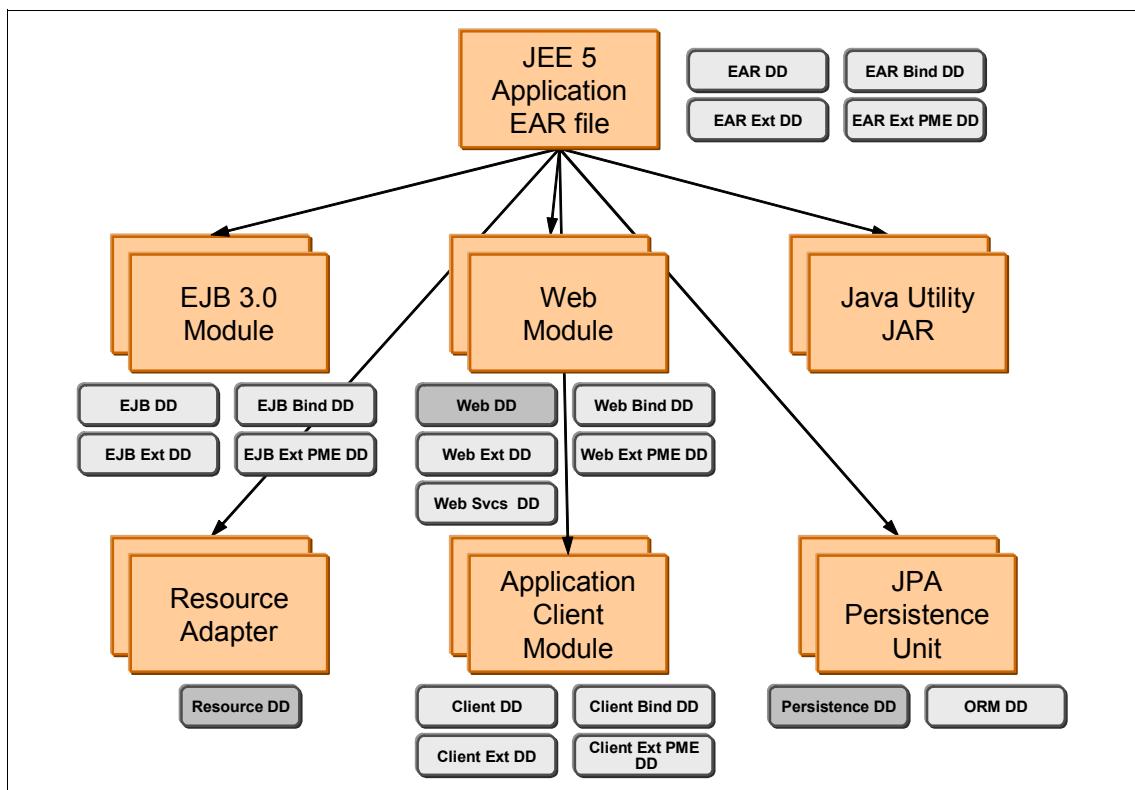


Figure 14-1 JEE 5 EAR file structure

One major difference between JEE 5 applications and previous J2EE applications is that, if the source code is properly annotated, in many cases the EAR file or modules do not need to contain any deployment descriptors. In previous J2EE versions, deployment descriptors were necessary to tell the application server how to deploy the modules and how clients should locate EJB interfaces, for example.

JEE 5, on the other hand, relies on default values, and as long as the default values are acceptable, you do not need to include a deployment descriptor. However, an optional deployment descriptor can be included and it will then override the defaults and the settings given by the annotations in the source code. This gives the deployer the flexibility to deploy the application as preferred for the target environment.

In Figure 14-1 on page 710 the required deployment descriptors are marked with light gray color and the optional ones with dark gray.

Table 14-1 lists the deployment descriptors valid for a WebSphere Application Server v7.0 EAR file.

Table 14-1 Enterprise Archive Deployment Descriptors

Filename	Required	Content
application.xml	No	Defines modules and security roles used by the enterprise application.
ibm-application-bnd.xml	No	Mappings for security roles.
ibm-application-ext.xml	No	WebSphere-specific application extensions.
ibm-application-ext-pme.xml	No	Configuration for WebSphere programming model extensions to the JEE specification.

14.1.1 Development tools

The development tools Rational Application Developer for WebSphere Software 7.5 (RAD) and Rational Application Developer Assembly and Deploy Features for WebSphere 7.0 (RAD-AD) provide editors for all the deployment descriptors.

RAD-AD comes with the WebSphere Application Server license and is a subset of the RAD product. It contains only the Eclipse plug-ins needed to develop, test, package, and deploy J2EE/JEE applications for WebSphere Application Server v7.0, but not the productivity enhancements features found in RAD. RAD-AD 7.0 also only supports testing on a WebSphere Application Server v7.0, while RAD 7.5 supports testing also on previous versions of WebSphere Application Server.

14.1.2 Working with deployment descriptors

To work with the deployment descriptors, start RAD or RAD-AD and create a new JEE application project, or import an existing. To create a new project select **File → New → Enterprise Application Project** and follow the wizards. To import an existing project select **File → Import...** and follow the wizards.

Then, in the Enterprise Explorer view, right-click the JEE project and select **Java EE → Generate Deployment Descriptor Stub**.

Expanding the JEE projects META-INF folder reveals the created `application.xml` deployment descriptor file. To edit it, either double-click the file or double-click the EAR file's deployment descriptor icon, as shown in Figure 14-2.

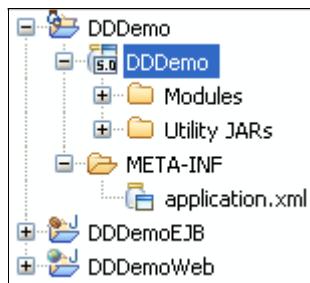


Figure 14-2 JEE EAR deployment descriptor icon in RAD-AD

On the right hand side of the deployment descriptor editor is a panel with fields for the information that can be entered into the deployment descriptor. See Figure 14-3. Fill out the fields and then press **Ctrl-S** to save the deployment descriptor.

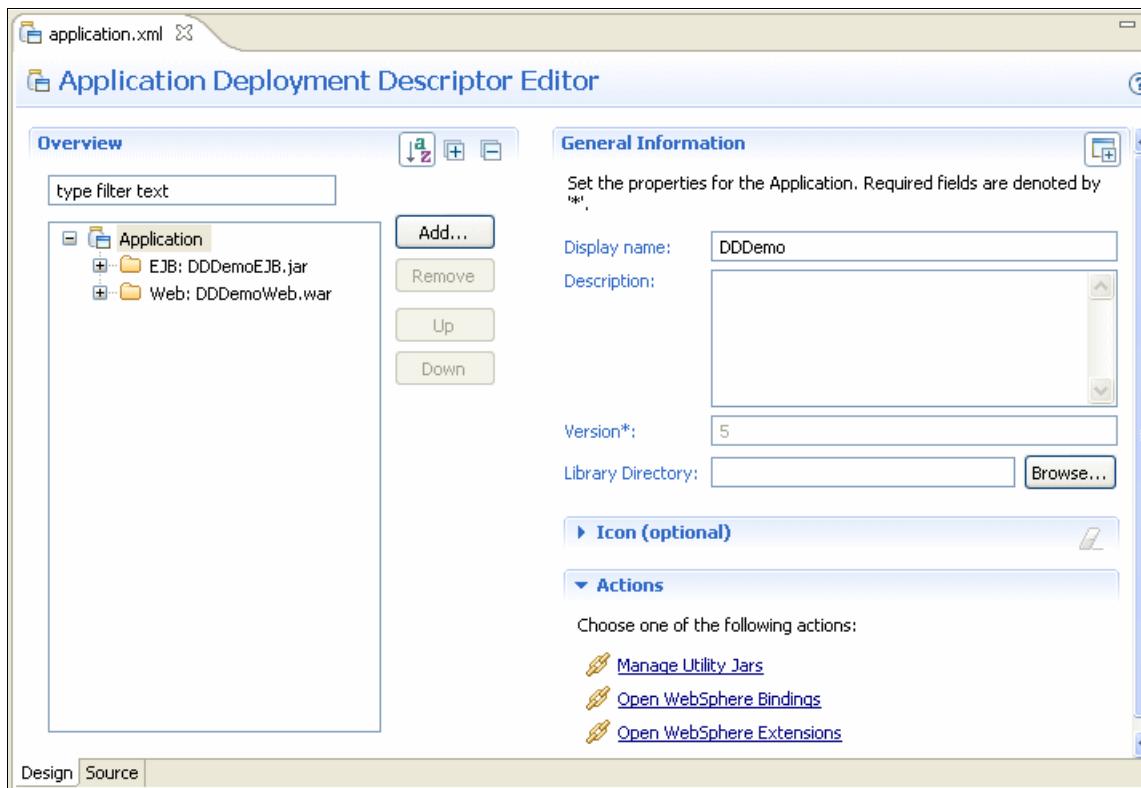


Figure 14-3 JEE EAR deployment descriptor editor

To create any of the other optional JEE EAR-file level deployment descriptors, right-click the JEE EAR project. Select **Java EE** and then one of the remaining deployment descriptor options; **Bindings**, **Extensions**, or **Programming Model Extensions Deployment Descriptor**. Selecting any of these options creates the corresponding deployment descriptor file in the JEE EAR file's META-INF folder. The file can then be accessed either by double-clicking it in the META-INF folder, or by clicking the corresponding link under the Actions heading on the main JEE EAR deployment descriptor editor (Figure 14-3).

14.2 EJB 3.0 modules

EJB 3.0 modules are packaged similar to EJB 2.x modules. One major difference, however is that with EJB 3.0 there is no such thing as Container Manager Persistence (CMP) or Bean Managed Persistence (BMP) beans.

Instead, EJB 3.0 modules should use JPA for data persistence. EJB 2.x and earlier modules still support CMP and BMP beans as per the J2EE specifications and are still supported by WebSphere Application Server v7.0. Table 14-2 lists the deployment descriptors for an EJB 3.0 module.

Table 14-2 EJB 3.0 Deployment Descriptors

Filename	Required	Content
ejb-jar.xml	No	EJB and EJB method definitions, transaction attributes, resource references, and so on.
ibm-ejb-jar-bnd.xml	No	Explicit binding names for EJBs, and EJBs and resource references.
ibm-ejb-jar-ext.xml	No	Configuration of WebSphere extensions to the JEE EJB module specification.
ibm-web-ext-pme.xml	No	Configuration for WebSphere programming model extensions to the JEE specification.

EJB interface bindings

WebSphere Application Server v7.0 binds EJB 3.0 interfaces and homes into two distinct JNDI namespaces, one JVM-local and one global namespace. Local interfaces and homes are bound to the JVM-local namespace, and remote interfaces and homes are bound to the global namespace.

Unless overridden by explicitly assigned bindings, the interfaces are bound using default names generated automatically by the EJB container. Each default name has a short version and a long version. The short name consists of only the Java package name and class name of the interface. The long name prefixes the short name with a component ID, which is composed of the enterprise application name, the module name and the component name.

Consider an enterprise application called RAD75EJBWebEAR that has an EJB module called RAD75EJB.jar with the following bean and interfaces:

- ▶ A session bean with an implementation class called EJBBankBean
- ▶ A local interface called itso.bank.service.EJBBankService
- ▶ A remote interface called itso.bank.service.EJBBankRemote

The auto-generated short and long names for the bean's interfaces are:

- ▶ ejblocal:itso.bank.service.EJBBankService
- ▶ ejblocal:RAD75EJBWebEAR/RAD75EJB.jar/EJBBankBean#itso.bank.service.EJB BankService
- ▶ itso.bank.service.EJBBankRemote

- ejb/RAD75EJBWebEAR/RAD75EJB.jar/EJBBankBean#itso.bank.service.EJBBankR
emote

The local names are bound into the JVM-local namespace called `ejblocal`. The remote names are bound to the global namespace, and to avoid cluttering the root of the namespace the long name is prefixed with `ejb/`.

The auto-generated default names can be overridden by placing a file named `ibm-ejb-jar-bnd.xml` in the EJB JAR module's META-INF directory with the preferred names. By overriding the default names you can define your own naming convention independently from how the beans are packaged into the application/module hierarchy.

EJB reference resolution using the AutoLink feature

When an EJB client (typically a servlet, or another EJB) wants to call an EJB, it first needs to locate the EJB home in the JNDI namespace. In EJB 2.1, and earlier, this had to be done with a few lines of code written explicitly by the EJB client developer. However, with the EJB 3.0 support and source code annotations WebSphere Application Server v7.0 uses a feature called AutoLink which automates this task in many cases, making the lookup code superfluous.

When the EJB container encounters an annotation for an EJB reference, it tries to automatically look up the referenced EJB. The AutoLink algorithm first looks to see if the EJB interface has been explicitly given a name in the module's bindings file. If not found, AutoLink searches within the referring module for an EJB that implements the interface. If it does not find exactly one EJB that implements the interface within the same module, AutoLink expands the search scope and searches within other modules defined in the application. If it finds exactly one EJB that implements the interface, it uses that as the reference target.

The scope of AutoLink is limited to the enterprise application in which the EJB reference appears and within the application server on which the referring module is assigned. If the target EJB resides in an application other than the client's, or it is deployed on an application server other than the client's, then AutoLink does not work. In this case, target bindings must be explicitly defined in the client's bindings file. For an EJB module, this is the `ibm-ejb-jar.bnd.xml` file, and for a Web module, it is the `ibm-web-bnd.xmi` file.

For more information and some examples of how to do this, refer to Chapter 9 in the Redbooks publication, *WebSphere Application Server Version 6.1 Feature Pack for EJB 3.0*, SG24-7611, which is available at:

<http://www.redbooks.ibm.com/abstracts/sg247611.html?Open>

The AutoLink feature only handles EJB references and is available for clients running in the EJB container, Web container, or application client container.

Note: If you installed the Feature Pack for EJB 3.0 WebSphere Application Server v6.1, the default was to scan annotations during the installation of an EJB 3.0 module. For WebSphere Application Server v7.0, the default is not to scan pre-Java EE 5 modules during the application install or at server startup.

To preserve backward compatibility with both the Feature Pack for EJB 3.0 and the Feature Pack for Web Services, you have a choice whether or not to scan existing Web modules for additional metadata. A server level switch is defined for each feature pack scan behavior. If the default is not appropriate, the switch must be set on each server and administrative server that requires a change in the default. The switches are the server custom properties `com.ibm.websphere.webservices.UseWSFEP61ScanPolicy={truelfalse}` and `com.ibm.websphere.ejb.UseEJB61FEPEScanPolicy={truelfalse}`.

Go to **Servers** → **Server Types** → **WebSphere application servers** → **server_name** → **Java and Process Management** → **Process Definition** → **Java Virtual Machine** → **Custom Properties**, and add a new property.

Just-In-Time generation of EJB deployed code

EJB modules must contain EJB deployed code in order for an application server to be able to run the EJBs. EJB deployed code contains application server specific code that bridges the EJB interface and implementation code to the application server's EJB implementation.

In previous versions of WebSphere Application Server and with previous versions of J2EE, the EJB deployed code could be generated using the following methods:

- ▶ During development, using the Prepare for Deploy action in Rational Application Developer or the Application Server Toolkit
- ▶ Before installing an EAR file to WebSphere Application Server, using the EJBDeploy tool from command line
- ▶ During installation of an EAR file to WebSphere Application Server, using the install panels in the administrative console

WebSphere Application Server v7.0 and the EJB 3.0 support introduces a new feature called Just-In-Time deployment. This feature removes the need to process the EJB modules to generate the deployed code. Instead, the EJB container dynamically generates the necessary code in-memory as needed when the application is running. This feature simplifies and speeds up the development, packaging, and deployment of EJBs.

For EJB 3.0 clients that are not running inside a Web container, EJB container, or client container that has been upgraded to the EJB 3.0 level, the Just-In-Time development does not generate the necessary classes. In this case, the createEJBStubs tool should be used and the generated classes would be made available on the client's classpath.

An example scenario where this is applicable is a servlet running in WebSphere Application Server v6.1 calling an EJB 3.0 bean running in WebSphere Application Server v7.0. In this case, the EJB stubs should be created manually and the generated classes added to the servlet's Web module.

For details and syntax on the createEJBStubs tool, refer to the WebSphere Application Server v7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rejb_3stubscmd.html

Mixing different EJB versions in an EAR file

WebSphere Application Server v7.0 allows you to mix EJB 1.1, 2.x and 3.0 beans in the same JEE 5 enterprise application. EJB 1.1 and 2.x beans can be directly carried forward in EJB 1.1 and EJB 2.x modules respectively. WebSphere can also handle EJB 1.1, 2.x and 3.0 session beans, and EJB 2.x message driven beans within the same EJB 3.0 module. CMP and BMP entity beans are not supported in EJB 3.0 modules and must remain in EJB 1.1 or EJB 2.x modules.

14.3 JPA persistence units

Persistence units using Java Persistence API (JPA) can be packaged either into the module that uses the persistence unit or in a separate utility JAR file (packaged as a standard .jar file). If packaged as a separate utility JAR file, it must be referenced from the module that uses the persistence unit using the module's META-INF.MF Class-path directive.

Persistence units require a `persistence.xml` file, which defines a JPA entity manager's configuration. Among other information, the `persistence.xml` file lists the entity classes and the data source to use.

A persistence unit can also include an optional `orm.xml` file which specifies the object-relational mapping configuration. The `orm.xml` file is an alternative to using annotations and can be used to override annotations in the source code to specify how the objects should be persisted to the database.

Table 14-3 lists the deployment descriptors valid for a JPA persistence unit.

Table 14-3 JPA Persistence Units Deployment Descriptors

Filename	Required	Content
persistence.xml	Yes	Entity manager's configuration, entity classes, data sources, and so on.
orm.xml	No	Object-to-relational mapping annotation overrides.

14.3.1 JPA access intents

WebSphere Application Server provides an optimization enhancement for EJB 2.x entity beans called access intents. However, because the EJB 3.0 specification does not support entity beans, the access intents support is not available for EJB 3.0 beans. Instead, WebSphere Application Server v7.0 provides JPA access intents which can be used to improve performance and scalability for JPA applications.

JPA access intents specify the isolation level and lock level used when reading data from a data source. JPA access intents can be used, providing that the following restrictions are honored:

- ▶ Access intent is available for the application in the Java EE server environment
- ▶ Access intent is applicable to non-query entity manager interface methods. Query should use query hint interface to set its isolation and read lock values.
- ▶ Access intent is only available for DB2 databases.
- ▶ Access intent is in effect only when pessimistic lock manager is used. To specify a pessimistic lock manager add the following statement to the persistence unit's property list:

```
<property name="openjpa.LockManager" value="pessimistic"/>.
```

Table 14-4 compares EJB 2.x access intents to JPA access intents.

Table 14-4 JPA Access Intents properties

EJB 2.x entity bean access intent	JPA access intent	Description
optimistic	isolation: Read Committed	Data is read but no lock is held. Version id is used on update to insure data integrity. Other transactions can read and update data.
	lockManager: Optimistic	
	query Hint: ReadLockMode: READ	

EJB 2.x entity bean access intent	JPA access intent	Description
pessimistic read	isolation: Repeatable Read	Data is read with shared locks. Other transactions attempting to update data are blocked.
	lockManager: Optimistic	
	query Hint: ReadLockMode: READ	
pessimistic update	isolation: Repeatable Read	Data is retrieved with update or exclusive lock. Other writes are blocked until commit. This access intent can be used to serialize update access to data when there are multiple writers.
	lockManager: Pessimistic	
	query Hint: ReadLockMode: WRITE	
pessimistic exclusive	isolation: Serializable	Data is retrieved with update or exclusive lock. Other writes are blocked until commit. This access intent can be used to serialize update access to data when there are multiple writers.
	lockManager: Pessimistic	
	query Hint: ReadLockMode:WRITE	

JPA access intents are specified in the `persistence.xml` deployment descriptor.

For more information about access intents, search the Information Center for JPA Access Intent.

For information about EJB 2.x access intents, see Section 13.5 in the Redbooks publication, *WebSphere Application Server v6.1*, SG24-7304, available at:

<http://www.redbooks.ibm.com/abstracts/sg247304.html?Open>

14.4 Resource adapters

A resource adapter archive (RAR) module, also called a connector module, contains code that implements a library for connecting with a back-end Enterprise Information System (EIS), such as CICS, SAP, and PeopleSoft. RAR files (called connectors) are packaged as a standard Java Archive with a .rar file extension. A resource adapter can be installed as a stand alone adapter or as part of an enterprise application, in which case the resource adapter is referred to as an embedded adapter.

A connector module contains a mandatory deployment descriptor file named `ra.xml`, residing in the module's META-INF directory.

14.5 Web modules

JEE 5 Web modules are packaged just like Web modules in earlier J2EE versions. A Web module can contain servlet code, JSPs, static HTML pages, images, JavaScript, stylesheets, and so on.

A common challenge when working with Web modules is to make sure the right version of a required Java library is loaded. Often Web application developers need to include specific third party libraries such as log4j, or Xalan/Xerces, and must make sure that the correct version of a library is loaded for an application. This requires knowledge on how the EAR and Web module's class loaders work. Refer to Chapter 13, "Understanding class loaders" on page 677 for detailed information about this topic.

A Web module supports several deployment descriptors, as shown in Table 14-5.

Table 14-5 Web module deployment descriptors

Filename	Required	Purpose
<code>web.xml</code>	Yes	Servlet definitions, URL mappings, and init parameters, servlet listeners, and so on.
<code>ibm-web-bnd.xml</code>	No	Mapping of logical resources used by the Web module to their runtime managed resources.
<code>ibm-web-ext.xml</code>	No	Configuration of WebSphere extensions to the JEE Web module specification.
<code>ibm-web-ext-pme.xml</code>	No	Configuration for WebSphere programming model extensions to the JEE specification.
<code>webservices.xml</code>	No	Configuration of Web services, and implementation code.

Note: If an `application.xml` deployment descriptor is not included in the EAR file, the context root for a Web module defaults to the Web module's name without the `.war` extension.

14.5.1 WebSphere extensions to Web modules

WebSphere Application Server provides multiple extensions for Web modules. These are configured in the `ibm-web-ext.xml` deployment descriptor in the Web module. To create this file in RAD or RAD-AD, right-click the Web module in the Enterprise Explorer view and select **Java EE → Generate WebSphere Extensions Deployment Descriptor**. To edit the file, either expand the Web module's `WebContent/WEB-INF` folder and double-click the `ibm-web-ext.xml` file, or click the **Open WebSphere Extensions** link on the Web module's deployment descriptor editor for the `web.xml` file, as shown in Figure 14-4.

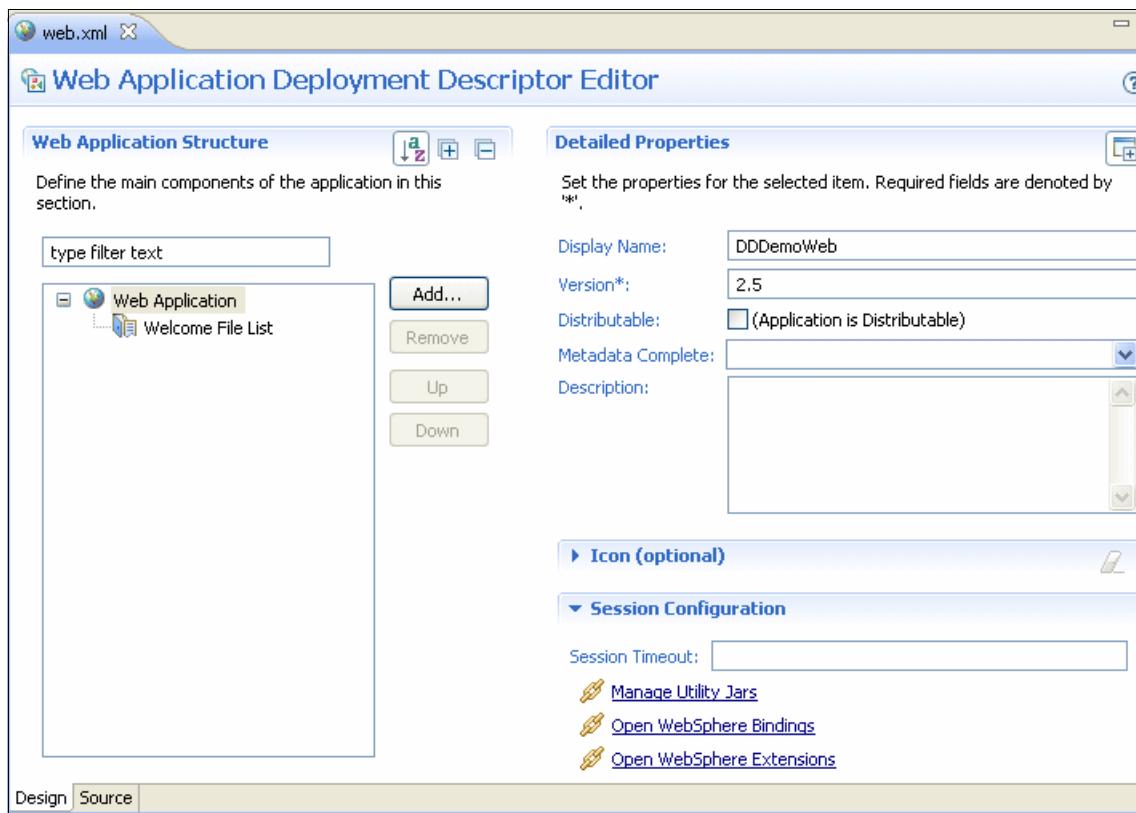


Figure 14-4 Web module's deployment descriptor editor

The Web module extensions editor is shown in Figure 14-5.

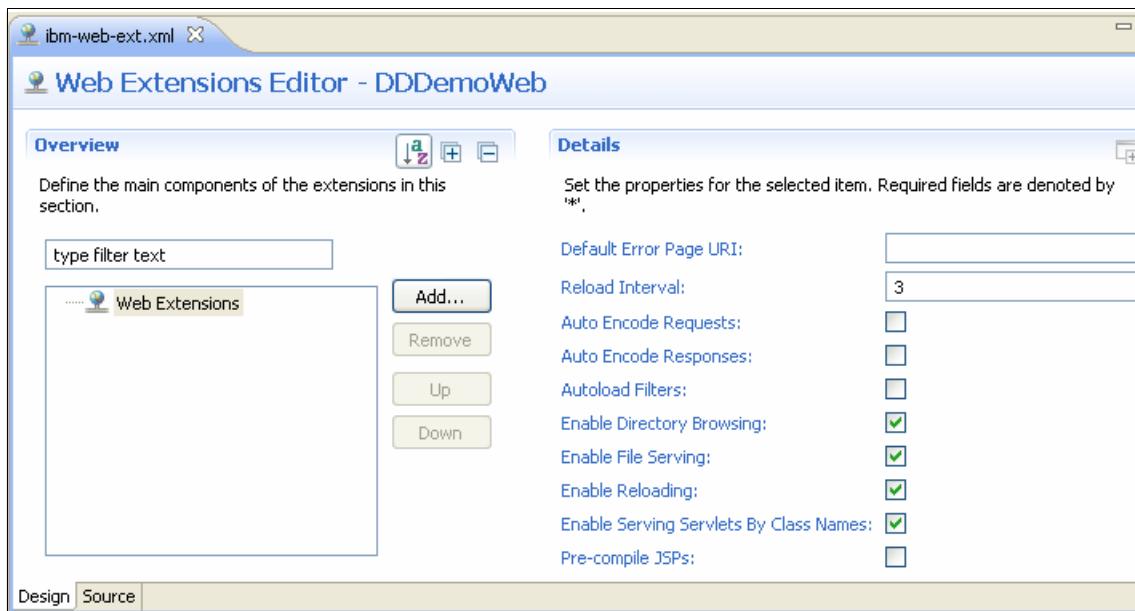


Figure 14-5 Editing WebSphere Web module extensions

Note: Previous versions of WebSphere Application Server provided a mechanism to share HTTP sessions across Web modules. Because this mechanism is not compliant with the servlet API specification, it had to be explicitly enabled for applications that required it. In WebSphere Application Server v7.0, this mechanism has been deprecated. If you have used this feature, you should redesign your application so that sessions are scoped at the Web module instead. If data really must be shared across Web module boundaries, use the IBM-enhanced session object, `IBMAplicationSession`, instead. You can search the Information Center for `IBMAplicationSession` for information about this.

14.5.2 File serving

When dealing with static content (HTML pages, images, style sheets, and so on), you can choose to have these resources served by WebSphere, or have them served by the HTTP server itself.

If you want WebSphere to serve the static content of your application, you must enable file serving. Enabling this activates a servlet which serves up any resource file packaged in the WAR file. The File serving enabled attribute is set to true by default. By changing it to false, the Web server plug-in will not send requests for static content to WebSphere, but leave it up to the HTTP server to serve them.

If you want the Web server to serve static content, you can experience better performance than using WebSphere in this instance, because the Web server is serving the content directly. Moreover, a Web server has more customization options than the file servlet can offer.

However, using the WebSphere file serving servlet has the advantage of keeping the static content organized in a single, deployable unit with the rest of the application. Additionally, this allows you to protect the static pages using WebSphere security.

To enable this option, check the **Enable File Serving** box.

14.5.3 Web application auto reload

If you check the **Enable Reloading** option, the class path of the Web application is monitored and all components, JAR or class files, are reloaded whenever a component update is detected. The Web module's class loader is shut down and restarted. The **Reload Interval** is the interval between reloads of the Web application. It is set in seconds.

The auto reload feature plays a critical role in hot deployment and dynamic reload of your application.

Important: You must set the Enable Reloading enabled option to true for JSP files to be reloaded when they are changed on the file system. Reloading a JSP does not trigger the reload of the Web module, because separate class loaders are used for servlets and JSP.

This option is enabled by default, with the reload interval set to 3 seconds. Thus, the classloader checks the classes on the classpath for updates every 3 seconds, and if any changes are found those classes are reloaded. However, unless changes are detected nothing should happen to the classloader or the classes loaded. In production mode, you might consider making the reload interval much higher.

14.5.4 Serve servlets by class name

The invoker servlet can be used to invoke servlets by class name. Note that there is a potential security risk with leaving this option set in production. It should be seen as more of a development-time feature, for quickly testing your servlets.

A better alternative than this option is to define servlet mappings in the Web deployment descriptor for the servlets that should be available.

The invoker servlet is configured by the Enable Serving Servlets By Class Names option.

14.5.5 Default error page

This page will be invoked to handle errors if no error page has been defined, or if none of the defined error pages matches the current error.

14.5.6 Directory browsing

This Boolean defines whether it is possible to browse the directory if no default page has been found.

This option should be turned off for improved security.

14.5.7 Pre-compile JSPs

When a JSP is hit for the first time, it is automatically compiled into a servlet and then executed. To avoid this performance penalty the first time a JSP is hit, WebSphere allows JSPs to be pre-compiled during application installation instead of at first invocation. Selecting this option will cause the installation of the application to WebSphere to take longer, but the JSPs will be served faster on the first hit.

14.5.8 Automatic HTTP request and response encoding

The Web container no longer automatically sets request and response encodings and response content types. The programmer is expected to set these values using the methods available in the Servlet 2.4, and later, API. If you want the application server to attempt to set these values automatically, check the **Auto Encode Requests** option in order to have the request encoding value set. Similarly, you can check the **Auto Encode Responses** option in order to have the response encoding and content type set.

The default value of the autoRequestEncoding and autoResponseEncoding extensions is false, which means that both the request and response character encoding is set to the Servlet 2.4 specification default of ISO-8859-1. Different character encodings are possible if the client defines character encoding in the request header, or if the code uses the setCharacterEncoding(String encoding) method.

If the autoRequestEncoding value is set to true, and the client did not specify character encoding in the request header, and the code does not include the setCharacterEncoding(String encoding) method, the Web container tries to determine the correct character encoding for the request parameters and data.

The Web container performs each step in the following list until a match is found:

1. Looks at the character set (charset) in the Content-Type header.
2. Attempts to map the server's locale to a character set using defined properties.
3. Attempts to use the DEFAULT_CLIENT_ENCODING system property, if one is set.
4. Uses the ISO-8859-1 character encoding as the default.

If you set the autoResponseEncoding value to true and the following conditions are true:

- ▶ The client did not specify character encoding in the request header.
- ▶ The code does not include the setCharacterEncoding(String encoding) method.

Then the Web container performs the following actions:

- ▶ It attempts to determine the response content type and character encoding from information in the request header.
- ▶ It uses the ISO-8859-1 character encoding as the default.

14.6 Example: Packaging an application

As an example of how to package a JEE 5 application using EJB 3.0 beans, we use the same ITSOBank application developed by the team who wrote the Redbooks publication, *Rational Application Developer V7.5 Programming Guide*, SG24-7672. To download the sample application, go to:

<http://www.redbooks.ibm.com/redpieces/abstracts/sg247672.html?Open>

From there, click the **Additional Material** link. Then download the file `7672codesolution.zip` and unpack it to a directory on your computer. Unpacking the ZIP file creates a number of directories. The directory that we are interested in is the `ejb` directory. This directory contains two ZIP files with RAD project interchange files. We will now import both files into RAD-AD:

1. Start RAD-AD.
2. To import the code select **File → Import...** Expand the **Other** section and select **Project Interchange**. Click **Next**.
3. Click the **Browse** button next to the From zip file field and browse to the `ejb` directory where you unzipped the sample code. Select the **RAD75EJB.zip** file and click **Open**.
4. Click the **Select All** button to select all projects in the file.
5. Then click **Finish**.
6. Repeat the process for the **RAD75EJBWeb.zip** project.

When the project files have been imported, the Enterprise Explorer view in RAD should look like Figure 14-6.



Figure 14-6 ITSO Bank application imported into RAD-AD

The Problems view shows 15 Warnings, but none of them are critical.

The workspace now has two Enterprise Application (EAR) projects, called RAD75EJBear and RAD75EJBWebEAR. RAD75EJBear contains some EJBs and a simple servlet for testing (in the RAD75EJBTTestWeb project). A more sophisticated Web application is available in the RAD75EJBWeb project, and this is the one we will use in our example.

The RAD75EJBWeb project uses the EJBs in the RAD75EJB project, which in turn relies on the Persistence Unit in the RAD75JPA project.

We will customize the RAD75EJBWeb project a little and export it as an EAR file. The EAR file will be deployed in Chapter 15, “Deploying applications” on page 753.

None of the actions that we do here are actually necessary for getting the ITSO Bank application to work, because the development team has done everything necessary in the Redbooks publication, *Rational Application Developer V7.5 Programming Guide*, SG24-7672. However, to show you some common packaging tasks and the functionality of the RAD-AD development tool, we do some customizations to the application.

The first thing we will is to remove the unnecessary deployment descriptors, which were included in the application by the development team.

- ▶ Expand the **RAD75EJB** project and expand **ejbModule**. Then expand the **META-INF** folder and delete the **ejb-jar.xml** file.
- ▶ Expand the **RAD75EJBWebEAR** project and expand its **META-INF** folder. Delete the **application.xml** file.
- ▶ Expand the **RAD75JPA** project and expand **src**. Then expand the **META-INF** folder and delete the **orm.xml** file.

The RAD75EJB project depends on the Persistence Unit defined in the RAD75JPA project. To verify that this dependency is correctly set up, right-click the **RAD75EJB** project and select **Properties**. Then click **Java EE Module Dependencies** in the left pane. The right pane shows that the RAD75JPA.jar project is selected, which means that the EJB project can access the classes in the JPA project. See Figure 14-7.

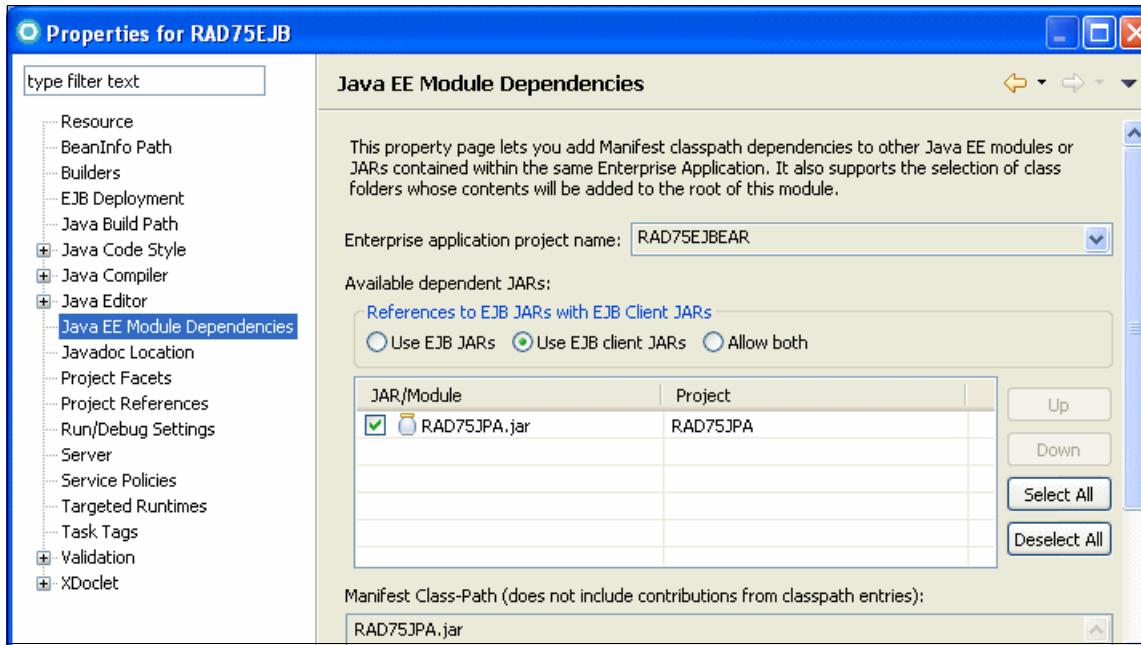


Figure 14-7 Java EE Module Dependencies

14.6.1 Configuring Web module extensions

For our application we customize the WebSphere Web module extensions:

1. Expand the **RAD75EJBWeb** project and double-click the **RAD75EJBWeb** heading (Figure 14-8) to open the Web module deployment descriptor.

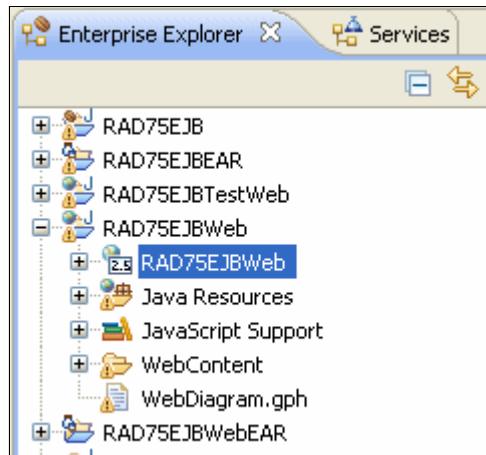


Figure 14-8 Opening the Web module deployment descriptor

- When the panel opens, click the **Open WebSphere Extensions** link in the bottom right corner as shown in Figure 14-9 to open the extensions editor.

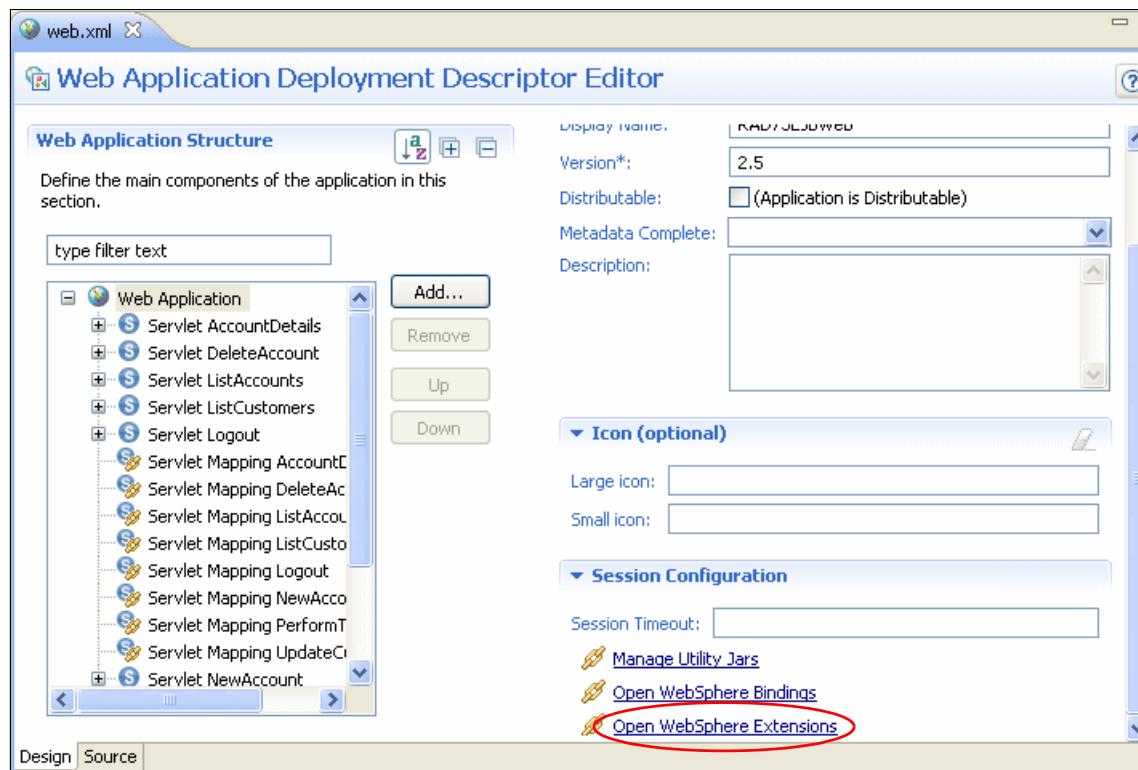


Figure 14-9 Web module deployment descriptor editor

- The Web module extensions editor contains options to configure the optional WebSphere extensions to Web modules. In our application, we do not want to serve servlets by classname and we also want to prevent directory browsing.
- Therefore, we uncheck the corresponding check boxes as shown in Figure 14-10. It is a best practice to disable these options in production environments so only the servlets and folders the developers had intended to are accessible.

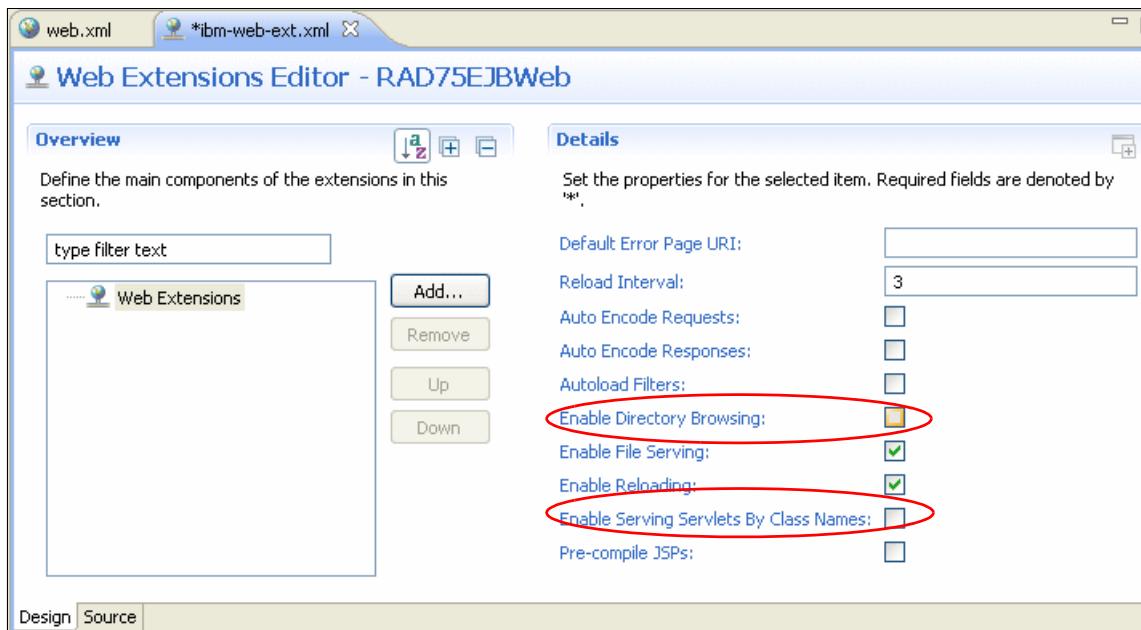


Figure 14-10 Web module extensions editor

5. When done, press Ctrl-S to save the deployment descriptor.

14.7 Exporting to an EAR file

To export the RAD75EJBWebEAR application as an EAR file with all its dependent modules, do the following steps:

1. Select the **RAD75EJBWebEAR** project and right-click. Select **Export...** and **EAR file** from the pop-up menu.
2. In the Export dialog, browse to a destination, such as C:\, as shown in Figure 14-11.

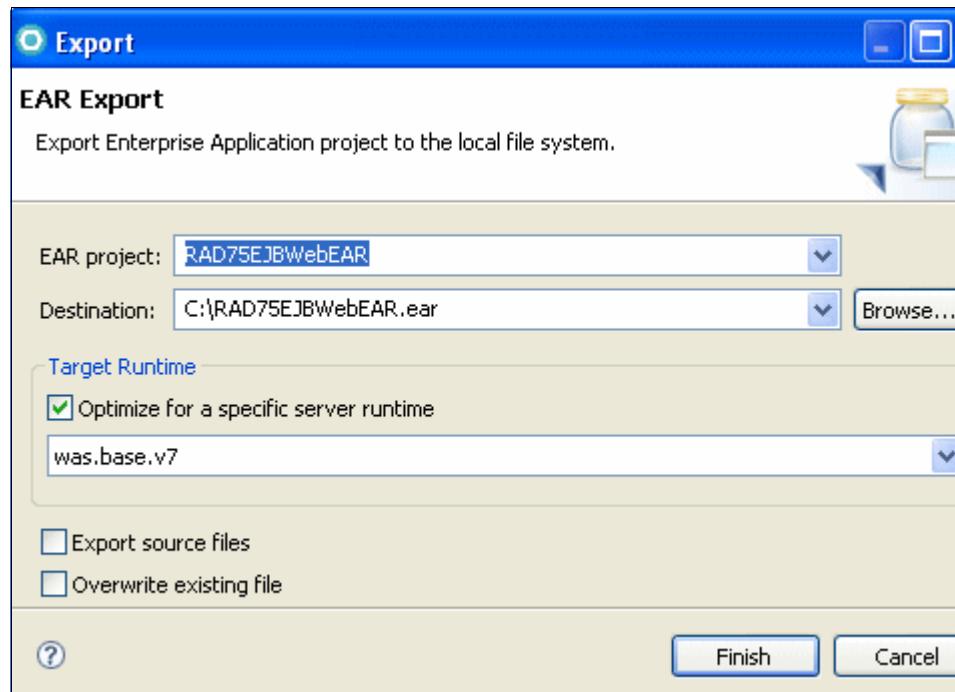


Figure 14-11 Exporting Enterprise Project to EAR file

3. Click **Finish**.
4. The EAR file exported is now prepared for installation in WebSphere Application Server v7.0.

You can optionally add more configuration details to the EAR file by creating a WebSphere Enhanced EAR file.

14.8 WebSphere Enhanced EAR

A WebSphere enhanced EAR is a regular JEE EAR file, but with additional configuration information for resources required by JEE applications. While adding this extra configuration information at packaging time is not mandatory, it can simplify deployment of JEE applications to WebSphere if the environments where the application is to be deployed are similar.

When an Enhanced EAR is deployed to WebSphere Application Server, WebSphere can automatically configure the resources specified in the Enhanced EAR. This reduces the number of configuration steps required to set up the WebSphere environment to host the application.

When an Enhanced EAR is uninstalled, the resources that are defined at the application level scope are removed as well. However, resources defined at a scope other than application level are not removed because they might be in use by other applications. Resources created at the Application level scope are limited in visibility to only that application.

Table 14-6 shows the resources supported by the Enhanced EAR and the scope in which they are created.

Table 14-6 Scope for resources in WebSphere Enhanced EAR file

Resource	Scope
JDBC providers	Application
Data sources	Application
Resource adapters	Application
JMS resources	Application
Substitution variables	Application
Class loader policies	Application
Shared libraries	Server
JAAS authentication aliases	Cell
Virtual hosts	Cell

J2C Resource Adapters can be configured either as embedded or external resources. An embedded RAR is packaged within an enterprise application (EAR), deployed as a part of JEE application installation, and is removed when the application is uninstalled from the server. An external RAR is packaged as a stand-alone RAR file, is deployed explicitly on a WebSphere node, and is not managed as a JEE application. If an adapter is used only by a single application, it should be configured as an embedded RAR. If it is to be shared between multiple applications, it should be an external RAR.

To view the application scoped resources using the administrative console, select **Applications** → **Application Types** → **WebSphere Enterprise Applications** → *<application>*. Select **Application scoped resources** in the References section. If there are no application scoped resources, you will not see this option.

14.8.1 Configuring a WebSphere Enhanced EAR

The supplemental information in an Enhanced EAR is modified by using the WebSphere Application Server Deployment editor. The information itself lives in XML files in a folder called `ibmconfig` in the EAR file's META-INF folder.

To access the Enhanced EAR deployment options right-click the **RAD75EJBWebEAR** project and select **Java EE**, and then the **Open WebSphere Application Server Deployment** option. This opens up the editor as shown in Figure 14-12.

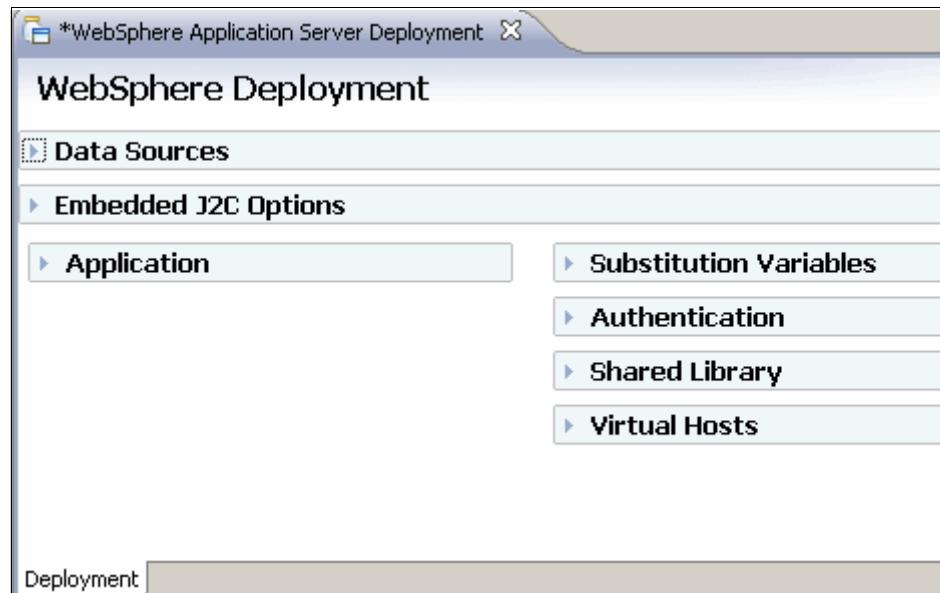


Figure 14-12 WebSphere Enhanced EAR editor

In the Application section in Figure 14-13, you can see the class loader policies and class loader mode configured for each of the containing module. ITSO Bank runs fine with the default policies and modes, so they do not need to be changed.

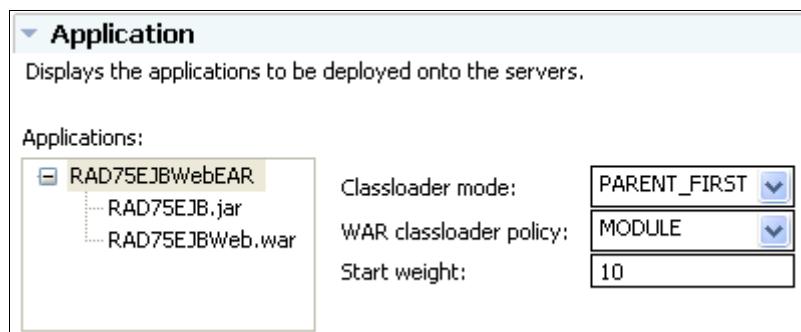


Figure 14-13 Configuring class loader mode and class loader policies

When creating an Enhanced EAR file, RAD-AD automatically adds a JDBC provider for the Derby database. We will change this to use DB2 instead as that is what we will use when deploying the application. To achieve this we need to add the following items:

- ▶ JAAS authentication alias
- ▶ JDBC provider for DB2
- ▶ Data source for DB2 database

Just to show the use of the editor, we will also configure a new virtual host for a domain called www.itsobank.ibm.com.

Configuring a JAAS authentication alias

To configure the JAAS authentication alias, do the following steps:

1. Expand the **Authentication** section.
2. Click the **Add** button.
3. In the dialog box that displays, enter:
 - itsobank as the alias
 - A user ID with access to the ITSOBANK database (db2inst1 in our case)
 - The password for the user ID.
 - ITSO Bank as the description

4. Click **OK**. See Figure 14-14 for the results.

The screenshot shows a configuration dialog for JAAS authentication. At the top, a section titled "JAAS authentication list:" contains a table with one row. The table has columns for "Alias" (containing "itsobank"), "User ID" (containing "db2inst1"), and "Description" (containing "ITSO Bank"). To the right of the table are three buttons: "Add...", "Edit...", and "Remove".

Figure 14-14 Configuring JAAS authentication alias for ITSO Bank

Configuring a DB2 JDBC provider

JDBC providers are configured in the Data Sources section, so expand this section. Before adding the DB2 provider, first delete the pre-configured Derby JDBC Provider (XA) provider by selecting it and clicking the **Remove** button.

To configure the DB2 JDBC provider, do the following steps:

1. Click the **Add** button next to the JDBC provider list.
2. In the dialog box:
 - Select **IBM DB2** as the Database type.
 - Select **DB2 Using IBM JCC Driver (XA)** as the JDBC provider type.

See Figure 14-15.

The screenshot shows a configuration dialog for creating a DB2 JDBC provider. It consists of two main sections: "Database type:" and "JDBC provider type:".
The "Database type:" section contains a list of database types: User-defined, IBM DB2, Cloudscape, Informix, Sybase, and Oracle. "IBM DB2" is selected.
The "JDBC provider type:" section contains a list of JDBC provider types: DB2 Using IBM JCC Driver, DB2 Using IBM JCC Driver (XA), DB2 Universal JDBC Driver Provider, DB2 Universal JDBC Driver Provider (XA), DB2 Legacy CLI-based Type 2 JDBC Driver, and DB2 Legacy CLI-based Type 2 JDBC Driver (XA). "DB2 Using IBM JCC Driver (XA)" is selected.

Figure 14-15 Creating a DB2 JDBC Provider

Click **Next**.

3. In the next dialog box, enter a name for the JDBC provider (for administration purposes only) and leave the other properties as the default values. See Figure 14-16.

Name:	ITSO Bank DB2 JDBC Provider (XA)
Description:	DB2 Using IBM JCC Driver (XA)
Implementation class name:	com.ibm.db2.jcc.DB2XADataSource
Class path:	<code> \${DB2_JCC_DRIVER_PATH}/db2jcc4.jar \${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar \${DB2_JCC_DRIVER_PATH}/db2jcc_license_cisuz.jar</code>
Native path:	<code> \${DB2_JCC_DRIVER_NATIVEPATH}</code>

Figure 14-16 Creating a DB2 JDBC provider

Click **Finish**.

4. Select the **ITSO Bank DB2 JDBC Provider (XA)** you just created and click the **Add** button next to the Data source list, as in Figure 14-17.

The screenshot shows the 'WebSphere Deployment' interface with the 'Data Sources' section selected. The 'JDBC provider list' table contains one entry: 'ITSO Bank DB2 JDBC Provider (XA)' with the implementation class 'com.ibm.db2.jcc.DB2XADataSource'. To the right of this table are three buttons: 'Add...', 'Edit...', and 'Remove'. The 'Add...' button is circled in red. Below the provider list is a table titled 'Data source defined in the JDBC provider selected above'. This table is currently empty. To its right are three buttons: 'Add...', 'Edit...', and 'Remove'. The 'Add...' button is also circled in red. At the bottom is a table titled 'Resource properties defined in the data source selected above', which is also empty. To its right are three buttons: 'Add...', 'Edit...', and 'Remove'.

Name	Implementation Class Name
ITSO Bank DB2 JDBC Provider (XA)	com.ibm.db2.jcc.DB2XADataSource

Name	JNDI Name	Type
------	-----------	------

Name	Value	Type
------	-------	------

Figure 14-17 Creating a DB2 data source

5. In the Create a Data Source dialog box, select **DB2 Using IBM JCC Driver (XA)** as the JDBC provider type and **Version 5.0 data source** as the data source type, as in Figure 14-18.

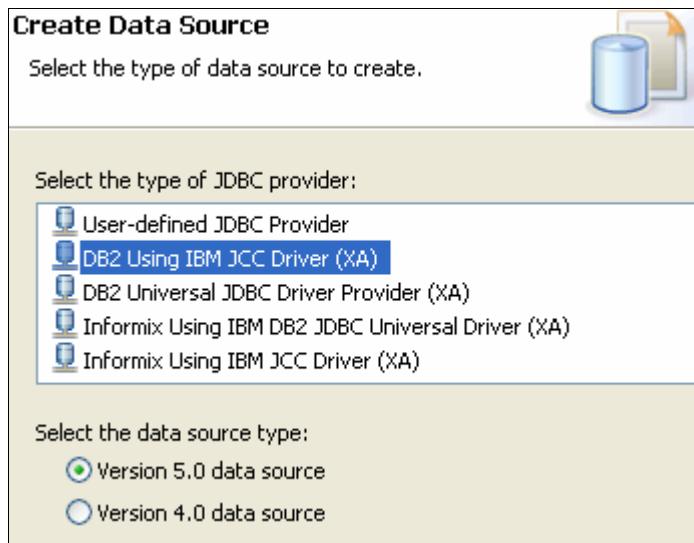


Figure 14-18 Creating a DB2 data source

Click **Next**.

6. In the dialog box displayed enter the appropriate values for the DB2 data source. See Figure 14-19.

Name:	ITSOBankDS
JNDI name:	jdbc/itsobank
Description:	DB2 Data Source for ITSO Bank
Category:	
Statement cache size:	10
Data source helper class name:	com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper
Connection timeout:	180
Maximum connections:	10
Minimum connections:	1
Reap time:	180
Unused timeout:	1800
Aged timeout:	0
Purge policy:	EntirePool
Component-managed authentication alias:	
Container-managed authentication alias:	itsobank
<input type="checkbox"/> Use this data source in container managed persistence (CMP)	
* Required field.	

Figure 14-19 Creating a DB2 data source

- Enter ITSOBankDS as the name.
- Enter jdbc/itsobank as the JNDI name.
- Enter DB2 Data Source for ITSO Bank as the description.
- Select **itsobank** as the Container-managed authentication alias (you might need to scroll the window to the far right to see the pull down arrow).
- Uncheck **Use this data source in container manager persistence (CMP)**. The ITSO Bank application uses JPA for persistence so we do not need to add support for CMP Entity beans for this data source.

Click **Next**.

7. Highlight the **databaseName** property in the Resource properties section. Enter ITSOBANK in the Value field. Then highlight the **driverType** property and change the value from type 4 to type 2. Type 2 means that the DB2 database is installed on the same machine as WebSphere Application Server, or that the DB2 Client software is installed on the same machine and configured to handle the remote connection if the database is on a remote machine. A type 4 driver can connect directly to a remote database over TCP/IP. See Figure 14-20.

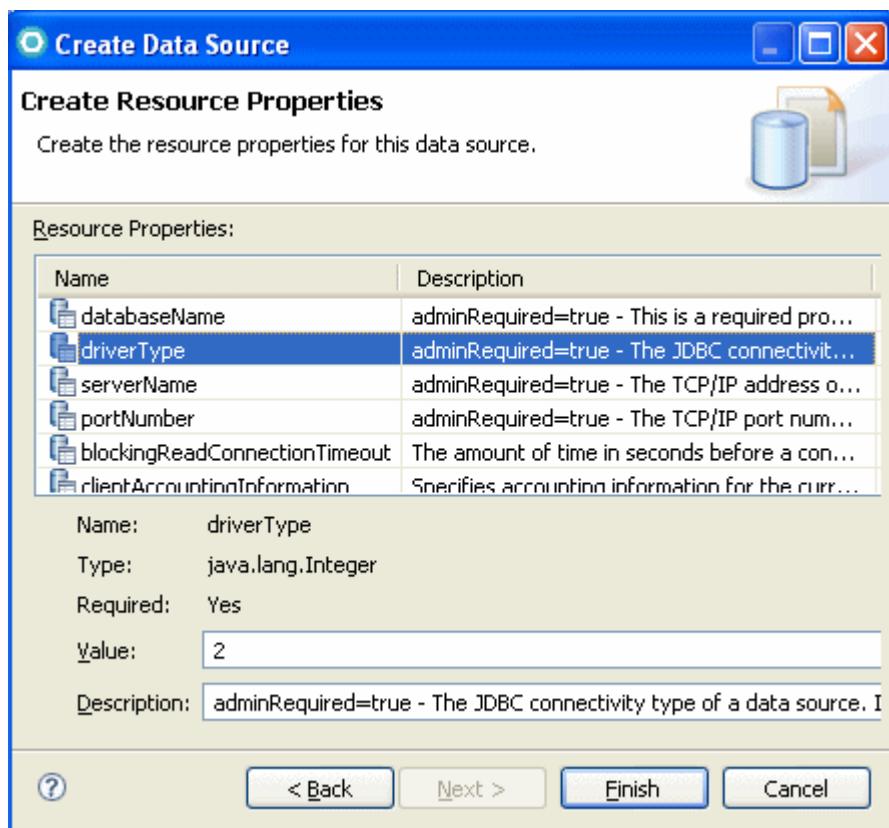


Figure 14-20 Setting database name and driver type

Click **Finish**.

When you are finished, your data source configuration should look like Figure 14-21.

The screenshot shows the 'Data Sources' section of the WebSphere Application Server Deployment tool. It displays three tables of configuration details:

- JDBC provider list:** A table with columns 'Name' and 'Implementation Class Name'. One row is shown: 'ITSO Bank DB2 JDBC Provider (XA)' with 'com.ibm.db2.jcc.DB2XADataSource' in the implementation class name column. To the right are 'Add...', 'Edit...', and 'Remove' buttons.
- Data source defined in the JDBC provider selected above:** A table with columns 'Name', 'JNDI Name', and 'Type'. One row is shown: 'ITSOBankDS' with 'jdbc/itsobank' in the JNDI Name column and 'V5' in the Type column. To the right are 'Add...', 'Edit...', and 'Remove' buttons.
- Resource properties defined in the data source selected above:** A table with columns 'Name', 'Value', and 'Type'. Four rows are listed:
 - 'databaseName': Value 'ITSOBANK', Type 'java.lang.String'
 - 'driverType': Value '2', Type 'java.lang.Integer'
 - 'serverName': Value '0.0.0.0', Type 'java.lang.String'
 - 'socketPort': Value '20000', Type 'java.lang.Integer'To the right are 'Add...', 'Edit...', and 'Remove' buttons, along with up and down arrows for reordering.

Figure 14-21 DB2 data source configured

Adding a virtual host

To configure a virtual host, do the following steps:

1. Expand the **Virtual Hosts** section of the Deployment tab and click the **Add** button next to the Virtual host name list.
2. In the Add Host Name Entry dialog box, enter `itsobank_host` and click **OK**. Your new virtual host will appear in the Virtual Hosts list. See Figure 14-22.

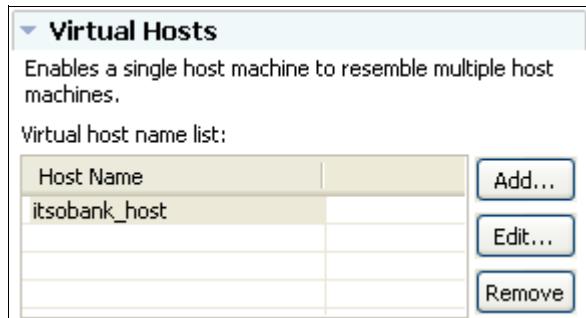


Figure 14-22 Add a new virtual host

3. Click the **Add** button next to the Host aliases list.
4. In the Add Host Alias Entry dialog box, enter www.itsobank.ibm.com for the host name and 80 for the port number. Click **OK**.

Repeat the procedure and add number 9080 as well. We will use this port when we deploy the application later. If your server uses another port use that port number instead. You can have as many host aliases as you like.

The host aliases will appear in the list (Figure 14-23).



Figure 14-23 Configuring the virtual host for ITSO Bank

5. When you are finished, press **Ctrl-S** to save the deployment descriptor editor.

Setting the default virtual host for Web modules

Just because we have configured a new virtual host, itsobank_host, in the Enhanced EAR file does not mean that all our Web modules automatically use it.

The default virtual host for a Web module created in the RAD or RAD-AD is default_host, which is also the case for the ITSO Bank application.

To modify the Web modules to use the itsobank_host instead, do the following steps:

1. Expand the **RAD75EJBWeb** project and double-click the **RAD75EJBWeb** heading as shown in Figure 14-8 on page 728 to open the Web module deployment descriptor.
2. In the lower right corner of the panel click the **Open WebSphere Bindings** link.
3. Change the Virtual Host Name to itsobank_host, as shown in Figure 14-24.



Figure 14-24 Setting default virtual host for a Web module

4. Save the deployment descriptor by pressing **Ctrl-S** and then close it.

Examining the WebSphere Enhanced EAR file

The information about the resources configured is stored in the ibmconfig subdirectory of the EAR file's META-INF directory. Expanding this directory reveals the well-known directory structure for a cell configuration, as seen in Figure 14-25. You can also see the scope level where each resource is configured.

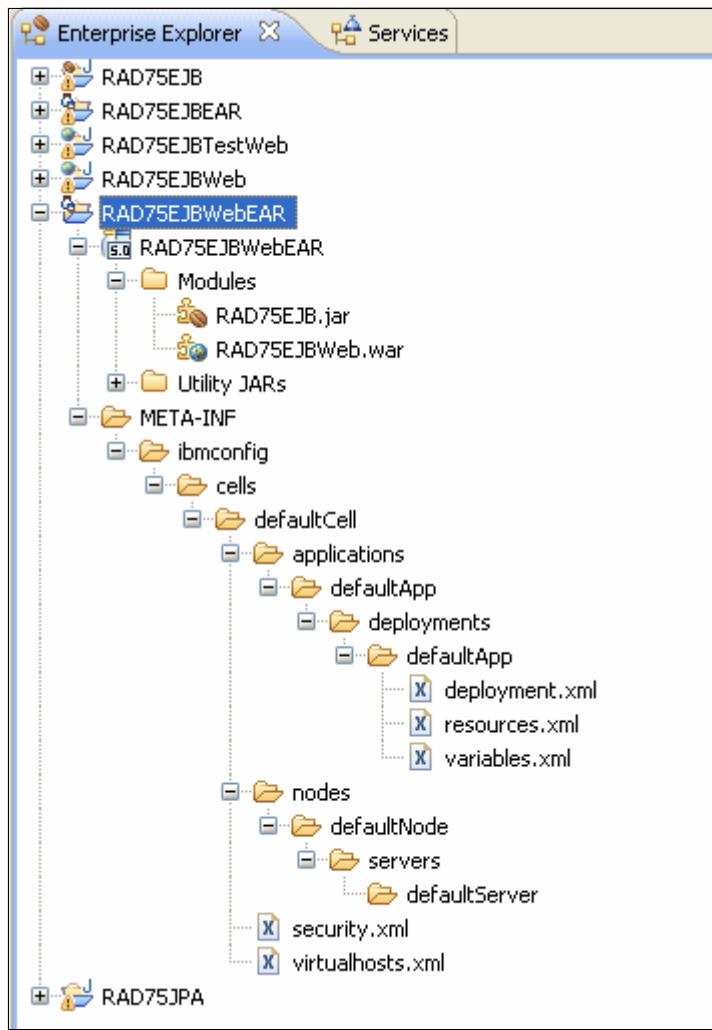


Figure 14-25 Enhanced EAR file contents

At deployment time, WebSphere Application Server uses this information to automatically create the resources.

After you have added the additional configuration information to the application you should now export the project as an EAR file. Refer to “Exporting to an EAR file” on page 730 for information about how to do this.

14.9 Packaging recommendations

Here are some basic rules to consider when packaging an enterprise application:

- ▶ EJB JAR modules and Web WAR modules comprising an application should be packaged together in the same EAR module, and executed within the same application server JVM process. This is to avoid remote EJB calls (RMI/IOP) across application server JVM processes, which is costly from a performance perspective.
- ▶ Utility classes used by a single Web module should only be placed within its WEB-INF/lib folder.
- ▶ Utility classes used by multiple modules within an enterprise application should be placed at the root of the EAR file as Utility Projects, so they are accessible both by servlets and EJBs.
- ▶ Utility classes used by multiple enterprise applications can be placed outside the applications on a directory referenced through a shared library definition.

See Chapter 13, “Understanding class loaders” on page 677 for more details on how WebSphere finds and loads classes.

14.10 Business-level applications

A business-level application is a concept which aims to expand the notion of “Application” beyond JEE. Its administration model provides the entire definition of an application as it makes sense to the business. In contrast with an enterprise application (EAR file) a business-level application is only a logical WebSphere configuration artifact, similar to a server or cluster, that is stored in the configuration repository.

Figure 14-26 shows the concept of business-level applications.

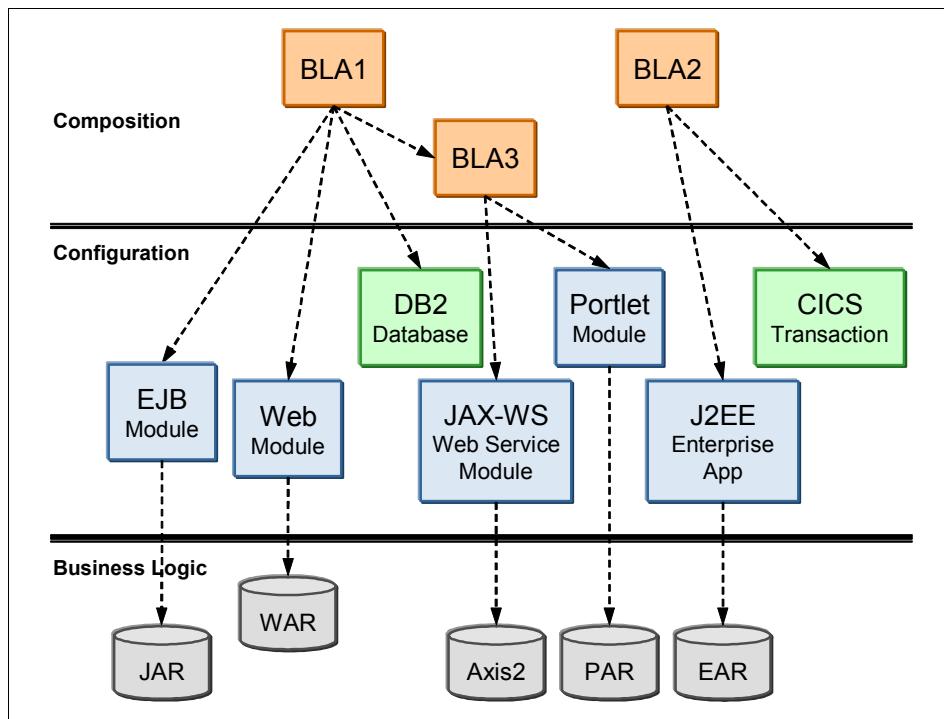


Figure 14-26 Business-level application concept

Business-level applications can be used in several different ways. Often a business application, such as an Order System, does not consist of only one enterprise application (EAR), but rather multiple applications that must all be running for the whole business application to work.

One way of using business-level applications is to group the separate enterprise applications which make up the business application into one manageable unit that can be started, stopped, updated, and so on. But a business-level application cannot only reference JEE components, but also assets that are not part of the JEE concept. For example, CORBA (C++) executables hosted in a Generic Server or files on the file system that are not managed by WebSphere but still required by the application.

A business-level application does not represent or contain application binary files, however. Instead it is a configuration that lists one or more composition units that represent the application binary files. A business-level application uses the binary files to run the application business logic. Administration of binary files is done using the normal methods for managing modules (for example, Web or EJB modules) and is separate from administration of the application definition.

A business-level application does not introduce any new programming, runtime, or packaging models:

- ▶ You do not need to change your application business logic. The business-level application function does not introduce new application programming interfaces (APIs).
- ▶ You do not need to change your application runtime settings. WebSphere supports all of the runtime characteristics, such as security, class loading and isolation, required by individual programming models to which business components are written.
- ▶ You do not need to change your application packaging. There is no specific unique packaging model that provides a business-level application definition.

Note: Business-level applications are only supported on WebSphere Application Server v7.0 nodes. They are not supported on any previous versions of WebSphere Application Server.

The terminology for business-level applications introduces two new terms; *assets* and *composition units*.

An *asset* represents one or more application binary files that are stored in an asset repository. Typical assets include application business logic such as EAR files, EJB modules, Web modules, Service Component Architecture (SCA) modules, shared library files, static content, and other resource files. The asset repository is managed by WebSphere Application Server and does not require any 3rd party software.

You must register files as assets before you can add them to one or more business-level applications. At the time of asset registration, you can import the physical application files into WebSphere's configuration repository or you can specify an external location where the asset resides.

A *composition unit* represents a configured asset in a business-level application. Configured in this context means installed, so a configured Web module means a Web module which is installed.

WebSphere Application Server handles three types of composition units:

- ▶ Asset composition units:
Composition units created from assets by configuring each deployable unit of the asset to run on deployment targets.

- ▶ Shared library composition units:
Composition units created from JAR-based assets by ignoring all the deployable objects from the asset and treating the asset JAR file as a library of classes.
- ▶ Business-level application composition units:
Composition units created from business-level applications that are added to existing business-level applications.

Figure 14-27 shows the relationship between assets, composition units, and business-level applications.

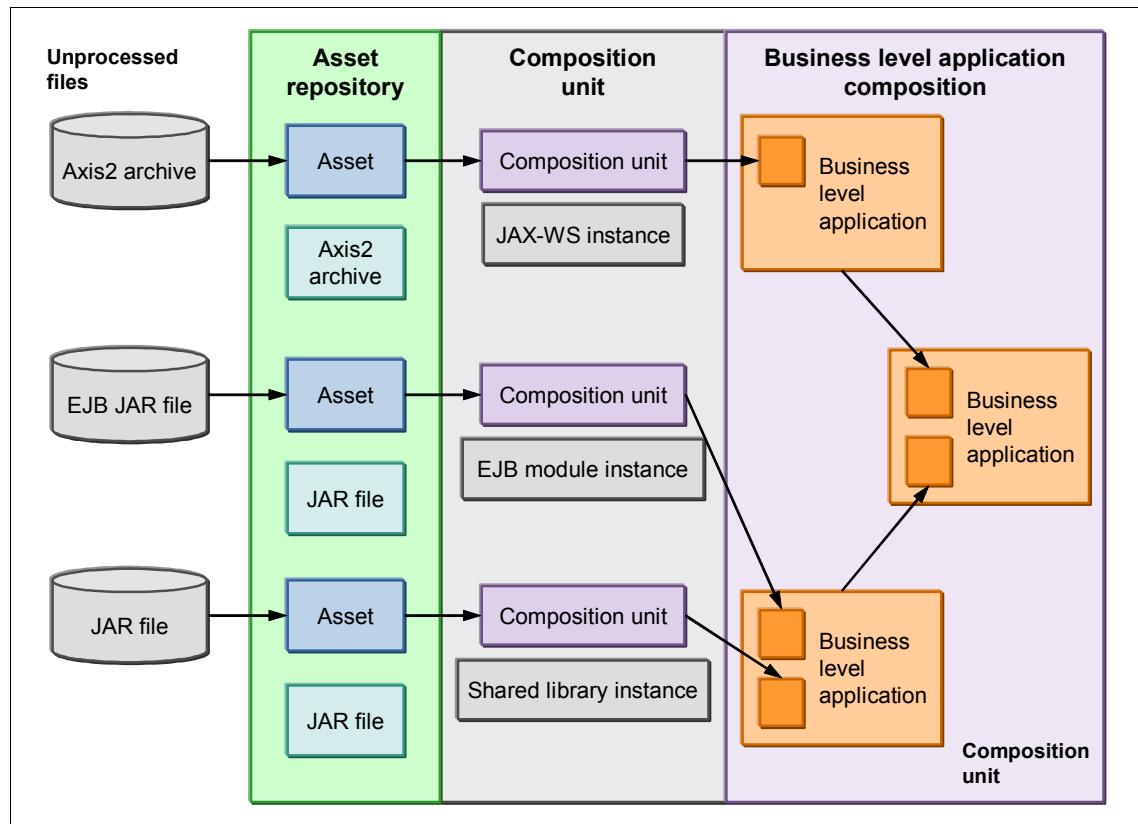


Figure 14-27 Relationship between business-level application artifacts

If an asset depends on another asset, a relationship can be created between them. For example, this would allow a Web module asset to reference classes in a shared library.

So, to summarize, a business-level application consists of composition units. When you add an asset to a business-level application, a composition unit is created for the asset. The composition unit is the configured (installed) asset.

14.10.1 Example: Creating a business-level application

As a basic example of how to create a business-level application, we use the RAD75EJBWeb application (see “Example: Packaging an application” on page 725) and combine it with the WebSphere DefaultApplication’s Web module (which includes, for example, the SnoopServlet) which can be extracted from the DefaultApplication’s EAR file. We create one asset for the RAD75EJBWeb application and another for the DefaultWebApplication. We then create a business-level application containing these two assets.

To create the two assets, do the following steps:

1. Open the WebSphere administrative console and select **Applications** → **Application Types**. Click the **Assets** link.
2. Click the **Import** button. Check the **Local file system** box and click the **Browse** button to locate the RAD75EJBWebEAR.ear file. Select the file and click **Open**. Then click the **Next** button.
3. Step1: Select options for importing an asset:

Enter a brief description of the asset, if wanted. If you want to import the asset to a specific path on your file system enter the path in the Asset binaries destination URL field. If you leave this field blank, the file will be imported to its default location, which is *profile_root/installedAssets/asset_name/BASE/*. Click **Next**.

Tip: If specifying another name for the asset, you must make sure to keep the asset’s file extension (such as .ear or .war), otherwise WebSphere cannot keep track of the asset type, and fails to import the asset.

4. On the Summary page, click **Finish**. The asset is now imported into WebSphere’s asset repository, but its not yet configured (so it is still just an asset, not a composition unit).
5. Repeat the process and import the DefaultWebApplication.war file. Because the two assets do not depend on each other and do not require any shared library, you do not need to set up any relationships.
6. **Save to master configuration** when done.

When the assets have been imported into the asset repository, we can now create a business-level application.

7. Select **Applications** → **Application Types** → **Business-level applications**. Click the **New** button.
8. Enter a name, such as ITSOBank System, and click **Apply**.
9. On the Business-level applications page, click the **Add** button under the Deployed assets section and select the **Add Asset** option, as shown in Figure 14-28.

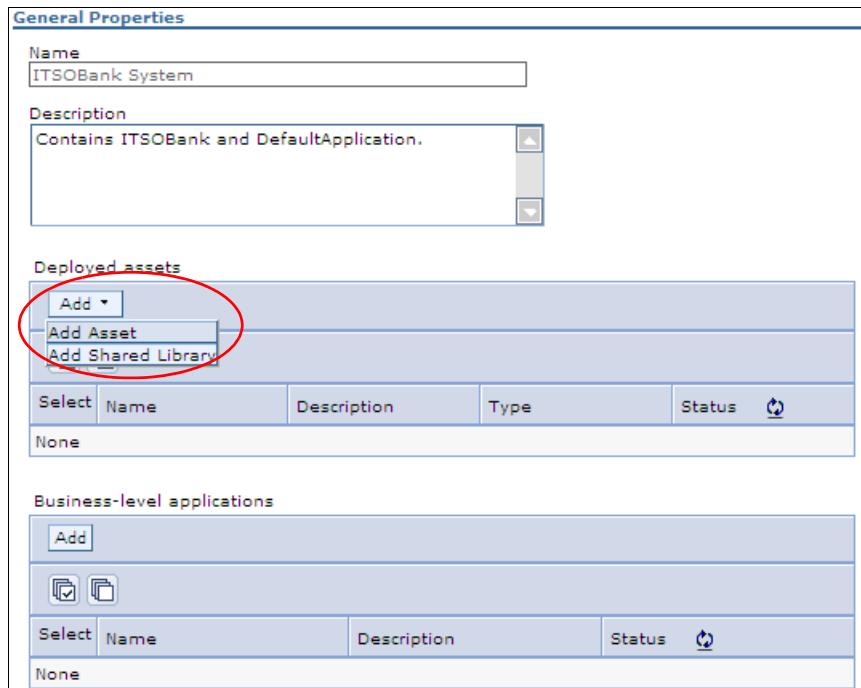


Figure 14-28 Business-level application configuration panel

10. On the next panel, select the **RAD75EJBWebEAR.ear** asset and click **Continue**.
11. You can now configure (install) this asset with the necessary deployment options. The installation process follows the normal steps for installing an application to WebSphere Application Server.
For detailed step-by-step instructions on how to do this, refer to 15.2, “Deploying the application” on page 772. Proceed through the installation panels until the Summary page is shown, and click **Finish**. WebSphere now installs the application, and it is now a composition unit, an asset that has been configured.

Note: At Step 1: Select installation options, WebSphere generates a unique application name such as app1143018803114601914 for the application. You might want to change this to something more descriptive.

12. Select **Applications** → **Application Types** → **Business-level applications** and click the link for the ITSOBank System application.
13. Repeat steps 9 and 10 and add the DefaultWebApplication.war file to the ITSOBank System business-level application as well.
14. **Save** to the master configuration when done.

The business-level application can now be started and stopped by selecting **Applications** → **Application Types** → **Business-level applications** and then clicking the corresponding links.

Note: To delete a business-level application, you must first unmap (delete) the composition units (configured assets) which belong to the business-level application. To do this, select the business-level application, and check the boxes next to the deployed assets and click the **Delete** button. When all assets have been deleted from the business-level application, you can delete the business-level application itself. The assets, however, still remain in WebSphere's asset repository and can be used to configure other business-level applications.

The individual applications, the composition units, which make up the business-level application can be managed individually by selecting **Applications** → **Application Types** → **WebSphere enterprise applications**. and using the links to start, stop, update, and so on.



Deploying applications

In 14.6, “Example: Packaging an application” on page 725, we discuss how to use the Rational Application Developer Assembly and Deploy Features for WebSphere 7.0 (RAD-AD) to perform common tasks for packaging an application.

In this chapter, we show you how to deploy the application. We explain how to set up the environment for the application and then deploy the application itself. Next, we explain how to deploy the client part of the application. You can also automate the deployment tasks in this chapter using command-line tools, as explained in Chapter 8, “Administration with scripting” on page 439.

We cover the following topics:

- ▶ “Preparing the environment” on page 755
- ▶ “Deploying the application” on page 772
- ▶ “Deploying application clients” on page 778
- ▶ “Updating applications” on page 783

Note: The application that we prepared is the ITSO Bank application, which was developed by the team who wrote the *Rational Application Developer V7.5 Programming Guide* Redbooks publication. The application can be downloaded from the Additional Materials section in that book, available at:

<http://www.redbooks.ibm.com/redpieces/abstracts/sg247672.html?Open>

The Additional Materials section contains two ZIP files. To prepare for this chapter, download the 7672code.zip file and unpack it to a directory on your computer. The database directory in this ZIP file contains scripts that we will use to prepare the database for the application.

If you are working on a pre-JEE 5 application or are using EJB 2.1, or earlier, modules, also refer to Chapter 14 in the WebSphere Application Server v6.1 Redbooks publication for specific details on earlier versions, available at:

<http://www.redbooks.ibm.com/abstracts/sg247304.html?Open>

15.1 Preparing the environment

In this chapter, we show you how to set up a fairly complete environment for the ITSO Bank application and deploy the EAR file. You will not always need or want to customize the environment as extensively as we do in this chapter. Some steps are optional. If all you want to do is deploy your application quickly, using the WebSphere defaults for directory names, log files, and so forth, skip to 15.2, “Deploying the application” on page 772.

The steps in this section are performed typically by the application deployer. To deploy the ITSO Bank application, do the following steps:

1. Create the DB2 database for ITSO Bank. This step is required.
2. Create an environment variable for ITSO Bank server. This step is optional.
3. Create an application server to host the application. This step is optional.
4. Customize the IBM HTTP Server configuration. This step is optional.
5. Define a JDBC provider, data source, and authentication alias. This step is required if you are not using an Enhanced EAR.
6. Define virtual hosts. This step is optional and not required if you are using an Enhanced EAR.

If the application to be deployed is a WebSphere Enhanced EAR file, the resources configured in the Enhanced EAR file are created automatically when the application is deployed.

15.1.1 Creating the ITSO Bank DB2 database

To set up the DB2 database, make sure you have DB2 installed and running. Then run the following commands:

- ▶ Open a command prompt.
- ▶ Change directory to the database\db2 folder in the 7672code folder created when unzipping the additional material.
- ▶ Execute the **createbank.bat** file to define the database and table.
- ▶ Execute the **loadbank.bat** file to delete the existing data and add records.
- ▶ Execute the **listbank.bat** file to list the contents of the database.

Each command opens a new window where the DB2 script executes. Each command also leaves a connection to the database open, so you might want to execute a **db2 connect reset** command in each window opened to disconnect from the database so no unused connections are kept open.

15.1.2 Creating an environment variable

We recommend that you use WebSphere environment variables, rather than hard-coded paths when deploying an application. In the following sections, we assume that you have declared an ITSOBANK_ROOT variable. You will use it when specifying, for example, the JVM log's location.

Be certain you declare this variable at the right scope. For example, if you define this variable at the application server scope, it will only be known at that level. As long as you work with the WebSphere Application Server Base or Express editions, this is fine. But if you later decide to use the Network Deployment edition and you create a cluster of application servers, the ITSOBANK_ROOT variable will need to be defined at the cluster or cell level.

Use the steps in 5.1.10, “Using variables” on page 263 to create a ITSOBANK_ROOT variable with a value of C:\apps\ITSOBANK.

There are several ways to organize WebSphere applications. Some companies prefer to create a directory for each application, as we do in our example, such as C:\apps*application_name*, and keep all resources and directories required by the application in subdirectories under this directory. This strategy works well when deploying only one application per application server, again as we do in our example, because the application server's log files could then all be changed to point to c:\apps*application_name*\logs.

Other companies prefer to organize resources by resource type, and so create directories such as c:\apps\logs*application_name.log*, c:\apps\properties*application_name.properties*, and so on.

And some companies prefer to stick with the vendor defaults as far as possible. For WebSphere, that means that the applications are installed in the *profile_root*/installedApps directory and the logs files are written to the *profile_root*/logs/*server_name* directory.

Which option you choose is a matter of personal preferences and corporate guidelines.

Note: Make sure you create the target directory you specify for the ITSOBANK_ROOT variable before proceeding. If the directory is not created, the application server will not start.

15.1.3 Creating the ITSO Bank application server

In a distributed server environment, you have the option of using a single application server, or creating multiple application servers or clusters.

The advantage of deploying multiple applications to a single application server is that it consumes less resources. There is no overhead for any extra application server processes. Another benefit is that applications can make in-process calls to each other. For example, servlets in one EAR file could access Local interfaces of EJBs in another EAR file.

One alternative to using a single application server is to deploy each application to its own server. The advantages of deploying only one application on an application server is that it gives you greater control over the environment. The JVM heap sizes and environment variables are set at the application server level. Thus, all applications running in an application server share the JVM memory given to the application server and all see the same environment variables. Running each application in its own application server could also make it easier to perform problem determination. For example, if an application runs amok and consumes a lot of CPU, you can determine which application it is by looking at the process ID of the application server.

In our example, we create a unique application server on which to run the ITSO Bank sample application.

Note: For a discussion of application server properties, see 6.4, “Working with application servers” on page 297.

To create an application server, do the following steps:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. Click the **New** button and provide the information node and server name, as shown in Figure 15-1.

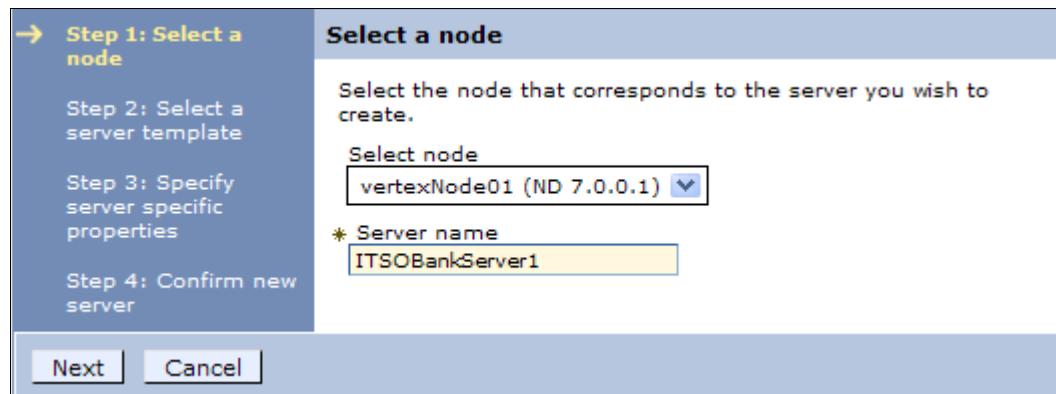


Figure 15-1 Creating the ITSO Bank application server

As you can see in the figure, the application server name will be **ITSOBankServer1**.

Click **Next**.

3. In Step 2, select which server template to use as the base for this new application server.

The DeveloperServer template is used when setting up a server for development use and will cause the JVM to prioritize for a quick start-up (by disabling bytecode verification, and performing JIT compilations with a lower optimization level). This option should not be used on a production server, where long run throughput is more important than early server startup.

If you have not created any templates on your own, then select the WebSphere **default**. Otherwise, select the server template you want to use and click **Next**.

4. In step 3, you can select to have WebSphere generate a unique set of port numbers for this application server. This ensures the ports defined for this server do not conflict with another server currently configured on this node. Check the **Generate Unique Http Ports** box and click **Next**.
5. On the Summary page, click **Finish**.

Changing the working directory

The next thing we want to do is to change the working directory for the application server process. This directory is the relative root for searching files. For example, if you do a `File.open("foo.gif")`, `foo.gif` must be present in the working directory. This directory will be created by WebSphere if it does not exist. We recommend that you create a specific working directory for each application server.

1. Select the server, **ITSOBankServer1**, you just created.
2. Expand the **Java and Process Management** in the Server Infrastructure section and select **Process Definition**.
3. Scroll down the page and change the working directory from `${USER_INSTALL_ROOT}` to `${ITSOBANK_ROOT}/workingDir`.
4. Click **OK**.

Note: The working directory will not be created automatically if you use a composed path, such as `C:/apps/ITSOBANK/workingDir`. If you want to use such a path, create it before starting the application server, or the startup sequence fails.

Changing the logging and tracing options

Next, we want to customize the logging and tracing properties for the new application server. There are several ways to access the logging and tracing properties for an application server.

- ▶ Select **Troubleshooting** → **Logs and Trace** in the navigation bar, then select a server.
- ▶ Select **Servers** → **Server Types** → **WebSphere application servers**, select a server, and then select **Logging and Tracing** from the Troubleshooting section.
- ▶ Select **Servers** → **Server Types** → **WebSphere application servers**, select a server, select **Process definition** from the Java and Process Management section. Select **Logging and Tracing** from the Additional Properties section.

Because we have just finished updating the application server process definition, we will take the third navigation path to customize the location of the JVM logs, the diagnostic trace logs, and the process logs.

1. Select **Logging and Tracing**.
2. Select **JVM Logs**.

This allows you to change the JVM standard output and error file properties. Both are rotating files. You can choose to save the current file and create a new one, either when it reaches a certain size, or at a specific moment during the day. You can also choose to disable the output of calls to `System.out.print()` or `System.err.print()`.

We recommend that you specify a new file name, using an environment variable to specify it, such as:

```
 ${ITSOBANK_ROOT}/logs/SystemOut.log  
 ${ITSOBANK_ROOT}/logs/SystemErr.log
```

On this page you can also modify how WebSphere will rotate your log files.

Click **OK**.

3. Select **Diagnostic Trace**.

Each component of the WebSphere Application Server is enabled for tracing with the JRs interface. This trace can be changed dynamically while the process is running using the Runtime tab, or added to the application server definition from the Configuration tab. As shown in Figure 15-2, the trace output can be either directed to memory or to a rotating trace file.

Change the trace output file name so the trace is stored in a specific location for the server using the `ITSOBANK_ROOT` variable and select the **Log Analyzer** format.

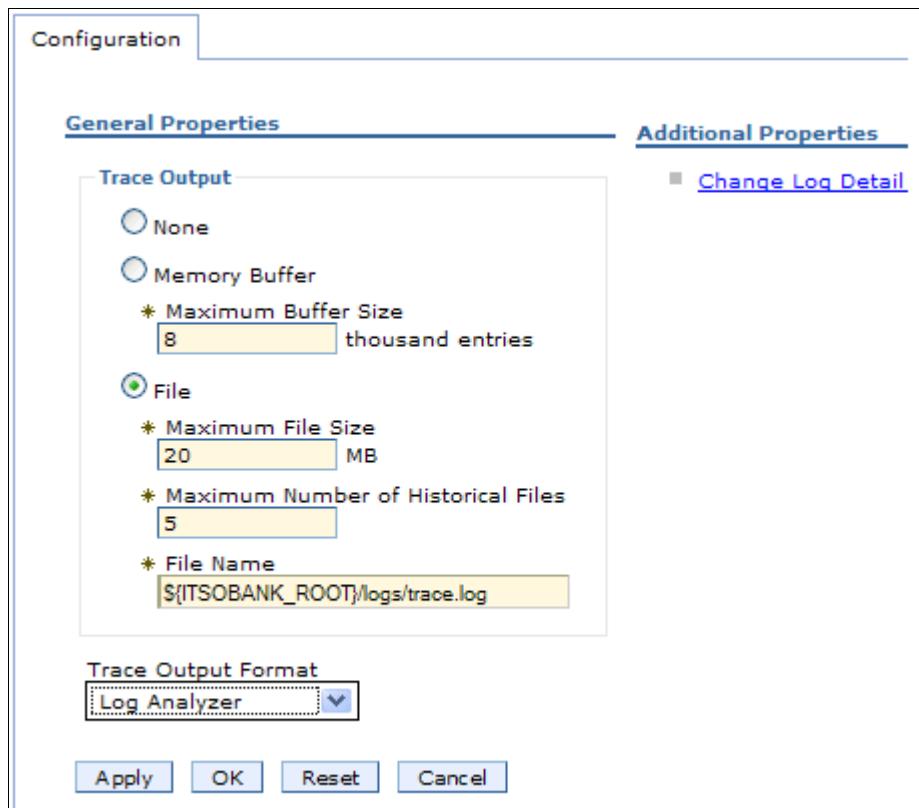


Figure 15-2 Specifying diagnostic trace service options

Click **OK**.

4. Select Process Logs.

Messages written by native code (JNI) to standard out and standard error streams are redirected by WebSphere to process logs, usually called native_stdout.log and native_stderr.log. Change the native process logs to:

```
 ${ITSOBANK_ROOT}/logs/native_stdout.log  
 ${ITSOBANK_ROOT}/logs/native_stderr.log
```

Click **OK**.

5. All log files produced by the application server are now redirected to the \${ITSOBANK_ROOT}/logs directory. Save the configuration.

Note: The rest of this example assumes a default HTTP port of 9080 for the Web container. Before proceeding, check the application server you created to determine the port you should use:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. Select the **ITSOBankServer1**.
3. Select **Ports** in the Communications section.
4. Scroll down the page and note the port listed for WC_defaulthost.

15.1.4 Defining the ITSO Bank virtual host

Enhanced EAR file users: If you are using an Enhanced EAR file, the virtual host can be defined at packaging time. See “Adding a virtual host” on page 741.

Web modules need to be bound to a specific virtual host. For our sample, we chose to bind the RAD75EJBWeb Web module to a specific virtual host called itsobank_host. This virtual host has the following host aliases:

- ▶ www.itsobank.ibm.com:80
- ▶ www.itsobank.ibm.com:9080

Any request starting with *itsobank_host_alias/RAD75EJBWeb*, such as <http://www.itsobank.ibm.com:9080/RAD75EJBWeb>, is served by the RAD75EJBWeb application.

Tip: You can restrict the list of hosts used to access the Web application by removing hosts from the virtual host definition.

Imagine you want to prevent users from directly accessing the ITSO Bank application from the WebSphere internal HTTP server when they invoke <http://www.itsobank.ibm.com:9080/RAD75EJBWeb>. In other words, you want to force all requests to go through the Web server plug-in. You can achieve this by removing www.itsobank.ibm.com:9080 from the virtual host aliases list.

To create the itsobank_host virtual host, do the following steps:

1. Select the **Environment** → **Virtual Hosts** entry in the navigation pane.
2. Click **New**.
3. Enter the virtual host name, itsobank_host.
4. Click **Apply**.
5. Select **Host Aliases** in the Additional Properties section.

6. Add the two aliases shown in Figure 15-3 by clicking **New**, entering the values, and clicking **OK**.



Figure 15-3 WebSphere Bank virtual host aliases

7. Click **OK**.
8. Save the configuration.

15.1.5 Creating the virtual host for IBM HTTP Server and Apache

Now that we have defined a itsobank_host virtual host, we need to configure the Web server to serve the host aliases in the virtual host. The steps below are valid for both the IBM HTTP Server V7 and Apache 2.x.

Configuring virtual hosting

Note: It is not necessary to create a virtual host in httpd.conf. It is required only if you want to customize the configuration, for example, by separating the logs for each virtual host. This is not normally done.

Creating virtual hosts is done using the `VirtualHost` directive, as in Example 15-1.

Example 15-1 Using VirtualHost

```
<VirtualHost www.itsobank.ibm.com:80>
    ServerAdmin webmaster@itsobank.ibm.com
    ServerName www.itsobank.ibm.com
    DocumentRoot "C:\IBM\HTTPServer\htdocs\itsobank"
    ErrorLog logs/itsobank_error.log
    TransferLog logs/itsobank_access.log
</VirtualHost>
```

If you want to have multiple virtual hosts for the same IP address, you must use the NameVirtualHost directive. See Example 15-2.

Example 15-2 Using the NameVirtualHost and VirtualHost directives

NameVirtualHost 9.11.12.13:80

```
<VirtualHost itso_server:80>
    ServerAdmin webmaster@itso_server.com
    ServerName itso_server
    DocumentRoot "C:\IBM\HTTPServer\htdocs\itso_server"
    ErrorLog logs/itso_server_error.log
    TransferLog logs/itso_server_access.log
</VirtualHost>

<VirtualHost www.itsobank.ibm.com:80>
    ServerAdmin webmaster@itsobank.ibm.com
    ServerName www.itsobank.ibm.com
    DocumentRoot "C:\IBM\HTTPServer\htdocs\itsobank"
    ErrorLog logs/itsobank_error.log
    TransferLog logs/itsobank_access.log
</VirtualHost>
```

The www.itsobank.ibm.com and the itso_server hosts have the same IP address, 9.11.12.13. We have set this by inserting the following line in the machine hosts file, located in %windir%\system32\drivers\etc or in /etc on UNIX systems):

```
9.11.12.13 www.itsobank.ibm.com itso_server
```

In a real-life environment, this would probably be achieved by creating aliases at the DNS level. In any event, you must be able to ping the host you have defined, using commands such as **ping www.itsobank.ibm.com**.

As you can see in Example 15-2, each virtual host has a different document root. Make sure that the directory you specify exists before you start the HTTP server. While testing the setup, you can place an index.html file at the document root stating which virtual host is being called. This lets you easily see which virtual host is being used.

You must restart the IBM HTTP Server to apply these changes. If you are running a Windows system, we recommend that you try to start the server by running **apache.exe** from the command line rather than from the Services window. This allows you to spot error messages thrown at server startup.

If your virtual hosts are correctly configured, invoking <http://www.itsobank.ibm.com> or http://itso_server returns different HTML pages.

15.1.6 Creating a DB2 JDBC provider and data source

Enhanced EAR file users: If you are using an Enhanced EAR file, the JDBC provider, data source, and J2C authentication entry can be defined at packaging time. See “Configuring a DB2 JDBC provider” on page 735.

The ITSO Bank sample application uses a relational database, via Java Persistence API, to store information. To access this database, a data source needs to be defined with a JNDI name that matches the data source configuration in the JPA module's persistence.xml file. The ITSO Bank sample application is configured for Derby by default. In Chapter 14, “Packaging applications for deployment” on page 709, however, we modified the ITSO Bank application to run against a DB2 database instead. We will now create the DB2 JDBC provider, data source, and JAAS authentication alias required to run against DB2.

For detailed information about JDBC providers and data sources, refer to Chapter 9, “Accessing databases from WebSphere” on page 487.

Configuring environment variables for DB2 JDBC driver

For the DB2 JCC JDBC Provider to find its classes, the DB2_JCC_DRIVER_PATH and DB2_JCC_DRIVER_NATIVEPATH environment variables must be set up. To set up these variables, do the following steps:

1. Select **Environment** → **WebSphere Variables**.
2. Locate and click the **DB2_JCC_DRIVER_PATH** entry.
3. In the value field, enter the path to where the DB2 JDBC driver is located. For example, for DB2, the location is likely to be:

C:\Program Files\IBM\SQLLIB\java

See Figure 15-4.

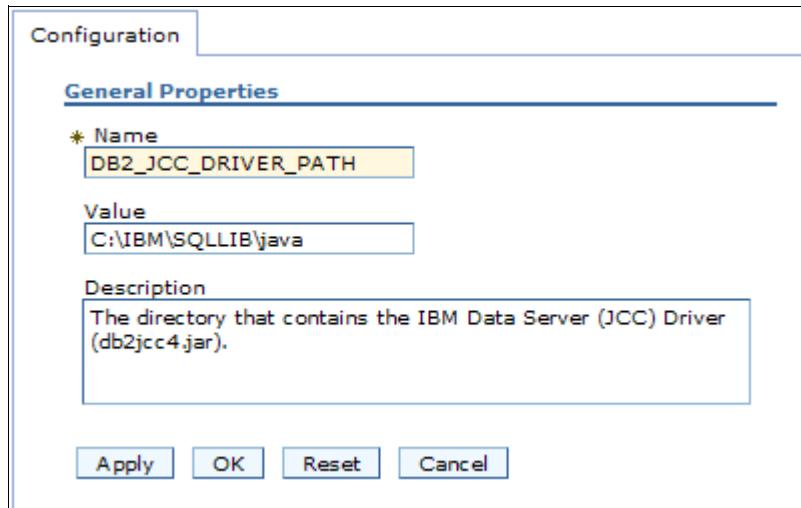


Figure 15-4 Configuring DB2 Driver Path

Click **OK**.

4. Repeat the process for the DB2_JCC_DRIVER_NATIVEPATH variable. For DB2, it should use the same path, C:\Program Files\IBM\SQLLIB\java.

Configuring J2C authentication data

The user ID and password required to access the database are specified in a J2C authentication data entry.

1. Select **Security** → **Global Security**. Expand the Java Authentication and Authorization Service section under the Authentication section and select **J2C authentication data**.

2. Click **New**, and specify the following information to create the authentication data. Once completed, the authentication information should be similar to Figure 15-5.



Figure 15-5 Creating ITSO Bank JAAS authentication alias

3. Click **OK**.

Creating the ITSO Bank JDBC provider

The following steps take you through the creation of a JDBC provider targeting a DB2 database. To create a JDBC provider from the administrative console, do the following steps:

1. Expand the **Resources** entry and then the **JDBC** entry. Then select the **JDBC Providers** entry.
2. Select the scope of this resource. In a stand-alone server environment, it is sufficient to create the data source at the server level. Otherwise, define it at the cluster or cell level. A rationale for this is to be able to share the definition across multiple servers in a cluster. To change this, select the server you are deploying to in the scopes list.
3. Click the **New** button.
4. In the Configuration dialog box, select the general properties for the JDBC provider, as shown in Figure 15-6.

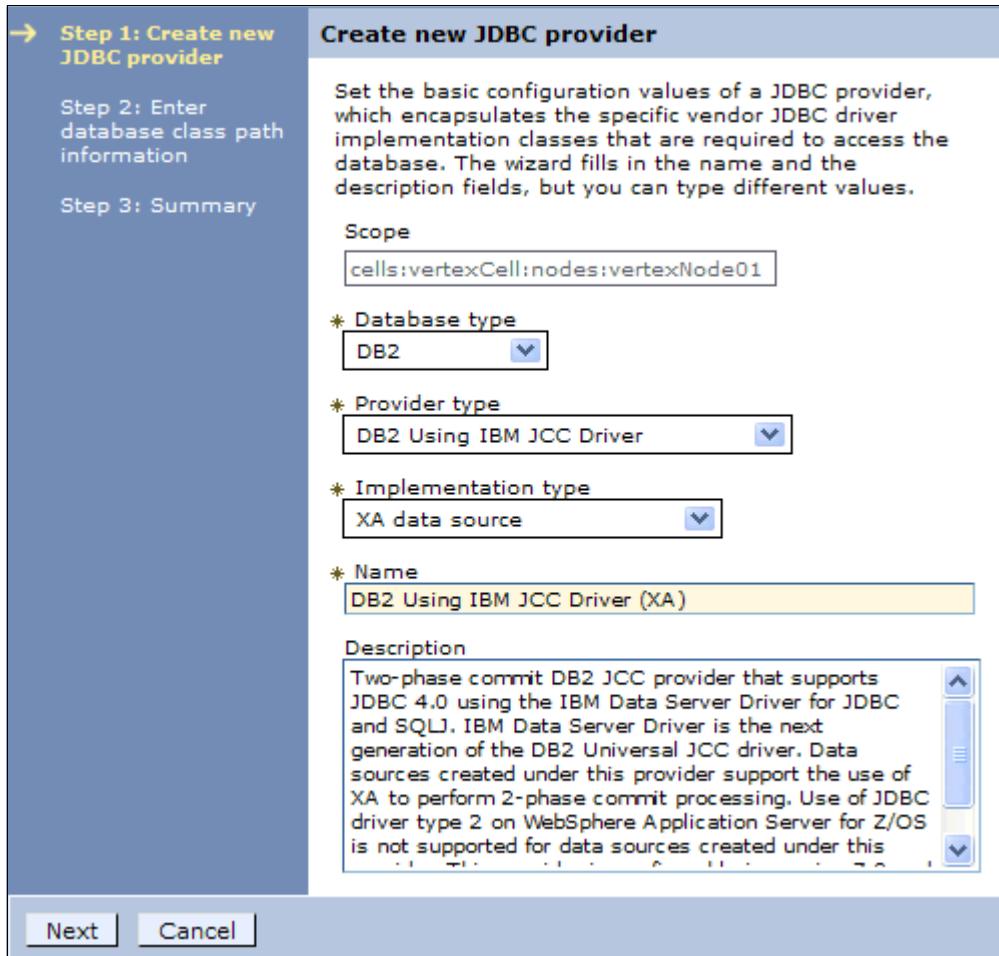


Figure 15-6 Creating a DB2 JDBC provider

- Database type: DB2
- Provider type: DB2 Using IBM JCC Driver
- Implementation type: XA data source
- Name: DB2 Using IBM JCC Driver (XA)

Click **Next**.

Note: We used the DB2 XA-capable JDBC Driver for the ITSO Bank sample. If your application does not require two-phase commit capabilities, use the regular driver. If using an XA-capable driver, it is a best practice to indicate that it is an XA-capable driver by including XA in its name, such as MyJDBCXA.

5. The next window allows you to change the location for the JDBC driver files, but because we configured the paths earlier, we do not need to do it again. Click **Next**.
6. On the Summary page, click **Finish**.

Creating the ITSO Bank data source

The next step is to create the data source for the ITSO Bank DB2 database. To create a data source, do the following steps:

1. Select **Resources** → **JDBC** → **JDBC Providers**.
2. Select the **DB2 Using IBM JCC Driver (XA)** and select **Data Sources** under Additional Properties.
3. Click **New** to add the new data source. See Figure 15-7.

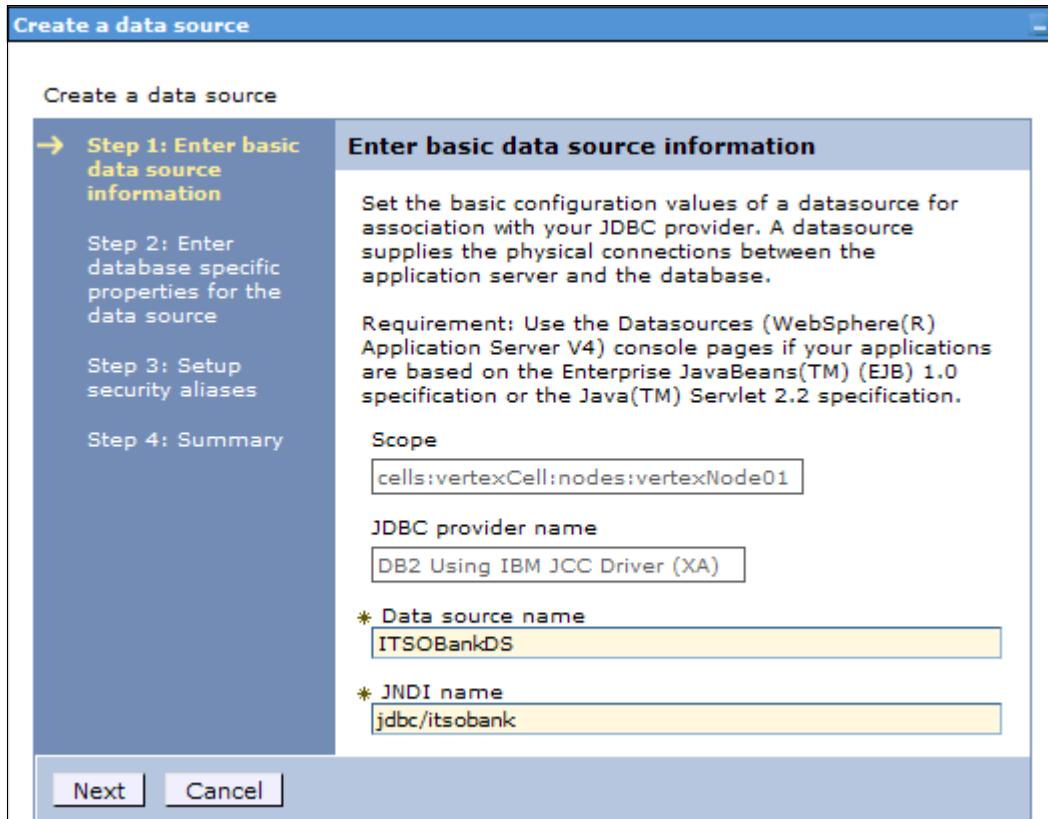


Figure 15-7 ITSO Bank basic data source properties

– Data source name

Enter the data source name, which must be unique in the administrative domain or cell. We recommend that you use a value indicating the name of the database this data source is targeting, such as "ITSOBankDS".

– JNDI name

Enter the name by which applications access this data source. If not specified, the JNDI name defaults to the data source name prefixed with jdbc/. For the ITSO Bank, set this field to jdbc/itsobank. This value can be changed at any time after the data source has been created.

4. Click **Next**. On the second page, enter the information shown in Figure 15-8.

Step 1: Enter basic data source information

→ Step 2: Enter database specific properties for the data source

Step 3: Setup security aliases

Step 4: Summary

Enter database specific properties for the data source

Set these database-specific properties, which are required by the database vendor JDBC driver to support the connections that are managed through the datasource.

Name	Value
* Driver type	2
* Database name	ITSOBANK
* Server name	
* Port number	50000

Use this data source in container managed persistence (CMP)

Previous **Next** **Cancel**

Figure 15-8 ITSO Bank data source database properties

- Driver type

Select the driver type to use.

If the database is on the same machine as the WebSphere installation you can use a type 2 driver and do not need to enter a server name. If the database is on a remote machine you use either a type 4 driver, which allows WebSphere to connect remotely to the database over TCP/IP, or a type 2 driver.

To use a type 2 driver with a remote database you need a local DB2 Client installation on the WebSphere machine and catalog the database on the DB2 Client. WebSphere then sees the database as local and the DB2 Client handles the remote calls. In our setup the database is on the same machine as the WebSphere installation so we choose a type 2 driver and do not enter a Server name.

- Database name

Enter the name of the database, ITSOBANK in our example.

- Port number

Our DB2 installation uses port 50000 so we keep the default value.

- Use this Data Source in container-managed persistence (CMP)

Because the ITSO Bank uses Java Persistence API for database persistence instead of CMP EJBs the data source does not need to be set up for CMP EJBs. So uncheck this option.

5. Click **Next**. On the third page, enter the information shown in Figure 15-8.



Figure 15-9 ITSO Bank database security alias properties

- Container-managed authentication alias

This is the preferred method for specifying authentication information for the database. Enter the J2C alias used for connecting to the data source by selecting the authentication alias created previously, *cell name/itsobank*.

6. Click **Next**, and then on the summary page, click **OK**.
7. Save the configuration.
8. Test the connection by selecting the data source and clicking the **Test Connection** button.

15.2 Deploying the application

In this section, we show the steps required to deploy the application to WebSphere Application Server. The EAR file we deploy is the RAD75EJBWebEAR file, which is a regular EAR file and not an Enhanced EAR file.

Follow these steps to deploy the application:

1. Select **Applications** → **New Application** from the administrative console navigation bar and click the **New Enterprise Application** on the panel shown.

Note: In 14.10, “Business-level applications” on page 745, we created a business level application and added this application to it. As a result, the application was installed. With the exception of this first step, the rest of this process reflects the steps you will see regardless of whether you are installing the application as the result of adding as an asset to a BLA, or are installing the application and creating a BLA as part of the process.

2. Check the **Local file system** box and click the **Browse** button to locate the RAD75EJBWebEAR.ear file. Select the file and click **Open**.

From the install windows, you can install files that are located either on the same machine as the browser you are using to access the WebSphere administrative console, the local file system option, or on the WebSphere Application Server itself, the remote file system option. If you select the Local file system option, the administrative console automatically uploads the file you select to the application server, or to the deployment manager if this is a distributed server environment. If you select the Remote file system check box, you can browse all the nodes in the cell to find the file. The file is then, if necessary, uploaded to the application server or deployment manager.

When you have made your selection, click the **Next** button.

3. WebSphere Application Server allows you to take a shortcut when installing an application. If you select the **Fast Path - Prompt only when additional information is required** option, only the windows where you actually need to fill out some information during installation are shown.

For this example, however, we will explain the options, so select **Detailed - Show all installation options and parameters**.

If you expand the **Choose to generate default bindings and mapping** you can alter the bindings for the application you are deploying. If you select the **Generate Default Bindings** option, WebSphere Application Server completes any incomplete bindings in the application with default values, but

it does not alter any existing bindings. Checking the **Override existing bindings** allows you to specify a bindings file which contains new bindings.

The contents of the application or module that you are installing determines which options are displayed on the bindings page. For our ITSO Bank application the options documented in Table 15-1 are displayed.

Because our application uses JEE 5 and EJB 3.0 and relies on the bindings and mappings generated automatically by the EJB container, there is no need to override our bindings. We leave all check boxes cleared.

Table 15-1 Application default bindings

Binding name	Detailed information
Specific bindings file	You can create a specific bindings file using your favorite editor and load it during application installation by clicking Browse next to the specific bindings file.
Unique prefix for beans	You can generate default EJB JNDI names using a common prefix. EJBs for which you did not specify a JNDI name will get a default name, built by concatenating the prefix and the EJB name. If you specify a prefix of myApp/ejb, then JNDI names default to myApp/ejb/EJBName, such as myApp/ejb/Account.
Virtual host bindings	You can bind all Web modules to a specific virtual host, such as itsobank_host.

Click **Next**.

4. The rest of the wizard is divided into steps. The number of steps depends on your application, for example, if it contains EJB modules or Web modules, you will see windows prompting for the information necessary to deploy them.
5. Step 1: Select installation options.

Step 1 gives you a chance to review the installation options. You can specify various deployment options, such as JSP precompiling, and whether you want to generate EJB deployment code (not applicable for EJB 3.0 beans).

- If you are deploying an Enhanced EAR file, this is where you make the decision whether to use the resource configuration information packaged in the Enhanced EAR file or not. If the EAR file you are installing is an Enhanced EAR, the install window preselects the **Process embedded configuration** check box. If you do not want to use the resource configuration information packaged in the Enhanced EAR file, you must deselect this check box. Because we have already configured the necessary resources needed, we make sure the **Process embedded configuration** check box is not selected.
- Selecting the **Pre-compile JavaServer Pages files** option makes WebSphere compile all JSPs in the EAR file during installation instead of

during runtime. Thus, the first user who accesses the application does not have to wait for the JSPs to compile.

A second alternative to pre-compiling JSPs is to use the `JspBatchCompiler` script found in the `bin` directory of the profile that you are using to compile the JSPs after the application is installed.

- This page also allows you to specify file permissions for files in your application. To use one of the predefined file permissions, select it, and then click **Set file permissions**. You can also specify your own file permissions using regular expressions.
- The administrative console displays the Application Build ID of the application being installed. This string is specified in the `MANIFEST.MF` file in the `EAR` file's `META-INF` folder and can be set using the Rational Application Developer Assembly and Deploy tool.

The following is an example of a version number specified in the `MANIFEST.MF` file:

`Implementation-Version: Version 1.2.3`

- The dispatching and servicing of remote resources are extensions to the Web container that allows frameworks, servlets, and JSPs to include content from outside of the current executing resource's JVM as part of the response sent to the client.

To enable these features, select the corresponding check boxes to allow dispatching or servicing includes to/from remote resources.

- The **Allow EJB reference targets to resolve automatically** option is used for EJB 2.1 or earlier or Web 2.3 or earlier modules and allows WebSphere Application Server to provide a default value or automatically resolve EJB references for any EJB reference that does not have a binding. Because our application is at EJB 3.0 this option does not apply to our application.

Click **Next**.

6. Step 2: Map modules to servers.

Select the server on which you want each module deployed. For better performance, we recommend that you deploy all modules from one application in a single server. Especially, do not separate the EJB clients, usually servlets in Web modules, from the EJBs themselves.

Click the  icon to select all modules in the ITSO Bank `EAR` file. In the Clusters and Servers box, select **ITSOBankServer1**. Then click **Apply**. This assigns all modules to the `ITSOBankServer1` application server. If you deploy to a cluster, select the cluster instead of the single application server.

See Figure 15-10.

Web servers: If you have a Web server defined, select both the Web server and ITSOBankServer1 in the server list. Press and hold the CTRL key to select multiple servers. Mapping Web modules to Web servers ensures the Web server plug-in will be generated properly.

Map modules to servers

Specify targets such as application servers or clusters of application servers where you want to install the modules that are contained in your application. Modules can be installed on the same application server or dispersed among several application servers. Also, specify the Web servers as targets that serve as routers for requests to this application. The plug-in configuration file (plugin-cfg.xml) for each Web server is generated, based on the applications that are routed through.

Clusters and servers:

Select	Module	URI	Server
<input checked="" type="checkbox"/>	RAD75EJB.jar	RAD75EJB.jar,META-INF/ejb-jar.xml	WebSphere:cell=vertexCell,node=vertexNode01,server=ITSOBankServer1
<input checked="" type="checkbox"/>	RAD75EJBTestWeb	RAD75EJBTestWeb.war,WEB-INF/web.xml	WebSphere:cell=vertexCell,node=vertexNode01,server=ITSOBankServer1

Figure 15-10 Mapping modules to application servers

Click **Next**.

7. Step 3: Provide JSP reloading options for Web modules.

This setting allows you to configure if and how often WebSphere should check for updates to JSP files, and if they should be reloaded or not. In a production environment, you might want to disable this to improve performance.

Click **Next**.

8. Step 4 and 5: Map shared libraries, and Map shared library relationships

If your application depends on shared libraries, you can specify them here.
Click **Next**.

9. Step 6: Initialize parameters for servlets.

For servlets that honor initialization parameters (specified by the init-param tag in the Web module's web.xml deployment descriptor) you can configure the value of the parameters.

10. Step 7: Provide JNDI names for beans.

Use this window to bind the enterprise beans in your application or module to a JNDI name. Because our application is at EJB 3.0 level we can leave it blank to have WebSphere Application Server use the default names.

Click **Next**.

11. Step 8: Bind EJB Business interfaces to JNDI names.

This window allows you to specify a JNDI name for the business interfaces of your EJBs. Because our application is at EJB 3.0 level we can leave it blank to have WebSphere Application Server assign default JNDI name.

12. Step 9: Map EJB References to beans.

Each EJB reference defined in your application must be mapped to an enterprise bean. Because our Web module is at Web 2.5 level we can leave it blank to have WebSphere Application Server use the default names. If the reference was in an EJB 2.x or earlier or Web 2.3 or earlier module we could check the **Allow EJB reference targets to resolve automatically** and would then not need to specify a target JNDI name either.

Click **Next**.

13. Step 10: Map virtual hosts for Web modules.

Select the virtual host we created for the application (itsobank_host).

Note: Only those virtual host definitions that are defined to the WebSphere environment are configurable at this step. To map the Web modules to the virtual host that is defined in the Enhanced EAR file, you need to configure that host after deploying the application.

Click **Next**.

14. Step 11: Map context roots for Web modules.

Select the context root to bind the module against.

Click **Next**.

15. Step 12: Metadata for modules.

Checking the metadata-complete attribute tells WebSphere Application Server to ignore any deployment information specified in source code annotations. Leave both check boxes cleared to use the information from the annotations.

Click **Next**.

16. Step 13: Summary.

The Summary window gives an overview of application deployment settings. If those settings are fine, click **Finish** to deploy the application.

17. Save the configuration.

If you are working in a distributed server environment, make sure you synchronize the changes with the nodes so they application is propagated to the target application server (s).

18. If you mapped the Web modules to a Web server, make sure the Web server plug-in is regenerated and propagated to the Web server. For a quick refresh, restart the Web server.

Deployment of the RAD75EJBWebEAR application is now complete. After starting the application you can now verify that the application works by pointing your browser to:

<http://www.itsobank.ibm.com:9080/RAD75EJBWeb>

If successful it should display the web page for the ITSO Bank application. Click the RedBank button and provide customer number 222-22-2222 to see an example of a customer's accounts, as shown in Figure 15-11.

The screenshot shows a Mozilla Firefox browser window with the title "List Accounts - Mozilla Firefox". The address bar displays the URL <http://www.itsobank.ibm.com:9080/RAD75EJBWeb>ListAccounts>. The main content area is titled "ITSO RedBank" and features a logo. A navigation menu at the top includes "Rates", "RedBank" (which is selected and highlighted in green), and "Insurance". Below the menu, form fields are filled with customer information: SSN: 222-22-2222, Title: Ms, First name: Pinar, and Last name: Ugurlu. Buttons for "Update", "New Customer", and "Delete Customer" are present. A table below lists account numbers and balances:

Account Number	Balance
002-222001	65 484,23
002-222002	87,96
002-222003	654,65

Buttons for "Add Account" and "Logout" are located at the bottom of the table. A footer navigation bar includes "ITSO Home" and "RedBank".

Figure 15-11 ITSO Bank Web application

If you have any problems related to virtual hosts, restart the server and try again. WebSphere Application Server might need a restart to pick up virtual hosts changes.

15.3 Deploying application clients

To run a Java-based client/server application, the client application executes in a client container of some kind. You might, for example, use a graphical Swing application that calls EJBs on an application server. WebSphere Application Server V7 supports several different application client environments:

- ▶ Java EE client (JEE client):

This client uses services provided by the JEE client container.

This client is a Java application program that accesses EJBs, JDBC databases, and JMS queues. The JEE application client program runs on client machines. This program allows the same Java programming model as other Java programs. However, the JEE application client depends on the application client runtime to configure its execution environment, and it uses the JNDI name space to access resources, the same as you would in a normal server application (like a servlet).

The JEE application client brings the JEE programming model to the client, and provides:

- XML deployment descriptors
- JEE naming (`java:comp/env`), including EJB references and resource references

The JEE application client is launched using the `launchClient` script, which sets up the environment with the necessary classpaths, and so on, for you.

- ▶ Java thin client:

This client does not use services provided by the JEE client container.

This client provides a lightweight Java client programming model and is best suited for use in situations where a Java client application exists, but the application must be enhanced to make use of EJBs. It can also be used where the client application requires a thinner, more lightweight environment than the one offered by the JEE application client. The thin client supports JVMs from IBM, Sun and HP-UX. When launching the thin application client, you must set up the correct classpaths yourself and make sure that the required libraries for your application and the WebSphere libraries are included.

- ▶ Pluggable application client:

This client does not use services provided by the JEE Client Container.

This client is similar to the Thin application client, but does not include a JVM. The user is required to provide a JVM, and it can use the Sun JDK or the IBM JDK.

Note: The pluggable client is deprecated in WebSphere Application Server V7, and replaced by the Java thin client.

- ▶ Applet application client:

In the Applet client model, a Java applet embedded in an HTML document executes in a Web browser. With this type of client, the user accesses an enterprise bean in the application server through the Java applet in the HTML document.

- ▶ ActiveX to EJB Bridge application client:

The ActiveX application client allows ActiveX programs to access enterprise beans through a set of ActiveX automation objects. The ActiveX application client uses the Java Native Interface (JNI) architecture to programmatically access the Java virtual machine (JVM) API. Therefore, the JVM code exists in the same process space as the ActiveX application (Visual Basic, VBScript, or Active Server Pages files) and remains attached to the process until that process terminates. The ActiveX to EJB Bridge is supported on Windows only.

For detailed capabilities of each client container, search the Information Center for Client Applications, or visit

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

Install the application client environments from the WebSphere installation windows by selecting the **Launch the installation wizard for WebSphere Application Clients** option. The installation package contains the following installable components:

- ▶ IBM Java Runtime Environment (JRE), or an optional full Software Development Kit
- ▶ Java EE application client and Java thin application client
- ▶ ActiveX to EJB Bridge runtime for ActiveX to EJB Bridge application client applications (only for Windows)
- ▶ Pluggable Client (deprecated)
- ▶ Samples for the various application client containers

Note: The JEE client is automatically installed as part of a full WebSphere install. In other words, if you will run the client application on a machine that already has WebSphere installed, you do not need to install the WebSphere JEE client on top.

15.3.1 Defining application client bindings

The ITSO Bank application also has a standalone application client which we will use to demo the WebSphere Application Server application client container.

First we need to import the project containing the standalone application client into Rational Application Developer Assembly and Deploy and export it as an EAR file. Refer to Chapter 14.6, “Example: Packaging an application” on page 725 for details on how to perform such tasks. Do the following steps:

1. Import the appclient\RAD75AppClient.zip Project Interchange file from the 7672codesolution.zip file. Select both projects when importing.
2. Because the RAD75AppClient project has a specific binding configured for the EJBBankBean in the RAD75EJBEAR file we need to modify this binding to point to the same bean in the RAD75EJBWebEAR file, which is the application we have installed on our server.
 - a. To do this expand the **RAD75AppClient** project and double-click the **RAD75AppClient** deployment descriptor, similar to what is shown in Figure 14-8 on page 728 but for the RAD75AppClient.
 - b. Then click the **Open WebSphere Bindings** link from the right panel.
 - c. Select the **EJB Reference (ejb/bank)** and change the name from ejb/RAD75EJBEAR/RAD75EJB.jar/EJBBankBean#itso.bank.service.EJBBankRemote to itso.bank.service.EJBBankRemote.
 - d. Press **Ctrl-S** to save the deployment descriptor.
3. Export the **RAD75AppClientEAR** project and its RAD75AppClient module as described in Chapter 14.7, “Exporting to an EAR file” on page 730.

15.3.2 Launching the J2EE client

A JEE client application needs a container to run in. In this example, we will use the JEE application client container. This container can be started using the `launchClient` program in the `install_root/bin` directory. The `launchClient` program has the following syntax:

```
Usage: launchClient [-profileName pName | -JVMOptions options | -help | -?] <userapp> [-CC<name>=<value>] [app args]
```

The elements of syntax are:

-profileName	This option defines the profile of the application server process in a multi-profile installation. The <code>-profileName</code> option is not required for running in a single profile environment or in an application client installation. The default is <code>default_profile</code> .
-JVMOptions	This is a valid Java standard or nonstandard option string. Insert quotation marks around the option string.
-help, -?	Print the usage information.
<userapp.ear>	Type the path/name of the .ear file containing the client application.

The `-CC` properties are for use by the application client runtime. There are numerous parameters available, but we describe only the more commonly used parameters here. For full explanation of all parameters, execute `launchClient -help`.

-CCverbose	Use this option with <code><true false></code> to display additional informational messages. The default is <code>false</code> .
-CCclasspath	This property is a classpath value. When an application is launched, the system classpath is not used. If you need to access classes that are not in the EAR file or part of the resource classpaths, specify the appropriate classpath here. Multiple paths can be concatenated.
-CCjar	This is the name of the client JAR file within the EAR file that contains the application you want to launch. This argument is only necessary when you have multiple client JAR files in the EAR file.
-CCBootstrapHost	This option is the name of the host server you want to connect to initially.
-CCBootstrapPort	This option is the server port number. If not specified, the WebSphere default value (2809) is used.

-CCproviderURL	This option provides bootstrap server information that the initial context factory can use to obtain an initial context. A WebSphere Application Server initial context factory can use either a CORBA object URL or an IIOP URL. CORBA object URLs are more flexible than IIOP URLs and are the recommended URL format to use. This value can contain more than one bootstrap server address. This feature can be used when attempting to obtain an initial context from a server cluster. In the URL, you can specify bootstrap server addresses for all servers in the cluster. The operation will succeed if at least one of the servers is running, eliminating a single point of failure. The address list does not process in a particular order. For naming operations, this value overrides the -CCBootstrapHost and -CCBootstrapPort parameters. An example of a CORBA object URL specifying multiple systems is: -CCproviderURL=corbaloc:iiop:myserver.mycompany.com:9810,:mybackupserver.mycompany.com:2809
-CCtrace	Use this option with <true false> to have WebSphere write debug trace information to a file. The value true is equivalent to a trace string value of com.*=all=enabled. Instead of the value true you can specify a trace string, for example, -CCtrace=com.ibm.ws.client.*=all=enabled. Multiple trace strings can be specified by separating them with a colon (:). You might need this information when reporting a problem to IBM Service. The default is false.
-CCtracefile	This option is the name of the file to which to write trace information. The default is to output to the console.
-CCpropfile	This option is the name of a properties file containing launchClient properties. In the file, specify the properties without the -CC prefix. For example: verbose=true.

The app args are for use by the client application and are ignored by WebSphere.

To start the ITSO Bank standalone application client using the launchClient command, execute the command shown in Example 15-3.

Example 15-3 Launching ITSO Bank standalone application client

```
C:\IBM\WebSphere\AppServer\profiles\ITSOBank\bin>launchClient.bat
c:\RAD75AppClientEAR_withModifiedBinding.ear
```

IBM WebSphere Application Server, Release 7.0
Java EE Application Client Tool

```
Copyright IBM Corp., 1997-2008
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the Java EE Application Client Environment.
[2009-04-04 13:42:04:546 CEST] 00000000 W UOW=null
source=com.ibm.ws.ssl.config.SSLConfig org=IBM prod=WebSphere
component=Application Server thread=[P=321234:
0=0:CT]
CWPKI0041W: One or more key stores are using the default
password.
WSCL0035I: Initialization of the Java EE Application Client Environment
has completed.
WSCL0014I: Invoking the Application Client class
itso.rad75.client.control.BankDesktopController
```

The application will open and display a graphical window, as shown in Figure 15-12.

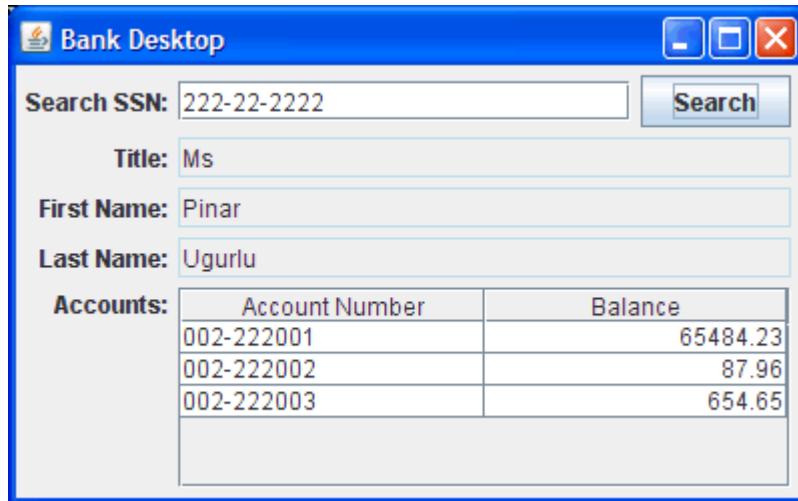


Figure 15-12 Running ITSO Bank standalone application client

15.4 Updating applications

WebSphere Application Server has features that allow applications to be updated and restarted at a fine-grained level. It is possible to update only parts of an application or module and only the necessary parts are restarted. You can:

- ▶ Replace an entire application (.ear file).

- ▶ Replace, add, or remove a single module (.war, EJB .jar, or connector .rar file).
- ▶ Replace, add, or remove a single file.
- ▶ Replace, add and remove multiple files by uploading a compressed file describing the actions to take.

If the application is running while being updated, WebSphere Application Server automatically stops the application, or only its affected components, updates the application, and restarts the application or components.

When updating an application, only the portion of the application code that changed needs to be presented to the system. The application management logic calculates the minimum actions that the system needs to execute in order to update the application. Under certain circumstances, the update can occur without stopping any portion of the running application.

WebSphere Application Server also has support for managing applications in a cluster for continuous availability. The action, Rollout Update, sequentially updates an application installed on multiple cluster members across a cluster. After you update an application's files or configuration, use the Rollout Update option to install the application's updated files or configuration on all cluster members of a cluster on which the application is installed.

Rollout Update does the following for each cluster member in sequence:

1. Saves the updated application configuration.
2. Stops all cluster members on a given node.
3. Updates the application on the node by synchronizing the configuration.
4. Restarts the stopped cluster members on that node.

This action updates an application on multiple cluster members while providing continuous availability of the application.

15.4.1 Replacing an entire application EAR file

To replace a full EAR with a newer version, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to update and click the **Update** button.
2. On the Preparing for the application update window, select the **Replace the entire application** option.
3. Select either the **Local file system** or **Remote file system** option. Click the **Browse** button to select the updated EAR file. Click **Next**.

4. Proceed through the remaining windows and make any changes necessary. For information about the windows, see “Deploying the application” on page 772. On the Summary window, click **Finish**.
5. When the application has been updated in the master repository, select the **Save** link.
6. If you are working in a distributed server environment, make sure that you also synchronize the changes with the nodes.
7. If the application update changes the set of URLs handled by the application (servlet mappings added, removed, or modified), make sure the Web server plug-in is regenerated and propagated to the Web server.

Note: It might take a few seconds for the WebSphere runtime to pick up the changes and restart the application as necessary. If your changes do not seem to have effect, wait and try again. You can also look at the SystemOut.log file for the application server to see when it has restarted the application.

15.4.2 Replacing or adding an application module

To replace only a module, such as an EJB or Web module of an application, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to update and click the **Update** button.
2. On the Preparing for the application update window, select the **Replace or add a single module** option.
3. In the Specify the path beginning with the installed application archive file... field, enter the relative path to the module to replace. For example, if you were to replace the HelloWeb module, enter HelloWeb. If you enter a path or file that does not exist in the EAR file, it will be added.
4. Select either the **Local file system** or **Remote file system** option and click the **Browse** button to select the updated module.
5. Click **Next**.
6. Proceed through the remaining windows and make any necessary changes. For information about the windows, see “Deploying the application” on page 772. On the Summary window, click **Finish**.

Note: If you are adding a Web module, make sure you select the detailed install option. This allows you to select the correct target server for the module in the Map modules to servers step, as well as specifying a context root for the module.

7. When the application has been updated in the master repository, select the **Save** link.
8. If you are working in a distributed server environment, make sure that you also synchronize the changes with the nodes.
9. If the application update changes the set of URLs handled by the application (servlet mappings added, removed, or modified), make sure the Web server plug-in is regenerated and propagated to the Web server.

Tip: Modules can also be managed using the Manage Modules page. Select **Applications** → **Application Types** → **WebSphere enterprise applications** and click the link for the application. Then click the **Manage Modules** link in the Modules section. Select the module to modify and then click the **Remove**, **Update**, or **Remove File** buttons.

Be aware: When writing this book, we added a new Web module to an already installed enterprise application. When we later selected the Manage Modules link for the application, the new module did not show. Selecting View Deployment Descriptor did not show the new module either. However, when exporting the whole application as an EAR file, the new module was included, so it had in fact been added.

15.4.3 Replacing or adding single files in an application or module

To replace a single file, such as a GIF image or a properties file in an application or module, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to update and click the **Update** button.
2. On the Preparing for the application installation window, select the **Replace or add a single file** option.
3. In the **Relative path to file** field, enter the relative path to the file to replace in the EAR file. For example, if you were to replace the logo.gif in the images directory of the HelloWeb.war Web module, you would enter HelloWeb.war/images/logo.gif. If you enter a path or file that does not exist in the EAR file, it will be added.

4. Select either the **Local file system** or **Remote file system** option and click the **Browse** button to locate the updated file. Click **Next**.
5. On the Updating Application window, click **OK**.
6. When the application has been updated in the Master repository, select the **Save** link.
7. If you are working in a distributed server environment, make sure that you also synchronize the changes with the nodes.

15.4.4 Removing application content

Files can also easily be removed either from an EAR file or from a module in an EAR file.

Removing files from an EAR file

To remove a file from an EAR file, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to remove the file from and click the **Remove File** button.
2. In the Remove file dialog box, select the file to be removed and click **OK**.
3. Save the configuration.

Removing files from a module

To remove a file from a module, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications** and click the link for the application to which the module belongs.
2. Click the **Manage Modules** link under the Modules section.
3. Select the module to remove the file from and click the **Remove File** button.
4. In the Remove a file from a module dialog, select the file to be removed and click **OK**.
5. Save the configuration.

15.4.5 Performing multiple updates to an application or module

Multiple updates to an application and its modules can be packaged in a compressed file, .zip, or .gzip format, and uploaded to WebSphere Application Server. The uploaded file is analyzed and the necessary actions to update the application are taken.

Depending on the contents of the compressed file, this method to update an application can replace files in, add new files to, and delete files from the installed application all in one single administrative action. Each entry in the compressed file is treated as a single file, and the path of the file from the root of the compressed file is treated as the relative path of the file in the installed application.

- ▶ To replace a file, a file in the compressed file must have the same relative path as the file to be updated in the installed application.
- ▶ To add a new file to the installed application, a file in the compressed file must have a different relative path than the files in the installed application.
- ▶ To remove a file from the installed application, specify metadata in the compressed file using a file named META-INF/ibm-partialapp-delete.props at any archive scope.

The ibm-partialapp-delete.props file must be an ASCII file that lists files to be deleted in that archive with one entry for each line. The entry can contain a string pattern, such as a regular expression that identifies multiple files. The file paths for the files to be deleted must be relative to the archive path that has the META-INF/ibm-partialapp-delete.props file.

- ▶ To delete a file from the EAR file (not a module), include a META-INF/ibm-partialapp-delete.props file in the root of the compressed file. In the .props file, list the files to be deleted. File paths are relative to the root of the EAR file.

For example, to delete a file named docs/readme.txt from the root of the HelloApp.ear file, include the line docs/readme.txt in the META-INF/ibm-partialapp-delete.props file in the compressed file.

- ▶ To delete a file from a module in the EAR, include a module_uri/META-INF/ibm-partialapp-delete.props file in the compressed file. The module_uri part is the name of the module, such as HelloWeb.war.

For example, to delete images/logo.gif from the HelloWeb.war module, include the line images/logo.gif in the HelloWeb.war/META-INF/ibm-partialapp-delete.props file in the compressed file.

- ▶ Multiple files can be deleted by specifying each file on its own line in the metadata .props file.

Regular expressions can also be used to target multiple files. For example, to delete all JavaServer Pages (.jsp files) from the HelloWeb.war file, include the line .*jsp in the HelloWeb.war/META-INF/ibm-partialapp-delete.props file. The line uses a regular expression, .*jsp, to identify all .jsp files in the HelloWeb.war module.

As an example, assume we have prepared the compressed HelloApp_update.zip file shown in Figure 15-13.

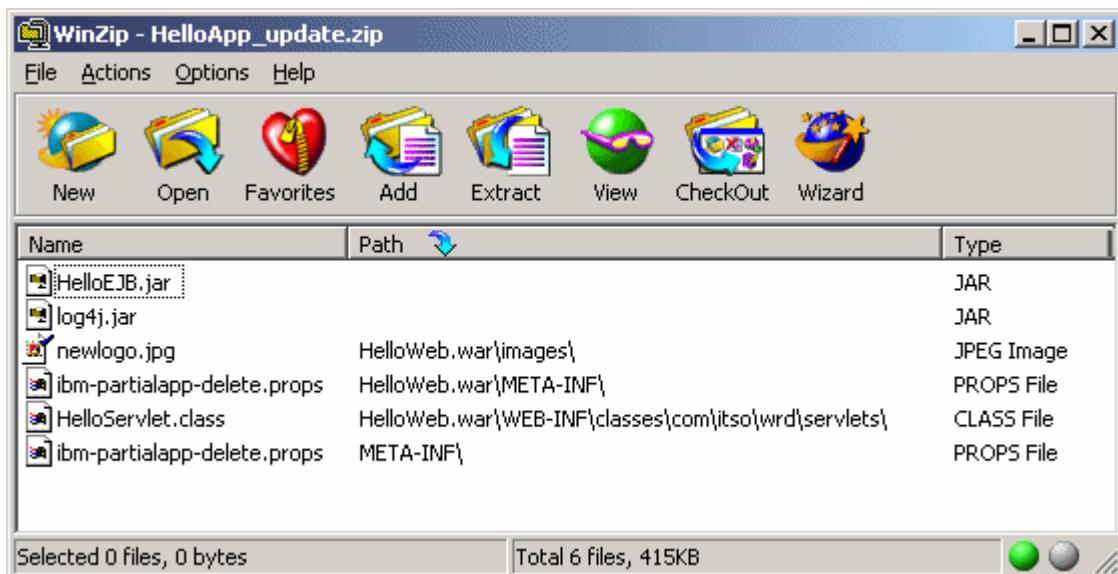


Figure 15-13 HelloApp_update.zip compressed file

The META-INF/ibm-partialapp-delete.props file contains the following line:

docs/readme.txt

The HelloWeb.war/META-INF/ibm-partialapp-delete.props contains the following lines:

images/logo.gif

When performing the partial application update using the compressed file, WebSphere does the following actions:

- ▶ Adds the log4j.jar file to the root of the EAR.
- ▶ Updates the entire HelloEJB.jar module.
- ▶ Deletes the docs/readme.txt file (if it exists) from the EAR file, but not from any modules.
- ▶ Adds the images/newlogo.jpg file to the HelloWeb.war module.
- ▶ Updates the HelloServlet.class file in the WEB-INF/classes/com/itso/wrd/servlets directory of the HelloWeb.war module.
- ▶ Deletes the images/logo.gif file from the HelloWeb.war module.

To perform the actions specified in the HelloWeb_updated.zip file, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to update and click the **Update** button.
2. On the Preparing for the application installation window, select the **Replace, add, or delete multiple files** option.
3. Select either the **Local file system** or **Remote file system** option and click the **Browse** button to select the compressed ZIP file with the modifications you have created. Click **Next**.
4. On the Updating Application window, click **OK**.
5. When the application has been updated in the Master repository, select the **Save To Master Configuration** link.
6. If in a distributed server environment, make sure the **Synchronize changes with Nodes** option is selected so that the application is distributed to all nodes. Click the **Save** button. The application is distributed to the nodes, updated, and restarted as necessary
7. If the application update changes the set of URLs handled by the application (servlet mappings added, removed or modified), make sure the Web server plug-in is regenerated and propagated to the Web server.

15.4.6 Rolling out application updates to a cluster

The Rollout Update feature allows you to easily roll out a new version of an application, or part of an application using the techniques described previously, to a cluster. The Rollout Update feature takes care of stopping the cluster members, distributing the new application, synchronizing the configuration, and restarting the cluster members. The operation is done sequentially over all cluster members in order to keep the application continuously available.

When stopping and starting the cluster members, the Rollout Update feature works on node level, so all cluster members on a node are stopped, updated, and then restarted, before the process continues to the next node.

Because the Web server plug-in module is not able to detect that an individual application on an application server is unavailable, the Rollout Update feature always restarts the whole application server hosting the application. Because of this, if HTTP session data is critical to your application, it should either be persisted to database or replicated to other cluster members using the memory-to-memory replication feature.

The order in which the nodes are processed and the cluster members are restarted is the order in which they are read from the cell configuration repository. There is no way to tell the Rollout Update feature to process the nodes and cluster members in any particular order.

Assume that we have an environment with two nodes, ITSOBankNode1 and ITSOBankNode2, and a cluster called ITSOBankCluster, which has one cluster member on each node (ITSOBankServer1 on ITSOBankNode1 and ITSOBankServer2 on ITSOBankNode2). Assume we have an application called RAD75EJBWebEAR deployed and running on the cluster.

To update this application using the Rollout Update feature, we would do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to update and click the **Update** button.
2. On the Preparing for the application installation window, select the appropriate action depending on the type of update. In this example, we will update the entire application EAR to a new version, so we select the **Replace the entire application** option.
3. Select either the **Local file system** or **Remote file system** option and click the **Browse** button to select the updated EAR file. Click **Next**.
4. Proceed through the remaining windows and make any changes necessary. For information about the windows, see “Deploying the application” on page 772. On the Summary window, click **Finish**.

- When the application has been updated in the master repository, the status window shown in Figure 15-14 is displayed.

ADMA5013I: Application RAD75EJBWebEAR installed successfully.

Application RAD75EJBWebEAR installed successfully.

If you want to do a rolling update of the application on the cluster(s) on which it is installed, then click Rollout Update. A rolling update will save all changes made in this session to the master configuration, then synchronize and recycle the cluster members on each node, one node at a time.

[Rollout Update](#)

To start the application, first save changes to the master configuration.

The application might not be immediately available while being started on all servers.

Changes have been made to your local configuration. You can:

- [Save directly to the master configuration](#).
- [Review changes before saving or discarding](#).

To work with installed applications, click the "Manage Applications" link.

[Manage Applications](#)

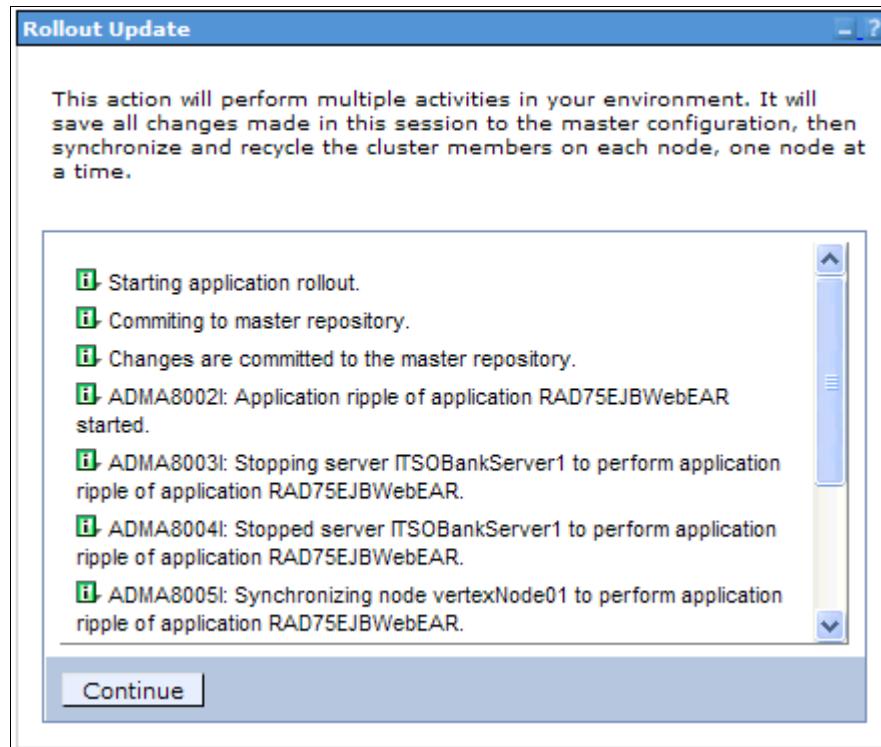
Figure 15-14 Preparing for application rollout

You then have two options to start the rollout action:

- Click the **Rollout Update** link.
- Click the **Manage Applications** link and on the Enterprise Applications window, select the application and click the **Rollout Update** button.

Note: Do not click the **Save directly to the master configuration** link or otherwise save the configuration yourself. The Rollout Update will do that for you. If you save the configuration yourself, the rollout update action will be canceled and it will be handled as a normal application update.

During the rollout, the window in Figure 15-15 is displayed in the status window.



Although the Rollout Update feature makes it very easy to roll out an application to a cluster while keeping the application continuously available, make sure that your application can handle the roll out.

For example, assume you have version 1.0 of an application running in a cluster consisting of two application servers, server1 and server2, and that HTTP session data is persisted to a database. When you roll out version 2.0 of the application and server1 is stopped, the Web server plug-in redirects the users on server1 to server2. Then, when server1 is started again, bringing up version 2.0 of the application, the plug-in will start distributing requests to server1 again. Now, if the application update incurred a change in the interface of any class stored in the HTTP session, when server1 tries to get these session objects from the database, it might run into a deserialization or class cast exception, preventing the application from working properly.

Another situation to consider is when the database structure changes between application versions, as when tables or column names change name or content. In that case, the whole application might need to be stopped and the database migrated before the new version can be deployed. The Rollout Update feature would not be suitable in that kind of scenario.

So it is very important to understand the changes made to your application before rolling it out.

WebSphere Virtual Enterprise: If you want even more advanced rollout and application versioning capabilities than what is available in WebSphere Application Server, take a look at the WebSphere Virtual Enterprise product at:

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

WebSphere Virtual Enterprise extends an existing WebSphere infrastructure and brings, among many other features, functions to validate new versions of applications in production environments, and then to roll them out seamlessly, with features to drain application servers from existing users before taking them offline for the update.

15.4.7 Hot deployment and dynamic reloading

Hot deployment and dynamic reloading characterize how application updates are handled when updates to the applications are made by directly manipulating the files on the server. In either case, updates do not require a server restart, though they might require an application restart:

- ▶ Hot deployment of new components:

Hot deployment of new components is the process of adding new components, such as WAR files, EJB JAR files, EJBs, servlets, and JSP files to a running application server without having to stop and then restart the application server.

However, in most cases, such changes require the application itself to be restarted, so that the application server runtime reloads the application and its changes.

- ▶ Dynamic reloading of existing components:

Dynamic reloading of existing components is the ability to change an existing component without the need to restart the application server for the change to take effect. Dynamic reloading can involve changes to the:

- Implementation of an application component, such as changing the implementation of a servlet
- Settings of the application, such as changing the deployment descriptor for a Web module

To edit the files manually, locate the binaries in use by the server. See 1.4.4, “Configuration and application data repository” on page 18. Although the application files can be manually edited on one or more of the nodes, these changes will be overwritten the next time the node synchronizes its configuration with the deployment manager. Therefore, we recommend that manual editing of an application’s files should only be performed in the master repository, located on the deployment manager machine.

Note: Unless you are familiar with updating applications by directly manipulating the server files, it might be better to use the administrative console Update wizard.

There are three settings that affect dynamic reload:

- ▶ Reload classes when application files are updated:

In order for application files to be reloaded automatically after an update, the **Override class reloading settings for Web and EJB modules** setting must be enabled and the **Polling interval for updated files** setting must be greater than 0.

Select **Applications** → **Application Types** → **WebSphere enterprise applications**, and click the link for the application. In the Detail properties section, click the **Class loading and update detection** link.

- ▶ Application Server class loader policy:

The application server's class loader policy should be set to Multiple. If it is set to Single, the application server will need to be restarted after an application update.

Select **Servers** → **Server Types** → **WebSphere application servers**, and click the server name. The setting is found in the General Properties section.

- ▶ JSP Reload options for Web modules:

A Web container reloads a Web module only when this setting is enabled.

Select **Applications** → **Application Types** → **WebSphere enterprise applications**, and click the link for the application. In the Web Module Properties section, click the **JSP and JSF options**, and then select the **JSP enable class reloading** option and enter a polling interval.

For more information about using hot deployment and dynamic reload, see the topics, *Updating applications* and *Hot deployment and dynamic reloading*, in the Information Center.

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 publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 800. Note that some of the documents referenced here might be available in softcopy only.

- ▶ *WebSphere Application Server V7 Default Messaging Provider Administration Guide*, SG24-7770
- ▶ *WebSphere Application Server V7 Security Guide*, SG24-7660
- ▶ *WebSphere Application Server V7 Web Services Guide*, SG24-7758
- ▶ *WebSphere Application Server V6.1 Problem Determination: IBM Redpaper Collection*, SG24-7461
- ▶ *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304
- ▶ *WebSphere Application Server Version 6.1 Feature Pack for EJB 3.0*, SG24-7611
- ▶ *Rational Application Developer V7.5 Programming Guide*, SG24-7672
- ▶ *WebSphere Application Server V7.0: Technical Overview*, REDP-4482
- ▶ *WebSphere Application Server V6: Diagnostic Data*, REDP-4085

Online resources

These websites are also relevant as further information sources:

- ▶ WebSphere Application Server V7 Information Center:
<http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp>
- ▶ List of supported software for WebSphere Application Server V7.0:
<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27012369>

- ▶ WebSphere Application Server V6.1 Information Center:
<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>
- ▶ IBM Support Assistant:
<http://www.ibm.com/software/support/isa/>
- ▶ IBM WebSphere Customization Tools V7.0 for Windows:
<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg24020368>
- ▶ Introducing the WCT for WebSphere z/OS Version 7:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS3357>
- ▶ WebSphere for z/OS Version 7 - Configuration Planning Spreadsheets:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS3341>
- ▶ IBM AIX Toolbox download information:
<http://www-03.ibm.com/systems/power/software/aix/linux/toolbox/download.html>
- ▶ IBM WebSphere Installation Factory V7.0.0.3 for AIX:
<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg24020213>
- ▶ *WebSphere z/OS V6.1 - Making the DMGR Mobile*, at:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101140>
- ▶ *Introducing The WebSphere V7 Job Manager for z/OS*, at:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101341>
- ▶ *Introducing the WebSphere V7 Secure Proxy Server for z/OS*, at:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101423>
- ▶ *Directing SYSPRINT Output to an HFS File in WebSphere for z/OS*, at:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD101087>
- ▶ *WebSphere Application Server for z/OS - Planning for Test, Production and Maintenance*, at:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100396>
- ▶ *WebSphere Application Server for z/OS V7 - Dispatch Timeout Improvements*, at:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101374>
- ▶ *Understanding SMF Record Type 120, Subtype 9*, at:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101342>
- ▶ *Migrating to WebSphere z/OS V7*, at:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101329>

- ▶ *Command assistance simplifies administrative scripting in WebSphere Application Server, at:*
http://www.ibm.com/developerworks/websphere/library/techarticles/0812_rhodes/0812_rhodes.html
- ▶ *Sample Scripts for WebSphere Application Server Versions 5 and 6, at:*
<http://www.ibm.com/developerworks/websphere/library/samples/SampleScripts.html>
- ▶ *Java Platform Enterprise Edition, v 5.0 API Specifications, at:*
<http://java.sun.com/javaee/5/docs/api/>
- ▶ IBM Developer Kit and Runtime Environment, Java Technology Edition, Version 6 diagnosis documentation:
<http://www.ibm.com/developerworks/java/jdk/diagnosis/60.html>
- ▶ WebSphere Virtual Enterprise:
<http://www.ibm.com/software/webservers/appserv/extend/virtualenterprise/>
- ▶ IBM Support Assistant:
<http://www-01.ibm.com/software/support/isa/>
- ▶ IBM Education Assistant module for The IBM Garbage Collection and Memory Visualizer:
http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.was_v7/was/7.0/ProblemDetermination/WASv7_GCMVOverview/player.html
- ▶ IBM Pattern Modelling and Analysis for Java Garbage Collector:
<http://www.alphaworks.ibm.com/tech/pmat>
- ▶ Tivoli Composite Application Manager for Web Resources documentation:
http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.itcamwas_wr.doc_6.2/welcome.html

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, Redpaper publications, Technotes, draft publications, and Additional materials, as well as order hardcopy Redbooks publications, at this website:

ibm.com/redbooks

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services

IBM



Redbooks

WebSphere Application Server V7 Administration and Configuration Guide

(1.5" spine)
1.5" <-> 1.998"
789 <-> 1051 pages



Redbooks®

WebSphere Application Server V7 Administration and Configuration Guide

Learn about WebSphere Application Server V7

This IBM Redbooks publication provides system administrators and developers with the knowledge to configure a WebSphere Application Server V7 runtime environment, to package and deploy applications, and to perform ongoing management of the WebSphere environment.

Configure and administer a WebSphere system

As one in a series of IBM Redbooks publications and Redpapers publications for V7, the entire series is designed to give you in-depth information about key WebSphere Application Server features. In this book, we provide a detailed exploration of the WebSphere Application Server V7 runtime administration process.

The book includes configuration and administration information for WebSphere Application Server V7 and WebSphere Application Server Network Deployment V7 on distributed platforms and WebSphere Application Server for z/OS V7.

The following publications are considered prerequisites to this book:

- ▶ *WebSphere Application Server V7.0: Technical Overview*, REDP-4482
- ▶ *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708

**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**