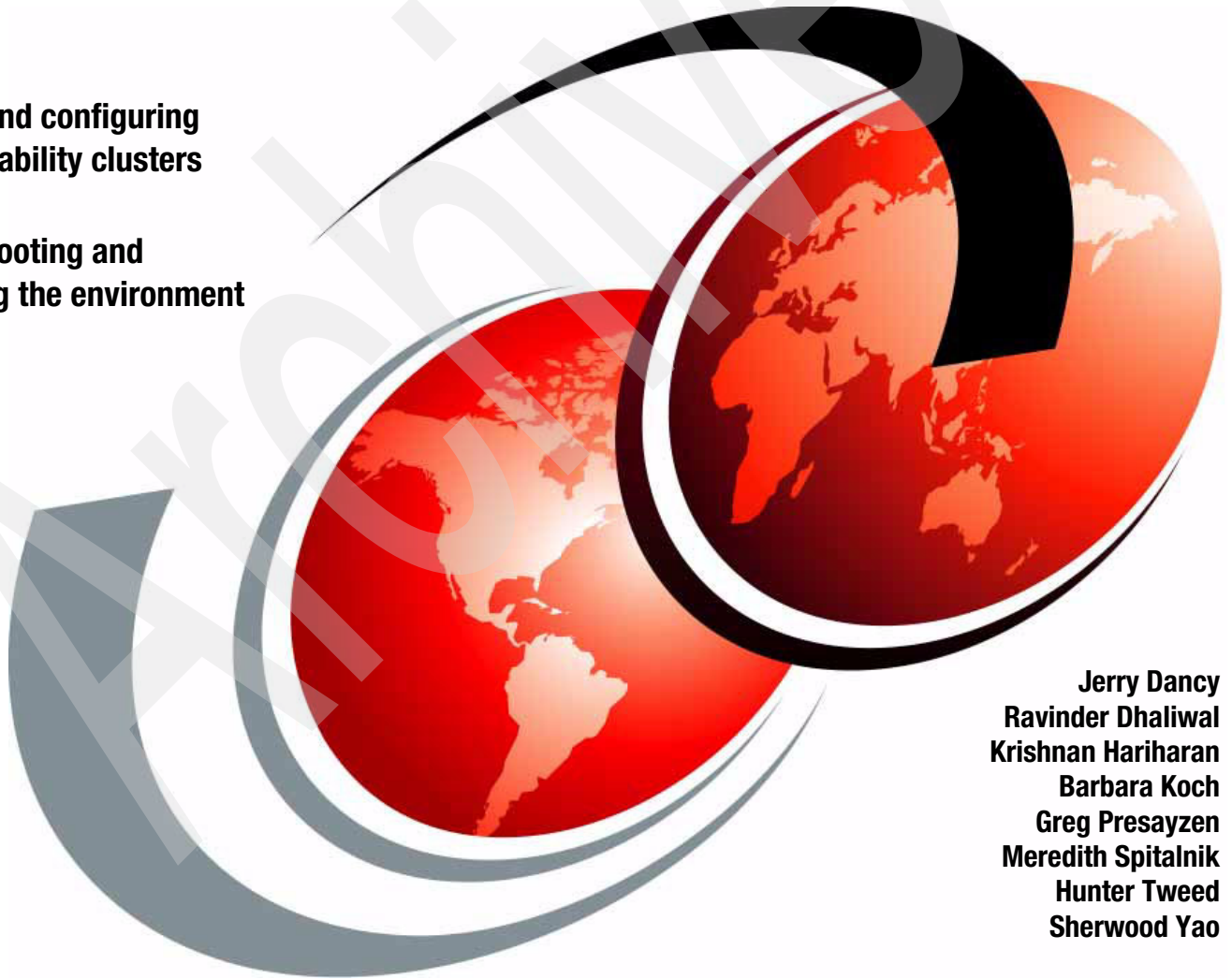


WebSphere Portal Version 6 Enterprise Scale Deployment Best Practices

Planning for Portal deployment

Building and configuring
high-availability clusters

Troubleshooting and
monitoring the environment



Jerry Dancy
Ravinder Dhaliwal
Krishnan Hariharan
Barbara Koch
Greg Presayzen
Meredith Spitalnik
Hunter Tweed
Sherwood Yao

Redbooks



International Technical Support Organization

**WebSphere Portal Version 6 Enterprise Scale
Deployment Best Practices**

March 2007

Archived

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

First Edition (March 2007)

This edition applies to WebSphere Portal Version 6.

© Copyright International Business Machines Corporation 2007. All rights reserved.

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

Contents

Notices	ix
Trademarks	x
Preface	xi
The team that wrote this IBM book	xi
Thanks to the following people for their contributions to this project:	xiii
Become a published author	xiv
Comments welcome	xiv
Chapter 1. Introduction	1
1.1 What is new in WebSphere Portal Version 6	3
1.1.1 WebSphere Portal and SOA	3
1.1.2 Simplified portlet creation	4
1.1.3 Workflow support	5
1.1.4 Interactive forms in portal interfaces	5
1.1.5 User interface enhancements	5
1.1.6 Composite applications and templates	6
1.1.7 Personalizing the Portal	9
1.2 WebSphere Portal programming model	12
1.2.1 Content and Web content management	15
1.3 Operations and deployment	15
1.3.1 Multiple LDAP support	15
1.3.2 Database domains	17
Chapter 2. Planning for your Portal 6.0 high- availability deployment	21
2.1 System requirements	22
2.1.1 Hardware requirements	22
2.1.2 Software requirements	24
2.1.3 Network connectivity requirements	26
2.2 Typical deployment scenarios	27
2.2.1 Single-server topology	27
2.2.2 Vertical scaling topology	27
2.2.3 Horizontal scaling topology	28
2.2.4 Mixed horizontal and vertical scaling topology	29
2.3 Clustering considerations for high-availability	29
2.3.1 WebSphere Process Server considerations	29
2.3.2 Multiple LDAP overview	30
2.3.3 Database domain (database split) overview	30
2.3.4 Profiles	32
2.3.5 Master configuration repository	33
2.3.6 Web server considerations	33
2.3.7 Hardware requirements for high availability	35
2.3.8 Example architectures in operation	36
2.4 Recommendations and best practices: how many environments do I need?	39
2.4.1 Purposes of each environment	41
2.4.2 Recommendations	42
2.5 Introduction to the deployment scenarios used in this book	44
2.5.1 Horizontal cluster with single database and single LDAP	44
2.5.2 Horizontal cluster with database domain	44

2.5.3	Horizontal cluster with database domain and multiple LDAP	45
2.5.4	Vertical cluster with single database and single LDAP	46
2.5.5	Vertical cluster with database domain	47
2.5.6	Vertical cluster with database domain and multiple LDAP	48
Chapter 3.	Different deployment scenarios: building clustered environments	51
3.1	Overview	52
3.2	Recommendations for navigating this chapter	52
3.3	Introduction	52
3.4	Horizontal cluster (includes WebSphere Process Server) installation and configuration steps	54
3.4.1	Install and upgrade WSAS and WPS on the Deployment Manager (DMGR)	56
3.4.2	Install WSAS and WPS on the primary cluster node, node1	70
3.4.3	Prepare the DMGR and node1 for the Portal install	80
3.4.4	Install Portal onto the managed node, node1	84
3.4.5	Configure primary node 1 to an external database	93
3.4.6	Create the cluster definition	107
3.4.7	Install WSAS and WPS on future cluster node, node 2, and subsequent nodes	109
3.4.8	Install Portal on the managed node, node 2, and subsequent nodes	117
3.4.9	Add node 2 and subsequent nodes to the cluster definition	119
3.4.10	Configure Portal nodes and the DMGR for LDAP security with Realm Support	122
3.4.11	Configure Portal to use a remote Web server	131
3.4.12	Optional steps after cluster creation	135
3.5	Horizontal cluster (without WebSphere Process Server) installation and configuration steps	136
3.5.1	Installation WSAS on the Deployment Manager	138
3.5.2	Install WebSphere Portal on the primary cluster node, node 1	146
3.5.3	Configure primary node 1 to an external database	152
3.5.4	Create cluster definition	153
3.5.5	Install WebSphere Portal on node 2 and subsequent nodes	160
3.5.6	Add node 2 and subsequent nodes to the cluster definition	160
3.5.7	Configure Portal nodes and the DMGR for LDAP security with Realm Support	166
3.5.8	Configure Portal to use remote Web Servers	166
3.5.9	Optional steps after cluster creation	167
3.6	Adding a Horizontal node to a cluster with WebSphere Process Server	168
3.7	Add a Horizontal node to a cluster without WebSphere Process Server	172
3.8	Vertical cluster installation and configuration steps	176
3.9	Vertical cluster (without WebSphere Process Server) install and configure steps	181
Chapter 4.	Multiple LDAP directory support	187
4.1	Advantages of multiple LDAP directories	188
4.1.1	Overview of virtual portals and realms	188
4.1.2	The relationship between virtual portals and realms	189
4.1.3	How realms were extended in WebSphere Portal Version 6	190
4.2	Familiarizing yourself with the LDAP infrastructure	192
4.2.1	Before you begin configuring multiple LDAP directories	192
4.2.2	Tools to help you understand your LDAP directory	193
4.2.3	Using an LDAP browser tool	193
4.2.4	Using ldapsearch.exe	195
4.2.5	Multiple LDAP directory scenarios	197
4.2.6	Planning for multiple LDAP directories	197
4.2.7	Scenario 1, default realm with multiple LDAP directories (one to many)	197
4.2.8	Scenario 2, one realm per LDAP directory (one-to-one)	198

4.2.9 Scenario 3, multiple realms & multiple LDAP directories (many-to-many)	199
4.3 Configuring multiple LDAP directories	200
4.3.1 Currently supported LDAP directories.	200
4.3.2 Pre-requisites	201
4.3.3 Disabling WebSphere Application Server security	202
4.3.4 Enabling security with realm support	207
4.3.5 Configuring an additional LDAP directory and enabling realms	214
4.3.6 Steps for adding an additional LDAP directory	216
4.3.7 Creating two virtual portals to test multiple LDAP directories	225
4.4 Domino LDAP and groups	227
4.5 Enabling multiple LDAP directories in a cluster	230
Chapter 5. Database domains	233
5.1 What are database domains?	234
5.1.1 Terminology	234
5.1.2 How is Portal data organized?	236
5.2 Advantages of database domains.	238
5.3 Examples of moving and sharing database domains	241
5.3.1 Properties files associated with database domains	242
5.4 Example 1, moving database domains	243
5.5 Example 2, moving domains between database types	259
5.6 Example 3, sharing domains amongst Portal nodes	275
5.6.1 A quick recap of example 2.	275
5.6.2 A summary of example 3	276
5.7 Example 4, sharing domains between separate clusters	293
5.7.1 A quick recap of example 3.	293
5.8 Using database domains in a production environment	313
Chapter 6. Strategies for tuning and testing	321
6.1 Functional test of a new installation	322
6.2 Planning for Performance testing and tuning	323
6.2.1 Planning your test	323
6.2.2 Establish consistent hardware and software set up	324
6.2.3 Apply base tuning parameters	324
6.2.4 Run baseline performance tests before deploying your applications	326
6.3 Scripting performance tests	326
6.3.1 Performance metrics.	326
6.3.2 Building scripts	327
6.3.3 Testing scenarios	329
6.3.4 Setting goals	329
6.4 Executing your test	330
6.4.1 Establish baseline performance	330
6.4.2 Deploy applications one at a time and retest	330
6.4.3 Analysis and comparison to baseline	331
6.4.4 Monitoring: what to monitor?	331
6.4.5 Monitoring database performance	333
6.4.6 Performance maintenance: keeping it fast	334
6.4.7 Final test should reflect user behavior	335
Chapter 7. Configuration management: moving between environments	337
7.1 The Portal staging process	338
7.1.1 Terminology	338
7.2 Deployment and build process	339
7.2.1 Object IDs	340

7.2.2 Custom unique names	340
7.2.3 The XML configuration interface	341
7.2.4 ReleaseBuilder	342
7.3 Transferring Portal artifacts using XMLAccess	342
7.3.1 Transfer process	344
7.3.2 Exporting a sample page using XMLAccess	344
7.3.3 Exporting and importing a new page	345
7.4 Moving a differential release using ReleaseBuilder	346
7.4.1 ReleaseBuilder process overview	346
7.4.2 Using the ReleaseBuilder tool	347
7.5 A step-by-step guide to moving a release between environments	349
7.5.1 Prepare source environment	349
7.5.2 Build the release	355
7.5.3 Prepare the target environment	357
7.5.4 Import release	357
7.5.5 Import the release into the target server	358
7.5.6 Post transfer actions	358
7.6 Configuration maintenance tasks	358
7.6.1 Handling orphan data	358
7.6.2 Delayed cleanup of deleted portal resources	361
7.6.3 Deploying themes and skins	362
7.7 Troubleshooting	364
7.7.1 Problems importing portlets	364
7.7.2 Problems importing pages	365
7.7.3 Problems importing policies	365
Chapter 8. Troubleshooting and monitoring	367
8.1 Problem determination approaches	368
8.1.1 WPSconfig task failure	368
8.1.2 Portal install failure	373
8.1.3 Portal upgrade failure	388
8.1.4 Portal JVM crash	393
8.1.5 Portal Runtime failure	397
8.1.6 Providing logs to support	400
8.2 The IBM serviceability tools strategy	402
8.3 Self service resources	403
8.3.1 WebSphere Portal Product Library	403
8.3.2 WebSphere Portal version 6.0 Information Center	403
8.3.3 WebSphere Support Pages	403
8.4 Monitoring WebSphere Portal	404
8.4.1 Monitoring techniques	405
8.4.2 Portal update strategy	406
Appendix A. Deployment automation options	407
A.1 Motivation of deployment automation	408
A.2 Tools and technologies used	408
A.2.1 ANT/Maven	408
A.2.2 WSAdmin and Jython	412
A.2.3 WebSphere Application Server Installation Factory	414
A.3 Sample scenarios of automation	415
A.3.1 Use Installation Factory to build custom WAS install image	416
A.3.2 Use WSADMIN to automate application server configuration change	418
A.3.3 Use ANT script to automate portal server operations	421

Appendix B. Portal 6 Update Installer	425
B.1 Portal 6 Update Installer	426
B.1.1 Downloading the Portal Update Installer	426
B.1.2 Clustering considerations	426
B.1.3 Sample fix installation	426
B.1.4 Update Portal Wizard Graphical User Interface overview	432
B.1.5 updatePortal command line overview	433
Appendix C. Services Oriented Architecture and Portal	441
C.1 Introduction	442
C.2 SOA Definition	442
C.3 SOA life cycle	443
C.3.1 People entry point	447
C.3.2 Process entry point	447
C.3.3 Information entry point	447
C.3.4 Connectivity and reuse entry point	448
C.3.5 People entry point	448
C.4 Composite applications	449
Related publications	453
IBM Redbooks	453
Online resources	453
How to get IBM Redbooks	453
Help from IBM	453
Index	455

Archived

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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


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

COPYRIGHT LICENSE:

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

Trademarks

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

AIX®	OpenPower™	Tivoli®
BladeCenter®	OS/390®	WebSphere®
Cloudscape™	Power Architecture™	Workplace™
CICS®	POWER™	Workplace Forms™
Domino®	pSeries®	Workplace Messaging®
DB2 Universal Database™	Rational®	Workplace Web Content
DB2®	Redbooks™	Management™
IBM®	Redbooks (logo)  ™	z/OS®
iSeries™	RDN™	zSeries®
Lotus Notes®	RS/6000®	z9™
Lotus®	SecureWay®	
Notes®	System z™	

The following terms are trademarks of other companies:

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

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

Snapshot, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, 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.

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

Active Directory, Internet Explorer, Microsoft, MS-DOS, Outlook, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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

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

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

Preface

IBM® WebSphere® Portal Version 6.0 is an enterprise portal solution with the complete portal services necessary to deliver a single point of personalized interaction to applications, content, business processes, and people for a unified user experience. WebSphere Portal provides the following:

- ▶ Enhanced Web content management, portal workflow, electronic forms integration and collaboration to help deliver improved operational efficiency and productivity.
- ▶ Powerful new tools and application templates, enabling the quick building of business services and applications that help accelerate application and content deployment, which innovatively deliver on the promise of a service oriented architecture (SOA).
- ▶ A responsive and reliable portal platform from a leader in the enterprise portal market.

This book introduces new key technical features in WebSphere Portal Version 6.0 and discusses how these new features have a strong bearing on best practices for deploying WebSphere Portal. Specifically, it focuses on the following areas:

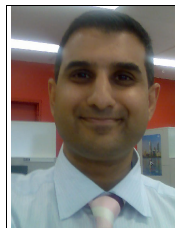
- ▶ Planning for deployment with an overview of several typical deployment scenarios.
- ▶ High-availability deployment strategies by installing IBM WebSphere Portal V6.0 in a clustered deployment/Network Deployment configuration.
- ▶ Configuring WebSphere Portal Version 6 to support multiple different LDAP directories.
- ▶ Considerations and implementation details for configuring Portal to share data between different database domains; thereby, providing more options for operators to establish a distributed portal operation.
- ▶ Strategies for tuning and testing your WebSphere Portal 6 environment, allowing you to better evaluate the total system throughput and responsiveness, identify the low-performing part of the system, and to execute tuning steps to bring overall satisfactory performance results.
- ▶ Configuration Management, namely the procedures necessary to move your WebSphere Portal environment through pre-production release phases (testing, staging) and deploy into the Production environment.
- ▶ Best practice approaches to problem determination and troubleshooting.
- ▶ Available tools or technologies for deployment automation, reducing the manual steps required by the portal server installation and configuration process.
- ▶ Discussion of how to use the IBM WebSphere Portal update installer application to install interim or cumulative fixes and fix packs in WebSphere Portal Version 6.0.
- ▶ A discussion of Services Oriented Architecture (SOA) and what this means within the context of Portal.

The team that wrote this IBM book

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



Jerry Dancy is as a Technical Lead for the WebSphere Portal Support Level 2 team. He has three years of experience in WebSphere Portal Support and previously worked as an Oracle® DBA for four years. He holds a degree in Accounting and CIS from Appalachian State University. His areas of expertise include install, upgrade, configuration, and clustering of WebSphere Portal. He has written extensively on WebSphere Portal installation, configuration, and clustering. Jerry is also an author of the IBM book *WebSphere Portal V5.0 Production Deployment and Operations Guide*, SG24-6391.



Ravinder Dhaliwal is a Senior Technical Consultant and Software Services Manager based in IBM Australia. He has more than 10 years of consulting experience, working in the Lotus® collaborative infrastructure design and deployment arena. For the last few years, Ravinder worked extensively with a wide variety of clients to design and deploy high-availability WebSphere Portal environments. He specializes in directory integration and single sign-on solutions with WebSphere Portal, having extensive experience with LDAP and the IBM Tivoli® products. He regularly acts as an instructor for WebSphere Portal and Lotus Domino® and has written extensively on Portal and Domino/Sametime integration. Ravinder has worked at IBM Lotus for six years and is also the co-author of the book *Lotus Domino 6.5.1 and Extended Products Integration Guide*, SG24-6357, as well as the *Domino Access for Microsoft Outlook*, SG24-9126, IBM Redpaper.



Krishnan Hariharan is a Senior IT Architect in IBM Software Services for Lotus (ISSL). In the role of a Solution Architect his core responsibilities include, providing technical vision and architectural direction to clients to solve their business challenges, providing technical leadership to enhance software and professional services sales, and creating ISSL offerings/assets by leveraging repeatable processes and capabilities that can generate more revenue for the organization. His areas of expertise are WebSphere Portal and Content Management. He joined IBM in 2002, and prior to joining IBM Software Group, he was part of the Portals, Content and e-Commerce practice within GBS. He is also a co-author of the IBM book, *A Portal Composite Pattern Using WebSphere Portal V5*, SG24-6087.



Barbara Koch Barbara Koch is a member of the Channel Technical Sales Team at IBM Germany. In her role as an IT Specialist, she assists IBM Business Partners in technical presales situations related to the Lotus software portfolio, such as IBM WebSphere Portal, IBM Workplace™, and Lotus Domino. In 2001, she joined IBM and worked for three years with IBM Software Services for Lotus with large clients where she was involved in the design and architecture of large-scale Lotus Messaging infrastructures. Finally, Barbara was also an author of the IBM book *IBM Workplace Collaboration Services: Release 2.5 Deployment Guide*, SG24-6777.



Greg Presayzen is a Staff Software Engineer for IBM Support based in Westford MA. He holds a degree in Computer Science, as well as minors in Economics and Mathematics & Statistics from the University of Massachusetts at Amherst. Before joining IBM in 2003, Greg worked in various IT roles at Compaq Computers and Fidelity Investments. Greg began his time at IBM supporting Lotus Domino and extended products. More recently he specialized in supporting infrastructure and deployment issues with the IBM Workplace and IBM WebSphere Portal related products. He is an IBM

Certified System Administrator for Workplace Messaging® and serves as Product Area Expert and escalation point for worldwide IBM Workplace technical problems.



Meredith Spitalnik is an Architect with Ascendant Technology, a premier IBM business partner based in the US <http://www.atech.com/>. She has nine years of consulting experience in the portal space, most of that working with large-scale enterprise deployments of WebSphere Portal. She holds a degree in Computer Science from Boston University. Her areas of expertise include installation, configuration, administration, troubleshooting, development, and deployment of WebSphere Portal, Personalization, and WCM. She has written several newsletter articles on a variety of

Portal-related topics.



Hunter Tweed works for the WebSphere Portal Support Level 2 team. He has two years of experience in WebSphere Portal Support. He holds a degree in Computer Science from North Carolina State University. His areas of expertise include install, upgrade, configuration, and clustering of WebSphere Portal. He has written extensively on Portal deployment issues and troubleshooting techniques.



Sherwood Yao is a Solution Architect for Workplace and Portal Foundation development organization. He joined the Portal development team as a software engineer in 2001 since the first major release of WebSphere Portal. He was responsible for the development of various WebSphere suite products installation and configuration program, and helped large scale deployment of IBM software solutions using WebSphere Portal.

Since 2005, Sherwood has worked as a consulting architect to guide multiple Fortune 500 clients to implement and deploy large scale end-to-end enterprise intranet solutions using WebSphere Portal and Workplace.



John Bergland is a Project Leader at the International Technical Support Organization, Cambridge Center. He manages projects that produce IBM Redbooks™ about Lotus software products. Before joining the ITSO in 2003, John worked as an Advisory IT Specialist with IBM Software Services for Lotus, specializing in Lotus Notes®, Domino messaging, and collaborative solutions.

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

- ▶ David Taber, IBM Software Group, WPLC, WebSphere Portal Clustering, IBM, Durham, NC, USA
- ▶ Marshall Lamb, IBM Software Group, WPLC, STSM, WPLC Componentization Architect, IBM, Durham, NC, USA
- ▶ Walter Haenel, IBM Software Group, WPLC, Portal Architect for Deployment and Operations, IBM, Boeblingen, Germany
- ▶ Brett Gordon, IBM Software Group, WPLC, WebSphere Portal L2 Security/Administration Support
- ▶ Geir Sjurseth, Ascendant Technology for technical review

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

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

Introduction

IBM WebSphere Portal Version 6.0 is an enterprise portal solution that provides the framework and services to deliver a single point of entry to corporate applications, content, business processes and people.

This single point of entry provides a unified user experience and can be tailored and personalized for specific individuals or business units.

Building on previous versions, WebSphere Portal 6.0 provides the following:

- ▶ Enhanced Web content management, workflow, electronic forms integration, and collaboration to help deliver improved operational efficiency and productivity.
- ▶ Powerful new tools such as application templates and workflow, enabling organizations to rapidly model business processes and services in to reusable applications. These tools greatly accelerate application and content deployment, innovatively delivering on the promise of a service oriented architecture (SOA).
- ▶ A responsive and reliable portal platform from a leader in the enterprise portal market.

This chapter introduces the key new features and improvements in WebSphere Portal Version 6. It is divided in to three main sections and provides a starting point to understand the major new capabilities in WebSphere Portal Version 6, and how you can leverage them.

- ▶ “What is new in WebSphere Portal Version 6” on page 3 provides a general overview of what is new in WebSphere Portal Version 6.
- ▶ “WebSphere Portal programming model” on page 12 takes a brief look at some of the changes in the WebSphere Portal Version 6 programming model.
- ▶ “Operations and deployment” on page 15 of this chapter is concerned with the changes to operations and deployment and serves as an introduction to the more detailed chapters in this book.

A number of these new features have a strong bearing on best practices for deploying WebSphere Portal. Where appropriate, we reference the other chapters in this book that discuss those features in more detail.

Note: While the focus of the features discussed in this chapter are those that pertain to deployment of WebSphere Portal 6, much of the underlying framework for this chapter is based upon the article *What's new in WebSphere Portal Version 6?*

http://www-128.ibm.com/developerworks/websphere/library/techarticles/0607_hepper/0607_hepper.html.

Accordingly, we wish to acknowledge the efforts of the authors of this article:

- ▶ Stefan Hepper (sthepper@de.ibm.com), WebSphere Portal Programming Model Architect, IBM
- ▶ Stefan Liesche (liesche@de.ibm.com), WebSphere Portal and Workplace Foundation Lead Architect, IBM
- ▶ Gregory Melahn (melahn@us.ibm.com), Senior Technical Staff Member, Workplace Content Architect, IBM
- ▶ Thomas Stober (tstober@de.ibm.com), Software Architect, IBM WebSphere Portal Development, IBM

1.1 What is new in WebSphere Portal Version 6

Before delving deeper into specific new features and capabilities, the following section highlights how WebSphere Portal Version 6 fits within the greater scope of Service Oriented Architecture (SOA).

1.1.1 WebSphere Portal and SOA

Service Oriented Architecture or SOA, is not a product or solution. Rather it is an architecture principle that looks at reusable business functions and processes, and how these building blocks can be assembled into usable applications.

Following are the benefits of an SOA approach:

- ▶ It allows businesses to *reuse and combine* new and existing functions.
- ▶ It enables business processes to be more flexible and responsive in how they deliver business solutions.

WebSphere Portal Server Version 6 is key to an SOA approach because it provides a common framework and interface to integrate information, people, and business processes.

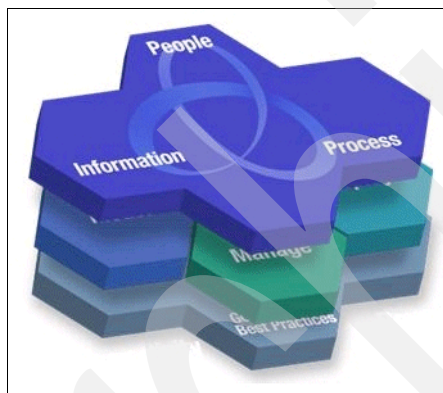


Figure 1-1 SOA integrates information and people as well as business processes

Essentially a portal is a series of different applications and services—some of which come with the portal itself (for example, content management, search, document management and so on), whilst other applications may already exist in your business environment.

These applications are then combined and assembled (together with any relevant business processes) at the *front-end* via portlets, thus making Portal into what is essentially a giant “composite” application, as shown in Figure 1-2 on page 4.



Figure 1-2 WebSphere Portal 6 as the “Front End” to SOA

In addition to the common interface, application, and content integration framework provided by WebSphere Portal Version 6, business users can also derive with the following significant benefits:

- ▶ Different access modes to applications and services.
- ▶ Roles and processes based on access that can be used to vary what applications and services a user can see.
- ▶ The ability for business users to create and extend portal applications without the need for any programming.
- ▶ Application templates that allow business users to easily assemble and customize linked applications and processes into composite applications.

These features are discussed in more detail in the following sections.

1.1.2 Simplified portlet creation

The portal-based interface provides a personalized experience, which considers the user's identity, role, and personal preferences. WebSphere Portlet Factory and WebSphere Portlet Factory Designer, while not new as products, are now part of the WebSphere Portal offerings. Run time and designer licenses are included in all three editions of WebSphere Portal Server.

WebSphere Portlet Factory is a comprehensive portlet development environment that automates the process of creating, deploying, and maintaining SOA-based portlets. Non-programmer employees can use WebSphere Portlet Factory to quickly create portlets that access your company's existing application. These portlets can then be assembled as building blocks into composite applications.

1.1.3 Workflow support

Portals bring together vital information, application, and content. They also bring together processes and enable people to interact with them in interesting ways. For example, people interact with processes and with each other. People are important elements in any process to keep things running smoothly, namely to manage issues that pop up or to resolve problems. People-centric collaboration is about enabling human and process interaction with consistent levels of service.

WebSphere Portal Version 6 therefore includes a workflow builder as a technical preview that enables business users to create and modify departmental workflows which, in a non-portal environment, would typically be implemented as e-mail flows.

1.1.4 Interactive forms in portal interfaces

Many interactions today involve the creation or completion of forms. When you arrive at an international airport, check into a hotel, provide conference feedback, and in many other situations, you are required to fill out forms, which businesses and agencies use to collect and structure data. Forms are a pervasive information exchange vehicle. You can use the sophisticated electronic forms capabilities of IBM Workplace Forms™ with WebSphere Portal V6.0. You can use these two products together to do the following:

- ▶ Include electronic forms in a standard portal interface
- ▶ Enable users to easily access information from other applications
- ▶ Enable users to collaborate, create, edit, or view electronic forms

1.1.5 User interface enhancements

WebSphere Portal V6 has significantly enhanced its user experience. The look-and-feel was completely re-worked to implement a modern, state-of-the-art UI design.

It includes a drop-down menu, which provides access to the main functional areas of WebSphere Portal, such as administration, content management, templating, and so on. Menus offer the choices that are available within the context, and they make using WebSphere Portal very intuitive.

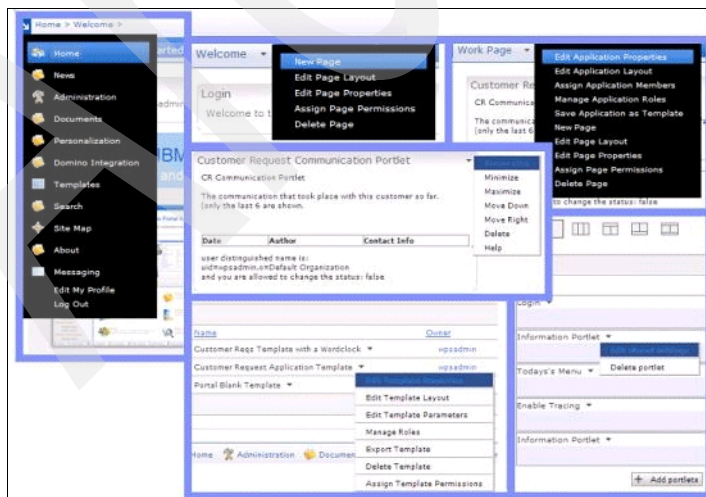


Figure 1-3 The main drop-down menu and contextual menus

The new Portlet Palette simplifies the creation and design of portal pages. The palette lists components that you can drag and drop directly onto a page. Using the palette simplifies the task of arranging portlets on a page, and makes the design experience much more intuitive.

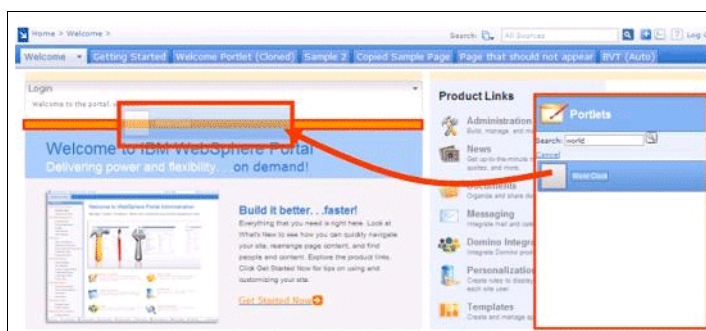


Figure 1-4 Drag and Drop: Adding portlets to a page using the Portlet Palette.

Behind the scenes, WebSphere Portal V6 leverages the Web development language Ajax (Asynchronous JavaScript™ and XML) in several places to move UI logic from the server into the browser. For example, contextual menus are implemented based on Ajax. The appropriate choice of menu options is determined on the browser system without request or response round trips to the portal server. WebSphere Portlet Factory uses Ajax to implement a *type-ahead* capability and to refresh UI fragments within a page.

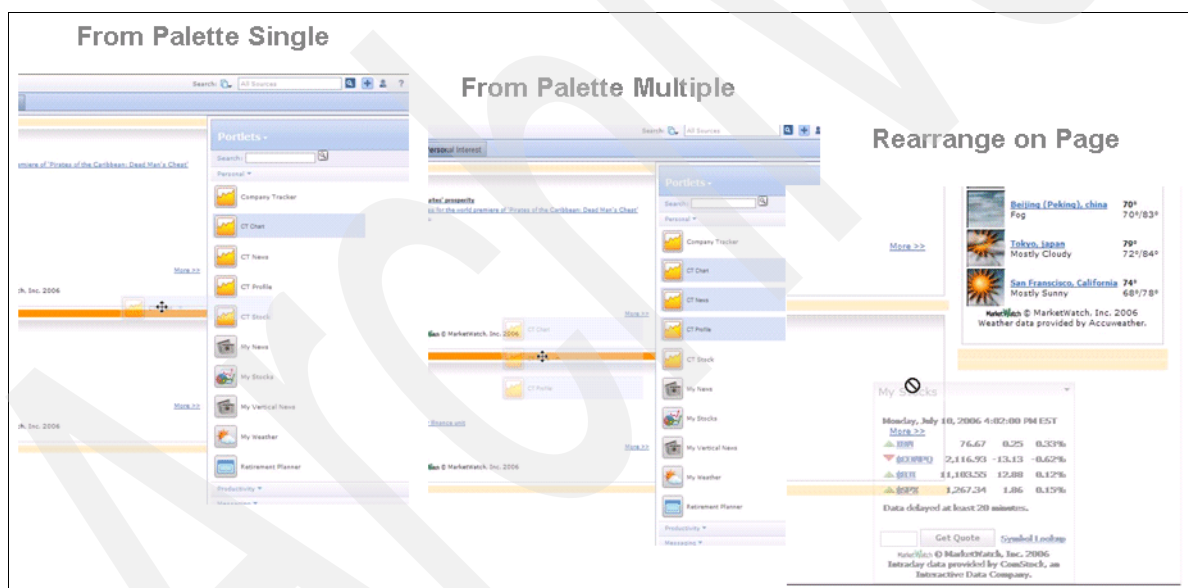


Figure 1-5 Drag, drop, and rearrange portlets

1.1.6 Composite applications and templates

While composing applications out of components is more efficient than creating them from scratch as monolithic applications, building complex business logic using a set of portlets can be pretty tedious. You need to do the following:

1. Deploy the individual components one after each other.
2. Arrange the deployed pieces on the staging system, as desired.
3. Define portlet interaction and access control according to the business logic to be implemented.

All these steps need active involvement of application developers, portal administrators, and the people with the necessary business domain skills.

To simplify this process, WebSphere Portal V6 introduces composite applications. Business analysts and application designers can easily assemble composite applications that implement complex business logic from individual components, such as portlets, processes, or other code artifacts.

Composite applications are a key tool in implementing meaningful business value within a SOA. By empowering users to define, create, and manage their own composite applications, WebSphere Portal V6 helps facilitate a strong business-driven usage model with fewer dependencies on the support by system administrators.

Composite applications use two fundamental aspects: templates and applications.

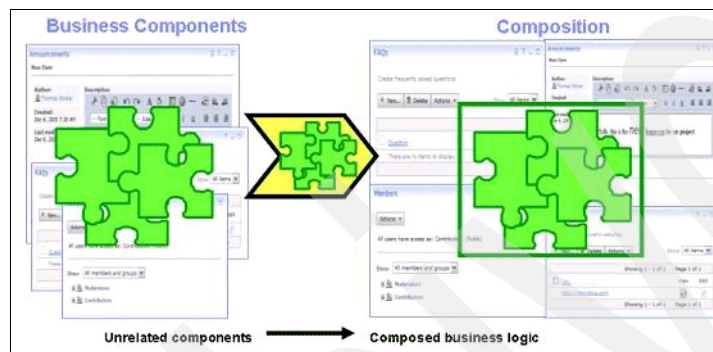


Figure 1-6 Composing business logic from components

Templates

A template describes a composite application in an abstract way, including information that defines how complex business logic is assembled out of a given set of components. The template is an XML file that references all components, such as portlets or Java™ code artifacts and specifies applicable meta information, such as specific configuration settings for each individual component. The template describes the composed application behavior by defining the desired interaction between the components, such as wires between portlets and access control logic to be enforced, such as application specific user roles.

The people who understand the business logic create the templates. WebSphere Portal provides tooling to make this task as simple as possible. The Template Builder is a portlet and is very similar to the page customizer.

Once created, templates are stored in a template library and are made available to be consumed by the user community. Again, templates are XML files that represent the abstract definition of a composite application.

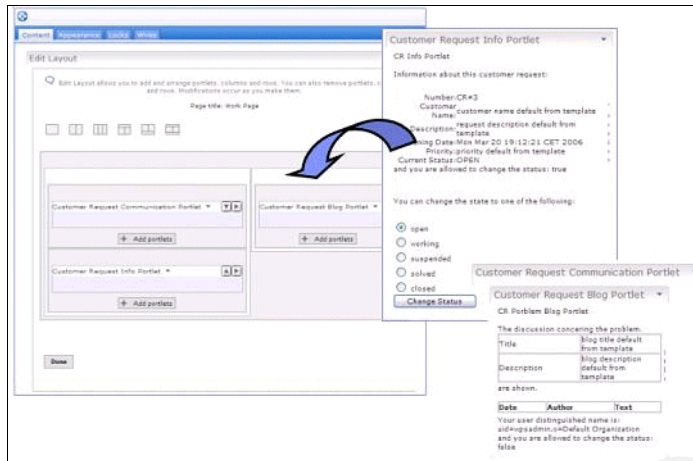


Figure 1-7 Creating a composite application with the Template Builder

Applications

Because templates are stored in a template library, you can pick a template and create a new instance of the composite application, which is described by the selected template definition. You can manage your application instances based on your own needs.

A simple example of templates and applications

Here is an example to show how templates and applications can be used.

A development team designs a company specific teamroom application. First, the team makes sure that they have all the components they need to assemble the intended functionality. In most cases, they simply pick portlets and logic from a catalog. In some cases, they implement a component from scratch. Once the components are identified, the application designer assembles the teamroom application by adding and arranging components as desired. The result is stored as an XML file in the template catalog and it is made available to all registered portal users within the company.

In our example, a project manager creates his individual teamroom instance for his project and uses this instance together with his colleague's. From the common template definition, there would soon be a growing number of teamrooms within the company.

Points of variability (parameters)

It is not realistic to assume that each instance of the same template is supposed to behave exactly the same. Typically, you want each application to have a unique configuration in certain places. In our teamroom example, the specific title of a teamroom needs to be set. WebSphere Portal V6 lets you define parameters in the template, wherever a point of variability is needed. During instantiation, the creator of the application completes the specific value for each parameter, and these values are applied to that particular instance only.

In our example, the developer of the teamroom template specifies the teamroom title as a parameter. Later, each project manager enters a meaningful title when he or she creates a specific teamroom instance. This concept of parameterization adds flexibility to the usage of your predefined templates.

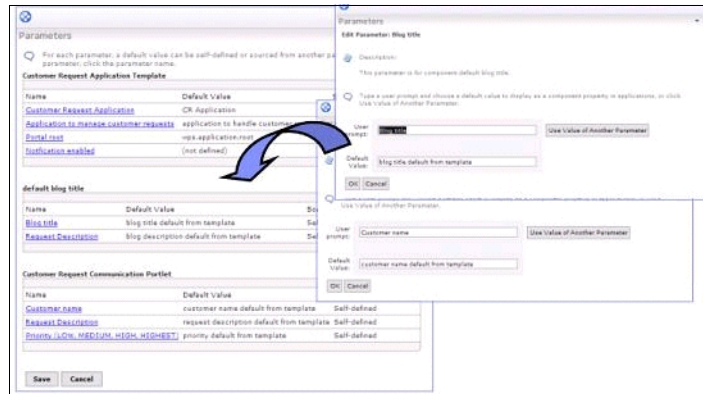


Figure 1-8 Leveraging points of variability in a template

Application roles

You can define access control settings through templates very efficiently. The template developer specifies roles, which are meaningful in the context of this particular composite application.

For example, a teamroom template has the roles “project lead”, “project member”, “guest”, and “administrator”. Each of these application roles aggregates a set of very specific portal access control roles. A project member might have editor access to the FAQ portlet as well as manager access to the document repository, while a guest would only have user access to the FAQ portlet and no access at all to the document repository. You can hide the complexity of many access control roles to individual components and expose them as few simplified application level roles with well-understood and easy to use names.

Membership

After a template is instantiated, the resulting application can be used by a community of users. The owner of the application adds members to the community as needed, and assigns each new member to the predefined application level roles. End-users can perform the user administration of their applications themselves.

Workflow

You can use workflow capabilities in your composite applications. New in WebSphere Portal V6 is the Workflow Builder portlet, which enables users (such as business analysts or application designers) to define processes. The **create process** definitions are treated as components within a template and can be used in the context of composite applications. For example, you can add a Workflow called “Document review process” to the Teamroom sample application.

Important: The Workflow Builder is a Tech Preview in WebSphere Portal V6.0.

1.1.7 Personalizing the Portal

WebSphere Portal lets you personalize your portal for different groups of users, and you can enable administration based on attributes. Originally, portal servers let you customize the content that the user sees based on the user's role. For example, a user in the manager group has access to the portlet displaying the salaries of his or her employees. Now in WebSphere Portal V6, you can define rules to modify the content the user sees based on the current request and set of rules that apply to this request.

WebSphere Portal V6 enables administrators to provide content based on specific attributes or rules so that each user's experience can be unique. Specifically, you can do the following:

- ▶ Filter content using rules, based on meta information attached to a user. For example, provide different content to a gold customer versus a standard customer.
- ▶ Use attribute-based administration to define visibility rules that show or hide content based on user meta data. For example, display a specific page or portlet each Monday that shows the goals for this week.
- ▶ Set policies and define specific properties that can be queried to influence the behavior of a portal resource or the user experience. For example, set the mail size quota.

You can also combine these different concepts. That is, if a user is a gold customer, you could make that customer's mail size quota larger than the size for standard users.

Figure 1-9 shows the Edit Page with Show Portlet Rule Manager enabled. Specifically, notice the upper-right side, which indicates the opposite toggle option available, Hide Portlet Rule Manager. You can now define a specific rule for each portlet that influences the visibility of that portlet.



Figure 1-9 Creating a new rule for a specific portlet

If you click the Create New Rule button, the Personalization Picker portlet displays, as shown in Figure 1-10. Then, you can enter a specific rule.

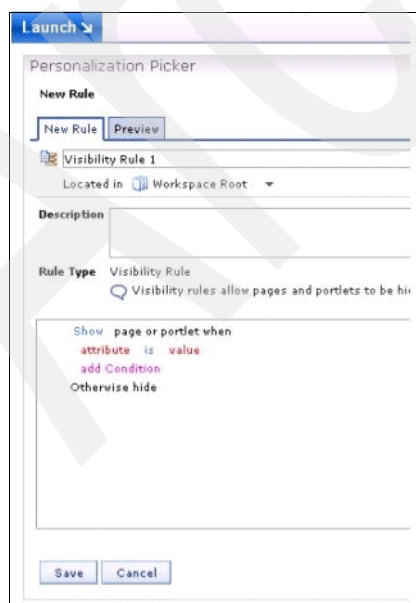


Figure 1-10 Creating a new rule with the Personalization Picker

In the default rule, click any of the words that are not displayed in black, and a wizard opens, as shown in Figure 1-11.

Next, you can select attributes to be evaluated for a specific rule. In the example shown in Figure 1-11, we want to evaluate one attribute of the user management system; therefore, we select Portal Users. A pop-up window displays, listing the available user-management properties. We select `ibm-hobby` from this list.

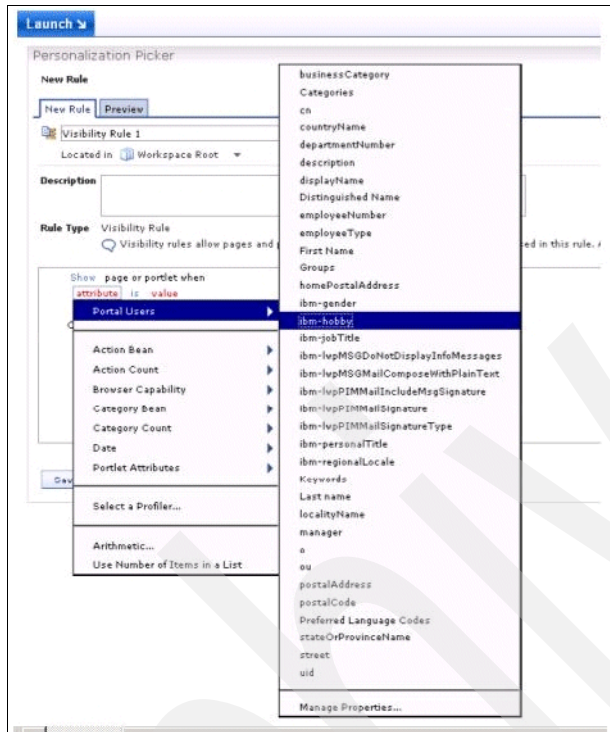


Figure 1-11 Selecting an attribute for the rule

We add a specific value (see Figure 1-11) that the selected attribute must have in order to run the rule. For this example, we require that the current user's hobby be sports in order to display this portlet to a user.

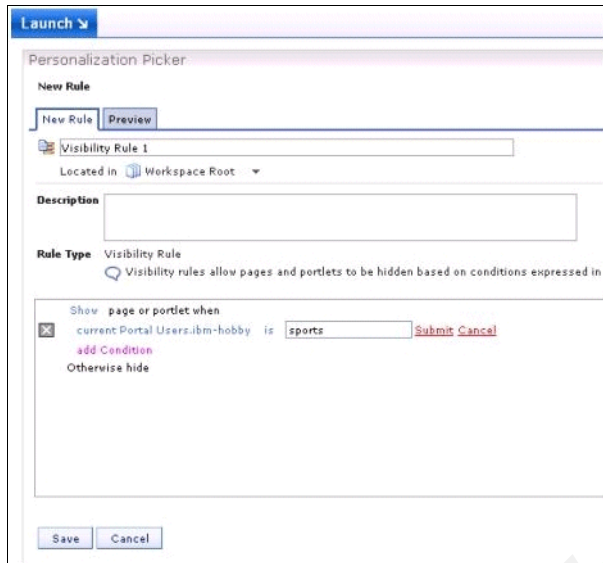


Figure 1-12 Adding a value the selected attribute must have to apply the current role

Figure 1-13 shows the final rule, which we save to enable it for this portlet. You can add more conditions with the addCondition button, or you can reverse the rule and hide the portlet if the rule evaluates to true.



Figure 1-13 Final rule. Only display the portlet if the user's hobby is "sports"

1.2 WebSphere Portal programming model

The WebSphere Portal programming model is an extension of the J2EE™ programming model. You use it to implement Web applications that leverage the rich set of WebSphere Portal platform capabilities. These include aggregation and integration of components into page hierarchies, flexible navigation, content and application aggregation, branding,

customization, personalization, content management, document management, search, and so on.

The WebSphere Portal programming model consists of various APIs, SPIs, JSP™ Taglibs, Eclipse plug-points, and descriptors that let you customize the aggregation steps at different levels.

The base model

The base WebSphere Portal programming model was introduced in WebSphere Portal V5.1 which, in Version 6, was extended but not fundamentally changed. A series of articles on the WebSphere Portal V5.1.0.1 Programming Model covers the following topics:

- ▶ Part 1: Introducing the WebSphere Portal Programming model gives you an introduction to the different parts of the model and how they fit together.

http://www-128.ibm.com/developerworks/websphere/library/techarticles/0512_hepper/0512_hepper.html

- ▶ Part 2. Advanced URL Generation tells how to create URLs pointing to different pages and portlets.

http://www-128.ibm.com/developerworks/websphere/library/techarticles/0603_behl/0603_behl.html

- ▶ Part 3. Integrating WebSphere Portal into your security environment and user management system shows how to create a single sign-on (SSO) infrastructure, and how to integrate WebSphere Portal into your existing user management system.

http://www-128.ibm.com/developerworks/websphere/library/techarticles/0606_buehler/0606_buehler.html

- ▶ Part 4. Making your portlets dynamic and context sensitive (to be published soon) describes how to create dynamic pages and portlets, and to leverage additional meta data to make your pages and portlets more context sensitive.

http://www-128.ibm.com/developerworks/websphere/library/techarticles/0608_engelhausen/0608_engelhausen.html

The information provided in these articles still applies to WebSphere Portal V6.0, and you can use and run the samples from these articles in WebSphere Portal V6.0. The next section describes the new features that were added to the WebSphere Portal programming model in Version 6.

New in the V6.0 programming model

Let us take a look at things that WebSphere Portal V6.0 adds on top of the base programming model. We already discussed some enhancements in the previous sections, such as composite applications, personalization, and rules.

WebSphere Portal V6.0 programming model new features include the following:

- ▶ Composite applications

Create business applications and templates as previously described.

- ▶ Themes and skins extension points

You can use defined extension points to customize the default themes and skins. You can separate your customization code from the default themes and skins code. Then, you can update the default themes and skins code easily, without affecting your customizations.

- ▶ Drag-and-drop
Define your own drag sources and drop targets to leverage the new drag-and-drop functionality in your portlets, themes, and skins.
- ▶ Policies
Create your own policies and plug into the policy and personalization system.
- ▶ Support for the Edit Default mode
Administrators and business users can distinguish between default settings they want to set on a portlet that is shared among users, and the private settings that they can set in Edit mode.
- ▶ Advanced Portlet URL Generation capabilities
Use the URL Generation SPIs introduced for themes and skins in V5.1.0.1 in portlets to create URLs pointing to other pages and portlets. WebSphere Portal V6 also provides a simplified version of this SPI as URL Generation API that is easier to use for the most common cases.
- ▶ Additional portal model and state SPIs
Get aggregated meta data, and also get and set the current page locale.
- ▶ Search
Use a common search API for the various search engines available from IBM.
- ▶ Workflow
Create your own data objects that are processed through the workflow builder tool.

We describe these new features in more detail in a future article series on the WebSphere Portal V 6.0 programming model. Watch the WebSphere Portal zone for this series.

WebSphere Portal V 6.0 provides additional extensions to APIs that are available already in WebSphere Portal V5.1.0.1, such as the Credential Vault API, Task Processing API, Web Content Management API, and Puma SPI. These additions are explained in the WebSphere Portal Information Center.

Another thing to note in WebSphere Portal V6.0 is that the IBM Portlet API is now deprecated in order to show the IBM commitment to the JSR 168 standard Portlet API. Also, many of the new functions are only available to standard portlets.

The IBM Portlet API is still supported in this and at least the next release. However, if you are going to touch one of your IBM Portlets anyway, we recommend that you port it to the standard API.

For more information about how to migrate your IBM portlet to the standard Portlet API, see *Comparing the JSR 168 Java Portlet Specification with the IBM Portlet API* at the following Web address:

http://www-128.ibm.com/developerworks/websphere/library/techarticles/0312_hepper/hepper.html

Also see *Converting the WorldClock portlet from the IBM Portlet API to the JSR 168 portlet API*, at the following Web address.

http://www-128.ibm.com/developerworks/websphere/library/techarticles/0412_paeffgen/0412_paeffgen.html

1.2.1 Content and Web content management

There have been significant improvements in the IBM Workplace Web Content Management™ component that is included with a restricted license in the Enable and Extend editions in WebSphere Portal V6.

In keeping with the IBM commitment to open standards, the Web Content Manager was re-based on the Java Content Repository (JCR) included with WebSphere Portal. This repository is based on the JSR 170 standard. It is the same repository used for Portal Documents and enables the Web Content Management component to provide greater scalability and performance than in earlier releases. Having content in the JCR also makes it easier to personalize the content using business rules created with WebSphere Portal's Personalization component.

Web Content Management administration is simpler because its nodes can now share the same repository, allowing full clustering support in both production and authoring environments. The use of WebSphere logging and caching services and WebSphere Portal access control management also help to simplify administration.

The JCR provides a way to better organize content into content libraries. This capability was already used by Portal Document Manager and is now used by Web Content Manager. Libraries help you to better manage Web content, making it easier to control access rights, separate test content from production content, organize multi-language content, and share content between sites.

The user experience for content authors was simplified and enhanced. New data types such as links, numbers and dates, searching, custom help, limit checking, improved navigation, views and better rich text editing were added. You can now easily author live content directly within a site by employing a new in-line authoring component.

Both content authors and users can more easily search content. Portal search collections can include Web content, and you can include a new search component into sites designed with Web Content Management.

WebSphere Portal Document Manager now lets you access documents directly from your Windows® Desktop or from Microsoft® Office products. Document libraries appear as network places in the Windows Explorer, giving you easy access to document management functions such as locking, versioning, and editing. The performance of the Portal Document Manager was improved to save document renditions and to streamline document preview.

1.3 Operations and deployment

WebSphere Portal Version 6 provides a number of significant enhancements designed to make user and database management more flexible.

These enhancements also give organizations much greater capability in providing high availability 24x7 portal environments.

1.3.1 Multiple LDAP support

In previous versions of WebSphere Portal Server, we could only use one LDAP directory to store user account information.

Users contained in the LDAP directory could however be logically grouped in to one or more “realms” (sometimes referred to as “horizontal partitioning”), and these realms could then be

tied to specific portal configurations (referred to as “virtual” portals), serving up different content depending on which realm the authenticating user belonged to.

Figure 1-14 shows this concept of realms in action. Notice that despite only having one LDAP directory, we have three distinct groupings of users (“Acme” employees, “Suppliers” and “Customers”), each of whom could be served up content from a “virtual” portal associated with their realm.

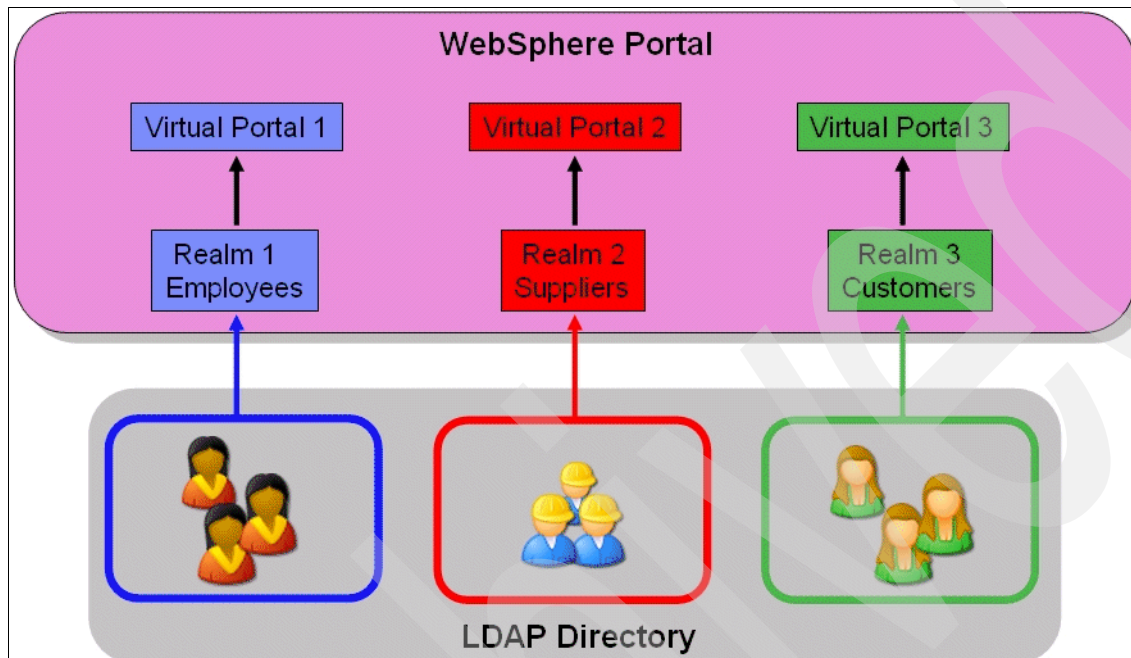


Figure 1-14 A single LDAP directory with multiple realms

A single LDAP directory does represent a potentially significant single point of failure, and in previous versions of WebSphere Portal Server, LDAP directory redundancy was provided by having multiple identical replicas of the LDAP directory in the environment.

Failover to one of these identical replicas had to be taken care of by a third-party load balancer product such as IBM Network Dispatcher.

WebSphere Portal Server Version 6 overcomes these limitations by doing the following:

- ▶ Extending the concept of realms to introduce support for multiple LDAP directories.
- ▶ Providing automatic failover to additional LDAP directory replicas without the need for a third-party load balancer.

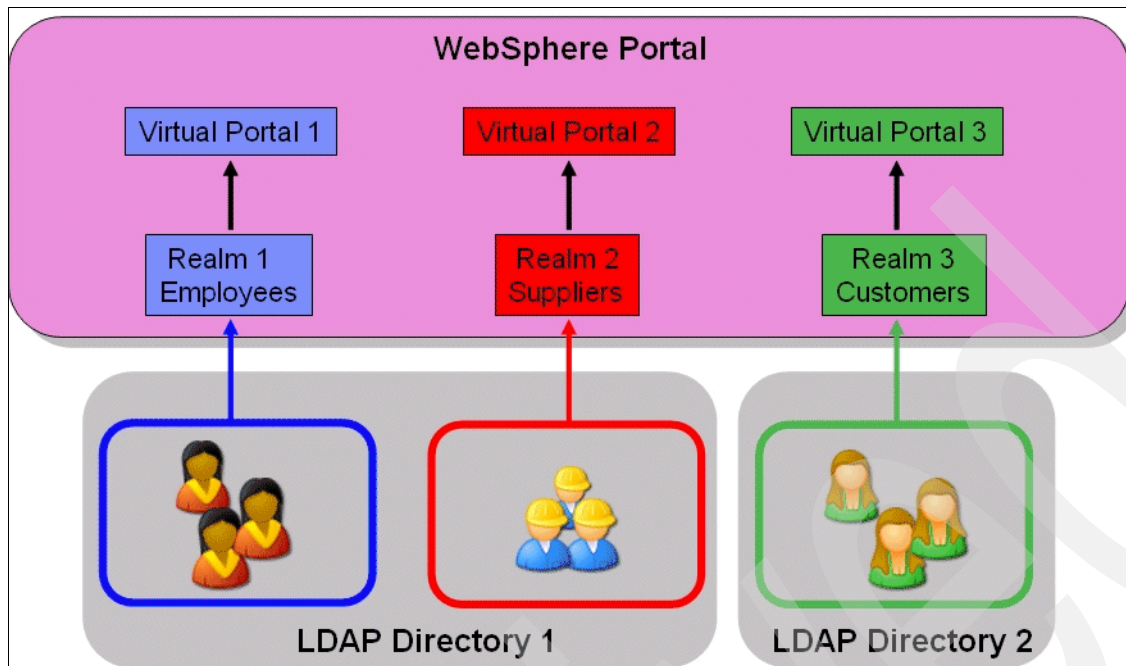


Figure 1-15 Multiple LDAP directories with multiple realms

Multiple LDAP directory support therefore reduces infrastructure investment and complexity as there is now less of a need for directory integration or for user consolidation approaches.

It also allows us greater flexibility when deploying a WebSphere Portal infrastructure inside and outside of firewalls, especially when using multiple LDAP directories in conjunction with the previously mentioned “virtual portal” capability.

Redundancy and failover can now be catered for much more simply with WebSphere Portal Server Version 6, by combining the strategy of previous versions of having multiple read-only LDAP replicas or “shadows”; however, now we no longer require any additional infrastructure components like an LDAP load balancer.

In the case that one LDAP directory becomes unavailable, Portal seamlessly connects to any of the other (identically) configured read-only LDAP directories that might be available.

The multiple LDAP features described in this section, together with steps on configuring multiple LDAP directories are covered in greater detail in Chapter 4.

1.3.2 Database domains

WebSphere Portal V6 provides enhanced flexibility in portal data management. The portal configuration repository now consists of independently manageable database or schema objects (database domains).

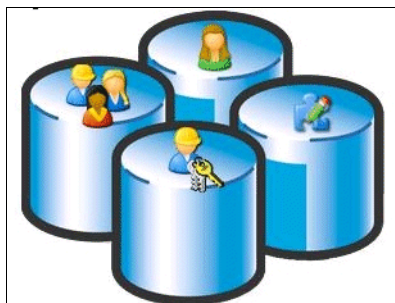


Figure 1-16 Database domains

By separating your data, you can share domains across multiple portals. With the shared data domains feature, operators can provide continuously available (24x7) portal sites with full customization support, while keeping operation procedures simple. Operators can create multiple, independent lines of production so they can change configurations, apply maintenance, and stage solution releases to one line at a time, while keeping other lines completely available to portal users.

Operators can also now spread different data domains across different database types. You can select the most effective database type for each domain to reduce operation cost and to provide the right level of availability for each domain.

The separation of portal data into multiple domains provides more options for operators to establish a distributed portal operation. With separation of data at the database level, the replication of data becomes easier. With database replication per data domain (using database provided replication mechanisms), operators can create globally distributed portal sites that are individually managed and operated. Each portal site will have dedicated release data domains and shared user customization data domains.

Releases can be staged into sites independently of other sites (for example, using XMLAccess). Staging lets you make identical portal solutions available on multiple sites. While multiple sites within the same location typically might share the same database server for customizations, in a global deployment with higher location independence requirements, user customizations can be hosted locally in each site with two-way replication between the sites. Users can access their customized portal site independently of the site that services their requests.

Database replication scripts are not part of the WebSphere Portal offerings. You need to create scripts to match specific infrastructure needs. You can also consider using database replication mechanisms provided by database vendors.

In WebSphere Portal V5.1 all data is part of a single DB repository. But the type of data existed already it was simply not split into domains. In WebSphere Portal V6, the portal repository consists of the following data domains: Release, Community, Customization, Feedback, LikeMinds, JCR, and Member Manager.

Release data are typically not modified during production (such as administrator defined pages, portlets, and portlet instances). Typically administrators create release data on an integration server and stage it to the production system. Community data are typically modified during production (such as shared documents or application resource). Documents and Web content are stored in the JCR. Customization data is associated with a particular user only. A Typical example of customization data is portlet data. Feedback and LikeMinds data are created and used during productive use of the portal by the Portal Personalization Runtime. Member Manager Data contains user profile data, for example look aside user profile data. More information about data domains and data domain administration is

provided in the WebSphere Portal V6 Infocenter, Transferring individual domains section. (See Appendix , “Related publications” on page 453 for more information).

Portal Configuration Management enhancements in WebSphere Portal V6 help operators become more efficient in managing the portal infrastructure. Operators can now configure WebSphere Application Server and WebSphere Portal Server from a single console or command line. You can display and change configurations for stand-alone portal servers as well as entire portal clusters consisting of multiple nodes. You can set runtime configurations and use graphical wizards for problem determination, and then view your results in a single place. See runtime trace setting options in Figure 1-17.

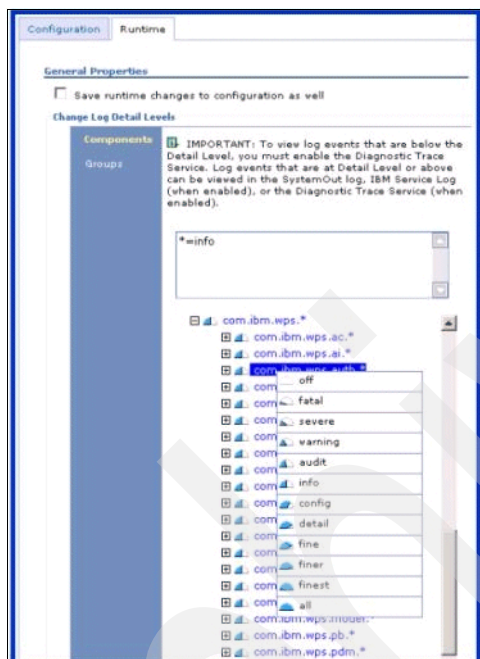


Figure 1-17 Configuring portal servers in a cluster

Archived

Planning for your Portal 6.0 high-availability deployment

This chapter provides the planning information for successfully building a Portal 6.0 High Availability Cluster, including System Requirements, Typical Deployment Scenarios, Clustering Considerations, Content Management Considerations, and an introduction to the Deployment Scenarios used in this book.

In this chapter, we discuss the following topics:

- ▶ Software pre-requisites
- ▶ Overview of typical deployment scenarios
- ▶ Architectural considerations / Clustering Considerations
- ▶ Content management considerations (pzn, pdm, wcm)
- ▶ Recommendations and best practices: how many environments do I need?
- ▶ Introduction to the deployment scenarios used in this book

2.1 System requirements

This section is an overview of the hardware, software and network connectivity requirements for running IBM WebSphere Portal 6 on various supported operating systems. A detailed list of required software and hardware fixes is provided in the Portal Information Center.

Use the following Web address for the IBM WebSphere Portal, Version 6.0 Information CenterLink:

<http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp>

Important: Please check the Portal 6 Information Center for the latest software or hardware fix requirements as this information may be updated periodically in the information center to reflect new supported configurations.

2.1.1 Hardware requirements

This section provides data for hardware configurations that IBM has tested. Use the following information as a guide for your installation of WebSphere Portal.

IBM AIX® systems

Processor:

- ▶ RS/6000® at 450 MHz at a minimum. Production environments should consider higher speeds.
- ▶ Power5

Physical memory:

- ▶ 4G is an optimal starting point for RAM in a production environment.

Disk space:

- ▶ Install directory: 3.5 GB
- ▶ Temp directory: 1.5 GB

Virtual memory/swap space:

- ▶ This value should be equal to double your physical memory.

HP-UX systems

Processor:

- ▶ HP 9000 on PA-RISC at 440 MHz or faster

Physical memory:

- ▶ 4G is an optimal starting point for RAM in a production environment.

Disk space:

- ▶ Install directory: 3.5 GB
- ▶ Temp directory: 1.5 GB

Virtual memory/swap space:

- ▶ This value should be equal to double your physical memory.

Linux® Intel® systems

Processor:

- ▶ Production environments should consider the Pentium® 4 processor at 1.4GHz or higher.

Physical memory:

- ▶ 4G is an optimal starting point for RAM in a production environment.

Disk space:

- ▶ Install directory: 3.9 GB
- ▶ Temp directory: 1.5 GB

Virtual memory/swap space:

- ▶ This value should be equal to double your physical memory.

Linux on POWER™

Processor:

- ▶ POWER is a term used to refer to IBM system product lines, including the following:
 - iSeries™ models that support LPAR (64-bit support only) with minimum of 450 CPW in Linux partition
 - pSeries® models that support Linux (64-bit support only)
 - BladeCenter® JS20
 - OpenPower™

They are based on IBM Power Architecture™ technology and run the same Linux distributions from Red Hat and Novell SUSE.

- ▶ Power5

Physical memory:

- ▶ 4 GB is an optimal starting point for RAM in a production environment.

Disk space:

- ▶ Install directory: 3.9 GB
- ▶ Temp directory: 1.5 GB

Virtual memory/swap space:

- ▶ This value should be equal to double your physical memory.

Important: For any linux operating system, the libstdc++-3.3.3-41 shared object library is required prior to installation of WebSphere Portal.

Linux on System z™

Processor:

- ▶ z800, z890, z900, z990, z9™

Physical memory:

- ▶ Under minimal load, WebSphere Portal can function with 2GB of RAM. However, 4GB is an optimal starting point for RAM in a production environment.

Disk space:

- ▶ Install directory: 3.9 GB
- ▶ Temp directory: 1.5 GB

Virtual memory/swap space:

- ▶ Swap space on z/Linux is recommended to be 15% of the virtual machine size.

Sun™ Solaris™ systems**Processor:**

- ▶ Sun Blade™ 2000 workstation at 1 GHz or higher is recommended.

Physical memory:

- ▶ 4G is an optimal starting point for RAM in a production environment.

Disk space:

- ▶ Install directory: 3.2 GB
- ▶ Temp directory: 1.5 GB

Virtual memory/swap space:

- ▶ This value should be equal to double your physical memory.
- ▶ For Solaris 9, swap space must be double your physical memory, especially if your physical memory is 1024 MB per processor.

Microsoft Windows systems**Processor:**

- ▶ Production environments should consider the Pentium 4 processor at 1.4 GHz or higher.

Physical memory:

- ▶ 4G is an optimal starting point for RAM in a production environment.

Disk space:

- ▶ Install directory: 3.9 GB
- ▶ Temp directory: 1.5 GB

Virtual memory/swap space:

- ▶ This value should be equal to double your physical memory.

File system:

- ▶ NTFS file system is recommended.

2.1.2 Software requirements

This section provides the minimum product levels that you should install for WebSphere Portal.

Supported operating systems (required on portal machine)

- ▶ Win XP SP1, SP2 - development
- ▶ Win 2000 Server SP4 and Advanced Server SP4

- ▶ Win 2003 Standard and Enterprise SP1
- ▶ Red Hat Enterprise Linux ES, AS, WS, Desktop for (x86) 3.0 U6 & 4.0 (WS & Desktop are only supported as developer platforms)
- ▶ Red Hat Enterprise Linux AS for (Power) 3.0 U6 & 4.0
- ▶ Red Hat Enterprise Linux AS for (zSeries®) 3.0 U6 & 4.0 U1
- ▶ SuSE, SuSE SLES 9, 2.6 Kernel (x86) SP1 (SuSE is only supported as a developer platform)
- ▶ SuSE SLES 9, 2.6 Kernel (Power) SP1
- ▶ SuSE SLES 9, 2.6 Kernel (zSeries) SP1
- ▶ AIX 5.2 ML05
- ▶ AIX 5.3 5300-02 Maintenance Level
- ▶ Solaris 9 with the Recommended Patch Cluster of November 2005 (or later)
- ▶ Solaris 10
- ▶ HP-UX - V11iV1, V11iV2 (V11iV2 requires a paper certification for V11iV2 support based on V11iV1 testing)

Supported application servers (required on portal machine)

- ▶ WebSphere Application Server Version 6.0.2.9 or later Base + WebSphere Process Server. The install images of the required Version 6.0.2.9 interim fixes can be found on the WebSphere Portal Setup CD.
- ▶ WebSphere Application Server Version 6.0.2.9 or later Base or WebSphere Application Server Version 6.0.2.9 or later Network Deployment + WebSphere Process Server Version 6.0.1.1.
- ▶ WebSphere Application Server Version 6.0.2.9 or later Base + Certain components of WebSphere Process Server Version 6.0.1.1.

Supported Web servers (optional)

- ▶ Apache Server 2.0.49 & 2.0.52
 - Includes CERT Advisory CA-2002-17.
- ▶ + IBM HTTP Server 2.0.47.1
 - The fix PK13784 is also required.
- ▶ IBM HTTP Server 6.0, 6.0.1, 6.0.2
 - The fix PK13784 is also required.
- ▶ Microsoft Internet Information Server 6.0
 - Supported on Microsoft Windows Server® 2003.
- ▶ + IBM Lotus Domino (as Web server) 7.0
- ▶ IBM Lotus Domino (as Web server) 6.5.5
- ▶ IBM Lotus Domino (as Web server) 6.5.4
- ▶ Sun Java™ System Web Server, Enterprise Edition 6.1 SP3
- ▶ Sun Java System Web Server, Enterprise Edition 6.0 SP9

Supported databases (required)

- ▶ + IBM Cloudscape™ Version 5.1.60.41
- ▶ + IBM DB2® Universal Database™ Enterprise Server Edition 8.2 FP5

- ▶ IBM DB2 Universal Database Enterprise Server Edition 8.1 FP12
- ▶ + IBM DB2 Universal Database Workgroup Server Edition 8.2 FP5
- ▶ IBM DB2 Universal Database Workgroup Server Edition 8.1 FP12
- ▶ Oracle Enterprise Edition 10g Release 1 and Release 2
- ▶ Oracle Standard Edition 10g Release 1 and Release 2
- ▶ Oracle Enterprise Edition 9i Release 2
- ▶ Oracle Standard Edition 9i Release 2
- ▶ Microsoft SQL Server Enterprise Edition 2000 SP4

Supported LDAP directories (optional)

- ▶ IBM SecureWay® Security Server for z/OS® and OS/390® Versions 1.4, 1.5, 1.6, 1.7, or 1.8
- ▶ IBM Tivoli Directory Server V6.0
- ▶ + IBM Tivoli Directory Server V5.2
- ▶ + IBM Lotus Domino 7.0.1
- ▶ + IBM Lotus Domino 6.5.5l
- ▶ +IBM Lotus Domino 6.5.4
- ▶ Novell eDirectory 8.7.3
- ▶ Sun Java System Directory Server V5.2
- ▶ Microsoft Active Directory® 2003
- ▶ Microsoft Active Directory 2000
- ▶ Microsoft Active Directory Application Mode (ADAM) 2003

Supported Web browsers (required)

- ▶ Microsoft Internet Explorer® V6.0 SP1 (SP2 for Windows XP)
- ▶ Microsoft Internet Explorer 6.0 SP1
- ▶ FireFox 1.5.0.3
- ▶ Mozilla Web Browser 1.7
- ▶ Netscape Communicator 8.1
- ▶ Apple Safari 2.0

Supported external security software (optional)

- ▶ IBM Tivoli Access Manager for e-business Version 6.0
- ▶ IBM Tivoli Access Manager for e-business Version 5.1
- ▶ Computer Associates eTrust SiteMinder 6.0
- ▶ Computer Associates eTrust SiteMinder 5.5

2.1.3 Network connectivity requirements

The following list provides network connectivity requirements for a Portal 6.0 High Availability Deployment:

- ▶ Network adapter and connection to a physical network that can carry IP packets, for example, Ethernet, token ring, ATM, and so on.

- ▶ Static IP address with an entry in DNS.
- ▶ Configured fully qualified host name. Portal 6.0 High Availability Deployment must be able to resolve an IP address from its fully qualified host name. To ensure that the host name is correctly configured in DNS, type one of the following commands at the command line of another server in the network:
 - Ping hostname.yourco.com
 - Windows: nslookup hostname.yourco.com
 - Linux: dig hostname.yourco.com

2.2 Typical deployment scenarios

This section focuses on typical Deployment Scenarios in WebSphere Portal 6 including a simple, single server topology, vertical scaling topology, horizontal scaling topology, and mixed horizontal and vertical scaling topology.

2.2.1 Single-server topology

In a single-server installation all of the components are installed on the same machine. This is the simplest Portal installation and is often used for a proof of concept or a demo machine. The emphasis of a single server installation is on getting it up and running quickly. At the end of the scenario you can build out the topology to include an external database or an LDAP user registry for user authentication or collaboration function as well as an external http server.

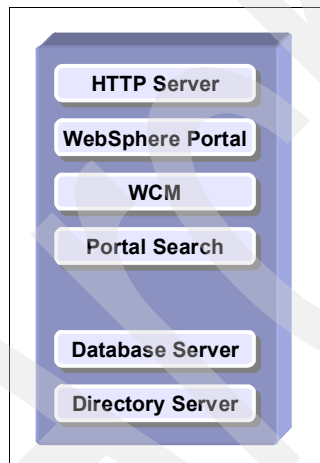


Figure 2-1 Single server installation

2.2.2 Vertical scaling topology

Vertical scaling topology includes multiple application servers running WebSphere Portal on a single physical machine. You can use vertical scaling to fully use a node within the conceptual node group in the case where resource congestions or locking conditions prevent a single application instance to scale up to the nodes limit. Please note that multiple java processes are instantiated when using vertical scaling. This is also the topology scenario that is described in detail and installed in Chapter 3.

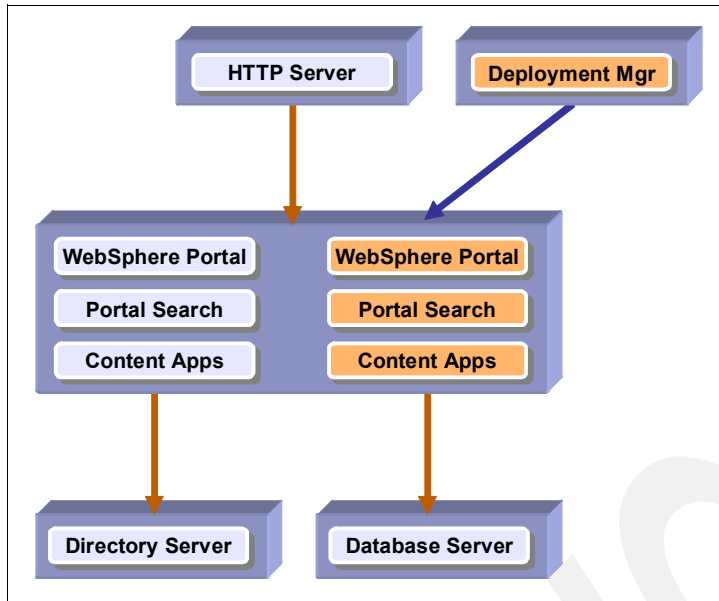


Figure 2-2 The vertical Portal cluster using WebSphere Application Server clustering capabilities.

2.2.3 Horizontal scaling topology

In a Horizontal scaling topology members of a WebSphere Portal cluster exist on multiple physical machines, effectively and efficiently distributing the workload of a single logical WebSphere Portal image. Clustering is most effective in environments that use horizontal scaling because of the ability to build in redundancy and failover, to easily add new horizontal cluster members to increase capacity, and to improve scalability by adding heterogeneous systems into the cluster. This is also the topology scenario that is described in detail and installed in Chapter 3.

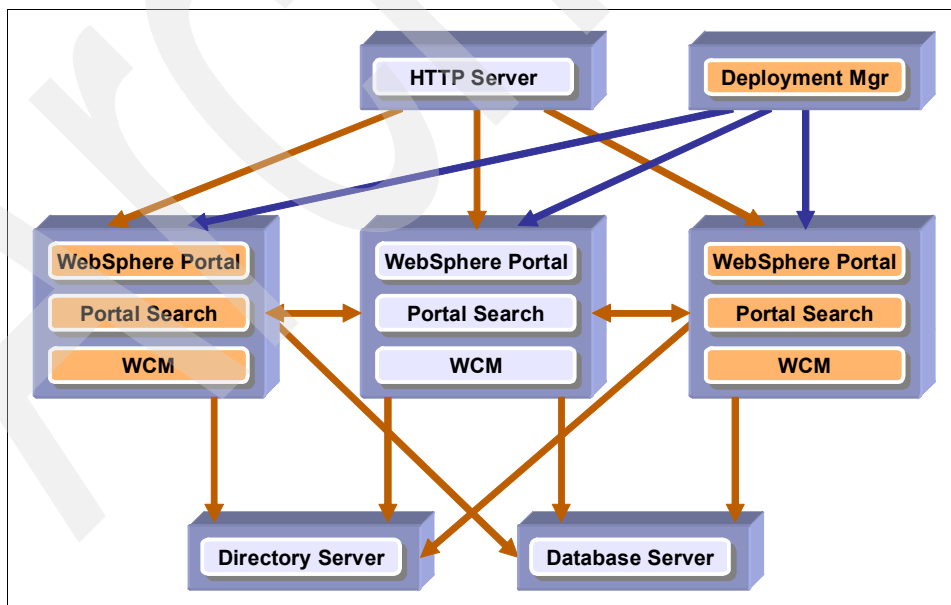


Figure 2-3 Illustrates three WebSphere Portal nodes

2.2.4 Mixed horizontal and vertical scaling topology

Mixed horizontal and vertical scaling topology is the best practice approach for large scale deployments. The use of vertical clustering takes full advantage of the resources of a multiprocessor system. In addition, horizontal cloning allows for upward scalability. To maximize security, horizontal clones should reside in different geographic locations to guard against natural disaster or site outages.

2.3 Clustering considerations for high-availability

This chapter focuses on considerations that allow you to operate your Portal infrastructure 24X7 while providing SOA capabilities. These considerations allow you to build a system that is architecturally designed for continuous operation.

2.3.1 WebSphere Process Server considerations

WebSphere Portal provides business process integration support through IBM WebSphere Process Server. It also ensures inter operability and flexibility as part of your service oriented architecture (SOA) through adoption of popular standards, such as BPEL, Web services, JMS, XML, and many more.

Because a cluster must consist of identical copies of an application server, either every instance of WebSphere Portal must be installed with WebSphere Process Server or every instance of WebSphere Portal must be installed without WebSphere Process Server.

Important: Business process support cannot be federated into a Deployment Manager (DMGR) managed cell after installation.

Features and benefits of WebSphere Process Server

The following describes some of the features and benefits of WebSphere Process Server. For more information, visit the WebSphere Process Server Web site on [ibm.com](http://www.ibm.com).

<http://www-306.ibm.com/software/integration/wps/>

- ▶ **Feature:** Integrates more out-of-the box Web services, application adapters, and advanced messaging capabilities.
- ▶ **Benefit:** Maximizes value and reuse of all your assets using the right quality of service to meet your business need at that time.
 - WebSphere Process Server can natively integrate with WebSphere ME infrastructures throughout your enterprise, in addition to leveraging the embedded JMS provider, Web services, and a wide range of application and technology adapters.
- ▶ **Feature:** Allows easy-to-use comprehensive human-centric business process management (BPM) scenarios.
- ▶ **Benefit:** Better management and flexibility of human related tasks including dynamic balancing of staff workloads, assigning work across a team, applying work management policies, reflecting organization structure and change, and automated task feeds.

- ▶ **Feature:** Integration with IBM WebSphere Service Registry and Repository.
- ▶ **Benefit:** Delivers a tailored service to each customer (each process instance) with real time dynamic choices at each step of the process based on customer need and service level.
 - Supports service governance with dynamic run-time discovery and invocation of services—WebSphere Service Registry and Repository integration with WebSphere Process Server provides true end-to-end governance for all services, dynamically discovering and invoking services and service metadata information at runtime. This allows real-time process behavior adaptation.
- ▶ **Feature:** Provides run-time administration improvements (dynamic reconfiguration with no need to rebuild or redeploy).
- ▶ **Benefit:** Administer, manage, and change processes without deploying.
- ▶ **Feature:** Enables tight integration between information services and business processes.
- ▶ **Benefit:** Make the right decisions based on the right information at the right time.
- ▶ **Feature:** Faster authoring of BPM solutions and easier integration to external services (WebSphere Integration Developer Tools but reflected in WebSphere Process Server runtime).
- ▶ **Benefit:** Allows integration developers to focus on creating business value by reducing solution maintenance and debugging.

2.3.2 Multiple LDAP overview

The use of multiple LDAP directories has a significant bearing on how you deploy and maintain WebSphere Portal; therefore, it is important to understand the advantages using more than one LDAP directory.

Advantages

- ▶ Greater deployment flexibility for WebSphere Portal.
- ▶ Reuse existing directories that may already contain important user data.
- ▶ Minimize or negate the need for directory consolidation.
- ▶ No need to change administration practices.
- ▶ Minimizes the LDAP directory being a single point of failure.

Note: Chapter 4, Section 4.1, “Advantages of multiple LDAP directories” on page 188 contains greater technical detail on the LDAP advantages as well as instructions on how to configure multiple LDAP support.

2.3.3 Database domain (database split) overview

To maximize availability, data may be distributed across multiple databases and shared between multiple lines of production.

Portal data can be categorized into the following four categories:

- ▶ **Configuration data** defines the portal server setup, such as database connection, object factories, and deployment descriptors. This type of data is typically constant over the uptime of a server node. Configuration data is typically kept in property files on the hard disk. This data is either protected by file system security or by application server administration rights.
- ▶ **Release data** is all portal content definitions, rules, and rights that are possibly designed externally and then brought into the portal by a staging process, such as Page Hierarchy, available Portlets and Themes, Templates, Credential Slots, Portal Personalization Rules, and Policies. These resources are not modified during production and need administrative rights to do so. Administrators typically create release data on an integration server and stage it to the production system. Release data are protected by PAC and contain only data, not code.

Web Content Management (WCM) managed content is also considered release data, since the content is authored in an environment outside the production environment (typically) and syndicated into production as part of a content workflow. The JCR is used to store WCM content, as well as Theme Policies and Personalization Rules and is thus included under the release category.

In a setup that consists of multiple lines of production, one copy of the release data exists per cluster. Administrators must make sure that the content of the release and JCR databases is consistent across the different lines, in particular after modification of the release data on the production system directly.

- ▶ **Customization data** is data that is associated with a particular user only and that cannot be shared across users or user groups. Typical examples are Portlet Data or Implicitly Derived Pages. Since this data is scoped to a single user only, PAC protection is simplified to a great extent.

In a setup that consists of multiple lines of production, customization data is kept in a database that is shared across the lines of production. Therefore the data is automatically in sync across the lines of production.

- ▶ **Community data** is all resources that are typically modified during production, such as application resources. Typically users and groups are allowed to modify or delete shared resources. Community resources are protected by PAC.

Key advantages to a configuration split: why configuration split?

There are some key advantages to using a database configuration split that one must consider when building a high availability cluster:

- Enables global deployment - Supports customers who want global deployment at different sites
- Improves staging.
- Enables flexible maintenance and backup.
- Reduces the overhead in high availability systems.
- Release DB + Community DB are shared across the cluster nodes.
- Customization DB is either external or shared with the other DBs.
- Sites can now replicate the customization DB (2-way sync).
- Users get routed to either production line.
- Both lines run on a copy of the same release data and the same shared user data.

- Users can modify customization data but not release data or community data.
- Working with workplace applications will only modify community data (+ optional external data sources referenced by the applications).
- Non-disruptive maintenance and backups fully supported!

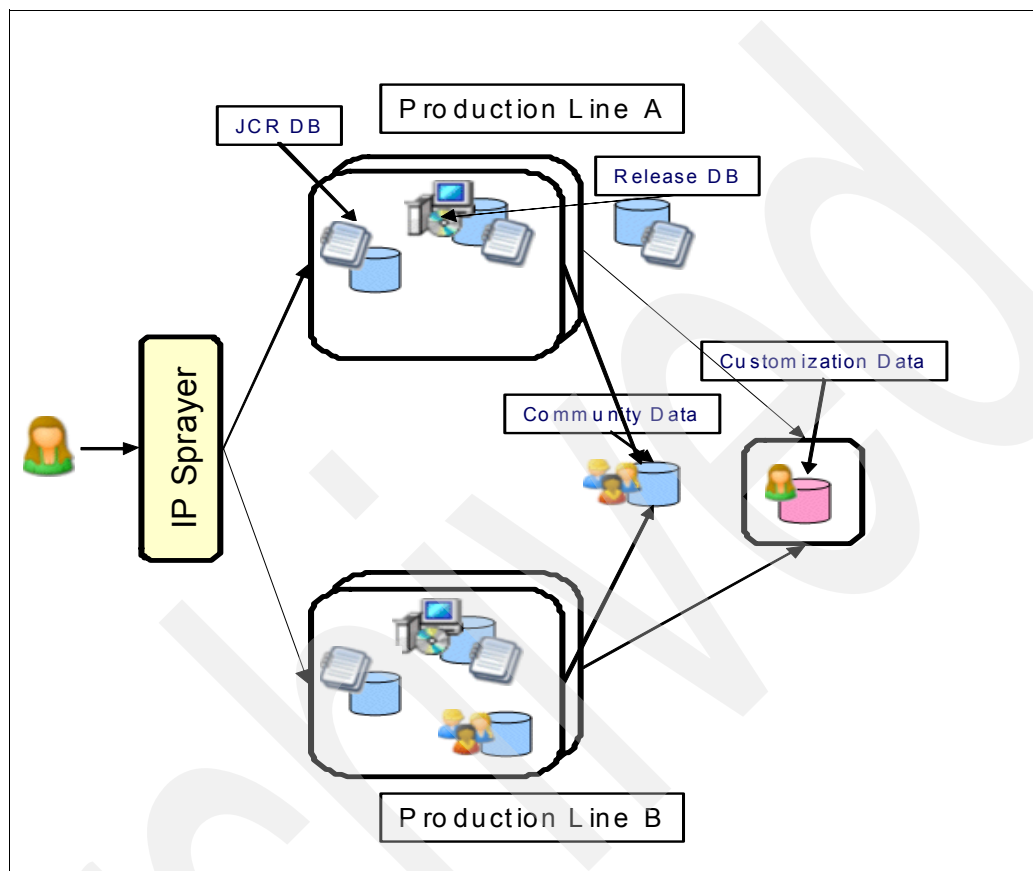


Figure 2-4 A configuration Split

Note: For much greater detail on how to configure this feature, please refer to Chapter 5, “Database domains” on page 233.

2.3.4 Profiles

WebSphere Application Server V6 introduced the concept of Profiles, which is separation of core product files and user data (or run time execution environment). Following are some key points to remember about profiles:

- The default install path is <was_root>/profiles but can be outside <was_root>.
- System admin commands are profile specific.
- Execute from <profile_directory>/bin or <was_root>/bin using -profileName parameter.
- Each profile has a unique cell name and a unique node name combination. There is one-to-one mapping between the node and a profile.
- Ports generated for the node agent are unique for all the profiles in the installation.

2.3.5 Master configuration repository

Master configuration repository is maintained by the DMGR. The DMGR maintains a master repository of the individual node configurations. Master configuration repository is the bases of what is replicated out to all of the existing members of a cell. Each node has its own representations of the configuration repository

Master configuration repository

Location:

`<dmgr_profile_root>/config`

Master configuration for all nodes.

Bases of what is replicated out to all of the individual members of a cell.

Configuration repository

`<profile_root>/config`

Enterprise Apps (config part)

`<config_root>/cells/<cell_name>/applications`

Nodes

`<config_root>/cells/<cell_name>/nodes`

Servers

`<config_root>/cells/<cell_name>/nodes/<node_name>/servers`

If you make changes on a local node to configuration the changes are temporary as the Deployment Manager is maintaining the master copy. DMGR synchronizes with the local node and overwrites the changes you made on the individual node.

Important: Permanent configuration changes should be made through the DMGR console.

2.3.6 Web server considerations

In this section we discuss the following items:

- ▶ HTTP session failover
- ▶ Failover and lost data
- ▶ WAS V6 Web server changes
- ▶ The recommended process to configure remote external Web server

HTTP session failover

Enable HTTP session failover in your cluster to enable multiple cluster members to serve data in the event that one of them fails.

In a clustered environment, all requests for a particular session are directed to the same WebSphere Portal server instance in the cluster. In other words, after a user establishes a session with a portal (for example, by logging in to the portal), the user is served by the same WebSphere Portal server instance for the duration of the session. To verify which server is

handling user requests for a session, you can view the global settings portlet in WebSphere Portal, which displays the node name of the WebSphere Portal server handling requests.

If one of the WebSphere Portal servers in the cluster fails, the request is rerouted to another WebSphere Portal server in the cluster. If distributed sessions support is enabled, either by persistent sessions or memory-to-memory session replication, then the new server can access session data from the database or from another WebSphere Portal server instance.

Distributed session support is not enabled by default and must be configured separately in WebSphere Application Server. Refer to the WebSphere Application Server documentation for more information:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.webSphere.nd.doc/info/ae/ae/cprs_persistent_sessions.html

Failover and lost data

Data that is stored within the JVM™ memory and not managed by the application server or the WebSphere Portal server for replication might be lost in the case of failover. Even with the distributed session support, during a failure, users cannot recover any uncommitted information that is not stored in sessions or other replicated data areas (such as a distributed Map or render parameters). In such cases, users might have to restart a transaction after a failover occurs. For example, if you are in the process of working with a portlet and have navigated between several different screens when a failover occurs, you might be placed back at the initial window, where you would need to repeat your previous steps. Similarly, if you are attempting to deploy a portlet when a failover occurs, the deployment might not be successful, in which case you must redeploy the portlet. Note, however, that with distributed session support enabled, the validity of user login sessions is maintained despite node failures.

In cases where a portlet does not support failover, a "Portlet Unavailable" message is displayed for the portlet after a failover occurs. If a portlet supports partial or incomplete failover, some data displayed by the portlet before the failover could disappear after the failover occurs, or the portlet might not work as expected. In such extreme cases, the user should log out and log back in to resume normal operation.

After a failover occurs, the request is redirected to another cluster member by the Web server plug-in. Most browsers issue a GET request as a response to a redirect after submitting a POST request. This ensures that the browser does not send the same data multiple times without the user's knowledge. However, this means that after failover, users might have to refresh the page in the browser or go back and resubmit the form to recover the POST data.

Note: Document Manager portlets supplied with WebSphere Portal and any other portlets that use POST data are affected by this behavior.

WAS V6 Web server changes

With WAS V6 there are some key architectural changes in how Web servers are represented. Web Servers have become an architectural component.

- Deployment Manager explicitly manages Web server, and applications are explicitly mapped to the Web server plus cluster.
- So if you are looking at the enterprise applications target mappings for Portal you will see one entry for the cluster and one entry for the Web server so the applications now have to be mapped to the Web server.
- Different apps can be mapped to different Web servers if desired.

- Plugin-cfg.xml can be automatically propagated to Web server (IHS or Web server on a managed node).

Recommended process to configure remote external Web server

- ▶ Use Plug-in install wizard to install Plug-in on the remote Web server.
- ▶ Also possible to create the Web server instance through Deployment Manager Administrative Console or via wsadmin commands in WAS V6.
- ▶ Copy configure<webServerName>.sh or .bat script to Deployment Manager.
- ▶ Run script on Deployment Manager.
- ▶ Use Deployment Manager Administrative Console to re-sync with all nodes to push required config updates to each Portal node.
- ▶ Most customizations to plugin-cfg.xml can be done through Deployment Manager Administrative Console.
- ▶ Separate process defined for local external Web server (co-residing with DMgr or with a Portal node in the cell).

2.3.7 Hardware requirements for high availability

Prior to building a portal cluster with full redundancy one must consider how much hardware or servers are needed and plan accordingly. In an ideal world each component (ldap, db2, http, wsas/was/wps, and dm) are fully redundant and each have a dedicated server. We realize that this is not always possible and an organization must prioritize.

Each number below represents a physical server machine. Please note that some of the servers are optional but recommended for full fault tolerance as to avoid single point of failure.

1. Node 1: contains DB client, WebSphere Process Server, WebSphere Portal Server, and WebSphere Application Server
2. Node 2: contains DB client, WebSphere Process Server, WebSphere Portal Server, and WebSphere Application Server
3. DM 1: Deployment Manager
4. DM 2: Backup Deployment Manager (Optional)
5. LDAP 1: Primary LDAP server
6. LDAP 2: Secondary LDAP server (Optional)
7. DB 1: Primary database server
8. DB 2: Secondary database server (Optional)
9. HTTP 1: Http server for Node 1
10. HTTP 2: Http server for Node 2
11. Edge server 1
12. Edge server 2 (Optional)

For a fully fault tolerant system with redundancy at each component, up to 12 servers are needed, meaning that a cluster can run with as little as three servers. One can consider this lightweight configuration for a test, or development environment but not for production. In section 2.3.8, “Example architectures in operation” on page 36, we go into the detailed architectures in operation as well as answer the questions of how many environments an organization needs.

2.3.8 Example architectures in operation

This section contains examples of architectures that IBM clients are using today.

Fault tolerant Portal cluster (with full redundancy)

The fault tolerant Portal cluster (with full redundancy) is a typical fault tolerant Portal cluster where caching is typically used. This solution avoids a signal-point-of-failure with redundancy across the board. Each Portal is running in a different physical location (in this example, one in Raleigh and one in Charlotte). Only one directory server and database server is active at one time. Support 24x7 can also be used.

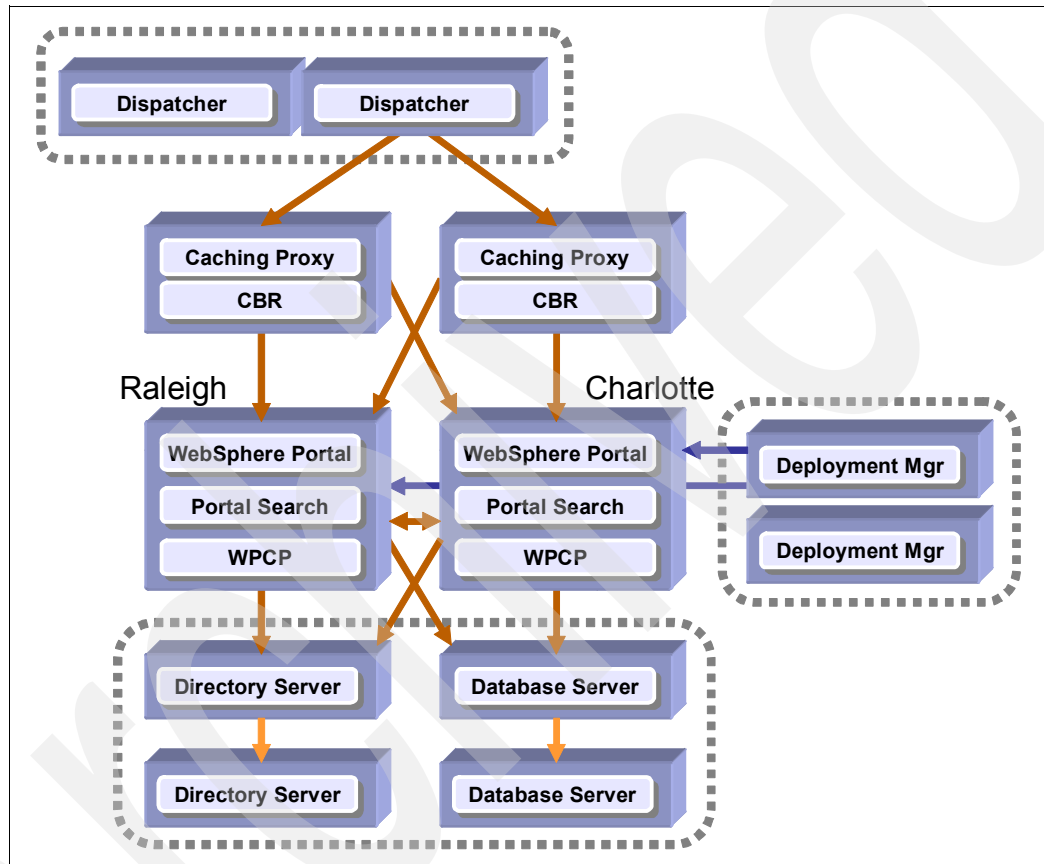


Figure 2-5 Fault tolerant Portal cluster avoids a single-point-of-failure and makes use of caching

Fault tolerant Portal cluster (with full redundancy and security)

The Fault Tolerant Portal cluster (with full redundancy and security) adds a security server (either Tivoli Access Manager or SiteMinder) providing 24x7 security operation. In addition, a Single Sign On environment can exist without having to logon more than once. This Portal architecture avoids a single-point-of-failure and makes use of caching and enhanced security. The location of security proxies allows for protection of cached content.

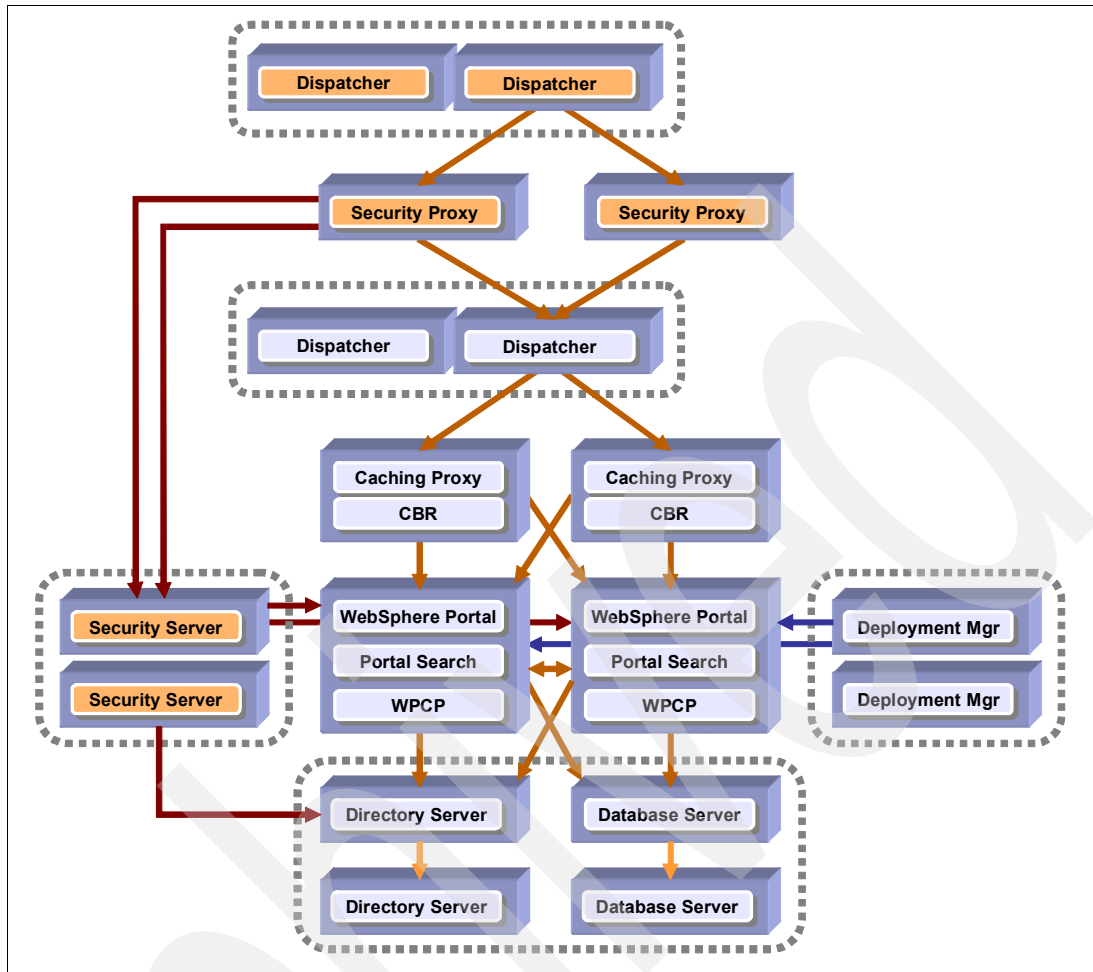


Figure 2-6 Fault tolerant Portal cluster makes use of caching and enhanced security

The Availability Gold Standard in Portal 5.1

In Portal 5.1 the Availability Gold Standard allows WebSphere Portal to run on two sets of clustered machines. This Portal architecture allows you to operate in a 24x7 environment while maintaining easy configuration and maintenance procedures. This is both an active and a passive configuration. The other side is used as a warm backup.

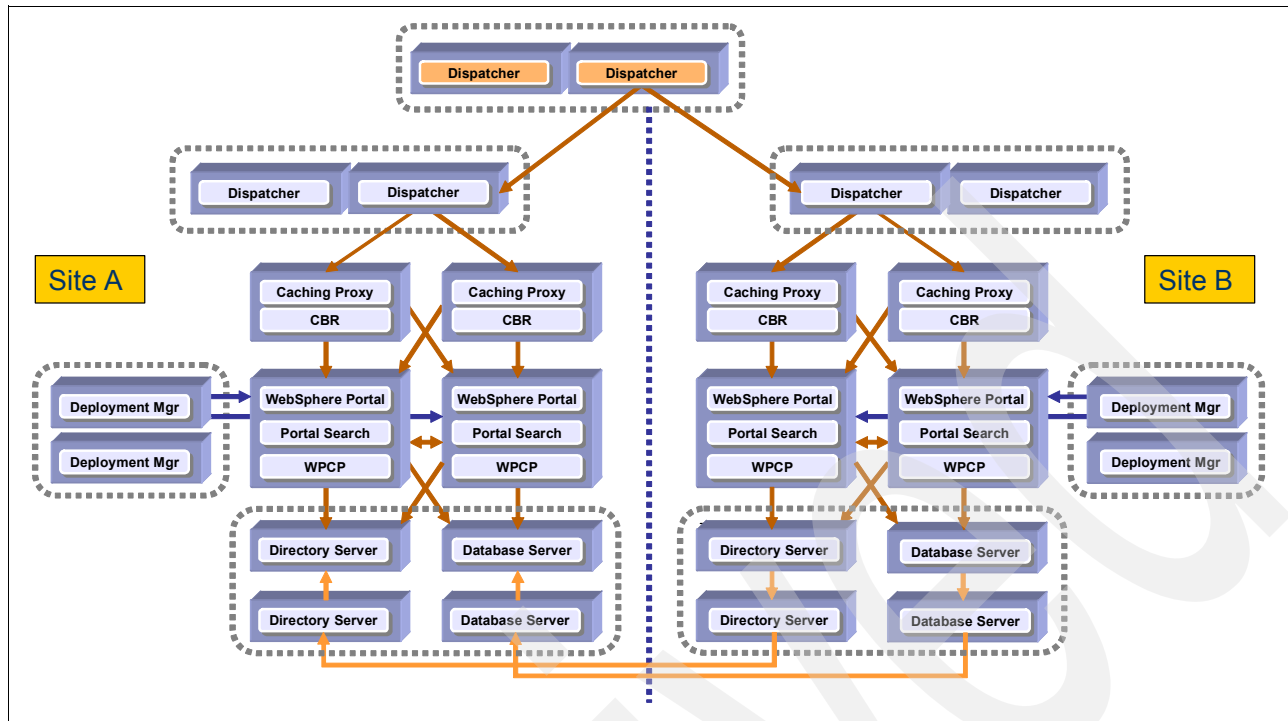


Figure 2-7 Portal 5.1 Availability Gold Standard architecture illustrates two separate WebSphere Portal infrastructures

The new Availability Gold Standard in Portal 6.0

In Portal 6.0 the Availability Gold Standard allows WebSphere Portal to run on two sets of clustered machines independently. This Portal architecture allows you to operate in a 24x7 environment while maintaining easy configuration and maintenance procedures. This is an active configuration where two separate portal infrastructures (cells) service user requests.

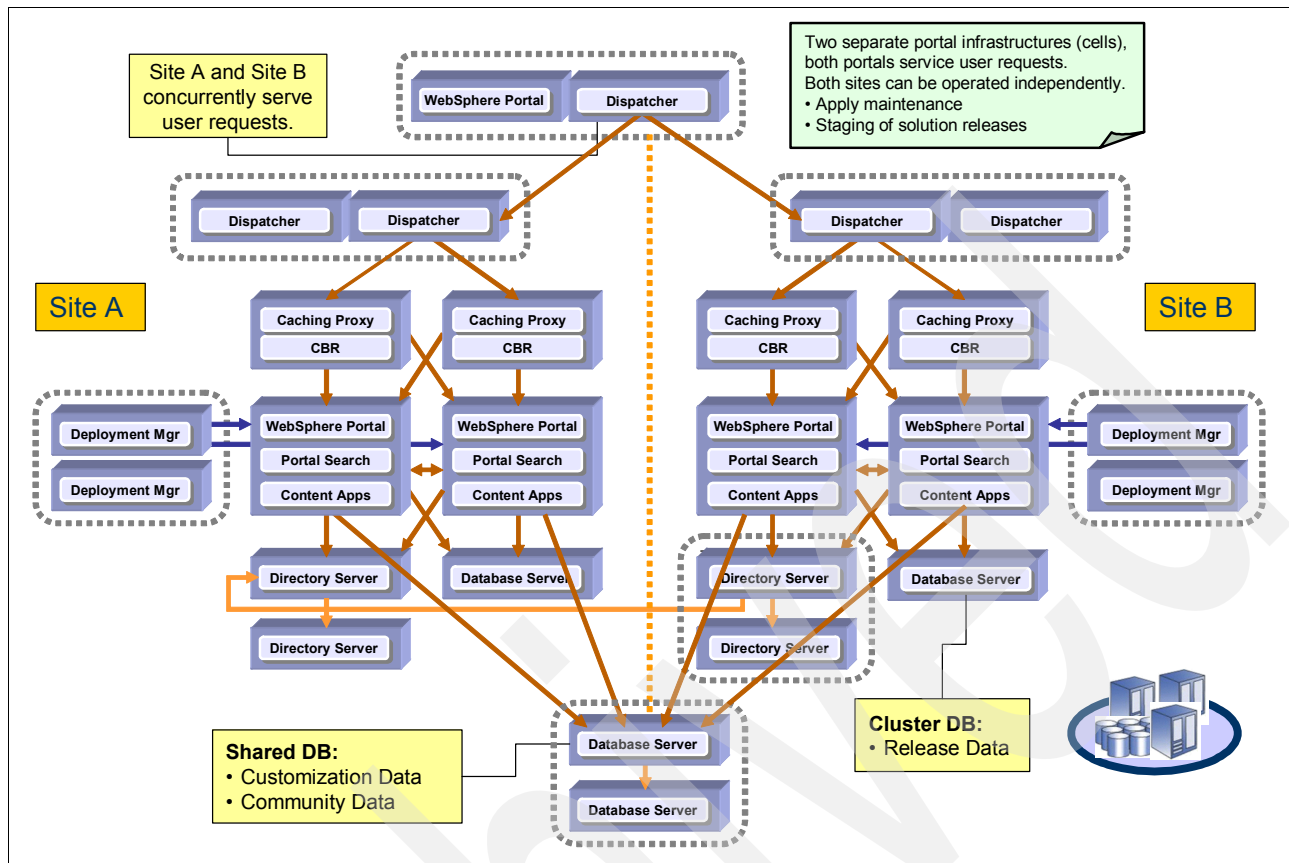


Figure 2-8 Portal 6.0 Availability Gold Standard architecture illustrates two separate WebSphere Portal infrastructures

Continuous operation:

The following statement speaks to the high availability of components operating in a 24x7 environment.

Glossary of Telecommunications Standard [1037C - 1997]

Operation in which certain components, such as nodes, facilities, circuits, or equipment, are in an operational state at all times. (188) Note: Continuous operation usually requires that there be fully redundant configuration, or at least a sufficient X out of Y degree of redundancy for compatible equipment, where X is the number of spare components and Y is the number of operational components.

In data transmission, operation in which the master station need not stop for a reply from a slave station after transmitting each message or transmission block.

2.4 Recommendations and best practices: how many environments do I need?

The following section is intended to provide an overview of typical environments that are used for developing, testing, and staging your WebSphere Portal Infrastructure. Ultimately, the decision about exactly how many environments you need is based upon the architecture of your production environment. There is not necessarily a single answer that applies to each

customer situation; instead, there are several factors that determine the extent to which you need separate pre-production environments.

Figure 2-9 illustrates an overview of commonly used environments to support development, integration, testing, staging, and ultimately production of your WebSphere Portal environment. As we describe in the upcoming sections, you may not need this many distinct environments.

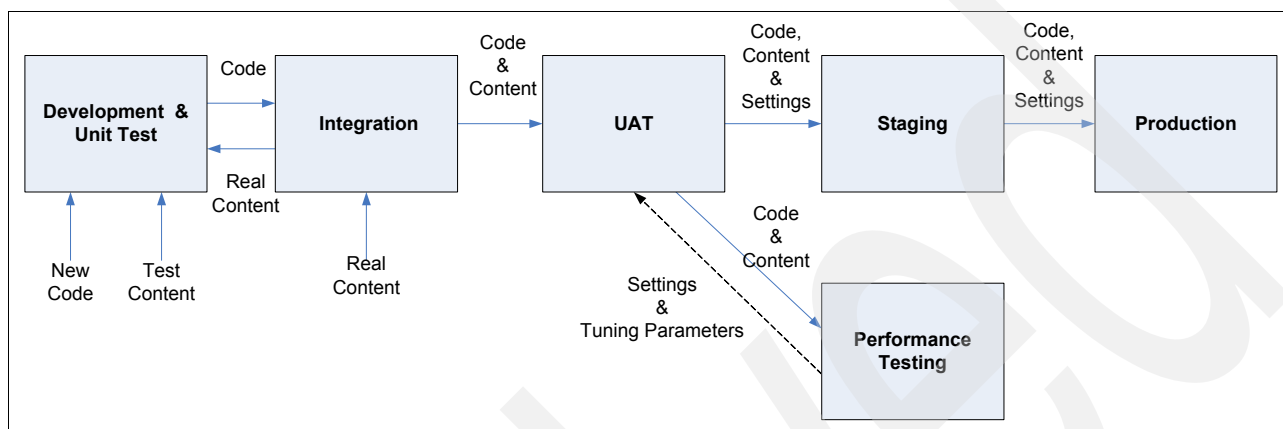


Figure 2-9 Commonly used environments

The two most important factors in determining the number of environments you need are your development scope and cycle, and your budget. Figure 2-10 illustrates the ideal set of environments, assuming that you are going to have at least one dedicated development team, at least one dedicated testing team, and an unlimited budget.

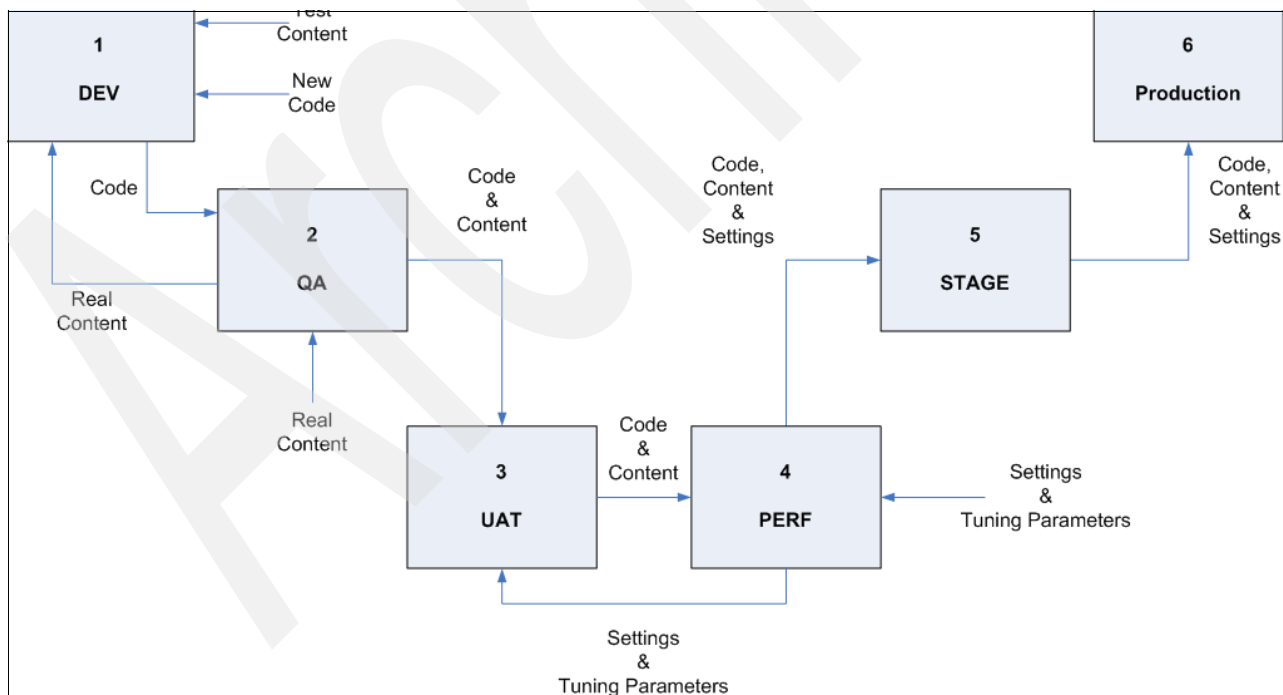


Figure 2-10 The ideal set of environments and data flow between them

Remember that one of these environments must be dedicated as the starting point of full control over the configuration of the system. The earlier in the release cycle this environment is (for example, Dev or QA) the easier configuration management will be for your administrators to move releases through the environments; consequently, this places more restrictions on your developers.

2.4.1 Purposes of each environment

Every organization has its own naming for these environments. Following are the names and purposes we use throughout this publication.

Development/Unit Test (DEV)

Your developers will have their own Test Environments on their desktops. DEV is the first integrated environment where they can first run their code in a real portal environment. This environment quickly becomes a clutter of tests, placeholders, strange stops and forgotten experiments—probably nothing that you want to even possibly migrate to production. But your developers need and appreciate the freedom of having this *sandbox* environment on which to play. Consider having a scheduled cleanup of this environment, perhaps concurrent with a production rollout, to delete everything and reset to the current production build.

Testing (QA)

This is the first environment that actually resembles production. Once developers unit test their code, it is deployed to the QA environment for evaluation by the testing team for functionality and determination that it meets the requirements. This environment has the back-end connections necessary to test code, but not necessarily all of the ones that are in the production environments. Your development team will often use this environment.

Integration testing (UAT)

This might be the first environment that has all connections to back-end environments with realistic or real data. This is where end-to-end testing occurs between portal and all other systems with which you integrate. These systems could include SSO (SiteMinder/TAM), SAP®, PeopleSoft®, SameTime, content management systems, and custom applications. Your testing team interfaces with this environment the most.

Performance testing (PERF)

This is the environment where you run your performance tests. The hardware in this environment should be identical to production in every way for the results to be meaningful. This is probably the most important environment you build. This is where you play with JVM tuning parameters, caching options, database tuning, and capacity planning. You can also use this environment for disaster recovery testing, as its usage and availability schedule will be known and controlled.

Staging (STAGE)

This is the last environment before Production and should be an exact replica of production in its configurations and content. It may be connected to an exact replica of the production LDAP server, or frequently to the same LDAP server. This is where you do final testing and evaluation of both the code, content, and the deployment process before you do the production deployment.

Production (PROD)

This is the production environment that your users access. All the other environments exist to support this one and ensure that anything that goes into the PROD environment was completely and thoroughly tested and is ready for your customers.

2.4.2 Recommendations

In an ideal world, you would have all of these environments, and they would all be identical in configuration and identical in hardware. But we understand that it may not always be feasible, so where do you cut back or collapse environments?

As many environments as you can manage should at least be federated nodes managed by Deployment Manager, if not actually clustered. This minimizes differences in administrative tasks, reducing the burden on your administrators. All environments should be on similar hardware, operating system and release levels—for example, all on Windows 2003 servers, all on RS/6000 servers running AIX 5.3, or all on SunFire servers running Solaris 10, though not necessarily with the same memory or CPU profiles. This minimizes runtime and administrative differences, reducing the likelihood of problems surfacing only late in the release cycle.

Your development environment does not necessarily need to be clustered. It should be configured with security enabled and connected to an LDAP with a structure that looks like production. It can be as simple as a mobile computer sitting on an empty desk somewhere, or it could be an exact replica of your production hardware. If you have a large number of custom portlet development projects or multiple development teams (for example, a development team for each department contributing content to the portal), you could have multiple development environments, one for each team, which all contribute content and code into the QA environment. Alternately, you could dispense with a dedicated DEV environment entirely, have your developers use test environments on their PCs, and deploy directly from version control into the integration or pre-prod environment.

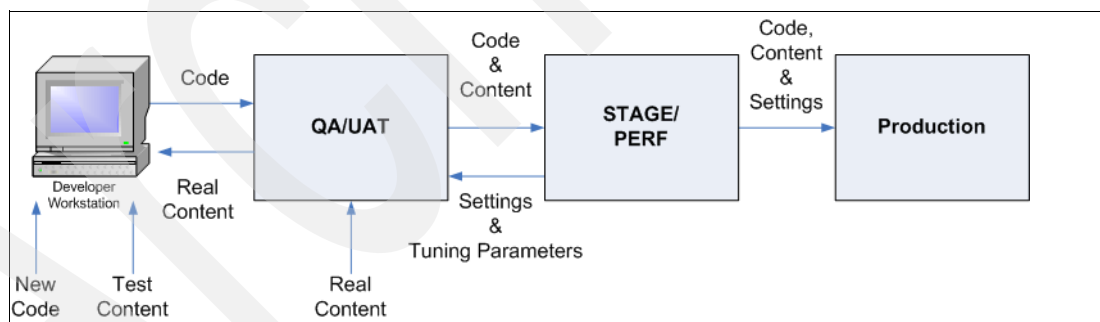


Figure 2-11 Scaled down environment set

QA and UAT testing could be combined into a single-test environment. With this configuration, you should designate the QA/UAT Test environment as the first environment under full Configuration Management administrative control. Your developers cannot “try things out” at will except on their own workstations, which may not have all of the connectivity of a *real* environment.

A dedicated Performance testing environment is highly desirable, but if you are severely hardware constrained, you can do your performance testing on your production servers before you go live. Be aware, however, that once you do your initial launch, it becomes very difficult to do additional performance tuning. If you are lucky enough to have a “slow” time like 2am to do performance testing, you can make changes then, but you would most likely be in violation of a 24x7 SLA at these times. If the hardware profiles of your Staging environment

are the same as production, you could use that environment to do performance testing in that environment before moving the next release into Production, as illustrated in Figure 2-11 on page 42; however, if the hardware in those environments is scaled down, then your only option for Performance testing is to do it in Production after going live during a time of low utilization by your customers. Alternatively, if you have a four-node cluster in Production, your Performance environment may consist of one or two nodes, as long as the hardware is the same, extrapolating the load by a factor of 1:4 or 1:2.

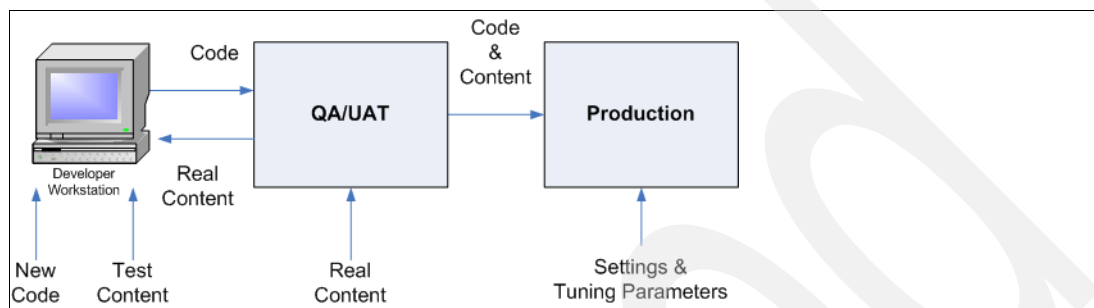


Figure 2-12 Minimal environment set

If you are really tight on hardware, you may even combine staging on the same server as your testing environment, as illustrated in Figure 2-12. This might be advisable if you have a very small or no dedicated testing team, but it limits your concurrent testing cycles and forces longer turnaround times for production rollouts. It also effectively limits the amount of concurrent development that can be supported, as every developer is going to be on their own schedule for writing and testing code, and server restarts must be coordinated with your entire development and testing teams. Again, unless the hardware profile of that environment is identical to that of your Production server, do performance testing in your production environment.

Therefore, at a minimum, you need two environments—Integration and Production. Much better is to have three—Test, Stage/Performance, and Production. The ideal, however, is to have all six environments. The more environments you have available, the less contention for environment availability your developers, testers, and administrators have to deal with and the shorter release cycles you can support.

2.5 Introduction to the deployment scenarios used in this book

The following section illustrates the various clustered environments built during the writing of this book. The section is written in order of how the environment was built so one can see the progression in the deployment. For detailed installation instructions of the various scenarios please refer to Chapter 3, “Different deployment scenarios: building clustered environments” on page 51.

2.5.1 Horizontal cluster with single database and single LDAP

The following diagram illustrates a horizontal cluster with a single database and single LDAP. Installation of this type of environment is described in detail in Chapter 3, both with WebSphere Process Server and without WebSphere Process Server. For WebSphere Process Server considerations please refer to section 2.3.1, “WebSphere Process Server considerations” on page 29.

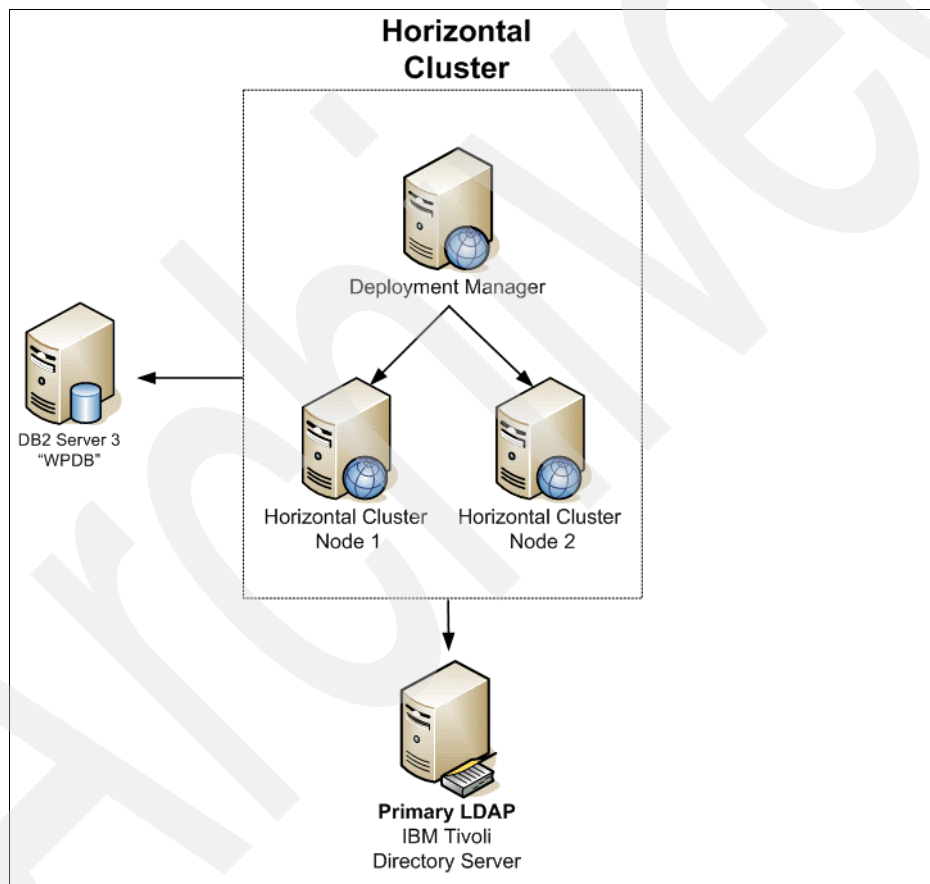


Figure 2-13 Horizontal cluster with single database and single LDAP

2.5.2 Horizontal cluster with database domain

Figure 2-14 on page 45 illustrates a horizontal cluster with database domains. Please note that there are three physical database servers each with 2 schemes serving the horizontal cluster. One database server contains the Release and WMM schemes, one database server contains JCR and Likeminds schemes, and one contains Customization and Community schemes. The benefits of database domain configuration are described in detail in section 2.3.3, “Database domain (database split) overview” on page 30.

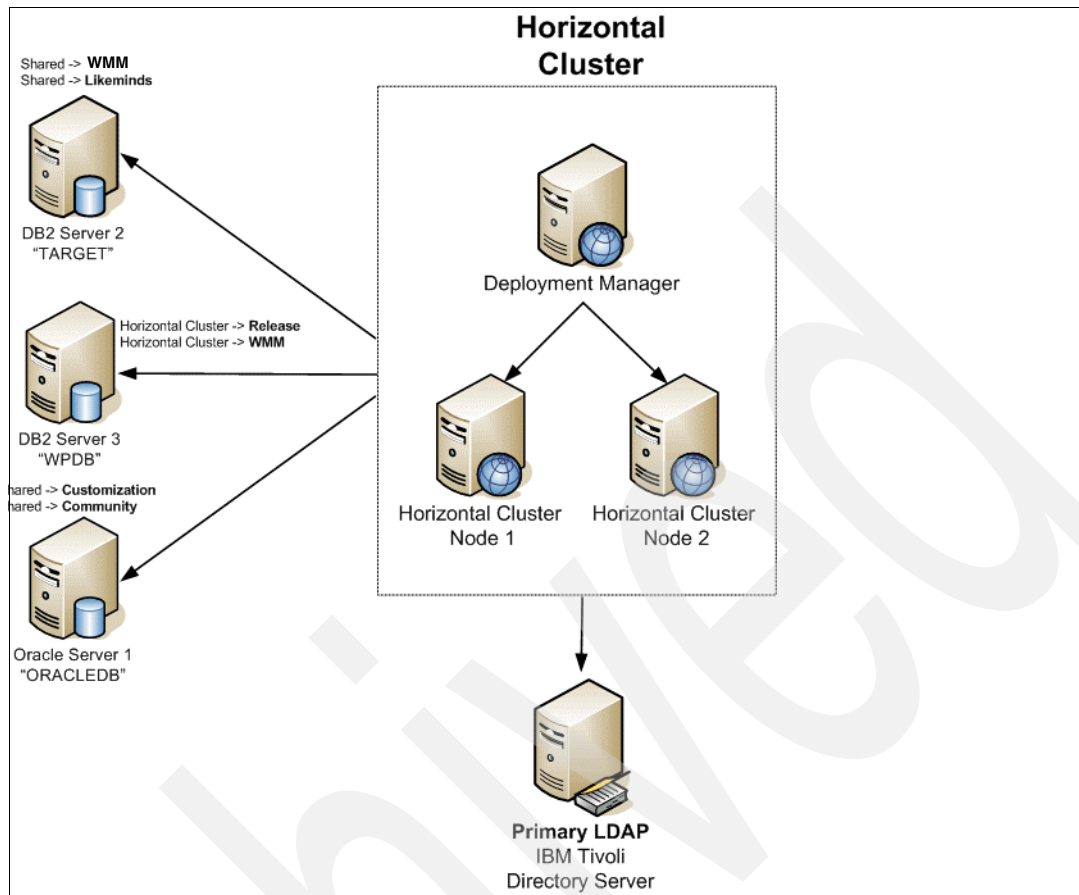


Figure 2-14 Horizontal cluster with database domain

2.5.3 Horizontal cluster with database domain and multiple LDAP

Figure 2-15 on page 46 illustrates a horizontal cluster with database domain and multiple LDAP. We utilize IBM Tivoli Directory Server as the primary LDAP and Lotus Domino 7.0.2 as the secondary LDAP. Users registered in both LDAP servers have access to the Portal environment on this type of configuration. An overview of multiple LDAP appears in section 2.3.2, "Multiple LDAP overview" on page 30.

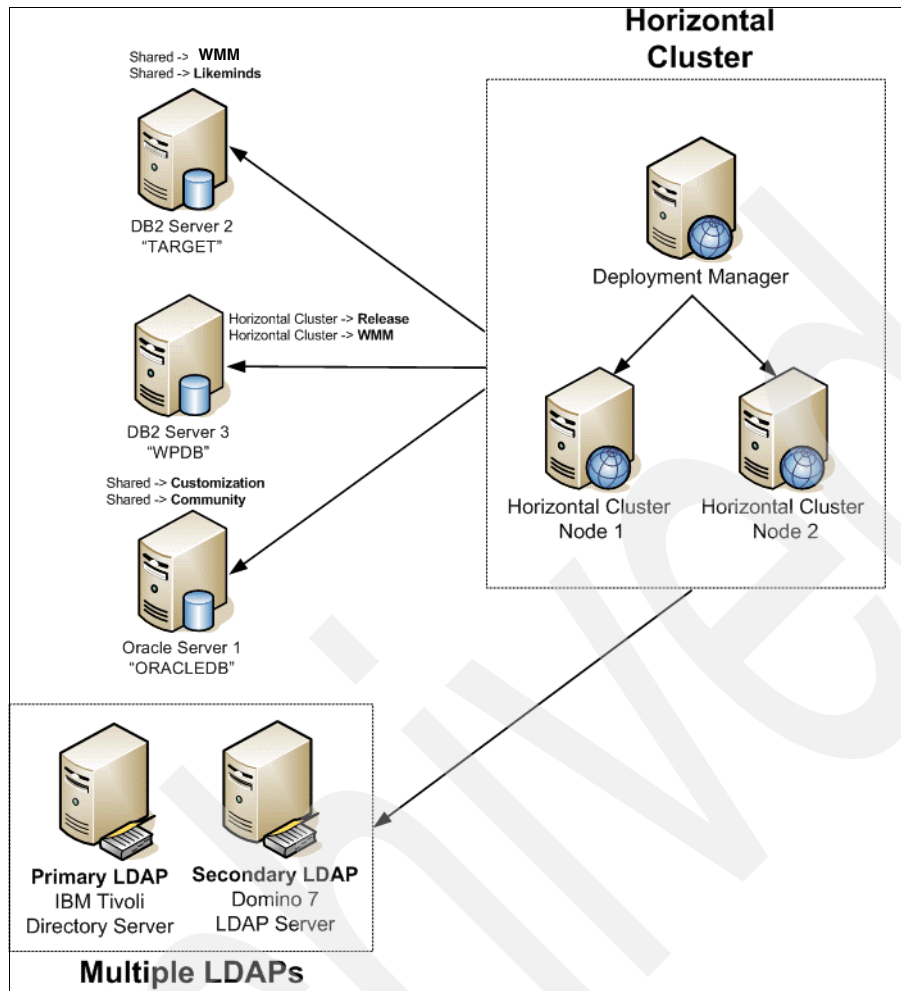


Figure 2-15 Horizontal cluster with database domain and multiple LDAP

2.5.4 Vertical cluster with single database and single LDAP

The following diagram illustrates a vertical cluster with a single database and single LDAP. Please note that the horizontal cluster is still part of our diagram as we are reusing the LDAP and Database servers while building the vertical cluster to utilize multiple LDAP and database domains features.

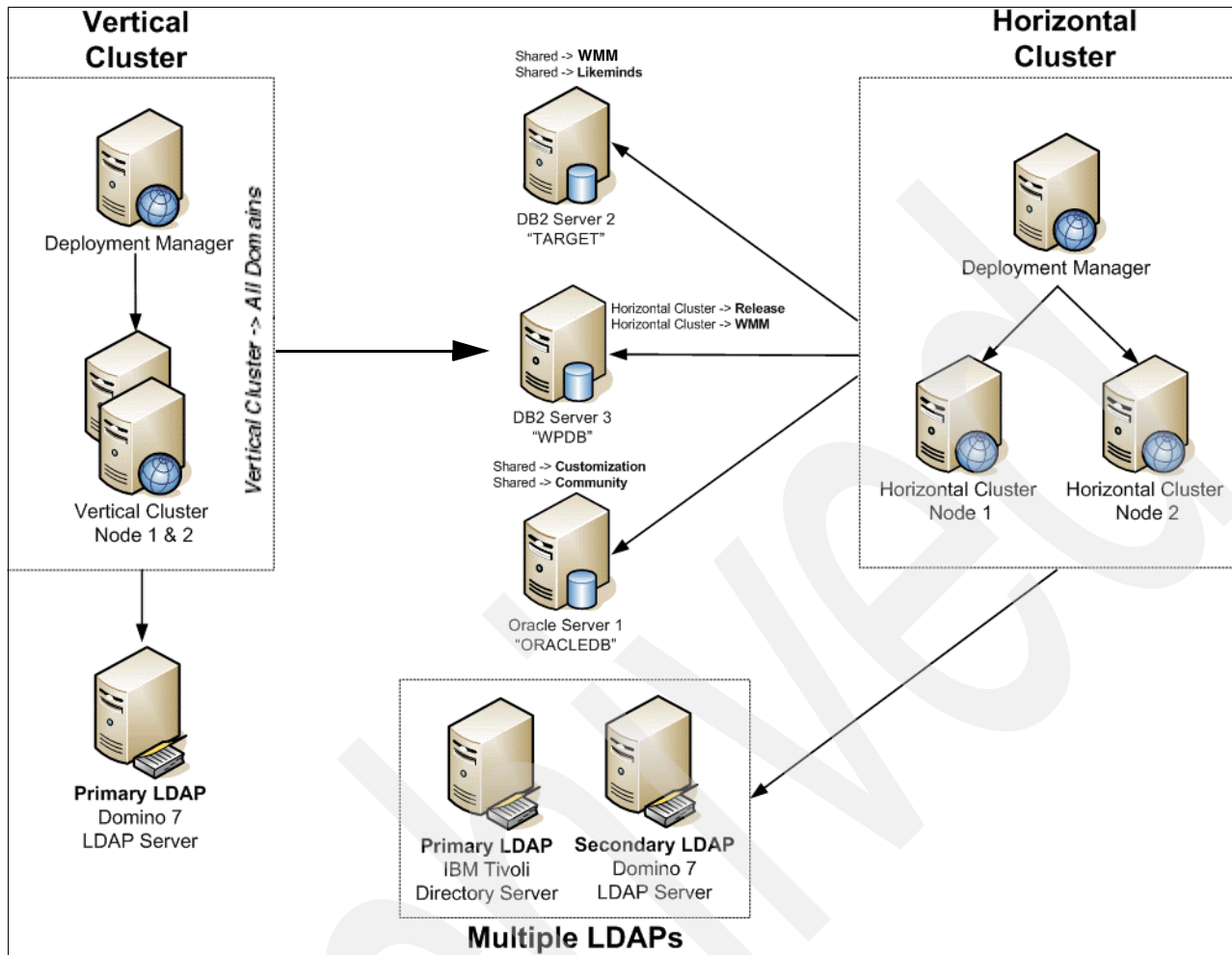


Figure 2-16 Vertical cluster with single database and single LDAP

2.5.5 Vertical cluster with database domain

Figure 2-17 on page 48 illustrates a vertical cluster with a database domain in which the database server is shared with *the horizontal cluster*. Please note that the schemes from the vertical cluster are split across all database servers, some of which are shared with the Horizontal Cluster, as appropriate. The benefits of database domain configuration are described in detail in section 2.3.3, "Database domain (database split) overview" on page 30.

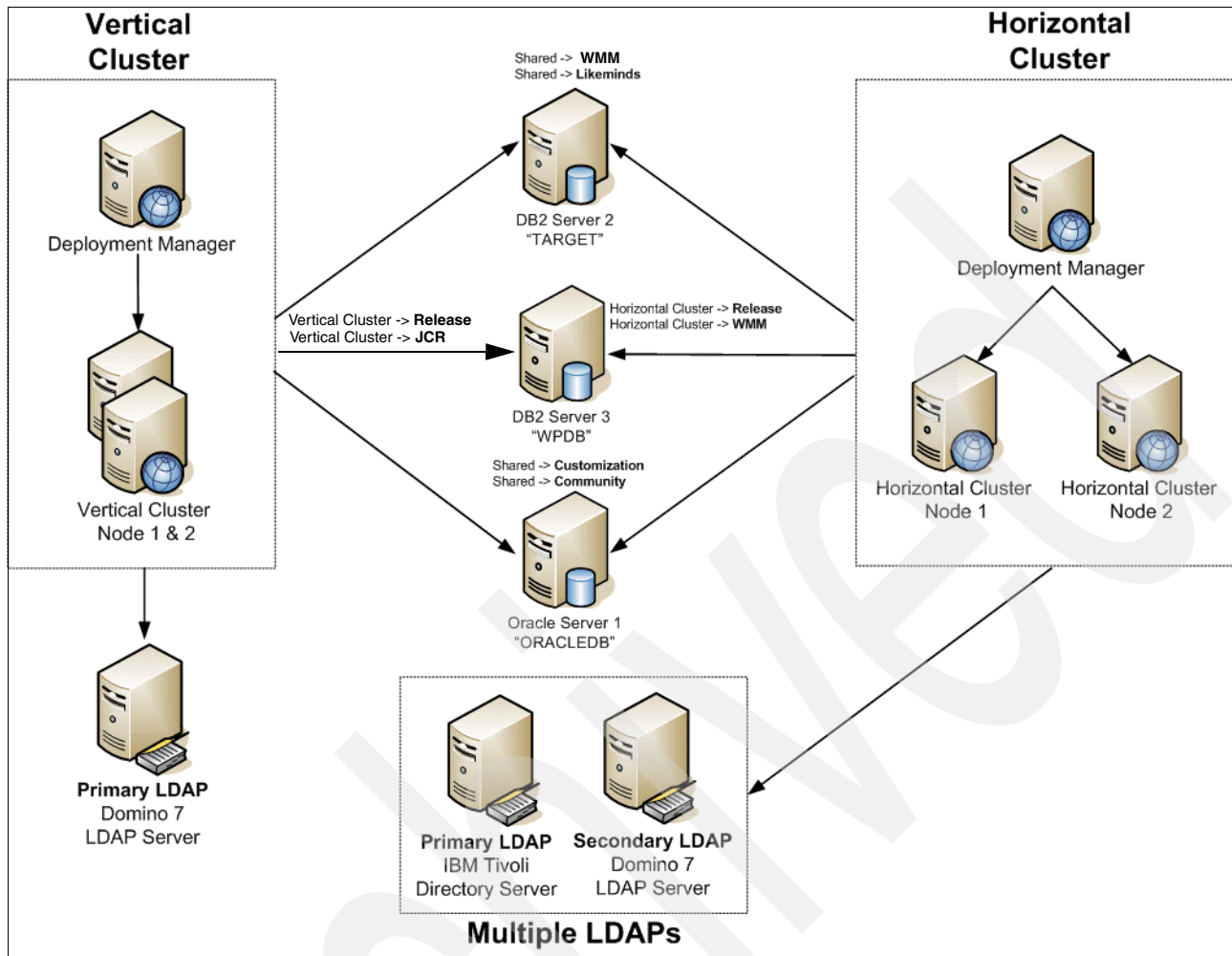


Figure 2-17 Vertical cluster with database domain

2.5.6 Vertical cluster with database domain and multiple LDAP

The following diagram illustrates a vertical cluster with database domain and multiple LDAP. This final environment is an enterprise scale deployment including a vertical cluster and a horizontal cluster. Both of these clusters use the newest features of Portal 6 such as database domains and multiple LDAP. We utilize both Oracle and DB2 for our database servers and use multiple LDAP with Domino 7.0.2 and IBM Directory Server.

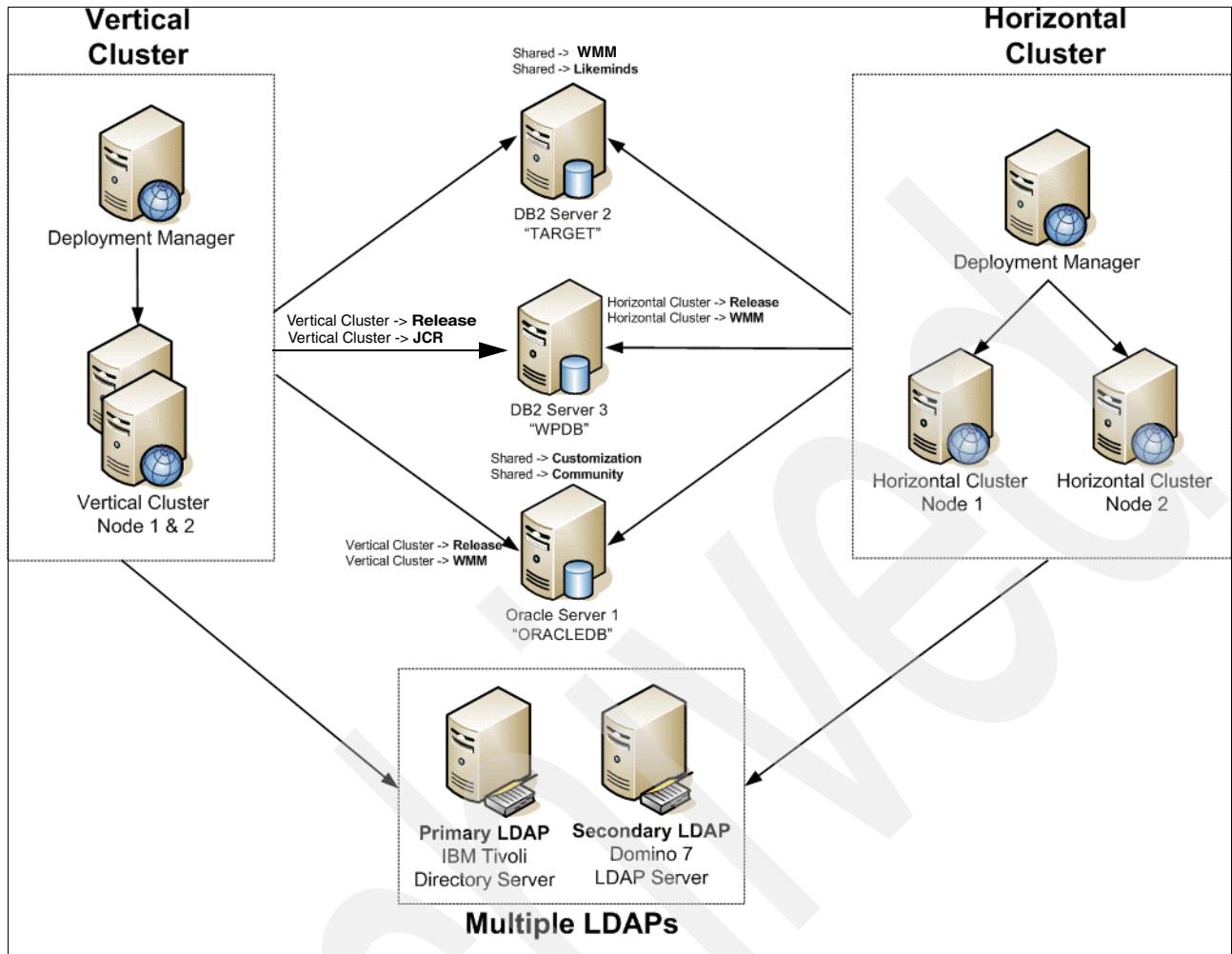


Figure 2-18 Vertical cluster with database domain and multiple LDAP

Archived

Different deployment scenarios: building clustered environments

This chapter discusses the processes for installing IBM WebSphere Portal V6.0 in a clustered deployment and Network Deployment configuration. We discuss the process using a detailed, step-by-step approach, describing four different typical deployment scenarios.

Note:

All of the instructions in this chapter assume you use a Microsoft Windows 2000 Server or Windows Server 2003 environment. We do not discuss the clustered deployment of IBM WebSphere Portal on Linux or AIX machines.

While we do not cover the commands specific to AIX or Linux, the conceptual approach applies to these platforms as well.

3.1 Overview

The following sections provide an architectural and conceptual overview of a Network Deployment installation and configuration.

This chapter also includes the step-by-step procedures for building a horizontal and vertical cluster for high availability, including both deployment *with* or *without* WebSphere Process Server.

3.2 Recommendations for navigating this chapter

We recognize that this is a long chapter and do not wish to overwhelm the reader with too much information. Accordingly, we recommend that you review each of the four different scenarios in order to follow the information that most appropriately applies to your deployment.

The scenarios highlighted in this chapter are as follows:

- ▶ 3.4, “Horizontal cluster (includes WebSphere Process Server) installation and configuration steps” on page 54
- ▶ 3.5, “Horizontal cluster (without WebSphere Process Server) installation and configuration steps” on page 136
- ▶ 3.6, “Adding a Horizontal node to a cluster with WebSphere Process Server” on page 168
- ▶ 3.7, “Add a Horizontal node to a cluster without WebSphere Process Server” on page 172
- ▶ 3.8, “Vertical cluster installation and configuration steps” on page 176
- ▶ 3.9, “Vertical cluster (without WebSphere Process Server) install and configure steps” on page 181

3.3 Introduction

Building and configuring a cluster can be a very complex task. You can build portal clusters in various ways. This chapter provides a best practice approach for building a cluster environment using WebSphere Portal. Your environment might require special considerations, but you should still follow the step-by-step approach as an overall guide.

Although this chapter is specifically written for Portal 6.0.0.0 and WSAS 6.0.2.9/WPS 6.0.1.1 versions, the same approach applies to any Portal 6.0.x version and any WSAS 6.0.x/WPS 6.0.x version as well.

The guide also uses the following acronyms:

- ▶ WP— WebSphere Portal
- ▶ WPS—WebSphere Process Server
- ▶ BPC—Business Process Container
- ▶ WSAS—WebSphere Application Server

Important: Please make a special note that WPS represents WebSphere Process Server and NOT WebSphere Portal.

Note: WSAS Version 6 introduces the concept of profiles. The result is that some property files that you are accustomed to finding under /IBM/WebSphere/AppServer are now at the profile level, like /IBM/WebSphere/AppServer/profiles/<profile_name>.

- For the purposes of this document we refer to:
 - /IBM/WebSphere/AppServer as <wsas_root>
 - and
 - /IBM/WebSphere/AppServer/profiles/<profile_name>
 - as <wsas_profile_root>

Note: In a productive environment we recommend shortening the Path Names, especially on Windows because of size limitations. So use, for example only, /Web/App/.

Scenarios that we discuss in this chapter

In this chapter we discuss several ways to deploy a WebSphere Portal Server Cluster.

For additional details about the different scenarios used throughout the entire book, please refer to section 2.5, “Introduction to the deployment scenarios used in this book” on page 44.

Important installation consideration with WebSphere Process Server

Chapter 2, “Planning for your Portal 6.0 high- availability deployment” on page 21 also introduces the WebSphere Process Server in section 2.2.3.1, “WebSphere Process Server considerations” on page 29. This add on component to WSAS allows Portal to take advantage of the SOA architecture. Portal can be installed and clustered without WebSphere Process Server, but then you lose the SOA features.

WebSphere Process Server is installed and configured by default when using the Portal Typical install path. However, one very important limitation exists with WebSphere Process Server. WebSphere Process Server does NOT allow a WebSphere Process Server profile to be federated if a Portal server already exists on the node. This limitation basically makes a node that was installed by the Portal installer using the typical install path to be UNCLUSTERABLE. So to work around this we MUST install WebSphere Application Server and WebSphere Process Server separately by using their native installers, federate the empty profile, and then install Portal onto the already existing, federated profile. This is mentioned in 3.4, “Horizontal cluster (includes WebSphere Process Server) installation and configuration steps” on page 54.

3.4 Horizontal cluster (includes WebSphere Process Server) installation and configuration steps

In this section you will find a step-by-step guide to configuring a WebSphere Portal v6.0.0.0 horizontal cluster using WebSphere Application Server v6.0.2.9 and WebSphere Process Server v6.0.1.1.

In a horizontal cluster topology members of a WebSphere Portal cluster exist on multiple physical machines representing each node, effectively and efficiently distributing the workload of a single, logical WebSphere Portal image.

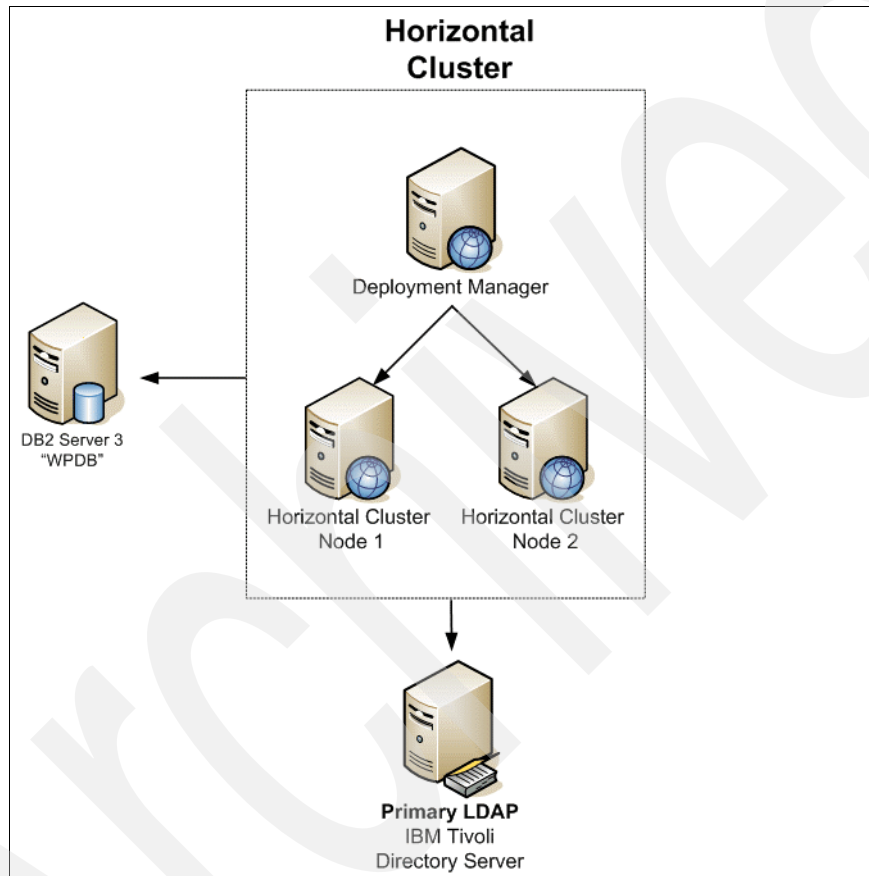


Figure 3-1 Horizontal cluster with WebSphere Process Server built in

In our deployment the following applies:

- ▶ DMC2: This machine served as the Deployment Manager (DMGR). The DMGR is the central administrative console for all nodes in the environment.
- ▶ HORIZC2N1 and HORIZC2N1: Nodes 1 and 2 hosted the business logic. After completion of the installation both machines were clustered for load balancing and failover.
- ▶ DBDOM7: This served as the RDBMS server that hosted the IBM WebSphere Portal database for all nodes and the DMGR.
- ▶ DBIDS represents our LDAP server (IBM Tivoli Directory Server V6), which hosted all directory information used by WebSphere Portal.

Installation steps

In Figure 3-2, we show an outline of the steps we walk you through as part of the installation and configuration steps for a horizontal cluster with WebSphere Process Server.

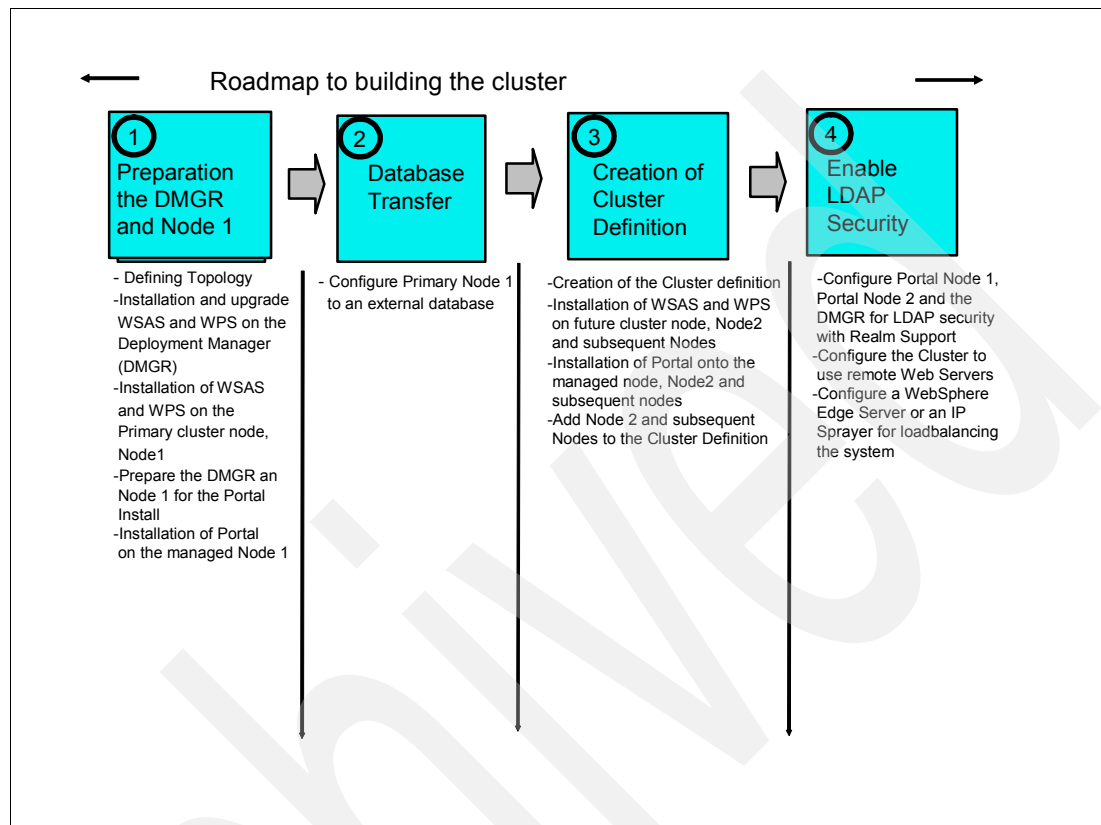


Figure 3-2 Flow chart of the installation steps overview

A clustered deployment of IBM WebSphere Portal Server V6 involves the following:

- ▶ Install and upgrade WSAS and WPS on the Deployment Manager.
- ▶ Install and upgrade WSAS and WPS on the primary cluster node, node 1.
- ▶ Prepare the DMGR a node 1 for the Portal Install.
- ▶ Install Portal onto the managed node 1.
- ▶ Configure primary node 1 to an external database.
- ▶ Create the Cluster definition.
- ▶ Install WSAS and WPS on future cluster node, node 2, and subsequent nodes.
- ▶ Install Portal onto the managed node, node 2, and subsequent nodes.
- ▶ Add node 2 and subsequent nodes to the cluster definition.
- ▶ Configure portal node 1, portal node 2, and the DMGR for LDAP security with Realm Support.
- ▶ Configure the cluster to use remote Web servers.
- ▶ Optional: Configure a WebSphere Edge Server or an IP Sprayer for load balancing the system (This is not described in this book).
- ▶ Optional: Configure the WebSphere portal cluster to use a second LDAP Directory as described in Chapter 4, "Multiple LDAP directory support" on page 187.

- Optional: Configure the WebSphere Portal Cluster to share the Database with a second Clustered environment as described in “Database domains” on page 233.

3.4.1 Install and upgrade WSAS and WPS on the Deployment Manager (DMGR)

Important: This section explicitly defines the required approach to build a WebSphere Portal (Portal/WP) cluster, which was installed on WebSphere Process Server (WPS). To do this you must install Portal into an already federated WSAS/WPS profile. Because of this requirement, we **MUST** install WSAS and WPS from their native installers and federate the node **BEFORE** using the Portal installer to install Portal.

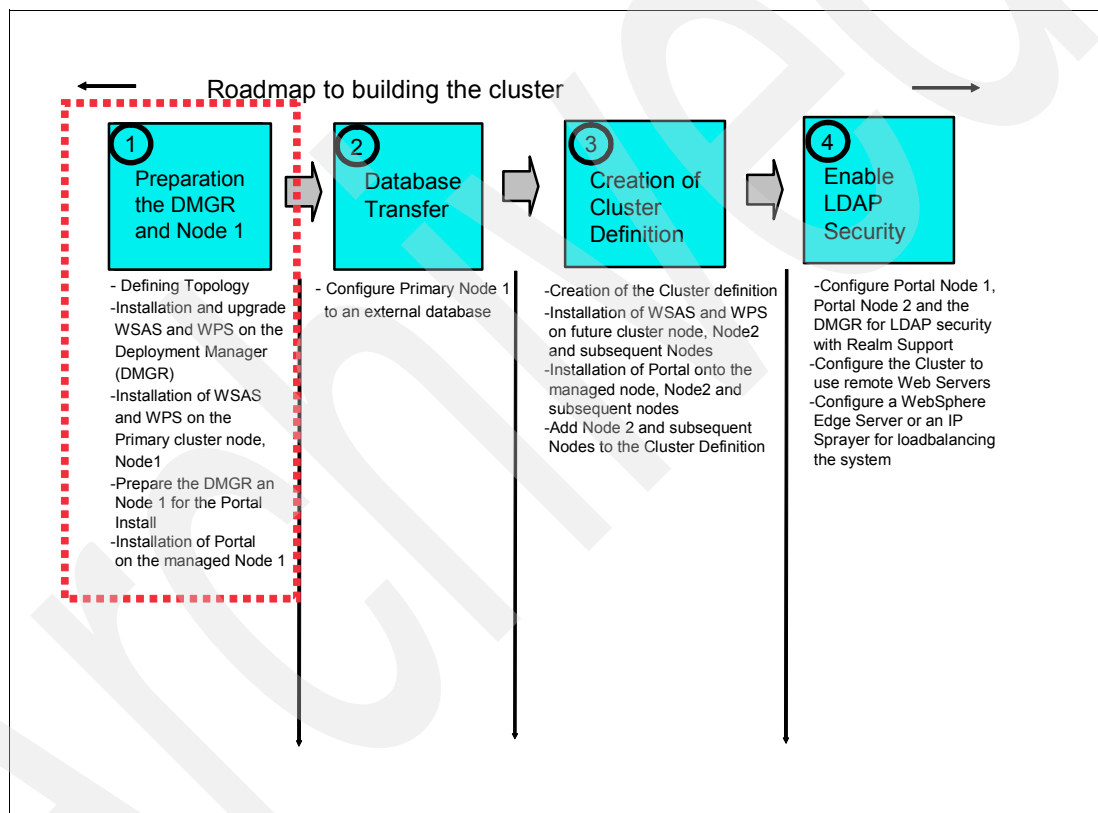


Figure 3-3 Flow chart installation steps: Step 1

WebSphere Application Server installation on DMGR (WSAS)

First install the DMGR using the following procedure:

1. Install WSAS DMGR by running the installer from
`<cd_root>/W-1/windows/ia32/ifpackage/WAS/install.exe`

Note: Make sure the installer window is titled “**Welcome to WebSphere Application Server Network Deployment, V6**”. This title means that you can use this installer to install either, DMGR or WSAS profiles. If the title is “WebSphere Application Server Version 6.0”, you are using an installer that only has the ability to install WSAS profiles and not DMGR profiles.

HINT: If you are installing from downloaded code, run the program from a local drive and not a mapped network drive.

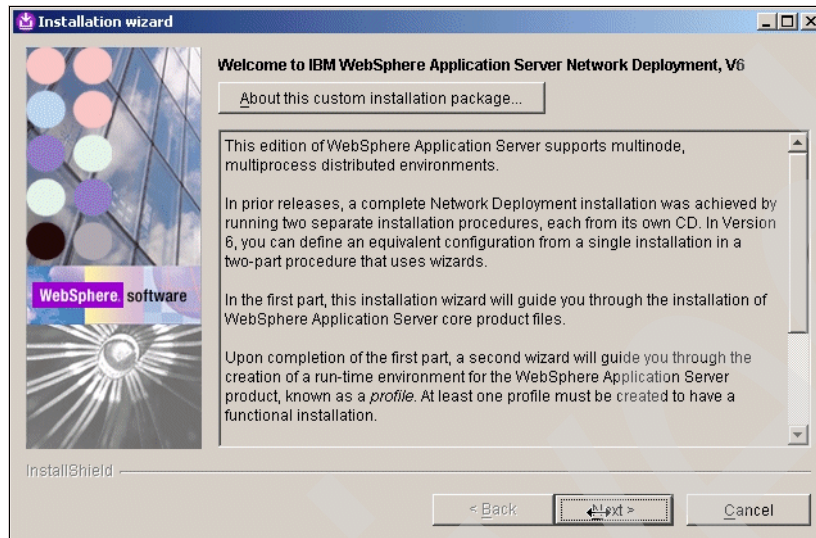


Figure 3-4 Welcome window

2. Accept the License agreement, and click Next.
3. The Installation wizards perform a pre-requisite check for you when this runs successfully. Click Next; otherwise, check your Operating System if it meets the requirements for WSAS as discussed in Section 2.1, "System requirements" on page 22.
4. If installing on Windows, when asked for the install location, please shorten the default path. There is a path name limitation in Windows. Windows cannot handle path names longer than 256 characters. For example:
D:\IBM\WebSphere\ApplicationServer
Click Next as shown in Figure 3-5.

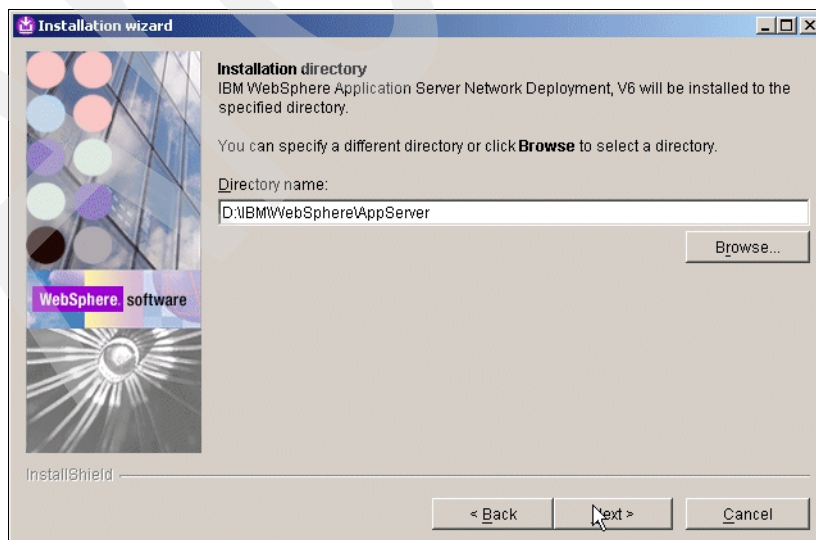


Figure 3-5 Installation directory window

The Installation Wizard displays the installation summary including interim fixes to be applied.

5. Review the options in the installation summary window, and click Next as shown in Figure 3-6.

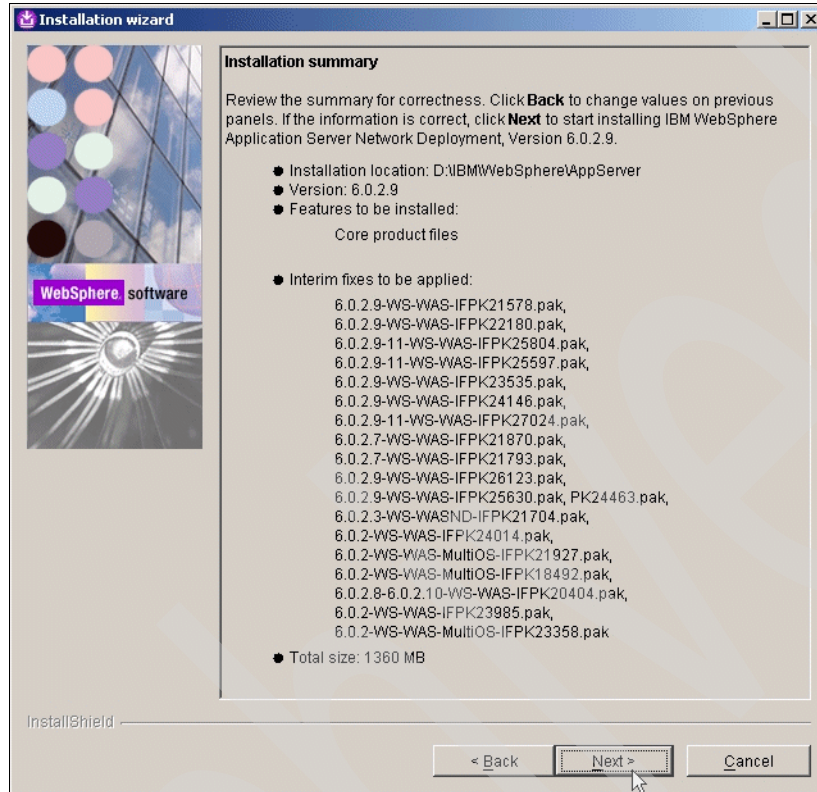


Figure 3-6 Installation summary

6. The WSAS installer from the Portal CDs automatically upgrades WSAS to 6.0.2.9.
7. The WSAS installer from the Portal CDs also automatically applies the following WSAS and WPS iFixes required for an install of Portal 6.0.0.0:
 - PK21578
 - PK22180
 - PK25804
 - PK25597
 - PK23535
 - PK24146
 - PK27024
 - PK21870
 - PK21793
 - PK26123
 - PK25630
 - PK24463
 - PK21704

- PK24014
- PK21927
- PK18492
- PK20404
- PK23985
- PK23358

Note: You can find WSAS fixes by accessing the following Web site and searching for the fix:

– <http://www-306.ibm.com/software/webservers/appserv/was/support/>

8. During the install (with a panel near the end), you are asked if you would like to create a profile. At this time please choose NOT to create a profile by making sure the “Launch the Profile creation wizard” check box remains UNCHECKED. We will create a WPS profile at the end of the WPS installation. Click Next as shown in Figure 3-7.

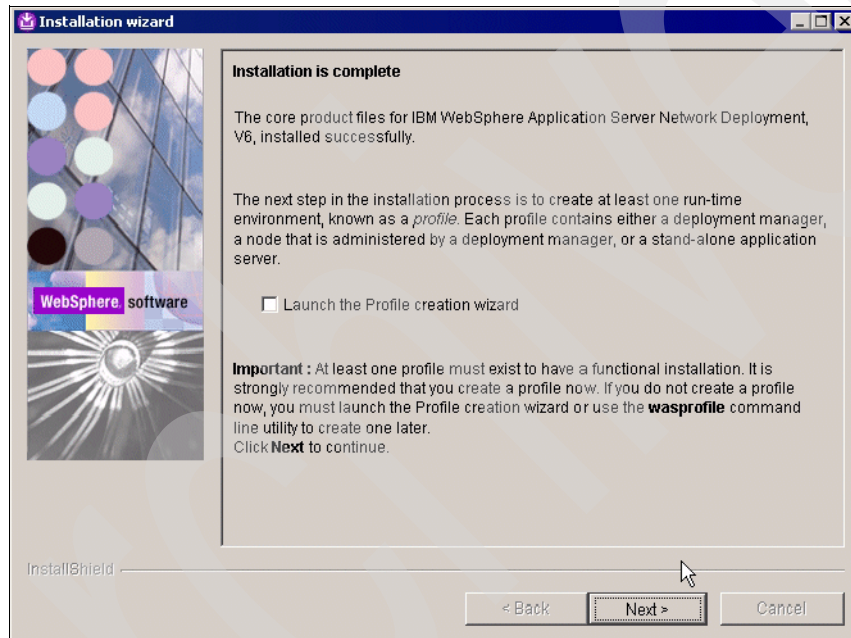


Figure 3-7 Installation is complete window

WebSphere Process Server installation on DMGR (WPS)

1. Install WPS 6.0.1.1 by running the installer from
`<cd_root>/W-2/windows/ia32/WBI/install.bat.`

Important: Please ensure you use the install.bat file and NOT the install.exe to install WPS.

2. The Installation wizard performs a pre-requisite check for you when this runs successfully. Click Next; otherwise, check that your operating system meets the requirements for WPS as discussed in Section 2.1, “System requirements” on page 22.
3. In the “Detected WebSphere Application Server” window, ensure you use the existing WSAS you just installed, and click Next as shown in Figure 3-8 on page 60.

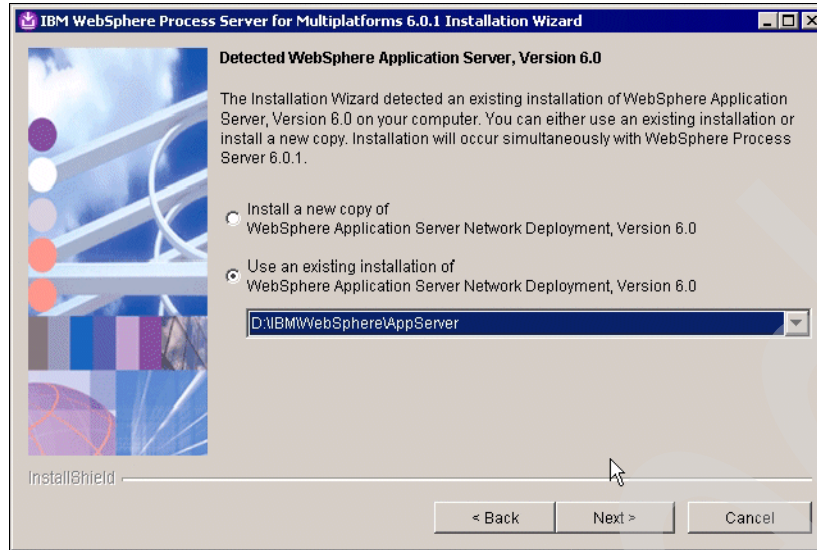


Figure 3-8 WSAS detection window

4. Review the options in the installation summary window, and click Next.
5. In the “Installation complete” window, we want to create a profile now. Select the **Launch the Profile Wizard** option, and click Next to launch the WPS profile creation wizard as shown in Figure 3-9.

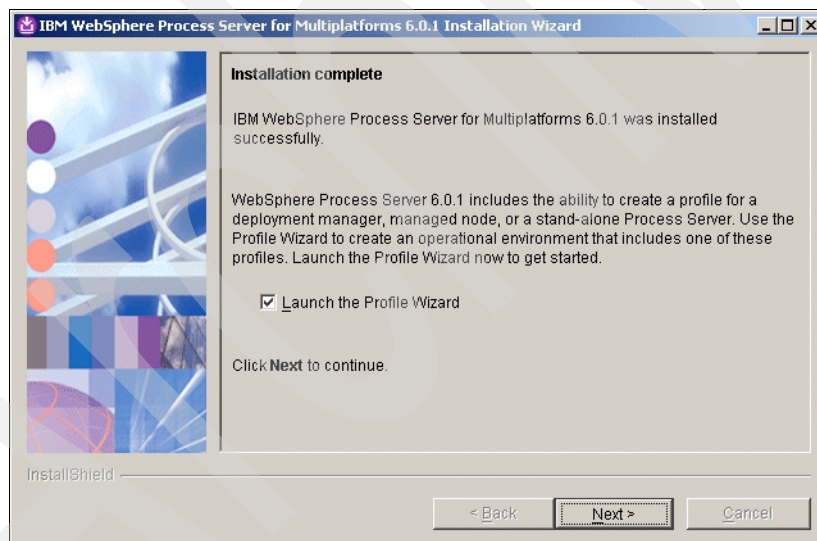


Figure 3-9 Installation is complete window

Note: If you have to launch the WPS profile creation wizard manually, please ensure you launch the WPS profile creation wizard and NOT the WSAS profile creation wizard. The WPS profile creation wizard script is located at `<wsas_root>/bin/ProfileCreator_wbi/pcatWindows.exe`.

6. In the “Welcome to the WebSphere ESB Profile Wizard” click Next.

7. After the profile creation wizard is launched, select the “Deployment Manager profile” radio button on the “Profile type selection” panel as shown in Figure 3-10 on page 61, and click Next.

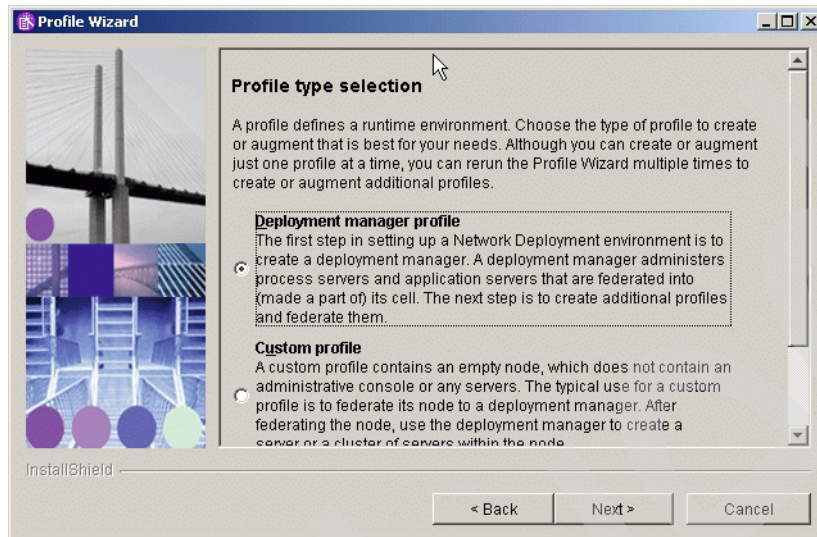


Figure 3-10 Profile type selection window

8. In the “Profile name” window, enter a profile name for the WebSphere Process Server instance. This name can be different for each node and for the DMGR. In our setup, we choose the node name as the profile name. Click Next as shown in Figure 3-11.

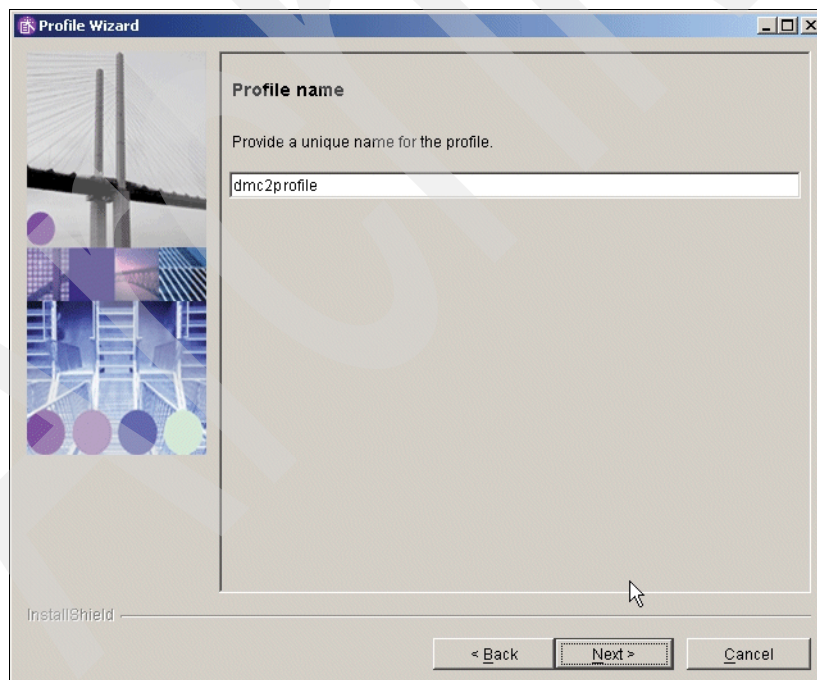


Figure 3-11 Profile name window

9. In the “Profile directory” window, verify the directory name. Click Next as shown in Figure 3-12 on page 62.

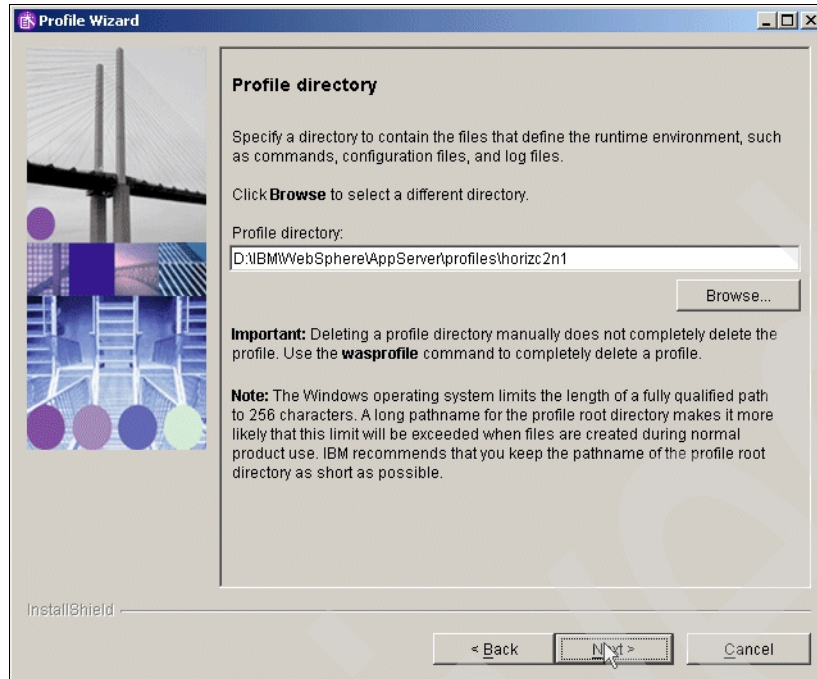


Figure 3-12 Verify the Profile directory selection window

10. In the “Node, host, and cell names” window, check the value in the node name, Host name and Cell name fields. Click Next as shown in Figure 3-13.

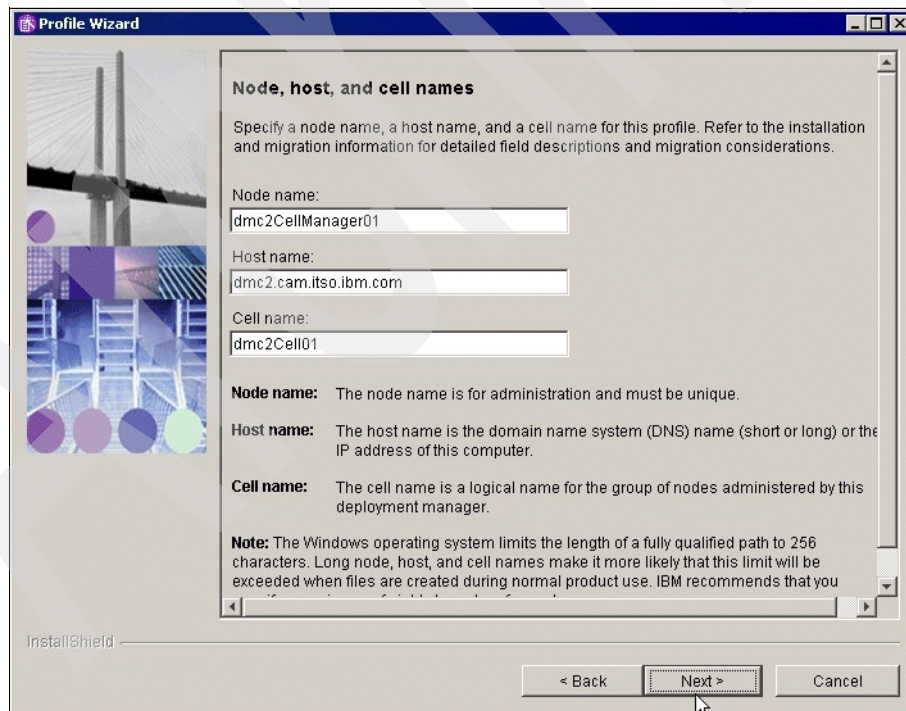


Figure 3-13 Node, host, and cell names selection window

11. Check the value of the Ports in the “Port value assignment” window. Click Next as shown in Figure 3-14 on page 63.

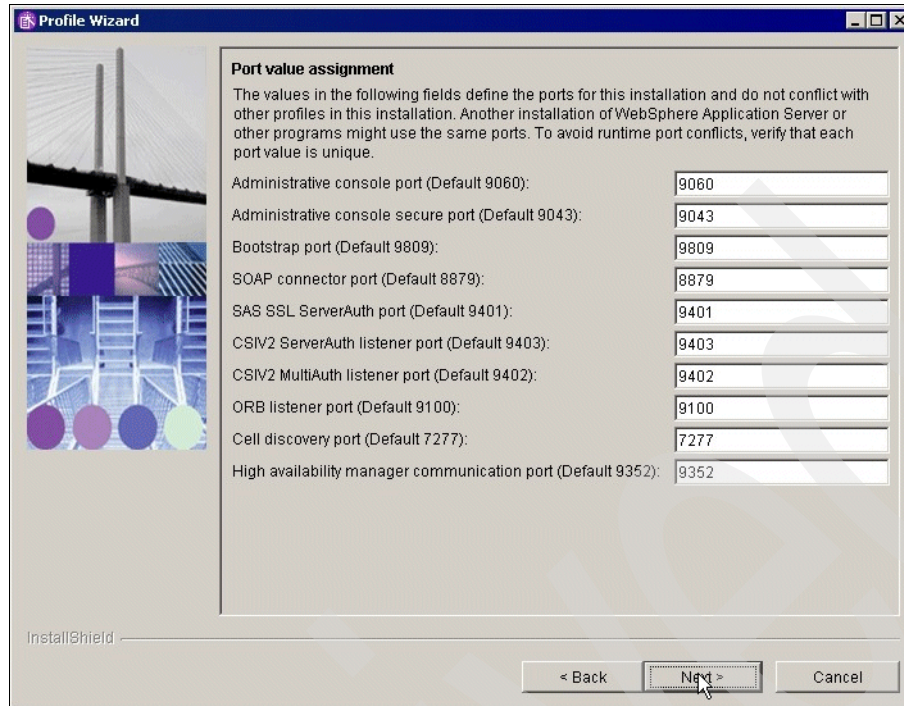


Figure 3-14 Ports selection window

12. In the “Windows Service Definition” window, you can configure WPS to run as a server. In our setup we do not want to run the WebSphere Process Server as a Windows service because of the longer start up time of the Windows system. So leave the defaults, and click Next as shown in Figure 3-15.

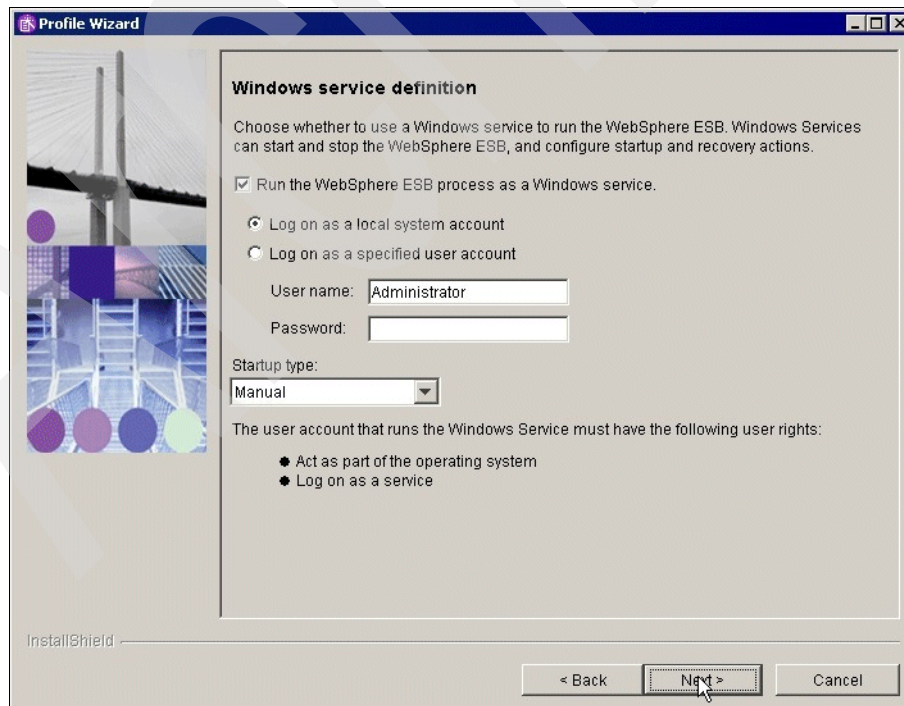


Figure 3-15 Windows service definition configuration window

13. In the “Services Component Architecture configuration” window, leave the defaults, and click Next.
14. In the “Profile Summary” window, check the values, and click Next as shown in Figure 3-16.

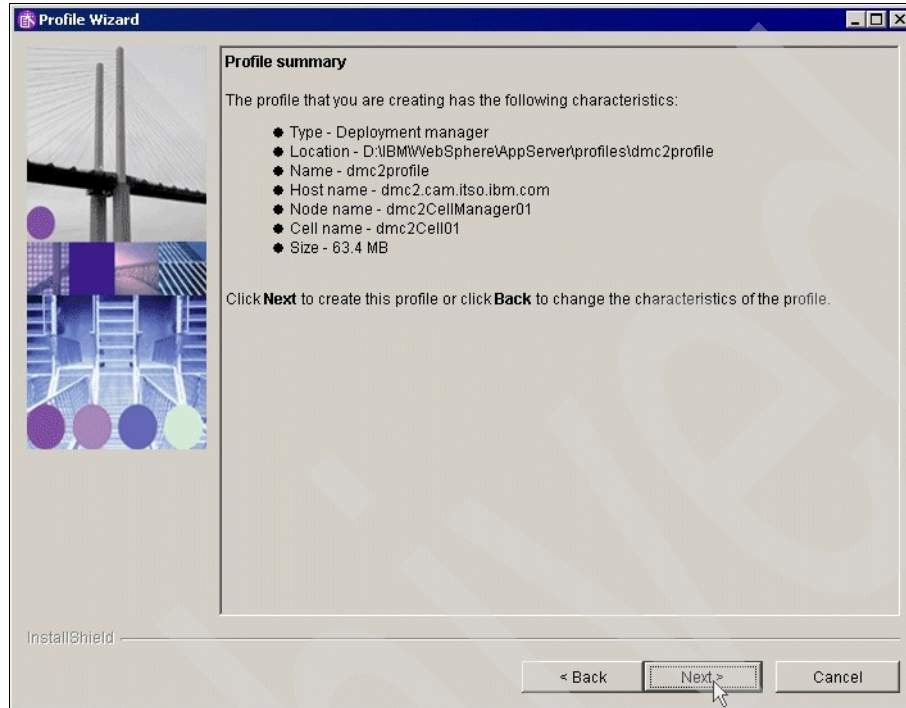


Figure 3-16 Profile summary window

15. In the “Profile creation is complete” window, check if the creation was successful. Click Finish as shown in Figure 3-17 on page 65.

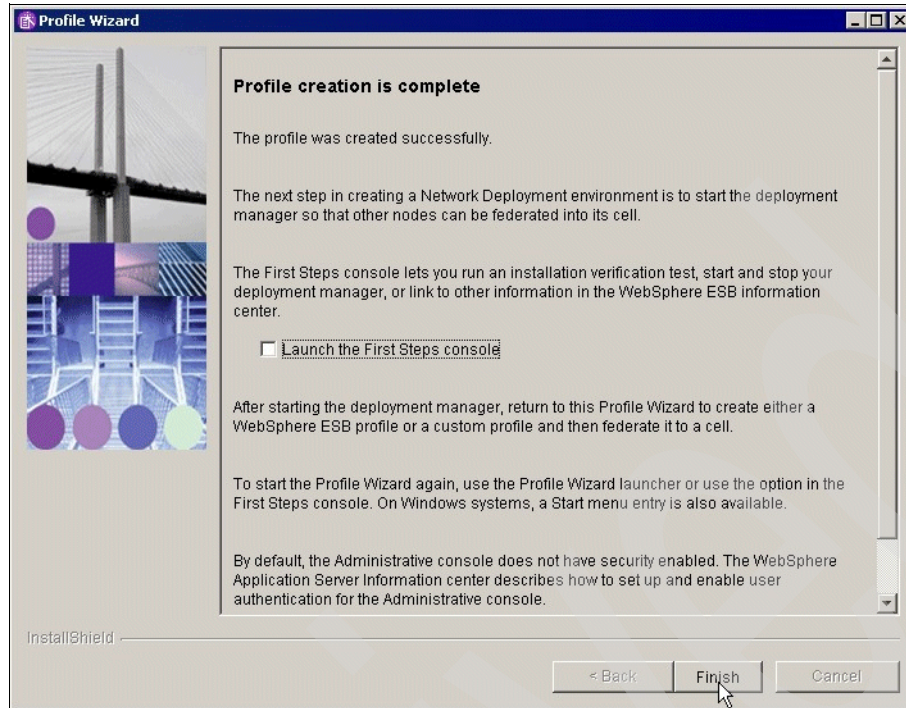


Figure 3-17 Profile creation window

Apply remaining WSAS and WPS iFixes for Portal 6.0.0.0

After the DMGR profile is created apply the remaining WSAS and WPS iFixes required for Portal 6.0.0.0. Apply these iFixes to the DMGR so that both the DMGR and the Portal nodes have the same level of WSAS and WPS, including the same iFixes.

1. Obtain the current WSAS update installer shipped with the CDs at
`<cd_root>/W-Setup/was6_fixes/win/updateinstaller.`
2. Copy the updateinstaller directory to [was_root]\AppServer\ of the file system.
3. Copy the necessary fixes to the [was_root]\AppServer\updateinstaller directory.

These fixes can be found on the Portal CDs at
`<cd_root>/W-Setup/wps6_fixes/win/updateinstaller/maintenance.` Copy the maintenance dir to [was_root]\AppServer\UpdateInstaller.

- IY82199
- IY83206
- JR23774
- JR24221
- JR24190

Hint: If the Installation of the JR24190 fix fails, then restart the Windows Server and uninstall and install the fix again.

Java SDK 1.4.2 SR4 Cumulative Fix – this JDK™ should be installed with the install. You can verify the level of the WSAS JDK by running the following command:

```
<was_root>/java/bin/java -version
```


If the Java SDK does not have the correct version, you can find the Cumulative Fix at the following Web site:

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg24011104>

Copy the maintenance directory to WebSphere\AppServer\UpdateInstaller\sdk and use the Update Installer that is already copied.

4. Start the WSAS update installer by running the <wsas_root>/updateinstaller/update.exe to apply the remaining WSAS and WPS iFixes that are required for Portal, but were not installed as part of the WSAS install.

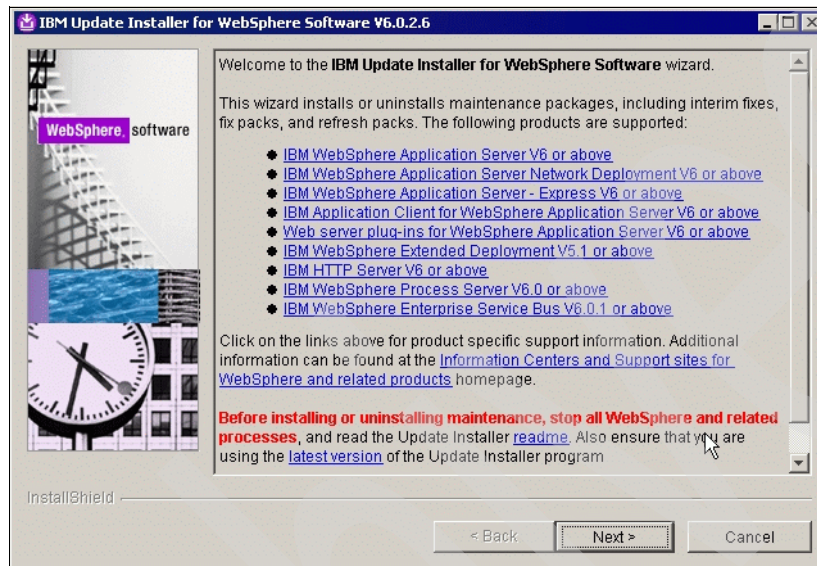


Figure 3-18 Application Server Update Installer welcome window

5. Type the directory location of the AppServer into the Directory Name field. Click Next as shown in Figure 3-19.

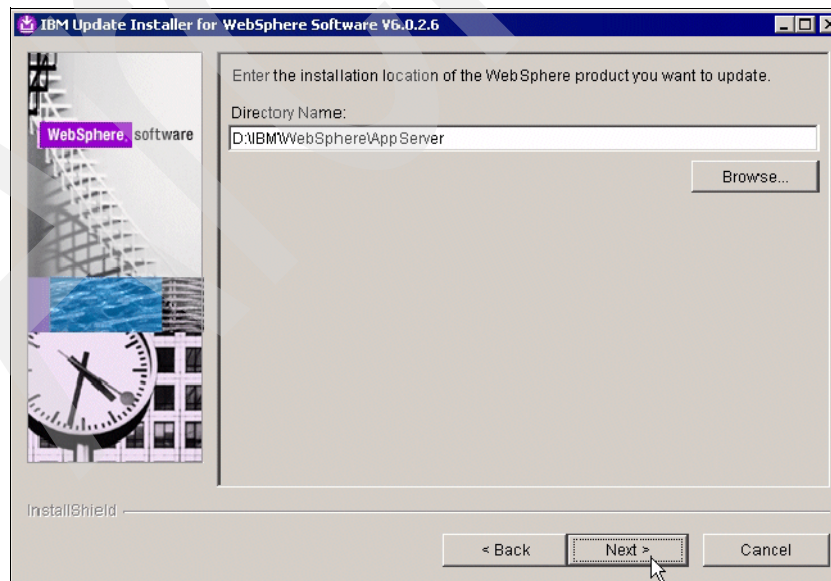


Figure 3-19 Choose the AppServer directory name window

6. Select **Install Maintenance Package**, and click Next as shown in Figure 3-20 on page 67.

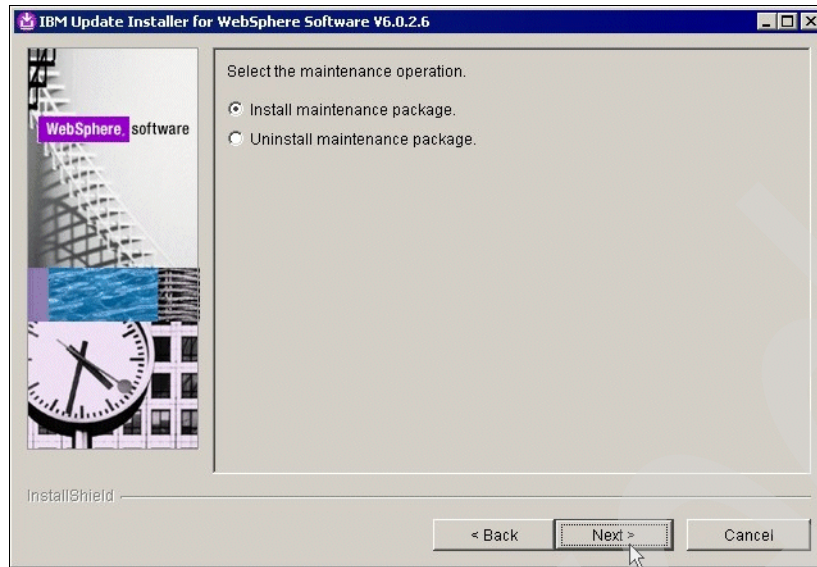


Figure 3-20 Maintenance package to install window

7. Browse for the first maintenance package to install, and click Next as shown in Figure 3-21. The first one should be IY82199.

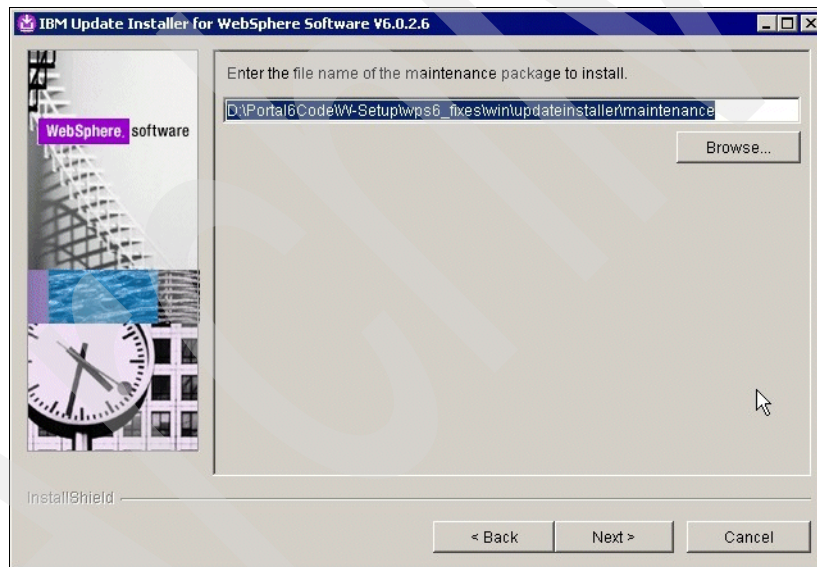


Figure 3-21 Fix selection window

8. In the "Confirmation window", verify the maintenance Package, and click Next as shown in Figure 3-22 on page 68.

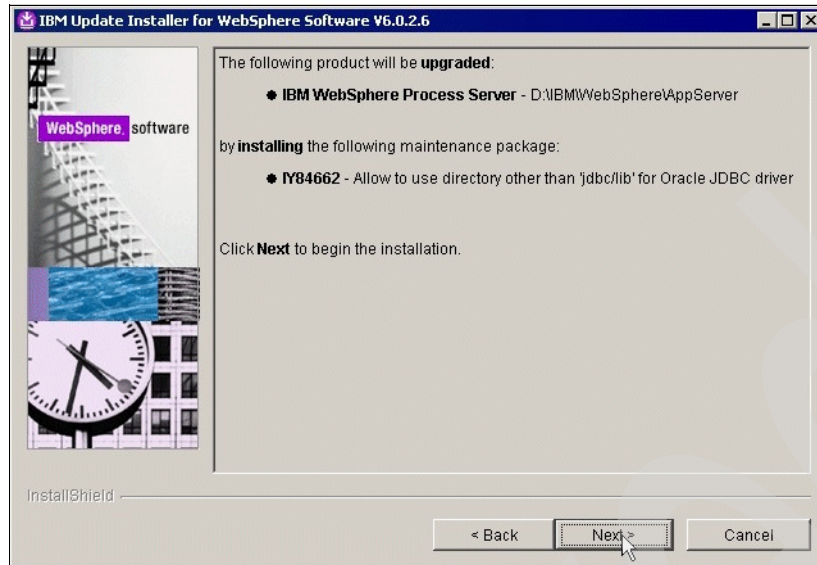


Figure 3-22 Confirmation window

9. In the “Success window”, verify that the maintenance package installed successfully. Click Relaunch to start the Update Installer again for the next maintenance package, as shown in Figure 3-23.

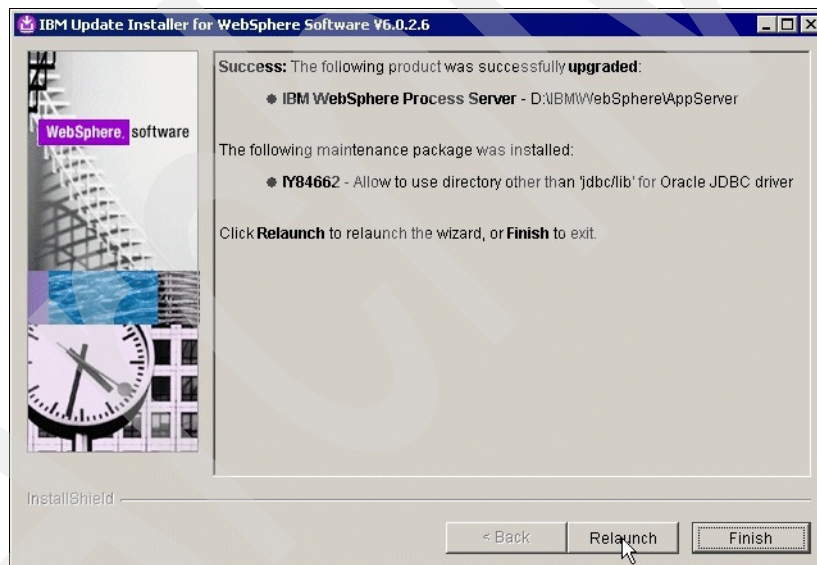


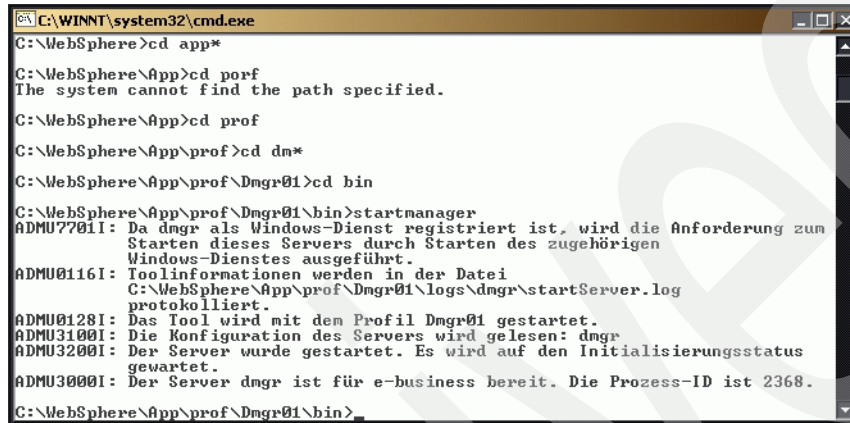
Figure 3-23 Fix successfully installed window

Important: Fixes are installed one at a time by the Portal Update Installer. Repeat steps 5-9 for each maintenance package listed in step number 3 (Copy the necessary fixes to the [was_root]\AppServer\updateinstaller directory.) on page 65.

Test your Deployment Manager installation

Use the following steps to test your Deployment Manager installation:

1. Open a command prompt and navigate to the following location:
`\IBM\WebSphere\AppServer\profiles\<profilename>\bin`
2. Start Deployment Manager by entering the following command:
`startmanager`
3. After the server successfully starts, you should see a window similar to Figure 3-24.



```
C:\WINNT\system32\cmd.exe
C:\WebSphere>cd app*
C:\WebSphere\app>cd porf
The system cannot find the path specified.
C:\WebSphere\app>cd prof
C:\WebSphere\app\prof>cd dm*
C:\WebSphere\app\prof\dmgr01>cd bin
C:\WebSphere\app\prof\dmgr01\bin>startmanager
ADMU7701I: Da dmgr als Windows-Dienst registriert ist, wird die Anforderung zum
Starten dieses Servers durch Starten des zugehörigen
Windows-Dienstes ausgeführt.
ADMU0116I: Toolinformationen werden in der Datei
C:\WebSphere\app\prof\dmgr01\logs\dmgr\startServer.log
protokolliert.
ADMU0128I: Das Tool wird mit dem Profil Dmgr01 gestartet.
ADMU3100I: Die Konfiguration des Servers wird gelesen: dmgr
ADMU3200I: Der Server wurde gestartet. Es wird auf den Initialisierungsstatus
gewartet.
ADMU3000I: Der Server dmgr ist für e-business bereit. Die Prozess-ID ist 2368.
C:\WebSphere\app\prof\dmgr01\bin>
```

Figure 3-24 DMGR started

4. Open a Web browser, and type the Web address for your Deployment Manager, for example:
`http://<fullyqualifiedhostname>:9060/ibm/console`
The port number is printed out at the end of the DMGR install, or refer to the SystemOut.log.
You should see the Login window of the Deployment Manager Admin console as shown in Figure 3-25 on page 70.

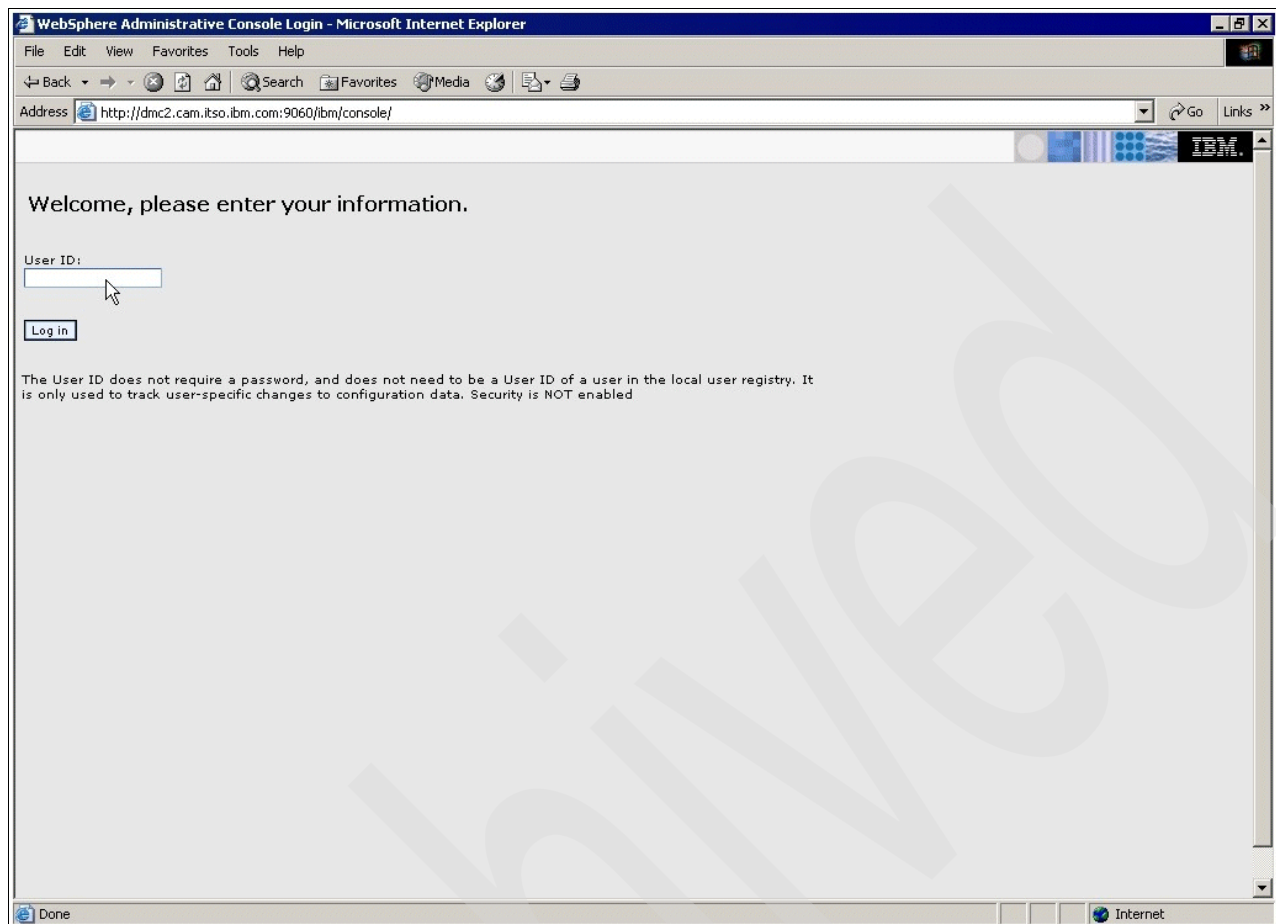


Figure 3-25 Deployment Manager welcome/login window

The default port for the WSAS administrative console changed to 9060 in WSAS 6.x.

3.4.2 Install WSAS and WPS on the primary cluster node, node1

This book explicitly defines the required approach to build a Portal cluster that was installed on WebSphere Process Server. To do this you must install Portal into an already federated WSAS and WPS profile. Because of this requirement, we MUST install WSAS and WPS from their native installers and federate the node BEFORE using the Portal installer to install Portal.

Important: At this time please install only the PRIMARY NODE (node 1).

Note: If you are using the same machine for the DMGR and node1 you are not required to install another instance of WSAS. You only need to create an additional profile on the existing instance of WSAS that was installed previously for the DMGR profile. If only creating an additional profile is what you require, then please start the Profile Wizard manually. The WPS profile creation wizard script is located at `<wsas_root>/bin/ProfileCreator_wbi/pcatWindows.exe`. Proceed with "Step 7 After the profile creation wizard is launched."

Install WebSphere Application Server on node 1

1. Install WSAS on node 1 by running the installer from the following location:
`<cd_root>/W-1/windows/ia32/ifpackage/WAS/`

Note: Make sure the installer window is titled “Welcome to IBM WebSphere Application Server Network Deployment, V6”. This title means that you can use this installer to install either, DMGR or WSAS profiles. If the title is “WebSphere Application Server Version 6.0”, you are using an installer that only has the ability to install WSAS profiles and not DMGR profiles

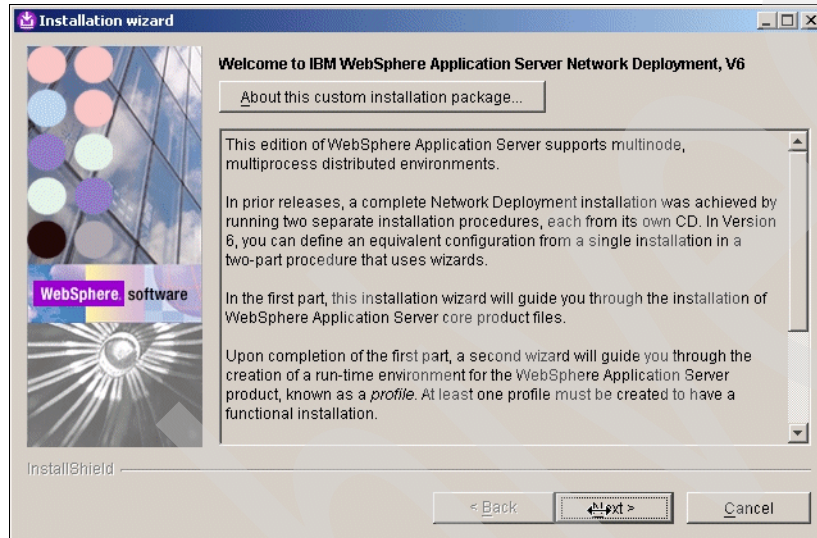


Figure 3-26 Welcome to IBM WebSphere Application Server

2. In the “Accept the license agreement” window, accept the license, and click Next.
3. The Installation Wizards performs a pre-requisite check for you when this runs successfully. Click Next; otherwise, make sure your operating system meets the requirements for WSAS as discussed in Section 2.1, “System requirements” on page 22.
4. If installing on Windows, when asked for the install location, please shorten the default path. There is a path name limitation in Windows. Windows cannot handle path names longer than 256 characters. For example:
D:\IBM\WebSphere\ApplicationServer
In a production environment we recommend using the shortest path name that is possible. Click Next as shown in Figure 3-27 on page 72.

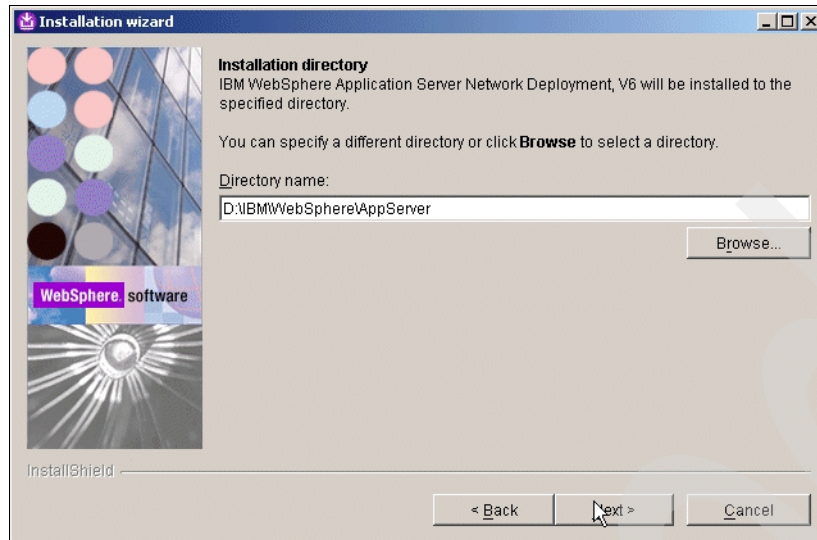


Figure 3-27 Installation directory window

5. The Installation Wizard displays the installation summary including interim fixes to be applied. Review the options in the installation summary window, and click Next.

Note: The WSAS installer from the Portal CDs automatically upgrades WSAS to 6.0.2.9.

The WSAS installer from the Portal CDs also automatically applies the following WSAS and WPS iFixes required for an install of Portal 6.0.0.0:

- PK21578
- PK22180
- PK25804
- PK25597
- PK23535
- PK24146
- PK27024
- PK21870
- PK21793
- PK26123
- PK25630
- PK24463
- PK21704
- PK24014
- PK21927
- PK18492
- PK20404
- PK23985
- PK23358

Note: You can find WSAS fixes by accessing the following Web site, and performing a search for the fix:

<http://www-306.ibm.com/software/webservers/appserv/was/support/>

6. In the “Installation complete” window, ensure that the “Launch the Profile creation wizard” check box remains **UNCHECKED**. We will create a WPS profile at the end of the WPS install. Click **Next** as shown in Figure 3-28.

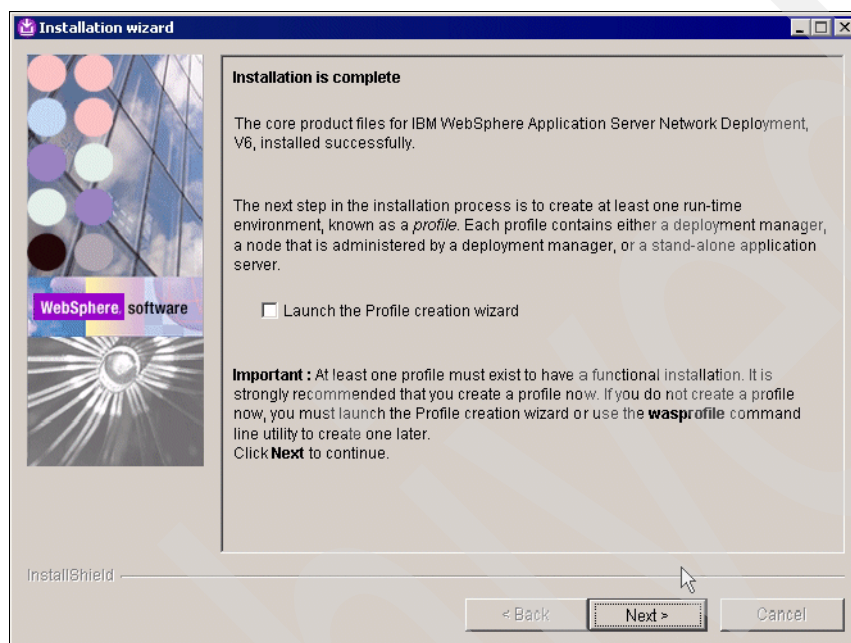


Figure 3-28 Installation is complete window

7. Click **Finish** to close the Installation wizard.

Install WebSphere Process Server on node 1

1. Install WPS 6.0.1.1 by running the installer from the following location:
`<cd_root>/W-2/windows/ia32/WBI/install.bat`

Important: Use the install.bat file and NOT the install.exe to install WPS.

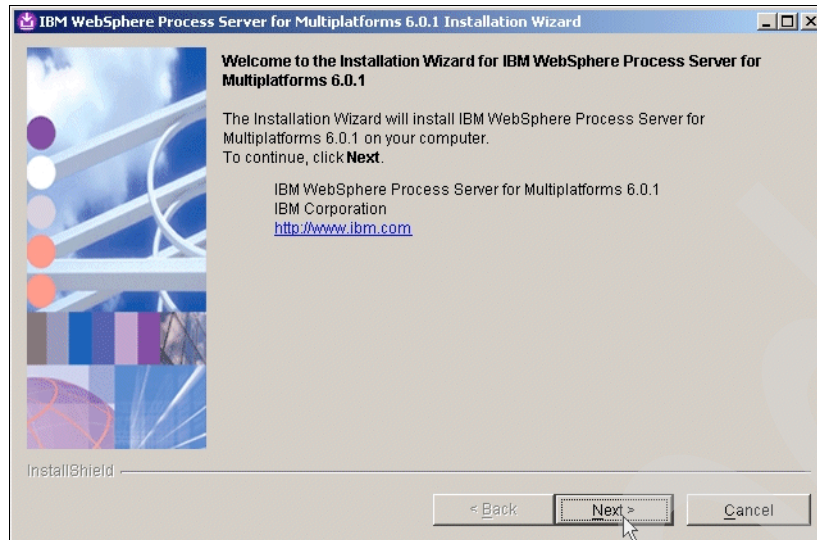


Figure 3-29 Welcome to WebSphere Process Server installation

2. The Installation Wizard performs a pre-requisite check for you when this runs successfully. Click Next; otherwise, ensure that your operating system meets the requirements for WSAS as discussed in Section 2.1, “System requirements” on page 22.
3. In the “Detected WebSphere Application Server” window, ensure that you use the existing WSAS you just installed, and click Next as shown in Figure 3-30.

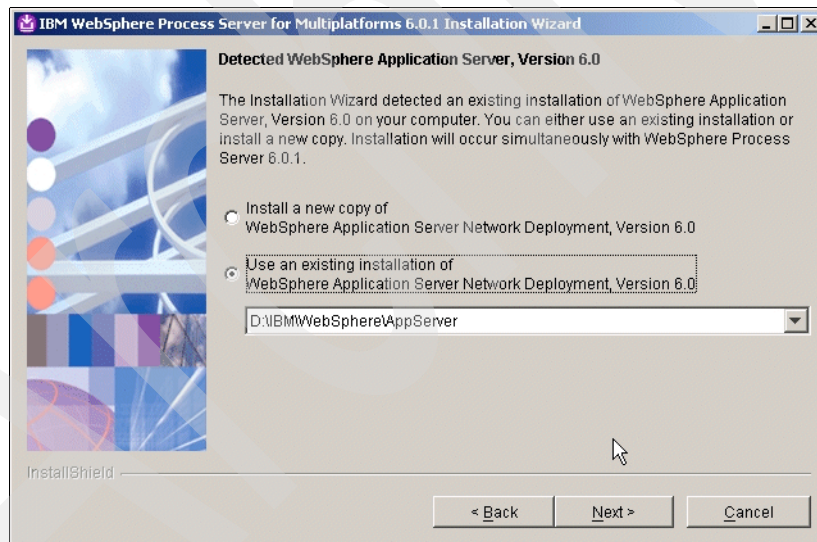


Figure 3-30 Detected WebSphere Application Server

4. Review the options in the installation summary window, and click Next.
5. In the “Installation complete” window, we want to create a profile. Select the “Launch the Profile Wizard” check box, and click Next to launch the WPS profile creation wizard as shown in Figure 3-31 on page 75.

Note: If you have to launch the WPS profile creation wizard manually, make sure that you launch the WPS profile creation wizard and NOT the WSAS profile creation wizard. The WPS profile creation wizard script is located at the following location:
<wsas_root>/bin/ProfileCreator_wbi/pcatWindows.exe

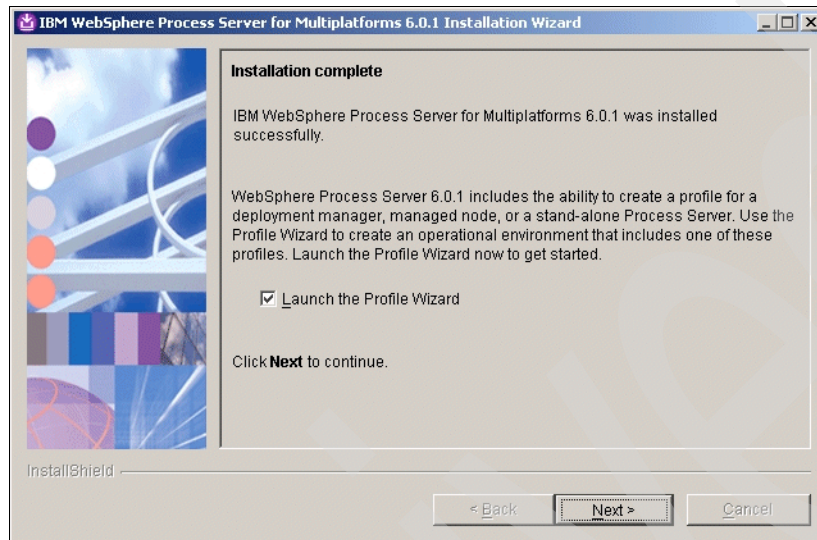


Figure 3-31 Launch Profile wizard

6. In the “Welcome to the profile creation wizard” window, click Next.
7. After the profile creation wizard is launched, select the “Custom profile” radio button on the “Profile type selection” panel, and click Next.

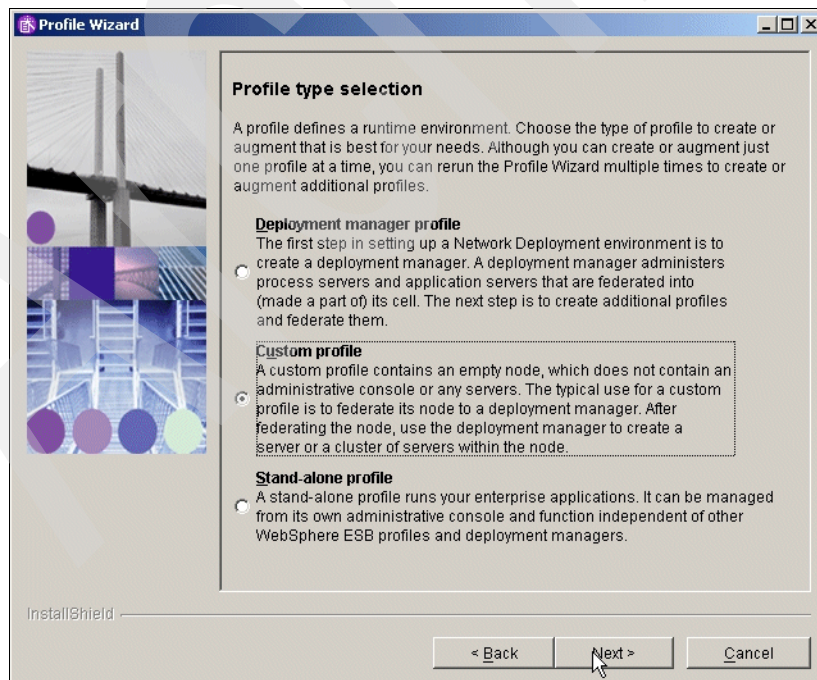


Figure 3-32 Profile type selection window

8. In the “Federation” window, you decide if you want to have the profile creation wizard automatically federate the Custom profile after creation. Allow the profile creation wizard to federate the Custom profile. To do this, **UNCHECK** the “Federate this node later using the addNode command” check box.

Specify the Full Qualified Hostname of the Deployment Manager Server, for example:

dmc2.cam.itso.ibm.com

Also, specify the SOAP Port of the Deployment Manager. The default value is 8879, and click Next as shown in Figure 3-33.

Note: Make sure that the clocks are synchronized to within five minutes of each other on the node 1 machine and the DMGR machine. If the clocks are not within five minutes, the addNode process will fail. Also check if there is an active firewall in the network and if the SOAP Port is accessible. If addNode process fails, fix the problem, and start it again.

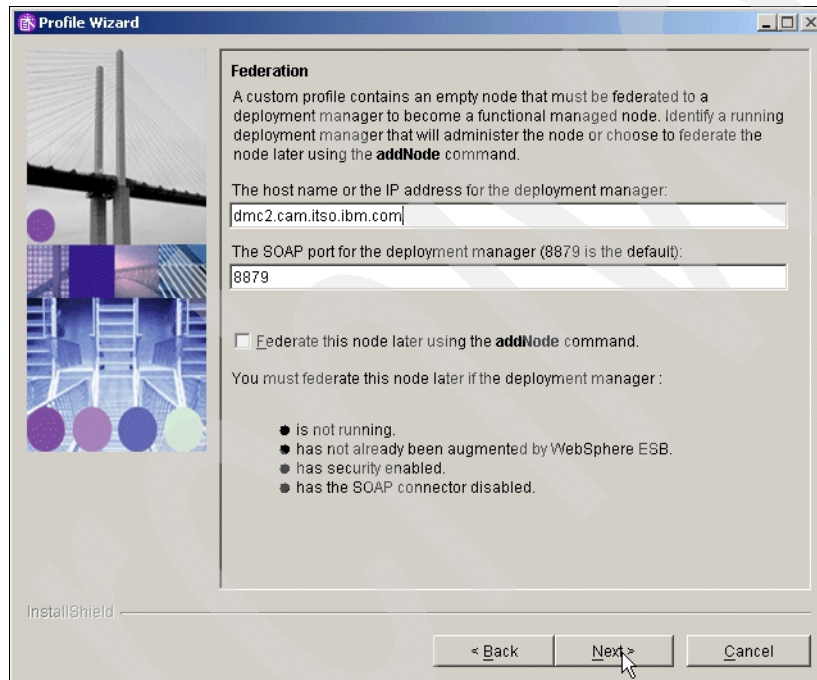


Figure 3-33 Federation window

9. In the “Profile name” window, type a profile name for the WebSphere Process Server Instance. This name can be different for each node and the DMGR. In our setup we chose the node name as the profile name. Click Next as shown in Figure 3-34 on page 77.

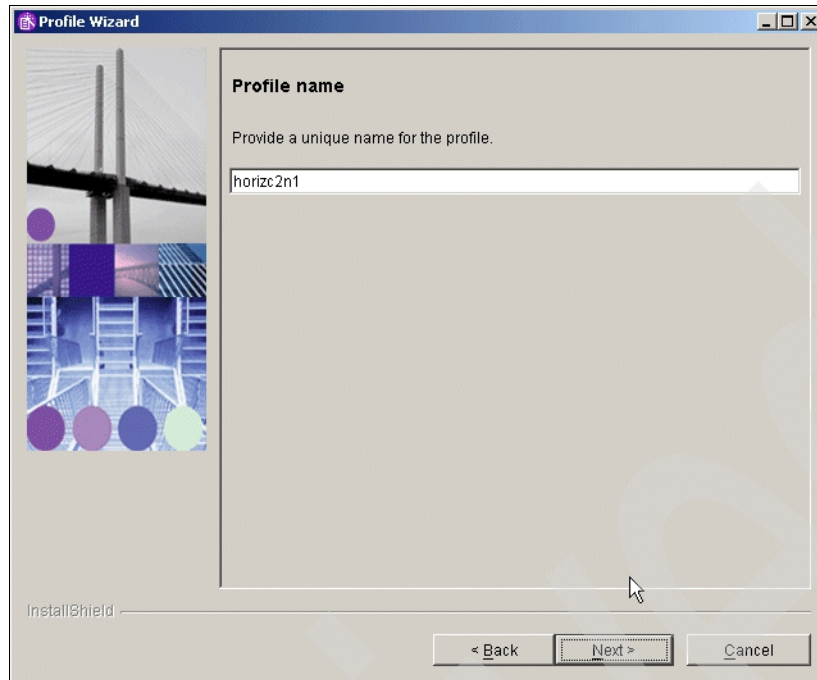


Figure 3-34 Profile Name window

10. In the “Profile directory” window, verify the directory name. In a production environment we recommend using the shortest path name that is possible. Click Next as shown in Figure 3-35.

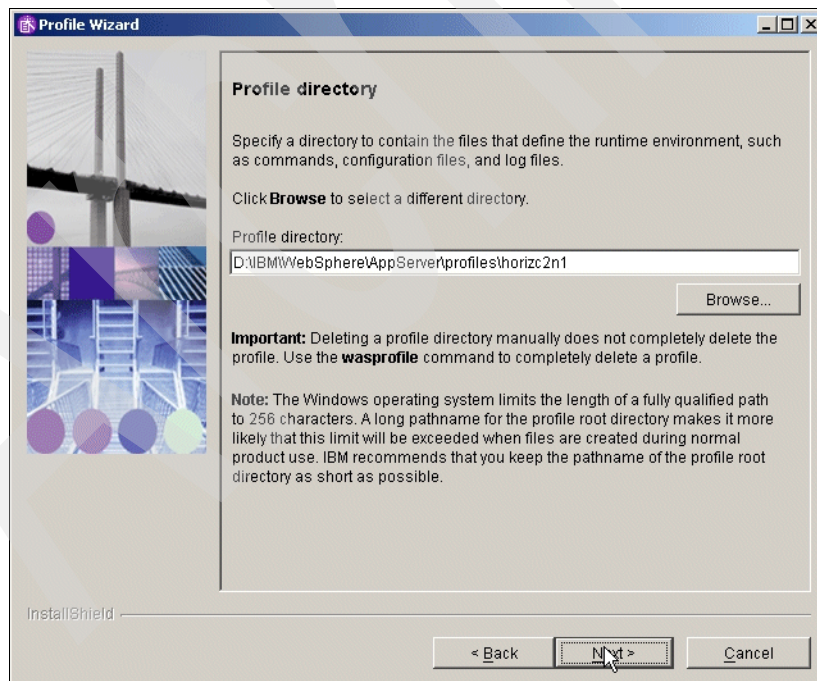


Figure 3-35 Profile directory window

11. In the “Node and host Names” window, check the value and length of node. We recommend only eight characters and Hostname. Click Next as shown in Figure 3-36 on page 78.

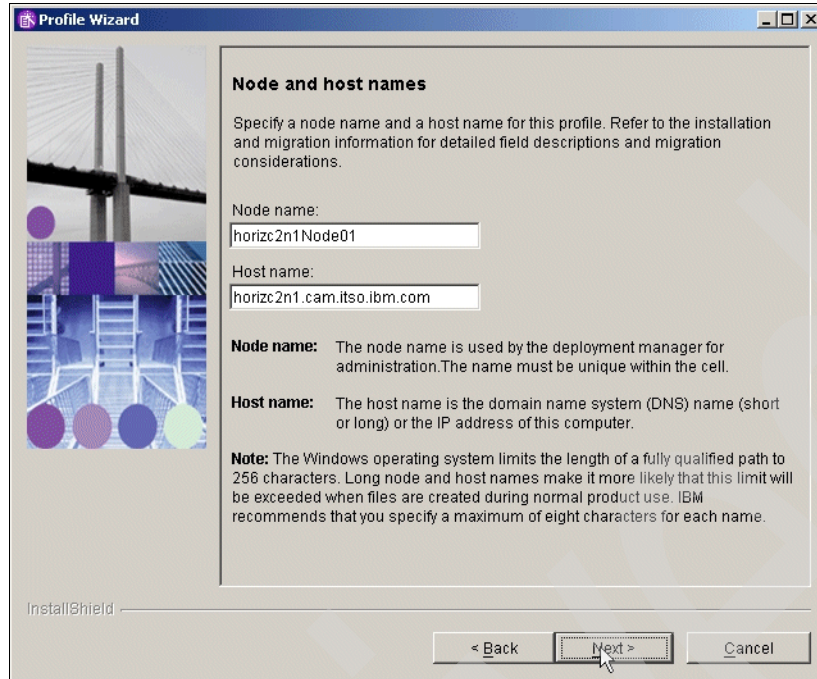


Figure 3-36 Node and Hostname

12. In the “Port value assignment” window, check the values of Ports, and click Next as shown in Figure 3-37.

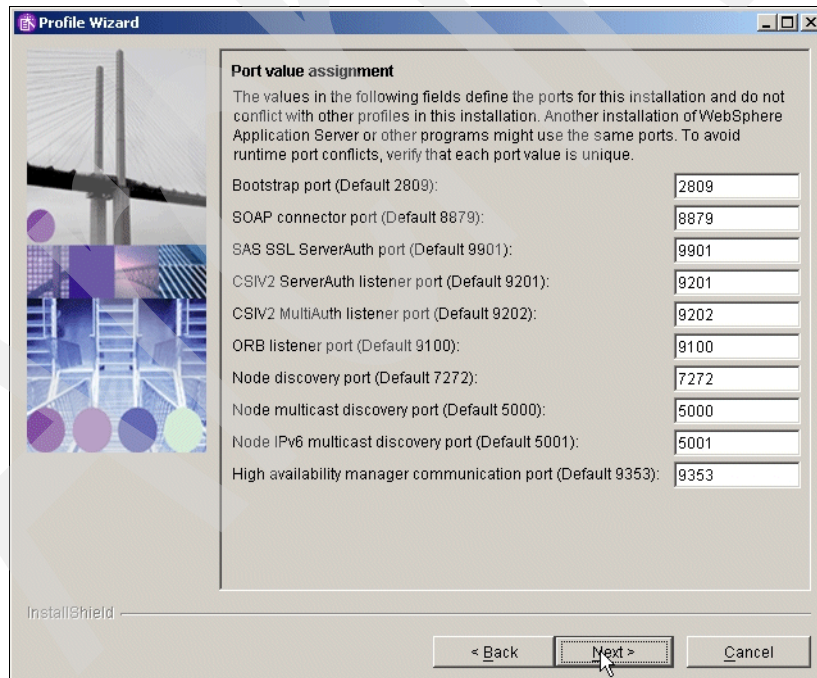


Figure 3-37 Port value assignment window

13. In the “Profile Summary” window, check the values, and then click Next.
14. In the “Profile creation is complete” window, check if the creation was successful, and click Finish.

15. For validation of the successful Federation, log on to the Admin Console of the DMGR.

- a. Click System administration → Cells → Local Topology. You should see the federated node in the list as shown in Figure 3-38.



Figure 3-38 Verify federation

16. After the Custom profile is created and federated to the DMGR, apply the remaining WSAS and WPS iFixes required for Portal 6.0.0.0. This fix may change. Verify on the Support Web site for the actual fixes.

17. Obtain the current WSAS update installer shipped with the CDs at the following location:
<cd_root>/W-Setup/was6_fixes/win/updateinstaller

18. Copy the updateinstaller directory to [was_root]\AppServer\ of the file system.

19. Copy the necessary fixes to the [was_root]\AppServer\updateinstaller directory.

These fixes can be found on the Portal CDs at the following location:

<cd_root>/W-Setup/wps6_fixes/win/updateinstaller/maintenance. Copy the maintenance dir to [was_root]\AppServer\UpdateInstaller.

- IY82199
- IY83206
- JR23774
- JR24221
- JR24190

Hint: If the installation of the JR24190 Fix fails, then restart the Windows Server, uninstall, and install the fix again.

Java SDK 1.4.2 SR4 Cumulative Fix— this JDK should be installed with the install. You can verify the level of the WSAS JDK by running the following command:

```
<was_root>/java/bin/java -version
```

If the Java SDK is not the correct version, you can find the Cumulative Fix at the following Web address:

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg24011104>

Copy the maintenance directory to WebSphere\AppServer\UpdateInstaller\sdk, and use the Update Installer that is already copied.

20. Start the WSAS update installer by running the <wsas_root>/updateinstaller/update.exe to apply the remaining WSAS and WPS iFixes that are required for Portal but were not installed as part of the WSAS install.
21. Enter the directory location of the AppServer, and click Next
22. Select **Install Maintenance Package**, and click Next.
23. Browse for the first maintenance package to install, and click Next. The first one should be IY82199.
24. In the “Confirmation window” verify the maintenance Package, and click Next.
25. In the “Success window” verify that the maintenance package installed successfully, and click Relaunch to start the Update Installer again for the next maintenance package.

Important: Fixes are installed one at a time by the Portal Update Installer. Repeat steps 20-24 for each maintenance package listed in step 19.

3.4.3 Prepare the DMGR and node1 for the Portal install

Use the following steps to prepare the DMGR and node1 for the Portal install:

1. Update the Deployment Manager machine with required WMM JAR files. These files are located on the Setup CD provided as part of the installation package for WebSphere Portal. Copy the following files from the <cd_root>/W-Setup/dmgr_wmmjars directory on the Setup CD to the /<wsas_root>/lib directory on the DMGR machine and to the primary node:
 - * wmm.jar
 - * wmm.ejb.jar
 - * wp.wire.jar

Important: If this is the first Portal node you will install into the cell, proceed to the next step, and continue with the primary node installation. If you already federated other managed nodes into the cell, you must also copy these JAR files to the /<wsas_root>/lib directory on each of those managed nodes, regardless of whether you intend to install WebSphere Portal on the nodes.

2. Change the time-out request for the Simple Object Access Protocol (SOAP) client for the DMGR and the node 1. The default, in seconds, is 180.
 - a. On the DMGR machine, locate the <dmgr_profile_root>/properties/ directory and edit the soap.client.props file. Change the line to the following:
`com.ibm.SOAP.requestTimeout=6000`
 - b. On the WSAS node1 machine, locate the <wsas_profile_root>/properties/ directory and edit the soap.client.props file. Change the line to the following:
`com.ibm.SOAP.requestTimeout=6000`
3. Ensure that the nodeagent is running on node1 so that the following changes are synchronized to the node.

Note: You can start or stop the nodeagent by running the following commands in the IBM/WebSphere/Appserver/profiles/<Profile_Name>/bin

To start the node agent, use the following:

startnode.bat

To stop the node agent, use the following:

stopnode.bat

4. Log in to the DMGR AdminConsole, and change the time-out values for the DMGR by navigating to the following location:

System Administration → Deployment Manager → Web container transport chains

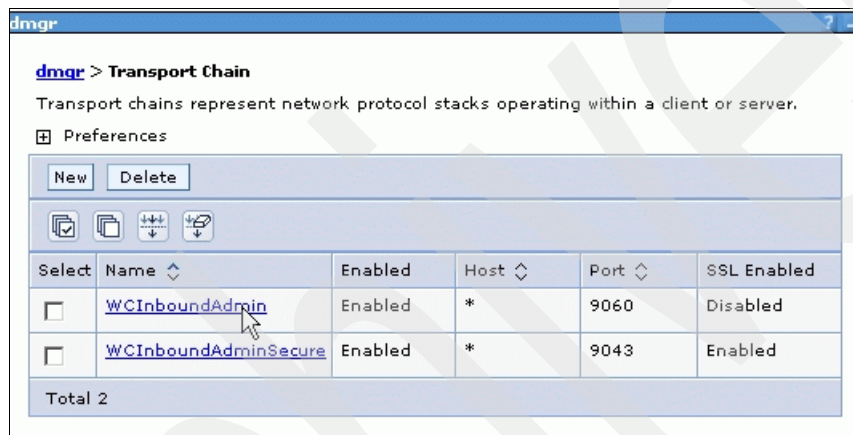


Figure 3-39 Transport chain

5. Increase the timeout values for each entry listed in the Web container transport chains section by clicking each entry. After clicking on an entry, complete the following steps to increase the timeout values:
 - a. Click HTTP Inbound Channel.
 - b. Change the Read timeout value to 180.
 - c. Change the Write timeout value to 180.

dmgr > Transport Chain > WCInboundAdminSecure > HTTP Inbound Channel (HTTP_2)
Channel for handling inbound HTTP requests from a remote client.

Configuration

General Properties	Additional Properties
* Transport Channel Name HTTP_2	■ Custom Properties
Discrimination weight 10	
* Maximum persistent requests 100	■ HTTP error and NCSA access logging
<input checked="" type="checkbox"/> Use persistent (keep-alive) connections	
* Read timeout 180 seconds	
* Write timeout 180 seconds	
* Persistent timeout 30 seconds	
<input type="checkbox"/> Enable access and error logging	

Figure 3-40 HTTP inbound channel

- d. Click Apply, and change the settings for the second entry, WCInboundAdmin.
- e. Save your configuration changes as shown in Figure 3-41.

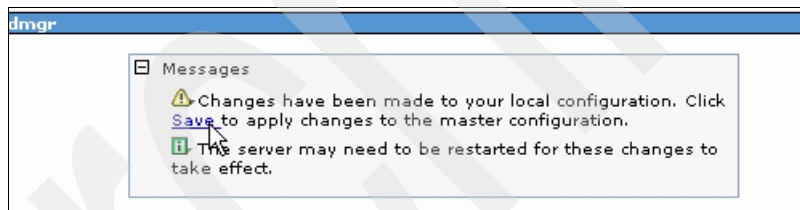


Figure 3-41 Save changes

- f. Synchronize with the node as shown in Figure 3-42.

dmgr

dmgr > Transport Chain > WCInboundAdminSecure > HTTP Inbound Channel (HTTP_2) >
Save

Save your workspace changes to the master configuration

Click Save to update the master repository with your changes. Click Discard to discard your changes and begin work again using the master repository configuration. Click Cancel to continue working with your changes.

☒ Total changed documents: 1

☒ Synchronize changes with Nodes

Save **Discard** **Cancel**

Figure 3-42 Sync with node

- g. Check if the synchronization completed successfully as shown in Figure 3-43.

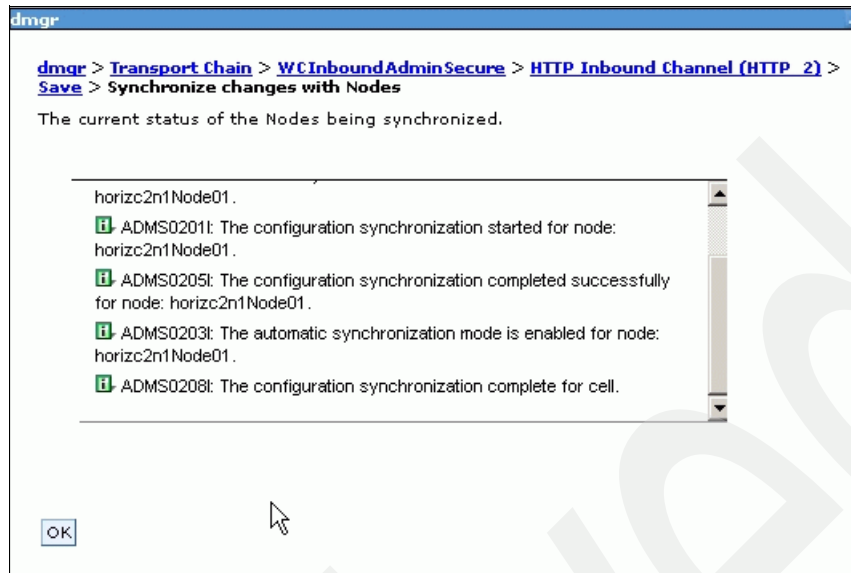


Figure 3-43 Sync with nodes was successful

6. Change the timeout request period for the Java Management Extensions (JMX™) connector.
- Click System administration → Deployment Manager → Administration Services → JMX connectors → SOAPConnector → Custom Properties.
 - Select the requestTimeout property, and increase the value from 600 to 6000. Click Apply as shown in Figure 3-44.

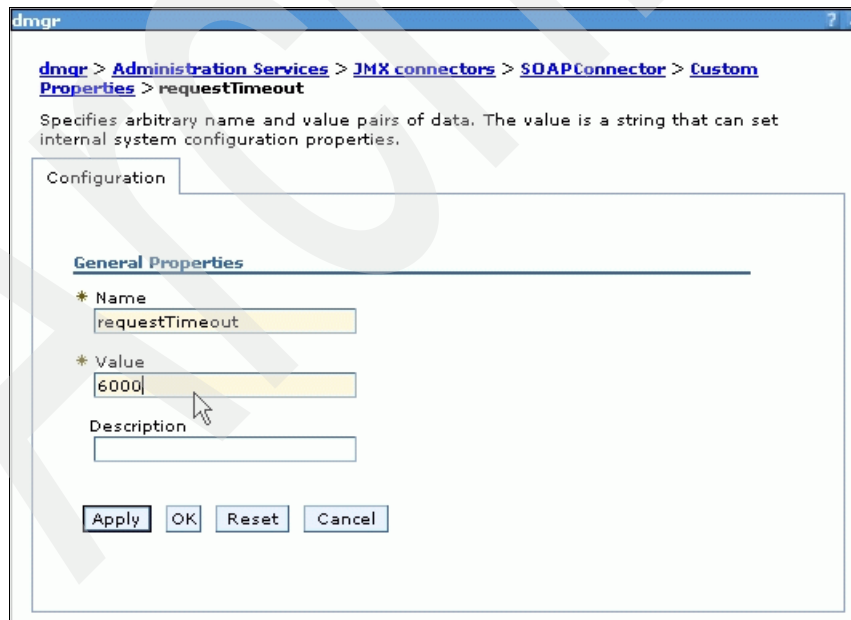


Figure 3-44 Java Management Properties

- Save your configuration changes, and synchronize with the node.
7. Disable automatic synchronization between this node and the DMGR.

- a. Click System Administration → Node Agents → nodeagent name for desired node → File synchronization service.
- b. Ensure that the Automatic Synchronization check box is NOT checked. Click Apply as shown in Figure 3-45.

Node agents > nodeagent > File synchronization service

The file synchronization service runs in the deployment manager and node agents, and ensures that configuration changes made to the cell repository are propagated to the appropriate node repositories. Specify the settings to configure the file synchronization service.

Configuration

General Properties

☒ Enable service at server startup

* Synchronization interval
1

☐ Automatic synchronization

☐ Startup synchronization

Exclusions

Apply OK Reset Cancel

Figure 3-45 AutoSync disabled

- c. Save your changes, and synchronize with the node.
8. Restart the DMGR and the node agent.
- Open a command prompt and change to the following directory:
 \IBM\WebSphere\Appserver\profiles\<Profile_Name>\bin on the DMGR and on the node
- Type the following commands:

Command	Description
stopmanager	Stops the Deployment Manager
startmanager	Starts the Deployment Manager
stopnode	Stops the node agent on the primary node1
startnode	Starts the node agent on the primary node1

3.4.4 Install Portal onto the managed node, node1

1. Start the Portal installer from <cd_root>/W-Setup/install.bat.

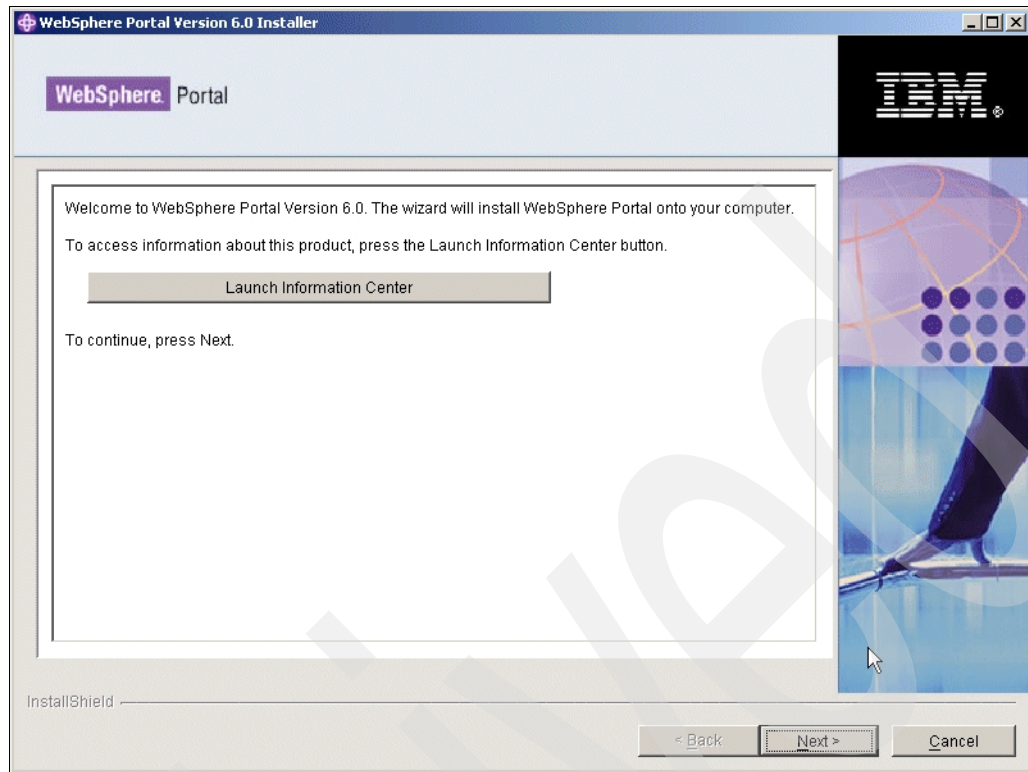


Figure 3-46 Portal Installer window

2. In the “Accept the license agreement” window, accept the license, and click Next.
3. In the “Installation Type” window, select **Custom** as the install type, and click Next as shown in Figure 3-47 on page 86.

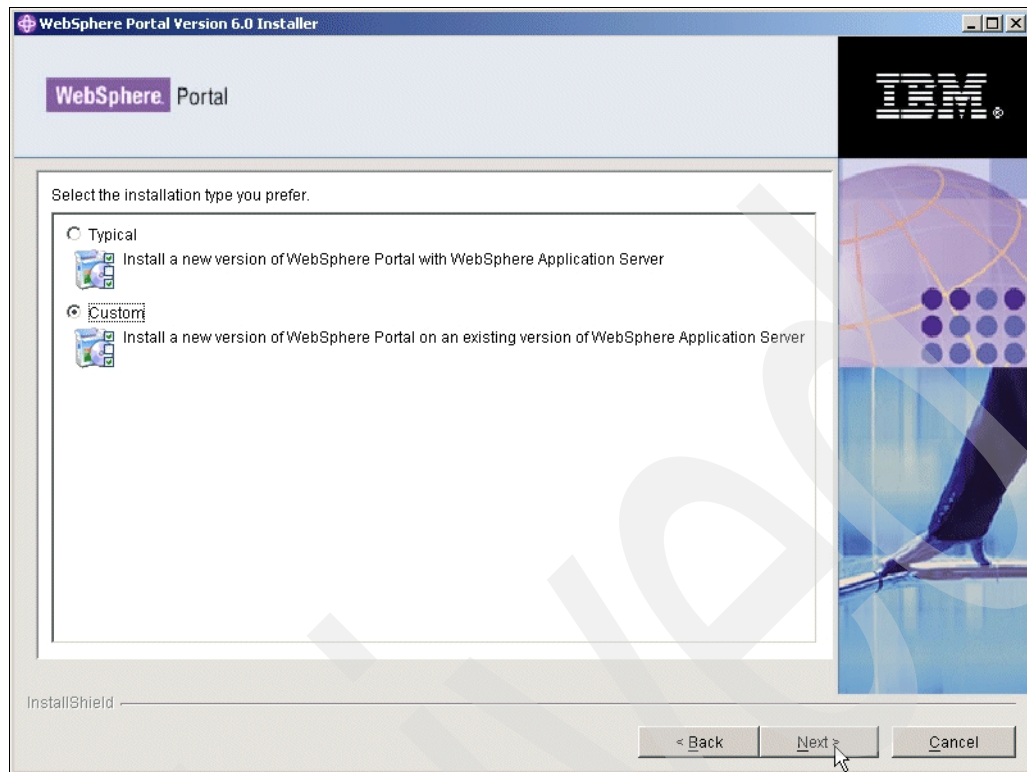


Figure 3-47 Installation type window

4. In the “Select the Location of the existing instance” window, select the existing WebSphere AppServer install location. Select **Install on a managed node**, and click Next as shown in Figure 3-48 on page 87.

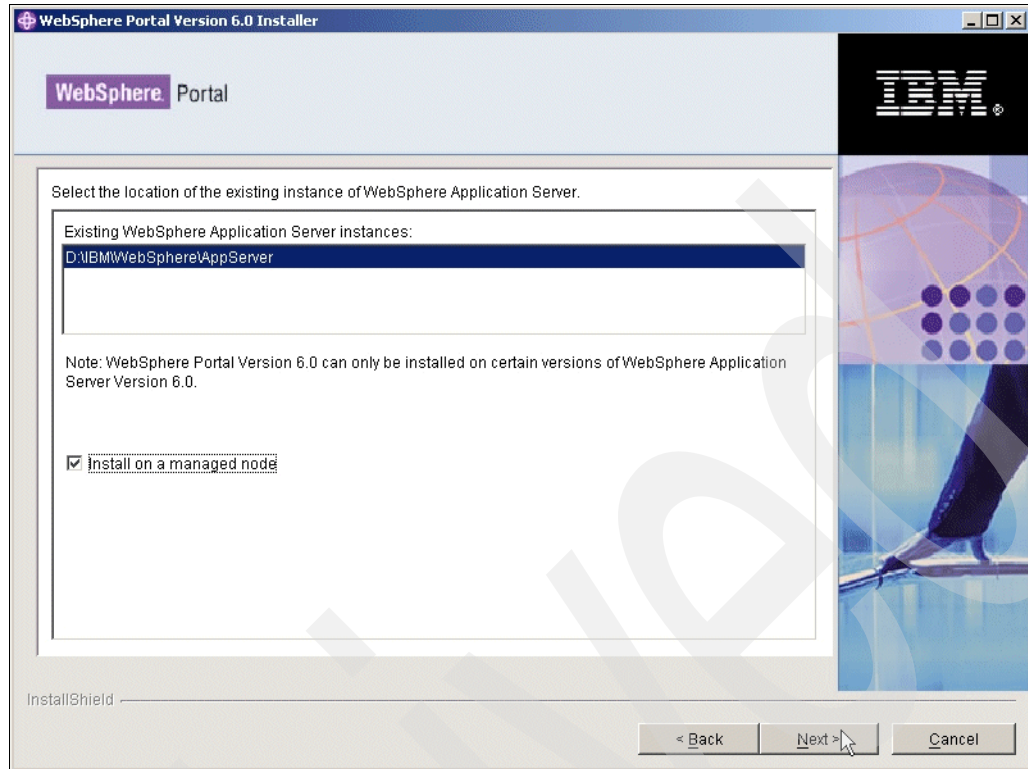


Figure 3-48 Select instance window

5. In the “Managed Node input” window, select **Primary Node**, and select the desired profile that you want to install Portal onto. Click Next as shown in Figure 3-49 on page 88.

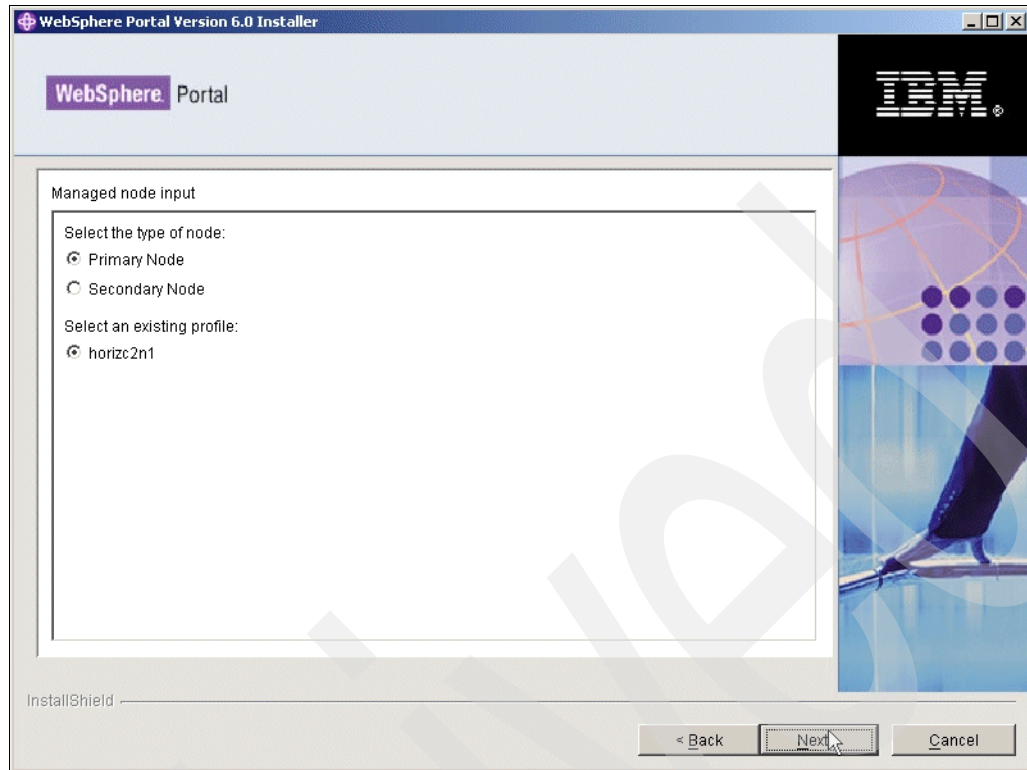


Figure 3-49 Node input window

6. In the “WebSphere Application Server Admin USER ID” window, define a WSAS Admin User and password. This is a new panel in the Portal installer because with Portal V6 the install enables security by default to the WMM database. You can use the default values like wasadmin and a password you want. If you switch to an LDAP Server in a later step or if your User Registry stays in WMM you should use a different user because of security issues. Click Next as shown in Figure 3-50 on page 89.

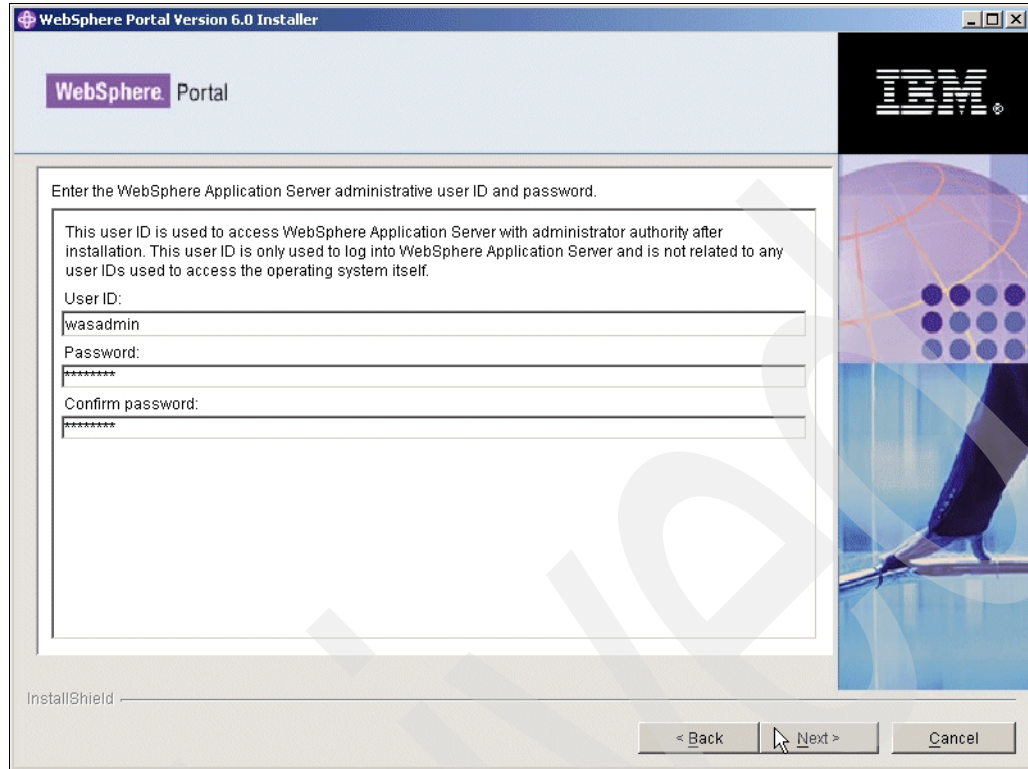


Figure 3-50 Wasadmin input

7. In the “WebSphere Portal install path” window, specify the location for Portal to be installed, and click Next.
8. In the “WebSphere Portal Server Admin User” window, define the Portal Admin User and password. You can use the default values like wpsadmin and a password you want. If you switch to an LDAP Server in a later step or if your User Registry should stay in WMM, you should use a different user because of security issues. Click Next as shown in Figure 3-51 on page 90.

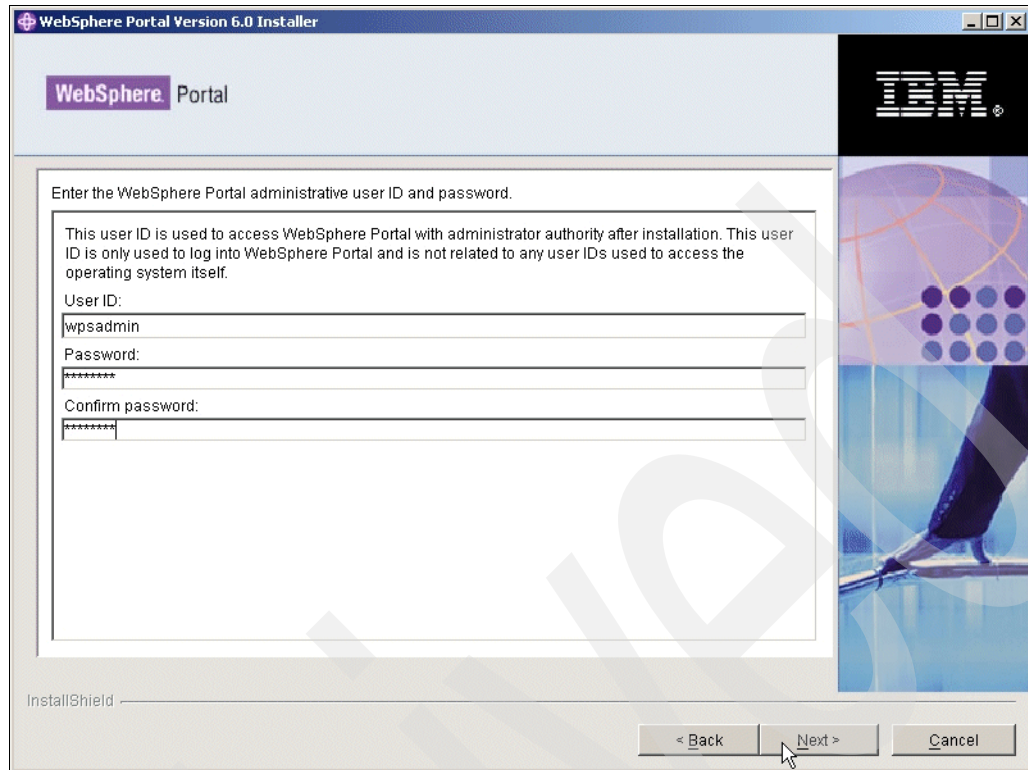


Figure 3-51 wpsadmin input window

9. In the “Select the products to run as a service” window, decide whether you want WSAS and Portal to run as a service. In this IBM book we choose NOT to run either as a Windows service because of the longer start up time for Windows with WP as a service.
10. In the “WebSphere Portal is ready to install” window, review the Summary panel, and click Next to begin the install.
11. When Installation successfully completes, notice the information like port number that is provided. Click Finish as shown in Figure 3-52 on page 91.

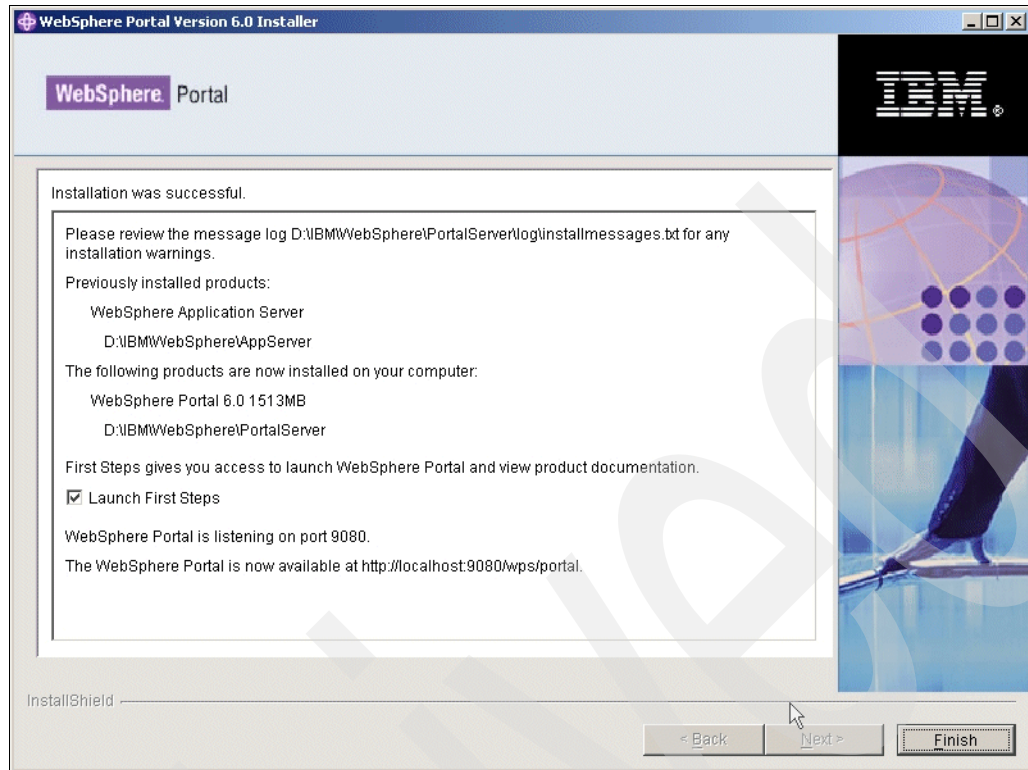


Figure 3-52 Installation completed

12. Verify the Portal install by accessing it through a browser and logging in with your Portal Admin User ID. A window similar to Figure 3-53 on page 92 is displayed. By default Portal is installed onto port 9080: <http://<fullyqualifiedhostname>:9080/wps/portal>.

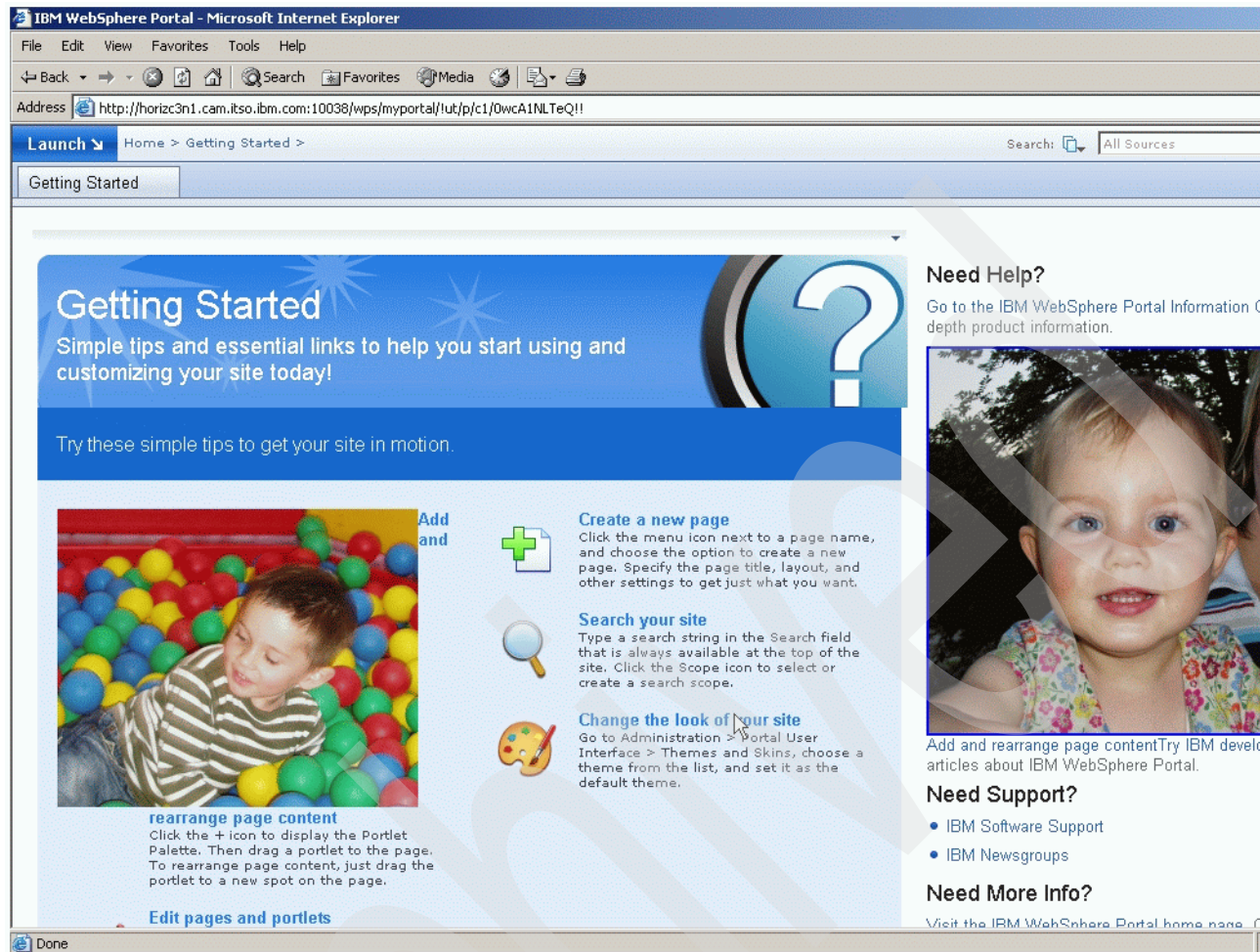


Figure 3-53 Portal window

13. After the Portal install completes successfully re-enable auto-sync.

Hint: During the Portal Installation the Security for Cell was enabled automatically. For login into the Admin Console, use your WAS Admin ID and password now.

Use the following steps to re-enable automatic synchronization between this node and the Deployment Manager:

- Log into the administrative console for the Deployment Manager.
- Click System Administration → Node Agents → node_name → File Synchronization Service.
- Select the Automatic Synchronization check box.
- Save your changes and synchronize with the node.
- Stop Portal using the following command:


```
stopserver WebSphere_Porta1
```
- Restart the node agent.

3.4.5 Configure primary node 1 to an external database

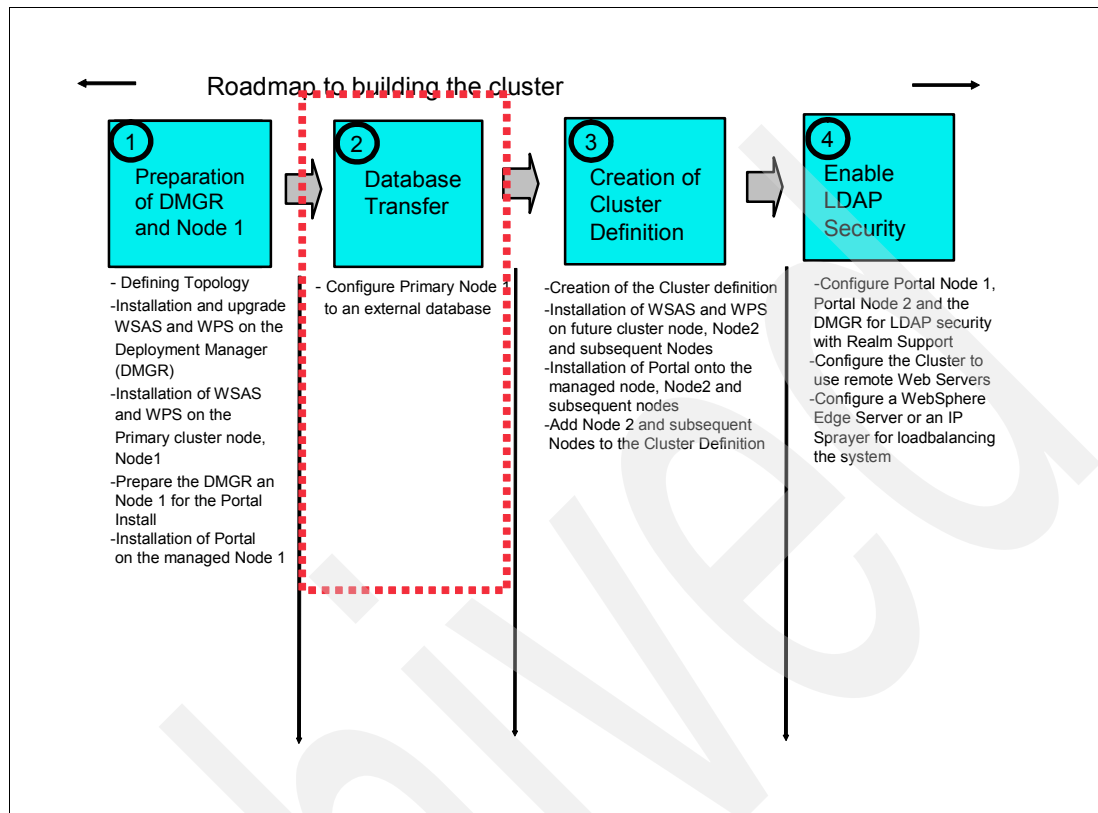


Figure 3-54 Flow chart installation steps: step 2

For detailed information about configuring the database, visit the following Web site:

http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/topic/com.ibm.wp.ent.doc/wpf/inst_start.html

In this scenario we only use one database, not the DB Domain Split. For instructions on how to configure WebSphere Portal Server with DB Domain Split refer to Chapter 5, "Database domains" on page 233.

Use the following steps to configure primary node 1 to an external database:

1. Ensure that the DMGR is running.
2. Install a supported db2 server on the remote machine dedicated to your data store.
3. Install DB2 Admin Client on primary node 1 and on the DMGR.

Note: You may also install the client on node 2 if you want at this point, but do not go on with the rest of the node 2 installation/configuration.

The DB2 Admin Client can be downloaded from the following Web site:

http://www-306.ibm.com/software/data/db2/udb/support/downloadv8_windows32bit.html

4. Run the following commands to create the Portal database on the db2 server:

Note: In Example 3-1 we use WPSDB as our database. Check the script also for your DB2 User Name and password.

Example 3-1 Command to create the Portal database on the db2 server

```
db2 "CREATE DB wpsdb using codeset UTF-8 territory us COLLATE USING UCA400_NO
PAGE SIZE 8192"
db2 "UPDATE DB CFG FOR wpsdb USING applheapsz 4096"
db2 "UPDATE DB CFG FOR wpsdb USING app_ctl_heap_sz 1024"
db2 "UPDATE DB CFG FOR wpsdb USING stmtheap 8192"
db2 "UPDATE DB CFG FOR wpsdb USING dbheap 2400"
db2 "UPDATE DB CFG FOR wpsdb USING locklist 1000"
db2 "UPDATE DB CFG FOR wpsdb USING logfilsiz 1000"
db2 "UPDATE DB CFG FOR wpsdb USING logprimary 12"
db2 "UPDATE DB CFG FOR wpsdb USING logsecond 20"
db2 "UPDATE DB CFG FOR wpsdb USING logbufsz 32"
db2 "UPDATE DB CFG FOR wpsdb USING avg_appls 5"
db2 "UPDATE DB CFG FOR wpsdb USING locktimeout 30"
db2 "CONNECT TO wpsdb USER db2admin USING password"
db2 "CREATE BUFFERPOOL ICMLSFREQBP4 SIZE 1000 PAGE SIZE 4 K"
db2 "CREATE BUFFERPOOL ICMLSVOLATILEBP4 SIZE 8000 PAGE SIZE 4 K"
db2 "CREATE BUFFERPOOL ICMLSMANBP32 SIZE 8000 PAGE SIZE 32 K"
db2 "CREATE BUFFERPOOL CMBMAIN4 SIZE 1000 PAGE SIZE 4 K"
db2 "CREATE REGULAR TABLESPACE ICMLFQ32 PAGE SIZE 32 K MANAGED BY SYSTEM USING
('ICMLFQ32') BUFFERPOOL ICMLSMANBP32"
db2 "CREATE REGULAR TABLESPACE ICMLNF32 PAGE SIZE 32 K MANAGED BY SYSTEM USING
('ICMLNF32') BUFFERPOOL ICMLSMANBP32"
db2 "CREATE REGULAR TABLESPACE ICMVFQ04 PAGE SIZE 4 K MANAGED BY SYSTEM USING
('ICMVQ04') BUFFERPOOL ICMLSVOLATILEBP4"
db2 "CREATE REGULAR TABLESPACE ICMSFQ04 PAGE SIZE 4 K MANAGED BY SYSTEM USING
('ICMSFQ04') BUFFERPOOL ICMLSFREQBP4"
db2 "CREATE REGULAR TABLESPACE CMBINV04 PAGE SIZE 4 K MANAGED BY SYSTEM USING
('CMBINV04') BUFFERPOOL CMBMAIN4"
db2 "CREATE SYSTEM TEMPORARY TABLESPACE ICMLSSYSTSPACE32 PAGE SIZE 32 K MANAGED
BY SYSTEM USING ('icmlssystspace32') BUFFERPOOL ICMLSMANBP32"
db2 "CREATE SYSTEM TEMPORARY TABLESPACE ICMLSSYSTSPACE4 PAGE SIZE 4 K MANAGED BY
SYSTEM USING ('icmlssystspace4') BUFFERPOOL ICMLSVOLATILEBP4"
db2 "DISCONNECT jcrdb"
db2 "TERMINATE"
```

5. Open a DB2 command line processor on the primary node 1 and the DMGR machine (Start → Programs → IBM DB2 → Command Line Tools → Command Line Processor).

- a. Type the following command:

```
db2 => catalog tcpip node nodename remote fullyqualifiedDB2servername server
port
```

Where *nodename* is the name of eight characters or fewer that you assign to the server, and *port* is the connection port used by DB2.

For example:

```
db2 => catalog tcpip node dbdom7 remote dbdom7.cam.itso.ibm.com server 50000
```

In this example, *dbdom7* is designated as the node name (to be used in the rest of the commands) and 50000 is the connection port (the default for TCP/IP connections). The

client always uses the same connection port as the DB2 server. You can verify the connection port for the DB2 server by opening a command prompt and issuing the **netstat -a** command on the DB2 Server Machine. You should see a port in the 50000 range.

- b. Type the following command (as appropriate) to catalog the databases that are hosted on the DB2 server.

```
db2 => catalog database databasename at node nodename
```

For example:

```
db2 => catalog database wpsdb at node dbdom7
```

Note: If you incorrectly catalog a database, you can uncatalog it by typing the following command:

```
db2=>uncatalog database database_name
```

- c. After you verify that all of the databases were correctly cataloged, test your connectivity to the database by typing the following command:

```
db2 => connect to databasename user DB2administratorname using password
```

For example:

```
db2 => connect to wpsdb user db2admin using password
```

6. Update the CLI Driver settings on the primary node 1 and on the DMGR.

- a. Locate the following file:

Windows: db2home/sql/lib/db2cli.ini

- b. Edit the file by adding the following to the end of the file:

* For Fix Pack 11:

[COMMON]

DYNAMIC=1

Note: An empty line is required after the dynamic=1 at the end of the file.

* For Fix Pack 12:

[COMMON]

ReturnAliases=0

Note: An empty line is required after the ReturnAliases=0 at the end of the file.

Configure the database using the configuration wizard

Note: This example uses the configuration wizard tool. Before running the configuration wizard please populate the wpconfig.properties with the WasPassword before running the task.

Be aware that the configuration wizard writes all the database password values in the wpconfig_domain.properties file after completion with the string, "ReplaceWithYourPassword". This is important to note because future tasks may require that these password be either provided at the command line with the -D option or defined correctly in the wpconfig.properties files.

Use the following steps to configure the database using the configuration wizard:

1. In Portal 5.1x, a helper file existed for the database-transfer task. Portal V6 does NOT have a helper file for database-transfer. The reason for this is because of the new architecture of the wpconfig files. Portal V6 uses a wpconfig_dbdomain.properties file to contain most of the database specific properties and replaces the need for a separate helper file.
2. Make a copy of the original wpconfig_dbdomain.properties file and the wpconfig_dbtype.properties file.
3. Edit the files with your database values to use with the configuration wizard. The configuration wizard is designed in Portal V6 to read directly from these files as shown in Table 3-1 for wpconfig_dbtype.properties and in Table 3-2 on page 97 for wpconfig_dbdomain.properties

Table 3-1 DB settings in wpconfig_dbtype.properties

Field	Example value	Comments
DBSafeMode	false	<p>This only applies to database-specific tasks. If this property is set to true, database-specific tasks such as create/modify/drop database, are not performed. The property should be used if a pre-configured database (from a previous installation) already exists. If the property is set to false, the database is updated and the pre-existing database configuration is overwritten.</p> <p>Recommended value: false</p> <p>Default value: false</p>
db2.DBDriver	COM.ibm.db2.jdbc.app.DB2Driver	The name of the class that SqlProcessor uses to import SQL files.
db2.DbLibrary	D:/IBM/SQLLIB/java/db2java.zip	Depends on the installation directory of the DB2 Administration Client.

In Table 3-2 on page 97, substitute the *appropriate name* for the dbdomain value.

Example: release.DbType

Following are the available dbdomains now:

- release
- customization
- community
- JCR
- WMM
- feedback
- Likeminds

- Designer (TechPreview and cannot be moved)
- Sync (TechPreview and cannot be moved)

Table 3-2 DB settings in *wpconfig_dbdomain.properties*

Field	Example value	Comment
<i>dbdomain.DbType</i>	db2	The type of database used to store information for WebSphere Portal.
<i>dbdomain.DbName</i>	wpsdb	The name of the WebSphere Portal domain database and schema.
<i>dbdomain.DbUrl</i>	jdbc:db2:wpsdb	The database URL used to access the WebSphere Portal database with JDBC™. The value must conform to standard JDBC URL syntax.
<i>dbdomain.DbUser</i>	db2admin	The user ID for the database administrator.
<i>dbdomain.DbPassword</i>	password	The password for the database administrator.

4. Make sure node agent is started and WebSphere Portal was stopped with the following command:

```
/IBM/WebSphere/AppServer/bin/serverstatus -all -username <wasadminuser> -password <wasadminpwd>
```
5. Start the configuration wizard from <wp_root>/config/wizard/configwizard.bat.
6. In the “Welcome to the Wizard” window, click Next.
7. In the “Select the task that you want to perform” window, select **Transfer data to another database**, and click Next as shown in Figure 3-55 on page 98.

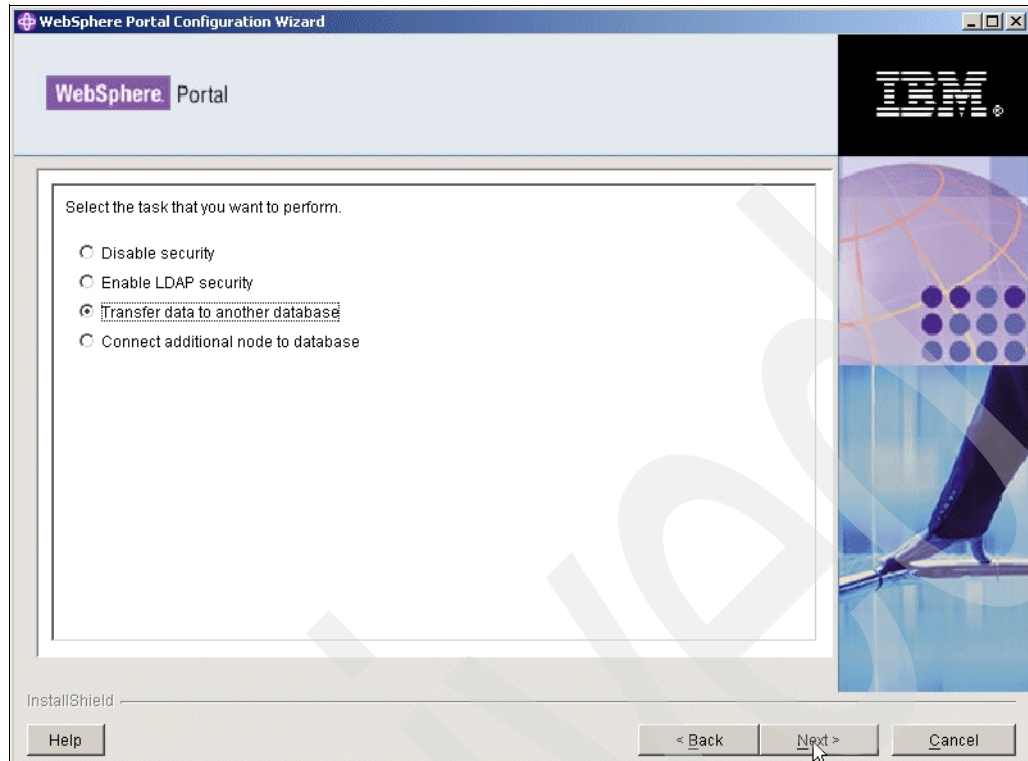


Figure 3-55 Transfer data to another database window

8. In the “WebSphere Application Server global security” window, provide the WSAS Admin User and password. Click Next as shown in Figure 3-56.

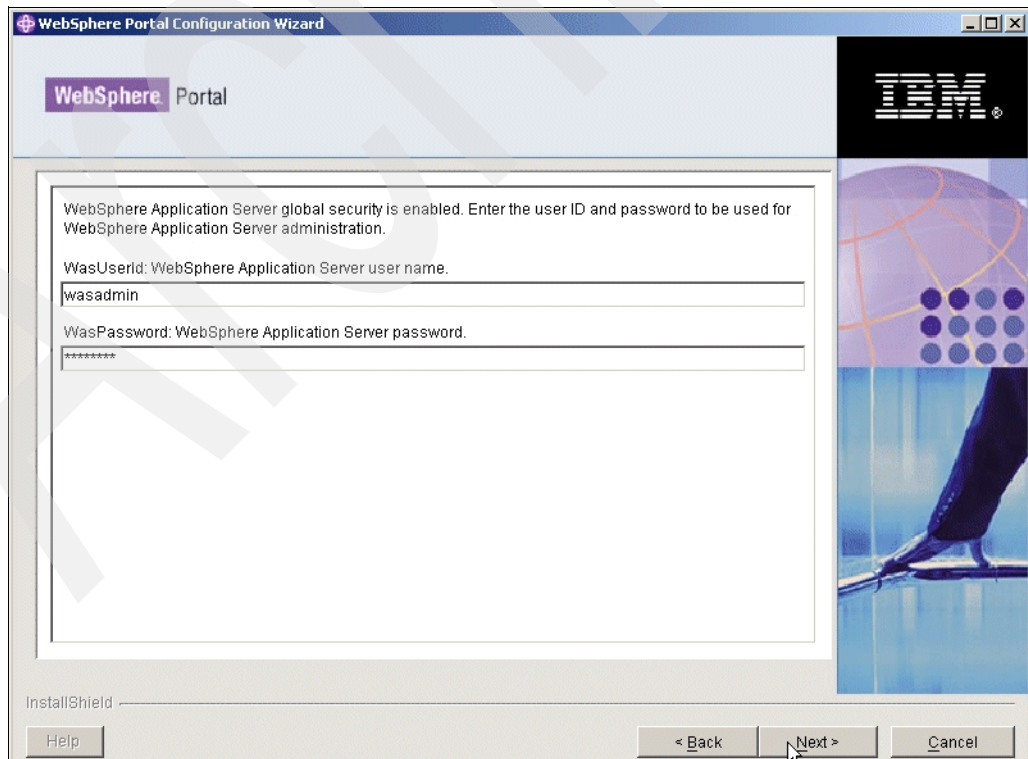


Figure 3-56 Username and password window

9. In the “Source Database Type” window, select the Source database type, which is **IBM Cloudscape**. Click Next as shown in Figure 3-57.

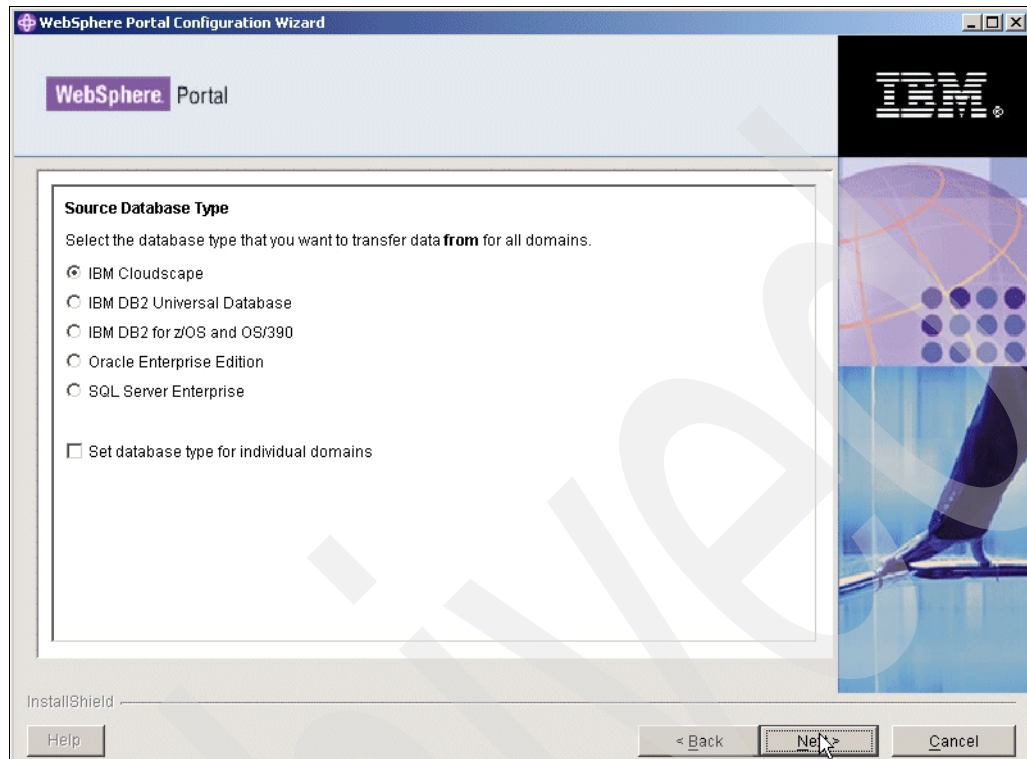


Figure 3-57 Source database type window

10. In the “Target Database Type” window, select the appropriate target database type, and click Next as shown in Figure 3-58 on page 100.

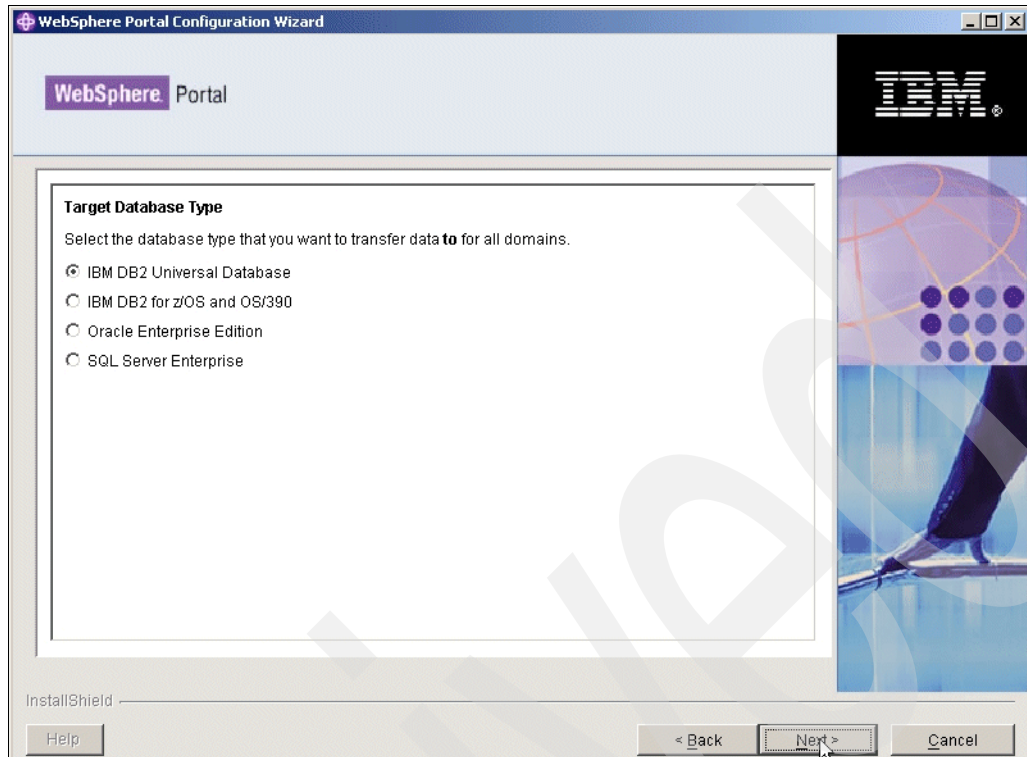


Figure 3-58 Target database type window

11. Verify the values pulled by the configuration wizard from the wpconfig files. Click Next as shown in Figure 3-59.

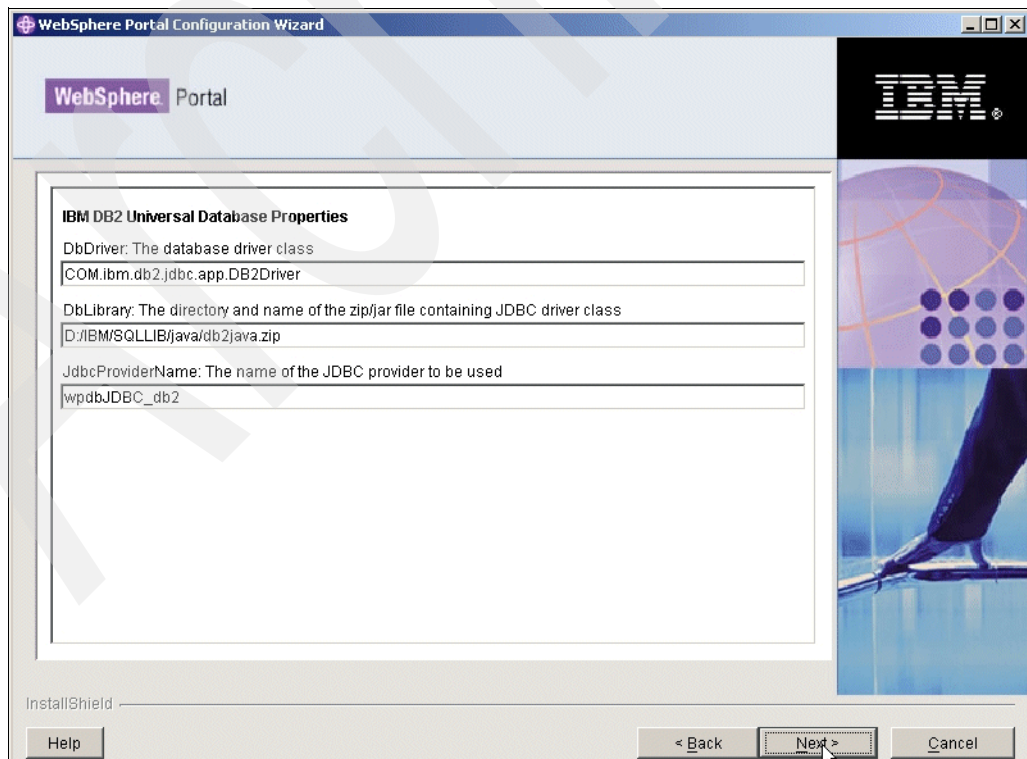
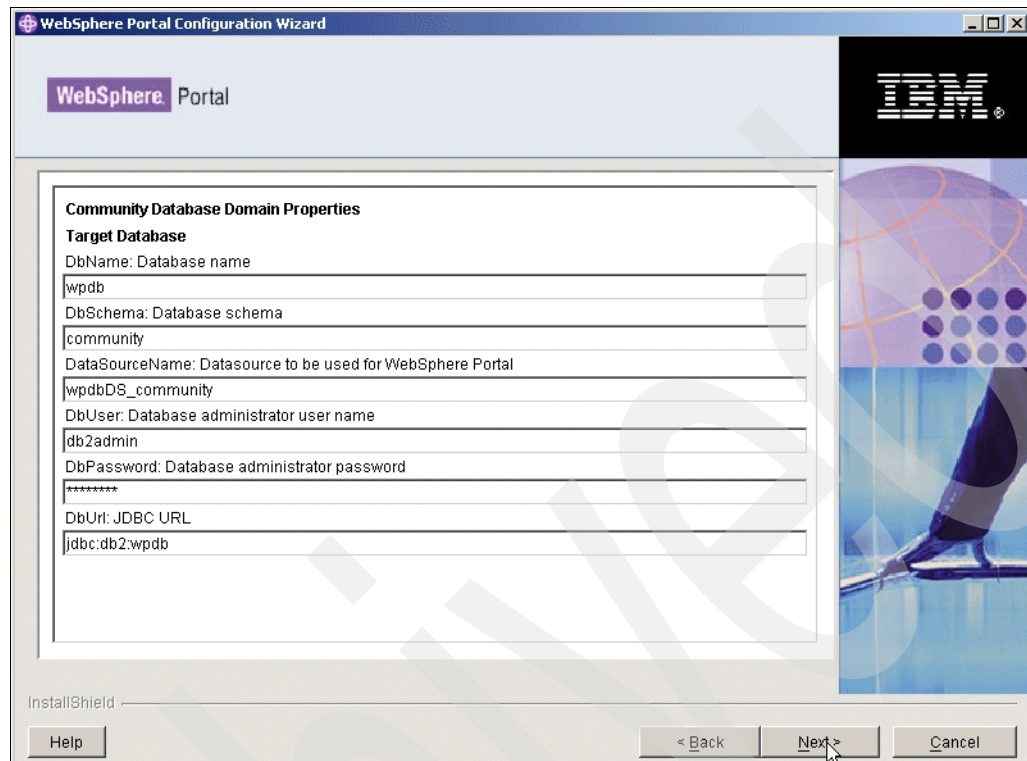


Figure 3-59 IBM DB2 Universal database properties window

12. Verify the values pulled by the configuration wizard from the wpconfig files, and manually type the password. Click Next as shown in Figure 3-60.



The image shows a screenshot of the 'WebSphere Portal Configuration Wizard' window. The title bar reads 'WebSphere Portal Configuration Wizard'. The window has a header with the 'WebSphere Portal' logo on the left and the IBM logo on the right. The main content area is titled 'Community Database Domain Properties' and contains a 'Target Database' section with the following fields:

- DbName: Database name (wpdb)
- DbSchema: Database schema (community)
- DataSourceName: Datasource to be used for WebSphere Portal (wpdbDS_community)
- DbUser: Database administrator user name (db2admin)
- DbPassword: Database administrator password (masked with asterisks)
- DbUrl: JDBC URL (jdbc:db2:wpdb)

At the bottom of the window, there is an 'InstallShield' progress bar and three buttons: 'Help', '< Back', and 'Next >', with a 'Cancel' button to the right. A mouse cursor is pointing at the 'Next >' button.

Figure 3-60 Community database domain properties window

13. Verify the values pulled by the configuration wizard from the wpconfig files, and manually type the password. Click Next as shown in Figure 3-61 on page 102.

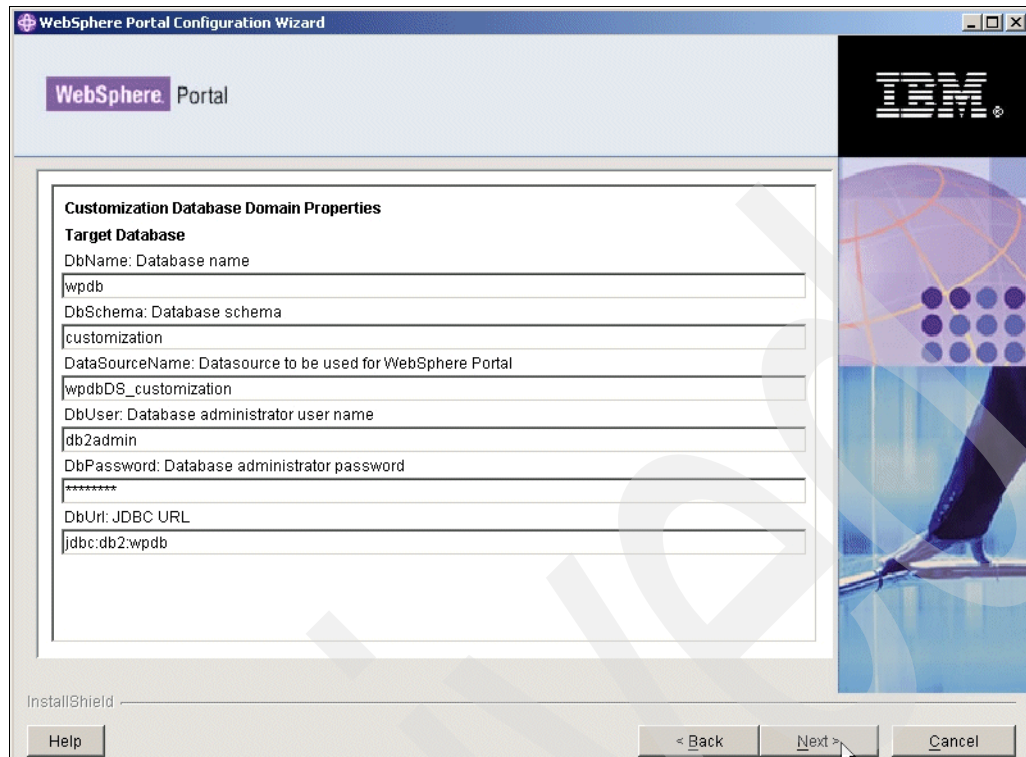


Figure 3-61 Customization database domain properties window

14. Verify the values pulled by the configuration wizard from the wpconfig files, and manually type the password. Click Next as shown in Figure 3-62.

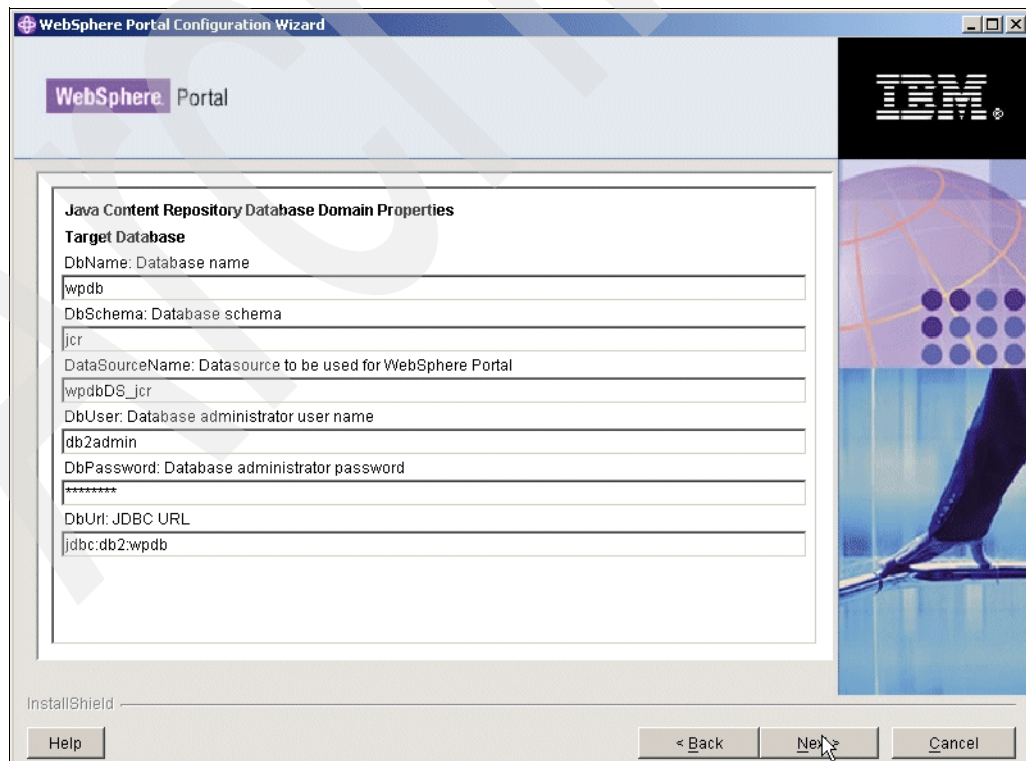
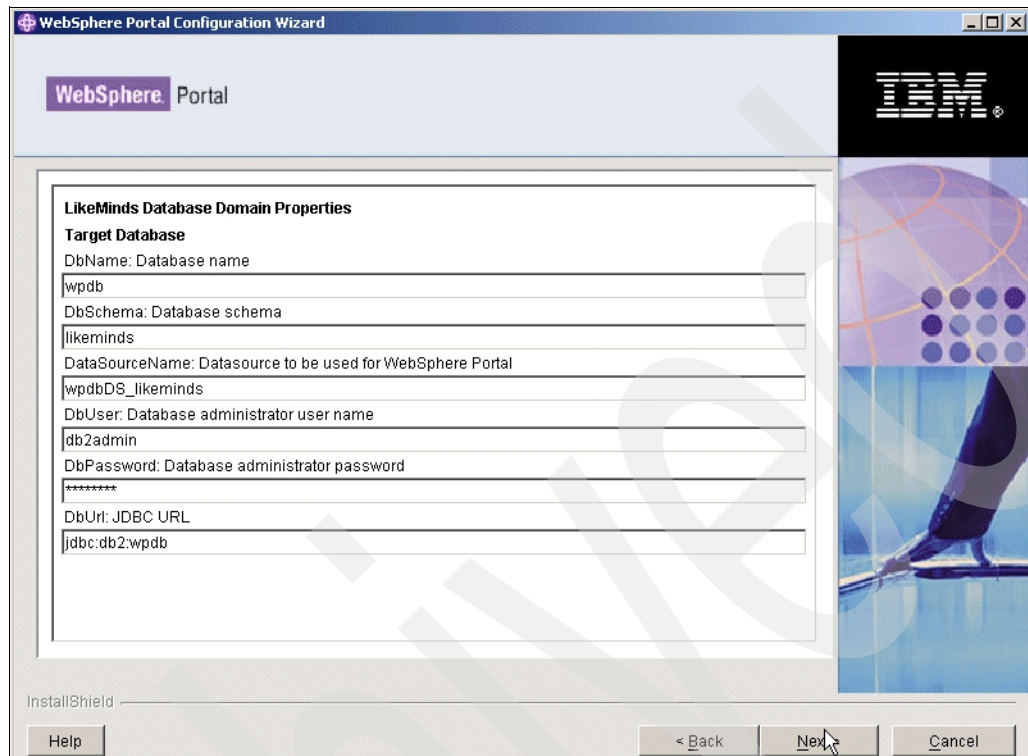


Figure 3-62 Java Content Repository database domain properties window

15. Verify the values pulled by the configuration wizard from the wpconfig files, and manually type the password. Click Next as shown in Figure .



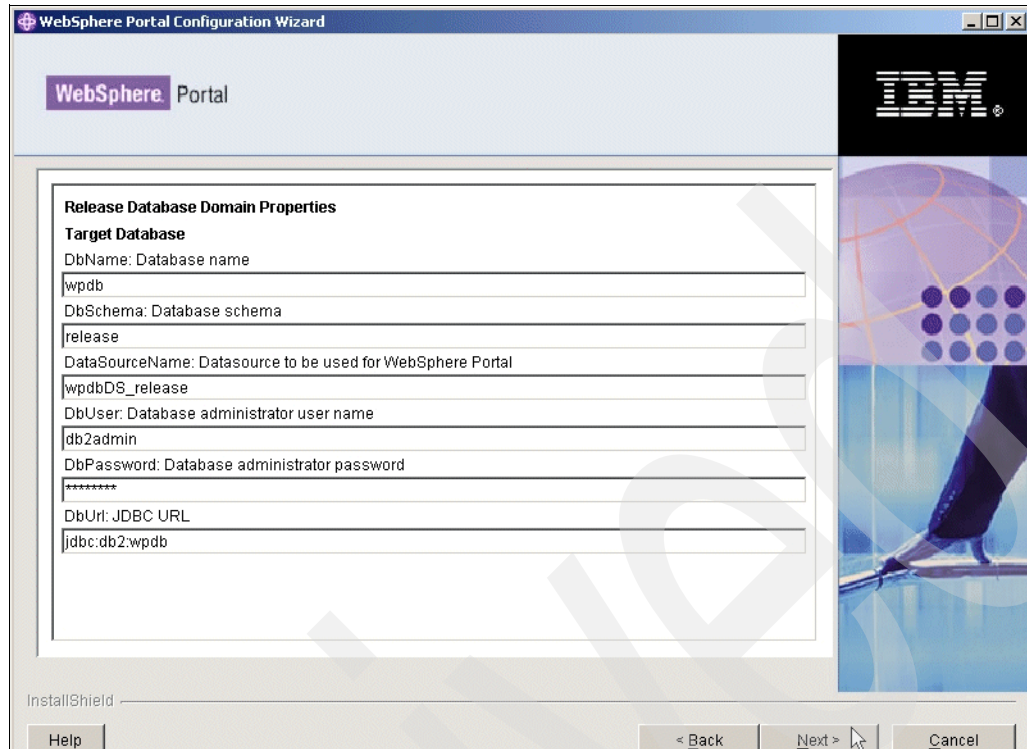
The image shows a screenshot of the 'WebSphere Portal Configuration Wizard' window. The title bar reads 'WebSphere Portal Configuration Wizard'. The main content area is titled 'LikeMinds Database Domain Properties' and contains a 'Target Database' section with the following fields and values:

Property	Value
DbName: Database name	wpdb
DbSchema: Database schema	likeminds
DataSourceName: Datasource to be used for WebSphere Portal	wpdbDS_likeminds
DbUser: Database administrator user name	db2admin
DbPassword: Database administrator password	*****
DbUrl: JDBC URL	jdbc:db2:wpdb

At the bottom of the window, there is an 'InstallShield' progress bar and three buttons: 'Help', '< Back', and 'Next >', with a mouse cursor hovering over the 'Next >' button. A 'Cancel' button is also visible on the far right. The IBM logo is in the top right corner, and a decorative graphic is on the right side of the window.

Figure 3-63 LikeMinds database domain properties window

16. Verify the values pulled by the configuration wizard from the properties file, and manually type the password. Click Next as shown in Figure 3-64 on page 104.



WebSphere Portal Configuration Wizard

WebSphere Portal

Release Database Domain Properties

Target Database

DbName: Database name
wpdb

DbSchema: Database schema
release

DataSourceName: Datasource to be used for WebSphere Portal
wpdbDS_release

DbUser: Database administrator user name
db2admin

DbPassword: Database administrator password

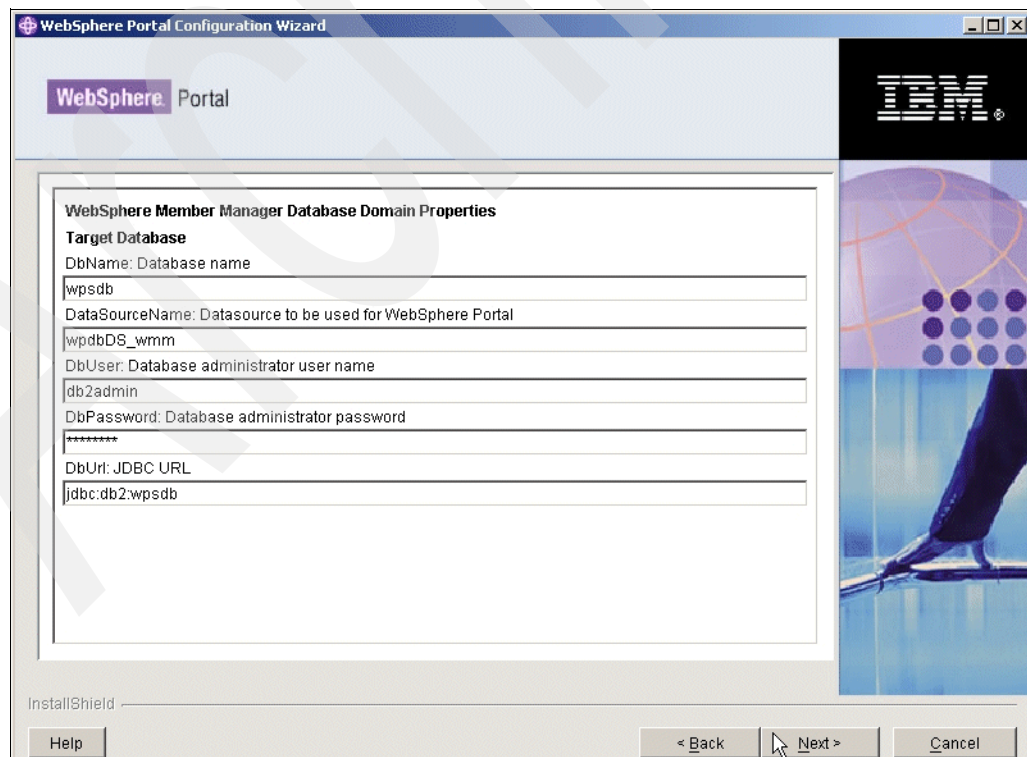
DbUrl: JDBC URL
jdbc:db2:wpdb

InstallShield

Help < Back Next > Cancel

Figure 3-64 Release database domain properties window

17. Verify the values pulled by the configuration wizard from the wpconfig files, and manually enter the password. Click Next as shown in Figure 3-65.



WebSphere Portal Configuration Wizard

WebSphere Portal

WebSphere Member Manager Database Domain Properties

Target Database

DbName: Database name
wpsdb

DataSourceName: Datasource to be used for WebSphere Portal
wpdbDS_wmm

DbUser: Database administrator user name
db2admin

DbPassword: Database administrator password

DbUrl: JDBC URL
jdbc:db2:wpsdb

InstallShield

Help < Back Next > Cancel

Figure 3-65 WebSphere Member Manager database domain properties

18. In the “Review the Summary” window, verify the settings, and click Next to start the database-transfer task.
19. In the “Wizard Completed Successfully” window, ensure that the script completed successfully, and click Finish to close the Configuration Wizard as shown in Figure 3-66.

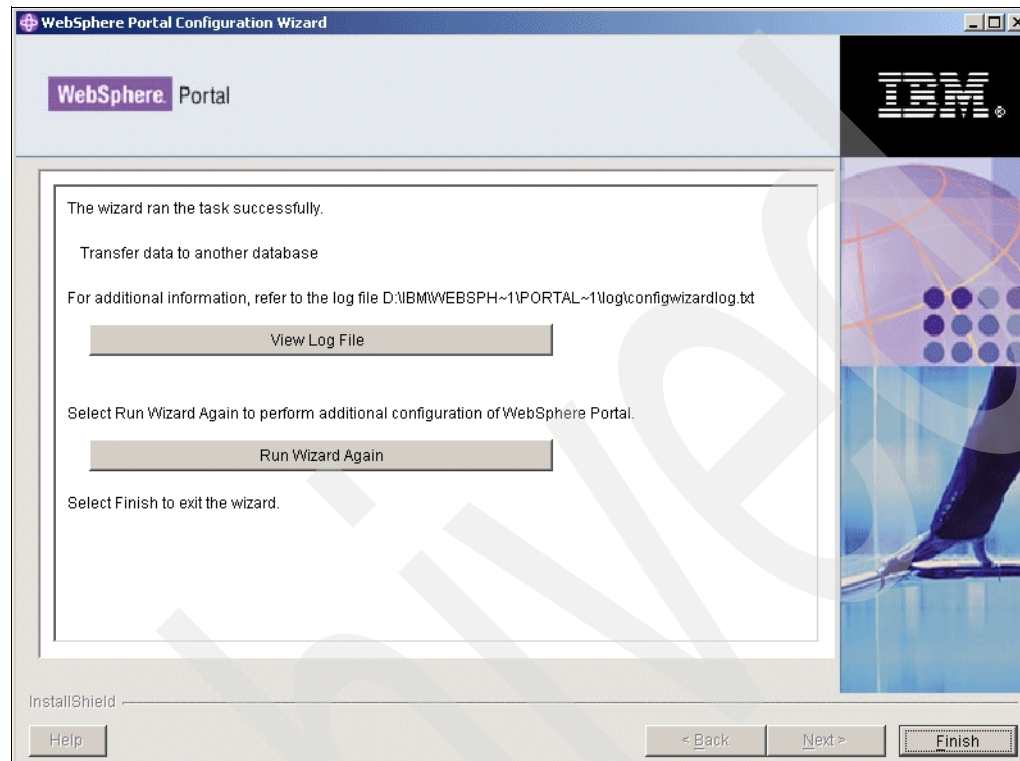


Figure 3-66 Transfer success window

20. After the task completes successfully, check that Portal is started. If it is not started, then start it, and verify the Portal by rendering the Portal from a browser. The default Portal URL is `http://<fullqualifiedhostname>:9080/wps/portal`.

Important: Due to an architecture change in the way that Portal uses the WSAS datasources, you can no longer test the Portal datasources successfully through the WSAS AdminConsole after the Portal install. Attempting to test the connection fails, but this is not an indication of a problem and Portal will still function.

Testing your JDBC data connection on the DMGR

1. Adding a new cell level (or scope) WebSphere variable through the Deployment Manager Administrative Console will allow the Test Connection feature to work. Follow the instructions below to create the additional WebSphere Variable:
 - a. Click Resources → JDBC Providers.
 - b. Clear out the entries for node and server, and click Apply.
 - c. Click the DB2 JDBC Provider. In this example it is called, wpdbJDBC_db2.
 - d. Make note of the WebSphere variable defined in the Class Path text box as shown in Figure 3-67 on page 106.

JDBC providers

☐ Messages

Modifying the implementation class name will eliminate the ability to create data sources and data sources version 4 from templates.

JDBC providers > wpdbJDBC_db2

JDBC providers are used by the installed applications to access data from databases.

Configuration

General Properties

* Scope
cells:dmc2Cell01

* Name
wpdbJDBC_db2

Description
DO NOT CHANGE DESCRIPTION
WITHIN THE TRIPLE
AMPERSANDS &&db2&&
(description is used for Portal
internal db type encoding)

Class path
\${DB2_JDBC_DRIVER_CLASSPATH}

Native library path

* Implementation class name
COM.ibm.db2.jdbc.DB2XADatasource

Apply OK Reset Cancel

Additional Properties

- [Data sources](#)
- [Data sources \(Version 4\)](#)

Figure 3-67 WebSphere variable defined in the Class Path text box

- Click Environment → WebSphere Variables.
- Set the scope to the node level of node 1 by clicking Browse Nodes, choosing node 1, and clicking OK.
- Click Apply.
- Scroll down until you find the variable DB2_JDBC_DRIVER_CLASSPATH. Notice in the 3rd column how the scope is defined down to the node level as shown in Figure 3-68.

<input type="checkbox"/>	DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH		cells:dmc2Cell01:nodes:horizc2n1Node01
<input type="checkbox"/>	DB2UNIVERSAL_JDBC_DRIVER_PATH		cells:dmc2Cell01:nodes:horizc2n1Node01
<input checked="" type="checkbox"/>	DB2_JDBC_DRIVER_CLASSPATH	D:/IBM/SQLLIB/java/db2java.zip	cells:dmc2Cell01:nodes:horizc2n1Node01

Figure 3-68 DB2_JDBC_Driver_Classpath variable view

- This variable was created at the node level (or scope), and we need to create a variable identical to this one at the cell level (or scope).
- Clear out the entry for node in the Scope section at the top, and click Apply.

- k. Click the New button, and create a variable with the same Name and Value as the DB2_JDBC_DRIVER_CLASSPATH that was defined at the node level. This may require moving the actual jdbc driver to the DMGR machine.
- l. Click Apply. Save the changes and synchronize with the nodes. Notice how the scope is defined only to the cell level.
- m. Restart the DMGR. Now attempts to test the datasource connections now should succeed.

3.4.6 Create the cluster definition

The next step after transferring the database to Database System is to create the cluster definition.

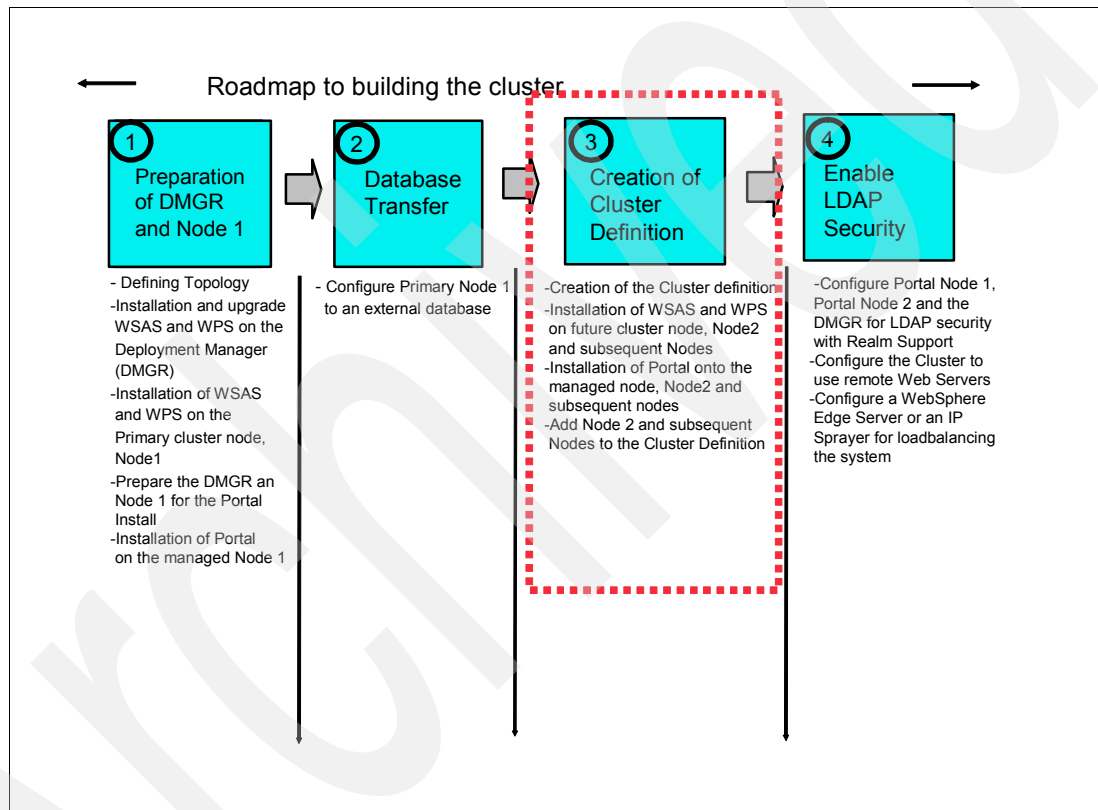


Figure 3-69 Flow chart installation steps: step 3

1. Edit the wpconfig.properties to match the Settings for the new WebSphere Portal cluster definition as shown in Table 3-3 on page 108.

Table 3-3 Cluster properties in wpconfig.properties

Field	Example value	Comments
WasPassword	wpsadmin	You must add the PortalAdminPwd and the WasPassword values to the wpconfig.properties file and all the database password values to the wpconfig_dbdomain.properties file, or supply these values on the command line. This is because of what was described before in that the configuration wizard replaces all the password values with the string, "ReplaceWithYourPassword" for security reasons. If you do not want to provide the PW in the file, you have to add the "-DWasPassword=password" option when running the cluster-setup script.
PortalAdminPwd	wpsadmin	
PrimaryNode	true	Verify that value for this node is true.
Clustername	PortalCluster	Specify the cluster name you want to use when creating the cluster. Do not use spaces or special characters in the cluster name.

Important: If you want to change the name of the cluster to something other than the default in the wpconfig.properties file, you **MUST** change it now **BEFORE** the cluster definition is created. This can be changed by editing the wpconfig.properties file and changing the ClusterName property.

- Edit the wpconfig_dbdomain.properties to match the settings for the new WebSphere Portal cluster definition:
 - Set all PW for the DB admin user.
 - Check the others entries.
- Check that DMGR and node are started with the following command:


```
D:\IBM\WebSphere\AppServer\profiles\<profile_name>\bin\serverstatus -all -username <wasadmin> -password <waspwd>
```
- Start WebSphere_Portal when not already started, with the following command:


```
D:\IBM\WebSphere\AppServer\profiles\<profile_name>\bin\startserver WebSphere_Portal
```

5. Create the cluster definition by running the following command:

```
<wp_root>/config/WPSconfig.bat cluster-setup >cluster-setup.log
```

Check the log file if the build was successful.

6. Restart DMGR, nodeagent and WebSphere_Portal to load the new configuration.
7. To check if the cluster definition and creation was successful, log into the Admin console of the DMGR and browse to Server → Clusters.

An entry with the *ClusterName* should be available as shown in Figure 3-70.

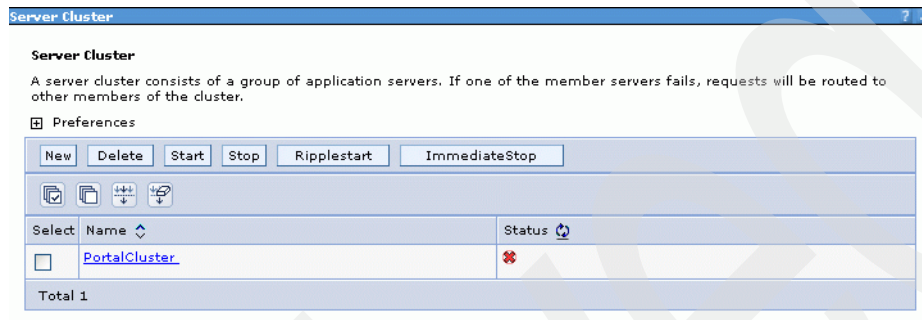


Figure 3-70 Check cluster definition

8. As a checkpoint in the process, you now have a 1 node cluster configured to an external database and using the WMM database for security.

3.4.7 Install WSAS and WPS on future cluster node, node 2, and subsequent nodes

Important: If you already have a WebSphere Portal Cluster up and running and you want to add an additional node to the cluster, proceed with the steps in Section 3.6, “Adding a Horizontal node to a cluster with WebSphere Process Server” on page 168.

Important: This section explicitly defines the required approach to build a Portal cluster that was installed on WebSphere Process Server. To do this you must install Portal into an already federated WSAS and WPS profile. Because of this requirement, we MUST install WSAS and WPS from their native installers, and federate the node BEFORE using the Portal installer to install Portal.

WebSphere Application Server installation on node 2 and subsequent nodes

1. Install WSAS on node 2 by running the installer from the following:

```
<cd_root>/W-1/windows/ia32/ifpackage/WAS/install.exe
```

Note: Make sure the installer window is titled “Welcome to IBM WebSphere Application Server Network Deployment, V6”. This title means that you can use this installer to install either, DMGR or WSAS profiles. If the title is “WebSphere Application Server Version 6.0”, you are using an installer that only has the ability to install WSAS profiles and not DMGR profiles.

2. In the “Accept the license agreement” window, accept the license, and click Next.

3. The Installation Wizard performs a pre-requisite check for you when this runs successfully. Click Next; otherwise, check that your operating system meets the requirements for WSAS as discussed in Section 2.1, “System requirements” on page 22.
4. If installing on Windows, when asked for the install location, shorten the default path. There is a path name limitation in Windows. Windows cannot handle path names longer than 256 characters. For example:

D:\IBM\WebSphere\ApplicationServer

Click Next.

The Installation Wizard displays the installation summary including interim fixes to be applied.

5. Review the options in the installation summary window, and click Next.

Note: The WSAS installer from the Portal CDs automatically upgrades WSAS to 6.0.2.9.

The WSAS installer from the Portal CDs also automatically applies the following WSAS and WPS iFixes required for an install of Portal 6.0.0.0:

- PK21578
- PK22180
- PK25804
- PK25597
- PK23535
- PK24146
- PK27024
- PK21870
- PK21793
- PK26123
- PK25630
- PK24463
- PK21704
- PK24014
- PK21927
- PK18492
- PK20404
- PK23985
- PK23358

Note: You can find WSAS fixes by accessing the following Web site and searching for the fix:

<http://www-306.ibm.com/software/webservers/appserv/was/support/>

6. In the “Installation complete” window, ensure that the **Launch the Profile creation wizard** check box remains UNCHECKED. We will create a WPS profile at the end of the WPS install. Click Next as shown in Figure 3-28 on page 73.

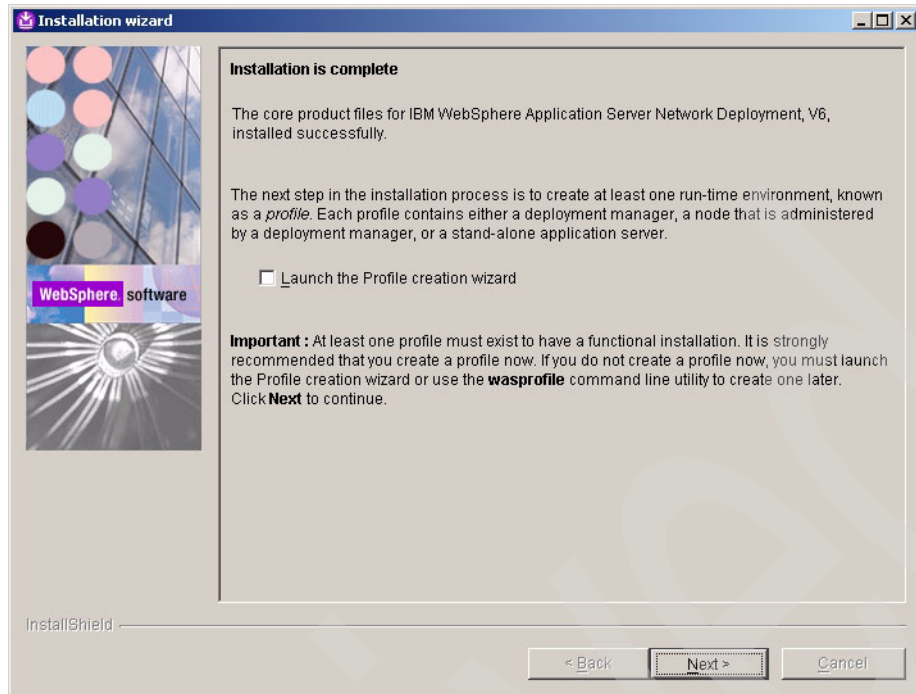


Figure 3-71 Installation is complete window

7. Click Finish to close the Installation wizard.

WebSphere Process Server installation on node 2

1. Install WPS 6.0.1.1 by running the installer from the following:
`<cd_root>/W-2/windows/ia32/WBI/install.bat`

Note: Ensure that you use the install.bat file and NOT the install.exe to install WPS.

2. The Installation Wizard performs a pre-requisite check for you when this runs successfully. Click Next; otherwise, check that your operating system meets the requirements for WSAS as discussed in Section 2.1, "System requirements" on page 22.

3. In the “Detected WebSphere Application Server” window, ensure that you use the existing WSAS you just installed, and click Next as shown in Figure 3-72.

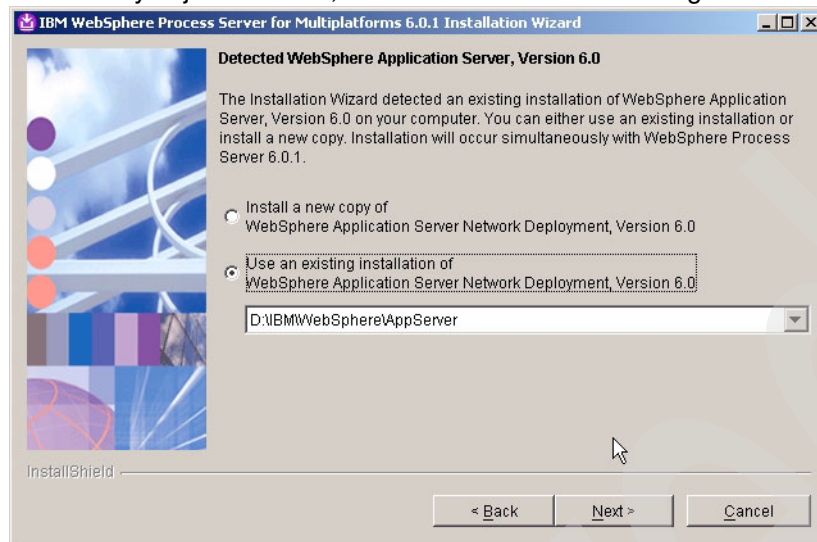


Figure 3-72 WSAS detection window

4. Review the options in the “Installation summary” window, and click Next.
5. In the “Installation complete” window, we want to create a profile now. Select **Launch the Profile Wizard**. Click Next to launch the WPSprofile Creation wizard as shown in Figure 3-73.

Note: If you have to launch the WPS Profile Creation wizard manually, make sure that you launch the WPS Profile Creation wizard and NOT the WSAS Profile Creation wizard. The WPS Profile Creation wizard script is located at the following:
<wsas_root>/bin/ProfileCreator_wbi/pcatWindows.exe

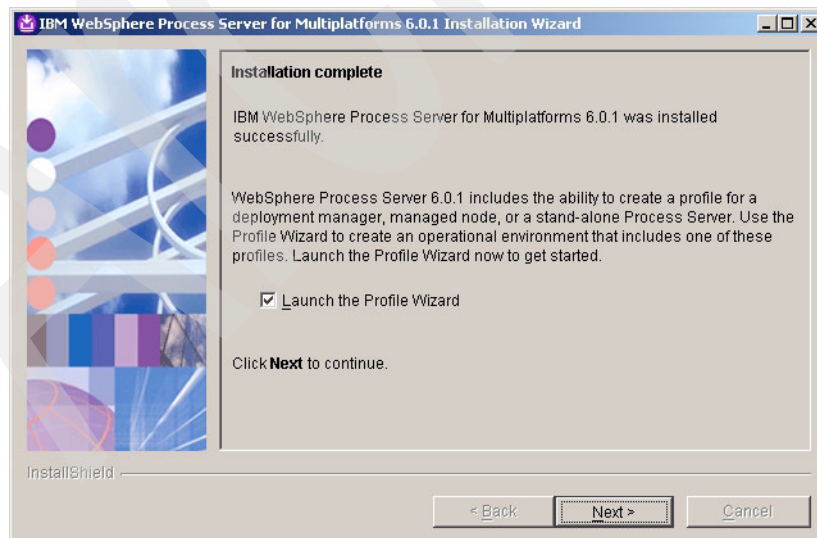


Figure 3-73 Installation is complete with profile wizard launch option window

6. In the “Welcome to the profile creation wizard” window, click Next.

7. After the Profile Creation wizard is launched, select the **Custom profile** option on the “Profile type selection” window, and click Next as shown in Figure 3-74.

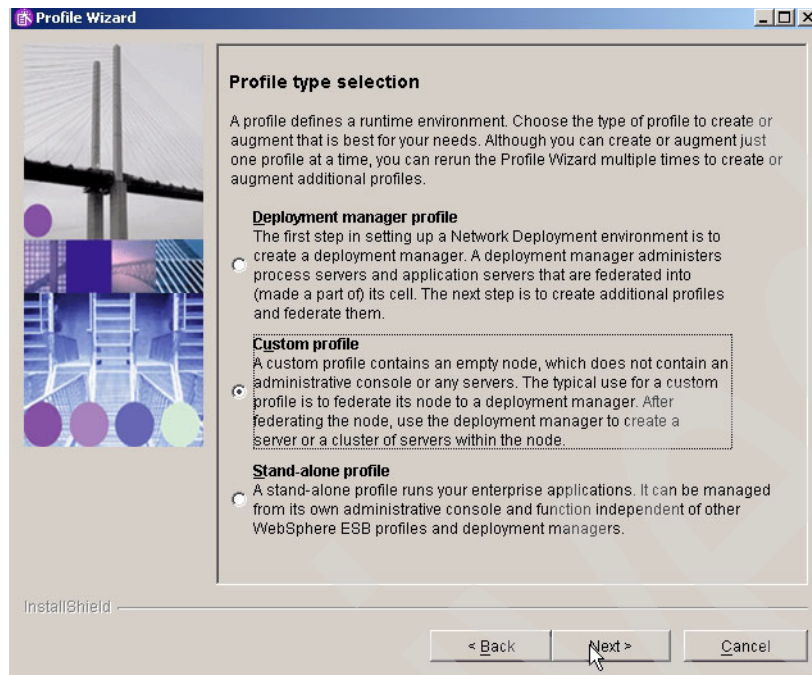


Figure 3-74 Profile type selection window

8. In the “Federation” window, enter the values, and click Next as shown in Figure 3-75 on page 114.
- Type the Deployment Manager Name where the node should be added later on into the cell.
 - Select the **Federate this node later using the addNode command** option because security is enabled in the cluster. You cannot use the automatic federation feature on secondary nodes.

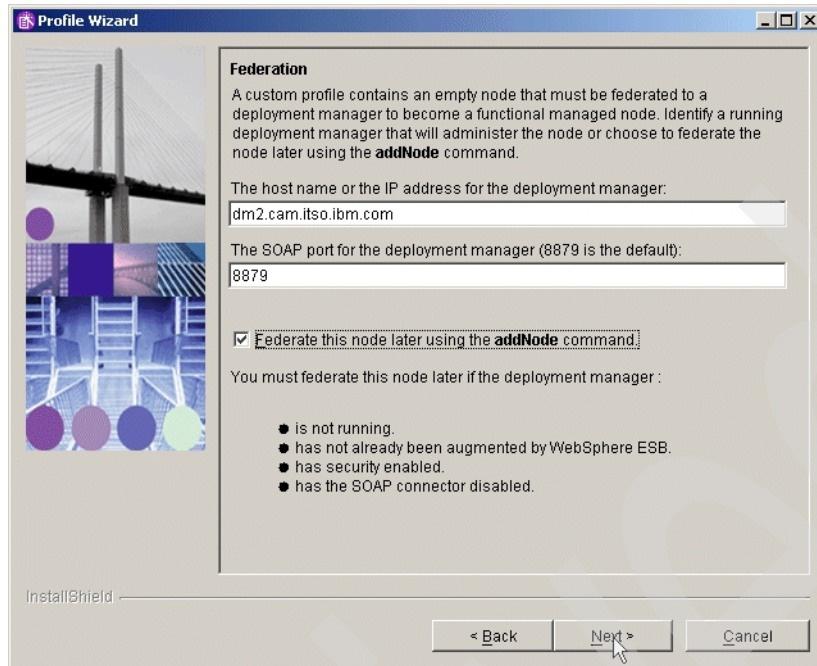


Figure 3-75 Federation window

9. In the “Profile name” window, enter a profile name for the WebSphere Process Server Instance. This name can be different for each node and for the DMGR. In our setup we choose the node name as the profile name. Click Next as shown in Figure 3-76.

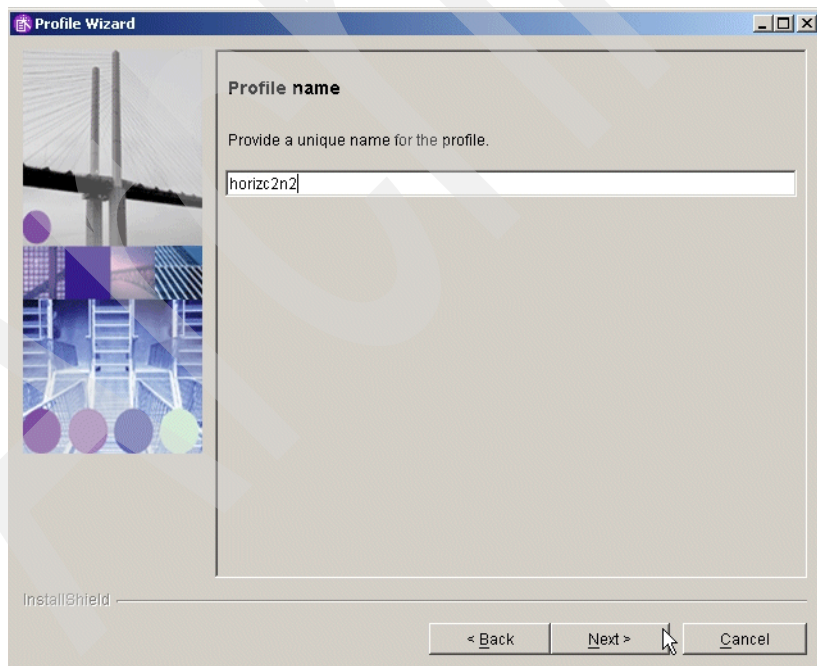


Figure 3-76 Profile name window

10. In the “Profile directory” window, verify the directory name. Click Next as shown in Figure 3-77 on page 115.

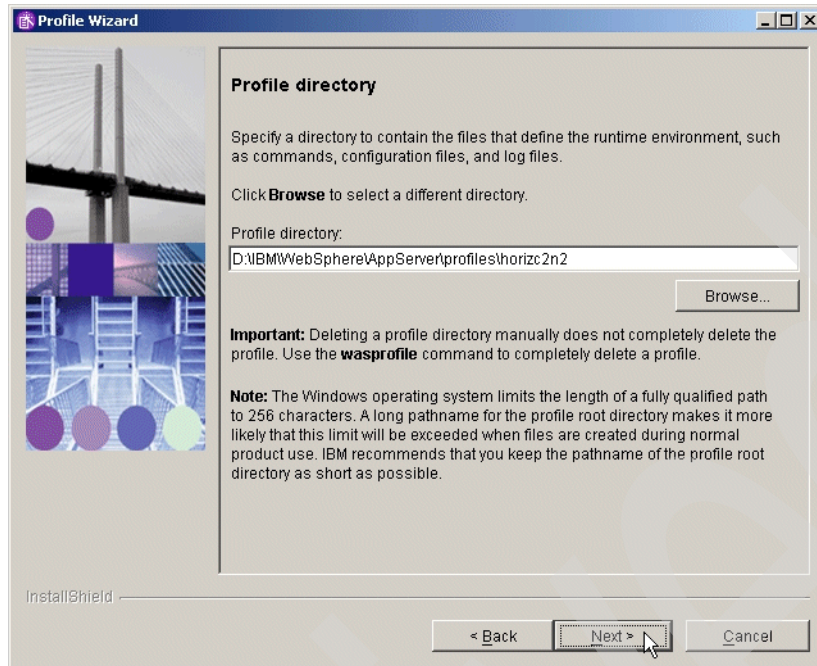


Figure 3-77 Profile directory window

11. In the “Node and Host Names” window, check the value of node and host name, and click Next as shown in Figure 3-78.

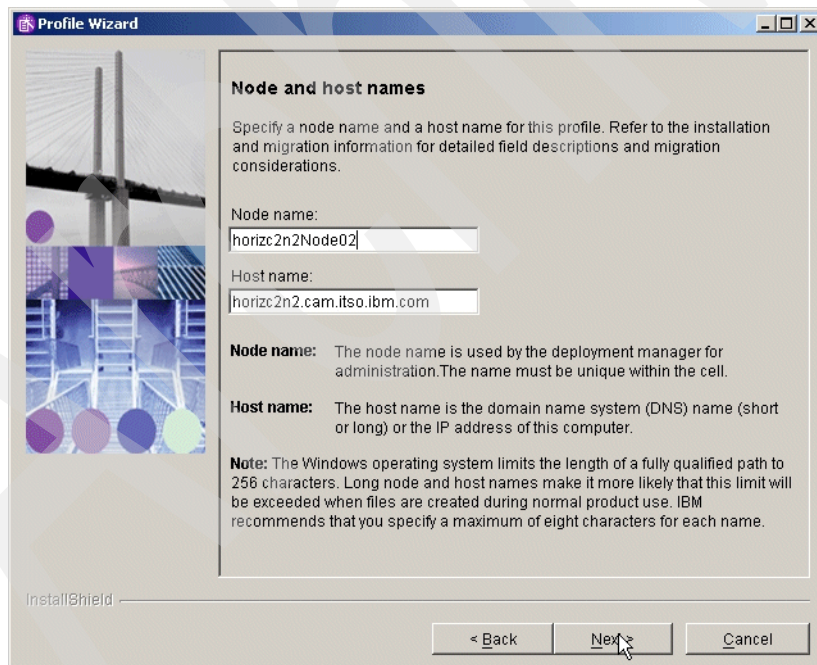


Figure 3-78 Node and host names selection window

12. In the “Profile Summary” window, check the values, and the click Next.
13. In the “Profile creation is complete” window, check if the creation was successful, and click Finish.

14. After the profile is created we will federate the node using the addNode command.

Before running the *addNode.bat* command ensure that the DMGR is started.

To add a node to the Deployment Manager cell, issue the addNode.bat command (on one line) on the command line of the node to be added:

```
<wsas_root>\profiles\<profile_name>\bin\addNode.bat  
<deployment_manager_host> <deployment_manager_port> -username  
<admin_user_id> -password <admin_password> >addnode.log
```

where:

- deployment_manager_host is the Deployment Manager host name.
- deployment_manager_port is the Deployment Manager SOAP connector-address. The default value is 8879.
- admin_user_id is the WebSphere Application Server administrative user name. This parameter is optional but is required if security is enabled.
- admin_password is the administrative user password. This parameter is optional but is required if security is enabled.

Example:

```
addNode.bat dmc2.cam.itso.ibm.com.abc.com 8879 -username <wsas_id> -password  
<wsas_pwd> >addnode.log
```

Note: To run the addNode command here you MUST supply username and password because security was enabled on the DMGR.

Visit the following Web site to see the appropriate Network Deployment Information Center for details on the addNode command.

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/welcome_nd.html

Note: Ensure that the clocks are synchronized to within five minutes of each other on node 1 machine and on the DMGR machine. If the clocks are not within five minutes, the addNode process will fail.

15. After the AddNode is created and federated to the DMGR, apply the remaining WSAS and WPS iFixes required for Portal 6.0.0.0.

16. Obtain the current WSAS update installer shipped with the CDs at the following location:

```
<cd_root>/W-Setup/was6_fixes/win/updateinstaller
```

17. Copy the updateinstaller directory to [was_root]\AppServer\ of the file system.

18. Copy the necessary fixes to the [was_root]\AppServer\updateinstaller directory.

These fixes can be found on the Portal CDs at

```
<cd_root>/W-Setup/wps6_fixes/win/updateinstaller/maintenance. Copy the maintenance  
dir to [was_root]\AppServer\UpdateInstaller.
```

- IY82199
- IY83206
- JR23774
- JR24221
- JR24190

Java SDK 1.4.2 SR4 Cumulative Fix—this JDK should be installed with the install. You can verify the level of the WSAS JDK by running the following command:

```
<wsas_root>/java/bin/java -version
```

If the Java SDK has not the correct version you can find the Cumulative Fix at the following Web site:

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg24011104>

Copy the maintenance directory to WebSphere\AppServer\UpdateInstaller\sdk, and use the Update Installer that is already copied.

Hint: If the Installation of the JR24190 Fix fails, then restart the Windows Server, uninstall the Fix, and then install the Fix again.

19. Start the WSAS update installer by running the <wsas_root>/updateinstaller/update.exe to apply the remaining WSAS and WPS iFixes that are required for Portal, but were not installed as part of the WSAS install.
20. Enter the directory location of the AppServer, and click Next.
21. Select Install Maintenance Package, and click Next.
22. Browse for the first maintenance package to install, and click Next. The first one should be IY82199.
23. In the “Confirmation window” verify the maintenance Package. Click Next.
24. In the “Success window” verify that the maintenance package installed successfully, and click Relaunch to start the Update Installer again for the next maintenance package.

Note: Important: Fixes are installed one at a time by the Portal Update Installer. Repeat steps 20-25 for each maintenance package listed in step 19 on page 79.

3.4.8 Install Portal on the managed node, node 2, and subsequent nodes

1. Before installing Portal, move the WMM jars. Update the secondary node with required WMM jar files. These files are located on the Setup CD provided as part of the installation package for WebSphere Portal. Copy the following files from the <cd_root>/W-Setup/dmgr_wmmjars directory on the Setup CD to the <wsas_root>/lib directory on the secondary node:
 - wmm.jar
 - wmm.ejb.jar
 - wp.wire.jar
2. Ensure the time-out request for the Simple Object Access Protocol (SOAP) client for node 2 was increased to 6000. The default, in seconds, is 180. Within the [D:\IBM\WebSphere\AppServer\profiles\horizc2n2\properties] directory, edit the soap.client.props file. Change the line to the following:

```
com.ibm.SOAP.requestTimeout=6000
```
3. Begin the Portal install by using the following command:

```
<cd_root>/W-Setup/install.bat -W startPortalServerSequence.active=false
```
4. In the “Accept the license agreement” window, accept the license, and click Next.
5. In the “Installation Type” window, select Custom as the install type, and click Next.

6. In the “Select the Location of the existing instance” window, select the existing WebSphere AppServer install location. Select the **Install on a managed node** option, and click Next.
7. In the “Managed Node input” window, select Secondary Node, and select the desired profile that you want to install Portal onto. Click Next as shown in Figure 3-79. The result of installing as a secondary node is an empty Portal.

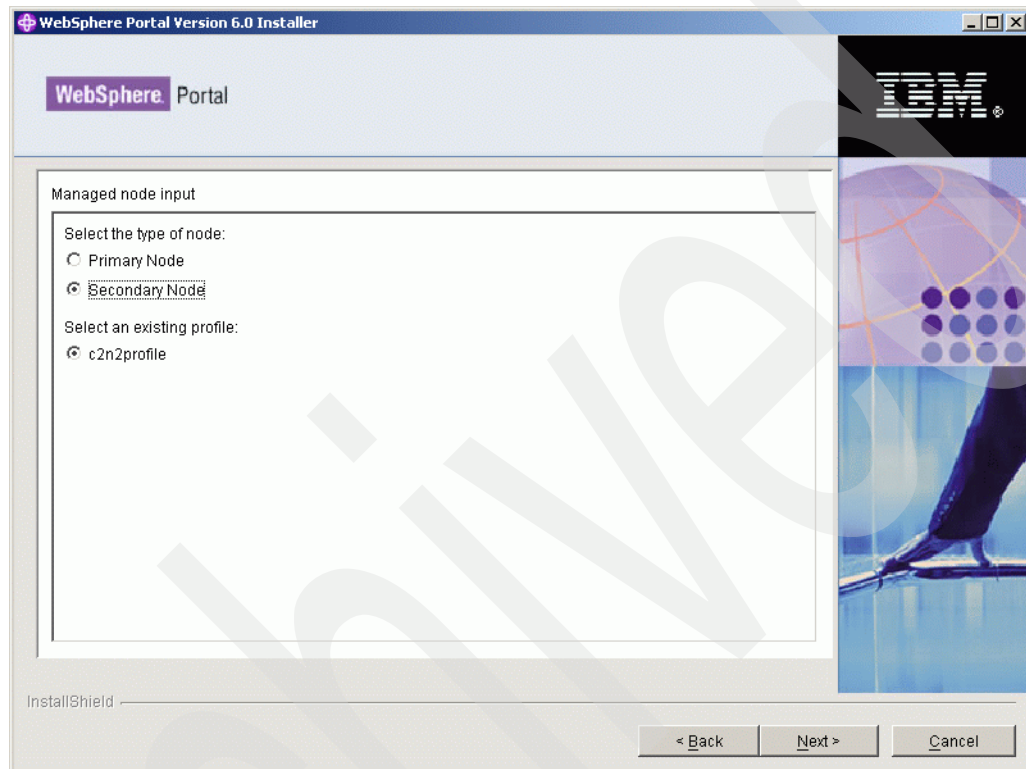


Figure 3-79 Secondary node window

8. In the “WebSphere application Server admin user id” window, define a WSAS Admin User and password. This is a new panel in the Portal installer because with Portal v6 the install enables security by default to the WMM database. You can use the default values like wasadmin and a password you want. Click Next.

You should use the same UserID and Password as for the primary node.

9. In the “WebSphere Portal install path” window, define the location for Portal to be installed, and click Next.
10. In the “WebSphere Portal Server Admin User” window, define the Portal Admin User and password. You can use the default values, such as wpsadmin and a password of your choice. Click Next.

Use the same UserID and password that you used for the primary node.

11. In the “Select the products to run as a service” window, decide whether you want WSAS and Portal to run as a service. In this guide we chose NOT to run either as a Windows service because of the longer startup time for windows with WP as a service.
12. In the “WebSphere Portal is ready to install” window, review the Summary, and click Next to begin the install.

13. When Installation successfully completes, notice the information, such as port number, that is provided. Click Finish.

Important: Do not attempt to verify that WebSphere Portal is operational after installation. Because you installed as a secondary node, no enterprise applications or portlets are installed onto the WebSphere Portal instance on the secondary node. This makes the Portal inoperable until it is added to the cluster.

3.4.9 Add node 2 and subsequent nodes to the cluster definition

Important: Next we run the cluster-setup task to add node 2 to the cluster. It is important to know that with Portal V6 the connect-database task is integrated into the cluster-setup task. If the PrimaryNode property is defined as False, the cluster-setup task performs the connect-database to point the secondary node to the existing cluster database. Therefore during this step we are required to ensure that the database properties are correct in the wpconfig.properties files.

Use the following steps to add node 2 and subsequent nodes to the cluster definition:

1. Install the DB2 Admin Client on node 2.
2. Open a DB2 command line processor on the Secondary node and subsequent nodes (Start → Programs → IBM DB2 → Command Line Tools → Command Line Processor).
 - a. Type the following command:

```
db2 => catalog tcpip node nodename remote fullyqualifiedDB2servername server port
```

Where *nodename* is the name of eight characters or fewer that you assign to the server, and *port* is the connection port used by DB2.
For example:

```
db2 => catalog tcpip node dbdom7 remote dbdom7.cam.itso.ibm.com server 50000
```

In this example, dbdom7 is designated as the node name (to be used in the rest of the commands) and 50000 is the connection port (the default for TCP/IP connections). The client always uses the same connection port as the DB2 server. You can verify the connection port for the DB2 server by opening a command prompt and issuing the **netstat -an** command. You should see a port in the 50000 range.
 - b. Type the following command (as appropriate) to catalog the databases that are hosted on the DB2 server.

```
db2 => catalog database databasename at node nodename
```

For example:

```
db2 => catalog database wpsdb at node dbdom7
```

Note: If you incorrectly catalog a database, you can uncatalog it by typing the following command:

```
db2=>uncatalog database database_name
```

- c. After you verify that all of the databases are cataloged correctly, test your connectivity to the database by typing the following command:

```
db2 => connect to databasename user DB2administratorname using password
```

For example:

```
db2 => connect to wpsdb user db2admin using password
```

3. Update the CLI driver settings on the node 2.

- a. Locate the following file:

Windows: db2home/sql1lib/db2cli.ini

- b. Edit the file by adding the following to the end of the file:

* For Fix Pack 11:

[COMMON]

DYNAMIC=1

Note: An empty line is required after the dynamic=1 at the end of the file.

* For Fix Pack 12:

[COMMON]

ReturnAliases=0

Note: An empty line is required after the ReturnAliases=0 at the end of the file.

4. Edit the wpconfig.properties in directory /IBM/WebSphere/PortalServer/config to match the settings for the WebSphere Portal cluster definition as shown in Table 3-4.

Table 3-4 Cluster properties in wpconfig.properties

Field	Example value	Comments
WasPassword	wpsadmin	You must add the PortalAdminPwd and the WasPassword values to the wpconfig.properties file and all the database password values to the wpconfig_dbdomain.properties file or supply these values on the command line. This is because the configuration wizard replaces all the password values with the string, "ReplaceWithYourPassword" for security reasons.
PortalAdminPwd	wpsadmin	-----
PrimaryNode	false	Verify that the value for this node is false because it is not the primary node.
Clustername	PortalCluster	ClusterName should be the name of the cluster created when running the cluster-setup task on the primary node, node 1.

Field	Example value	Comments
ServerName	WebSphere_Portal_2	ServerName is REQUIRED to be changed from WebSphere_Portal. The cluster-setup task is written to automatically remove the WebSphere_Portal server during the action-remove-appserver-wps task.

5. Make a backup of the original wpconfig_dbdomain.properties and wpconfig_dbtype.properties files on node 2 and subsequent nodes in directory /IBM/WebSphere/PortalServer/config.
6. Copy the wpconfig_dbdomain.properties and wpconfig_dbtype.properties from node 1 to node 2 to ensure the same database configuration. Check if the Values meet your node 2, special in the wpconfig_dbtype.properties file value in field db2.DbLibrary.
7. Again, in Portal V6 the connect-database task is integrated into the cluster-setup task; therefore, we must run the validate database tasks. If the passwords are defined in the wpconfig_dbdomain.properties file, the -D options below are not required at the command line.

```
WPSconfig.bat validate-database-driver
WPSconfig.bat validate-database-connection-wps -DDbPassword=password
WPSconfig.bat validate-database-connection-jcr -DJcrDbPassword=password
WPSconfig.bat validate-database-connection-feedback
-DFeedbackDbPassword=password
WPSconfig.bat validate-database-connection-likeminds
-DLikemindsDbPassword=password
WPSconfig.bat validate-database-connection-wmm -DWmmDbPassword=password
```

8. Check that DMGR and node are started with the following command:

```
D:\IBM\WebSphere\AppServer\bin\serverstatus -all -username <wasadmin>
-password <waspwd>
```

9. Start WebSphere_Portal when not already started, with the following command:

```
D:\IBM\WebSphere\AppServer\bin\startserver WebSphere_Portal
```

10. Create the cluster definition by running the following command:

```
<wp_root>/config/WPSconfig.bat cluster-setup >cluster-setup.log
```

Check the log file if the build was successful.

11. Restart DMGR, nodeagent, and WebSphere_Portal to load the new configuration.
12. To check if the cluster definition and creation were successful, log into the Admin console of the DMGR and browse to the following:

Server → Clusters → PortalCluster → ClusterMember

The entry with the “WebSphere_Portal_2” should be available as shown in Figure 3-80 on page 122.

Server Cluster						
Server Cluster > PortalCluster > Cluster members						
A group of servers that are managed together and participate in workload management for a group of applications. Requests to cluster members are rerouted when failures occur.						
Preferences						
<input type="button" value="New"/> <input type="button" value="Delete"/> <input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="ImmediateStop"/> <input type="button" value="Terminate"/>						
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>						
Select	Member name	Node	Version	Configured weight	Runtime weight	Status
<input type="checkbox"/>	WebSphere_Portal	horizc2n1Node01	6.0.2.9	2	2	
<input type="checkbox"/>	WebSphere_Portal_2	horizc2n2Node02	6.0.2.9	2	2	

Figure 3-80 Check cluster definition

13. Verify the Portal install by accessing it through a browser. By default Portal is installed onto port 9081 for each secondary node:

`http://<fullqualifiedhostname>:9081/wps/portal`

3.4.10 Configure Portal nodes and the DMGR for LDAP security with Realm Support

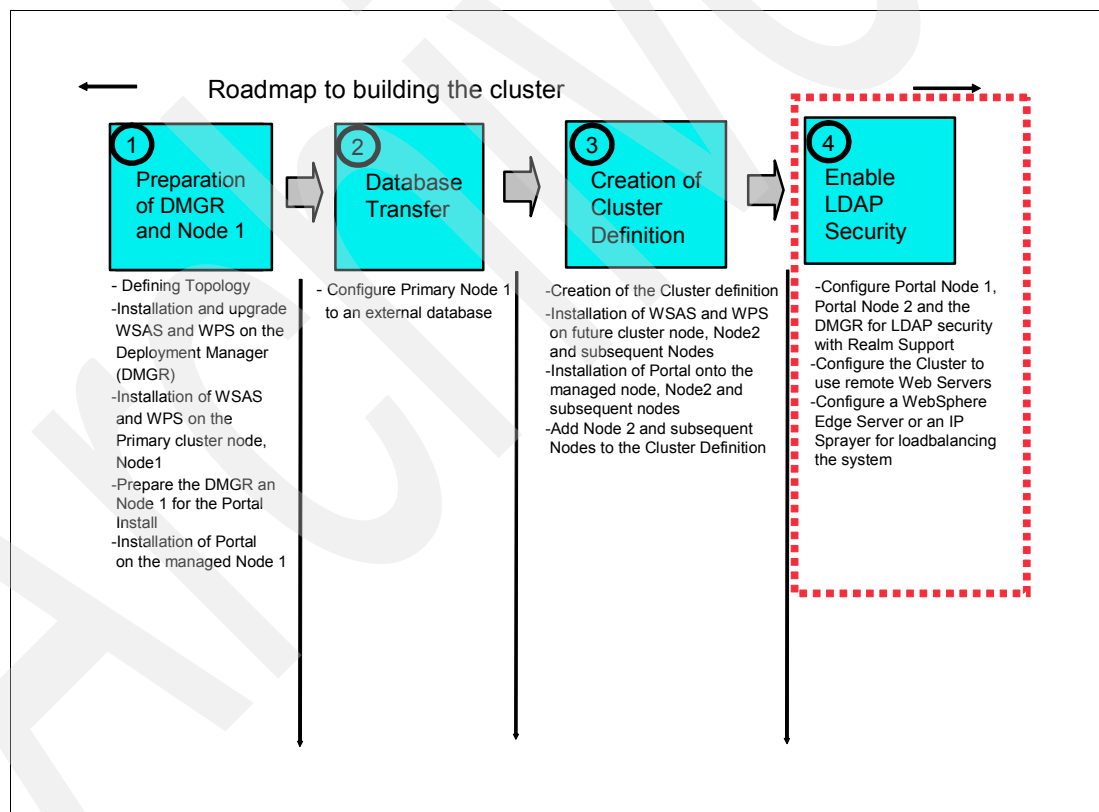


Figure 3-81 Flow chart installation steps: step 4

Refer to the following InfoCenter link for the details of LDAP/security configuration:

http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/topic/com.ibm.wp.ent.doc/wpf/intr_lldap.html

Note: We recommend that you use the enable-security-wmmur-ldap task because overall Portal now recommends using this task to enable security so you can have the flexibility to configure realm support and virtual portals in the future. If you have no plans for these features running this task will NOT cause a problem. Alternately, you can certainly choose to implement other security types at this step by running other tasks, such as enable-security-ldap, and so on.

Ensure that the Portal server was stopped on each node. Also, because security comes enabled by default with Portal V6, we are now required to run the disable-security task BEFORE enabling any type of additional Portal security.

Also, you MUST run the disable-security and the enable-security-wmmur-ldap tasks on the Primary node.

1. Stop the WebSphere_Portal Server instances on all nodes, and check if the node agents and the Deployment Manager are started.
2. Make a copy of the original helper file. Edit the following helper file:
/`<wp_root>/config/helpers/security_disable.properties`
Change the following properties to match your current security configuration:
`wmm.DbPassword`
`WasPassword`
Change the following properties to match what you want your Portal ID and password to be after disabling security:
`PortalAdminId`
`PortalAdminPwd`
`PortalAdminGroupId`
3. Run the config wizard to disable security. Invoke the config wizard by running the following script: `<wp_root>/config/wizard/configwizard.bat`. Again, make sure you run the task on the Primary node.
4. In the “Welcome” window, click Next.
5. In the “Select the Task” window, choose **Disable security**, and click Next as shown in Figure 3-82 on page 124.

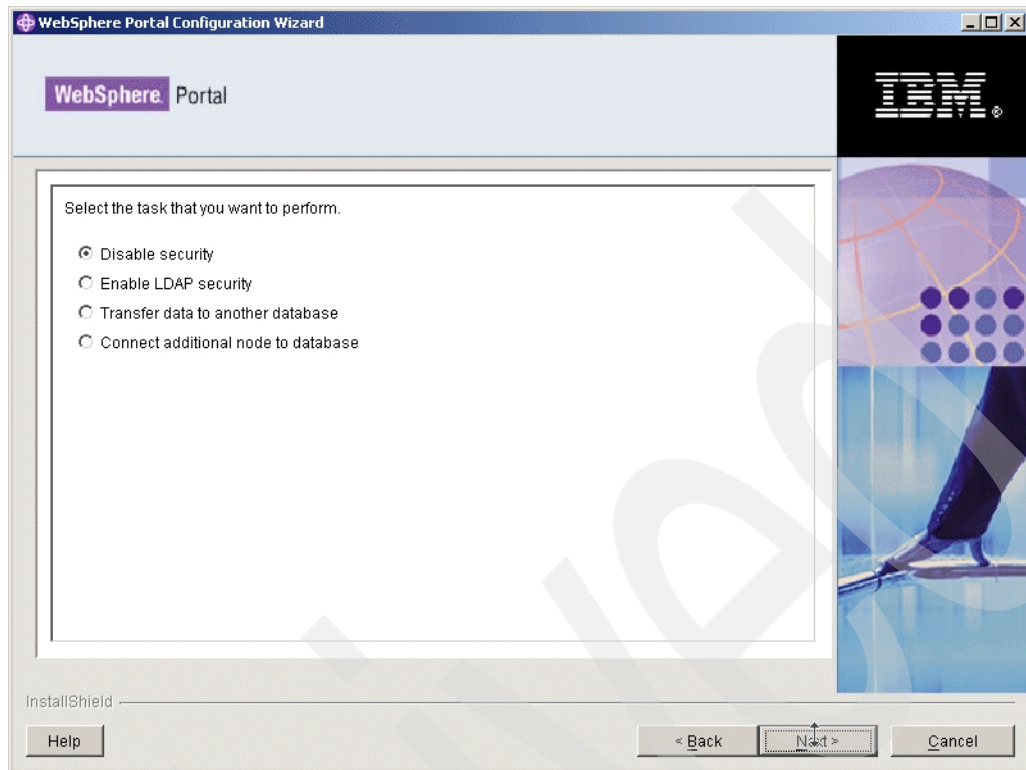


Figure 3-82 Task selection window

6. In the “WSAS global security” window, enter the WSAS Admin password, and click Next.
7. In the “Helper File Location” window, select the proper location of the helper file, and click next as shown in Figure 3-83 on page 125.

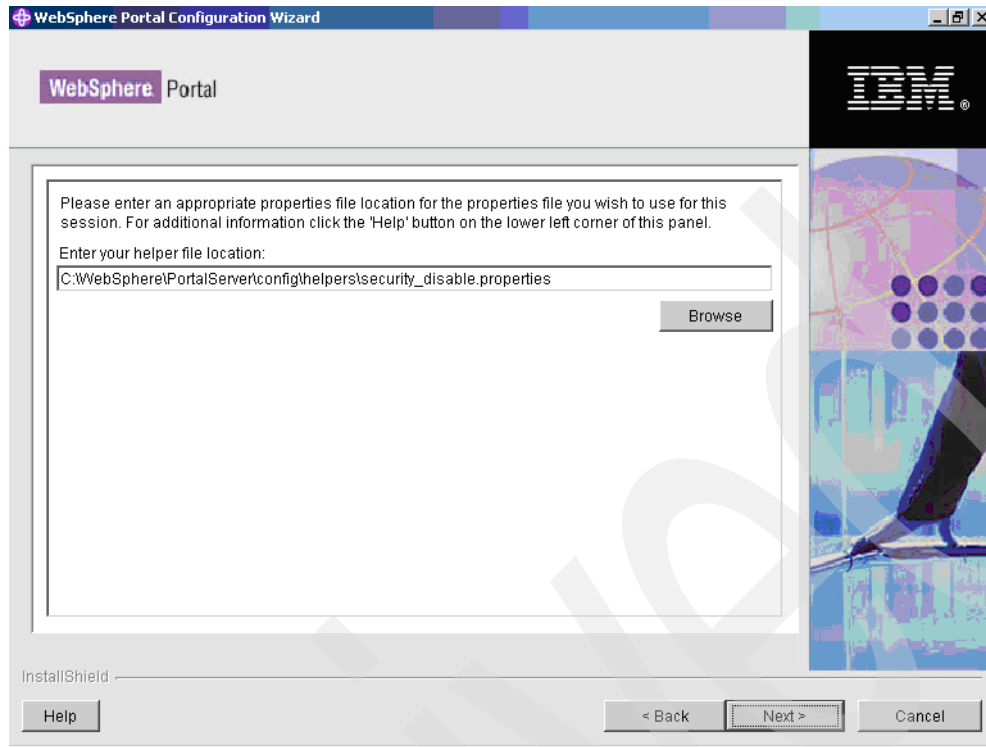


Figure 3-83 Helper file location window

8. In the “Usernames and Passwords” windows, enter the WMM database ID and password. Click Next as shown in Figure 3-84.

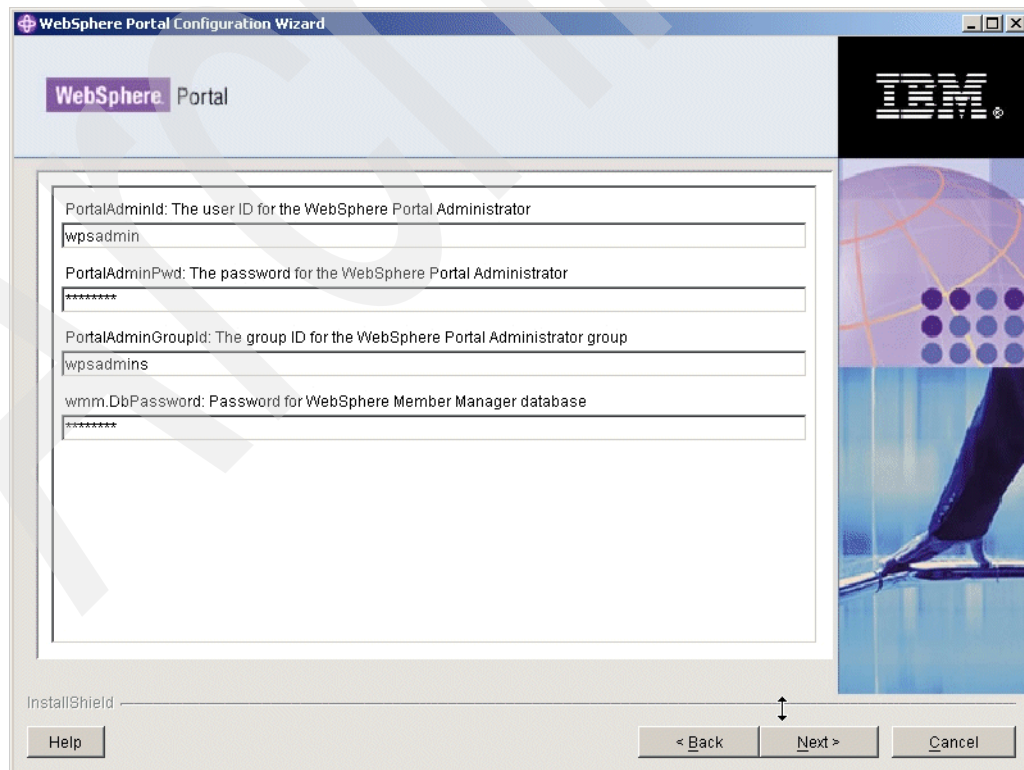


Figure 3-84 WMM database ID and password window

9. Review the summary panel, and click Next to start the task.
10. In the “Wizard Completed Successfully” window, check if the script completed successfully, and click Finish to close the Configuration Wizard.
11. After the disable-security task finishes, ensure that all Portal servers are stopped, and ensure that the nodeagents and the DMGR are running before you run the enable-security-wmmur-ldap task.
12. Browse to the helper files /<wp_root>/config/helpers/, and create a backup copy of the original security helper file. Edit the security helper file to change all the LDAP values to match your LDAP configuration. In our environment it is IBM Directory Server, as shown in Table 3-5.

Table 3-5 Security helper file properties

Property	Description
IBM WebSphere Application Server properties	
WasUserid	The distinguished name in the LDAP directory for the WebSphere Application Server administrator. This can be the same name as the WebSphere Portal server administrator (PortalAdminId). Example: uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com
WasPassword	The password for the WasUserid name.
Database Properties	
wmm.DbPassword	Connection information for wmm DB.
WebSphere Portal server configuration properties	
PortalAdminId	The distinguished name of the WebSphere Portal server administrator in the LDAP directory. This name must be a member of the WebSphere Portal server administrators group defined by the PortalAdminGroupId property. Note: This account must include a value for the mail attribute. If the account does not have a value for the mail attribute, enabling LDAP security will fail. Example: uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com
PortalAdminPWD	Password for the WebSphere Portal server administrator. Note: Do not include the following characters in the password because they can cause authentication failures: ! @ () # \$ %
PortalAdminGroupIdShort	The short form of the WebSphere Portal server administrators group name. Example: wpsadmins
WebSphere Portal server security properties	
LTPAPassword	The password used to encrypt and decrypt the LTPA keys.
LTPATimeout	Time period in minutes at which an LTPA token expires. Example: 120

Property	Description
SSORequiresSSL	Indicates whether single sign-on is enabled only for HTTPS Secure Socket Layer (SSL) connections. Type <code>false</code> . If you want to configure SSL, do so only after you enable LDAP security and verify the LDAP directory configuration.
SSODomainName	The domain name for all single sign-on hosts. Example: <code>cam.itso.ibm.com</code>
General global security properties	
useDomainQualifiedUserNames	Indicates whether to qualify user names with the security domain within where they reside (<code>true</code> or <code>false</code>). The default value (<code>false</code>) is recommended for most environments.
cacheTimeout	Time out for the security cache. The default value (600) is recommended for most environments.
issuePermissionWarnings	Indicates whether during application deployment and application start, the security run time emits a warning if applications are granted any custom permissions (<code>true</code> or <code>false</code>). The default value (<code>true</code>) is recommended for most environments.
activeProtocol	The authentication protocol for RMI/IIOP requests when security is enabled. The default value (<code>BOTH</code>) is recommended for most environments.
activeAuthmechanism	The authentication mechanism when security is enabled. The default value (<code>LTPA</code>) is recommended for most environments.
LDAP properties	
LookAside	You can either install with LDAP only or with LDAP using a Lookaside database. The purpose of a Lookaside database is to store attributes that cannot be stored in your LDAP server. This combination of LDAP plus a Lookaside database is needed to support the database user registry. Value type: * <code>true</code> - LDAP + Lookaside database * <code>false</code> - LDAP only Default value: <code>false</code>
LDAPHostName	The host name for your LDAP server. Example: <code>dbids.cam.itso.ibm.com</code>
LDAPPort	The LDAP server port number. Typically, you type 389. Do not type 636, which is the port typically used for SSL connections. If you want to configure an SSL port for LDAP, do so after you enable LDAP security and verify the LDAP directory configuration.

Property	Description
LDAPAdminUId	<p>The distinguished name in the LDAP directory that WebSphere Portal server and WebSphere Member Manager use to bind to the directory. The level of access given this name determines the level of access that Workplace Collaboration Services has to the directory. This name does not have to contain a uid attribute.</p> <p>Note: Give this account read-only access to prevent users from using the Sign-up link to register accounts in the directory and from using the Edit My Profile link to change attributes in the directory, such as their e-mail addresses.</p> <p>Example: uid=ldapadmin,cn=users,dc=IBM,dc=com</p>
LDAPAdminPwd	The password for the name assigned to the LDAPAdminUId property.
LDAPServerType	Do not change; instead, leave as IBM_DIRECTORY_SERVER.
LDAPBindID	<p>Distinguished name that WebSphere Application Server uses to bind to the directory.</p> <p>Example: uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com</p>
LDAPBindPassword	The password for the LDAPBindID name.
Advanced LDAP properties	
LDAPSuffix	<p>The LDAP suffix for your directory server. This property determines the naming context at which to begin directory searches for users and groups. Tip: For Domino as LDAP this value is typically empty</p> <p>Example: dc=IBM,dc=com</p>
LDAPUserPrefix	<p>The leftmost attribute of user names in the directory. Type the value in lowercase characters.</p> <p>Example: uid</p>
LDAPUserSuffix	<p>The naming context at which to begin searches for user names in the directory.</p> <p>Example: cn=users</p> <p>Do not include the LDAPSuffix value as part of this value. For example, do not type cn=users,dc=IBM,dc=com.</p>
LDAPGroupPrefix	<p>The leftmost attribute of group names in the directory. Type the value in lowercase characters.</p> <p>Example: cn</p>
LDAPGroupSuffix	<p>The naming context at which to begin searches for group names in the directory. Tip: For Domino as LDAP this value is typically empty</p> <p>Example: cn=groups</p> <p>Do not include the LDAPSuffix value as part of this value. For example, do not type cn=groups,dc=IBM,dc=com.</p>
LDAPUserObjectClass	<p>The object class used for users.</p> <p>Example: inetOrgPerson</p>

Property	Description
LDAPGroupObjectClass	The object class used for groups. Example: groupOfUniqueNames
LDAPGroupMember	The attribute used for the members of groups. Example: uniqueMember
LDAPUserFilter	The filter used to search for user accounts. The filter must include the following text: (&((<userprefix>=%v)(mail=%v))(objectclass=<userobjectclass>)), where <userprefix> is the value specified for the LDAPUserPrefix property and <userobjectclass> is the value specified for the LDAPUserObjectClass property. Example: (&((uid=%v)(mail=%v))(objectclass=inetOrgPerson))
LDAPGroupFilter	The filter used to search for groups accounts. The filter must include the following text: (&(<groupprefix>=%v)(objectclass=<groupobjectclass>)), where <groupprefix> is the value specified for the LDAPGroupPrefix property and <groupobjectclass> is the value specified for the LDAPGroupObjectClass property. Example: (&(cn=%v)(objectclass=groupOfUniqueNames))
LDAPGroupMinimumAttributes	Attributes loaded for group searches and related to performance. Leave this property blank.
LDAPUserBaseAttributes	Attributes loaded for user login-related to performance. Type givenName, sn, preferredLanguage. Also type the following values to allow users, for example, calendar users, to set international time and date preferences in the Edit My Profile page: ibm-regionalLocale, ibm-timeZone, ibm-preferredCalendar, ibm-firstDayOfWeek, ibm-firstWorkDayOfWeek
LDAPUserMinimumAttributes	Attributes loaded for user searches and related to performance. Leave this property blank.
LDAPsearchTimeout	Value in seconds for the amount of time the LDAP server has to respond before canceling a request. Example: 120
LDAPreuseConnection	Indicates whether LDAP connections are reused (true or false). If your environment uses a front-end server to spray requests to multiple back-end LDAP directory servers, type false. If your environment does not use an intermediate server but instead authenticates directly with the LDAP directory server, type true.
LDAPIgnoreCase	Indicates whether LDAP searches are case-sensitive (true or false).
PDM LDAP Properties	
WpsContentAdministrators	The group ID for the WebSphere Content Administrator group Example: cn=wpsContentAdministrators,cn=groups,ou=firebrigade,dc=IBM,dc=com
WpsContentAdministratorsShort	The WebSphere Content Administrators group ID. Example: wpsContentAdministrators

Property	Description
WpsDocReviewer	The group ID for the WebSphere Document Reviewer group. Example: cn=wpsDocReviewer,cn=groups,ou=firebrigade,dc=IBM,dc=com
WpsDocReviewerShort	The WebSphere Document Reviewer group ID. Example: wpsDocReviewer
WCM LDAP Properties	
WcmAdminGroupId	The group ID for the Web Content Management Administrators group. This should be the fully qualified distinguished name (DN) of a current administrative user for the WebSphere Application Server. For LDAP configuration this value should not contain spaces. Example: cn=wcmadmins,cn=groups,ou=firebrigade,dc=IBM,dc=com
WcmAdminGroupIdShort	The Web Content Management Administrators group ID. Example = wcmadmins

13. Import the contents of the helper file into the wpconfig.properties file by issuing the following command:

```
<wp_root>/config/WPSconfig -DparentProperties="<full_path_to_helper_file>"
-DsaveParentProperties=true
```

14. Open the wpconfig.properties file, and make sure the WpsHostName and WpsHostPort are correct.

15. Run the following task to validate the LDAP values:

```
WPSconfig.bat validate-wmmur-ldap
```

16. Run the following task on the primary node ONLY to configure the LDAP security settings for both WSAS/WP nodes and the DMGR. This enables security on the entire cluster:

```
WPSconfig.bat enable-security-wmmur-ldap >enable.log
```

Hint: To enable LDAP with Realm support you cannot use the Configuration Wizard.

17. Because we enabled security using the enable-security-wmmur-ldap task that enables realm support, we are required to manually edit the wmmWASAdmin.xml file on the DMGR. If this file is not edited with the shortname you cannot run the stopServer.bat or the serverStatus.bat on the nodes using the shortname as the username; instead, you will be required to use the full LDAP DN.

The current <dmgr_profile_root>/config/wmm/wmmWASAdmin.xml should look similar to Example 3-2:

Example 3-2 Example of the .xml file

```
?xml version="1.0" encoding="UTF-8"?>
<wmmWASAdmins>
  <admin loginId=" uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com"
loginPassword="anvu7zPZ7jbrZLa4h89Tfg==" uniqueUserId="
uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com"/>
</wmmWASAdmins>
```

Add another line between the <wmmWASAdmins> tag that includes the shortname. Since both IDs will have the same password you can simply copy the current <admin logonId> tag entry and modify it like Example 3-3:

Example 3-3 Copy and modify current <admin logonId> tag entry

```
<?xml version="1.0" encoding="UTF-8"?>

<wmmWASAdmins>

<adminlogonId=" uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com"
logonPassword="anvu7zPZ7jbrZLa4h89Tfg==" uniqueUserId="
uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com"/>
<adminlogonId="wpsadmin" logonPassword="anvu7zPZ7jbrZLa4h89Tfg==" uniqueUserId="
uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com"/>

</wmmWASAdmins>
```

18. Perform a full synchronization to ensure all the security settings are pushed from the DMGR to the nodes. Restart the DMGR and the nodeagents on each node. Stop the nodeagents by providing the full LDAP DN on the command line. After they restart, the new config settings should take affect and then they should be able to be stopped using the shortname.

19. Stop WebSphere_Portal on the second node.

20. Update the <wp_root>/config/wpconfig.properties file on each secondary node in the cluster with the same LDAP user registry information you used to configure the primary node.

Update the wpconfig.properties by moving the LDAP helper file from node 1 to node 2 and running the following command:

```
<wp_root>/config/WPSconfig -DparentProperties="<full_path_to_helper_file>"
-DsaveParentProperties=true
```

21. Complete the security configuration by running the enable-jcr-security configuration task on each secondary node.

Run the following command from the <wp_root>/config directory:

```
WPSconfig.bat enable-jcr-security -DPortalAdminId=portal_admin_id >enable.log
```

where portal_admin_id is the fully qualified distinguished name (DN) of the portal administrator (for example, uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com).

22. Restart the Portal server cluster member on each secondary node.

23. Verify the new security settings by rendering the DMGR AdminConsole and Portal from a browser.

3.4.11 Configure Portal to use a remote Web server

With WSAS 6, the Web server architecture changed significantly. The Web server is now listed as a separate server in the AdminConsole and can be managed from there as well.

Details on how to configure a Web server to WSAS 6 can be found in the WSAS InfoCenter at the following Web sites:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/tins_webplugins_single.html

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/tins_webplugins_remotesa.html

Use the following steps to configure Portal to use a remote Web server:

1. Install the Web server on a remote machine.
`/<cd root>/W-9/IHS/install.exe`
2. Install the plug-in on a remote machine.
`/<cd_root>/W-9/plugin/install.exe`
3. During the installation of the WebSphere Plug-in on the Web Server a script was created. By running this script, all the Web server settings are automatically added to the DMGR Server.
4. Move `configurewebserver1.bat` script from `<plugin_root>/bin` (Example: `C:\WebSphere\Plugins\bin`) to the `<wsas_root>/bin` on the DMGR machine.
5. Edit the `<dmgr_profile_root>/properties/soap.client.props` file temporarily, and add the current user ID and password for the following properties:

`com.ibm.SOAP.loginUserId`

`com.ibm.SOAP.loginPassword`

6. Open a command prompt on the DMGR, and browse to `<wsas_root>/bin`. Run the bat file by typing the following command:

`configurewebserver1.bat`

This script creates the Web server node in the AdminConsole.

The script also tries to map all the existing Enterprise Applications (EAs) to the Web server entry, but may fail on some Windows environments because of the fact that some of the Portal EAs have more than 256 characters in their paths.

The results are that after this fails the node and server entry are created successfully, but none of the EAs are mapped to the Web server. This means that when you regenerate the Web server plug-in it does not know about any of the EAs. Therefore, none of the EAs are listed in the `plug-in-cfg`, which means that if the plug-in is moved to Web server, it cannot serve the EAs.

7. If the script runs successfully, you need to log out and log back into the DMGR AdminConsole to see the changes before moving to step 8 on page 134.

If you are on Windows and the script fails because of the 256 character limit, you must follow the following procedure to manually map the EAs to the Web server.

- a. To accomplish this you must manually map each of the Enterprise Applications to the `WebSphere_Portal` server AND the `webserver1` server through the AdminConsole by first navigating to `Applications → Enterprise Applications`.
- b. In this example, we will map the `wps` EA as an example. This should be done for each EA that you want the Web server to serve.

Click the EA name, in this case “wps” as shown in Figure 3-85 on page 133.

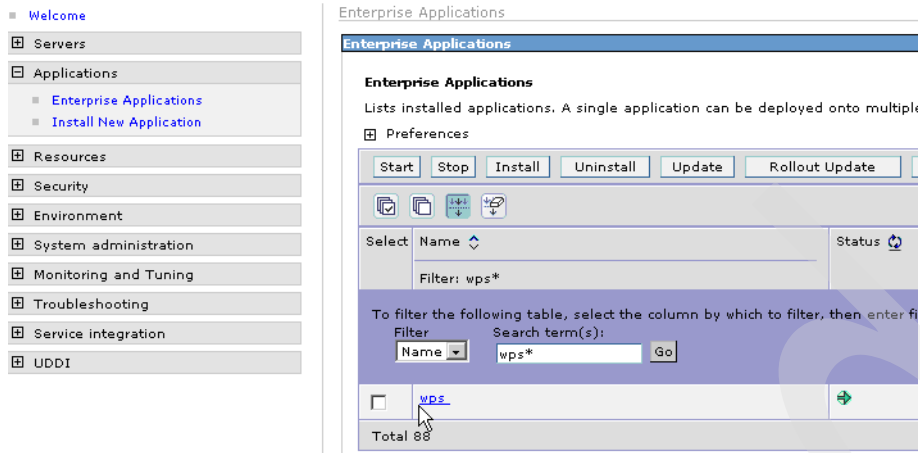


Figure 3-85 Enterprise applications view in the Deployment Manager Administrative Console

c. Click Map modules to servers as shown in Figure 3-86.

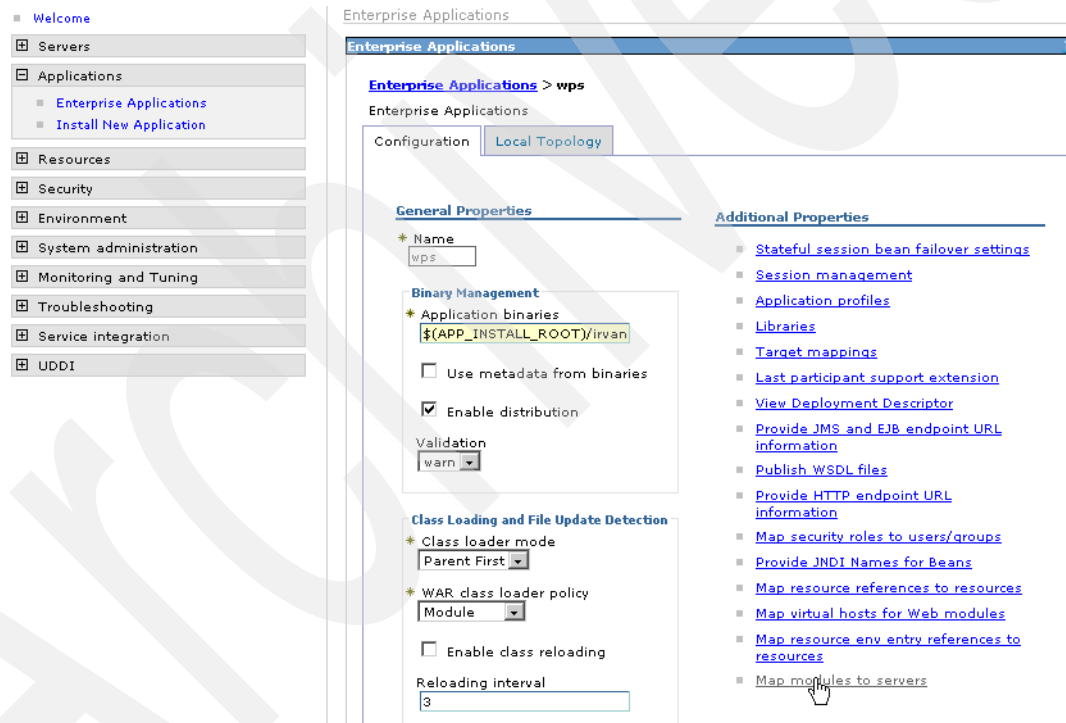


Figure 3-86 Map modules to servers

d. Select the Module, WebSphere Portal Server (wps.war), and then highlight both the webserver1 and WebSphere_Portal entries listed in the Clusters and Servers box. Click Apply as shown in Figure 3-87 on page 134.

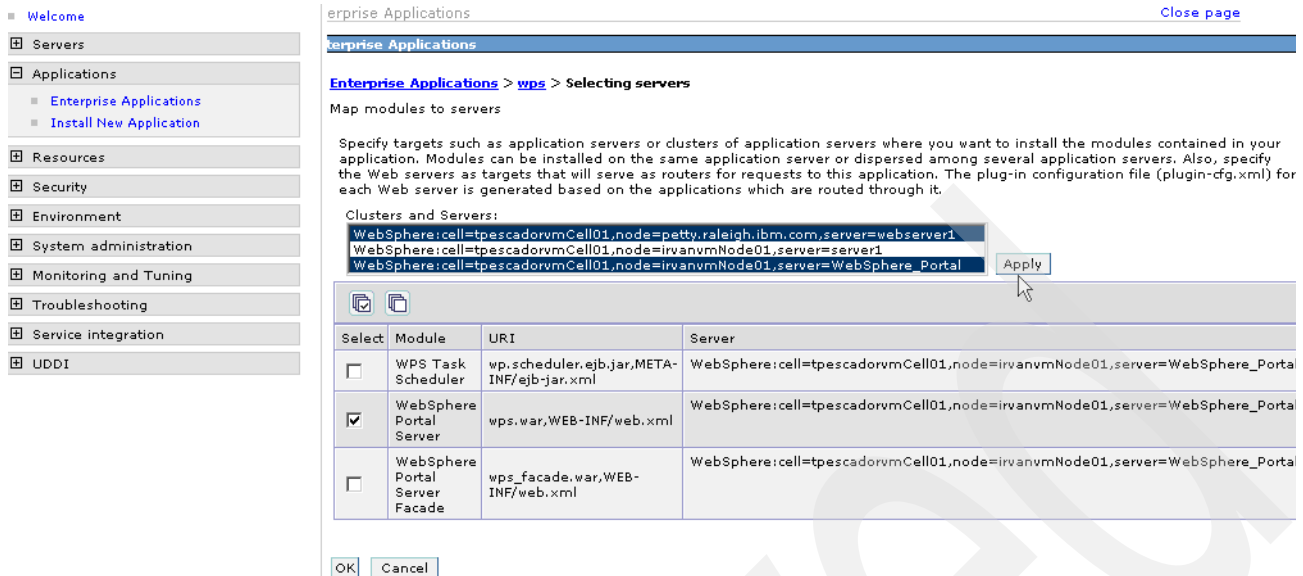


Figure 3-87 Map modules to server

- e. Now you will see that the Module WebSphere Portal Server (wps.war) is now mapped to both servers as shown in Figure 3-88.

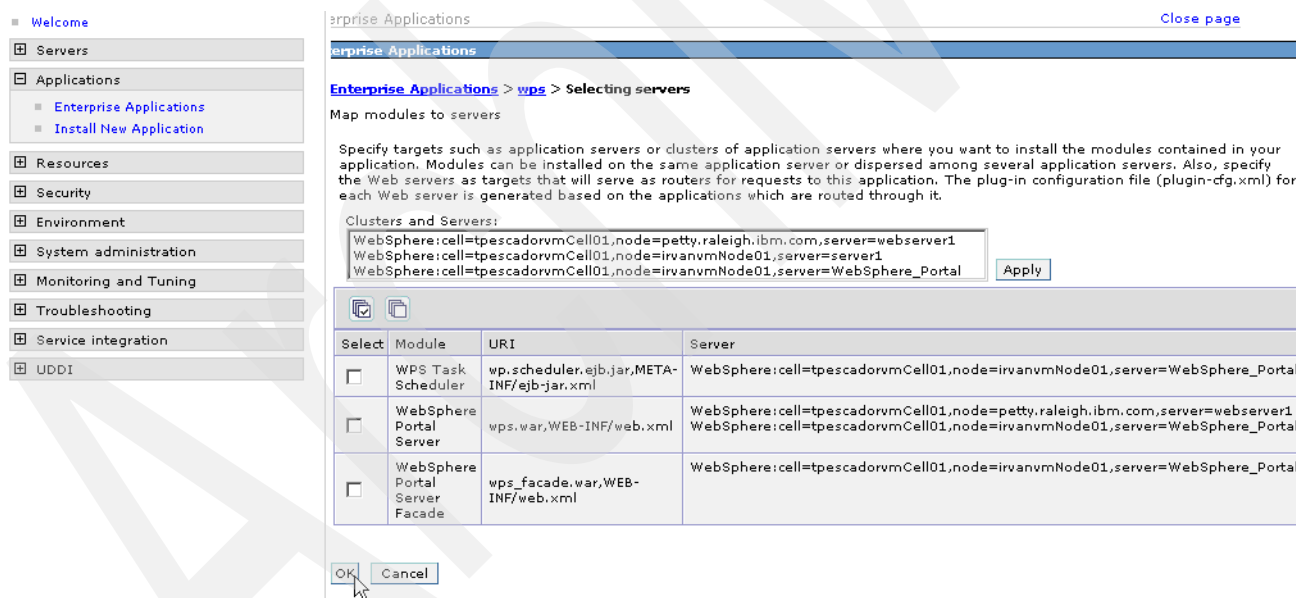


Figure 3-88 Module WebSphere Portal Server (wps.war) is now mapped to both servers

8. Regenerate the plug-in by navigating to Servers → Web servers. Choose the webserver1 entry, and click Generate Plug-in as shown in Figure 3-89 on page 135.

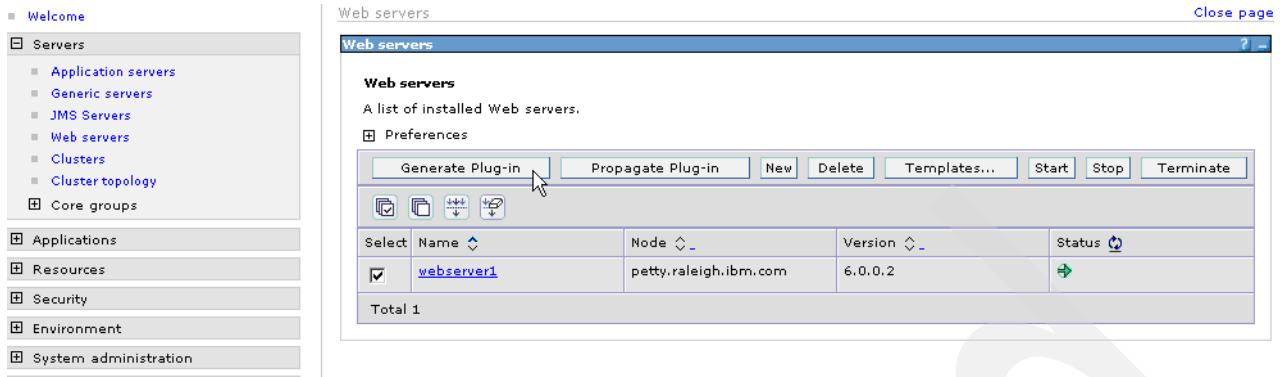


Figure 3-89 Regenerate the plug-in

9. Copy the plug-in from the DMGR:

```
<dmgr_profile_root>/<profile_name>/config/cells/<cellname>/nodes/<nodename>/server
s/webserver1/plugin-cfg.xml
```

to the following remote Web server path:

```
<plugin_root>/config/webserver1
```

10. Change the WpsHostName and WpsHostPort properties in the wpconfig.properties on all nodes to reflect the Web server values.
11. Restart the DMGR, Web server, and Portal.
12. Verify that the Portal can be accessed through the Web server.

Note: At this point the cluster setup is finished. If you want to add an additional node to the cluster please proceed with the steps in section 3.6, “Adding a Horizontal node to a cluster with WebSphere Process Server” on page 168.

3.4.12 Optional steps after cluster creation

Following are some optional steps you can perform at you create the cluster.

- ▶ Configure a WebSphere Edge Server or an IP Sprayer for load balancing the system (This is not described in this book.).
- ▶ Configure the WebSphere Portal Cluster to us a second LDAP directory as described in Chapter 4, “Multiple LDAP directory support” on page 187.
- ▶ Configure the WebSphere Portal Cluster to share the database with a second clustered environment as described in Chapter 5, “Database domains” on page 233.

3.5 Horizontal cluster (without WebSphere Process Server) installation and configuration steps

In this section we provide a step-by-step guide to configuring a WebSphere Portal v6.0.0.0 horizontal cluster using WebSphere Application Server v6.0.2.9 (without WebSphere Process Server).

In a horizontal cluster topology members of a WebSphere Portal cluster exist on multiple physical machines representing each node, effectively and efficiently distributing the workload of a single logical WebSphere Portal image.

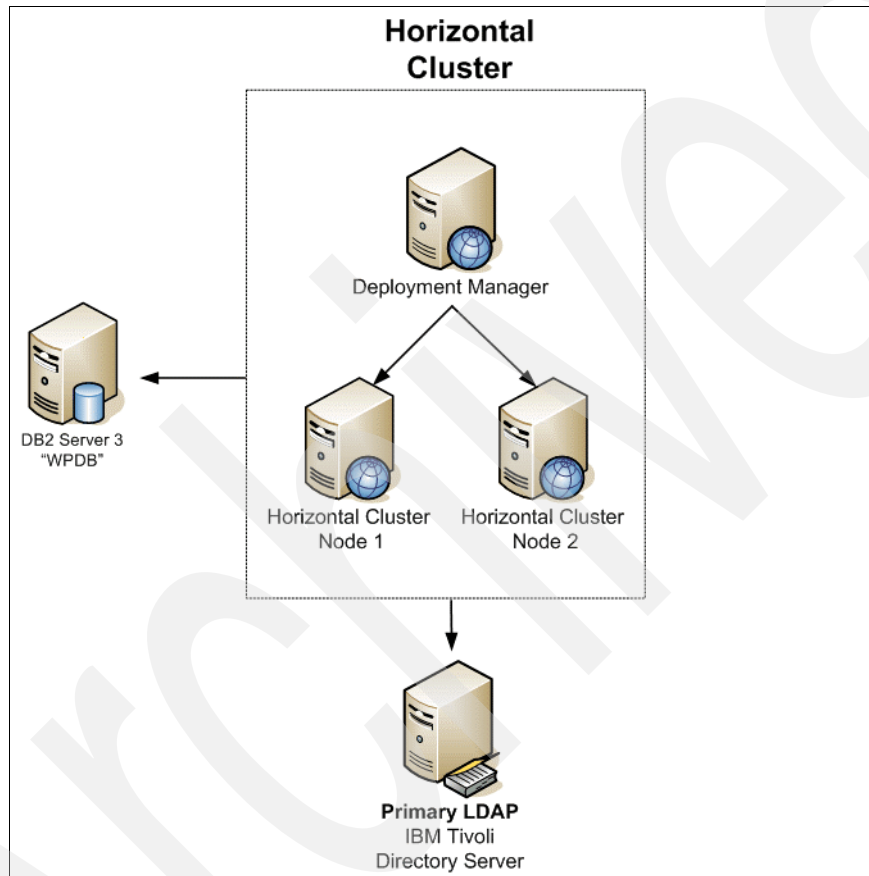


Figure 3-90 Horizontal cluster built in section 3.3

In our deployment the following applies:

DMC3: This machine served as the Deployment Manager. The DMGR is the central administrative console for all nodes in the environment.

HORIZC3N1 and HORIZC3N1: nodes 1 and 2 hosted the business logic. After completion of the installation, both machines were clustered for load balancing and failover.

DBDOM7: This served as the RDBMS server that hosted the IBM WebSphere Portal database for all nodes and for the DMGR.

DBIDS represents our LDAP server (IBM Tivoli Directory Server V6), which hosted all directory information used by WebSphere Portal.

Installation steps

In Figure 3-91, we show an outline of the steps we walk you through as part of the installation and configuration steps for a horizontal cluster with WebSphere Process Server.

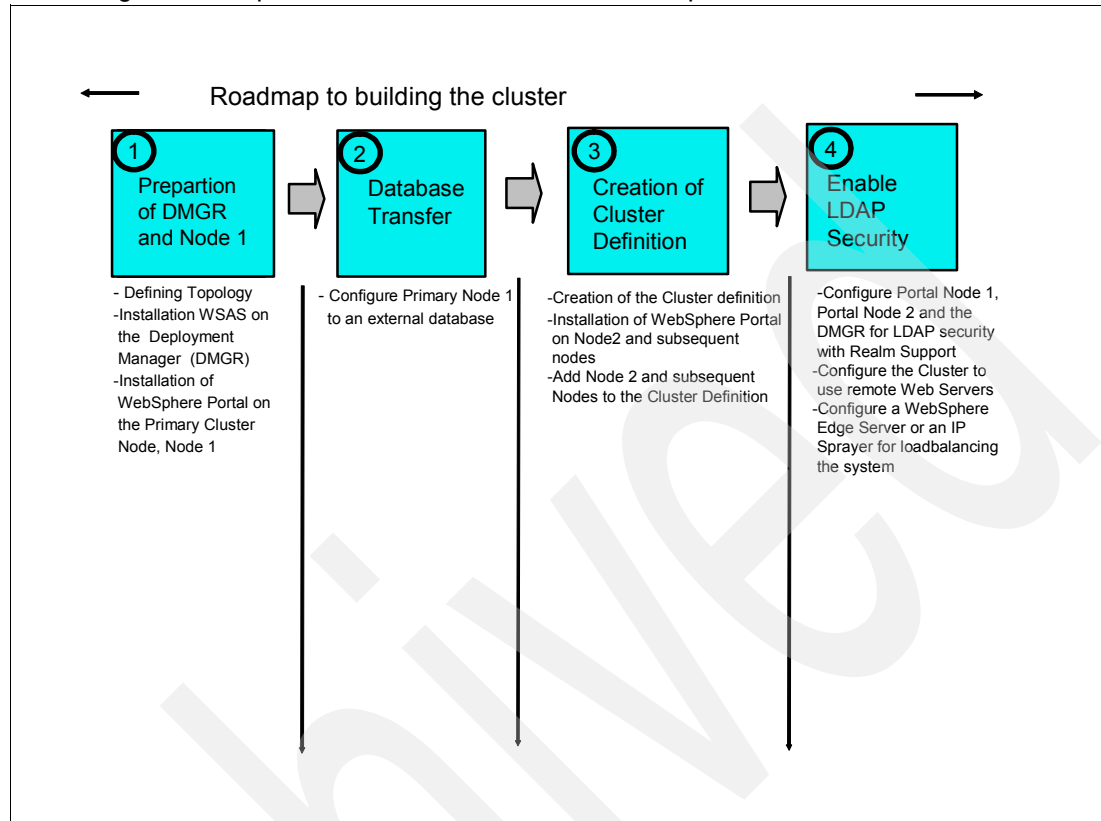


Figure 3-91 Flow chart for installation and configuration steps

A clustered deployment of IBM WebSphere Portal Server V6 involves the following:

- ▶ Installing WSAS on the Deployment Manager.
- ▶ Installing WebSphere Portal on the primary cluster node, node 1.
- ▶ Configuring primary node 1 to an external database.
- ▶ Creating a cluster definition.
- ▶ Installing WebSphere Portal on node 2 and subsequent nodes.
- ▶ Connecting node 2 and subsequent nodes to the database server.
- ▶ Adding node 2 and subsequent nodes to the cluster definition.
- ▶ Configuring Portal nodes and the DMGR for LDAP security with Realm Support.
- ▶ Configuring the cluster to use remote Web Servers.
- ▶ Optional: Configuring a WebSphere Edge Server or an IP Sprayer for load balancing the system (This is not described in this book.).
- ▶ Optional: Configuring the WebSphere Portal Cluster to use a second LDAP directory as described in Chapter 4, "Multiple LDAP directory support" on page 187.
- ▶ Optional: Configure the WebSphere Portal Cluster to share the database with a second clustered environment as described in Chapter 5, "Database domains" on page 233.

3.5.1 Installation WSAS on the Deployment Manager

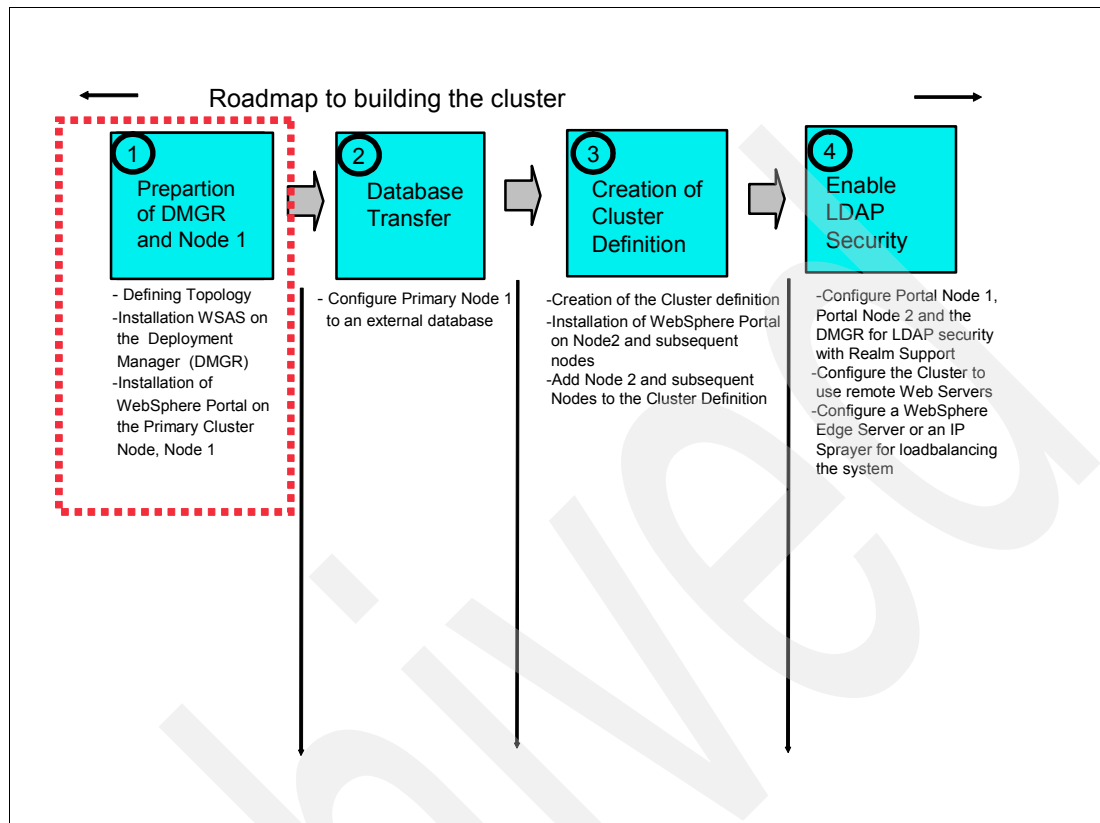


Figure 3-92 Flow chart about the installation steps: step 1

Install the DMGR using the following procedure:

1. Install WSAS DMGR by running the installer from the following location:
`<cd_root>/W-1/windows/ia32/ifpackage/WAS/install.exe`

Note: Make sure the installer window is titled “**Welcome to WebSphere Application Server Network Deployment, V6**”. This title means that you can use this installer to install either, DMGR or WSAS profiles. If the title is “WebSphere Application Server Version 6.0”, you are using an installer that only has the ability to install WSAS profiles but not DMGR profiles.

Note: If you are installing from downloaded code, run the program from a local drive that is not a mapped network drive.

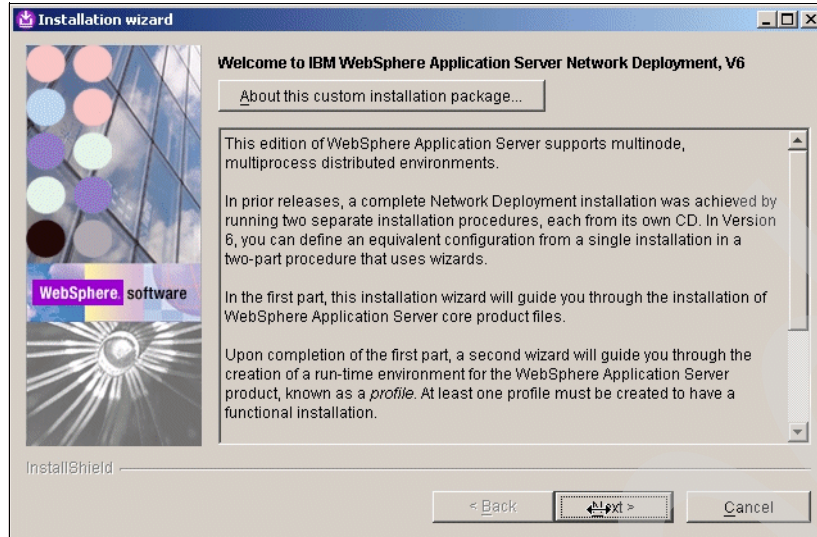


Figure 3-93 Welcome window

2. Accept the License agreement, and click Next.
3. The Installation Wizards performs a pre-requisite check for you when this runs successfully. Click Next; otherwise, check that your operating system meets the requirements for WSAS as discussed in section 2.1, "System requirements" on page 22.
4. If installing on Windows, when asked for the install location, shorten the default path. There is a path name limitation in Windows. Windows cannot handle path names longer than 256 characters. For example:
 D:\IBM\WebSphere\ApplicationServer
 Click Next as shown in Figure 3-94.

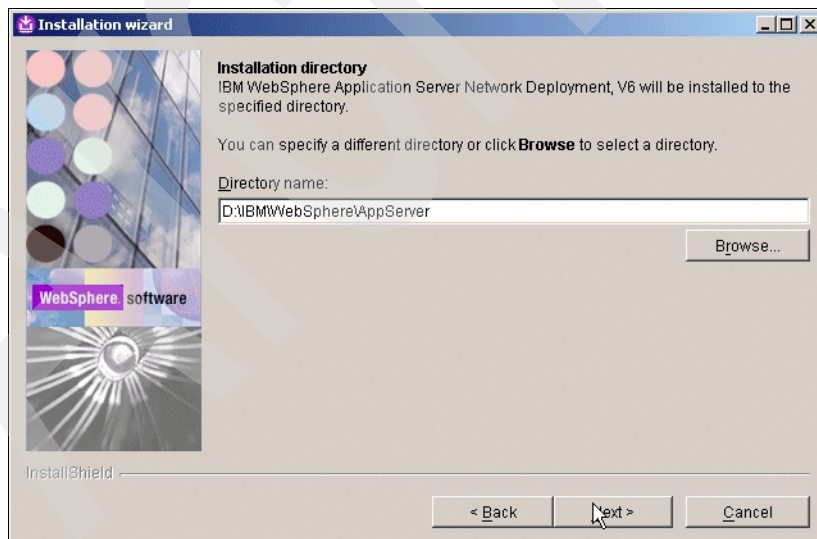


Figure 3-94 Installation directory window

5. The Installation Wizard displays the installation summary including interim fixes to be applied.

Review the options in the installation summary window, and click Next as shown in Figure 3-95.

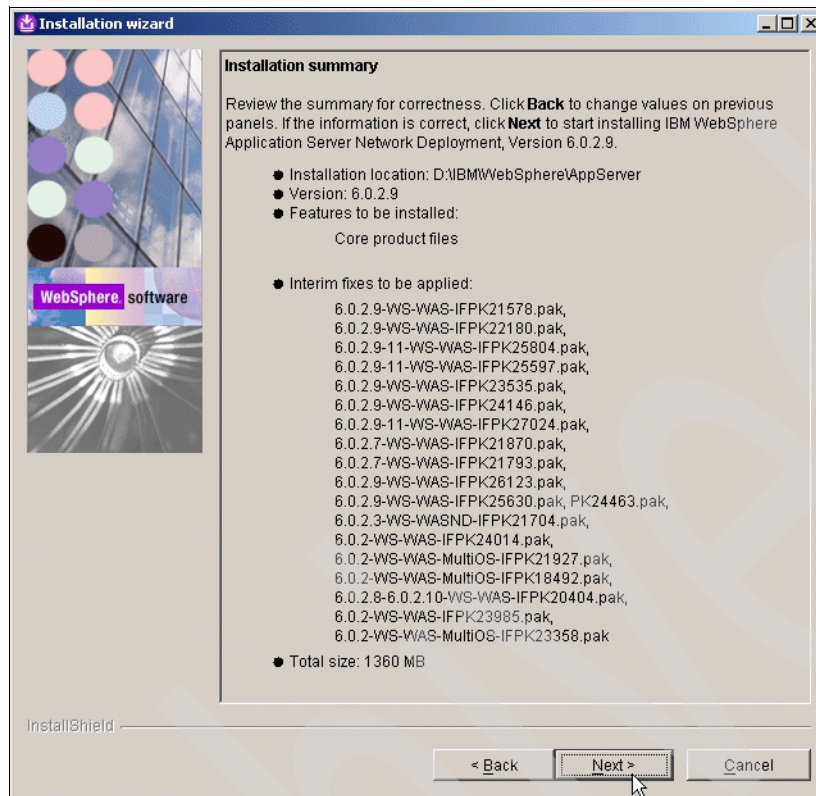


Figure 3-95 Installation summary

6. The WSAS installer from the Portal CDs automatically upgrades WSAS to 6.0.2.9.
7. The WSAS installer from the Portal CDs also automatically applies the WSAS and WPS iFixes required for an install of Portal 6.0.0.0.
 - PK21578
 - PK22180
 - PK25804
 - PK25597
 - PK23535
 - PK24146
 - PK27024
 - PK21870
 - PK21793
 - PK26123
 - PK25630
 - PK24463
 - PK21704
 - PK24014
 - PK21927

- PK18492
- PK20404
- PK23985
- PK23358

Note: You can find WSAS fixes by visiting the following Web site, and then searching for the fix:

<http://www-306.ibm.com/software/webserver/appserv/was/support/>

8. During the install (with a panel near the end) you are prompted to create a profile. Select the **Launch the Profile creation wizard** option to create a profile. Click Next as shown in Figure 3-96.

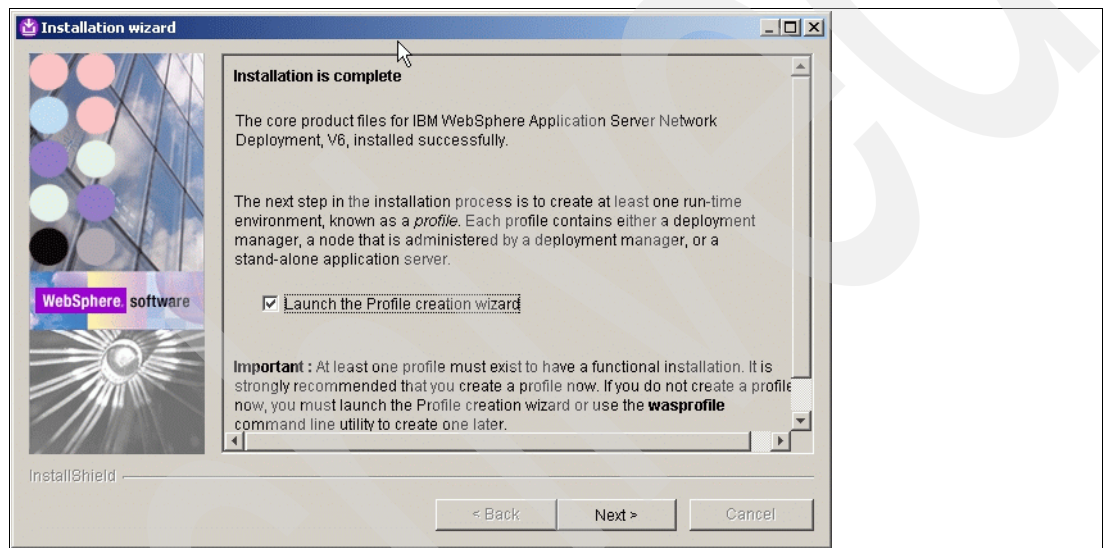


Figure 3-96 Installation completed window

9. In the “Welcome to Profile Creation Wizard”, click Next as shown in Figure 3-97 on page 142.

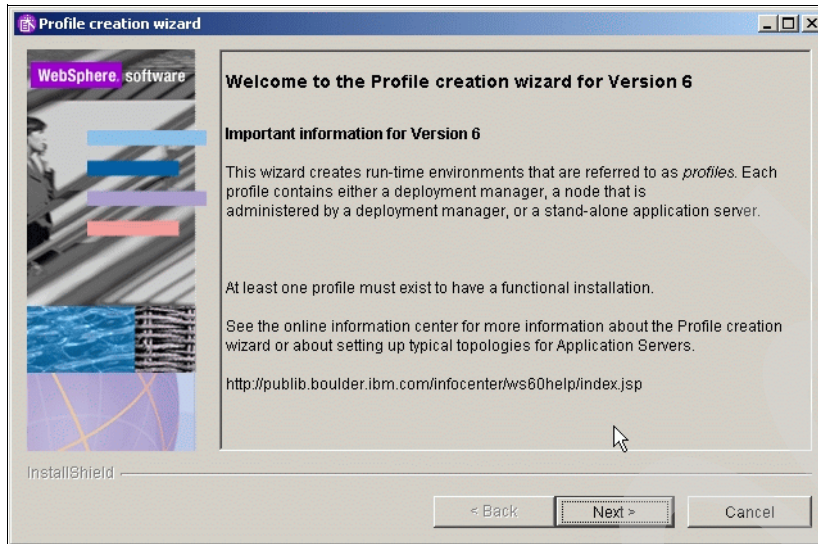


Figure 3-97 Welcome to Profile Creation Wizard window

10. In the “Profile Selection” window, select the **Create a Deployment Manager profile** option, and click Next as shown in Figure 3-98.

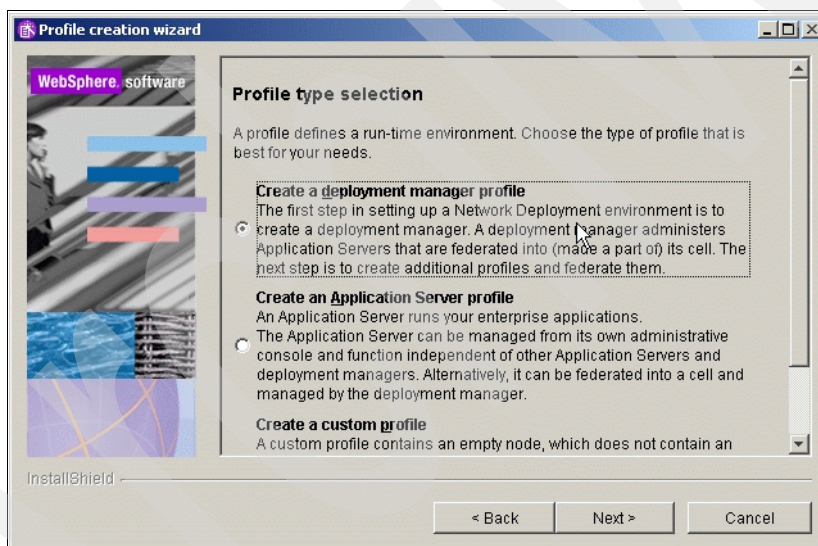


Figure 3-98 Profile type selection window

11. In the “Profile name” window, enter a profile name for the WebSphere Application Server Instance. This name can be different for each node and for the DMGR. In our setup we chose the node name as the profile name. Click Next as shown in Figure 3-99 on page 143.

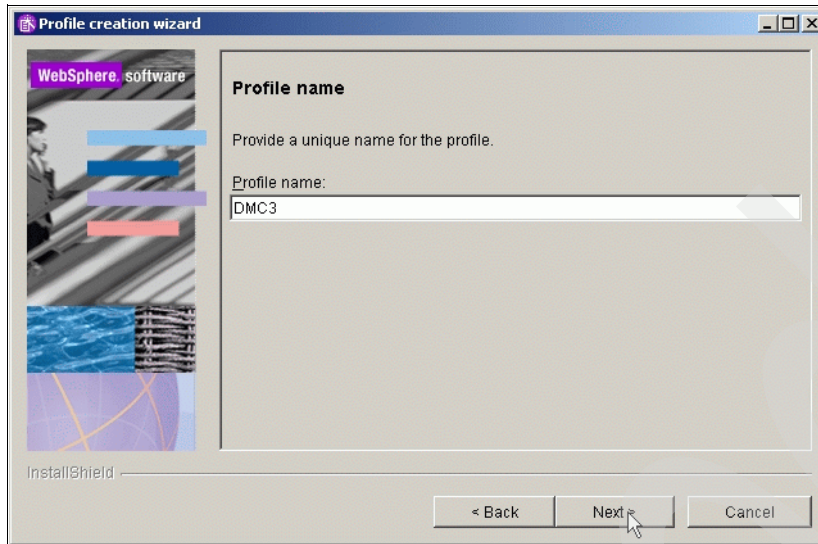


Figure 3-99 Profile name window

12. In the “Profile directory” window, verify the directory name, and click Next as shown in Figure 3-100.

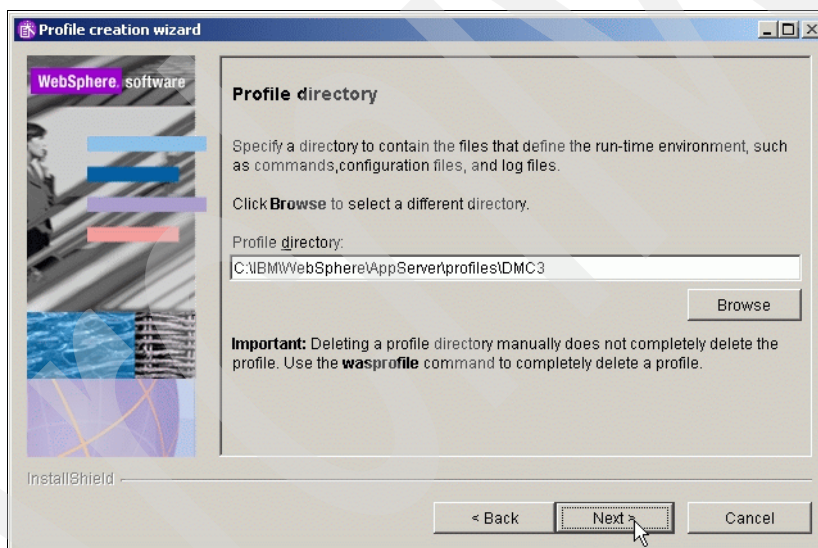


Figure 3-100 Profile directory window

13. In the “Node, host, and cell names” window, check the values in the Node name, Host name, and Cell name fields. Click Next as shown in Figure 3-101 on page 144.

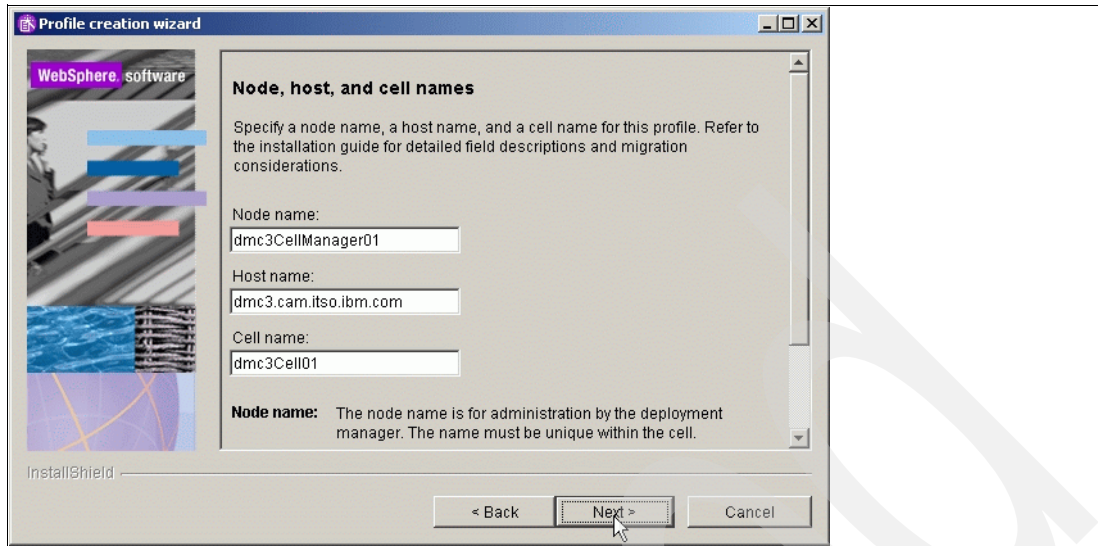


Figure 3-101 Node, host, and cell names window

14. In the “Port value assignment” window, check the values of Ports, and click Next as shown in Figure 3-102.

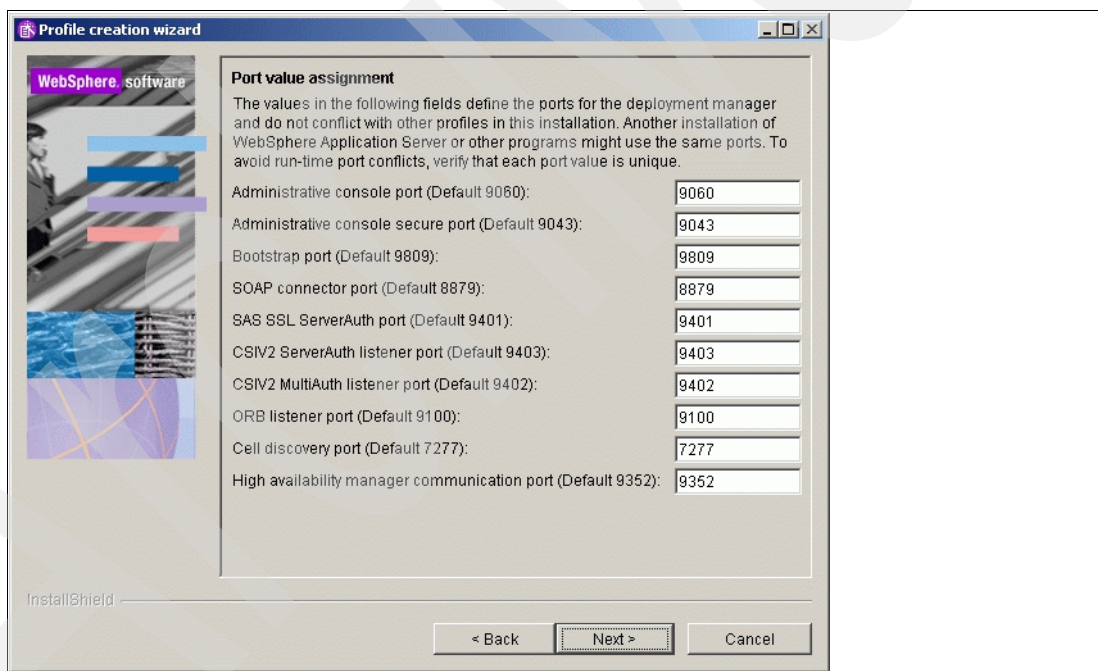


Figure 3-102 Port value assignment window

15. In the “Windows Service Definition” window, you can configure WSAS to run as a service. In our setup we do not want to run the WebSphere Application Server as a Windows Service because of the longer start up time of the Windows system, so leave the defaults and click Next as shown in Figure 3-103 on page 145.

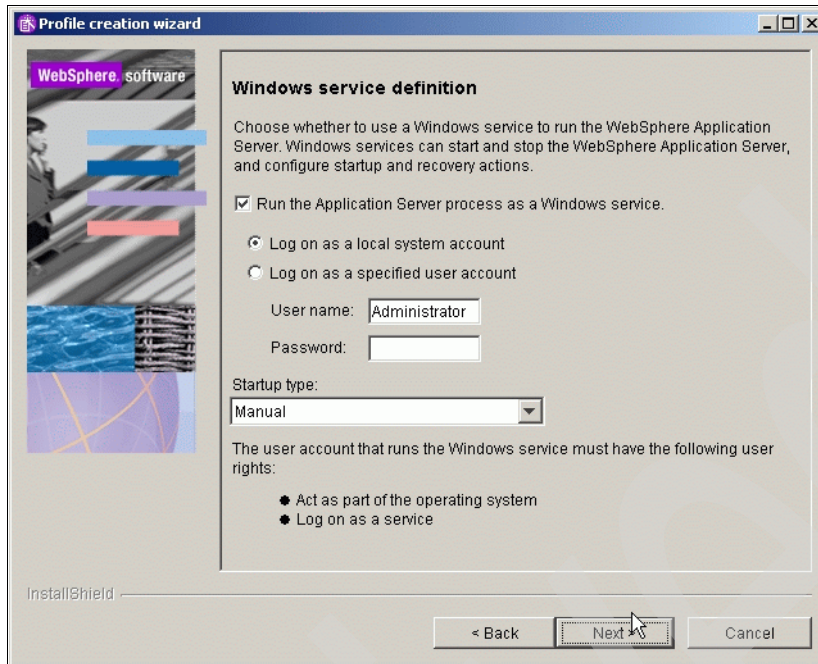


Figure 3-103 Run as Service window

16. In the “Profile Summary” window, check the values, and then click Next as shown in Figure 3-104.

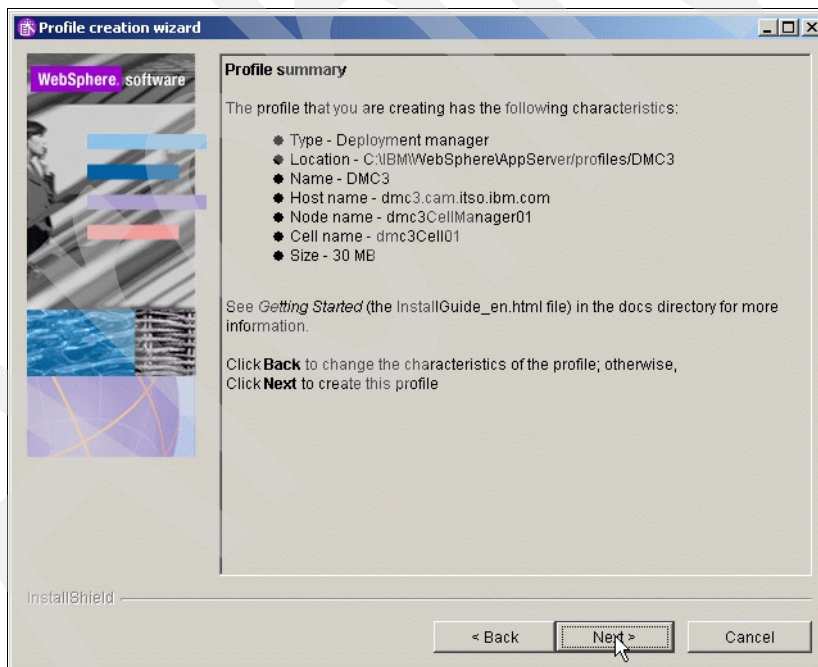


Figure 3-104 Profile creation summary window

17. In the “Profile creation is complete” window, check that the creation was successful. Click Finish as shown in Figure 3-105 on page 146.

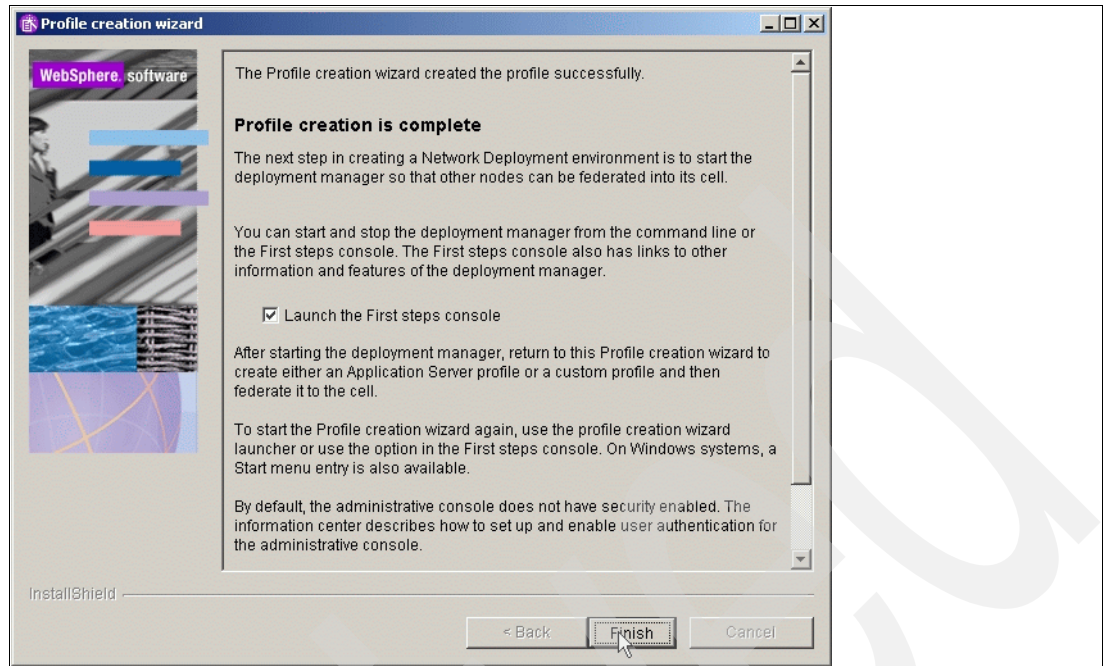


Figure 3-105 Profile creation is complete window

Now the installation steps for the DMGR server are completed, and you can proceed to install the primary node.

3.5.2 Install WebSphere Portal on the primary cluster node, node 1

The following section provides step-by-step instructions on installing WebSphere Portal without WebSphere Process Server.

Note: If you are installing from downloaded code, run the program from a local drive and not a mapped network drive.

1. Launch the install.bat script in the W-Setup folder of your Portal 6 Gold code.
2. Accept the License agreement, and click Next.
3. In the "Installation Type" window, select **typical**, and click Next as shown in Figure 3-106 on page 147.

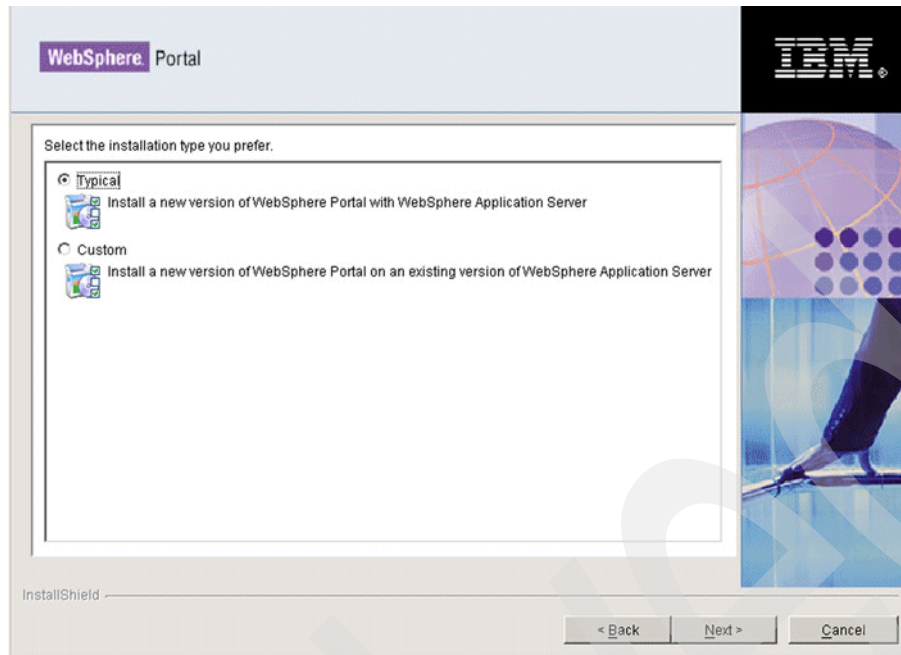


Figure 3-106 Installation type window

4. In the “Installation directory” window, type the directory where you want to install WebSphere Application Server. If the directory that you specify does not exist, it will be created. For example:
D:\IBM\WebSphere\ApplicationServer
Click Next.
5. In the “Properties for this instance” window, verify the cell name, node name, and host name of WebSphere Application Server information, and click Next as shown in Figure 3-107 on page 148.

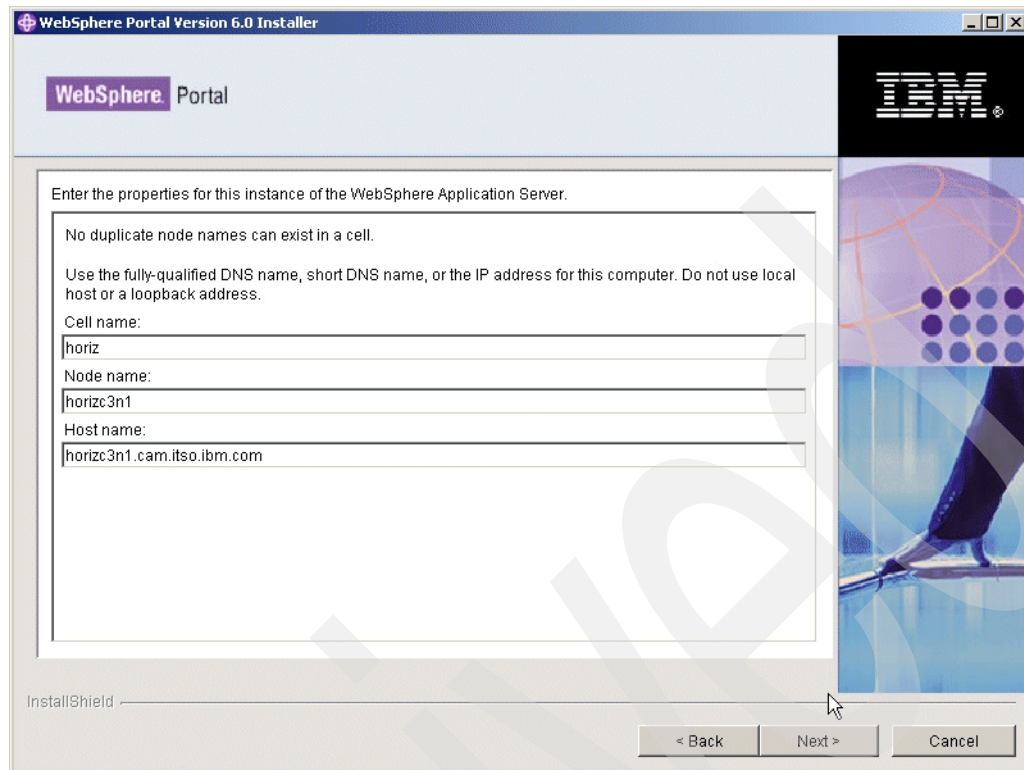


Figure 3-107 Specify cell name, node name, and host name information window

6. In the “Was administrative user” window, type the User ID and password for the WebSphere Application Server administrator. Click Next as shown in Figure 3-108.

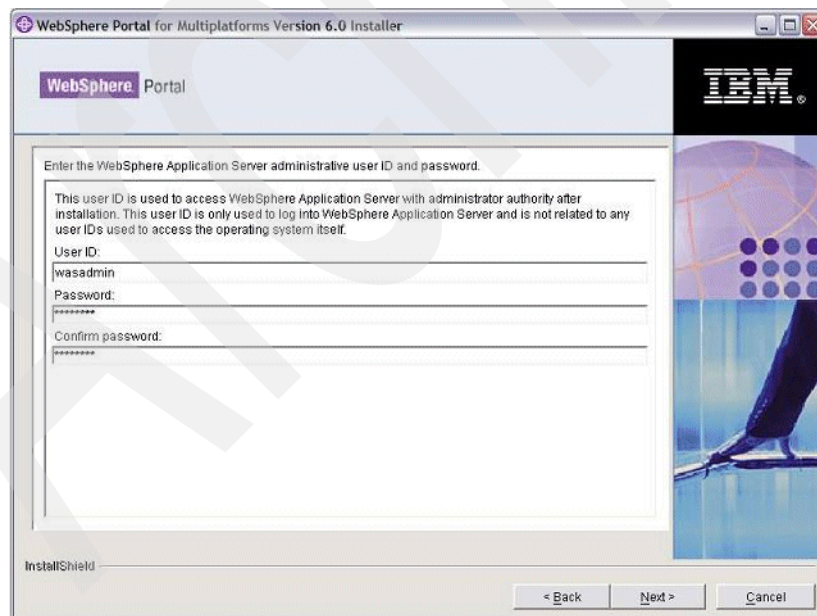


Figure 3-108 Specify user ID and password window

7. In the “Install Business Process Support”, select “Do not install WebSphere Process Server”, otherwise you will not be able to add that node to a managed cell or use it as part of a cluster at a later time., if you want to enable the Business Process Support please refer to section 3.4, “Horizontal cluster (includes WebSphere Process Server) installation and configuration steps” on page 54 and Click Next as shown in Figure 3-109.

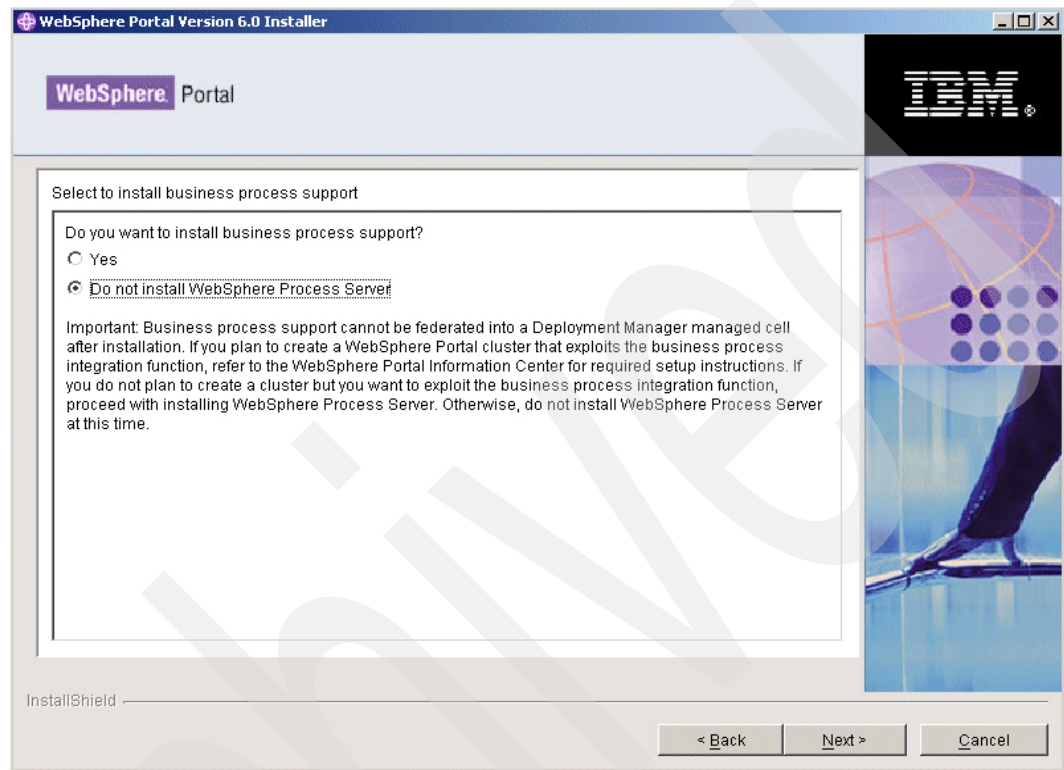


Figure 3-109 Do not install WebSphere Process Server

8. In the “Installation directory” window, type the directory where you want to install WebSphere Portal Server. If the directory that you specify does not exist, it will be created. For example:
D:\IBM\WebSphere\Portal Server
Click Next.
9. In the “Portal administrative user” window, enter the user ID and password for the WebSphere Portal Server administrator. Click Next as shown in Figure 3-110 on page 150.

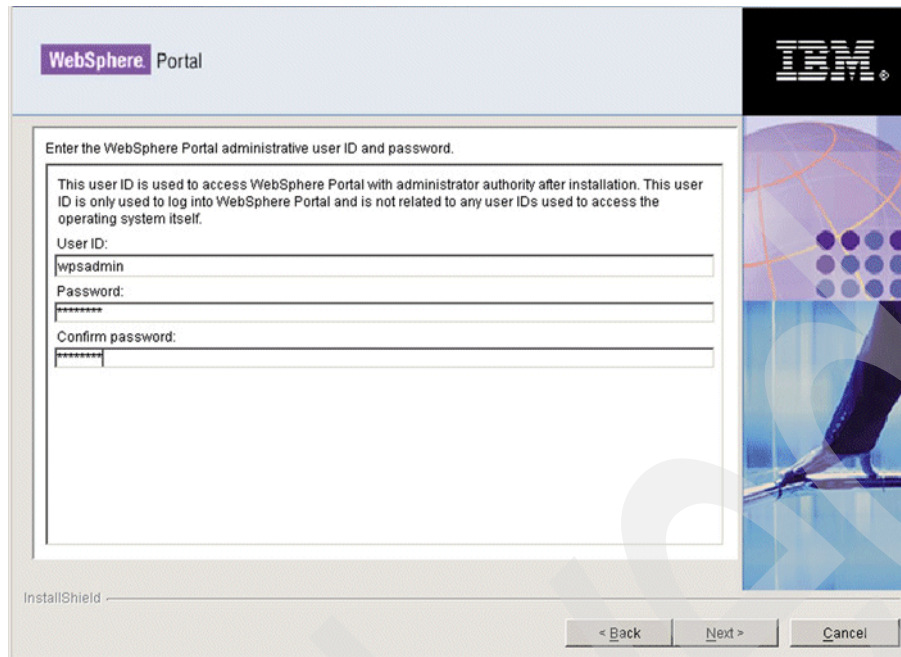


Figure 3-110 Select WebSphere Portal user ID and password window

10. In the “Windows Service Definition” window, you can configure Portal to run as a service. In our setup we do not want to run the WebSphere Portal Server as a Windows Service because of the longer start up time of the Windows system, so leave the defaults, and click Next as shown Figure 3-111.

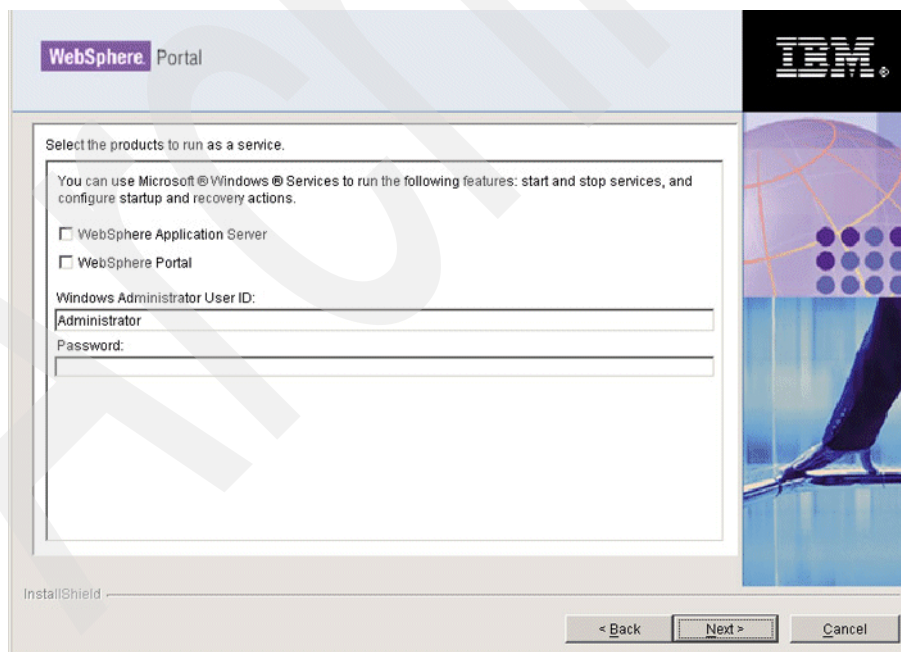


Figure 3-111 Windows service selection

11. In the “WebSphere Portal is ready to install” window, review the summary panel, and click Next to begin the install.

12. When Installation successfully completes, notice the information provided, such as port number. Click Finish.

13. Verify that WebSphere Portal is running by opening the following Web page in a browser:

http://example.com:port_number/wps/portal

where example.com is the fully qualified host name of the machine that is running WebSphere Portal and port_number is the port number that is displayed on the confirmation panel. For example,

<http://horizc3n1.cam.itso.ibm.com:10038/wps/portal>

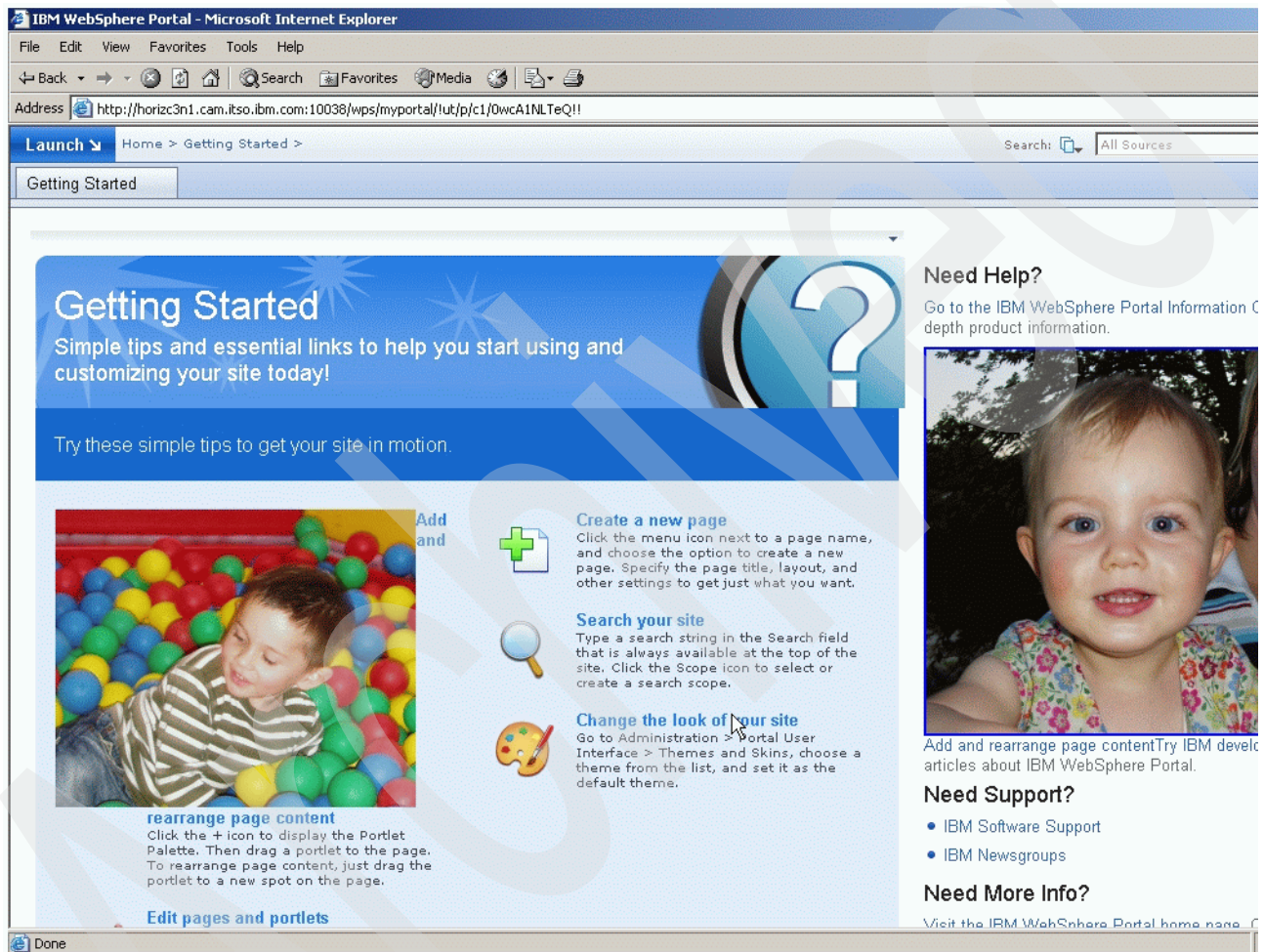


Figure 3-112 Portal window

3.5.3 Configure primary node 1 to an external database

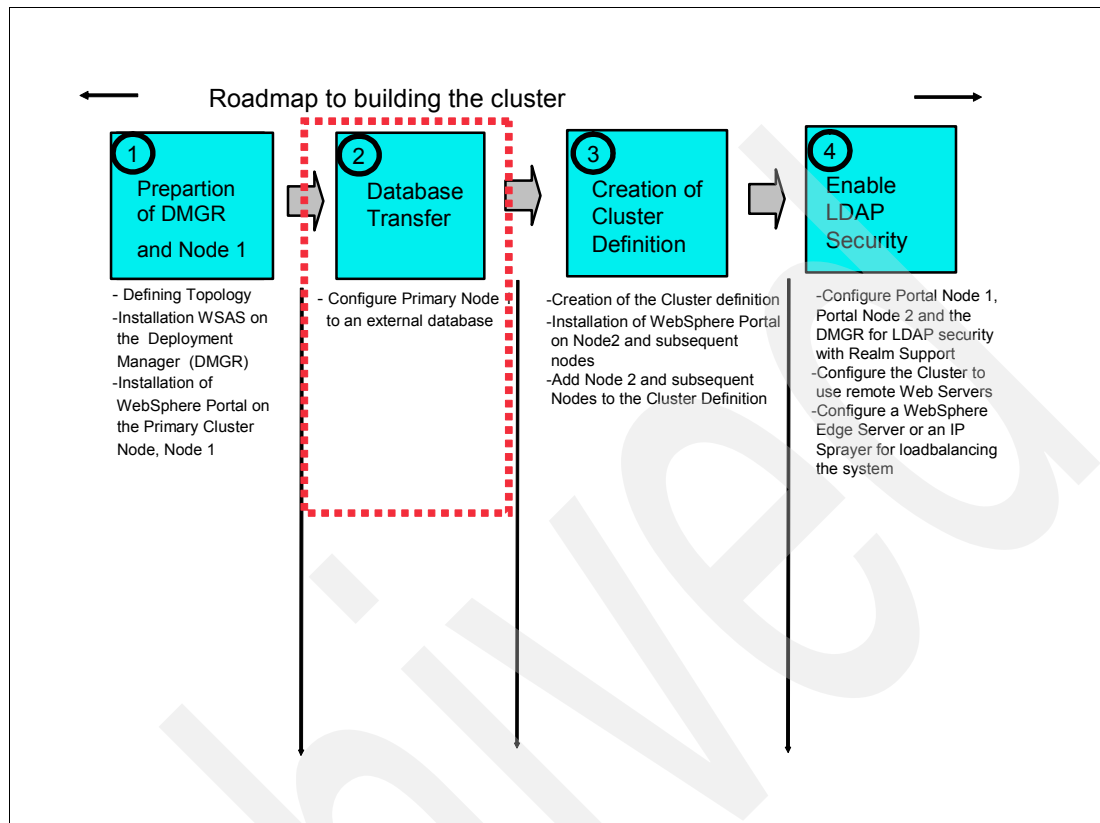


Figure 3-113 Flow chart about installation steps: step 2

Configure primary node 1 to an external database as described in section 3.4.5, "Configure primary node 1 to an external database" on page 93.

Following is a summary of the steps covered in section 3.4.5, "Configure primary node 1 to an external database" on page 93:

- Install DB2 Admin Client on primary node 1 and the DMGR.
- Create the DB that you want to use for WebSphere Portal.
- Catalog the DB on the primary node 1 and the DMGR.
- Configure the database using the configuration wizard.
- Create a new JDBC Resource and a WebSphere Variable for the DMGR.

3.5.4 Create cluster definition

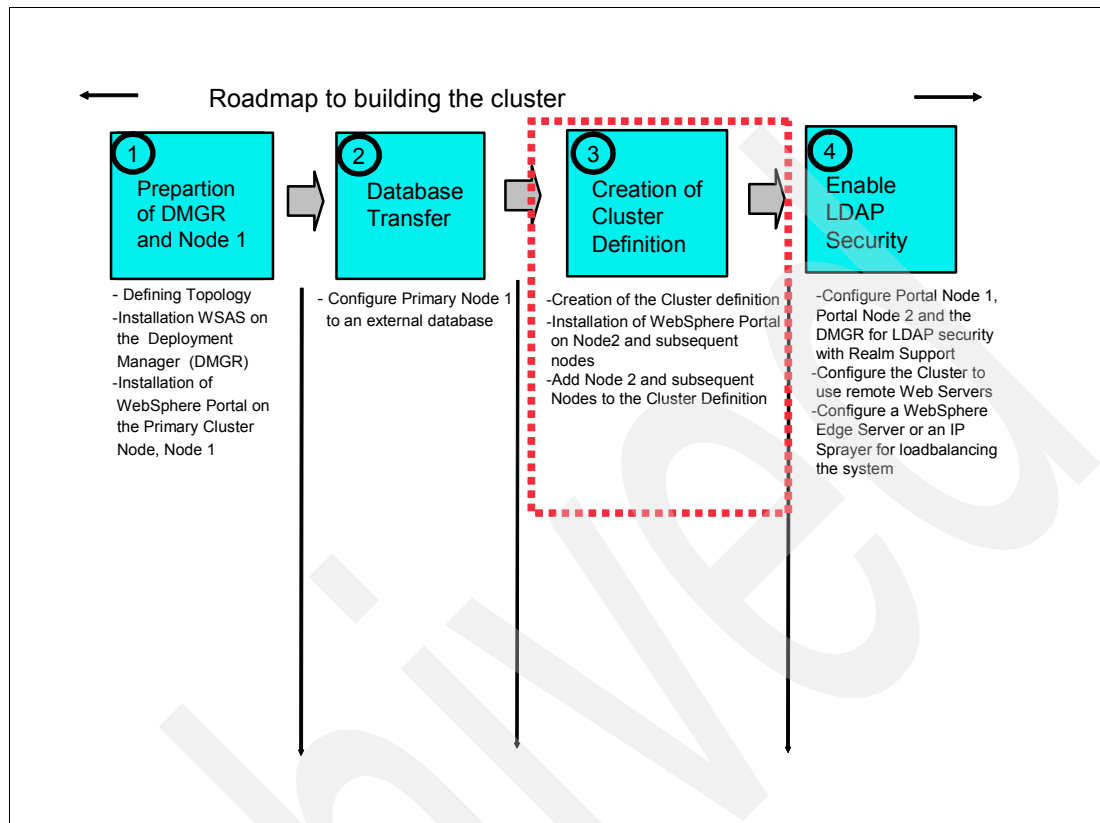


Figure 3-114 Flow chart of installation steps: step 3

The following steps will result on node 1 federation into the cluster and connectivity with Deployment Manager.

1. Increase the ConnectionTimeout value for the WebSphere_Portal application server and the DMGR. On some machine configurations, WebSphere Portal components might cause a timeout exception when the node is being added to the WebSphere Application Server Network Deployment cell. Increasing the timeout value eliminates the risk of this happening.
 - a. Start the WebSphere Administrative Console on node 1:
Start → Programs → IBM WebSphere → Application Server Network Deployment V6 → Profiles → wp_profile → Administrative Console
 - b. Log into the administrative console for the WebSphere Portal node.
 - c. To change the timeout values for the WebSphere_Portal application server, choose Servers → Application Servers → WebSphere_Portal → Web container settings → Web container transport chains
 - d. Increase the timeout values. For each entry listed in the Web container transport chains section, complete the following steps to increase the timeout values:
 - i. Click HTTP Inbound Channel.
 - ii. Change the Read timeout value to 180.
 - iii. Change the Write timeout value to 180.
 - iv. Save the configuration changes.

- e. Start the WebSphere Administrative Console on DMGR:
Start → Programs → IBM WebSphere → Application Server Network Deployment V6 → Profiles → <ProfileName> → Administrative Console
 - f. Log into the administrative console for the Deployment Manager.
 - g. To change the timeout values for the Deployment Manager, choose System Administration → Deployment Manager → Web container transport chains.
 - h. Increase the timeout values for the DMGR as you did previously for the WebSphere Portal node. You might find that there are fewer entries listed on the Deployment Manager than on the application server node. It is not necessary to add additional entries to the Deployment Manager configuration; instead, simply update those timeout values that are already listed.
2. Change the timeout request period for the Java Management Extensions (JMX) connector on the DMGR:
 - a. Click System administration → Deployment Manager → Administration Services → JMX connectors → SOAPConnector → Custom Properties.
 - b. Select the requestTimeout property, and increase the value from 600 to 6000.
 - c. Save your configuration changes.
 3. Restart the WebSphere_Portal application server and the Deployment Manager.
 4. Change the timeout request period for the Simple Object Access Protocol (SOAP) client. The default, in seconds, is 180. Edit the soap.client.props file in the was_profile_root/properties directory:
Change the line to:
`com.ibm.SOAP.requestTimeout=6000`
 5. Change to the following directory:
`portal_server_root/config`
 6. Use the following steps to copy Member Manager files to the Deployment Manager machine:
 - a. Create the wasextarchive.jar file, which contains the Member Manager binaries. Run the following command from the portal_server_root/config directory on the WebSphere Portal node:
`WPSconfig.bat archive-was-ext`
 - b. Copy the wasextarchive.jar file to the installation root folder of the DMGR machine.
`IBM/WebSphere/AppServer as <wsas_root>`
The wasextarchive.jar file is located in the following directory on the primary node:
`portal_server_root/config/work`
 7. Stop the DMGR by issuing the following command from the <wsas_root>/bin directory on the DMGR machine:
`IBM/WebSphere/AppServer/bin/stopManager.bat`
 - c. Extract the contents of the wasextarchive.jar file to the app_server_root directory on the DMGR machine.
Run the following command from the <wsas_root> directory:
`java\bin\jar -xvf wasextarchive.jar`
 - d. Verify that the <wsas_root>/lib directory contains files that start with wmm*.

- e. Restart the Deployment Manager by issuing the following command from the <was_root>/bin directory:
 IBM/WebSphere/AppServer/bin/startManager.bat
8. Ensure that the database software required for the Member Manager domain is installed.
 - a. Log into the Deployment Manager administrative console, and choose Resources → JDBC Providers.
 - b. Click the JDBC provider that contains the Member Manager data source.
 - c. In the Classpath field, note the name of the environment variable specified. For example, \${DB2_JDBC_DRIVER_CLASSPATH}, as shown in Figure 3-115.

JDBC providers > wpdbJDBC_db2

JDBC providers are used by the installed applications to access data from databases.

Configuration

General Properties

* Scope
cells:dmc3Cell01

* Name
wpdbJDBC_db2

Description
XA DB2 Universal JDBC Driver-compliant Provider. Datasources created under this provider support the use of XA to perform 2-phase commit

Class path
{DB2_JDBC_DRIVER_CLASSPATH}

Native library path

* Implementation class name
COM.ibm.db2.jdbc.DB2XADataSource

Apply OK Reset Cancel

Additional Properties

- Data sources
- Data sources (Version 4)

Figure 3-115 JDBC provider

- d. Select Environment → WebSphere Variables in the navigation tree.
- e. Select the Deployment Manager node to filter the list of variables as shown in Figure 3-116.

WebSphere Variables

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

Scope: Cell=dmc3Cell01, Node=dmc3CellManager01

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](#).

Cell
dmc3Cell01

Node
dmc3CellManager01 Browse Nodes

Cluster
Browse Clusters

Server
Browse Servers

Figure 3-116 WebSphere variables scope

- f. Create a new WebSphere variable. To do this click New, and enter the following values as shown in Figure 3-117.
 - Type the name of the variable previously specified by the JDBC provider ({DB2_JDBC_DRIVER_CLASSPATH}).
 - In the Value field, enter the directory and name of the ZIP or JAR file that contains the JDBC driver class. For example: db2_install/java/db2java.zip.

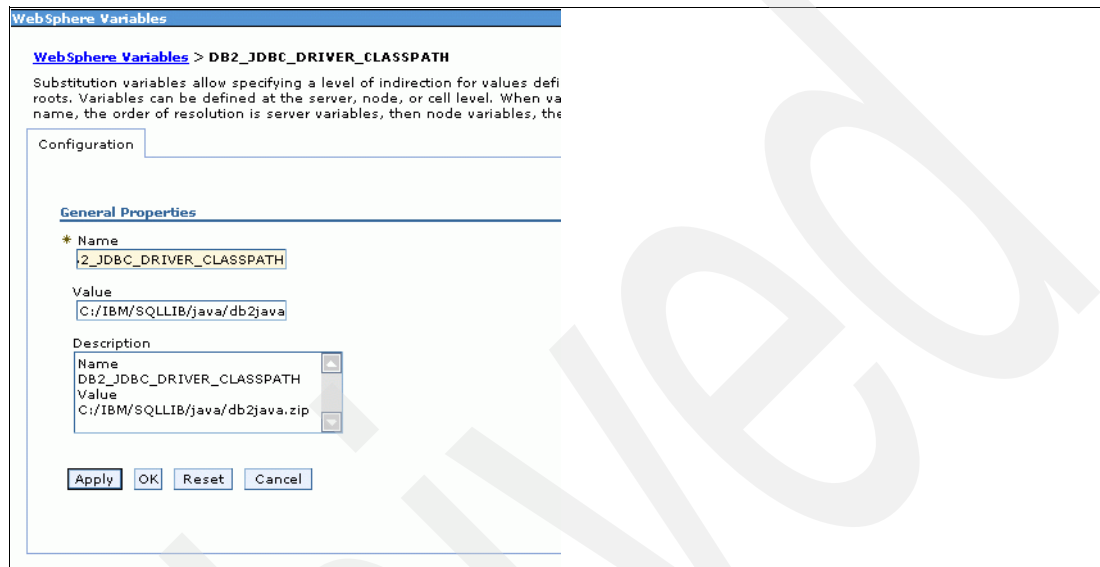


Figure 3-117 Classpath settings

- g. Save your changes to the Deployment Manager configuration.
9. Configure the Deployment Manager machine with the same security setting for your node by completing the following steps:
 - a. Use a text editor to open the wpconfig.properties file.
portal_server_root/config/wpconfig.properties
 - b. Ensure that the following properties are uncommented and specify appropriate values:
 - DMGrHostName property: Specify the host name of the machine where the Deployment Manager is running, for example:
dmc3.cam.itso.ibm.com
 - DMGrSoapPort property: Specify the port number where the Deployment Manager machine receives SOAP requests. The default port number is 8879.
 - c. Enable security on the Deployment Manager machine:
Run the following command from the portal_server_root/config directory:


```
WPSconfig.bat enable-security-wmmur-dmgr -DPortalAdminPwd=password -DWasPassword=password -DLTPAPassword=password
```
 - d. Restart the DMGR and WebSphere Portal.
 - e. To verify the security settings, try to log into the administrative console on the DMGR.
10. Edit the <wsas_profile_root>/bin/setupCmdLine.bat script to increase the Java heap size. Increasing the Java heap size enables the addNode and removeNode commands to add or remove large applications without failing. Edit the file, and add the following line (on one line) to the bottom of the script:

```
SET JVM_EXTRA_CMD_ARGS=  
-Djava.security.properties=$WAS_HOME/java/jre/lib/security/java.security -Xms256m  
-Xmx1024
```

11. Increase the heap for the Deployment Manager process:

- a. Open the administrative console.
- b. Open System Administration → Deployment Manager → Java and Process Management → Process Definition → Java Virtual Machine.
- c. Specify **256** for **Initial Heap Size** and **1024** for **Maximum Heap Size**.
- d. Save changes, and restart DMGR.

12. Validate whether this installation of WebSphere Portal supports federation into a managed cell. Run the pre-node-federation task.

Run the following command from the portal_server_root/config directory:

```
WPSconfig.bat pre-node-federation
```

If the task succeeds, continue to the next step. However, if the task fails, this indicates that the portal was installed with the business process integration feature provided by WebSphere Process Server. This installed configuration does not support federation into a managed cell.

- If you require business process support and need to federate the portal server, you must reinstall WebSphere Portal according to the instructions for installing into a managed node, these steps are described in “Horizontal cluster (includes WebSphere Process Server) installation and configuration steps” on page 54.
- If you need to federate the portal server but do not require business process support, you can install WebSphere Portal on an unmanaged node and choose not to install business process support during installation. This installed configuration allows for subsequent federation into a managed cell.

13. Add the node to the DMGR cell by entering the addNode command. Ensure that you include the -includeapps parameter.

Before running the *addNode.bat* command ensure that the DMGR was started.

To add a node to the DMGR cell, issue the addNode.bat command (on one line) on the command line of the node to be added:

```
<wsas_root>\bin\addNode.bat <deployment_manager_host>  
<deployment_manager_port> -username <admin_user_id> -password  
<admin_password> -includeapps >addnode.log
```

where:

deployment_manager_host is the Deployment Manager host name.

deployment_manager_port is the Deployment Manager SOAP connector-address. The default value is 8879.

admin_user_id is the WebSphere Application Server administrative user name. This parameter is optional but is required if security is enabled.

admin_password is the administrative user password. This parameter is optional but is required if security is enabled.

Example:

```
addNode.bat dmc3.cam.itso.ibm.com 8879 -username wpsadmin -password password  
-includeapps >addnode.log
```

Note: To run the addNode command here you **MUST** supply username and password because security was enabled on the DMGR.

See the appropriate Network Deployment Information Center for details on the addNode command.

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/welcome_nd.html

Note: Ensure that the clocks are synchronized to within five minutes of each other on the node 1 machine and the DMGR machine. If the clocks are not within five minutes, the addNode process will fail.

14. Update the Deployment Manager configuration for the new WebSphere Portal.

Note: The post-portal-node-federation-configuration task requires complete and accurate database information in the wpconfig_dbdomain.properties file. Before running the task, ensure that the database properties are correct and that password values are specified.

- a. Run the post-portal-node-federation-configuration task.

Run the following command from the portal_server_root/config directory:

```
WPSconfig.bat post-portal-node-federation-configuration  
-DWasPassword=password > postportal federation.log
```

- b. Check the log file if the task ended successful.
- c. Restart the node agent on the WebSphere Portal node.
 - i. Open the administrative console on the DMGR, and choose System Administration → Node Agents.
 - ii. Select the check box beside the node agent that you want to restart, and click Restart as shown in Figure 3-118.

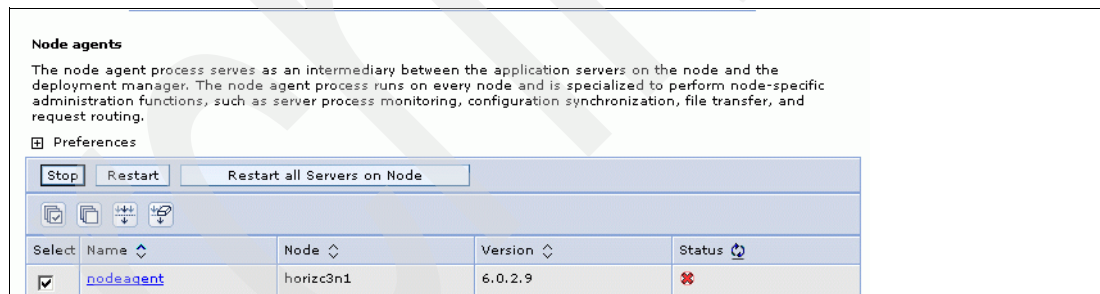


Figure 3-118 Node agents restart

- d. Restart the Deployment Manager.

- Stop the DMGR.

IBM/WebSphere/AppServer/bin/stopManager.bat

- Start the DMGR.

IBM/WebSphere/AppServer/bin/startManager.bat

15. After the primary node is installed and running as a managed node under Deployment Manager control, create the cluster.

- a. Edit the wpconfig.properties in folder portal_server_root/config/ to match the settings for the new WebSphere Portal cluster definition as shown in Table 3-6 on page 159.

Table 3-6 Cluster properties in wpconfig.properties

Field	Example value	Comments
WasPassword	wpsadmin	You must add the PortalAdminPwd and the WasPassword values to the wpconfig.properties file and all the database password values to the wpconfig_dbdomain.properties file, or supply these values on the command line. This is because the configuration wizard replaces all the password values with the string, "ReplaceWithYourPassword" for security reasons.
PortalAdminPwd	wpsadmin	-----
PrimaryNode	true	Verify that the value for this node is true.
Clustername	PortalCluster	Specify the cluster name you want to use when creating the cluster. Do not use spaces or special characters in the cluster name.

Important: If you wish to change the name of the cluster to something other than the default in the wpconfig.properties file, you **MUST** change this now **BEFORE** the cluster definition is created. This can be changed by editing the wpconfig.properties file and changing the ClusterName property.

- b. Check that DMGR and the node are started with the following command:
D:\IBM\WebSphere\AppServer\bin\serverstatus -all -username <wasadmin>
-password <waspwd>
- c. Start WebSphere_Portal, when not already started, with the following command:
D:\IBM\WebSphere\AppServer\bin\startserver WebSphere_Portal

Important: The cluster-setup task requires complete and accurate database information in the wpconfig_dbdomain.properties and wpconfig_dbtype.properties files. Before running the task, ensure that the database properties are correct and that password values are specified.

- d. Create the cluster definition by running the following command:
<wp_root>/config/WPSconfig.bat cluster-setup >cluster-setup.log
Check the log file if the build was successful.

16.Restart DMGR, nodeagent, and WebSphere_Portal to load the new configuration.

17.To check if the cluster definition and creation were successful, log into the Admin console of the DMGR and browse to Server → Clusters.

An entry with the “ClusterName” should be available as shown in Figure 3-119.

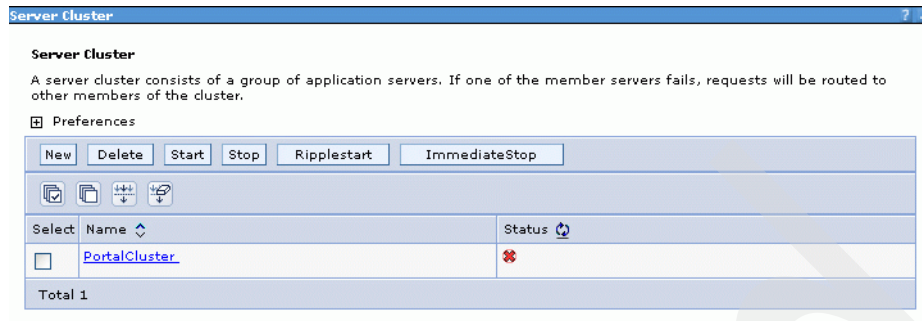


Figure 3-119 Check cluster definition

18. As a checkpoint in the process, you now have a 1 node cluster configured to an external database and using the WMM database for security.

3.5.5 Install WebSphere Portal on node 2 and subsequent nodes

Install WebSphere Portal on node 2 and subsequent nodes as described in section 3.3.2 3.5.2, “Install WebSphere Portal on the primary cluster node, node 1” on page 146.

Important: If you already have a WebSphere Portal Cluster up and running and you want to add an additional node to the cluster, proceed with the steps in section 3.7, “Add a Horizontal node to a cluster without WebSphere Process Server” on page 172.

Note: Use the same User ID and password for the WebSphere Application Server and the WebSphere Portal Server as you did for primary node 1.

3.5.6 Add node 2 and subsequent nodes to the cluster definition

The following steps result on node 2 federation into the cluster, connectivity with Deployment Manager, and connectivity with the database.

1. Install the DB2 Admin Client on node 2 and subsequent nodes.
2. Open a DB2 command line processor on the node 2 (Start → Programs → IBM DB2 → Command Line Tools → Command Line Processor).

a. Enter the following command:

```
db2 => catalog tcpip node nodename remote fullyqualifiedDB2servername server port
```

Where *nodename* is the name of eight characters or fewer that you assign to the server, and *port* is the connection port used by DB2.

For example:

```
db2 => catalog tcpip node dbdom7 remote dbdom7.cam.itso.ibm.com server 50000
```

In this example, dbdom7 is designated as the node name (to be used in the rest of the commands) and 50000 is the connection port (the default for TCP/IP connections). The

client always uses the same connection port as the DB2 server. You can verify the connection port for the DB2 server by opening a command prompt and issuing the **netstat -an** command. You should see a port in the 50000 range.

- b. Enter the following command (as appropriate) to catalog the databases that are hosted on the DB2 server.

```
db2 => catalog database databasename at node nodename
```

For example:

```
db2 => catalog database wpsdb at node dbdom7
```

Note: If you incorrectly catalog a database, you can uncatalog it by entering the following command:

```
db2=>uncatalog database database_name
```

- c. After you verify that all of the databases were cataloged correctly, test your connectivity to the database by entering the following command:

```
db2 => connect to databasename user DB2administratorname using password
```

For example:

```
db2 => connect to wpsdb user db2admin using password
```

3. Update the CLI Driver settings on the node 2.

- a. Locate the following file:

Windows: db2home/sql1lib/db2cli.ini

- b. Edit the file by adding the following to the end of the file:

* For Fix Pack 11:

[COMMON]

DYNAMIC=1

Note: An empty line is required after the dynamic=1 at the end of the file.

* For Fix Pack 12:

[COMMON]

ReturnAliases=0

Note: An empty line is required after the ReturnAliases=0 at the end of the file.

4. Make a backup of the original wpconfig_dbdomain.properties and wpconfig_dbtype.properties files on Node2 and subsequent nodes.
5. Copy the wpconfig_dbdomain.properties and wpconfig_dbtype.properties from Node1 to Node2 to ensure the same database configuration. Check if the values meet your node 2, especially in the wpconifg_dbtype.properties file value in field db2.DbLibrary.
6. Test the database connection by running the validate database tasks. If the passwords are defined in the wpconfig_dbdomain.properties file, the following -D options are not required at the command line.

```
WPSconfig.bat validate-database-driver
```

```
WPSconfig.bat validate-database-connection-wps -DDbPassword=password
```

```
WPSconfig.bat validate-database-connection-jcr -DJcrDbPassword=password
```

```
WPSconfig.bat validate-database-connection-feedback
```

```
-DFeedbackDbPassword=password
```

```
WPSconfig.bat validate-database-connection-likeminds
```

```
-DLikemindsDbPassword=password
```

```
WPSconfig.bat validate-database-connection-wmm -DWmmDbPassword=password
```

7. Change the timeout request period for the Simple Object Access Protocol (SOAP) client. The default, in seconds, is 180. Edit the `soap.client.props` file in the `was_profile_root/properties` directory:

Change the line to:

```
com.ibm.SOAP.requestTimeout=6000
```

8. Validate whether this installation of WebSphere Portal supports federation into a managed cell. Run the `pre-node-federation` task.

Run the following command from the `portal_server_root/config` directory:

```
WPSconfig.bat pre-node-federation
```

If the task succeeds, continue to the next step. However, if the task fails, this indicates that the portal was installed with the business process integration feature provided by WebSphere Process Server. This installed configuration does not support federation into a managed cell.

- If you require business process support and need to federate the portal server, you must reinstall WebSphere Portal according to the instructions for installing into a managed node, these steps are described in “Horizontal cluster (includes WebSphere Process Server) installation and configuration steps” on page 54.
- If you need to federate the portal server but do not require business process support, you can install WebSphere Portal on an unmanaged node and choose not to install business process support during installation. This installed configuration allows for subsequent federation into a managed cell.

9. Add the portal application server on the secondary node to the DMGR cell by entering the `addNode` command. Do not include the `-includeapps` parameter.

Before running the `addNode.bat` command ensure that the DMGR was started.

To add a node to the DMGR cell, issue the `addNode.bat` command (on one line) on the command line of the node to be added:

```
<wsas_root>\bin\addNode.bat <deployment_manager_host>  
<deployment_manager_port> -username <admin_user_id> -password  
<admin_password> >addnode.log
```

where the following applies:

`deployment_manager_host` is the Deployment Manager host name.

`deployment_manager_port` is the Deployment Manager SOAP connector-address. The default value is 8879.

`admin_user_id` is the WebSphere Application Server administrative user name. This parameter is optional but is required if security is enabled.

`admin_password` is the administrative user password. This parameter is optional but is required if security is enabled.

Example:

```
addNode.bat dmc3.cam.itso.ibm.com 8879 -username wpsadmin -password password  
>addnode.log
```

Note: To run the `addNode` command here you MUST supply username and password because security was enabled on the DMGR.

See the appropriate Network Deployment Information Center for details on the `addNode` command.

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/welcome_nd.html

Note: Ensure that the clocks are synchronized to within five minutes of each other on node 2 machine and the DMGR machine. If the clocks are not within five minutes, the addNode process will fail.

It is important not to use the -includeapps option. The applications will not be transferred as they are already in the Deployment Manager's master configuration. Applications stored in the master configuration are still only assigned to the primary node. In order to share them with a secondary node, the node must be added to the cluster.

10. Update the PrimaryNode property value on the secondary node to indicate its status. Use a text editor to open the wpconfig.properties file.

portal_server_root/config/wpconfig.properties

Ensure that the following properties are uncommented, and specify appropriate values:

PrimaryNode property: Because this node is a secondary node, set this property to false.

11. Update the Deployment Manager configuration for the new WebSphere Portal.

Note: The post-portal-node-federation-configuration task requires complete and accurate database information in the wpconfig_dbdomain.properties file. Before running the task, ensure that the database properties are correct and that password values are specified.

- a. Run the post-portal-node-federation-configuration task.

Run the following command from the portal_server_root/config directory:

```
WPSconfig.bat post-portal-node-federation-configuration  
-DWasPassword=password > postportal federation.log
```

- b. Check the log file if the task ended successful. If yes, proceed with the next step.

- c. Restart the node agent on the WebSphere Portal node.

- i. Open the administrative console on the DMGR, and choose System Administration → Node Agents.

- d. Select the check box beside the node agent that you want to restart, and click Restart.

12. Update the PrimaryNode property value on the secondary node to indicate its status.

Edit the wpconfig.properties to match the Settings for the WebSphere Portal Cluster Definition as shown in Table 3-7 on page 164.

Table 3-7 Cluster properties in wpconfig.properties

Field	Example value	Comments
WasPassword	wpsadmin	You must add the PortalAdminPwd and the WasPassword values to the wpconfig.properties file and all the database password values to the wpconfig_dbdomain.properties file or supply these values on the command line. This is because the configuration wizard replaces all the password values with the string, "ReplaceWithYourPassword" for security reason.s
PortalAdminPwd	wpsadmin	-----
PrimaryNode	false	Verify that the value for this node is false, because it is not the primary node.
Clustername	PortalCluster	ClusterName should be the name of the cluster created when running the cluster-setup task on the primary node, node 1.
ServerName	WebSphere_Portal_2	ServerName is REQUIRED to be changed from WebSphere_Portal. The cluster-setup task is written to automatically remove the WebSphere_Portal server during the action-remove-appserver-wps task.

Important: The cluster member name you specify for this property must be unique within the cell and cannot have the same value as the ServerName property on the primary node or other secondary nodes.

- e. Check that DMGR and the node are started with the following command:

```
D:\IBM\WebSphere\AppServer\bin\serverstatus -all -username <wasadmin>
-password <waspwd>
```
- f. Start WebSphere_Portal, when not already started, with the following command:

```
D:\IBM\WebSphere\AppServer\bin\startserver WebSphere_Portal
```

Important: In Portal 5.1.x, you ran the connect-database task to connect node 2 to the database. In Portal 6, the connect-database task is incorporated into the cluster-setup task.

The cluster-setup task requires complete and accurate database information in the wpconfig_dbdomain.properties and wpconfig_dbtype.properties files. Before running the task, ensure that the database properties are correct and that password values are specified.

- g. Create the cluster definition by running the following command:

```
<wp_root>/config/WPSconfig.bat cluster-setup >cluster-setup.log
```

Check the log file if the build was successful.

13. Restart DMGR, nodeagent, and WebSphere_Portal_2 to load the new configuration.

14. To check if the second node is a member of the cluster definition, log into the Admin console of the DMGR and browse to Server → Clusters → ClusterMembers.

An entry with the “WebSphere_Portal_2” should be available as shown in Figure 3-120.

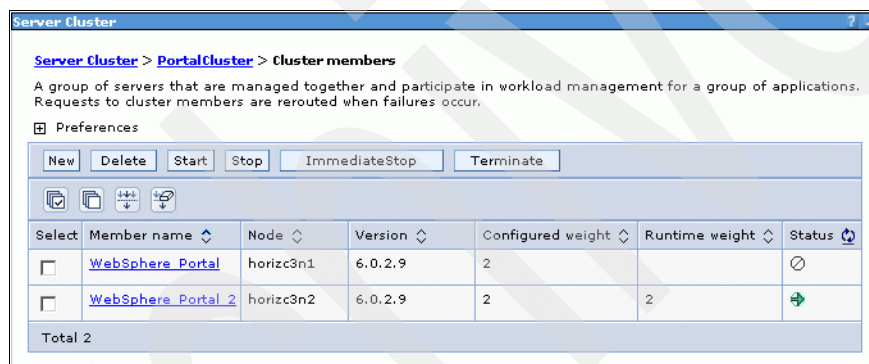


Figure 3-120 Check cluster definition

Note: Because WebSphere Application Server typically creates unique ports for all servers, the default port number for the WebSphere Portal application server is changed when the node is added to the cluster. To determine the port number after creating the new cluster member, complete the following steps in the DMGR administrative console.

- i. In the Navigation pane, click Servers → Application Servers.
- ii. Click the application server name for the secondary node.
- iii. Click Ports under the Communications settings, and verify the port number listed for the WC_defaulthost port.

3.5.7 Configure Portal nodes and the DMGR for LDAP security with Realm Support

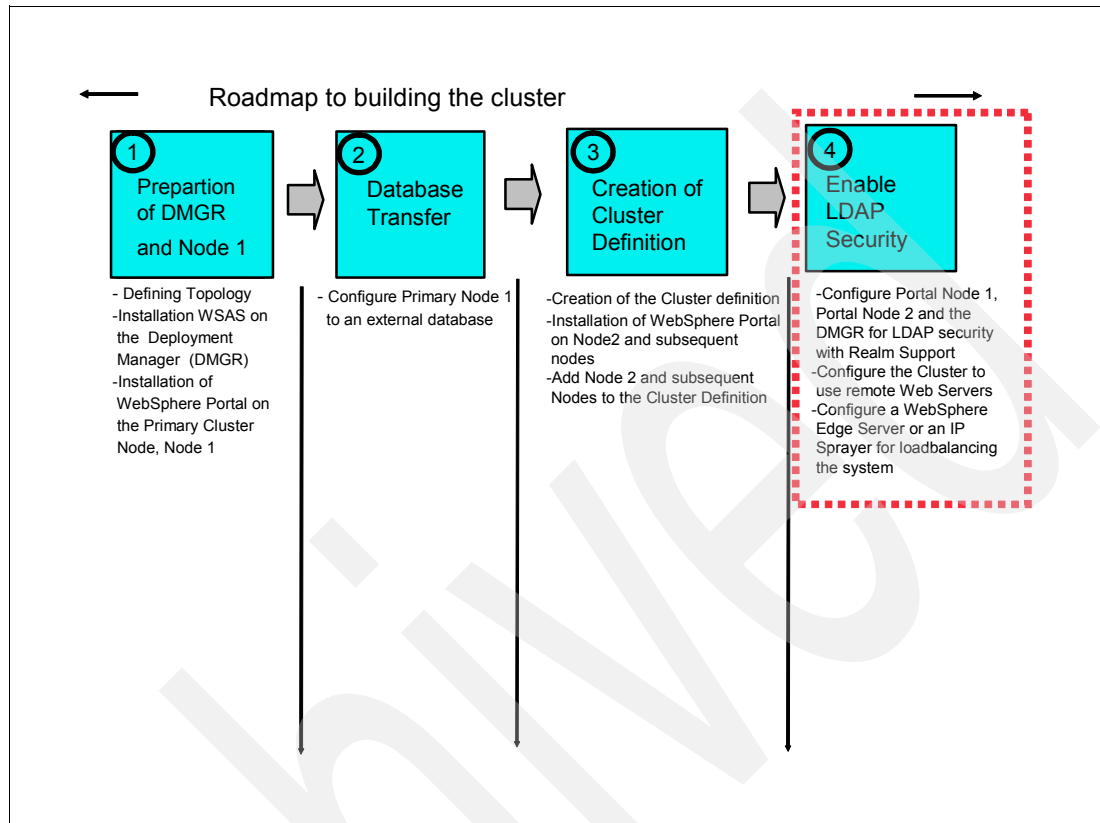


Figure 3-121 Flow chart about installations steps: step 4

Configure security as described in section 3.4.10, "Configure Portal nodes and the DMGR for LDAP security with Realm Support" on page 122.

Following is a summary of the steps covered in the section 3.4.10, "Configure Portal nodes and the DMGR for LDAP security with Realm Support" on page 122:

- Disable security on the primary node.
- Enable security on the primary node.
- Enable JCR Security on the secondary node and subsequent nodes.

3.5.8 Configure Portal to use remote Web Servers

Configure a remote Web Server as described in Section 3.4.11, "Configure Portal to use a remote Web server" on page 131.

Summary of the Steps that are covered in the Section 3.4.11, "Configure Portal to use a remote Web server" on page 131:

- Install Web servers.
- Install WebSphere Application Server plug-in on the Web servers.
- Add the Web servers to the Deployment Manager cell.
- Generate the WebSphere Plugin.

Note: At this point the cluster setup is finished. If you want to add an additional node to the cluster, proceed with the steps in section 3.6, “Adding a Horizontal node to a cluster with WebSphere Process Server” on page 168.

3.5.9 Optional steps after cluster creation

Following are optional steps you can take after creating the cluster.

- ▶ Configure a WebSphere Edge Server or an IP Sprayer for loadbalancing the system (This is not described in this book.).
- ▶ Configure the WebSphere Portal Cluster to use a second LDAP Directory as described in Chapter 4, “Multiple LDAP directory support” on page 187.
- ▶ Configure the WebSphere Portal Cluster to share the Database with a second clustered environment as described in Chapter 5, “Database domains” on page 233.

3.6 Adding a Horizontal node to a cluster with WebSphere Process Server

This section provides instructions for adding an additional node to an already defined cluster:

1. Install db2 client on the node.

2. Open a DB2 command line processor on the node:

(Start → Programs → IBM DB2 → Command Line Tools → Command Line Processor).

a. Enter the following command:

```
db2 => catalog tcpip node nodename remote fullyqualifiedDB2servername server port
```

Where *nodename* is the name of eight characters or fewer that you assign to the server, and *port* is the connection port used by DB2.

For example:

```
db2 => catalog tcpip node dbdom7 remote dbdom7.cam.itso.ibm.com server 50000
```

In this example, *dbdom7* is designated as the node name (to be used in the rest of the commands) and 50000 is the connection port (the default for TCP/IP connections). The client always uses the same connection port as the DB2 server. You can verify the connection port for the DB2 server by opening a command prompt and issuing the **netstat -an** command. You should see a port in the 50000 range.

b. Enter the following command (as appropriate) to catalog the databases that are hosted on the DB2 server.

```
db2 => catalog database databasename at node nodename
```

For example:

```
db2 => catalog database wpsdb at node dbdom7
```

Note: If you incorrectly catalog a database, you can uncatalog it by entering the following command:

```
db2=>uncatalog database database_name
```

c. After you verify that all of the databases were cataloged correctly, test your connectivity to the database by entering the following command:

```
db2 => connect to databasename user DB2administratorname using password
```

For example:

```
db2 => connect to wpsdb user db2admin using password
```

3. Update the CLI Driver settings on the additional Nodes.

a. Locate the following file:

Windows: *db2home/sql/lib/db2cli.ini*

b. Edit the file by adding the following to the end of the file:

* For Fix Pack 11:

[COMMON]

DYNAMIC=1

Note: An empty line is required after the *dynamic=1* at the end of the file.

* For Fix Pack 12:

[COMMON]

ReturnAliases=0

Note: An empty line is required after the ReturnAliases=0 at the end of the file.

4. Install WSAS and WPS using the process described in section 3.4.7, “Install WSAS and WPS on future cluster node, node 2, and subsequent nodes” on page 109.
5. Create the Process Server profile as described in section “Install WSAS and WPS on future cluster node, node 2, and subsequent nodes”, step 6 on page 112.
6. After the profile is created, we will federate the node using the addNode command as described in section “Install WSAS and WPS on future cluster node, node 2, and subsequent nodes”, step 14 on page 116.
7. Install the necessary fixes for WebSphere Portal as described in section “Install WSAS and WPS on future cluster node, node 2, and subsequent nodes”, step 15 on page 116.
8. Install WebSphere Portal using the process described in section 3.4.8, “Install Portal on the managed node, node 2, and subsequent nodes” on page 117.
9. Stop WebSphere_Portal server on the new node, and ensure that the nodeagent is running.

10. Update the <wp_root>/config/wpconfig.properties file on each additional node in the cluster with the same LDAP user registry information you used to configure the primary node.

Update the wpconfig.properties by moving the LDAP helper file from Node1 to additional Nodes and running the following command:

```
<wp_root>/config/WPSconfig -DparentProperties="<full_path_to_helper_file>"  
-DSaveParentProperties=true
```

11. Complete the security configuration by running the enable-jcr-security configuration task on each secondary node.

Run the following command from the <wp_root>/config directory:

```
WPSconfig.bat enable-jcr-security -DPortalAdminId=<portal_admin_id> >enable.log
```

where portal_admin_id is the fully qualified distinguished name (DN) of the portal administrator (for example, uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com).

12. Restart the nodeagent on the additional nodes.
13. Edit the wpconfig.properties to match the Settings for the WebSphere Portal Cluster Definition as shown in Table 3-8 on page 170.

Table 3-8 Cluster properties in wpconfig.properties

Field	Example value	Comments
WasPassword	wpsadmin	You must add the PortalAdminPwd and the WasPassword values to the wpconfig.properties file and all the database password values to the wpconfig_dbdomain.properties file or supply these values on the command line. This is because the configuration wizard replaces all the password values with the string, "ReplaceWithYourPassword" for security reasons.
PortalAdminPwd	wpsadmin	-----
PrimaryNode	false	Verify that the value for this node is false because it is not the primary node.
Clustername	PortalCluster	ClusterName should be the name of the cluster created when running the cluster-setup task on the primary node, node1.
ServerName	WebSphere_Portal_3	ServerName is REQUIRED to be changed from WebSphere_Portal. The cluster-setup task is written to automatically remove the WebSphere_Portal server during the action-remove-appserver-wps task.

14. Make a backup of the original wpconfig_dbdomain.properties and wpconfig_dbtype.properties files on Node2 and subsequent nodes.
15. Copy the wpconfig_dbdomain.properties and wpconfig_dbtype.properties from Node1 to additional Nodes to ensure the same database configuration. Check if the values meet your node, especially in the wpconfig_dbtype.properties file value in field db2.DbLibrary.
16. Again, in Portal V6 the connect-database task was integrated into the cluster-setup task; therefore, we must run the validate database tasks. If the passwords are defined in the wpconfig_dbdomain.properties file, the following -D options are not required at the command line.

```

WPSconfig.bat validate-database-driver
WPSconfig.bat validate-database-connection-wps -DDbPassword=password
WPSconfig.bat validate-database-connection-jcr -DJcrDbPassword=password
WPSconfig.bat validate-database-connection-feedback
-DFeedbackDbPassword=password
WPSconfig.bat validate-database-connection-likeminds
-DLikemindsDbPassword=password

```

WPSconfig.bat validate-database-connection-wmm -DWmmDbPassword=password

17. Check that DMGR and the node are started with the following command:

```
D:\IBM\WebSphere\AppServer\bin\serverstatus -all -username <wasadmin>
-password <waspwd>
```

18. Start WebSphere_Portal, when not already started, with the following command:

```
D:\IBM\WebSphere\AppServer\bin\startserver WebSphere_Portal
```

19. Create the cluster definition by running the following command:

```
<wp_root>/config/WPSconfig.bat cluster-setup >cluster-setup.log
```

Check the log file if the build was successful.

20. Restart DMGR, nodeagent, and WebSphere_Portal to load the new configuration.

21. To check if the cluster definition and creation were successful, log into the Admin console of the DMGR and browse to Server → Clusters → PortalCluster → ClusterMember.

An entry for “WebSphere_Portal_3” should be available.

22. Because WebSphere Application Server typically creates unique ports for all servers, the default port number for the WebSphere Portal application server is changed when the node is added to the cluster. To determine the port number after creating the new cluster member, complete the following steps in the Deployment Manager administrative console.

- a. Click Servers → Application Servers in the navigation pane.
- b. Click the application server name for the secondary node.
- c. Click Ports under the Communications settings, and verify the port number listed for the WC_defaulthost port.

23. Regenerate the Web server plug-in as described in section 3.5.8, “Configure Portal to use remote Web Servers” on page 166.

- a. Regenerate the Web server plug-in using the DMGR administrative console.
- b. If you are using a remote Web server, copy the updated plug-in configuration file (plugin-cfg.xml) to the Web server's plug-in configuration directory.
- c. Stop and start the Web server.
- d. Restart all nodes in the cluster.

Now we successfully added an additional node to the cluster definition.

3.7 Add a Horizontal node to a cluster without WebSphere Process Server

This section provides instructions on adding an additional node to an already defined cluster:

1. Install db2 client on the node.

2. Open a DB2 command line processor on the node.

(Start → Programs → IBM DB2 → Command Line Tools → Command Line Processor).

- a. Enter the following command:

```
db2 => catalog tcpip node nodename remote fullyqualifiedDB2servername server port
```

Where *nodename* is the name of eight characters or fewer that you assign to the server, and *port* is the connection port used by DB2.

For example:

```
db2 => catalog tcpip node dbdom7 remote dbdom7.cam.itso.ibm.com server 50000
```

In this example, *dbdom7* is designated as the node name (to be used in the rest of the commands) and 50000 is the connection port (the default for TCP/IP connections). The client always uses the same connection port as the DB2 server. You can verify the connection port for the DB2 server by opening a command prompt and issuing the **netstat -an** command. You should see a port in the 50000 range.

- b. Enter the following command (as appropriate) to catalog the databases that are hosted on the DB2 server.

```
db2 => catalog database databasename at node nodename
```

For example:

```
db2 => catalog database wpsdb at node dbdom7
```

Note: If you incorrectly catalog a database, you can uncatalog it by entering the following command:

```
db2=>uncatalog database database_name
```

- c. After you verify that all of the databases were cataloged correctly, test your connectivity to the database by entering the following command:

```
db2 => connect to databasename user DB2administratorname using password
```

For example:

```
db2 => connect to wpsdb user db2admin using password
```

3. Update the CLI Driver settings on the additional Nodes.

- a. Locate the following file:

Windows: *db2home/sql/lib/db2cli.ini*

- b. Edit the file by adding the following to the end of the file:

* For Fix Pack 11:

[COMMON]

DYNAMIC=1

Note: An empty line is required after the *dynamic=1* at the end of the file.

* For Fix Pack 12:

[COMMON]

ReturnAliases=0

Note: An empty line is required after the ReturnAliases=0 at the end of the file.

4. Install WebSphere Portal using the process described in section 3.5.5, “Install WebSphere Portal on node 2 and subsequent nodes” on page 160.
5. Connect the additional node to the database as described in section 3.5.6, “Add node 2 and subsequent nodes to the cluster definition” on page 160.
6. Add the node to the Deployment Manager cell as described in section 3.5.7, “Configure Portal nodes and the DMGR for LDAP security with Realm Support” on page 166, steps 1-5.

7. Stop WebSphere_Portal server on the new node, and ensure that the nodeagent is running.
8. Update the <wp_root>/config/wpconfig.properties file on each additional node in the cluster with the same LDAP user registry information you used to configure the primary node.

Update the wpconfig.properties by moving the LDAP helper file from Node1 to additional Nodes and running the following command:

```
<wp_root>/config/WPSconfig -DparentProperties="<full_path_to_helper_file>"  
-DSaveParentProperties=true
```

9. Complete the security configuration by running the enable-jcr-security configuration task on each secondary node.

Run the following command from the <wp_root>/config directory:

```
WPSconfig.bat enable-jcr-security -DPortalAdminId=<portal_admin_id> >enable.log
```

where portal_admin_id is the fully qualified distinguished name (DN) of the portal administrator (for example, uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com).

10. Restart the nodeagent on the additional nodes.
11. Update the PrimaryNode property value on the additional node to indicate its status.

Edit the wpconfig.properties to match the Settings for the WebSphere Portal Cluster Definition as shown in Table 3-7 on page 164.

Table 3-9 Cluster properties in wpconfig.properties

Field	Example value	Comments
WasPassword	wpsadmin	You must add the PortalAdminPwd and the WasPassword values to the wpconfig.properties file and all the database password values to the wpconfig_dbdomain.properties file or supply these values on the command line. This is because the configuration wizard replaces all the password values with the string, “ReplaceWithYourPassword” for security reasons.
PortalAdminPwd	wpsadmin	-----

Field	Example value	Comments
PrimaryNode	false	Verify that the value for this node is false, because it is not the primary node.
Clustername	PortalCluster	ClusterName should be the name of the cluster created when running the cluster-setup task on the primary node, node 1.
ServerName	WebSphere_Portal_3	ServerName is REQUIRED to be changed from WebSphere_Portal. The cluster-setup task is written to automatically remove the WebSphere_Portal server during the action-remove-appserver-wps task.

Important: The cluster member name you specify for this property must be unique within the cell and cannot have the same value as the ServerName property on the primary node or other secondary nodes.

- c. Check that DMGR and the node are started with the following command:

```
D:\IBM\WebSphere\AppServer\bin\serverstatus -all -username <wasadmin>
-password <waspwd>
```
- d. Start WebSphere_Portal, when not already started, with the following command:

```
D:\IBM\WebSphere\AppServer\bin\startserver WebSphere_Portal
```

Important: The cluster-setup task requires complete and accurate database information in the wpconfig_dbdomain.properties and wpconfig_dbtype.properties files. Before running the task, ensure that the database properties are correct and that password values are specified.

- e. Create the cluster definition by running the following command:

```
<wp_root>/config/WPSconfig.bat cluster-setup >cluster-setup.log
```

Check the log file if the Build was successful.
12. Restart DMGR, nodeagent, and WebSphere_Portal_2 to load the new configuration.
13. To check if the second node is a member of the cluster definition, log in to the Admin console of the DMGR and browse to Server → Clusters → ClusterMembers.

An entry with the “WebSphere_Portal_3” should be available.
14. Because WebSphere Application Server typically creates unique ports for all servers, the default port number for the WebSphere Portal application server is changed when the node is added to the cluster. To determine the port number after creating the new cluster member, complete the following steps in the DMGR administrative console.
 - a. In the Navigation pane, click Servers → Application Servers.

- b. Click the application server name for the secondary node.
 - c. Click Ports under the Communications settings, and verify the port number listed for the WC_defaulthost port.
15. Regenerate the Web server plug-in as described in section 3.4.11, “Configure Portal to use a remote Web server” on page 131.
- a. Regenerate the Web server plug-in using the Deployment Manager administrative console.
 - b. If you are using a remote Web server, copy the updated plug-in configuration file (plugin-cfg.xml) to the Web server's plug-in configuration directory.
 - c. Stop and start the Web server.
 - d. Restart all nodes in the cluster.

We successfully added an additional node to the cluster definition.

3.8 Vertical cluster installation and configuration steps

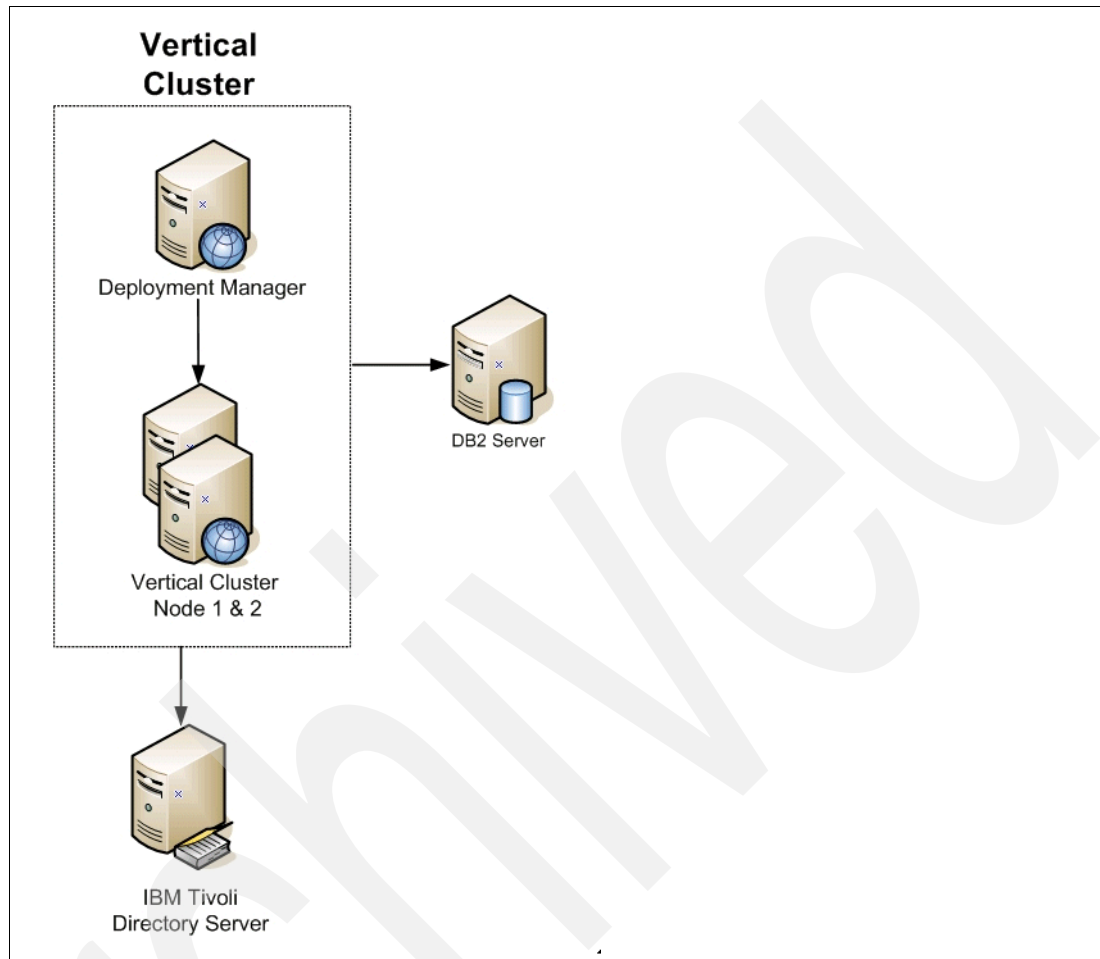


Figure 3-122 Vertical cluster deployment architecture

In our deployment, the following applies:

- ▶ DMC1: This machine served as the Deployment Manager. The Deployment Manager is the central administrative console for all nodes in the environment.
- ▶ VerC1N1 and VerC1N2: Nodes 1 and 2 hosted the business logic. After completion of the installation, both machines were clustered for load balancing and failover.
- ▶ DBDOM7: This served as the RDBMS server that hosted the IBM WebSphere Portal database for all nodes and the Deployment Manager.
- ▶ DBIDS represents our LDAP server (IBM Tivoli Directory Server V6), which hosted all directory information used by WebSphere Portal.

The process of creating a vertical cluster is virtually identical to the steps for creating a horizontal cluster. As with the horizontal cluster setup we do the following initial steps:

- ▶ “Install and upgrade WSAS and WPS on the Deployment Manager (DMGR)” on page 56.
- ▶ “Install WSAS and WPS on the primary cluster node, node1” on page 70.
- ▶ “Prepare the DMGR and node1 for the Portal install” on page 80.
- ▶ “Install Portal onto the managed node, node1” on page 84.

- ▶ “Configure primary node 1 to an external database” on page 93.
- ▶ “Create the cluster definition” on page 107.
- ▶ “Configure Portal nodes and the DMGR for LDAP security with Realm Support” on page 122.
- ▶ “Configure Portal to use a remote Web server” on page 131.
- ▶ “Optional steps after cluster creation” on page 135.

The only difference is the process for creating a vertical cluster member on the primary node which is described in the following section.

Adding a vertical cluster member on the primary node

Add vertical cluster members to share the workload demands of your cluster across multiple members running on the same physical machine.

1. Access the administrative console on the Deployment Manager by opening a browser and typing the following into the address bar:

`http://DM01:9060/ibm/console`

where DM01 is the DMGR node or host name. The port number might differ based on your installation.

2. Click Servers → Clusters in the console navigation tree, select the cluster name, and then click Cluster members from the list of additional properties.



Figure 3-123 Cluster members view

3. Click New to create the cluster member.

- a. Define the name of cluster member.

Note: Do not use spaces in the cluster member name.

Create new cluster members

Add application servers to a cluster.

Step 1: Enter basic cluster member information

Step 2: Summary

Enter basic cluster member information

Enter information about this new cluster member, and click Apply to add this cluster member to the member list. Use the Edit function to edit the properties of a cluster member that are already included in this list. Use the Delete function to remove a cluster member from this list.

* Member name
verc1n2

Select node
[verc1n1Node01(6.0.2.9)]

* Weight
2

☒ Generate Unique Http Ports

Apply

Edit Delete

Select	Application servers	Nodes	Version	Weight
<input checked="" type="checkbox"/>				

Next Cancel

Figure 3-124 Enter basic cluster member information

- Select an existing node where IBM WebSphere Portal is installed.
- Check the box Generate Unique HTTP Ports.
- Click Apply, and then click Next to view the summary.

Important: If you are not using an external Web server in your clustered environment, you must update the virtual host entries for the new port created when adding a cluster member. You can do this by updating the default_host virtual host in the administrative console and adding a new alias entry for the port number (use a "*" wildcard character for the host name). Refer to the WebSphere Application Server information center for more information.

Create new cluster members

Add application servers to a cluster.

Step 1: Enter basic cluster member information

Step 2: Summary

Summary

Summary of actions:

Server name = verc1n2
Node = verc1n1Node01(6.0.2.9)
Weight = 2
Core Group = DefaultCoreGroup
Clone Template = dmclCell01/verc1n1Node01/WebSpt
Generate Unique Http Ports = true
Node group = DefaultNodeGroup

Previous Finish Cancel

Figure 3-125 Summary window view

- Click Finish, and save the changes.
 - The new cluster topology can be viewed from the Servers → Cluster Topology view.
 - The Servers → Application Servers view lists the new server cluster members.

- Start the server.
- 5. Save your changes and resynchronize the nodes.
 - a. In the administrative console for the Deployment Manager, click Save on the task bar, and save your administrative configuration.
 - b. Select System Administration → Nodes, select the node from the list, and click Full Resynchronize.
- 6. Check the Port Number for the Vertical cluster member, and create a host alias for it:
 - a. Navigate to Servers → Application Servers → “VerticalClusterMember” → Ports. Check the Port Number for WC_defaulthost.
 - b. Create a host alias for the port that is used by the Vertical cluster member that you checked in the previous step. In our environment it is 9081.
 - c. Navigate to Environment → Virtual Hosts → Default Host → Host Aliases, and click New.
 - d. In the Hostname field provide a (*), and in the Port field provide the Port that is used by the Vertical cluster member as shown in Figure 3-126.

Virtual Hosts

Messages

Changes have been made to your local configuration. Click [Save](#) to apply changes to the master configuration.

The server may need to be restarted for these changes to take effect.

[Virtual Hosts](#) > [default_host](#) > [Host Aliases](#) > *

An alias is the DNS host and port number that a client uses to form the URL request of a Web Application resource. Application resources include servlets and JSP pages. An example of a URL request is `http://myhost:8080/servlet/snoop`. When no port number is specified, the default port 80 is used.

Configuration

General Properties

* Host Name
*

* Port
9081

Apply OK Reset Cancel

Figure 3-126 Host aliases view

- e. The host aliases should now look like Figure 3-127 on page 180.

Virtual Hosts		
Virtual Hosts > default_host > Host Aliases		
A list of one or more DNS aliases by which the virtual host is known.		
Preferences		
New Delete		
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>		
Select	Host Name	Port
<input type="checkbox"/>	*	9080
<input type="checkbox"/>	*	80
<input type="checkbox"/>	*	9443
<input type="checkbox"/>	*	9443
<input type="checkbox"/>	*	9081
<input type="checkbox"/>	verc1n1.cam.itso.ibm.com	9080
<input type="checkbox"/>	verc1n1.cam.itso.ibm.com	80
Total 7		

Figure 3-127 Host aliases

7. Log into the administrative console for WebSphere Application Server. If you are using the Web server in a clustered environment, open the administrative console for the Deployment Manager machine.
 - a. Select Environment → WebSphere Variables.
 - b. Narrow the scope of the listed variables:
 - i. Click Browse Nodes in the Scope field.
 - ii. Select the node containing your application server.
 - iii. Click Browse Servers.
 - iv. Select the application server for WebSphere Portal (for example, WebSphere_Portal), and click OK.
 - v. Click Apply to update the list of variables.
 - vi. Update the value of the WCM_HOST variable with the fully qualified host name of the Web server.
 - vii. Update the value of the WCM_PORT variable with the port number used to access the Web server.
8. Save your changes and resynchronize the nodes.
 - a. In the administrative console for the DMGR, click Save on the task bar, and save your administrative configuration.
9. Select System Administration → Nodes, select the node from the list, and click Full Resynchronize.
10. Restart the portal server.
11. Regenerate the Web server plug-in.
 - a. Regenerate the Web server plug-in using the Deployment Manager administrative console.
 - b. If you are using a remote Web server, copy the updated plug-in configuration file (plugin-cfg.xml) to the Web server's plug-in configuration directory.
12. Stop and start the Web server.

3.9 Vertical cluster (without WebSphere Process Server) install and configure steps

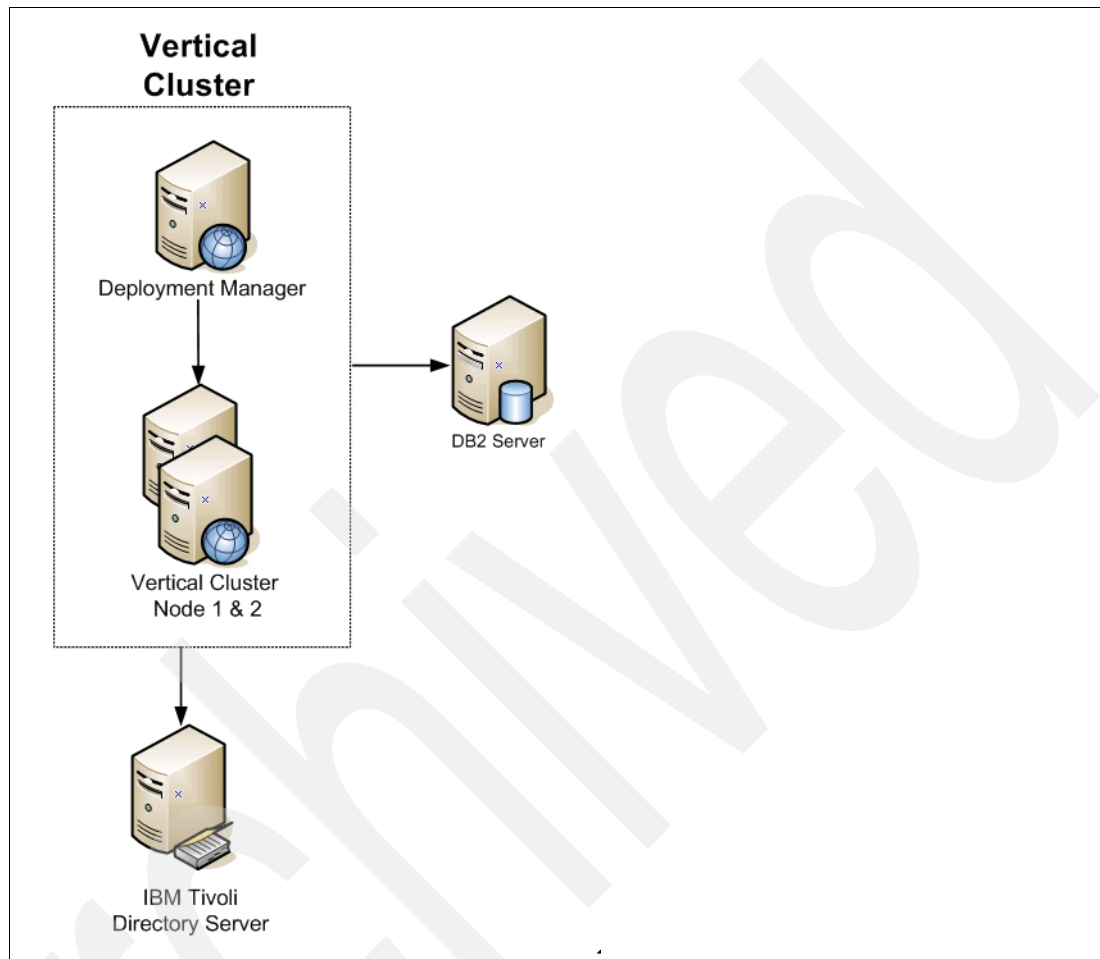


Figure 3-128 Vertical cluster deployment architecture

In our deployment the following applies:

- ▶ DMC1: This machine served as the Deployment Manager. The Deployment Manager is the central administrative console for all nodes in the environment.
- ▶ VerC1N1 and VerC1N2: Nodes 1 and 2 hosted the business logic. After completion of the installation, both machines were clustered for load balancing and failover.
- ▶ DBDOM7: This served as the RDBMS server that hosted the IBM WebSphere Portal database for all nodes and the Deployment Manager.
- ▶ DBIDS represents our LDAP server (IBM Tivoli Directory Server V6), which hosted all directory information used by WebSphere Portal.

The process of creating a vertical cluster (without WebSphere Process Server) is virtually identical to the steps for creating a horizontal cluster without Process Server. As with the horizontal cluster setup we perform the following initial steps:

- ▶ "Installation WSAS on the Deployment Manager" on page 138.
- ▶ "Install WebSphere Portal on the primary cluster node, node 1" on page 146.

- ▶ “Configure primary node 1 to an external database” on page 152.
- ▶ “Create cluster definition” on page 153.
- ▶ “Configure Portal nodes and the DMGR for LDAP security with Realm Support” on page 166.
- ▶ “Configure Portal to use remote Web Servers” on page 166.
- ▶ “Optional steps after cluster creation” on page 167.

The only difference is the process for creating a vertical cluster member on the primary node, which is described in the following section.

Adding a vertical cluster member on the primary node

Add vertical cluster members to share the workload demands of your cluster across multiple members running on the same physical machine.

1. Access the administrative console on the Deployment Manager by opening a browser and typing the following into the address bar:

`http://DM01:9060/ibm/console`

Where DM01 is the Deployment Manager node or host name. The port number might differ based on your installation.

2. Click Servers → Clusters in the console navigation tree, select the cluster name, and then click cluster members from the list of additional properties.



Figure 3-129 Cluster members view

3. Click New to create the cluster member.

- a. Define the name of cluster member.

Note: Do not use spaces in the cluster member name.

Create new cluster members

Add application servers to a cluster.

Step 1: Enter basic cluster member information

Step 2: Summary

Enter basic cluster member information

Enter information about this new cluster member, and click Apply to add this cluster member to the member list. Use the Edit function to edit the properties of a cluster member that are already included in this list. Use the Delete function to remove a cluster member from this list.

* Member name
verc1n2

Select node
[verc1n1Node01(6.0.2.9)]

* Weight
2

☒ Generate Unique Http Ports

Apply

Edit Delete

Select	Application servers	Nodes	Version	Weight
<input checked="" type="checkbox"/>				

Next Cancel

Figure 3-130 Enter basic cluster member information

- Select an existing node where IBM WebSphere Portal is installed.
- Check the box Generate Unique HTTP Ports.
- Click Apply, and then click Next to view the summary.

Important: If you are not using an external Web server in your clustered environment, you must update the virtual host entries for the new port created when adding a cluster member. You can do this by updating the default_host virtual host in the administrative console and adding a new alias entry for the port number (use a ("*") wildcard character for the host name). Refer to the WebSphere Application Server information center for more information.

Create new cluster members

Add application servers to a cluster.

Step 1: Enter basic cluster member information

Step 2: Summary

Summary

Summary of actions:

Server name = verc1n2
Node = verc1n1Node01(6.0.2.9)
Weight = 2
Core Group = DefaultCoreGroup
Clone Template = dmclCell01/verc1n1Node01/WebSpt
Generate Unique Http Ports = true
Node group = DefaultNodeGroup

Previous Finish Cancel

Figure 3-131 Summary window view

- Click Finish, and save the changes.
 - The new cluster topology can be viewed from the Servers → Cluster Topology view.
 - The Servers → Application Servers view lists the new server cluster members.

- c. Start the server.
5. Save your changes and resynchronize the nodes.
 - a. In the administrative console for the Deployment Manager, click Save on the task bar, and save your administrative configuration.
 - b. Select System Administration → Nodes, select the node from the list, and click Full Resynchronize.
6. Check the Port Number for the Vertical cluster member, and create a host alias for it.
 - a. Navigate to Servers → Application Servers → “VerticalClusterMember” → Ports. Check the Port Number for WC_defaulthost.
 - b. Create a host alias for the port that is used by the Vertical cluster member that you checked in the previous step. In our Environment it is 9081.
 - c. Navigate to Environment → Virtual Hosts → Default Host → Host Aliases, and click New.
 - d. In the Hostname field provide a (*). In the Port field, provide the Port that is used by the Vertical cluster member as shown in Figure 3-126 on page 179.

Virtual Hosts

Messages

Changes have been made to your local configuration. Click [Save](#) to apply changes to the master configuration.

The server may need to be restarted for these changes to take effect.

[Virtual Hosts](#) > [default_host](#) > [Host Aliases](#) > *

An alias is the DNS host and port number that a client uses to form the URL request of a Web Application resource. Application resources include servlets and JSP pages. For example, the "myhost:8080" portion of <http://myhost:8080/servlet/snoop>. When no port number is specified, the default port 80 is used.

Configuration

General Properties

* Host Name
*

* Port
9081

Apply OK Reset Cancel

Figure 3-132 Host aliases view

- e. The host aliases should now look like Figure 3-133 on page 185.

Virtual Hosts		
Virtual Hosts > default_host > Host Aliases		
A list of one or more DNS aliases by which the virtual host is known.		
Preferences		
New Delete		
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>		
Select	Host Name	Port
<input type="checkbox"/>	*	9080
<input type="checkbox"/>	*	80
<input type="checkbox"/>	*	9443
<input type="checkbox"/>	*	9443
<input type="checkbox"/>	*	9081
<input type="checkbox"/>	verc1n1.cam.itso.ibm.com	9080
<input type="checkbox"/>	verc1n1.cam.itso.ibm.com	80
Total 7		

Figure 3-133 Host aliases

7. Log into the administrative console for WebSphere Application Server. If you are using the Web server in a clustered environment, open the administrative console for the Deployment Manager machine.
 - a. Select Environment → WebSphere Variables.
 - b. Narrow the scope of the listed variables:
 - i. Click Browse Nodes in the Scope field.
 - ii. Select the node containing your application server.
 - iii. Click Browse Servers.
 - iv. Select the application server for WebSphere Portal (for example, WebSphere_Portal), and click OK.
 - v. Click Apply to update the list of variables.
 - vi. Update the value of the WCM_HOST variable with the fully qualified host name of the Web server.
 - vii. Update the value of the WCM_PORT variable with the port number used to access the Web server.
8. Save your changes and resynchronize the nodes.
 - a. In the administrative console for the DMGR, click Save on the task bar, and save your administrative configuration.
9. Select System Administration → Nodes, select the node from the list, and click Full Resynchronize.
10. Restart the portal server.
11. Regenerate the Web server plug-in.
 - a. Regenerate the Web server plug-in using the DMGR administrative console.
 - b. If you are using a remote Web server, copy the updated plug-in configuration file (plugin-cfg.xml) to the Web server's plug-in configuration directory.
 - c. Stop and start the Web server.

Archived

Multiple LDAP directory support

One of the most crucial components of any WebSphere Portal environment is the user registry. Typically the user registry is in the form of an LDAP directory, and careful consideration needs to be paid as to how we deploy it in order to minimize administration whilst maximizing redundancy and high availability.

This chapter discusses a key new feature of WebSphere Portal Version 6, namely the ability to support multiple different LDAP directories.

We will take a brief look at the advantages of using multiple LDAP directories and discuss some typical deployment scenarios before discussing in detail how to configure Portal for use with multiple LDAP directories.

Finally we look at some of the considerations for configuring multiple LDAP directories in a clustered portal environment.

4.1 Advantages of multiple LDAP directories

The use of multiple LDAP directories has a significant bearing on how we deploy and maintain WebSphere Portal; therefore, it is important to understand what the advantages are of using more than one LDAP directory.

- ▶ **Greater deployment flexibility for WebSphere Portal.** The ability to use multiple LDAP directories of different types means we have much greater freedom in determining how and where we can deploy WebSphere Portal.

For example, we could deploy an extranet portal and leverage an LDAP directory (such as IBM Tivoli Directory Server) in the DMZ for access by our external customers; however, internal employees could access the same extranet portal through the firewall using credentials held in an internal corporate LDAP directory such as Domino or Novell, e-Directory or Active Directory.

- ▶ **Reuse existing directories that may already contain important user data.** Most environments contain more than one user directory. This is usually a by product of having multiple applications as well as file and print services. The ability of WebSphere Portal to be able to use multiple directories means that we no longer have to consolidate user accounts in to a single LDAP directory but can instead make use of the directories we already have.

This is particularly useful in cases where two or more organizations merge or when one organization acquires another and each has different application or corporate LDAP directories containing their users and groups.

Also, if we are deploying portlets for applications that already have their own LDAP directory, then it makes sense to reuse the application's own LDAP directory and avoid potentially time consuming and costly coding and testing to accommodate a different LDAP directory.

- ▶ **Minimize or negate the need for directory consolidation.** In prior releases of WebSphere Portal, consolidating user account information from multiple sources into a single LDAP directory to be used by portal was a fairly common occurrence. This often involved significant administration effort and ongoing maintenance, user disruption, and in some cases required the use of additional software such as IBM Tivoli Directory Integrator.

The ability to support multiple, different LDAP directories should minimize or completely eliminate the need to consolidate directories in this way.

- ▶ **No need to change current administration practices.** Another advantage of multiple LDAP directory support that is often overlooked is that we can continue to manage our existing LDAP directories using the same administrators and tools that we use at present. This minimizes administrator disruption and the need to train or re-train administrators on how to manage user accounts in a different directory (for example managing users in a Domino LDAP directory is significantly different from managing users accounts in Active Directory).
- ▶ **Minimizes the LDAP directory being a single point of failure.** In a multiple LDAP directory portal environment, the failure of an LDAP directory no longer affects all users who access the Portal (see Note below).

4.1.1 Overview of virtual portals and realms

Before discussing in detail how to configure WebSphere Portal Server to use multiple LDAP directories, it is important to have a quick overview of the *virtual portal* feature in WebSphere Portal Server and the associated concept of *realms*.

Readers familiar with WebSphere Portal Server Version 5.1 may already have an understanding of virtual portals and realms, in which case you may choose to skip to the next section, and proceed to the section entitled 4.1.3, “How realms were extended in WebSphere Portal Version 6” on page 190.

4.1.2 The relationship between virtual portals and realms

WebSphere Portal Server Version 5.1 introduced the ability to create *virtual portals*.

Virtual portals are logical portals that share the same hardware and software installation, yet have the ability to serve up different content to different users.

For example, a single portal installation could contain three separate virtual portals—one for employees, one for customers, and one for suppliers.

Each of these virtual portals would then serve up content and applications specific to the employees, customers, or suppliers. Each virtual portal could also be customized to have its own unique look and feel.

One of the limitations with WebSphere Portal Server Version 5.1 however, was that all employees, customers, and suppliers had to exist in the same LDAP directory. Therefore in order to associate users with the correct virtual portal, they had to be logically grouped in to one or more *realms*, and then these realms were associated to a specific virtual portal.

Tip: The use of *realms* is often referred to as “horizontal partitioning”.

Figure 4-1 on page 190 shows an example of the relationship between virtual portals and realms.

Notice that despite only having a single LDAP directory, we have three distinct groupings or realms of users (Employees, Suppliers, and Customers), each of whom are served up their own distinct content from the “virtual” portal associated with their realm.

Following are the advantages of using realms:

- ▶ It allows us to aggregate users from one or more branches of the (same) LDAP directory and expose them to WebSphere Portal Server as a coherent whole.
- ▶ They provide flexible user management, as realms can overlap, for example, a user can belong to multiple realms (best practice however is to try and limit overlapping realms to avoid unnecessary confusion and administrative overhead).
- ▶ A realm can combine multiple suffixes in the LDAP directory and a user suffix can belong to one or more realms.

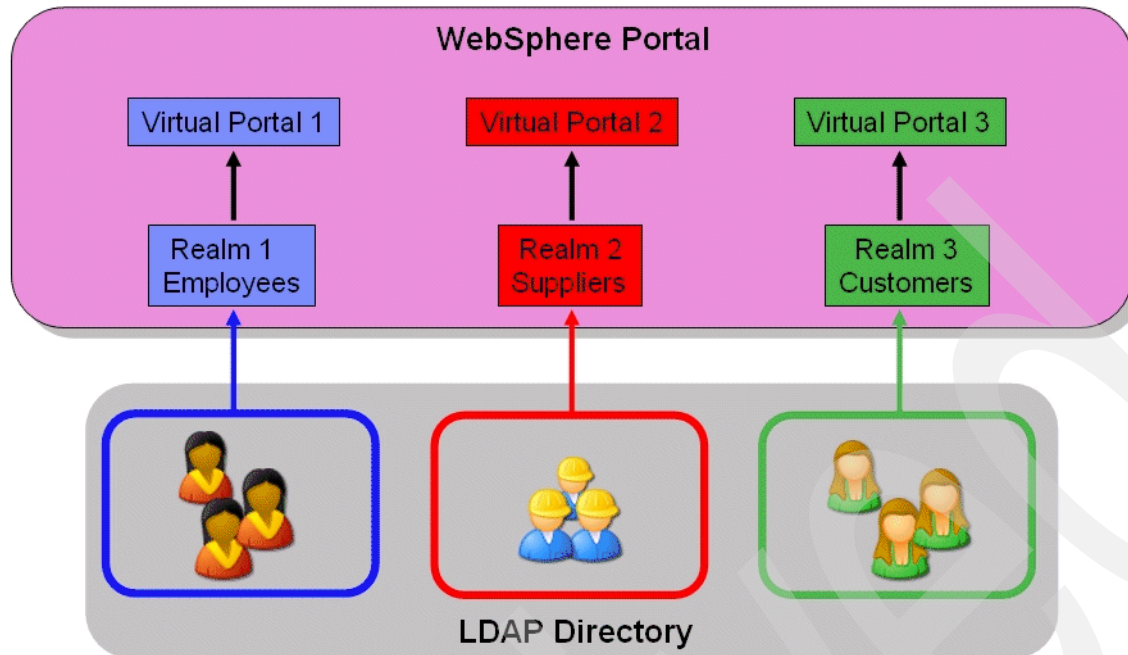


Figure 4-1 A single LDAP directory with multiple virtual portals and realms in WebSphere Portal Server 5.1

4.1.3 How realms were extended in WebSphere Portal Version 6

WebSphere Portal Server Version 6 extends the concept of realms by adding support for multiple LDAP directories.

Whereas in the previous example, all employees, suppliers, and customers had to exist in a single LDAP directory, Figure 4-2 on page 191 shows how we can utilize a separate LDAP directory containing only our customers. This is particularly advantageous as it allows us to manage a specific grouping of users independently.

Also, as previously mentioned, there is no need for the LDAP directories to be of the same type. In Figure 4-2 on page 191 our employees and suppliers could be in an IBM Tivoli Directory Server, while our customers could be stored in a Sun One Directory Server.

Depending on our topology, the LDAP directory for customers could even be on the other side of a firewall.

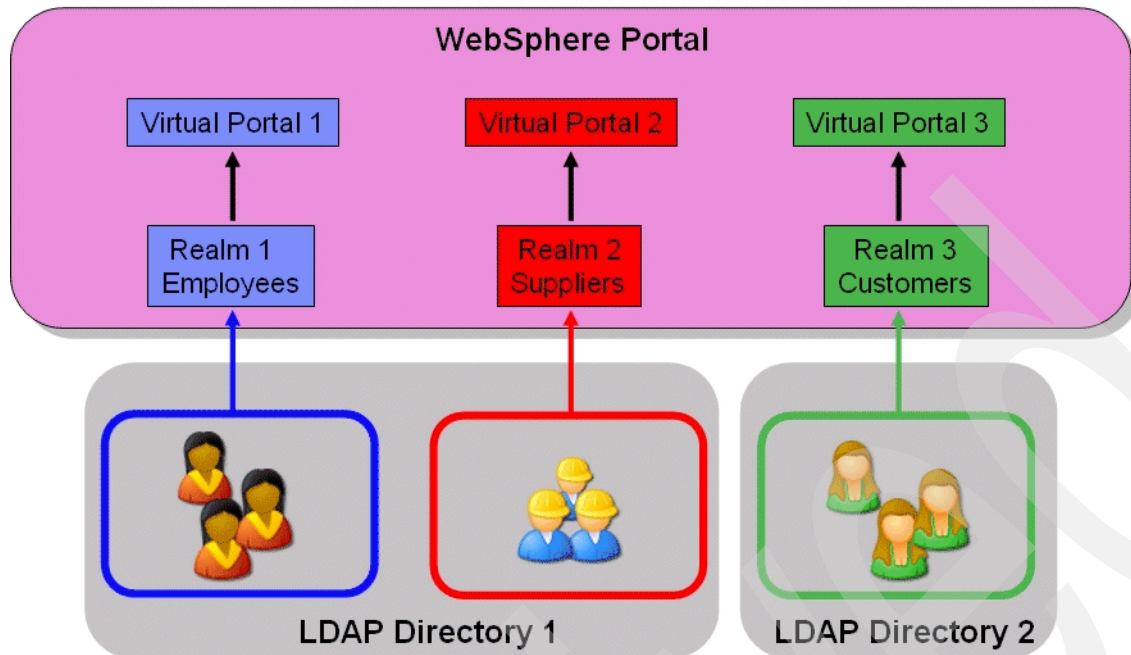


Figure 4-2 Multiple LDAP directories with multiple virtual portals and realms in WebSphere Portal Version 6

4.2 Familiarizing yourself with the LDAP infrastructure

Some of the most common problems associated with deploying a Portal infrastructure arise from issues with the LDAP directory.

As WebSphere Portal Server Version 6 supports the use of multiple LDAP directories (and not all the directories have to be of the same type), it is conceivable that we may now encounter a directory (or directories) with which we are not familiar.

Although LDAP is an industry standard directory protocol, software vendors often implement certain LDAP features in different ways; therefore, subtle differences between LDAP directories can often lead to problems when configuring WebSphere Portal Server and are often the source of much frustration.

It is therefore important to understand and be familiar with your LDAP directory's structure and layout (schema) before embarking on any WebSphere Portal Server configuration.

Section 4.2.1, "Before you begin configuring multiple LDAP directories" on page 192, lists some tips and recommendations for working with LDAP directories.

If you are already experienced with configuring WebSphere Portal Server with an LDAP directory, you can skip the next section and proceed directly to Section 4.3, "Configuring multiple LDAP directories" on page 200.

4.2.1 Before you begin configuring multiple LDAP directories

- ▶ Make sure the LDAP directory is currently supported for use with WebSphere Portal Server Version 6. See Section 4.3.1, "Currently supported LDAP directories" on page 200.
- ▶ Familiarize yourself with the LDAP directory schema, particularly the names of the object classes and attributes. Section 4.2.2, "Tools to help you understand your LDAP directory" on page 193 provides details about some of the tools that can help with this familiarization.
- ▶ Make sure you obtain a user name and password in the LDAP directory that WebSphere Portal Server can use to connect (or "bind") to the LDAP directory. Your organization's LDAP administrator should be able to assist you with this.

Ensure that this username has the appropriate access rights to search the LDAP directory for users and groups and has read/write access to the directory. See note below.

Note: Write access to the LDAP directory is only required if you are going to allow Portal users to "Sign Up" and to create their own login account from Portal or if you have deployed portlets which for some reason need to create or update entries in the LDAP directory. If you are not using these features, then read access to the LDAP directory should be sufficient.

If you are going to be using a read-only LDAP directory with WebSphere Portal Server then you should follow the instructions in the technote located at the following Web address (originally written for WebSphere Portal Server 5.1), prior to enabling security; otherwise, the enable security task will fail:

<http://www-1.ibm.com/support/docview.wss?rs=899&uid=swg21221196>

- Make sure that the servers on which you are installing WebSphere Portal Server can connect to and query the LDAP directory server. This is particularly important if firewalls are involved.

Ensure that you have network connectivity on the correct ports. By default, the LDAP protocol uses TCP/IP port 389 (port 636 if using SSL), so it is important to ensure that network connectivity on these ports is in place prior to installing or configuring WebSphere Portal Server.

There are a number of tools that can be used for this. In Section 4.2.2, “Tools to help you understand your LDAP directory” on page 193, we provide details about the tools we used in our lab environment.

Confirming connectivity to the LDAP directory with one of these tools not only helps you to familiarize yourself with the LDAP directory schema, it also confirms that WebSphere Portal Server can successfully connect to the LDAP directory and search for users and groups.

4.2.2 Tools to help you understand your LDAP directory

Prior to configuring multiple LDAP directories, it is very important to establish connectivity to your LDAP directory and understand your LDAP directory schema.

This section describes several approaches for performing these tasks and the tools you can use to complete them.

4.2.3 Using an LDAP browser tool

Using an LDAP browser or similar tool is essential for familiarizing yourself with the LDAP directory schema and is extremely helpful when specifying things such as the attribute and object class names that are required during WebSphere Portal Server security configuration.

Attribute and object class names are one of the most common LDAP objects that vendors implement differently from one another, so it is important to be able to view and refer to them prior to the installation and configuration of WebSphere Portal Server.

For example, in the Domino LDAP schema, the object class for *group* is called *dominoGroup*; however, in IBM Tivoli Directory Server the object class for groups is called *groupofUniqueNames*.

For our testing in the lab we downloaded a free LDAP browser/editor from the following Web site:

<http://www.iit.edu/~gawojar/ldap/>

This LDAP browser tool is particularly useful because as well as browsing the LDAP directory it also allows you to directly edit LDAP entries.

One of the pre-requisites for this LDAP browser/editor is that Java must be installed on your machine. Java can be downloaded from the following Web site:

<http://www.java.com/en/>

If you are using a Windows system, then after Java is installed, you need to set the JAVA_HOME system variable system variable by right-clicking **My Computer** and going to the **Environment Variables** section.

On the Path statement, add the following value: C:\Program Files\Java\jre1.5.0_06 (for example, the location where Java is installed on your machine).

Tip: Do not put quotes around the path even though it contains a space. This produces a Java error because the LDAP browser program misinterprets the JAVA_HOME variable if it is enclosed in quotes.

1. To launch the LDAP browser/editor, click the **lbe.bat** batch file.
2. Prior to launching it, though, you need to edit the lbe.bat file, and specify the JAVA_HOME variable with the same path as you specified in the Windows environment variables.

See Example 4-1.

Example 4-1 Specifying the JAVA_HOME variable in lbe.bat

```
:setjavahome  
rem #### MODIFY #####  
set JAVA_HOME=C:\Program Files\Java\jre1.5.0_06  
rem #####
```

3. After you launch the LDAP browser/editor, create a profile for your LDAP directory.

The profile consists of the LDAP directory server's host name, base DN, user name, and password with which to bind to the LDAP Directory server, ports, and so on.

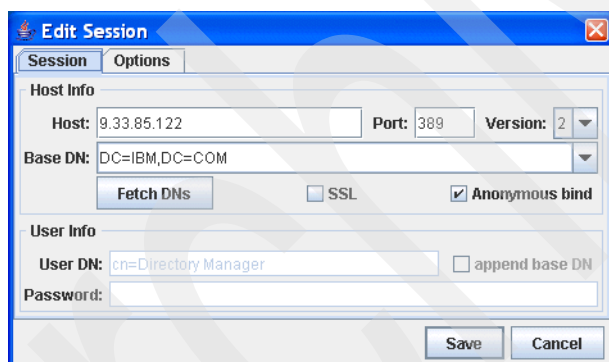


Figure 4-3 Creating a profile with the LDAP browser tool

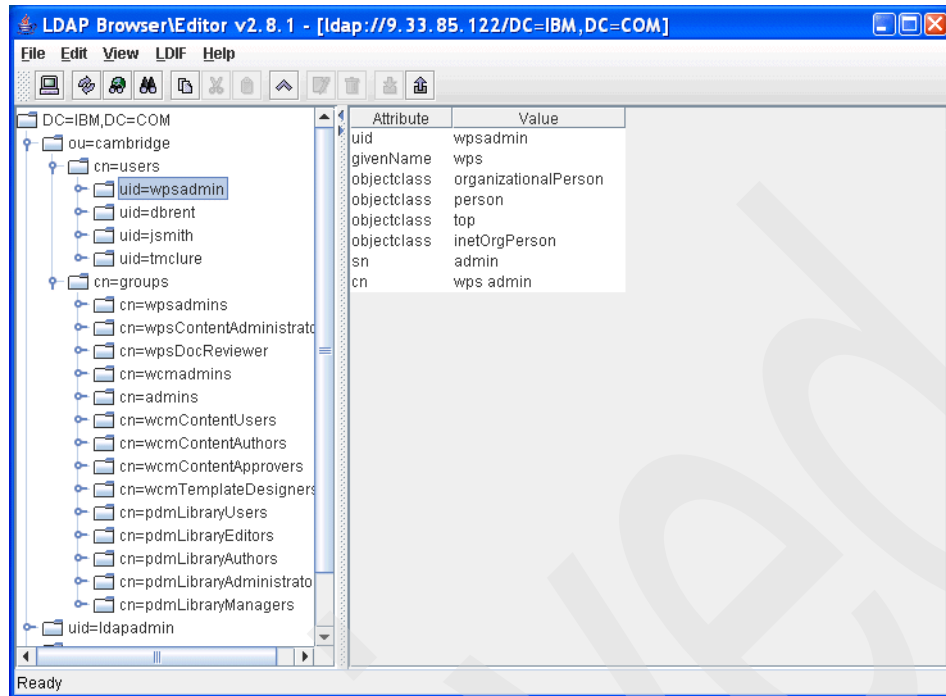


Figure 4-4 Free LDAP browser tool used in our testing

Another popular LDAP browser tool is available from Softerra and can be downloaded from the following Web site:

<http://www.softerra.com>

This particular tool is Windows based and does not require Java; however, it is only a browser not an editor and therefore does not allow you to directly edit LDAP entries.

4.2.4 Using Ldapsearch.exe

Ldapsearch.exe is a commonly used and widely available command line utility that can be extremely helpful in verifying connectivity, authentication, and searching in an LDAP directory.

Ldapsearch.exe is already installed with Lotus Domino Server, Lotus Notes, and IBM Tivoli Directory Server.

During our testing in the lab, we used the Ldapsearch.exe that ships with the Lotus Notes client. If you have Lotus Notes on your workstation it can be found in the Notes Client's Program directory (typically, <install drive>\Program Files\Lotus\Notes).

Tip: Ldapsearch.exe has many options and flags. The following syntax was successful during our testing because it confirmed network connectivity, binding, and searching.

Therefore, we recommend that you use this type of search query when testing with your LDAP directory:

```
ldapsearch -h <hostname> -D "<fully distinguished bind user name>" -w <bind password>
-b "<search base>" "<user name>"
```

If your LDAP search is successful, be sure to make a note of the values you used. These values can then be reapplied to the various security configuration steps for WebSphere Portal Server.

Pay particular attention to the use of quotation marks (" ") in the Ldapsearch example above, and ensure that they are included in your search query where appropriate.

Example 4-2 ldapsearch query to confirm that we can connect to and search our LDAP directory

```
C:\> ldapsearch.exe -h dbids.cam.itso.ibm.com -D "uid=wpsadmin,CN=Users,OU=Cambridge,
DC=ibm,DC=com" -w password -b "DC=ibm,DC=com" "(cn=wps admin)"
```

The following list gives a brief explanation of the values we specified for our Ldapsearch query in Example 4-2.

- ▶ -h is the LDAP server's host name. In our case: dbids.cam.itso.ibm.com.
- ▶ -D is the fully distinguished user name with which to connect (or bind) to the LDAP directory. In our case: uid=wpsadmin,CN=Users,ou=Cambridge,DC=ibm,DC=com.
Where uid is the unique identifier of the user, in our case wpsadmin.
CN=Users, OU= and DC= are the various containers that combine to make up the user's fully distinguished name.
- ▶ -w is the password assigned to the user. In this case, the password was password.
- ▶ -b is the search base or location in the directory to search for users' names and groups. In our case users and groups exist under CN=Users.
- ▶ The final value of CN is the Common Name of a user in the LDAP directory that you want to search for.

Common name is typically the user's First Name Last Name, and in our example, the user had the first name of *wps* and a last name of *admin*.

Example 4-3 Results from a typical ldapsearch query

```
uid=tmclure,cn=users,ou=cambridge,dc=IBM,dc=com
objectclass=organizationalPerson
objectclass=person
objectclass=top
objectclass=inetOrgPerson
uid=tmclure
sn=mclure
givenName=troy
cn=troy mclure
```

Example 4-3 shows the results of another Ldapsearch for the LDAP user cn=troy mclure,ou=cambridge,dc=ibm,dc=com. You may remember this search from such previous

IBM Redbooks as “Domino 6.5.1 and Extended Products Integration Guide” and “Domino Access for Microsoft Outlook®”.

4.2.5 Multiple LDAP directory scenarios

Before discussing the steps required to configure WebSphere Portal Server for multiple LDAP directories, we will take a brief look at some typical deployment scenarios.

These scenarios are important in helping you plan a suitable LDAP directory strategy for your WebSphere Portal Server environment. These scenarios will also assist you in determining the right kind of configuration for your environment and will further reinforce the relationship between multiple LDAP directories and realms.

4.2.6 Planning for multiple LDAP directories

Before configuring multiple LDAP directories you should consider the following:

- ▶ Which LDAP directory will be your primary directory, for example, the one you will use when enabling security in portal for the first time. If you already have security enabled then this is not so much of an issue.
- ▶ How many realms you wish to define, what their purpose will be, and the relationship between the realms and the LDAP directories.
- ▶ How many virtual portals you wish to define, and how they are related to the realms.

Important: There are a number of important considerations for customers using Domino LDAP, especially if your portal environment currently has security enabled with a different LDAP directory and you are considering using Domino LDAP as an additional LDAP directory. See Section 4.4, “Domino LDAP and groups” on page 227 for more details.

4.2.7 Scenario 1, default realm with multiple LDAP directories (one to many)

Figure 4-5 on page 198 shows a typical deployment scenario you may wish to consider in your environment. In this scenario we only have a single realm defined in WebSphere Portal Server. This is the default realm of “**portal**” that is automatically created when we enable WebSphere Portal Server security with realms.

This is the simplest scenario in terms of deployment and configuration for multiple LDAP directories and is suitable when we do not wish to logically group users in any particular way.

This scenario simply requires enabling WebSphere Portal Server with realm support (see section 4.3.4, “Enabling security with realm support” on page 207) and using the default realm of “**portal**” that is automatically created.

As can be seen in Figure 4-5 on page 198, the default realm is associated with two different LDAP directories, for example, we have a one-to-many relationship between the realm and the LDAP directories.

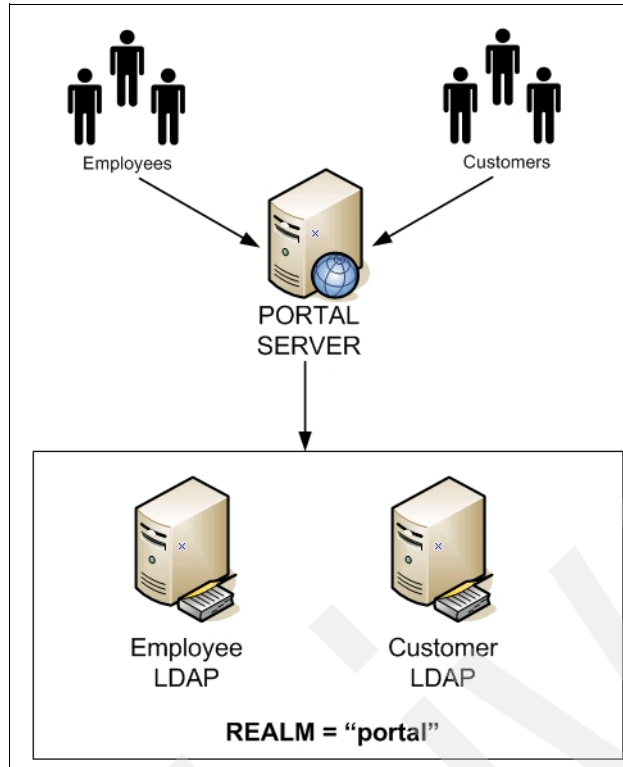


Figure 4-5 Default realm with multiple LDAP directories (one-to-many)

4.2.8 Scenario 2, one realm per LDAP directory (one-to-one)

In this scenario, WebSphere Portal Server is configured with two realms and two LDAP directories so we have a one-to-one relationship between realms and LDAP directories.

Figure 4-6 on page 199 shows this scenario. In fact this is the scenario that we configured in our lab environment.

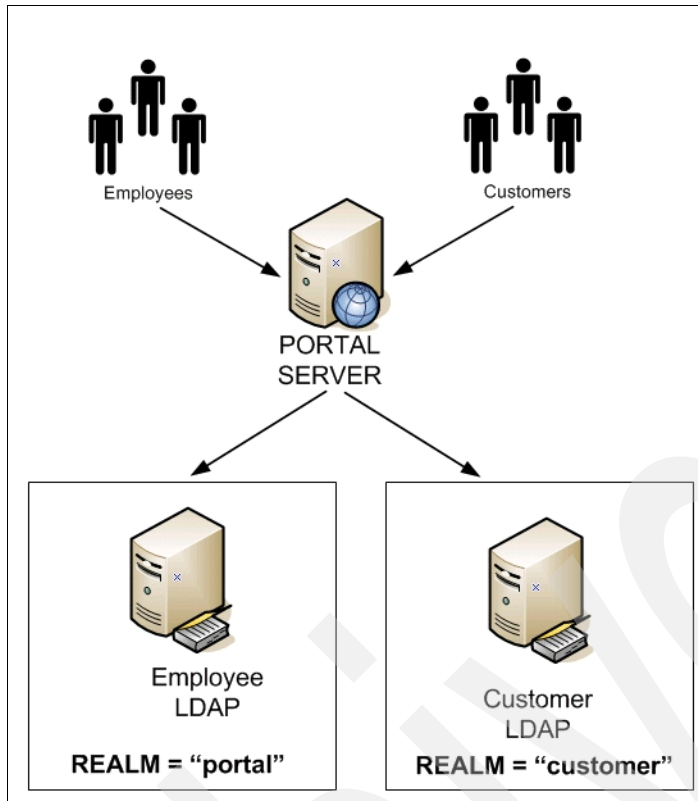


Figure 4-6 Multiple realms with multiple LDAP directories (one-to-one)

The one-to-one scenario is particularly useful when we already have pre-existing distinct logical groups of users.

In this scenario, as we already have all our customers in their own LDAP directory and employees are in a separate one it makes logical sense to associate each directory with its own realm.

Note: Notice how in Figure 4-6 we still use the default realm of **portal** for our employees, whilst creating a new realm called **customers** for our second LDAP directory.

The reason we chose this particular scenario to implement in our lab environment was because it is easily extensible should we decide to add an additional third or fourth LDAP directory. It is also representative of a typical real world scenario.

4.2.9 Scenario 3, multiple realms & multiple LDAP directories (many-to-many)

This scenario is an extension of scenario two; however, instead of an equal number of realms and LDAP directories, here we have a many-to-many relationship. For example, as shown in Figure 4-7 on page 200, the realm *customer* is made up of three separate LDAP directories, while the default realm of *portal* and the *supplier* realm are mapped to their own individual LDAP directories.

This scenario reflects many typical environments where customer data may well be held in different systems, each with their own LDAP directory.

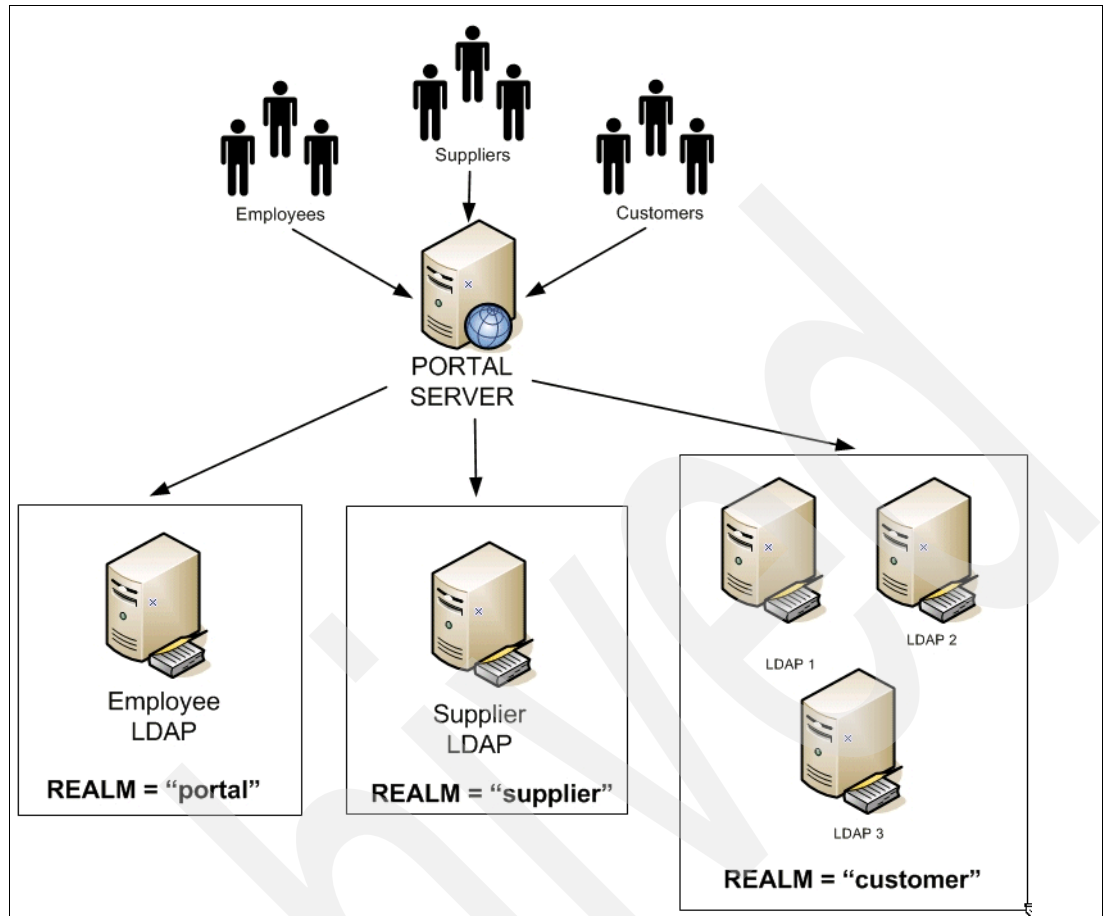


Figure 4-7 Multiple realms with multiple LDAP directories (many-to-many)

4.3 Configuring multiple LDAP directories

This section discusses in detail how to configure WebSphere Portal Server for multiple LDAP directories.

As previously mentioned, we walk you through the steps to configure the environment described in 4.2.8, “Scenario 2, one realm per LDAP directory (one-to-one)” on page 198.

4.3.1 Currently supported LDAP directories

Before configuring a primary or secondary LDAP directory ensure that the LDAP directory in question is currently supported for use with WebSphere Portal Server.

The current list of supported LDAP directories are as follows:

- ▶ Domino 6.5.4, 6.5.5 & 7.x
- ▶ IBM Tivoli Directory Server 5.2 & 6.0
- ▶ Novell eDirectory 8.7.3
- ▶ Active Directory 2000 & 2003
- ▶ Active Directory 2003 with ADAM (see note below)
- ▶ Sun One Directory Server 5.2

Note: ADAM is an acronym for Active Directory Application Mode.

Tip: Check the WebSphere Portal Server Version 6 Infocenter for the latest list of supported LDAP directories. The Infocenter is located at the following Web site:

<http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp>

Make sure that you applied any of the latest fixpacks that may be available for your LDAP directory.

4.3.2 Pre-requisites

Aside from ensuring that all LDAP directories to be used with WebSphere Portal server are supported, there are a number of other pre-requisites that must be met in order to use multiple LDAP directories.

- ▶ Security in WebSphere Portal Server must be enabled with realm support. If you currently have security enabled without realm support, then you must disable and re-enable it. See section 4.3.3, “Disabling WebSphere Application Server security” on page 202.
 - To use Realms, the WebSphere Application Server user registry provided by Member Manager must be used as the authentication mechanism in WebSphere Application Server.
 - When in non-security mode, using *enable-security-ldap*, or registering a custom user registry as the WebSphere Application Server Authentication mechanism means that realms cannot be enabled.
- ▶ A realm must be mapped to a Virtual Portal (you can simply use the default realm of *portal* if you wish).
- ▶ In order to log into a particular virtual portal, a user must be a member of the realm that is associated with that virtual portal.
- ▶ Prior to any configuration, decide which directory will act as your primary LDAP directory, and enable security to it first before configuring any additional LDAP directories.
- ▶ When using multiple LDAP directories, you must ensure that the distinguished names and the log on attributes (for example, the UID) are unique over all LDAP directories. In other words, which ever attribute or attributes you defined to allow users to log in to portal (such as UID, e-mail address, common name, and so on), must be unique across all LDAP directories.

In our lab environment we ran in to issues as we had a user called “John Delaney” in one LDAP directory and a user called “Jack Delaney” in another.

As we had configured portal to use the UID attribute for log in, we now had the situation of two users in different LDAP directories both with a uid=jdelaney (see Figure 4-8 on page 202).

Once multiple LDAP directories were configured, neither user could log in.

We overcame this issue by editing the “shortname” field for Jack Delaney in the Domino LDAP directory to uid=jdelaney2 to make it unique.

Important tip: In our lab environment, we also had duplicate wpsadmin users as well as duplicate wpsadmins groups.

Even though they had different distinguished names (DNs) and resided in different parts of their respective LDAP directories, we found that this still caused our portal server to shut down every time we attempted to log in as a user from either directory.

Therefore you should ensure that *only* your primary LDAP directory contains the portal admin user ID and group.

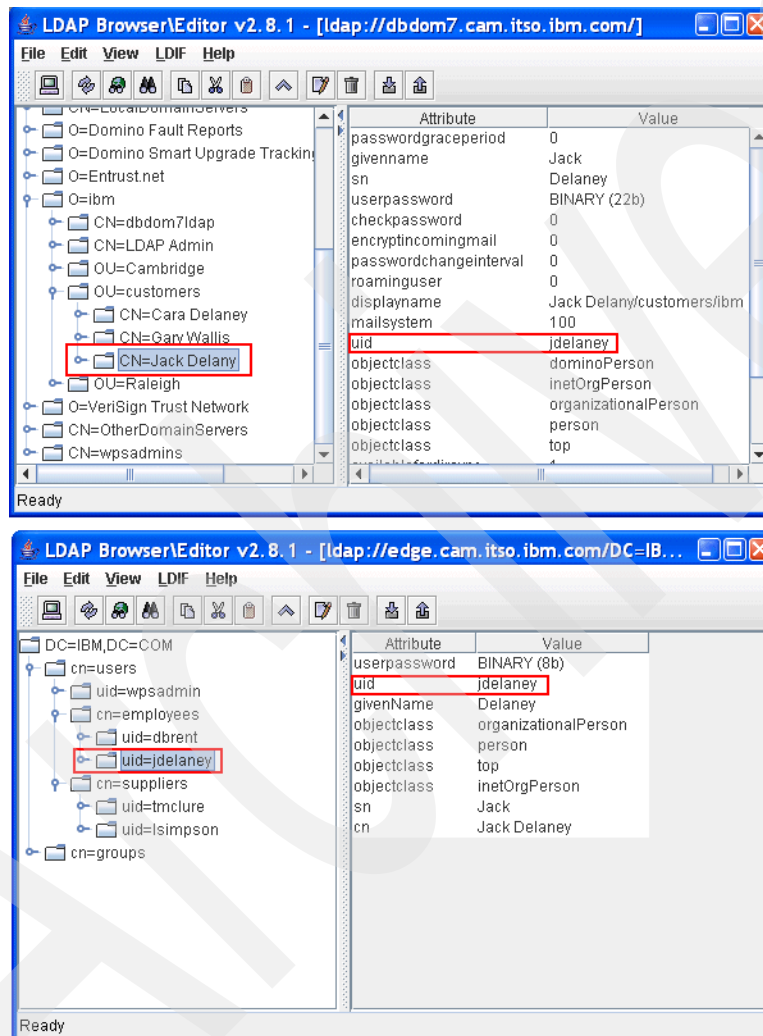


Figure 4-8 Duplicate UIDs across multiple LDAP directories

4.3.3 Disabling WebSphere Application Server security

By default, when you install a WebSphere Portal Server, security is already enabled. However it is not enabled with realm support so it needs to be disabled and then re-enabled with realm support if we are to use multiple LDAP directories.

If you have an existing WebSphere Portal server environment already up and running with security already enabled *with* realm support then there is no need to disable security.

Disabling security

Use the following steps to disable security on your portal server if you do not already have security with realm support enabled.

1. Make a copy of the security_disable.properties helper file. The file is located in `<wp_root>/config/helpers/security_disable.properties`.
2. Edit the security_disable.properties helper, and modify the following properties to the values that match your current security configuration, specifically the following:

- wmm.DbPassword
- WasPassword

Also, you must change the following properties in the security_disable.properties helper file to what you desire your portal administrator user name and password to be *after* security is disabled.

- PortalAdminId
- PortalAdminPwd
- PortalAdminGroupId

3. Run the config wizard to disable security. Invoke the config wizard by running the following script, `<wp_root>/config/wizard/configwizard.bat`.

Note: In a clustered environment, make sure you disable security on the primary node. See Chapter 3 for details on disabling security in a cluster.

4. Click next on the Welcome Page window as shown in Figure 4-9.

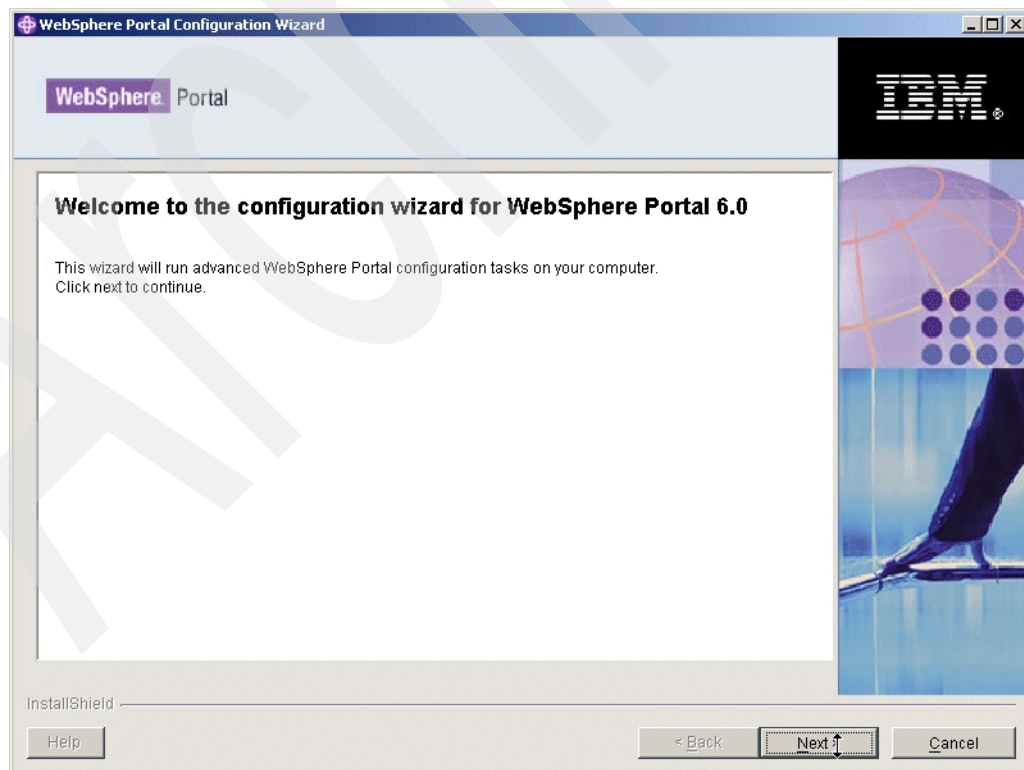


Figure 4-9 Configuration wizard welcome page

5. Select **Disable security**, and click Next.

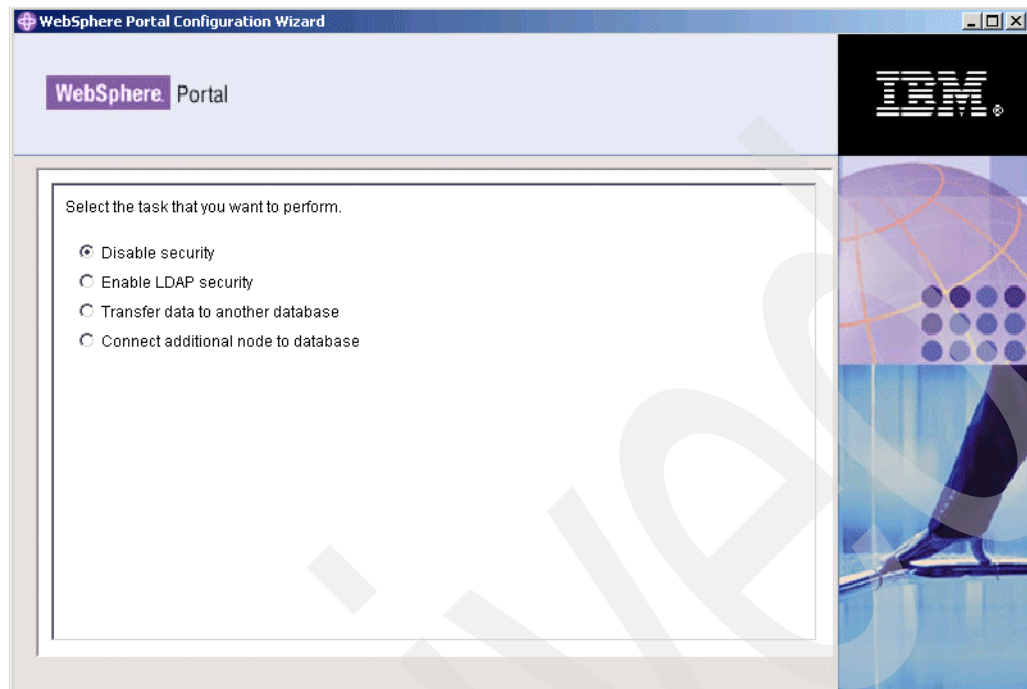


Figure 4-10 Disabling security with the configuration wizard

6. Enter the WebSphere Application Server Administrator user name and password, and then click Next.

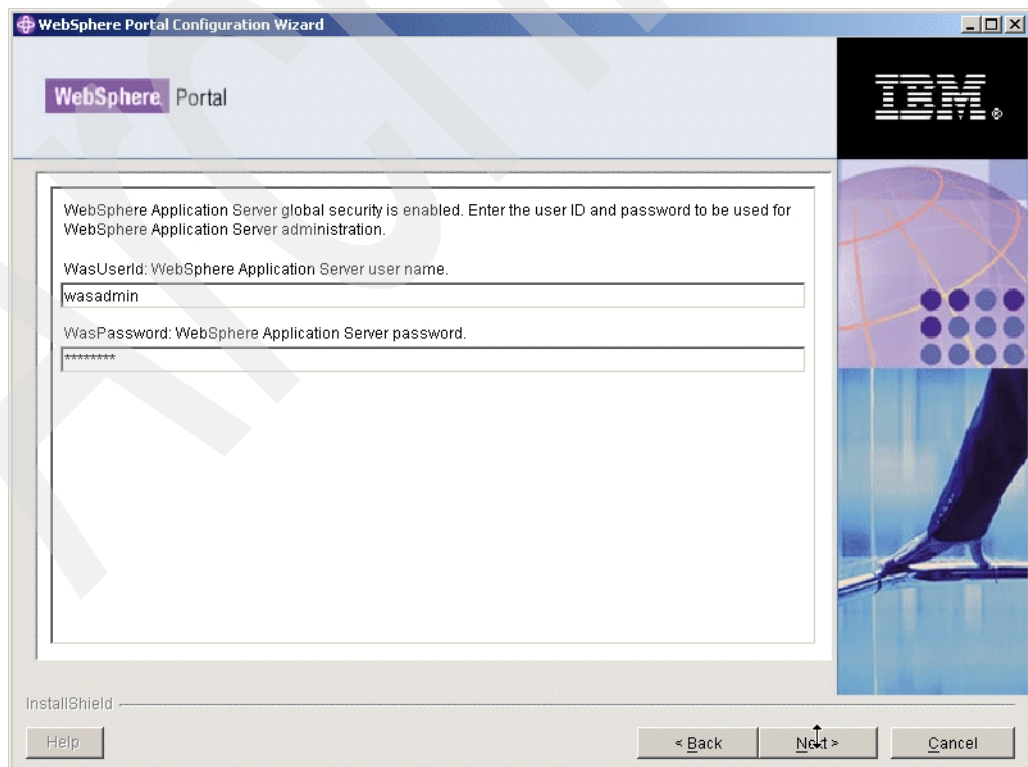


Figure 4-11 Enter the WebSphere Application Server Administrator user name and password

7. Select the location of the security_disable.properties helper file we previously edited, and click Next.

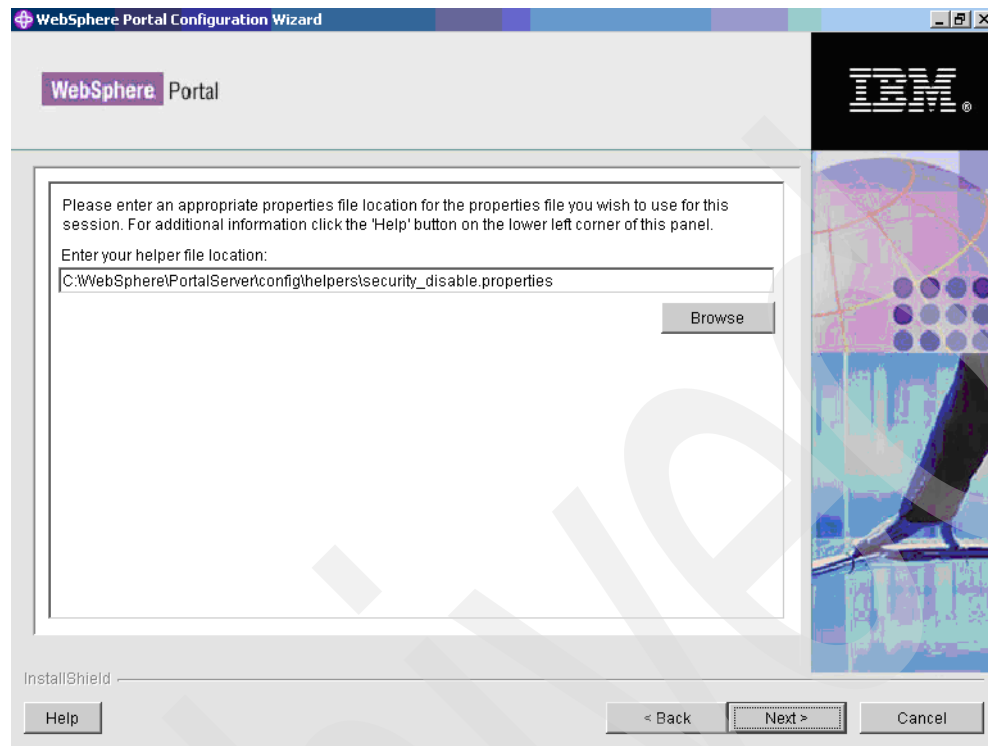


Figure 4-12 Helper file location window

8. Enter the WMM database user name and password, and then click Next.

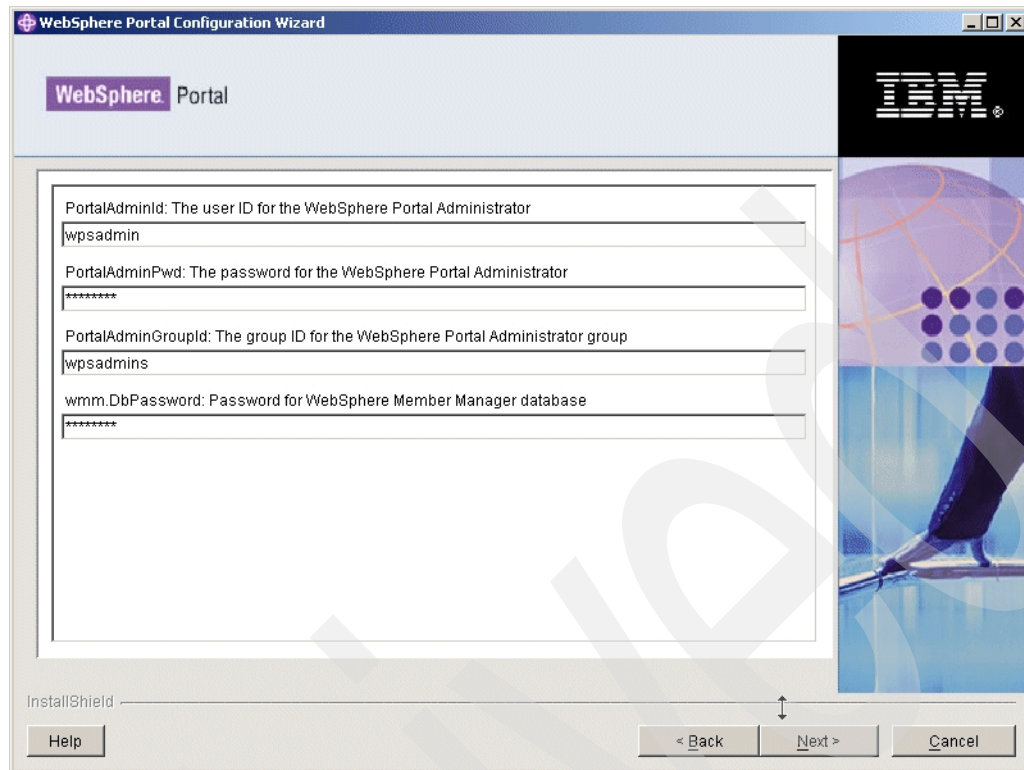
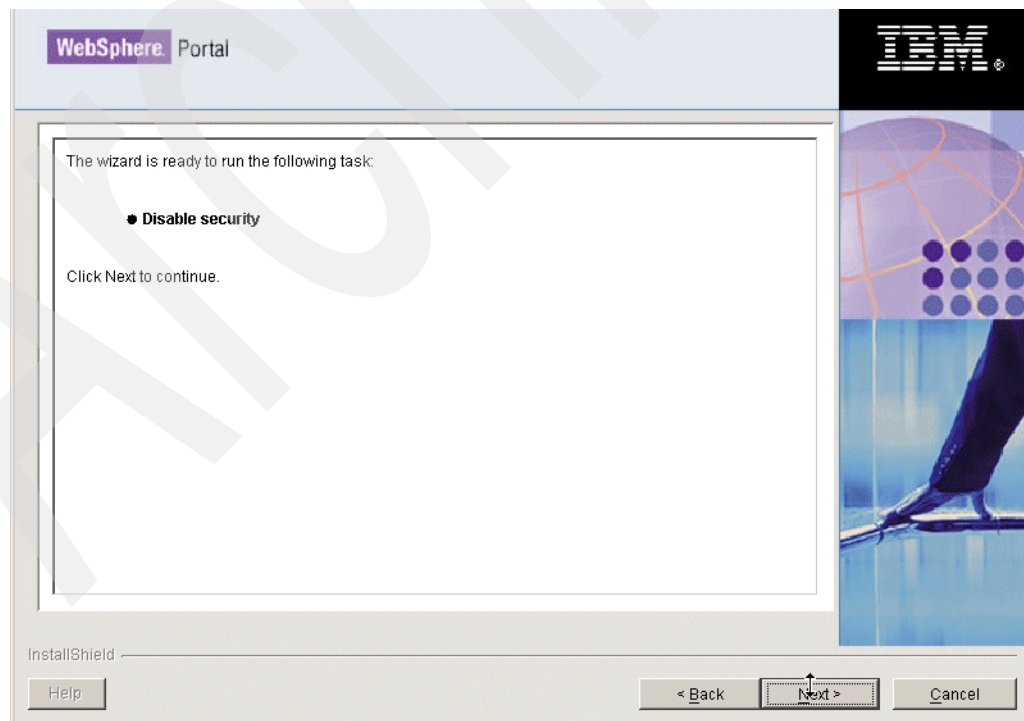


Figure 4-13 WMM database ID and password window

9. Review the summary panel and click next to start begin the process of disabling security.



10. After the wizard successfully completes, ensure that the portal server is stopped.

Note: In a clustered environment, ensure that *all* portal servers are stopped *and* that all nodeagents and the Deployment Manager (DMGR) are running before re-enabling security with realms. See Chapter 3, for details on enabling security with realm support in a cluster.

4.3.4 Enabling security with realm support

Enabling security with realm support cannot be carried out using the configuration wizard. It has to be configured manually.

Note: See Chapter 3 for the steps on enabling security with realm support in a cluster.

1. Make a copy of the wpconfig.properties and wpconfig_domain.properties files located in the <wp_root>/config/ directory.
2. Edit the wpconfig.properties file with the values appropriate for your LDAP directory server.

Table 4-1, Table 4-2 on page 208, Table 4-3 on page 208, Table 4-4 on page 209, Table 4-5 on page 209, Table 4-6 on page 211, and Table 4-7 on page 212 all list the properties that you need to update with values specific to your environment, prior to enabling security with realms.

See the Infocenter document entitled “Configuring LDAP for realm support” for more details on these properties and their recommended values.

Tip: Ensure that all LDAP related values and DN's are entered in *lower case*. This is especially important for Domino LDAP as, although Domino LDAP returns the DN capitalized (for example, CN=wpsadmin,OU=Cambridge,O=ibm), in our lab testing we found that specifying CN= or OU= in upper case in the wpconfig.properties file, rather than as cn= or ou= in lower case, actually caused our enable security with realms task to fail.

Table 4-1 WebSphere Application Server properties

Property	Value
WasUserid	The user ID for WebSphere Application Server security authentication. This should be the fully qualified distinguished name (DN) of a current administrative user for the WebSphere Application Server. For LDAP configuration this value should not contain spaces.
WasPassword	The password for WebSphere Application Server security authentication. Note: If a value is specified for WasPassword, a value must also be specified for WasUserid. If WasPassword is left blank, WasUserid must also be left blank.

Table 4-2 Portal configuration properties

Property	Value
PortalAdminId	<p>The user ID for the WebSphere Portal administrator, which should be the fully qualified distinguished name (DN).</p> <p>Note:</p> <ul style="list-style-type: none"> - For LDAP configuration this value should not contain spaces. - Make sure to type the value in lower case, regardless of the case used in the distinguished name (DN).
PortalAdminPwd	The password for the WebSphere Portal administrator, as defined in the PortalAdminId property.
PortalAdminGroupId	<p>The group ID for the group to which the WebSphere Portal administrator belongs.</p> <p>Note: Make sure to type the value in lower case, regardless of the case used in the distinguished name (DN).</p>
WpsContentAdministrators	The group ID for the WebSphere Content Administrator group.
WpsContentAdministratorsShort	The WebSphere Content Administrators group ID.
WpsDocReviewer	The group ID for the WebSphere Document Reviewer group.
WpsDocReviewerShort	The WebSphere Document Reviewer group ID.

Table 4-3 WebSphere Portal Security LTPA and SSO configuration

Property	Value
LTPAPassword	The password for the LTPA token.
LTPATimeout	Specifies the number of minutes after which an LTPA token will expire.
SSODomainName	<p>Specifies the domain name for all allowable single signon host domains.</p> <ul style="list-style-type: none"> - Enter the part of the domain that is common to all servers that participate in single signon. For example, if WebSphere Portal has the domain portal.us.ibm.com and another server has the domain another_server.ibm.com, enter ibm.com. - To specify multiple domains, use a semicolon (;) to separate each domain name. For example, your_co.com;ibm.com.

Table 4-4 LDAP properties configuration

Property	Value
LookAside	<p>You can either install with LDAP only or with LDAP using a Lookaside database. The purpose of a Lookaside database is to store attributes that cannot be stored in your LDAP server.</p> <p>Note: Set Lookaside to true if you are using IBM Workplace Web Content Management, the Common Mail portlet, or the Common Calendar portlet.</p>
WmmDefaultRealm	The default realm of the Member Manager user registry (UR) configuration. Set this property before enabling security with enable-security-wmmur-ldap.
LDAPHostName	The host information for the LDAP server that WebSphere Portal will use.
LDAPPort	The server port of the LDAP directory.
LDAPAdminUId	<p>The user ID for the administrator of the LDAP directory. Member Manager uses this ID to bind to the LDAP to retrieve users' attributes, create new users and groups in the LDAP and update user attributes. This ID is not required to be the LDAP admin DN, but rather an ID with sufficient authority for the use cases just cited. If this property is omitted, the LDAP is accessed anonymously and read-only.</p> <p>Note: Make sure to type the value in lower case, regardless of the case used in the distinguished name (DN).</p>
LDAPAdminPwd	The password for the LDAP directory administrator, as defined in the LDAPAdminUId property. If the LDAPAdminUID is blank, this property must be blank as well.
LDAPServerType	<p>The type of LDAP server to be used.</p> <p>Value types</p> <ul style="list-style-type: none"> - Tivoli Directory Server: IBM_DIRECTORY_SERVER - Lotus Domino: DOMINO502 - Active Directory: ACTIVE_DIRECTORY - Sun Java System Directory Server: IPLANET - Novell eDirectory: NDS

Table 4-5 Advanced LDAP configuration

Property	Value
LDAPSuffix	The LDAP Suffix. Choose a value appropriate for your LDAP server. This is the distinguished name (DN) of the node in the LDAP containing all user and group information for the Portal being configured.

Property	Value
LdapUserPrefix	The RDN™ prefix attribute name for user entries. Choose a value appropriate for your LDAP server.
LDAPUserSuffix	<p>The DN suffix attribute name for user entries. Choose a value appropriate for your LDAP server.</p> <p>Note: Make sure to type the value in lower case, regardless of the case used in the distinguished name (DN).</p>
LdapGroupPrefix	<p>The DN suffix attribute name for group entries. Choose a value appropriate for your LDAP server.</p> <p>Note: Make sure to type the value in lower case, regardless of the case used in the distinguished name (DN).</p>
LDAPGroupSuffix	<p>The DN suffix attribute name for group entries.</p> <p>Value types</p> <ul style="list-style-type: none"> - Tivoli Directory Server: cn=groups - Lotus Domino: this value is null - Active Directory: cn=groups - Sun Java System Directory Server: ou=groups - Novell eDirectory: ou=groups
LDAPUserObjectClass D	<p>The LDAP object class of the Portal users in your LDAP directory that will log into the Portal being configured.</p> <p>Value types</p> <ul style="list-style-type: none"> -Tivoli Directory Server: inetOrgPerson - Lotus Domino: dominoPerson - Active Directory: user - Sun Java System Directory Server: inetOrgPerson - Novell eDirectory: inetOrgPerson
LDAPGroupObjectClass	The LDAP object class of all the groups in your LDAP directory that the Portal will access.
LDAPGroupMember	The attribute name in the LDAP group object of the <i>membership</i> attribute. Choose a value appropriate for your LDAP server.

Property	Value
LDAPUserFilter	<p>The filter used by WebSphere Application Server for finding users in the LDAP.</p> <p>Value types</p> <ul style="list-style-type: none"> - Tivoli Directory Server: (&(uid=%v)(objectclass=inetOrgPerson)) - Lotus Domino: (&(l(cn=%v)(uid=%v))(l(objectclass=dominoPerson)(objectclass=inetOrgPerson))) - Active Directory: (&(l(cn=%v)(samAccountName=%v))(objectclass=user)) - Sun Java System Directory Server: (&(uid=%v)(objectclass=inetOrgPerson)) - Novell eDirectory: (&(uid=%v)(objectclass=inetOrgPerson))
LDAPGroupFilter	<p>The filter used by WebSphere Application Server for finding groups in the LDAP.</p> <p>Value types</p> <ul style="list-style-type: none"> - Tivoli Directory Server: (&(cn=%v)(objectclass=groupOfUniqueNames)) - Lotus Domino: (&(cn=%v)(l(objectclass=dominoGroup)(objectclass=groupOfNames)(objectclass=groupOfUniqueNames))) - Active Directory: (&(cn=%v)(objectclass=group)) - Sun Java System Directory Server: (&(cn=%v)(objectclass=groupOfUniqueNames)) - Novell eDirectory: (&(cn=%v)(objectclass=groupOfUniqueNames))

Table 4-6 IBM Workplace Web Content Management properties

Property	Value
WcmAdminGroupId	The group ID for the Web Content Management Administrators group. This should be the fully qualified distinguished name (DN) of a current administrative user for the WebSphere Application Server. For LDAP configuration this value should not contain spaces.
WcmAdminGroupIdShort	The Web Content Management Administrators group ID.

Table 4-7 WebSphere Portal Security LTPA and SSO configuration (Optional)

Property	Value
SSORequiresSSL	The property that specifies that single sign-on function is enabled only when requests are over HTTPS Secure Socket Layer (SSL) connections.

3. Edit the wpconfig_domain.properties file. Update the properties shown in Table 4-8 with the values appropriate for your environment.

Table 4-8 wpconfig_domain.properties properties

Property	Value
wmm.DbUser	The user ID for the database administrator.
wmm.DbUser	The password for the database administrator. Note: A value must be set for this property. You cannot leave it empty.

4. Stop the portal server

Run the following task to validate the values you entered in the wpconfig.properties and wpconfig_domain.properties files.

Figure 4-14 shows a successful validation of the security settings.

WPSconfig.bat validate-wmmur-ldap

```

C:\WINDOWS\system32\cmd.exe

[[ldapcheck] #####
[[ldapcheck] ldapURL      : edge.cam.itso.ibm.com:389
[[ldapcheck] ldapUser     : uid=wcsadmin,cn=users,dc=ibm,dc=com
[[ldapcheck] ldapPassword : *****
[[ldapcheck] ldapSslEnabled : false
[[ldapcheck] ldapSslEnabled : false
[[ldapcheck] objectDn      : cn=wcsadmins,cn=groups,dc=ibm,dc=com
[[ldapcheck] #####
[[ldapcheck] Checking for 'cn=wcsadmins,cn=groups,dc=ibm,dc=com'
[[ldapcheck] #####
[logmsg] 2006.10.18 14:50:12.469 action-validate-ldap-wcm-admin-group
[logmsg] EJPCA3102I: Configuration task "action-validate-ldap-wcm-admin-group" is complete.

action-post-config:
Wed Oct 18 14:50:12 EDT 2006
[delete] Deleting: C:\IBM\WEBSPPH~1\PORTAL~1\config\work\was\wp_portal.properties
[delete] Deleting: C:\IBM\WEBSPPH~1\PORTAL~1\config\wpconfig_ascii.properties

BUILD SUCCESSFUL
Total time: 15 seconds

C:\IBM\WebSphere\PortalServer\config>

```

Figure 4-14 Validating credentials for realms

5. Run the following task on the Portal server to enable security with realms.

WPSconfig.bat enable-security-wmmur-ldap


```

C:\WINDOWS\system32\cmd.exe
[exec] ADMU3100I: Reading configuration for server: WebSphere_Portal
[exec] ADMU3200I: Server launched. Waiting for initialization status.
[exec] ADMU3000I: Server WebSphere_Portal open for e-business; process id i
s 2224
[logmsg] 2006.10.18 19:43:38.969 enable-wcm-security-wmmur-ldap
[logmsg] EJPCA3102I: Configuration task "enable-wcm-security-wmmur-ldap" is
complete.

action-full-sync-nodes:
Wed Oct 18 19:43:40 EDT 2006
[logmsg] 2006.10.18 19:43:40.109 enable-security-wmmur-ldap
[logmsg] EJPCA3154I: Configuring security with CUR

action-post-config:
Wed Oct 18 19:43:40 EDT 2006
[delete] Deleting: C:\IBM\WEBSPH~1\PORTAL~1\config\work\was\wp_portal.properties
[delete] Deleting: C:\IBM\WEBSPH~1\PORTAL~1\config\wpconfig_ascii.properties

BUILD SUCCESSFUL
Total time: 19 minutes 13 seconds
C:\IBM\WebSphere\PortalServer\config>

```

Figure 4-15 Enabling security with realms

Note: In a clustered environment, run the `enable-security-wmmur-ldap` task on the primary node only. This configures LDAP security on the Deployment Manager and subsequently all other portal nodes in the cluster.

See Chapter 3 for details on how to enable security with realms in a cluster.

Tip: You might like to run the `validate-wmmur-ldap` and `enable-security-wmmur-ldap` tasks and pipe the output to a text file. For example:

```
WPSconfig.bat enable-security-wmmur-ldap > enable.log
```

In the event there is a problem you can then analyze the resulting log file to help determine the cause of the issue.

6. In a *clustered* environment, if you enabled security with realms, you also need to manually edit the `wmmWASAdmin.xml` file on the Deployment Manager.

If you do not edit this file, then stopping and logging into the Deployment Manager server will require you to use the fully distinguished user name with the `stopServer.bat` and `serverStatus.bat` files.

The `wmmWASAdmin.xml` file is located on the Deployment Manager at `<dmgr_profile_root>/config/wmm/wmmWASAdmin.xml` and should look similar to the one in Example 4-4.

Example 4-4 Editing the `wmmWasAdmin.xml`

```

<?xml version="1.0" encoding="UTF-8"?>
  <wmmWASAdmins>
    <admin logonId=" uid=wpsadmin,cn=users,dc=IBM,dc=com"
      logonPassword="anvu7zPZ7jbrZLa4h89Tfg==" uniqueUserId="
      uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com"/>
  </wmmWASAdmins>

```

7. Add another line between the `<wmmWASAdmins>` tags that include the short name for the WebSphere Application Server Administrator user name.

Since the fully distinguished name and short name have the same password, you can simply copy and paste the current <admin logonId> tag entry and modify it as in Example 4-5.

Example 4-5 Adding the shortname for the WebSphere Application Server Administrator

```
<?xml version="1.0" encoding="UTF-8"?>
  <wmmWASAdmins>
    <adminlogonId=" uid=wpsadmin,cn=users,ou=firebrigade,dc=IBM,dc=com"
      logonPassword="anvu7zPZ7jbrZLa4h89Tfg==" uniqueUserId="
      uid=wpsadmin,cn=users,dc=IBM,dc=com"/>
    <adminlogonId="wpsadmin" logonPassword="anvu7zPZ7jbrZLa4h89Tfg=="
      uniqueUserId=" uid=wpsadmin,cn=users,dc=IBM,dc=com"/>
  </wmmWASAdmins>
```

Tip: If you do not want to edit the wmmWASAdmin.xml file on the Deployment Manager, another option is to enable the Deployment Manager for Localmode as described in an Infocenter document located at the following Web address:

http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/topic/com.ibm.wp.ent.doc/wpf/wmm_mltpl_realm.html#wmm_mltpl_realm__local

4.3.5 Configuring an additional LDAP directory and enabling realms

As mentioned in section 4.2.8, “Scenario 2, one realm per LDAP directory (one-to-one)” on page 198, in our lab environment we built the following:

- ▶ A non-clustered / non-managed WebSphere Portal Server node with security enabled (to IBM Tivoli Directory Server).
- ▶ The IBM Tivoli Directory Server as a primary LDAP server containing all our employees.
- ▶ A Domino 7.0.2 LDAP server as a secondary LDAP server containing all our customers.
- ▶ The following three realms:
 - The default realm of *portal*, which is automatically created when you enable security with realm support (and which in our case maps to the IBM Tivoli Directory Server).
 - A realm called *employees* that maps to the IBM Tivoli Directory Server.
 - A realm called *customers* that maps to the Domino 7.0.2 LDAP server

Figure 4-16 on page 215 is a graphical representation of the environment we built in our test lab.

Important: You can see from the list of realms that the IBM Tivoli Directory Server belongs to both the default realm of *portal* as well as the realm called *employees*.

This is not a problem because realms can overlap if necessary and the primary LDAP directory is always automatically associated with the default realm.

It is generally recommended to avoid overlapping realms if possible as it can make administration and troubleshooting difficult; however, for the purposes of our lab environment, we created the additional realm of *employees* to help make the distinction between LDAP directories clearer for the reader. Also, this scenario reflects a typical real world environment.

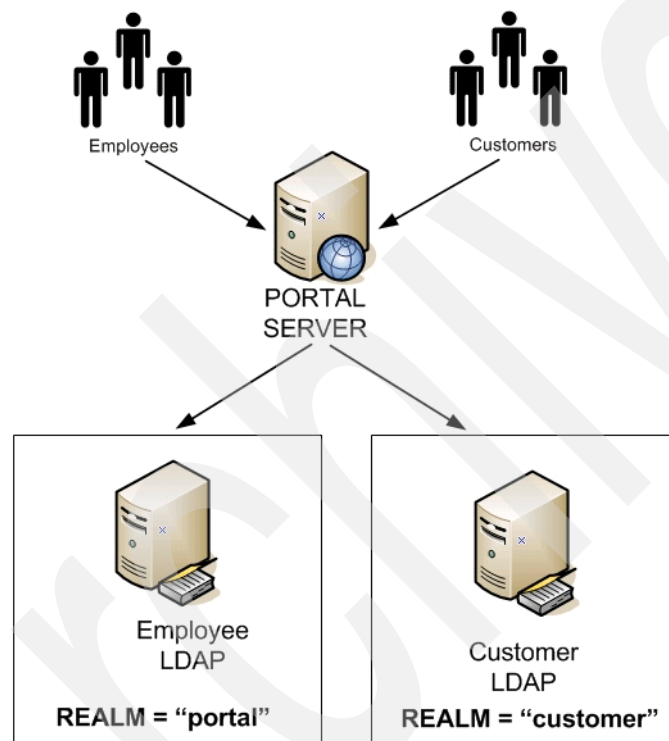


Figure 4-16 Two realms and two LDAP directories

Figure 4-17 on page 216 shows our employee users in the primary IBM Tivoli Directory Server LDAP directory.

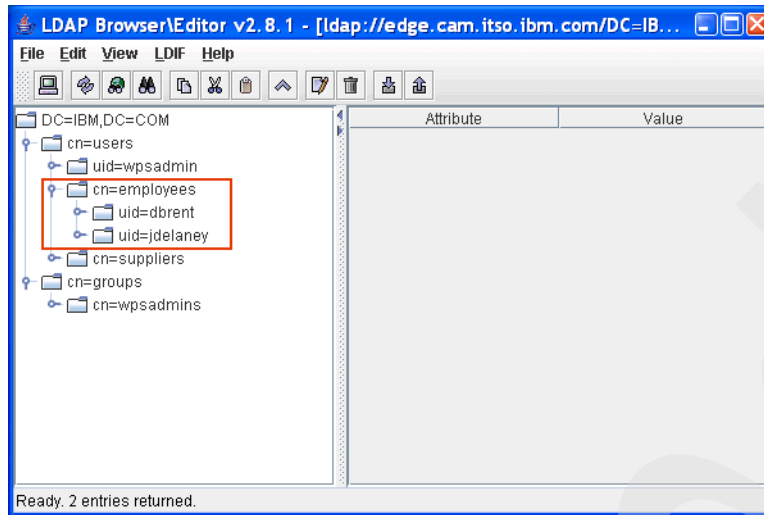


Figure 4-17 Employees in the IBM Tivoli Directory Server

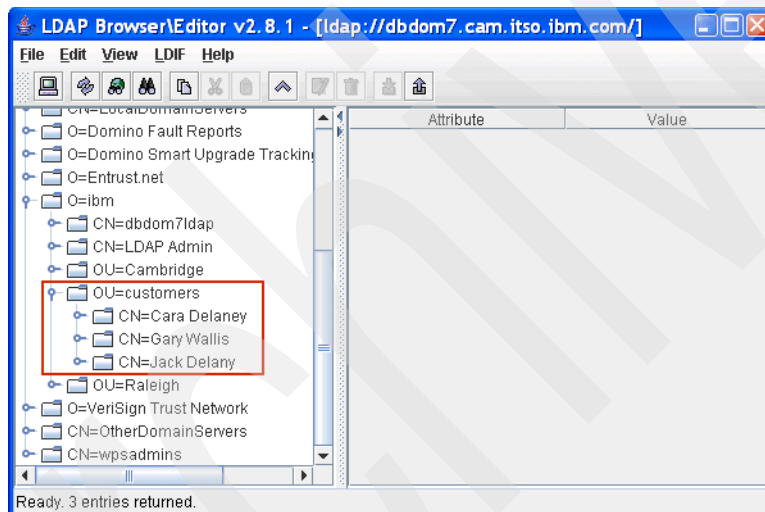


Figure 4-18 Customers in the Domino LDAP directory

4.3.6 Steps for adding an additional LDAP directory

Attention: The following instructions are for a non-clustered environment. If you are enabling an additional LDAP directory in a cluster, see section 4.5, “Enabling multiple LDAP directories in a cluster” on page 230.

1. Stop the WebSphere Portal Server.

As security is enabled you will need to supply a valid user name and password.

2. Backup wmm.xml.

Locate the wmm.xml file, and create a back up copy before changing any values. The file is located in the following locations:

Windows: <WebSphere_Portal_Server_Home>\wmm

Unix: <WebSphere_Portal_Server_Home>/wmm

Tip: The Member Manager configuration files within a clustered environment are moved from the `portal_server_root/wmm/` directory to the `was_profile_root/config/wmm` directory through a configuration task that uploads and replicates them to all the cluster nodes.

See section 4.5, “Enabling multiple LDAP directories in a cluster” on page 230 for more details on how to configure multiple LDAP directories in a clustered environment.

3. Enable realms in `wmm.xml`.

Edit the `wmm.xml` file to enable the multiple realms feature (horizontal partitioning). Find the XML attribute `horizontalPartitioning` and set the value to `true`.

Example 4-6 Enabling horizontal partitioning in `wmm.xml`

```
<wmm name="member manager"
      description="member manager"
      defaultRealmName="portal"
      horizontalPartitioning="true"
```

4. Add the federation repository configuration.

The Member Manager federation repository is designed to store federation information between different repositories.

It is only used when horizontal partitioning (realms) is enabled, which is why we have to add the configuration settings manually to the `wmm.xml` file.

- a. Locate the beginning of the `<repositories>` tag in the `wmm.xml` file.
- b. Add the stanza shown in Example 4-7, making sure you add the stanza below the `<repositories>` tag and just before the beginning of the `<lookAsideRepository>` tag.

Important: Make sure when you add the federation repository that you substitute the values for your own particular WMM database.

In our lab environment, WMM was stored in Cloudscape so we used the Cloudscape values for `databaseType` and `dataAccessManagerClassName`. Step 5 and Table 4-9 on page 218 list the values to use for the federation repository for databases other than Cloudscape.

Example 4-7 Adding the federation repository configuration for Cloudscape

```
<federationRepository
  name="wmmDBFederation"
  UUID="DB1"
  supportTransactions="true"
  adapterClassName="com.ibm.ws.wmm.db.DataBaseFederationAdapter"
  dataSourceName="jdbc/wpdbDS"
  databaseType="cloudscape"
  dataAccessManagerClassName=
    "com.ibm.ws.wmm.db.dao.cloudscape.WMMCloudscapeDao"/>
```

5. The attributes for the federation repository configuration are listed below.

Make sure you use the values appropriate for `databaseType`, `dataSourceName`, and `dataAccessManagerClassName` that are appropriate for the database you are using for WMM. These are listed in Table 4-9 on page 218.

- **name** The name of the repository. For the Member Manager federation repository, the default name is wmmDBFederation.
- **UUID** The universal unique identifier of the repository. You can use any name as long as it is unique in the wmm.xml file.
- **supportTransactions** Whether or not the repository supports transactions. Should be set to "true" for Member Manager federation repository.
- **adapterClassName** The implementation class name of the repository adapter. Use the value of com.ibm.ws.wmm.db.DataBaseFederationAdapter.
- **dataSourceName** The JNDI name of the data source that points to the Member Manager database. The default is jdbc/wmmDS. If you are using a different name, then you need to update this attribute. In Example 4-7 on page 217 we changed our database name to wpdbDS as this is the name of the Cloudscape database we used to store WMM.

Tip: You can confirm the datasource name for WMM by looking at the JDBC providers section in the WebSphere Application Server Administration console.

- **databaseType** The database type of the Member Manager database. In our lab environment we used Cloudscape; however, other possible values and their associated dataAccessManagerClassName are listed in Table 4-9.

Example 4-8 shows another federation repository stanza for when WMM is stored in Oracle. The values we changed are shown in bold.

Example 4-8 Adding the federation repository configuration for Oracle

```
<federationRepository
  name="wmmDBFederation"
  UUID="DB1"
  supportTransactions="true"
  adapterClassName="com.ibm.ws.wmm.db.DataBaseFederationAdapter"
  dataSourceName="wpdbDS_wmm"
  databaseType="oracle"
  dataAccessManagerClassName=
    "com.ibm.ws.wmm.db.dao.oracle.WMMOracleDao" />
```

Table 4-9 databaseType for the Member Manager database

Database	databaseType	dataAccessManagerClassName
DB2	db2	com.ibm.ws.wmm.db.dao.db2.WMMDB2Dao
Cloudscape	cloudscape	com.ibm.ws.wmm.db.dao.cloudscape.WMMCloudscapeDao
Oracle	oracle	com.ibm.ws.wmm.db.dao.oracle.WMMOracleDao
Microsoft SQL Server 2000	sqlserver	com.ibm.ws.wmm.db.dao.sqlserver.WMMSQLServerDao
DB2 on ZOS	db2_zos	com.ibm.ws.wmm.db.dao.db2zos.WMMDB2ZOSDao
DB2 on iSeries	db2_iseries	com.ibm.ws.wmm.db.dao.db2iseries.WMMDB2ISeriesDao

6. Add an additional <ldapRepository> section to the file.

This new section contains the configuration information for the additional LDAP directory you want to use with portal.

- a. Select the existing section that starts with <ldapRepository> and ends with </ldapRepository>. Choose File → Copy.
- b. Place the cursor in a new line just below the end of the existing </ldapRepository> entry, and choose File → Paste.

Edit the section you just pasted with values for the additional LDAP directory.

Change the values of the following XML attributes in the newly pasted section with the values most appropriate for your LDAP directory.

Example 4-9 shows some of the values we used for our second (Domino) LDAP directory. Leave all other immediate attributes of the <ldapRepository> tag set to their default values.

Example 4-9 Adding and additional LDAP repository in wmm.xml

```
<ldapRepository name="wmmLDAP"
  UUID="LDAP2"
  adapterClassName="com.ibm.ws.wmm.ldap.domino.Domino6LdapAdapterImpl"
  supportDynamicAttributes="false"
  configurationFile="wmmLDAPAttributes_DM.xml"
  wmmGenerateExtId="false"
  supportGetPersonByAccountName="true"
  profileRepositoryForGroups="LDAP2"
  supportTransactions="false"
  adminId="cn=LDAP Admin,o=ibm"
  adminPassword="password"
  ldapHost="dbdom7.cam.itso.ibm.com"
  ldapPort="389"
  ldapTimeOut="6000"
  ldapAuthentication="SIMPLE"
  ldapType="0"
  sslEnabled="false"
  sslTrustStore="C:\WebSphere\AppServer\etc\DummyServerTrustFile.jks"
  dirContextsMaxSize="20"
  dirContextsMinSize="5"
  dirContextTimeToLive="-1"
  cacheGroups="false"
  groupsCacheTimeOut="600"
  cacheAttributes="true"
  attributesCacheSize="2000"
  attributesCacheTimeOut="600"
  cacheNames="true"
  namesCacheSize="2000"
  namesCacheTimeOut="600">
```

Tip: Add an additional <ldapRepository> section for each additional LDAP directory you want to use with WebSphere Portal Server.

Important: In Example 4-9 you see that for the additional LDAP repository we defined the value for the configurationFile property as wmmLDAPAttributes_DM.xml.

In the <portal_server_root>/wmm directory you will find a configuration file for each supported LDAP directory. This file defines the attribute names and object classes for your particular LDAP directory; therefore, you need to specify the correct file for the additional LDAP repository you are adding to wmm.xml.

In our case the wmmLDAPAttributes_DM.xml is for Domino LDAP. See Table 4-13 on page 224 for a list of the various LDAP attribute configuration files.

Tip: There are different wmm.xml sample files for the various combinations of LDAP directories supported by WebSphere Portal, including when you configure an LDAP directory with a Lookaside configuration.

These sample files are located in the following locations:

Windows: <WebSphere_Portal_Server_Home>\wmm
Unix: <WebSphere_Portal_Server_Home>/wmm

These sample files are an invaluable guide when configuring wmm.xml for additional LDAP directories. See Table 4-12 on page 221 for a full list of sample wmm.xml files.

In our lab environment we used IBM Tivoli Directory Server and Domino LDAP, so we referenced the sample file wmm_LDAP_LA_DM.xml to assist us when editing wmm.xml.

Table 4-10 shows the attributes that need to be updated for the additional LDAP repository in the wmm.xml file.

Table 4-11 on page 221 also shows the values for the adapterClassName that need to be specified for the additional LDAP repository. Make sure that you use the correct adapterClassName for the type of LDAP directory that you are trying to add.

In our case, our second LDAP directory was a Domino LDAP directory, so we chose com.ibm.ws.wmm.ldap.domino.Domino6LdapAdapterImpl for our adapterClassName.

Table 4-10 Values to update in the wmm.xml file for an additional LDAP directory

XML attribute name	Value
UUID	The universal unique identifier of the repository. You can use any name as long as it is different from other repository's UUIDs. In our case as the primary LDAP directory was LDAP1 we added our additional LDAP directory as LDAP2.
adapterClassName	The implementation class name of the repository adapter (see Example 4-8 on page 218).
configurationFile	The relative or absolute path to the Member Manager LDAP attributes XML file. Member Manager comes with the template files for different LDAP directory servers (see Table 4-4 on page 209).

XML attribute name	Value
profileRepositoryForGroups	Defines the UUIDs of the repositories, which can contain members in this repository. Usually, this attribute includes the UUID of this repository itself. Multiple UUIDs should be separated by semicolon (;).
adminID	The Distinguished Name (DN) of the LDAP administrator that will be used to create the LDAP connection. This LDAP administrator should have enough access rights to perform defined operations.
adminPassword	<p>The password of the LDAP administrator. Although clear text password is accepted, it is highly recommended that the password be encrypted for security reason.</p> <p>From a command prompt, run the following task:</p> <pre>WPSconfig.{shlbat} wmm-encrypt -DPassword=password</pre> <p>Upon completion, examine the task output. Success is indicated with BUILD SUCCESSFUL and the encrypted password.</p>
ldaphost	The host name or IP address of the LDAP server.

Table 4-11 *adapterclassname* for additional LDAP repository

LDAP Server Type	adapterClassName
IBM Directory Server	com.ibm.ws.wmm.ldap.ibmdir.IBMDirectoryAdapterImpl
Active Directory Server 2000	com.ibm.ws.wmm.ldap.activedir.ActiveDirectoryAdapterImpl
Active Directory Server 2003	com.ibm.ws.wmm.ldap.activedir.ActiveDirectory2003AdapterImpl
Domino Directory Server	com.ibm.ws.wmm.ldap.domino.Domino6LdapAdapterImpl
Novell eDirectory Server	com.ibm.ws.wmm.ldap.novell.NovelleDirectoryAdapterImpl
SUN One Directory Server	com.ibm.ws.wmm.ldap.sunone.SunOneDirectoryAdapterImpl

Table 4-12 *Sample wmm.xml files for different LDAP directory combinations*

Sample wmm file	Description
wmm_LDAP_IDS_AD	Sample wmm.xml file for IBM Tivoli Directory Server and Active Directory.
wmm_LDAP_LA_AD	Sample wmm.xml for Active Directory with Lookaside configuration.
wmm_LDAP_LA_DM	Sample wmm.xml for Lotus Domino with Lookaside configuration.

Sample wmm file	Description
wmm_LDAP_LA_IDS	Sample wmm.xml for IBM Tivoli Directory Server with Lookaside configuration.
wmm_LDAP_LA_IDS_DM	Sample wmm.xml file for IBM Tivoli Directory Server and Lotus Domino with Lookaside configuration.
wmm_LDAP_LA_NDS	Sample wmm.xml file for Novell NDS with Lookaside configuration.
wmm_LDAP_LA_SO	Sample wmm.xml file for Sun One Directory Server with Lookaside configuration.

7. Edit the node maps configuration in the wmm.xml file. The node maps tell WebSphere Portal about the entry points for users and groups, and so forth, in the additional LDAP directory.

For Member Manager database repository, the member node and repository node are always the same. There is a default node map that maps Member Manager node *o=Default Organization* to Member Manager Database repository node *o=Default*.

For our additional LDAP directory, we specified that all users and groups exist under *o=ibm* as shown in Example 4-10.

Example 4-10 Node maps configuration for Domino LDAP

```

<nodeMaps>
  <nodeMap node="o=ibm" pluginNode="o=ibm"/>
</nodeMaps>

```

8. Configure LDAP supported entry types for the additional LDAP directory.

Locate the `<supportedLdapEntryTypes>` tag within the `<ldapRepository>` section for the additional LDAP directory (for example, the section you previously copy and pasted).

Edit the XML attributes with the correct values for your specific LDAP directory.

Example 4-11 contains the values for our additional Domino LDAP directory.

Example 4-11 Configuring supported entry types for an additional Domino LDAP directory

```

<supportedLdapEntryTypes>
  <supportedLdapEntryType name="Person"
    rdnAttrTypes="cn"
    objectClassesForRead="inetOrgPerson"
    objectClassesForWrite="inetOrgPerson"
    searchBases="o=ibm"/>
  <supportedLdapEntryType name="Group"
    rdnAttrTypes="cn"
    objectClassesForRead="groupOfNames;dominoGroup"
    objectClassesForWrite="groupOfNames;dominoGroup"
    searchBases="o=ibm"/>
  <supportedLdapEntryType name="Organization"
    rdnAttrTypes="o"
    objectClassesForRead="organization"
    objectClassesForWrite="organization"/>
  <supportedLdapEntryType name="OrganizationalUnit"

```

```
rdnAttrTypes="ou"
objectClassesForRead="organizationalUnit"
objectClassesForWrite="organizationalUnit"/>
</supportedLdapEntryTypes>
```

Notice how in Example 4-11 on page 222 we specified the attribute types and object classes for Person, Organization, and Organizational Unit as well as for our various search bases.

Tip: Use an LDAP browser type of tool (see Section 4.2, “Familiarizing yourself with the LDAP infrastructure” on page 192) to help you find the correct attribute and object class names for your particular LDAP directory.

Alternatively you can look at the “wmmLDAPAttributes_XXX.xml” sample file that corresponds to your particular LDAP directory.

9. Save the **wmm.xml** file.

Tip: Open wmm.xml with a browser to see if you made any typing or formatting errors. If there are no problems you should see the contents of the file. If you made a typing or formatting error (for example, missed a bracket) then you should see an error in the browser.

10. Backup then check the appropriate wmmLDAPAttributes_XXX.xml, and edit it if necessary.

For every supported LDAP Directory type there is a corresponding wmmLDAPAttributes template file that *may* need to be edited (see Table 4-13 on page 224). These template files are located in the following locations:

Windows: <WebSphere_Portal_Server_Home>\wmm
Unix: <WebSphere_Portal_Server_Home>/wmm

As our second LDAP server was a Domino LDAP directory, we used the wmmLDAPAttributes_DM.xml file; however, all the information regarding Domino LDAP attributes in the file was correct for Domino and so no changes were required.

Tip: This is not the case for all LDAP directories. For example, in earlier testing with IBM Tivoli Directory Server as an additional LDAP directory, we found that we had to edit the corresponding wmmLDAPAttributes_IDS.xml file because some of the attribute names contained in it were incorrect.

Specifically, for the wmmAttributeName=”groupMember” stanza, under the pluginAttributeName entry, we had to change the value from “member” to “uniqueMember”.

If you end up having to edit the file that corresponds to your LDAP directory, ensure that you edit the correct one (see Table 4-13 on page 224); however, prior to editing the attributes file, make sure you create a backup copy of it.

Table 4-13 Member Manager LDAP attribute files

LDAP Directory Server	Template file
IBM Tivoli Directory Server	wmmLDAPAttributes_IDS.xml
Active Directory	wmmLDAPAttributes_AD.xml
Domino	wmmLDAPAttributes_DM.xml
Novell eDirectory	wmmLDAPAttributes_NDS.xml
SUN One	wmmLDAPAttributes_SO.xml

11. Save the wmmLDAPAttributes_XXX.xml file if you made any changes.

12. Backup the wmmur.xml file.

This file contains the suffixes for users and groups and the realm mappings to the different LDAP directories that we previously defined in the wmm.xml file. The file is located in the following locations:

Windows: <WebSphere_Portal_Server_Home>\wmm

Unix: <WebSphere_Portal_Server_Home>/wmm

Before editing this file make sure you create a backup copy of it.

13. Edit the wmmur.xml file.

In the wmmur.xml file, we define which nodes will map to which realms. In our scenario we had the following three realms:

- The default realm *portal* that is automatically created for us when we enable security with realms (in our lab scenario, the default realm was associated with our primary IBM Tivoli Directory Server).
- The realm *employees*, which contains users in our IBM Tivoli Directory Server.
- The realm *customers*, which contains users in our Domino LDAP directory.

Add a second <node> definition tag for the Base DN of the additional LDAP directory to the default *portal* realm.

In Example 4-12 we added a node to the default *portal* realm that maps to the suffix of our Domino LDAP directory (o=ibm).

Example 4-12 Editing the default realm in the wmmur.xml file

```
<realm id="portal" delimiter="@" default="true">
  <node wmmnode="dc=ibm,dc=com"/>
  <node wmmnode="o=ibm"/>
</realm>
```

14. Add any additional realms in wmmur.xml.

Example 4-13 on page 225 shows the two additional realms we added to the wmmur.xml file.

- The realm *customer* maps to our Domino LDAP directory and we mapped the node mapped to the suffix of our Domino LDAP directory (O=ibm).

Notice that we did not specify a suffix for where our Domino groups are located. This is because Domino LDAP does not have a group suffix (See Section 4.4, “Domino LDAP and groups” on page 227 for more details).

- The realm *employees* maps to our IBM Tivoli Directory Server and we chose to map the node to the suffix where our employee users and groups are located.

Example 4-13 Adding additional realms in the wwmur.xml file

```
<realm id="customers" delimiter="@" default="false">
  <node wmmnode="o=ibm"/>
</realm>

<realm id="employees" delimiter="@" default="false">
  <node wmmnode="cn=employees,dc=users,dc=ibm,dc=com"/>
</realm>
```

15. Save wmmur.xml.

Tip: Open wmmur.xml with a browser to see if you made any typing or formatting errors. If there are no problems you should see the contents of the file. If you do have a typing or formatting error (for example, missed a bracket) then you should see an error in the browser.

16. Restart the WebSphere Portal Server.

You should now be able to log in with users from both the primary and secondary LDAP servers.

Note: If you plan to create users and groups via Portal, you should add the defaultParent entries to the wummur.xml file. This will allow WMM to know what suffixes to use when creating user and group objects in the LDAP directory.

This is described in more detail in the wmmur.xml section of the Infocenter document located at the following Web site.

http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp?topic=/com.ibm.wp.ent.doc/wpf/wmm_mltpl_realm.html

4.3.7 Creating two virtual portals to test multiple LDAP directories

As an additional test for our Multiple LDAP configuration, we also created two virtual portals: one called *employees* as shown in Figure 4-19 on page 226 and one called *customers* as shown in Figure 4-20 on page 226.

Virtual Portal Manager

Create New Virtual Portal
 Provide the information requested below and click OK to create a new virtual portal

Virtual portal title:
 Employees

Virtual portal description:
 Employees in our primary IDS LDAP Directory

URL Context:
 /wps/portal/employees

User realm:
 employees

Initial admin user group: (a valid user realm must be specified first)
 wpsadmins

Default theme:
 IBM

Warning: Creating a virtual portal may take several minutes.

OK Cancel

Figure 4-19 Creating a virtual portal for employees

Virtual Portal Manager

Create New Virtual Portal
 Provide the information requested below and click OK to create a new virtual portal

Virtual portal title:
 Customers

Virtual portal description:
 Customers in our secondary Domino LDAP

URL Context:
 /wps/portal/customers

User realm:
 customers

Initial admin user group: (a valid user realm must be specified first)
 DominoPortalUsers

Default theme:
 IBM

Warning: Creating a virtual portal may take several minutes.

OK Cancel

Figure 4-20 Creating a virtual portal for customers

We then assigned a group from the IBM Tivoli Directory Server to manage our employee virtual portal and a group from the Lotus Domino LDAP Directory to manage our customer virtual portal (see section 4.4, “Domino LDAP and groups” on page 227 for important information about using Domino LDAP and groups in a multiple LDAP environment).

We then created some content and customizations in each portal both as an employee and as a customer.

Finally we ensured that when customers (in the Domino LDAP Directory) logged into the customer virtual portal that they could only see the customer content and customizations and also when an employee logged in (from the IBM Tivoli Directory Server), that they could only see the employee content and customizations.

Note: The steps for creating a virtual portal are considered beyond the scope of this book. See the Infocenter document entitled “Multiple Virtual Portals” for more information about how to create and administer a virtual portal in WebSphere Portal Server Version 6.

Tip: When you create realms and associate them with virtual portals, the administrator of the default Virtual Portal (*portal*) cannot log in to the other virtual portals.

You can get around this by adding a super admin to the other realms you specified previously in the `wmmur.xml` file.

Add the DN of the super admin ID to the realms that you want the super admin to log into (where the super admin is not already part of another `wmmNode`).

```
<realm id="employees" delimiter="@" default="false">
  <node wmmnode="cn=employees,dc=users,dc=ibm,dc=com"/>
  <node wmmnode="uid=wpsadmin,cn=users,cn=cambridge,dc=ibm,dc=com"/>
</realm>
```

4.4 Domino LDAP and groups

Important: The following section details a limitation with groups in a multiple LDAP environment when using Domino LDAP.

At present, Domino LDAP has a limitation when used in a multiple LDAP directory environment with WebSphere Portal Server.

Specifically, when Domino LDAP is an additional LDAP directory you cannot search for (or use) groups that are stored in the Domino Directory unless you specify a group suffix entry. This is because groups in Domino LDAP exist under the Root DSE of the directory and are not part of the directory hierarchy. This is often referred to as flat groups or a flat group directory naming structure.

If you specify a group suffix of some kind (as we did in section 4.3.6, “Steps for adding an additional LDAP directory” on page 216), you will, however, not be able to search for (or use) any Domino groups within portal because none exist in the directory hierarchy.

In Figure 4-21 on page 228 you can see the Domino LDAP group `cn=wpsadmins` is located under the Root DSE rather than under the `o=ibm` organization suffix, where all our user entries are located.

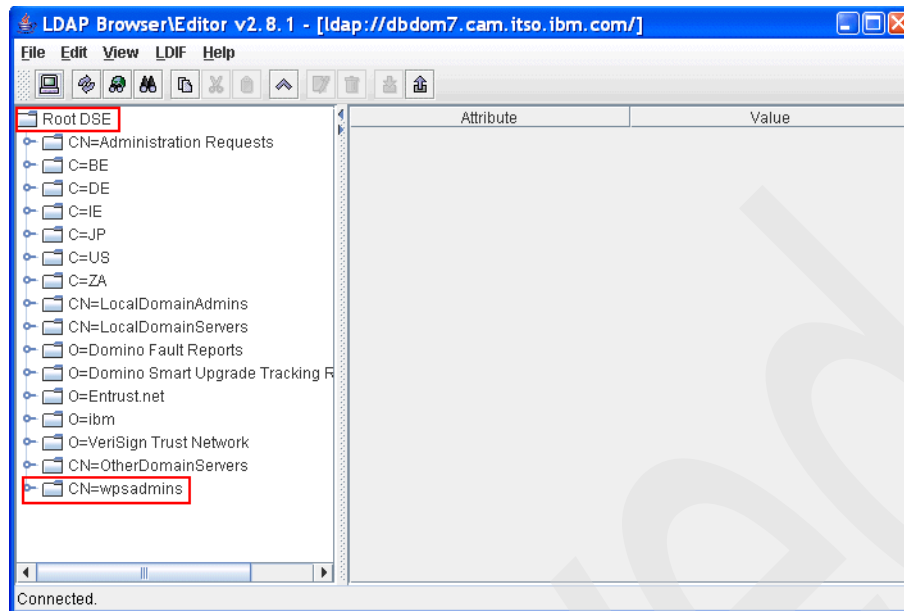


Figure 4-21 Domino LDAP groups under the root DSE

In Figure 4-22, you can see that under the Organizational Unit (o=ibm) there is no group container or groups.

Note: This limitation (for example, no group suffix in Domino LDAP), was raised with Lotus Development and should hopefully be addressed in a future release of Domino.

See the section entitled “Possible work arounds” on page 229 for some suggestions on how you can overcome this limitation in the meantime.

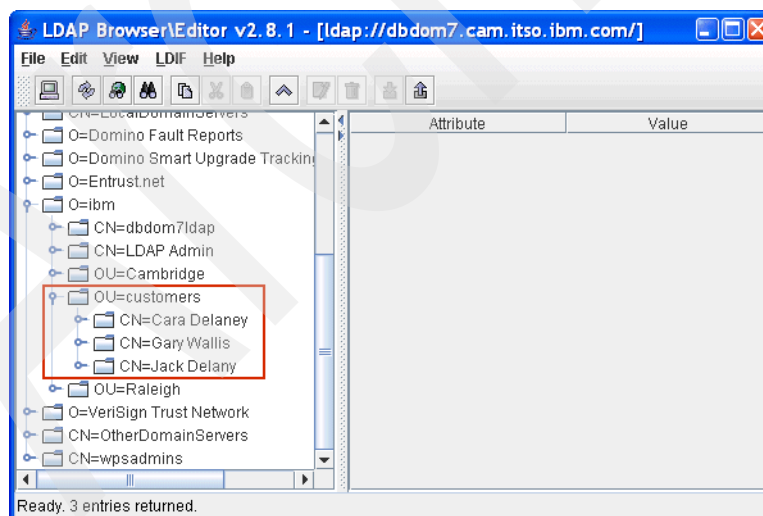


Figure 4-22 No group container or group suffix in Domino LDAP

The problem therefore arises because when specifying Domino LDAP in a multiple LDAP environment in the wmm.xml file you have to specify a null value for the Domino group suffix.

In Example 4-14 you can see that when we configured Domino LDAP in a multiple LDAP directory environment, we specified a group suffix and search base for groups of o=ibm; however, as shown in Figure 4-22 on page 228 there is no group container or groups under o=ibm.

Example 4-14 Specifying group values for Domino LDAP

```
<supportedLdapEntryTypes>
  <supportedLdapEntryType name="Person"
    rdnAttrTypes="cn"
    objectClassesForRead="inetOrgPerson"
    objectClassesForWrite="inetOrgPerson"
    searchBases="o=ibm"/>
  <supportedLdapEntryType name="Group"
    rdnAttrTypes="cn"
    objectClassesForRead="groupOfNames"
    objectClassesForWrite="groupOfNames"
    searchBases="o=ibm"/>
  <supportedLdapEntryType name="Organization"
    rdnAttrTypes="o"
    objectClassesForRead="organization"
    objectClassesForWrite="organization"/>
  <supportedLdapEntryType name="OrganizationalUnit"
    rdnAttrTypes="ou"
    objectClassesForRead="organizationalUnit"
    objectClassesForWrite="organizationalUnit"/>
</supportedLdapEntryTypes>
```

Possible work arounds

To overcome the current limitation of not having a container for groups in the Domino LDAP directory, there are a number of possible work arounds:

1. Edit your Domino groups to make them hierarchal, for example, change them so that they now reside in the directory hierarchy.

This is what we did in our lab testing. We created a group in the Domino LDAP directory called DominoPortalUsers and edited its name to include an /ibm as shown in Figure 4-23. This resulted in the group having a DN of cn=DominoPortalUsers, o=ibm and therefore being part of the directory hierarchy and thus accessible to WebSphere Portal Server.

Figure 4-23 Creating a hierarchical group in the Domino LDAP directory

Attention: Although editing the Domino group in this way was fine in a lab environment, you will probably find it is not a practical alternative in any reasonably sized production Domino environment. This is because groups in Domino are used for mail addressing, access control, security, and in agents and LotusScript code. Modifying them in this way, even via an automated process, may result in unforeseen problems with Domino security, mail, and applications.

Editing groups in this way is also not supported; therefore, this work around is not recommended for a production environment.

2. Create a secondary Domino Directory containing modified groups and make it accessible to portal via Domino's Directory Assistance feature.

Domino has the capability to use multiple Domino and LDAP directories through a feature known as *Directory Assistance*. It should therefore be possible to create a new additional Domino Directory and populate it with our Domino groups and then modify the groups to make them hierarchical (as shown in Figure 4-23 on page 229).

The modification of these groups and any subsequent update, additions, and deletions could also be automated via a LotusScript agent, or these groups could simply be dedicated to portal and managed independently of the primary Domino Directory.

Whilst there is some administrative overhead with this workaround it does ensure that your Domino environment is not impacted in anyway.

Note: See the Domino Administration Help for more information about configuring Directory Assistance.

3. It should be technically possible to modify the Domino LDAP schema to include a new "group hierarchical name" attribute in the group object class. This is not a trivial task and should not be undertaken lightly. Also, as with option 1, it is not supported so any such modification is at your own risk and may result in unforeseen problems.
4. Use a product such as IBM Tivoli Directory Integrator to consolidate your Domino groups in a virtual directory accessible to portal.

4.5 Enabling multiple LDAP directories in a cluster

The steps for configuring multiple LDAP directories in a clustered environment are essentially the same as those described in section 4.3.5, "Configuring an additional LDAP directory and enabling realms" on page 214; however, there is one key difference that needs to be taken in to consideration.

In a clustered environment, Member Manager files are not located in <portal_server_root>/wmm. They are actually moved to the <was_profile_root>/config/wmm directory when the security configuration task runs, and are then subsequently uploaded to the Deployment Manager, which in turn replicates them to all other cluster nodes.

Therefore, as we needed to manually edit the Member Manager files to configure multiple LDAP directories, we carried out the following steps:

1. Shut down the Deployment Manager.
2. Edit the relevant wmm files located in <was_profile_root>/config/wmm on the Deployment Manager.

3. Restart the Deployment Manager and carry out a full re-synchronization with all the cluster nodes.

Important: If the wmm.xml file references any files that are not in the following list, ensure that you copy the referenced files manually to the Deployment Manager and to each node in the cluster. For example, in our lab environment, we copied the wmmLDAPAttributes_DM.xml to the Deployment Manager and cluster nodes.

- ▶ wmm.xml
- ▶ wmmAttributes.dtd
- ▶ wmmAttributes.xml
- ▶ wmmAttributesDescription.xml
- ▶ wmmAttributesMap.dtd
- ▶ wmmDBAttributes.xml
- ▶ wmmDBAttributesDescription.xml
- ▶ wmmLAAttributes.xml
- ▶ wmmLDAPAttributes.xml
- ▶ wmmLDAPServerAttributes.xml
- ▶ wmmur.xml
- ▶ wmmWASAdmin.xml

Note: In this chapter we edited the WMM files directly to make it easier for you to follow our steps. However, we recommend that you do not edit these files directly; instead, leave the Deployment Manager running and *check out* the wmm files from the DMGR configuration repository before you make any changes to them.

You can do this by running the following command from the <portal_server_root>/config directory:

```
Windows: WPSconfig.bat check-out-wmm-cfg-files-from-dmgr
UNIX: ./WPSconfig.sh check-out-wmm-cfg-files-from-dmgr
```

After you check them out, you can then edit the files as per the instructions in Section 4.3.5, “Configuring an additional LDAP directory and enabling realms” on page 214.

The files can then be *checked in* to the Deployment Manager configuration repository by running the following command from the <portal_server_root>/config directory:

```
Windows: WPSconfig.bat check-in-wmm-cfg-files-to-dmgr
UNIX: ./WPSconfig.sh check-in-wmm-cfg-files-to-dmgr
```

Remember to also carry out a full re-synchronization with all the cluster nodes after you make the necessary changes.

Archived

Database domains

Following on from our discussion in Chapter 4, “Multiple LDAP directory support” on page 187, this chapter covers the following two other significant features in WebSphere Portal Version 6:

1. The ability to split our portal data amongst multiple database types.
2. The ability to share portal data between two separate Portal instances (for example, between two separate standalone portal environments or between two separate portal clusters.)

As with multiple LDAP directory support, the ability to split and share portal data into different database domains has a strong bearing on how we operate and deploy our Portal environment and in particular how we provide high availability.

We take a look at the advantages of splitting and sharing our portal data. We also provide some specific examples and techniques that are meant to highlight the new features and abilities built into the new database-transfer task and allowed by the new database domain architecture. After learning of these new features and techniques, you should be able to take advantage of them and design procedures that are sensible for your specific environment.

We conclude the chapter by walking you through a typical high-availability portal deployment scenario that utilizes database domains to split and share our portal data.

5.1 What are database domains?

Database domains are a feature in WebSphere Portal Version 6 that allow us to decentralize our portal data.

WebSphere Portal Version 5.1 did provide a mechanism to split portal data amongst different database schemas; however, the limitation was that these database schemas all had to exist in the same type of database instance (for example, all DB2 or all Oracle).

Also, and perhaps more importantly, stand alone portal nodes could not share a common set of a data without being clustered. This architectural limitation also extended to sharing a common set of data *between* clusters, for example, in Portal 5.1 two separate Portal clusters could not share the same set of data.

WebSphere Portal Version 6 overcomes these limitations by giving us the following:

- ▶ The ability to split portal data amongst different database instances *and* database types (later in this chapter for example we will use both DB2 and Oracle).
- ▶ More importantly, with WebSphere Portal Version 6 we can now also *share* portal data between portal servers and clusters. This is a significant enhancement as it gives us much greater flexibility in areas such as high availability, maintenance, upgrades, and backups.

So what is a database domain? Put simply it is the schema (a set of database objects, for example, tables, indices, and so forth) that defines a particular type of portal data.

For example, in Figure 5-1 on page 235 we can see that our portal data is stored in a database called WPSDB and that this database contains individual schemas for each of the types of portal data we want to store (for example, a JCR schema for PDM documents, a Release schema for our Release data and so on).

Note: The term database domains is often referred to as *Configuration split* or *Database split*; however, for the purposes of consistency with other IBM documentation throughout this book we will use the term database domains.

5.1.1 Terminology

Before we take a look at database domains in more depth, it is important to define what we mean by some of the terminology we use.

This is important because different database manufacturers use different terms, and often terms are used interchangeably, which can lead to confusion.

The terminology we will be using in this chapter is as follows:

- ▶ **Physical Server.** This is the hardware on which the database software is installed.
- ▶ **Instance (or Database Instance).** is an actual physical installation of the database software. For example our physical server might have an instance of DB2 installed on it. With most commercial databases it is possible to have multiple database instances installed on the same physical server (see Figure 5-1 on page 235).
- ▶ **Databases.** These are the structures that store our portal data. Databases are associated with a particular database instance.
- ▶ **Schemas.** A collection of related database elements (such as tables, indices and so forth) that define our data and provide it with structure. When we talk about Domains, we are in

essence talking about the schema and its related structures. We use the terms Schemas and Domains as interchangeable terms throughout this chapter.

Figure 5-1 provides a graphical representation of these terms. It depicts a typical example when using DB2 and shows the following two databases:

- ▶ WPSDB, which contains the schemas for all our portal data
- ▶ TOOLSDB, which is the DB2 Tools database that has schemas of its own

Note: Figure 5-1 shows a typical small-to-medium business (SMB) type scenario whereby all our portal data is stored in a single database. For larger organizations and environments it is not uncommon to have multiple portal databases (for example, separate databases for the JCR and Community schemas).

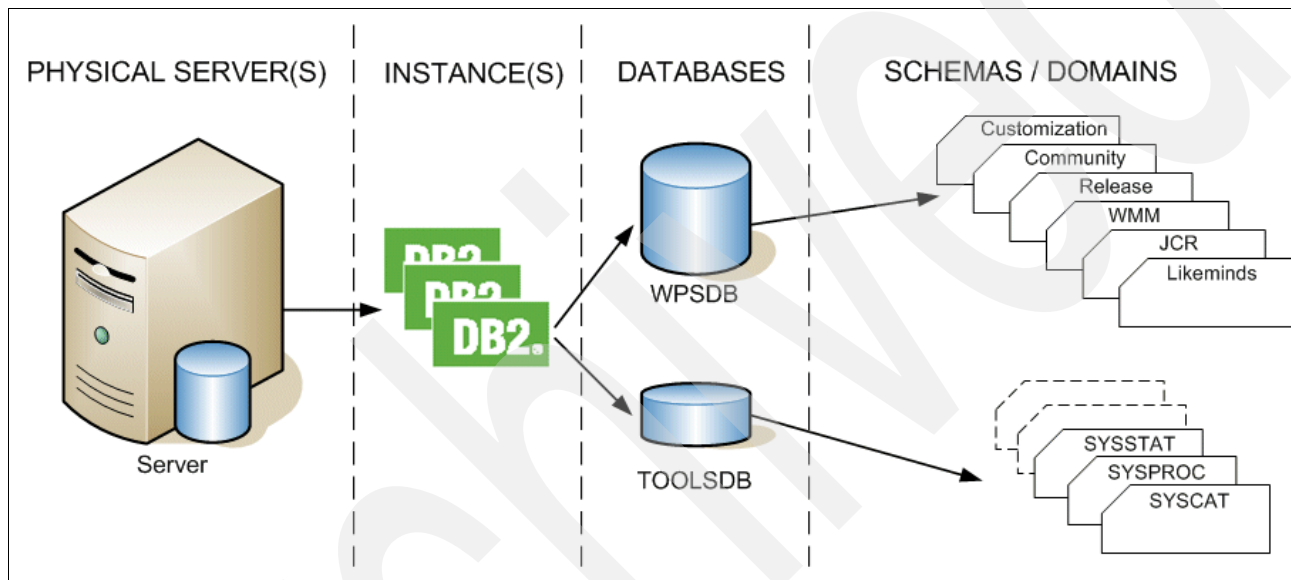


Figure 5-1 Defining our terminology

Figure 5-2 on page 236 shows how our terminology relates to DB2. The figure is taken from the DB2 Control Center and highlights each of the terms we have defined in this section.

Important: Figure 5-2 on page 236 represents a typical DB2 schema architecture AFTER an initial database-transfer was run to move the Portal data from Cloudscape to DB2. When looking through the database schemas you may notice there is no schema for the WebSphere Member Manager (WMM) domain.

This is because the `wpconfig_dbdomain.properties` file does NOT contain a property for `wmm.DbSchema` like provided for the other domains (i.e., `release`, etc.). Therefore, the database-transfer code creates the WMM objects in the schema defined by the `wmm.DbUser` value, which is also the value used to log in to the database and in this case was `DB2ADMIN`. For example in Figure 5-2 on page 236 our WMM data is stored in the schema called `DB2ADMIN`.

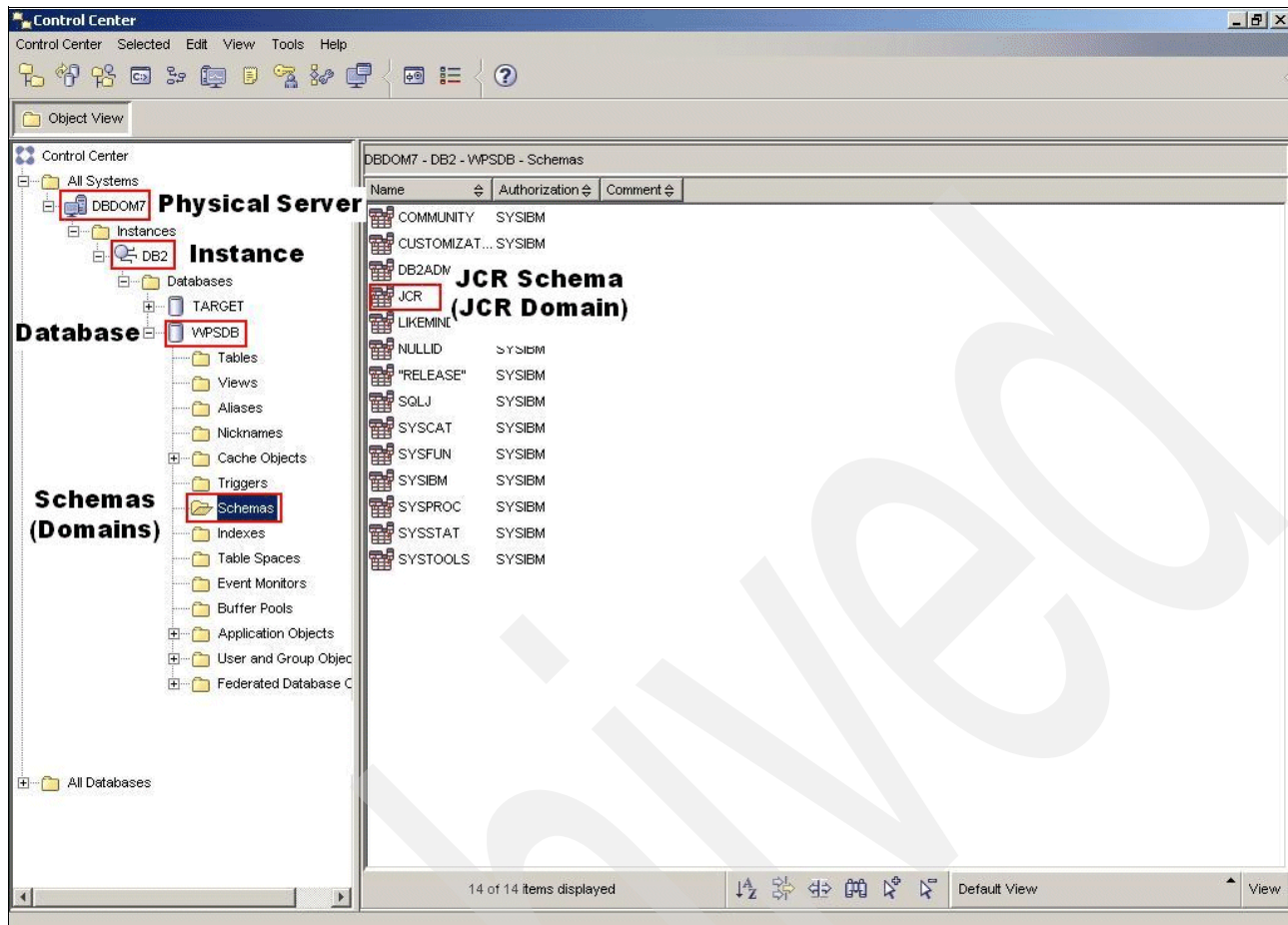


Figure 5-2 How our terminology maps to DB2

5.1.2 How is Portal data organized?

Portal data is broken down into four logical groups. Each group has different uses, characteristics, and different rates of access and growth. As a result there is some data that is particularly suited to being split and shared amongst our portal nodes, whilst there is other data where it is less appropriate (and in some cases not possible).

Portal data is therefore organized as follows:

- ▶ **Configuration data.** This is data that defines the portal server setup, such as database connections, object factories, and deployment descriptors.

This type of data is usually constant over the uptime of a portal server node and is typically kept in property files on the portal server's hard disk and protected by file system security. With WebSphere Portal Version 6 much of the configuration data is now managed by WebSphere Application Server Network Deployment (WAS ND). Configuration data therefore cannot be split or shared (see Table 5-1 on page 237).
- ▶ **Release data.** This type of data defines all portal resource definitions, rules, and rights. These are often designed externally (for example, in a development or staging environment) and then brought into the portal by a staging process using tools such as XMLAccess (see Chapter 7, "Configuration management: moving between environments" on page 337). Typical release data includes elements such as Page Hierarchy, available Portlets and Themes, Templates, and Credential Slots.

These resources are typically not modified during production and need administrative rights to do so.

Release data is protected by portal access control and contains only data, not code. In a setup that consists of multiple lines of production, data must exist in one *copy* of the release per cluster (see Figure 5-3 on page 239 for an example of this).

Web Content Management managed content is also considered release data, since the content is authored in an environment outside the production environment (typically) and syndicated into production as part of a content workflow. The JCR is used to store WCM content, as well as Theme Policies and Personalization Rules and is thus included under the release category.

As a result, Release data cannot be split or shared and administrators must make sure that the content of the release databases is consistent across the different lines (in particular after modification of the release data on the production system).

- **Customization.** This is typically only associated with a particular user, but it is data that can be shared amongst portal server nodes (see Table 5-1). Some examples are PortletData (for example, for a mail portlet it could be the user's mail server and mail file name) or implicitly derived pages. Since this data applies to a single user only, portal access control protection is greatly simplified.

In an environment that consists of multiple lines of production (see Figure 5-3 on page 239), customization data is kept in a database that is *shared* across the lines of production. Therefore the data is automatically in sync across the lines of production, and no matter which line of the production the user logs into, their customizations are still available to them.

- **Community.** These are modified during production. This type of data includes items such as shared documents or application resources. Users and groups are allowed to modify or delete shared resources.

Table 5-1 Which portal data can be shared

Database Schema/Domain	Shareable	Storage method
Configuration	No	Filesystem
Release	No **	Database
Customization	Yes	Database
Community	Yes	Database
JCR	No **	Database
Feedback	Yes	Database
Likeminds	Yes	Database
WMM	Yes	Database

Important: ** Release data can be shared amongst portal servers in the same cluster.

Note: There are two additional databases called Designer and Sync that are included for use with a number of the *tech preview* features in WebSphere Portal Version 6. These databases are beyond the scope of this book.

5.2 Advantages of database domains

Now that we defined what we mean by database domains and discussed how portal data is organized, we can now discuss in more detail the advantages of splitting and sharing our portal data.

As discussed in the previous section, separation of portal data allows us to store each type of data in its own database domain. Then these domains can be transferred to different database servers and database instances and shared amongst multiple portal nodes.

Sharing data amongst Portal servers

Figure 5-3 on page 239 shows a typical database domains scenario. In this scenario we have multiple lines of production; however, the community and customization data is shared between them. Each production line has a copy of the same Release data because the Release data cannot be shared.

The advantages of splitting and sharing database domains in this way are as follows:

- ▶ As the community and customization data are shared between both lines of production, it is synchronized by default, thus eliminating the need to conduct any database replication between the production lines.
- ▶ A user can be routed to either line of production and will still have access to their user specific customizations.
- ▶ Upgrades and maintenance can be carried out in parallel to normal operations and are much simpler because the release data is logically separated from the community and customization data. Therefore updates to the release data do not affect the other shared database domains. This allows some clusters or servers to remain in production during any upgrade or maintenance process.

Updating Portal

Now that we have the ability to share data amongst portal nodes, the process of updating our portal environment is much simpler and much less risky.

Figure 5-3 on page 239 illustrates how we could conduct an upgrade as follows:

- ▶ Production line B is taken out of service temporarily to test new release data.
- ▶ Users are routed to line A.
- ▶ Testers or administrators can access production line B with the new Release data but still use the same user data as production line A.
- ▶ If the test fails, it is possible to revert to the previous release data.
- ▶ If the test is successful, production line B is put back into service and then users must be re-routed from line A to line B. Then production line A is taken out of service to have its Release data updated.

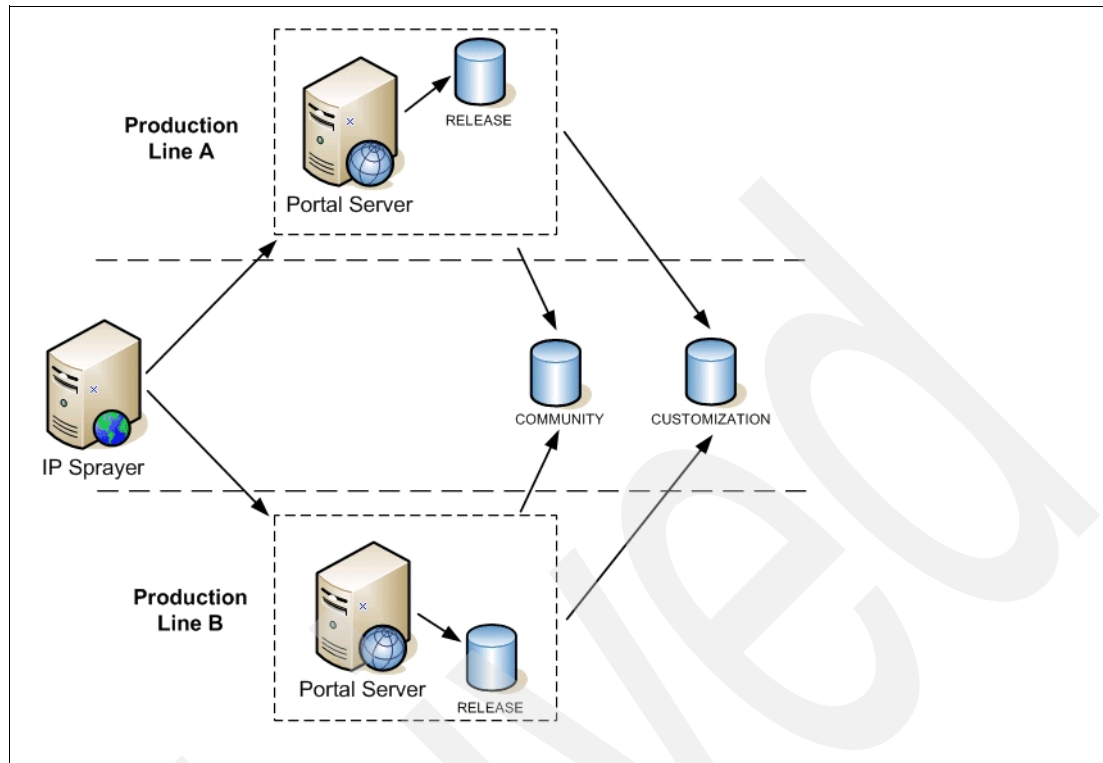


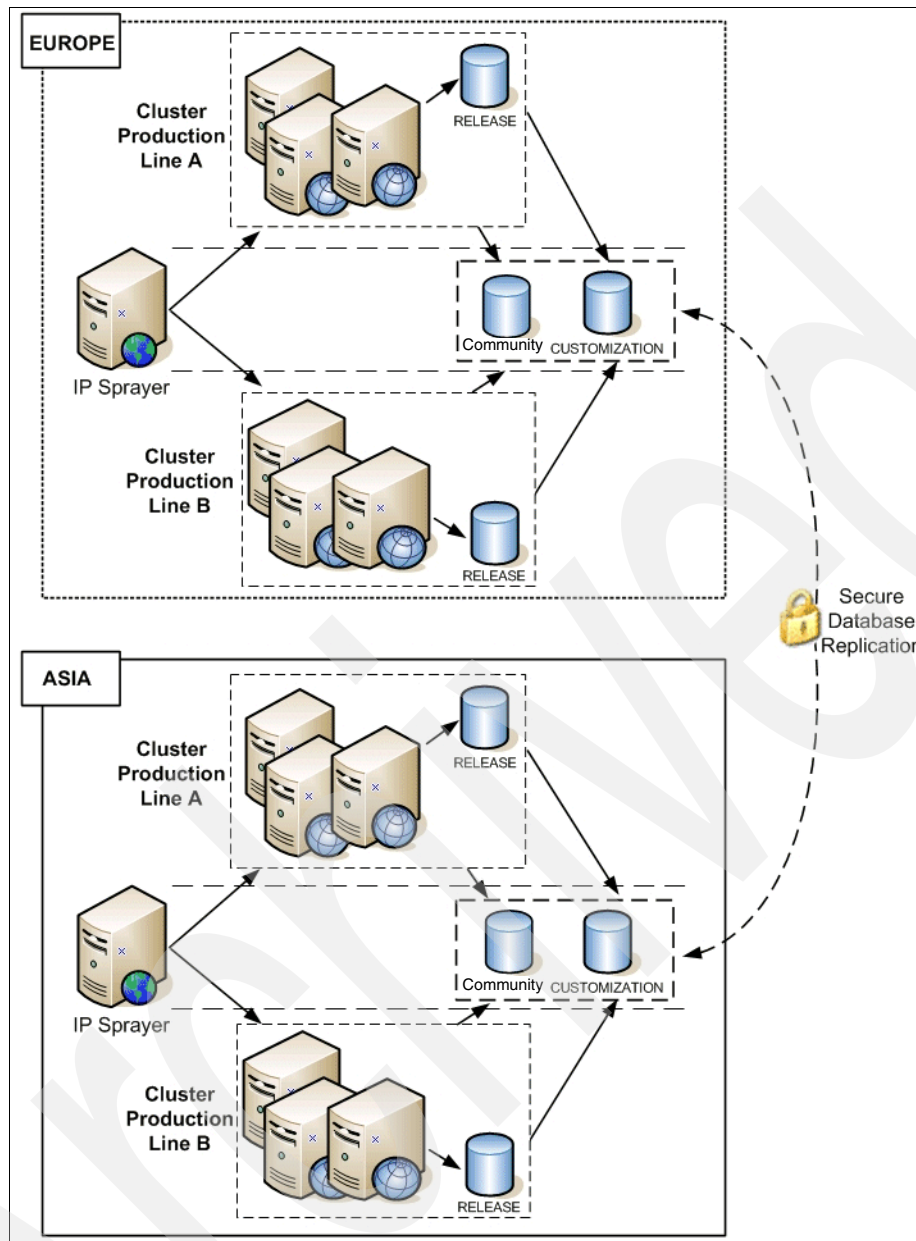
Figure 5-3 Multiple lines of production with shared community & customization domains

This scenario can be extended further to cross site or geographical boundaries.

Example 5-1 on page 240 shows how multiple lines of production can exist in different continents (each of the production lines A and B are both clustered environments). In this example, the customization and community data is shared between the production line clusters and replicated securely (using database replication features) between Europe and Asia through a secure link. And each geographical location shares common data between its own lines of production.

We can then take down an entire geography for maintenance and still provide 24x7 operations, by redirecting users to the geography that is still up and running.

It also allows for users in any geography to log in anywhere and see the same community and customization data.



Example 5-1 Global deployment

Backing up user data

One of the challenges of operating a 24x7 environment is the act of backing up portal data without disrupting users. In Figure 5-4 on page 241 we have two clusters that are sharing the same Customization data. In this scenario the backup is carried out while the user data stays live. During the backup, only the database entries for individual users are locked.

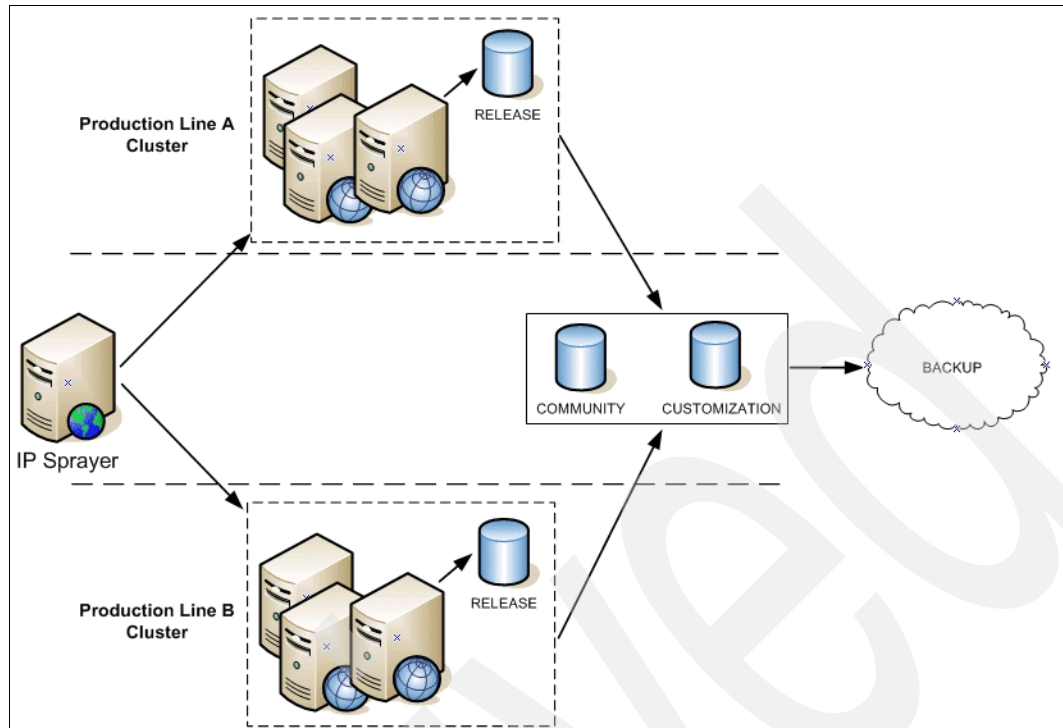


Figure 5-4 Backing up user data

5.3 Examples of moving and sharing database domains

In order to best demonstrate how to split and share database domains, we will now look at four examples from our own testing. These examples help to demonstrate how database domains are a powerful new feature that was added to the Portal 6 database architecture. These examples show how to move or share a database domain. Each example leads in to the next one, successively introducing greater high availability than its predecessor.

These examples are not intended to be an end-to-end step guide on how to configure a particular type of Portal environment (for example, two separate clusters sharing the same database domains). Rather, these examples are designed to inform and educate you on several techniques that you can use when moving or sharing database domains.

By the end of this chapter, the examples should give the reader sufficient knowledge to be able to build a new (or re-architect an existing) Portal infrastructure to take advantage of database domains.

To further assist you, the techniques learned from these examples are described in a typical end-to-end production scenario in Section 5.8, “Using database domains in a production environment” on page 313.

Note: The particular database domains used in the following examples such as jcr, release, customization, and so on, are used for illustrative purposes only. They were not selected in the examples for any specific reason; however, where a database domain was moved or shared for a specific reason, an appropriate explanation is provided.

5.3.1 Properties files associated with database domains

The following four properties files need to be modified when moving or sharing database domains. In each of the various examples discussed in this chapter, you will see that we edited the following files in this order:

1. **wpconfig.properties.** We need to ensure that this file contains the correct WAS and Portal administrator user IDs and passwords prior to any database domain activity.
2. **wpconfig_dbtype.properties.** This file needs to be edited next to include all the relevant database connection information (such as database drivers) that are needed by Portal before we attempt to move or share any database domains.
3. **wpconfig_sourceDB.properties.** This file needs to be edited to define where our database domains currently reside. This file is not updated as part of any database transfer or database connect task, so we must manually edit it to tell Portal where any database domains that we may wish to move or share *currently* are.
4. **wpconfig_dbdomain.properties.** This file needs to be edited to define where we want to move or connect our database domains *to*.

Keep these properties files, and the order in which they will be used, in mind when carrying out any database domain activities.

Table 5-2 lists these properties files and summarizes when they need to be modified when working with database domains.

Tip: We highly recommend that you make a copy of any properties file before you edit them.

Table 5-2 Properties files used for database domains

Property file	Modified when...
wpconfig.properties	Transferring or moving to another database
wpconfig_dbtype.properties	Connecting portal to an additional database server that is different from the one we are currently using (for example, we are using Cloudscape or DB2 and are now moving some or all Domains to Oracle)
wpconfig_sourceDB.properties	Transferring to a new database server
wpconfig_dbdomain.properties	Connecting or transferring to a new database server

Note: Throughout this chapter we make extensive use of the configuration wizard.

Important: When configuring WebSphere Portal, there are multiple different methods that can be employed.

In the following examples, we make extensive use of editing the appropriate properties files and then executing the required configuration task via the configuration wizard.

Although this method may differ from how you configure Portal (for example, you may edit the properties files and use the command line to configure Portal), we specifically chose this method so that if, for some reason, the configuration task, we would not have to manually re-enter values in the configuration wizard panels (and therefore we would also minimize the risks of introducing typos or incorrect information).

Also, we found that representing this configuration information with screen shots from the configuration wizard provides the reader with much greater clarity than simply reproducing the contents of properties files.

Using the configuration wizard in these examples also allows us to walk you through the configuration process step-by-step and helps to further re-iterate just what properties and values are being used.

For every example however, we provided the command line alternative in case you do not want to use the configuration wizard.

5.4 Example 1, moving database domains

To introduce the practical aspects of working with database domains, we begin with a very simple example. In this example we began with a single-portal node using a remote DB2 server and instance.

The example begins after the initial Portal server installation occurred with Cloudscape, and the subsequent database transfer from Cloudscape to DB2 took place (using the configuration wizard or the database-transfer task).

On the remote DB2 instance we created a database called WPSDB that contained all our database domains (schemas) as shown in Figure 5-5 on page 244.

The WPSDB database contained the following database domains:

- ▶ Customization
- ▶ Release
- ▶ Community
- ▶ WMM (The domain is actually named after the administrative user for the database. In our case it was called db2admin.)
- ▶ JCR
- ▶ Likeminds

Note: For simplicity, we opted not to use the Feedback Domain, and the *Designer* and *Sync* domains were not included as they are used by some of the tech preview features in WebSphere Portal Version 6. They are also beyond the scope of this IBM Redpaper.

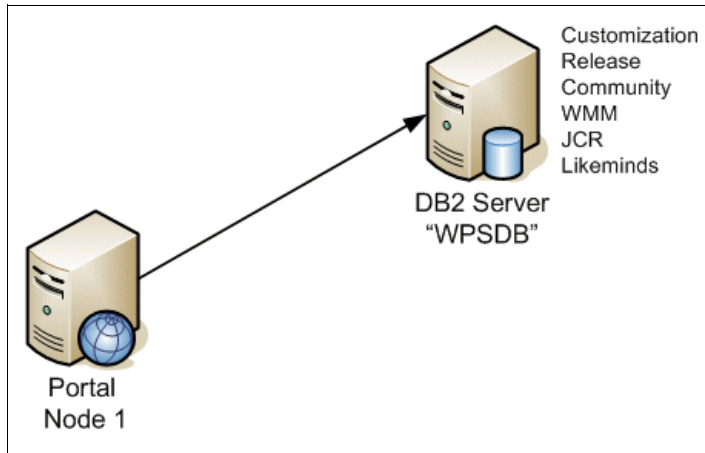


Figure 5-5 All database domains stored in one database called WPSDB

We aimed to split the portal data by moving the JCR and Likeminds domains to a second DB2 server and instance. We then stored them in a database we created there earlier called TARGET.

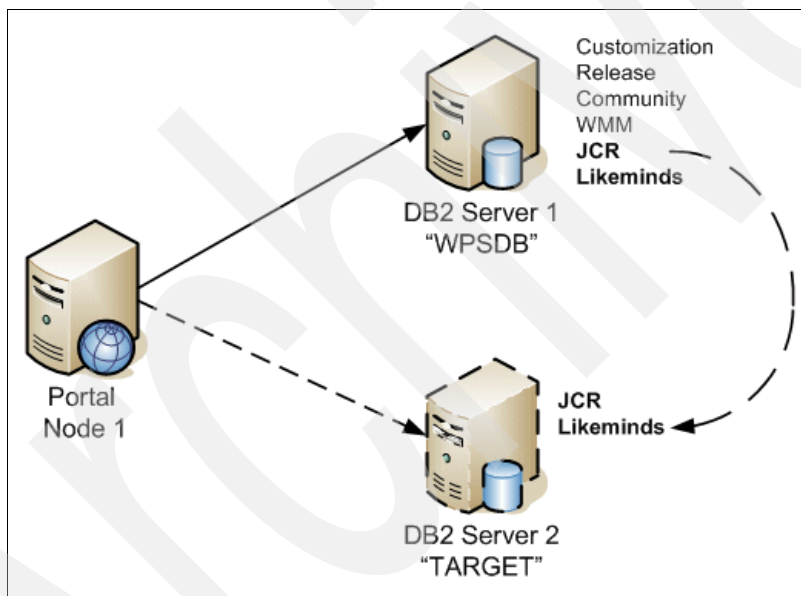


Figure 5-6 Moving JCR & Likeminds to a second DB2 Server and database called TARGET

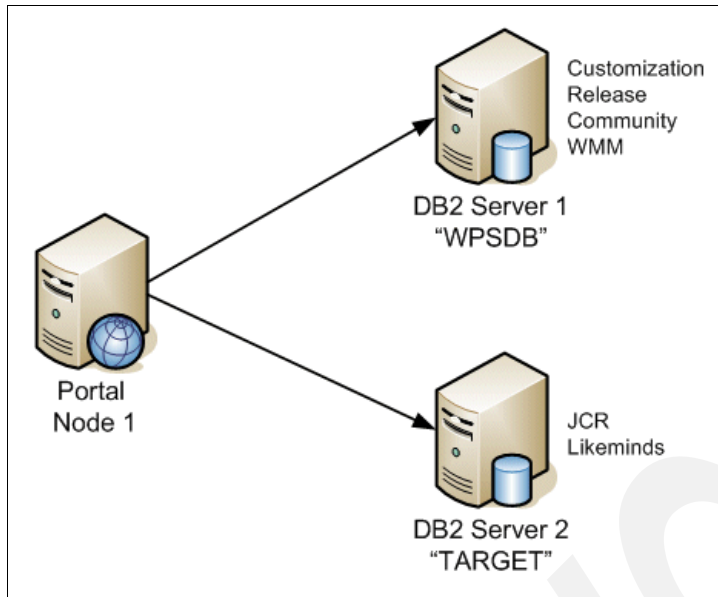


Figure 5-7 Our portal environment after splitting the JCR and Likeminds domains

Figure 5-7 shows how our portal environment looked after the JCR and Likeminds domains were moved to another database server.

Steps for moving database domains between the same database type

Note: Although the following steps describe how to split domains from one DB2 server to another DB2 server, they are still applicable for other types of database servers such as Oracle or SQL Server (see the Infocenter for the values that are specific for your database).

We chose to use all DB2 servers in our first example for clarity. As you will see later on in this chapter, our other examples use both DB2 and Oracle.

1. We built a new DB2 database server (Server 2) and created a new database on it called TARGET. See Chapter 3, “Different deployment scenarios: building clustered environments” on page 51 for details on how to create a new DB2 database suitable for use with Portal.
2. We then cataloged the databases on DB Server 2 on the Portal server node and ensured we could connect to the new TARGET database on DB2 Server 2. See Chapter 3, “Different deployment scenarios: building clustered environments” on page 51 for details on how to do this for DB2.

Tip: In an existing clustered environment you would need to perform step 2. on the primary node and all other Portal nodes in the cluster.

3. Edit `wpconfig_dbtype.properties`. We needed to edit this properties file in order to provide the configuration wizard (or database-transfer task) with the location of the drivers that will allow it to connect to the destination DB2 database server/database.

In our case we only needed to specify the details for DB2 (as shown in **bold** in Example 5-2 on page 246).

Important: Regardless of which operating system your portal server is using, make sure any file system paths use the forward slash /.

Example 5-2 Editing wpconfig_dbtype.properties for DB2

```
#####
# DB2 Properties
#####

# DbDriver: The name of class SqlProcessor will use to import SQL files
db2.DbDriver=COM.ibm.db2.jdbc.app.DB2Driver

# DbLibrary: The directory and name of the zip/jar file containing JDBC driver
class
# Please use the system specific file separator names, e.g. for windows semicolon
and for unix colon.
db2.DbLibrary=C:/IBM/SQLLIB/java/db2java.zip

# JdbcProviderName: The name of jdbc provider to be used
db2.JdbcProviderName=wpdbJDBC_db2

#####
# END: DB2 Properties
#####
```

4. Edit the wpconfig_sourceDB.properties file. When you first install Portal, all database domains are stored in Cloudscape. After our Portal was installed we then transferred all the database domains to a DB2 server (DB2 Server 1 as shown in Figure 5-5 on page 244).

The wpconfig_sourceDB.properties file, however, is not updated as part of the database transfer process and therefore still lists all our database domains as being in the original Cloudscape database.

Therefore, in order to move any database domains (for example, in our case JCR and Likeminds), we need to update wpconfig_sourceDB.properties to reflect the fact that these database domains are now in DB2.

When we then run either the configuration wizard or the WPCConfig database-transfer task to move our database domains, the updated values in the wpconfig_sourceDB.properties file is read and the wizard or task then knows the current location of the database domains we want to move.

Example 5-3 on page 247 shows the properties in the wpconfig_sourceDB.properties file associated with the JCR database domain. The values we changed are as follows (and are shown in **bold** in Example 5-3 on page 247):

- **source.jcr.DbType**. Specifies the type of database where the Domain is stored. In our case DB2.
- **source.jcr.DbName**. Specifies the name of the database where the Domain is stored. In our case WPSDB (see Figure 5-5 on page 244).
- **source.jcr.DbSchema**. Specifies the name of the database schema/domain. In our case jcr.
- **source.jcr.DbUrl**. Specifies where the database is and how we connect to it.
- **source.jcr.DbUser**. Specifies the user name to use when connecting to the database where our database domain is stored.

- **source.jcr.DbPassword.** Specifies the password to use for the previously specified user name.

Example 5-3 Editing wpconfig_sourceDB.properties for the JCR and Likeminds Domains

```
#####
# JCR Database Properties
#####

# DbType: The type of database to be used for WebSphere Portal JCR domain
source.jcr.DbType=db2

# DbName: The name of the WebSphere Portal JCR domain database
# Note: This value should also appear as the database element in DbUrl
# Note: Non-Windows platforms when using DB2 only. This value is the TCPIP Alias
# for the database.
source.jcr.DbName=wpsdb

# DbSchema: The WebSphere Portal JCR domain database schema name
# Follow the documentation of the target database management system
# in order to define a valid schema name as restrictions apply for
# some database management systems.
source.jcr.DbSchema=jcr

# DbUrl: The wp JCR domain database URL
source.jcr.DbUrl=jdbc:db2:wpsdb

# DbUser: The database administrator user ID
source.jcr.DbUser=db2admin

# DbPassword: The database administrator password
source.jcr.DbPassword=password

#####
# END: JCR Database Properties
#####4
```

5. Repeat step 4 for any other database domains you want to move (in our case we repeated step 4 for the Likeminds Domain).

6. Edit wpconfig_dbdomain.properties. Now that we specified the *current* location of our database domains, we need to specify where we want to move them *to*.

In our case, we wanted to move them to a different DB2 server and into a database called TARGET.

The values we changed in the wpconfig_dbdomain.properties are as follows (and are shown in **bold** in Example 5-4 on page 248):

- **jcr.DbType.** Specifies the type of database where the Domain will be stored. In our case DB2.
- **jcr.DbName.** Specifies the name of the database where the Domain is to be moved to. In our case TARGET (see Figure 5-7 on page 245).
- **jcr.DbSchema.** Specifies the name of the schema/domain.
- **jcr.DbUrl.** Specifies how to connect to the new database to where we want to move our database domains.

- **jcr.DbUser.** Specifies the user name to use when connecting to the database where we want to move our database domain to.
- **jcr.DbPassword.** Specifies the password to use for the previously specified user name.

Tip: Repeat step 6 for each of the database domains you want to move.

Example 5-4 Editing wpconfig_dbdomain.properties for JCR and Likeminds

```
#####
# JCR Database Properties
#####

# DbType: The type of database to be used for WebSphere Portal JCR domain
jcr.DbType=db2

# DbName: The name of the WebSphere Portal JCR domain database
# Note: This value should also appear as the database element in DbUrl
# Note: Non-Windows platforms when using DB2 only. This value is the TCPIP Alias
# for the database.
jcr.DbName=target

#

DbSchema: The LikeMinds database schema name
jcr.DbSchema=jcr

# DbUrl: The wp JCR domain database URL
jcr.DbUrl=jdbc:db2:target

# DbUser: The database administrator user ID
jcr.DbUser=db2admin

# DbPassword: The database administrator password
jcr.DbPassword=password

#####
# LikeMinds Database Properties
#####

# DbType: The type of database to be used for LikeMinds
likeminds.DbType=db2

# DbName: The name of the WebSphere Portal likeminds domain database
# Note: This value should also appear as the database element in DbUrl
# Note: Non-Windows platforms when using DB2 only. This value is the TCPIP Alias
# for the database.
likeminds.DbName=target

# DbSchema: The LikeMinds database schema name
likeminds.DbSchema=likeminds
```

```
# DbUrl: The LikeMinds database URL
likeminds.DbUrl=jdbc:db2:target

# DbUser: The database administrator user ID
# **Required for DB2, DB2 for z/OS and OS/390 **
likeminds.DbUser=db2admin

# DbPassword: The database administrator password
likeminds.DbPassword=password
```

Repeat step 6 for any other database domains you may wish to move.

7. Prior to step 8, which involves starting the database transfer process, we verified the contents of our destination database called “TARGET”.

As can be seen in Figure 5-8, there are no schemas (database domains) in the TARGET database other than the default DB2 schema objects.

We performed this step again, after the database transfer process successfully completed, in order to verify that our database domains were moved (see Figure 5-21 on page 259).

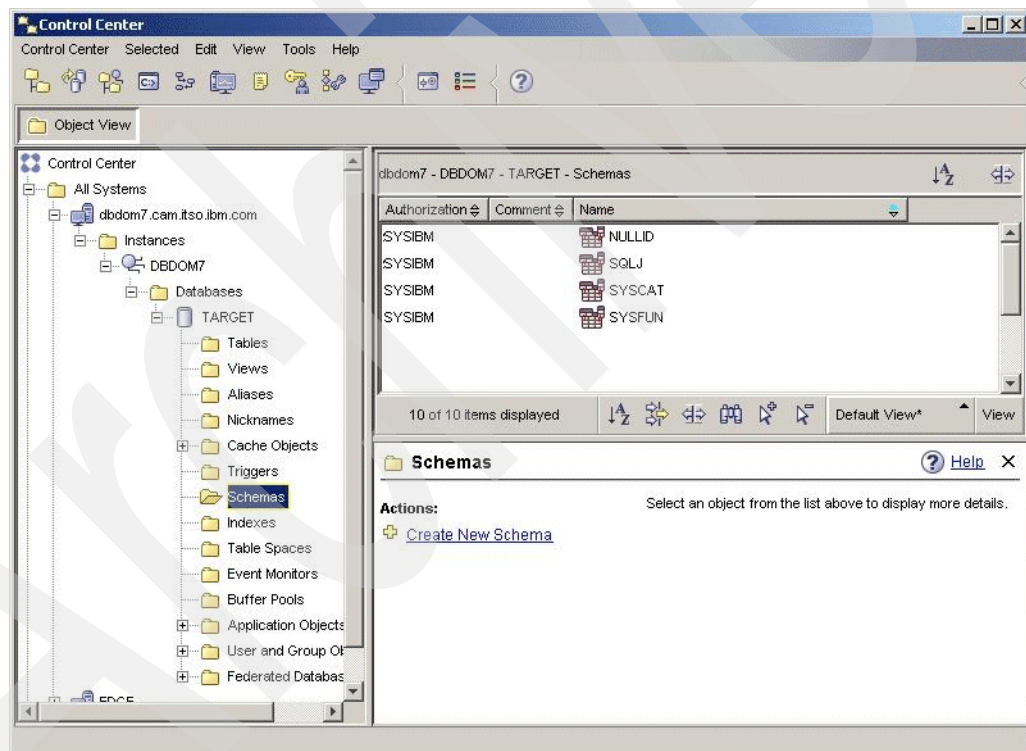


Figure 5-8 Remote database before we moved data

8. Run the configuration wizard or WPCong database-transfer task. For the database domains we want to move, we just specified where they currently are and where we want to move them to. As a result, we are now ready to transfer the database domains (in our case the JCR and Likeminds) by running the configuration wizard or WPCong database-transfer task.

The configuration wizard can be found in <portal_server_root>/config/wizard. Run the configwizard.bat file to start the wizard.

Alternatively you can run the following command from <portal_server_root>/config/ directory (substituting domain1, domain2, and so forth, with the names of your database domains).

Windows: WPSconfig database-transfer -DTransferDomainList=domain1,etc

Unix: ./WPSconfig database-transfer -DTransferDomainList=domain1,etc

Tip: The domain names that can be specified when using the command line are release, customization, community, jcr, wmm, feedback, and likeminds. Following is an example:

```
./WPSconfig database-transfer -DTransferDomainList=jcr, likeminds
```

See the Infocenter document entitled “Transferring between databases manually” for more information about the command line options.

Tip: When moving database domains and using DB2, if possible try to ensure that the new (destination) database server does not contain any databases with the same name as the original (source) server. This is because if you have the same database name on 2 different DB2 servers, then the DB2 client on the Portal server has to be cataloged using aliases because you cannot catalog the same database name twice on the DB2 client to point to 2 different DB2 servers.

You should be able to work around duplicate database names by using database aliases on the DB2 client; however, a discussion on database aliases is beyond the scope of this book.

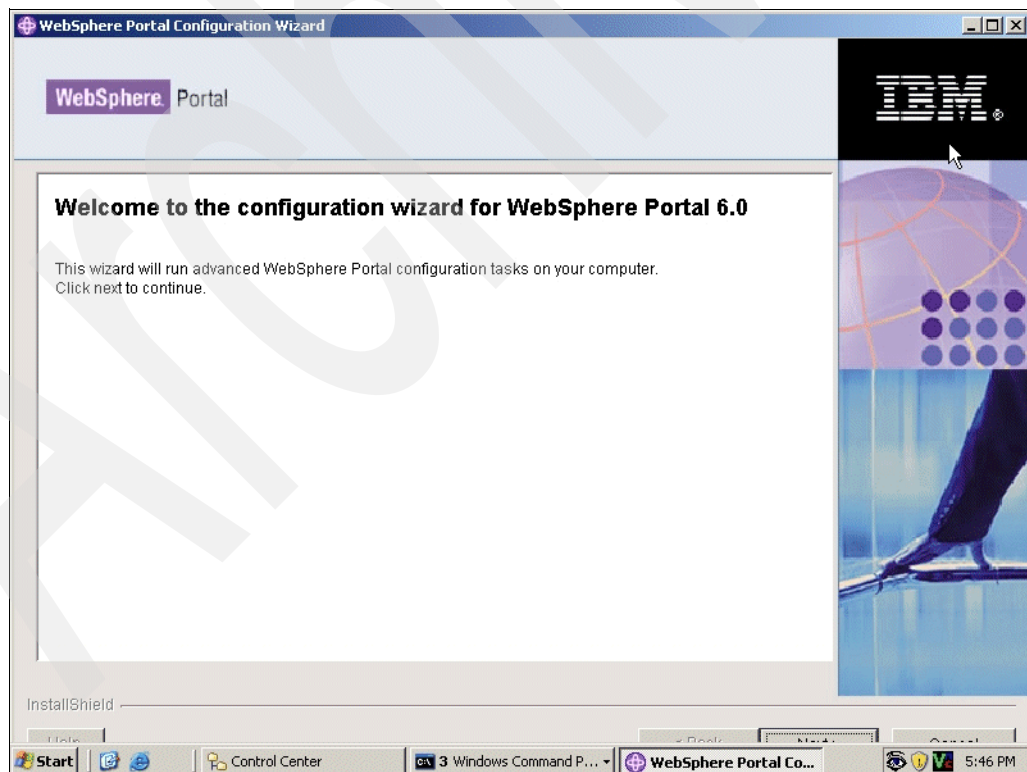


Figure 5-9 Running the configuration wizard

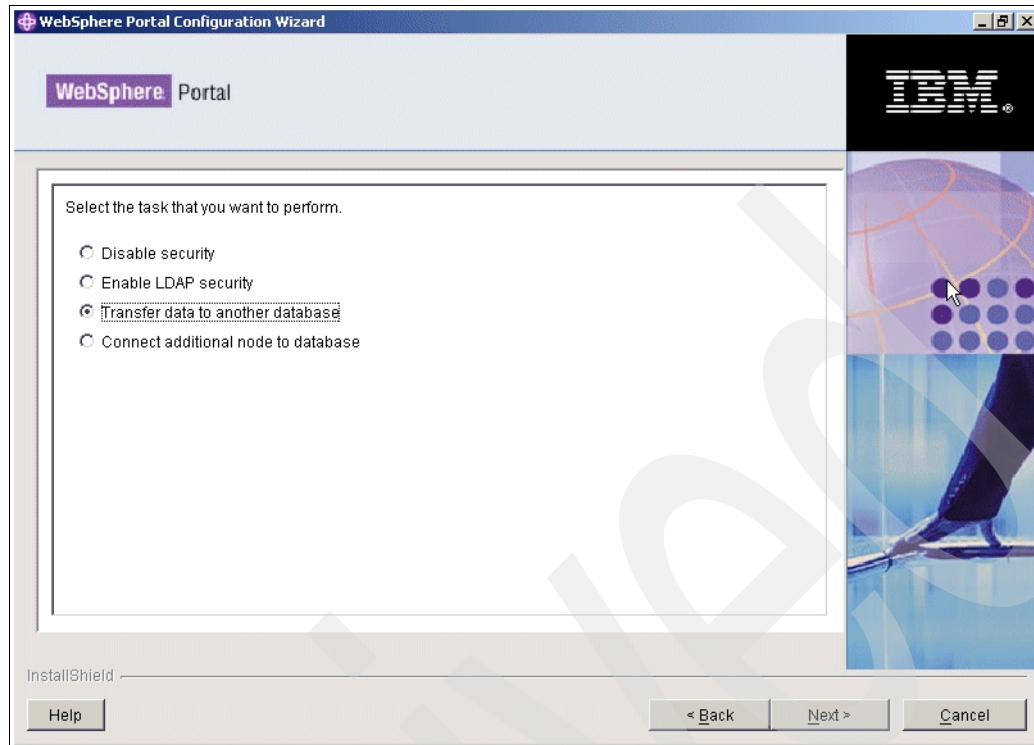


Figure 5-10 Select the “Transfer data to another database” option

Select the option **Transfer data to another database**, and then specify the WebSphere Application Server Administrator user name and password as shown in Figure 5-11.

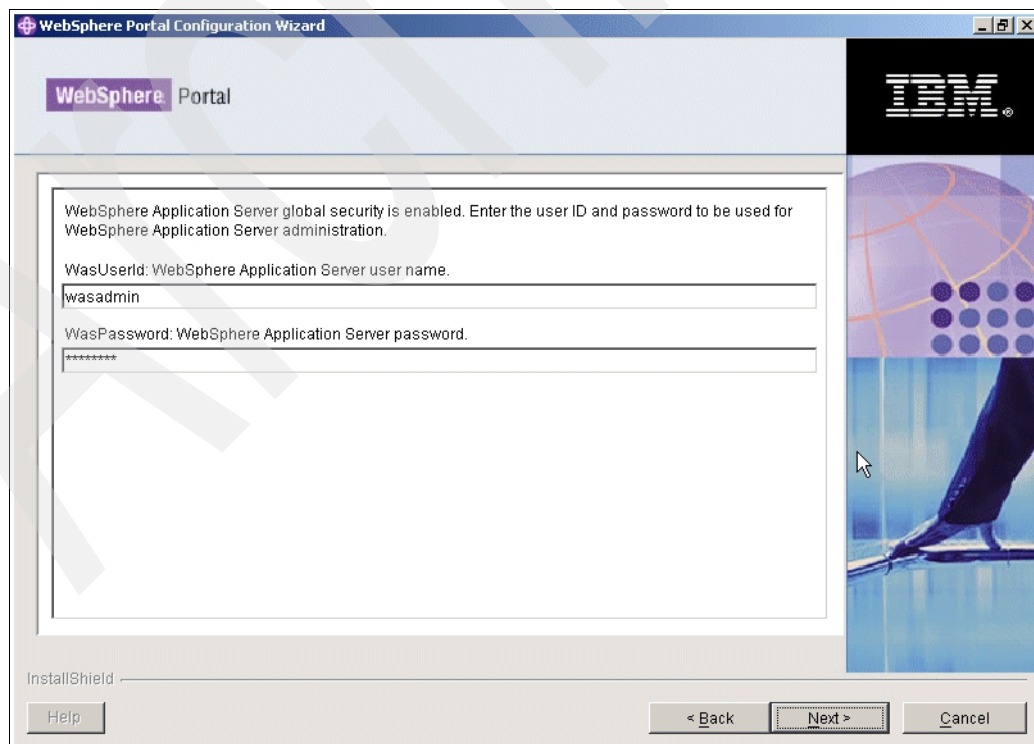


Figure 5-11 Specify the WebSphere Application Server Administrator user name and password

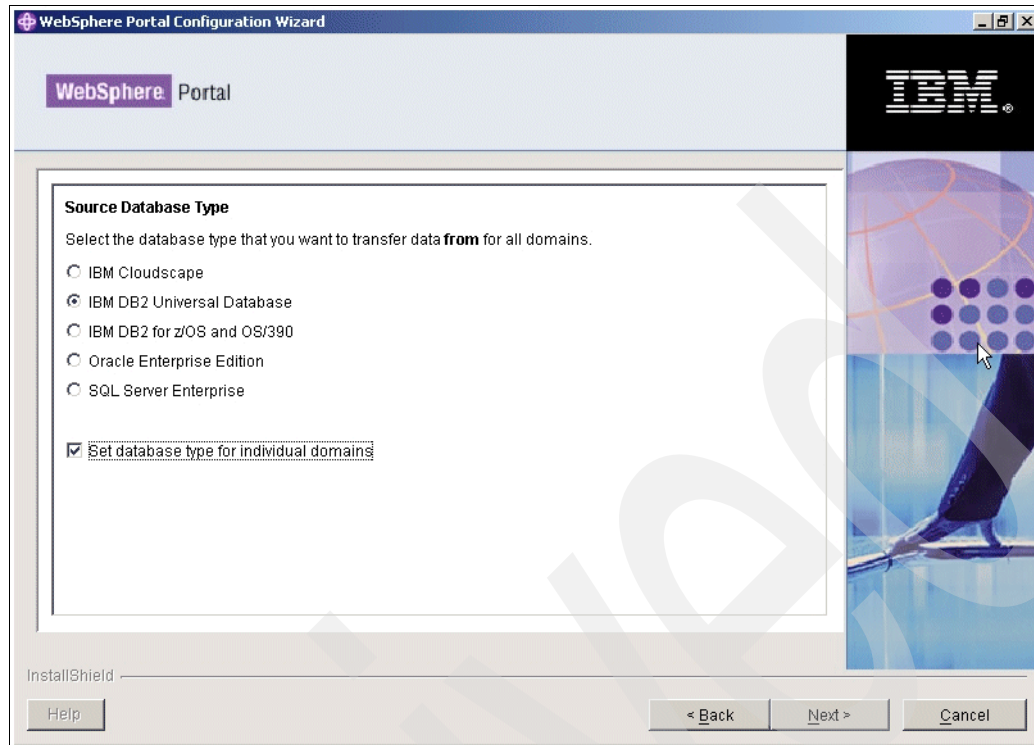


Figure 5-12 Select the source database type & individual database domains

Select **Set database type for individual domains** as shown in Figure 5-12. This option allows us to specify which individual database domains we want to transfer to another database server.

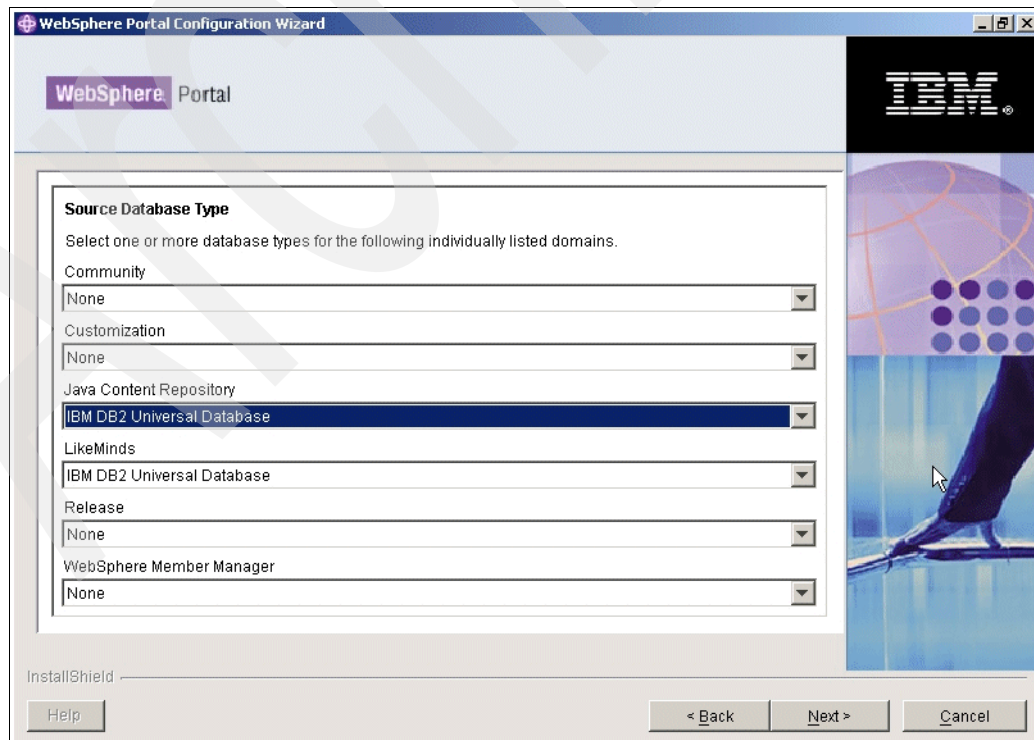


Figure 5-13 Specify the source database type

In Figure 5-13 on page 252, we selected the database domains we want to move. In our case we only specified JCR and Likeminds as the source database domains to be moved.

Tip: Leave any database domains that you do **not** want to move, set to None.

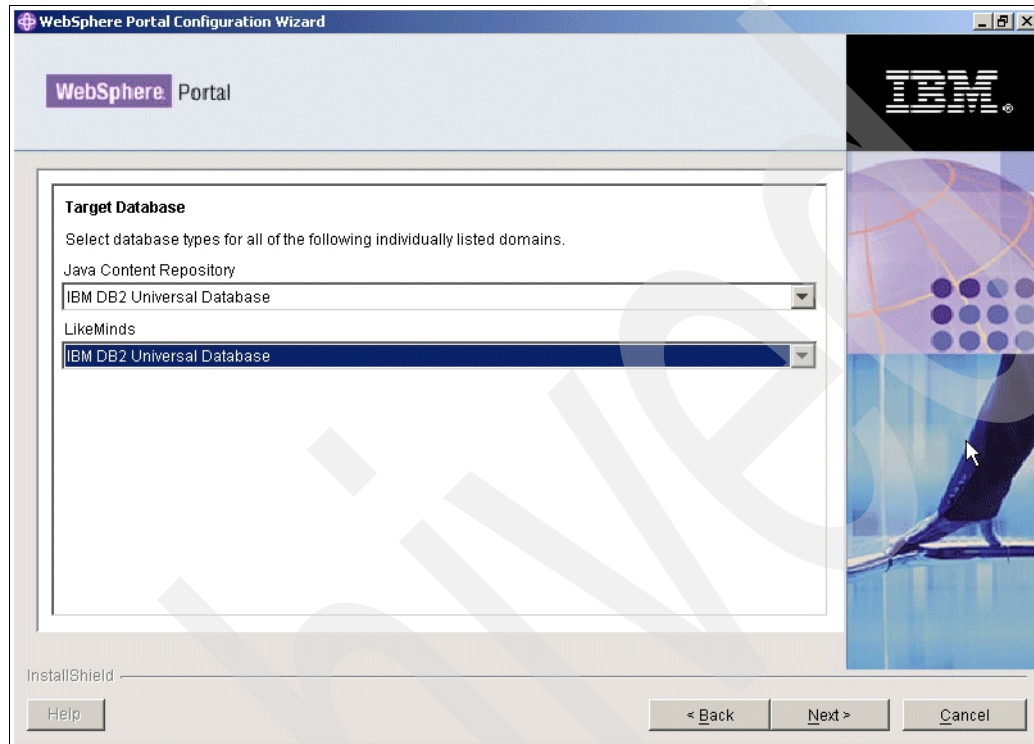


Figure 5-14 Specify target databases

In Figure 5-14, we specified which database type will store the database domains we want to move. In our case both database domains will be moved to a DB2 database server.

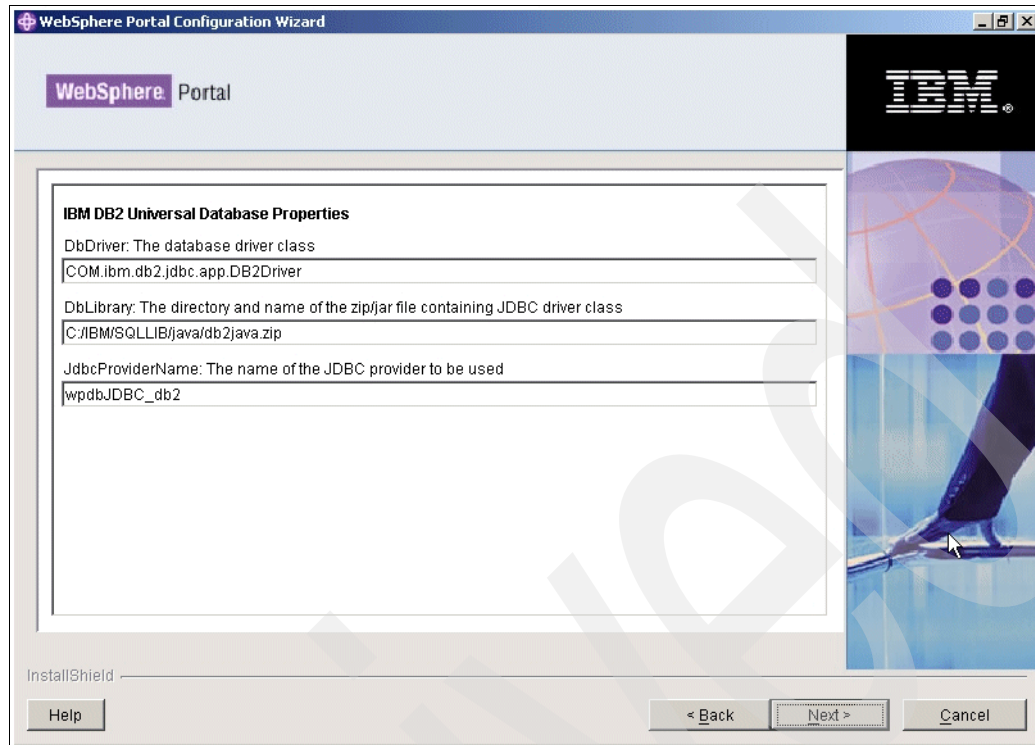


Figure 5-15 Specify DB2 database properties

Figure 5-15 shows the DB2 properties that we specified in the `wpconfig_dbtype.properties` file in step 3 (see Example 5-2 on page 246). These values define how portal server will connect to DB2 database servers.

Ensure that these values are correct before proceeding; otherwise, the database transfer process will not be successful.

The screenshot shows the 'WebSphere Portal Configuration Wizard' window. The title bar reads 'WebSphere Portal Configuration Wizard'. The main content area is titled 'Java Content Repository Database Domain Properties'. It is divided into two sections: 'Source Database' and 'Target Database'. The 'Source Database' section includes fields for 'DbName: Database name' (containing 'wpsdb'), 'DbSchema: Database schema' (containing 'jcr'), 'DbUser: Database administrator user name' (containing 'db2admin'), 'DbPassword: Database administrator password' (containing '*****'), and 'DbUri: JDBC URL' (containing 'jdbc:db2:wpsdb'). The 'Target Database' section includes fields for 'DbName: Database name' (containing 'target') and 'DbSchema: Database schema'. At the bottom, there is an 'InstallShield' progress bar, a 'Help' button, and navigation buttons '< Back', 'Next >', and 'Cancel'. The right side of the window features a vertical banner with the IBM logo and a graphic of a globe and a hand.

Figure 5-16 Specify a user name and password for the database domains we want to move

Figure 5-16 shows the details for the first database domain we want to move (in our case JCR). Provide a valid user name and password to connect to the target database or database server; otherwise, the database transfer process will fail.

In our case this was the user **db2admin** and the password was **password**.

Tip: After the configuration wizard runs, for security reasons, any password values you may have specified in the properties files it reads from are replaced with the generic text string of `ReplaceWithYourDbAdminPwd`.

The screenshot shows the 'WebSphere Portal Configuration Wizard' window. The title bar reads 'WebSphere Portal Configuration Wizard'. The main window has a header with the 'WebSphere Portal' logo and the IBM logo. The central area is titled 'LikeMinds Database Domain Properties'. It contains two sections: 'Source Database' and 'Target Database'. The 'Source Database' section has fields for 'DbSchema: Database schema' (value: 'likeminds'), 'DbUser: Database administrator user name' (value: 'db2admin'), 'DbPassword: Database administrator password' (value: '*****'), and 'DbUrl: JDBC URL' (value: 'jdbc:db2:wpsdb'). The 'Target Database' section has fields for 'DbName: Database name' (value: 'target'), 'DbSchema: Database schema' (value: 'likeminds'), and 'DataSourceName: Datasource to be used for WebSphere Portal'. At the bottom, there is an 'InstallShield' label, a 'Help' button, and three navigation buttons: '< Back', 'Next >', and 'Cancel'. On the right side of the window, there is a decorative graphic featuring a globe and a hand pointing at a grid of dots.

Figure 5-17 Specify a user name and password for the database domains we wish to move

Figure 5-17 shows the details for the next database domain we want to move (in our case Likeminds).

Provide a valid user name and password to connect to the target database or database server; otherwise, the database transfer process will not be successful.

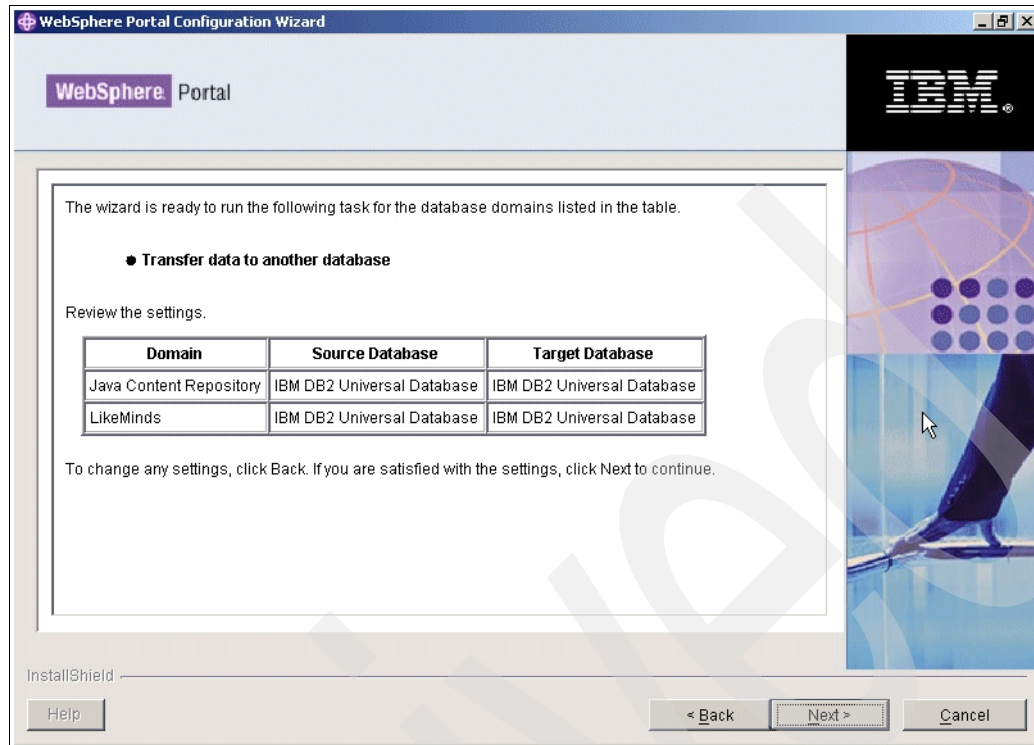


Figure 5-18 Summary window

Figure 5-18 shows a summary of the database domains that will be transferred to the new database server. It lists where they currently are and where they will be moved to. Click Next to begin the transfer process.

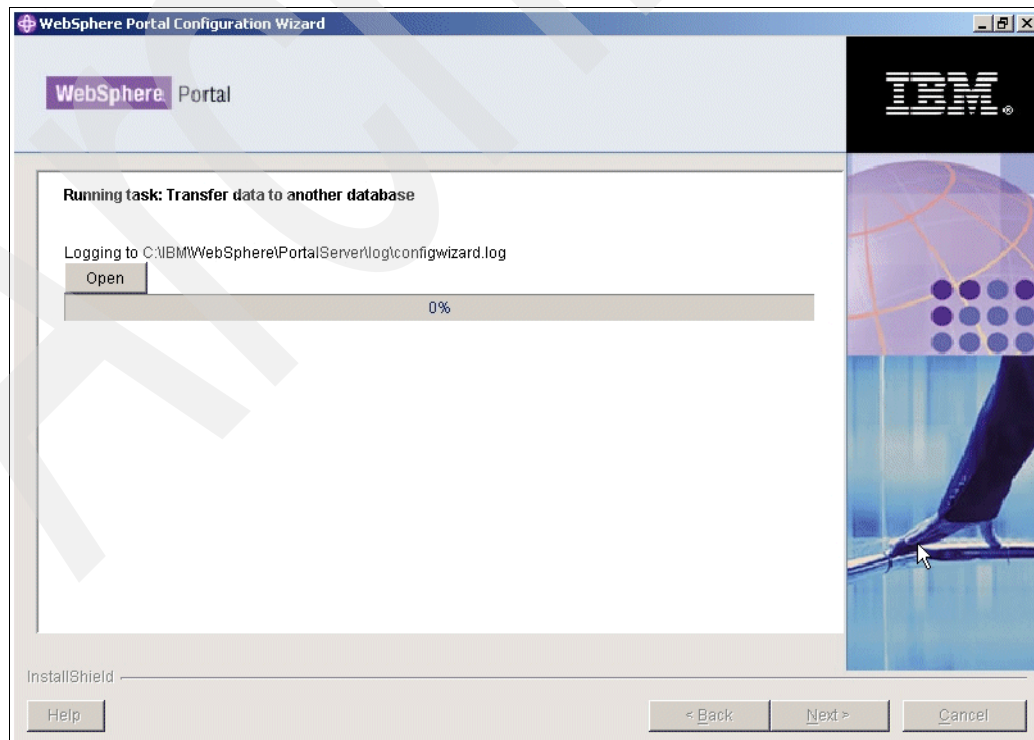


Figure 5-19 Transfer process

Figure 5-19 on page 257 shows the progress of the database transfer process.

Clicking the Open button shows you the content of the configwizard.txt file, thus allowing you to check the status of the transfer process.

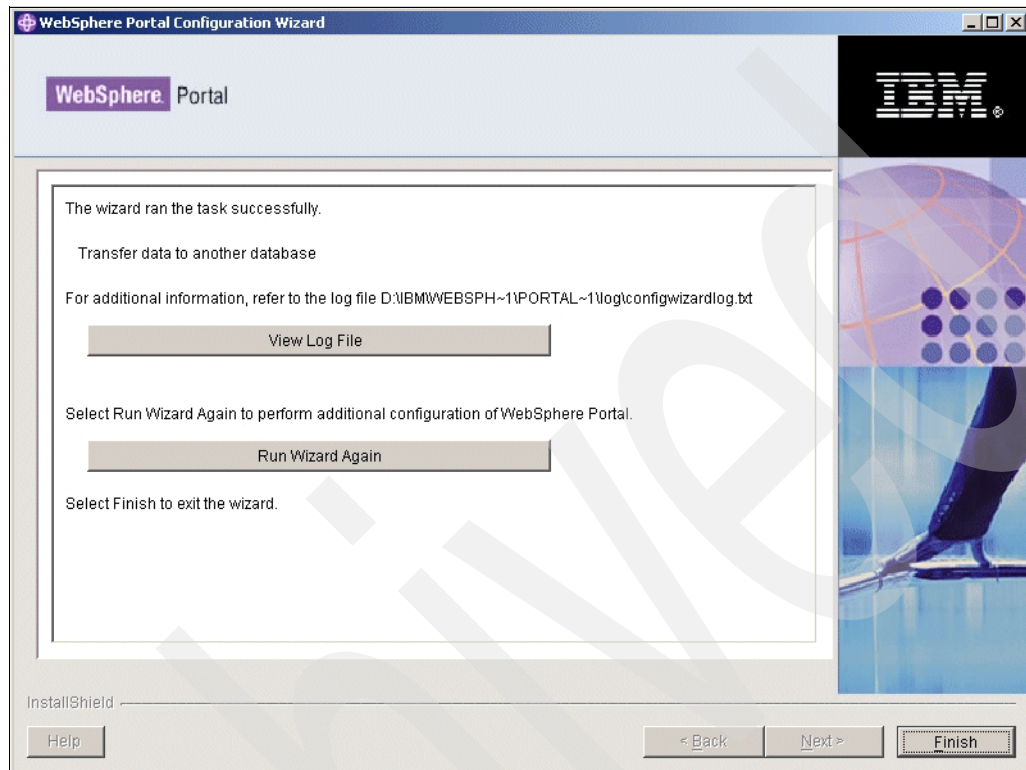


Figure 5-20 Successful database transfer

9. After the transfer process successfully completes, you can verify that the database domains moved to the new server by using the tools provided by your database software.

In our case, we repeated step 7, and used the Control Center tool that comes with DB2 to verify that the database called TARGET now contains the schemas (for example, database domains) for both JCR and Likeminds (see Figure 5-21 on page 259).

Compare Figure 5-21 on page 259 with the contents of the TARGET database in Figure 5-8 on page 249. You will see that in Figure 5-8 on page 249 there were no database domains listed under schemas; however, in Figure 5-21 on page 259 we can now see the JCR and Likeminds schemas.

This proves that our database domains were successfully moved to the new DB2 database server.

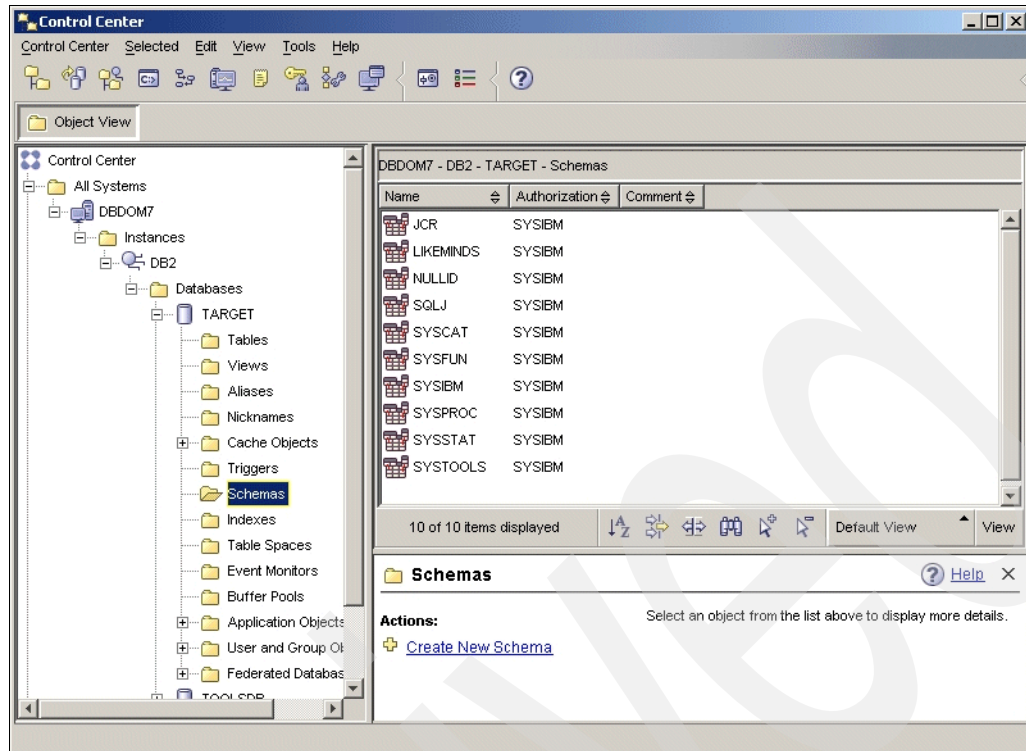


Figure 5-21 Verify the results of the database domain moves

5.5 Example 2, moving domains between database types

Example 2 is similar to example 1; however, instead of moving database domains to another database server of the same type, this time we move database domains from DB2 to Oracle.

Although the steps involved in moving database domains to Oracle (or any other supported database server) are very similar to the steps outlined in example 1, it is important to understand some of the configuration differences. You also need to understand under what circumstances to move database domains to database servers of different types.

Following are some of the reasons we might want to move database domains to database servers of different types:

- ▶ We may want to use an existing database environment other than the one we currently use with WebSphere Portal. This could help us to realize some cost savings by reusing existing skills and infrastructure.
- ▶ We may merge with or take over another organization that does not use the same database software we currently use, and it would not be cost effective to migrate them to our database software.
- ▶ We may want to migrate WebSphere Portal from one type of database software to another for commercial or technical reasons.

Important: You can use the steps listed in this section to move individual database domains from one database server to another, and you can also use the steps to migrate your entire Portal from one type of database server to another.

A quick recap of Example 1

At the end of example 1 our portal environment comprised one portal server connected to two DB2 database servers. The JCR and Likeminds database domains were moved to the second DB2 database server (in to the TARGET database) and the remaining database domains stayed on their original database server (in the WPSDB database).

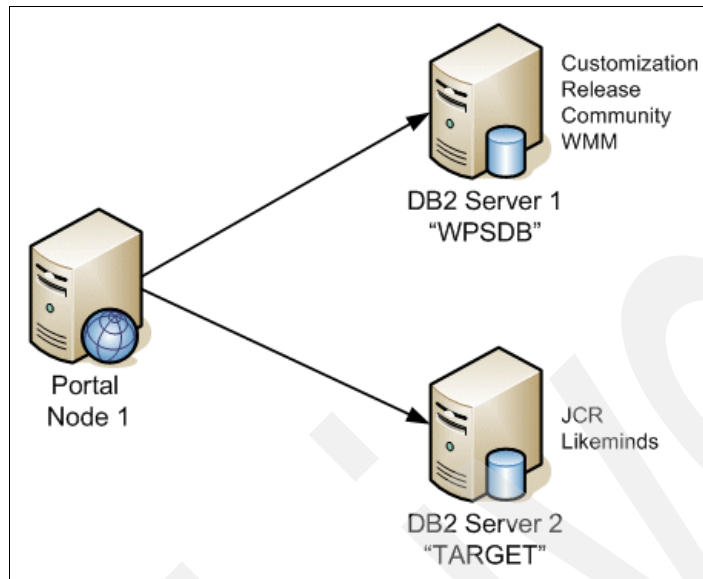


Figure 5-22 Example 1, Portal connected to two DB2 servers

A summary of Example 2

In example 2, we now want to move *all* the remaining database domains from their original DB2 server and database called WPSDB to an Oracle database server with a database called ORACLEDB.

This new environment is described in Figure 5-23 on page 261.

Following are the remaining database domains that we now want to move to Oracle:

- ▶ Customization
- ▶ Release
- ▶ Community
- ▶ WMM (The domain is actually named after the administrative user for the database. In our case it was called db2admin.)

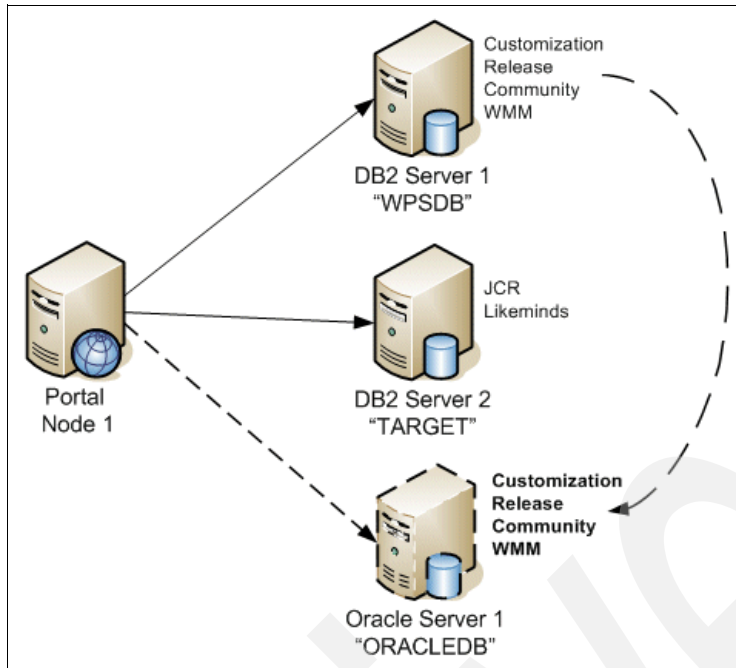


Figure 5-23 Example 2, moving the remaining database domains to Oracle

After we move the remaining database domains from the original DB2 Database Server and "WPSDB" database, we will no longer require the original DB2 server for our Portal environment as shown in Figure 5-24.

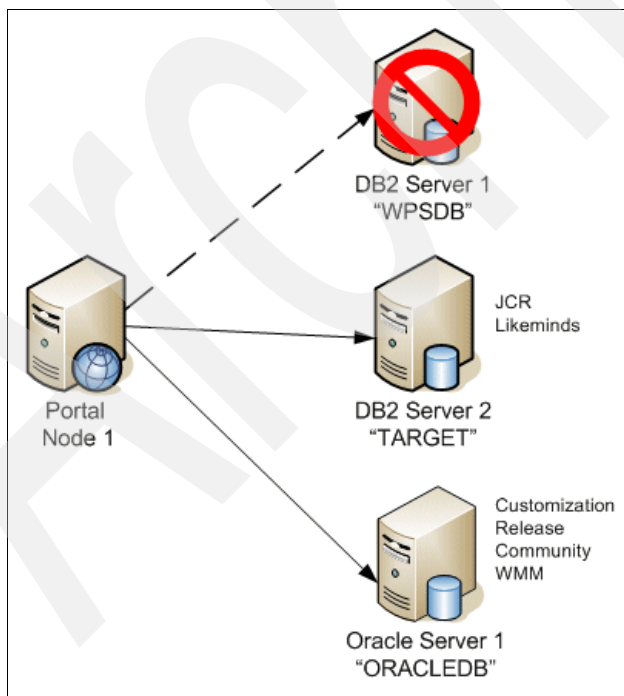


Figure 5-24 Example 2, retiring the original DB2 server and Portal database

Steps for moving database domains between database types

Tip: See the Infocenter document entitled “Installing Oracle” for more details on how to install and configure Oracle for use with WebSphere Portal.

Also, the WebSphere Portal Version 6 Cluster Guide also contains the steps for configuring Oracle for use with WebSphere Portal. This is a particularly useful document and can be found at the following Web address:

http://www-1.ibm.com/support/docview.wss?rs=688&context=SSHRKX&context=SSYJ99&context=SSBRWT&context=SSQKRQ&context=SS3NNG&context=SSRUWN&context=SS6JVV&q1=cluster+guide&uid=swg21246630&loc=en_US&cs=utf-8&lang=en

1. We installed and configured a remote Oracle database server and created a database called ORACLEDB.
2. We then prepared our portal server for Oracle by copying the relevant Oracle database driver on to the portal server and checking connectivity between Portal and the Oracle database server.

See the Infocenter document entitled “Planning for Oracle” for more details on how to install and configure WebSphere Portal for use with Oracle.

3. Edit `wpconfig_dbtype.properties`. We edited this properties file to provide the configuration wizard (or database-transfer task) with the location of the drivers that would allow it to connect to the destination Oracle database server/database.

From example 1, the `wpconfig_dbtype.properties` already contained details for DB2, as well as for Cloudscape. So we specified the details for how to connect to Oracle as shown in **bold** in Example 5-5.

Important: Regardless of which operating system your portal server is using, make sure any file system paths use the forward slash /.

Example 5-5 Editing `wpconfig_dbtype.properties` for Oracle

```
#####
# Oracle Properties
#####

# DriverManager: The name of class SqlProcessor will use to import SQL files
oracle.DbDriver=oracle.jdbc.driver.OracleDriver

# DbLibrary: The directory and name of the zip/jar file containing JDBC driver
class
# Please use the system specific file separator names, e.g. for windows semicolon
and for unix colon.
oracle.DbLibrary=C:/IBM/WebSphere/OracleDriver/ojdbc14.jar

# JdbcProviderName: The name of jdbc provider to be used
oracle.JdbcProviderName=wpdbJDBC_oracle

#####
# END: Oracle Properties
#####
```

4. Edit the `wpconfig_sourceDB.properties` file. In example 1, we edited the values for the JCR and Likeminds database domains to point them from Cloudscape to DB2.

The `wpconfig_sourceDB.properties` file is not updated as part of any database transfer process and therefore still lists the remaining database domains (customization, Release, community, and WMM) as being in the original Cloudscape database even though they are actually in DB2.

Therefore, in order to move any of our remaining database domains, we first updated `wpconfig_sourceDB.properties` to reflect the fact that these database domains are in fact currently in DB2.

When we then run either the configuration wizard or the WPCfg database-transfer task to move our database domains, the updated values in the `wpconfig_sourceDB.properties` file will be read and the wizard or task will then know where the database domains are that we want to move to Oracle.

Example 5-6 shows the properties in the `wpconfig_sourceDB.properties` file associated with the release database domain. The values we changed are as follows (and are shown in **bold** in Example 5-6):

- **source.release.DbType**. Specifies the type of database where the Domain is currently stored. In our case DB2.
- **source.release.DbName**. Specifies the name of the database where the Domain is currently stored. In our case WPSDB (see Figure 5-5 on page 244).
- **source.release.DbSchema**. Specifies the name of the source schema/domain.
- **source.release.DbUrl**. Specifies where the database currently is and how we connect to it.
- **source.release.DbUser**. Specifies the user name to use when connecting to the database where our database domain is currently stored.
- **source.release.DbPassword**. Specifies the password to use for the previously specified user name.

Tip: Repeat step 4 for each of the database domains you want to move.

Example 5-6 Editing `wpconfig_sourceDB.properties` for DB2

```
#####
# Release Database Properties
#####

# DbType: The type of database to be used for WebSphere Portal Release domain
source.release.DbType=db2

# DbName: The name of the WebSphere Portal Release database
# Note: This value should also appear as the database element in DbUrl
# Note: Non-Windows platforms when using DB2 only. This value is the TCP/IP
# Alias for the database.
source.release.DbName=wpsdb

# DbSchema: The WebSphere Portal Release domain database schema name
#           Follow the documentation of the target database management system
#           in order to define a valid schema name as restrictions apply for
#           some database management systems.
source.release.DbSchema=release
```

```
# DbUrl: The wps release domain database URL
source.release.DbUrl=jdbc:db2:wpsdb

# DbUser: The database administrator user ID
source.release.DbUser=db2admin

# DbPassword: The database administrator password
source.release.DbPassword=password

#####
# END: Release Database Properties
#####
```

5. Repeat step 4 for any other database domains you want to move (in our case we repeated step 4 for the customization, community and WWM Domains).

6. Edit `wpconfig_dbdomain.properties`. Now that we specified the *current* location of our database domains, we need to specify where we want to move them to.

In our case, we wanted to move our remaining four database domains to an Oracle server and into a database called ORACLEDB.

Example 5-7 shows the values we changed for the Customization database domain (in **bold**).

- **customization.DbType**. Specifies the type of database where the Domain will be stored once it is moved. In our case Oracle.
- **customization.DbName**. Specifies the name of the database where the Domain is to be moved to. In our case ORACLEDB. See Figure 5-24 on page 261.
- **customization.DbSchema**. Specifies a unique name for the schema for this particular database domain.
- **customization.DataSourceName**. Specifies a unique name for the data source for this particular database domain.
- **customization.DbUrl**. Specifies how to connect to the new database where we want to move our database domains to.
- **customization.DbUser**. Specifies the user name to use when connecting to the database where we want to move our database domain to.
- **customization.DbPassword**. Specifies the password to use for the previously specified user name.

Tip: Repeat step 6 for each of the database domains you want to move.

Example 5-7 Editing the `wpconfig_dbdomain.properties` for the customization domain

```
#####
# Customization Database Properties
#####

# DbType: The type of database to be used for WebSphere Portal Customization
# domain
customization.DbType=oracle

# DbName: The name of the WebSphere Portal Customization domain database
```

```
# Note: This value should also appear as the database element in DbUrl
# Note: Non-Windows platforms when using DB2 only. This value is the TCPIP
# Alias for the database.
customization.DbName=oracledb

# DbSchema: The WebSphere Portal Customization domain database schema name
#           Follow the documentation of the target database management system
#           in order to define a valid schema name as restrictions apply for
#           some database management systems.
customization.DbSchema=wps_customization

# DataSourceName: The name of datasource to be used for WebSphere Portal
# Customization domain
customization.DataSourceName=wpdbDS_customization

# DbUrl: The wp customization domain database URL
customization.DbUrl=jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb

# DbUser: The database administrator user ID
customization.DbUser=wps_customization

# DbPassword: The database administrator password
customization.DbPassword=password

#####
# END: Customization Database Properties
#####
```

Important: The data sourceName needs to be unique when transferring to database domains that exist in the same database instance. In our case we added an underscore followed by the database domain name and adopted this naming convention for each of the database domains we moved to Oracle.

For example, we changed wpdbDS to wpdbDS_customization for the customization data source name.

Repeat step 6 for any other database domains you may want to move.

7. Run the configuration wizard or WPConfig database-transfer task. For the database domains we want to move, we specified where they currently are and where we want to move them to.

As a result, we are now ready to transfer the database domains (in our case release, customization, community and WWM) by running either the configuration wizard or WPConfig database-transfer task.

The configuration wizard can be found in <portal_server_root>/config/wizard. Run the configwizard.bat file to start the wizard. Alternatively you can run the following command from <portal_server_root>/config/ directory (substituting the names of your database domains):

```
Windows: WPSconfig database-transfer
-DTransferDomainList=release,customization, community, wmm
UNIX: ./WPSconfig database-transfer
-DTransferDomainList=release,customization,community,wmm
```

See the Infocenter document entitled “Transferring between databases manually” for more information about the command line options.

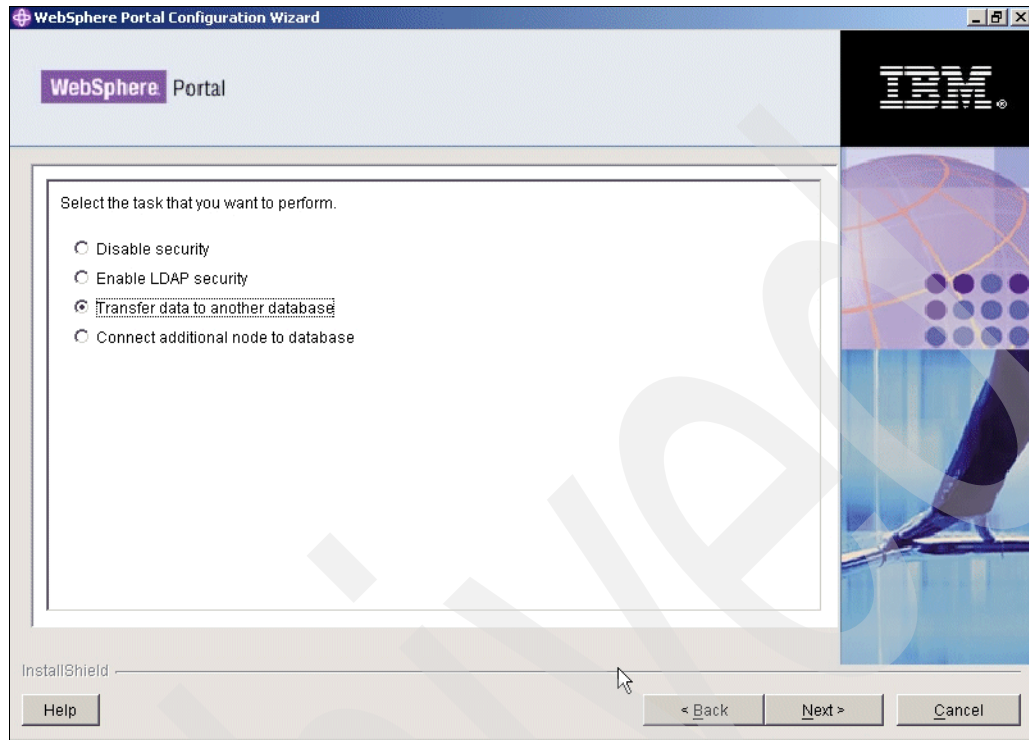


Figure 5-25 Select “Transfer data to another database”

Select the option to **Transfer data to another database**, and then specify the WebSphere Application Server Administrator user name and password.

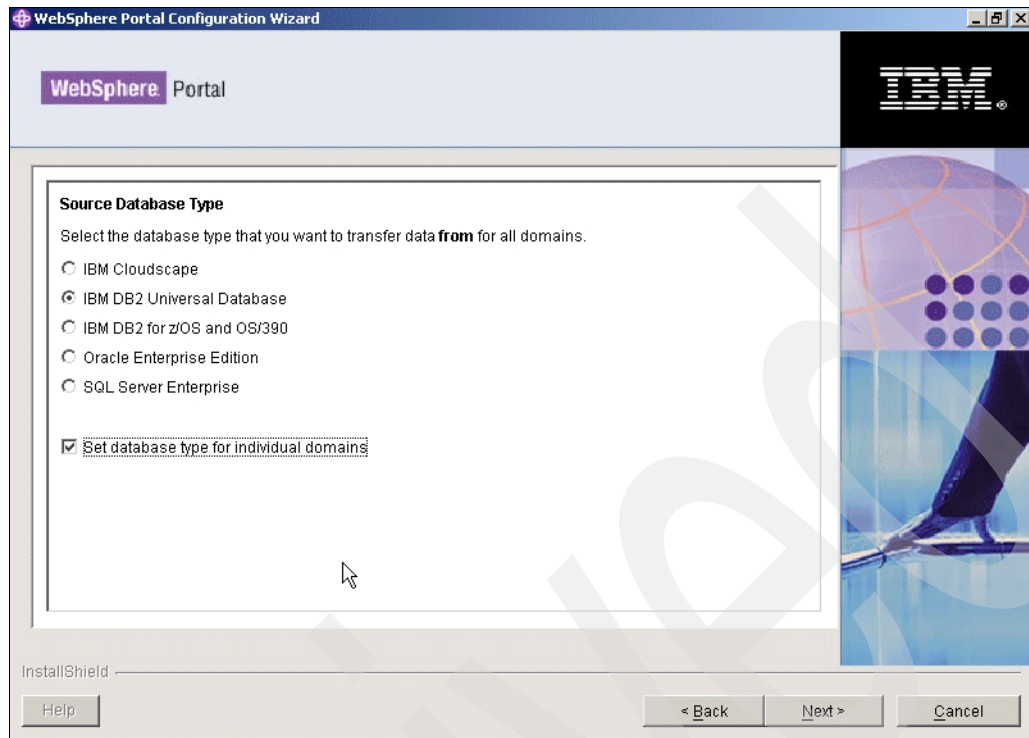


Figure 5-26 Select Source Database Type

Select **Set database type for individual domains** as shown in Figure 5-26. This option allows us to specify the individual database domains we want to transfer to Oracle.

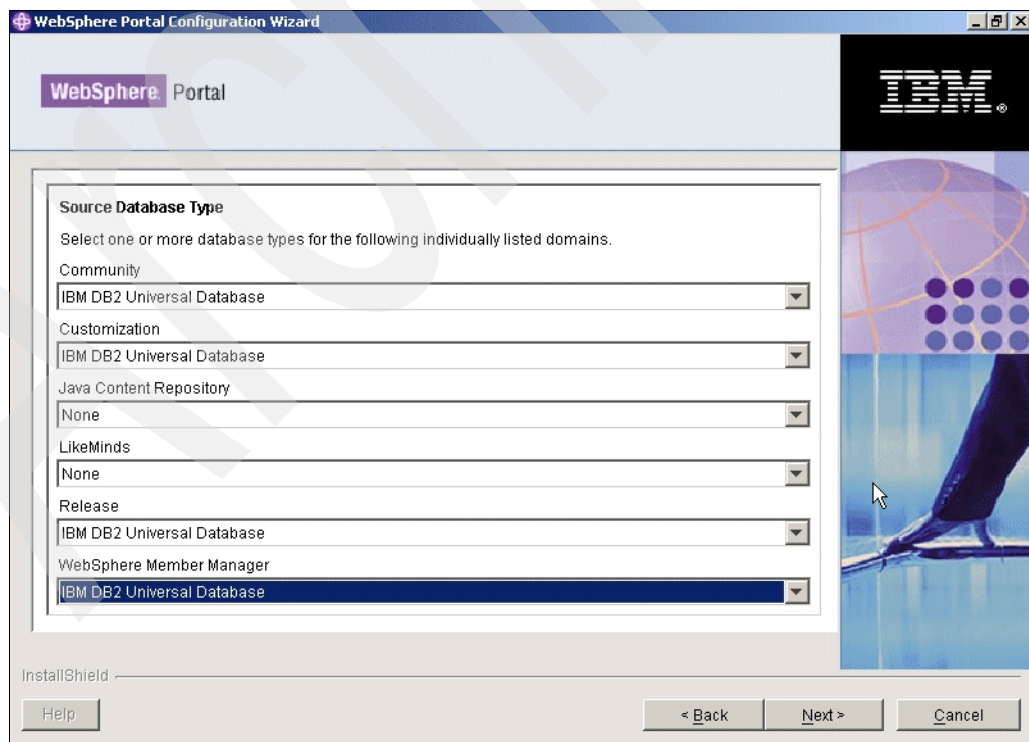


Figure 5-27 Selecting the source database domains

In Figure 5-27 on page 267 we specify where the database domains that we want to move currently reside. In our case the release, community, customization and WMM domains are all currently stored in DB2.

Important: Leave any database domains that you do **not** want to move set to “None”.

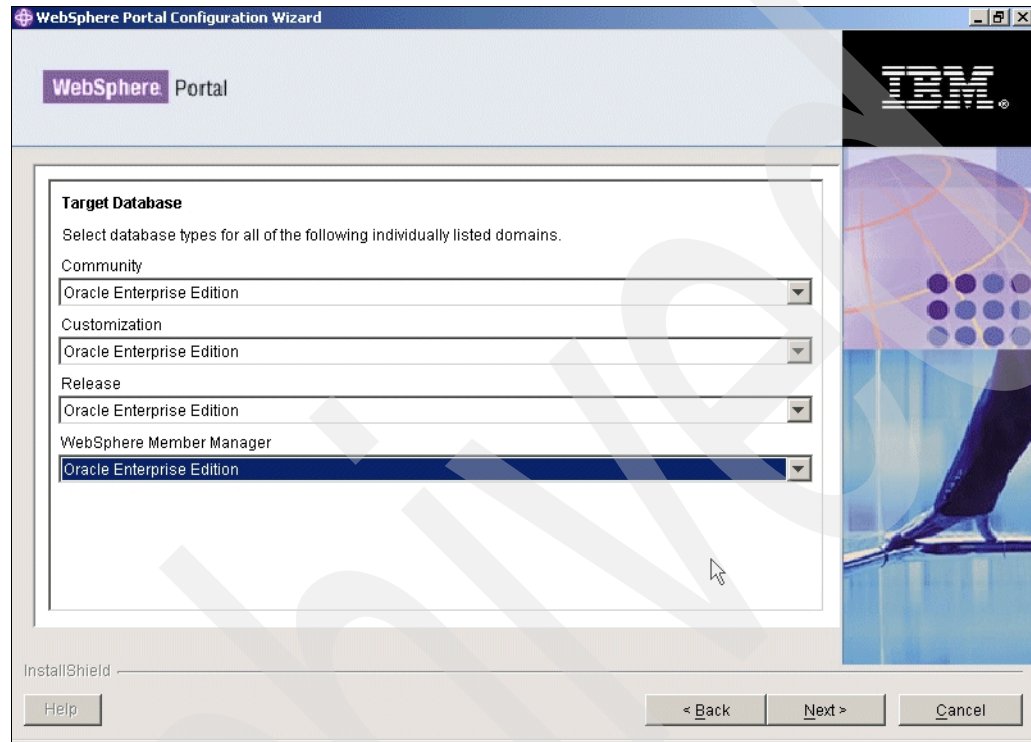


Figure 5-28 Selecting the database domains we want to move to Oracle

In Figure 5-28 we specified which databases server type will store the database domains we want to move. In our case the four database domains to be moved will all be stored in Oracle.

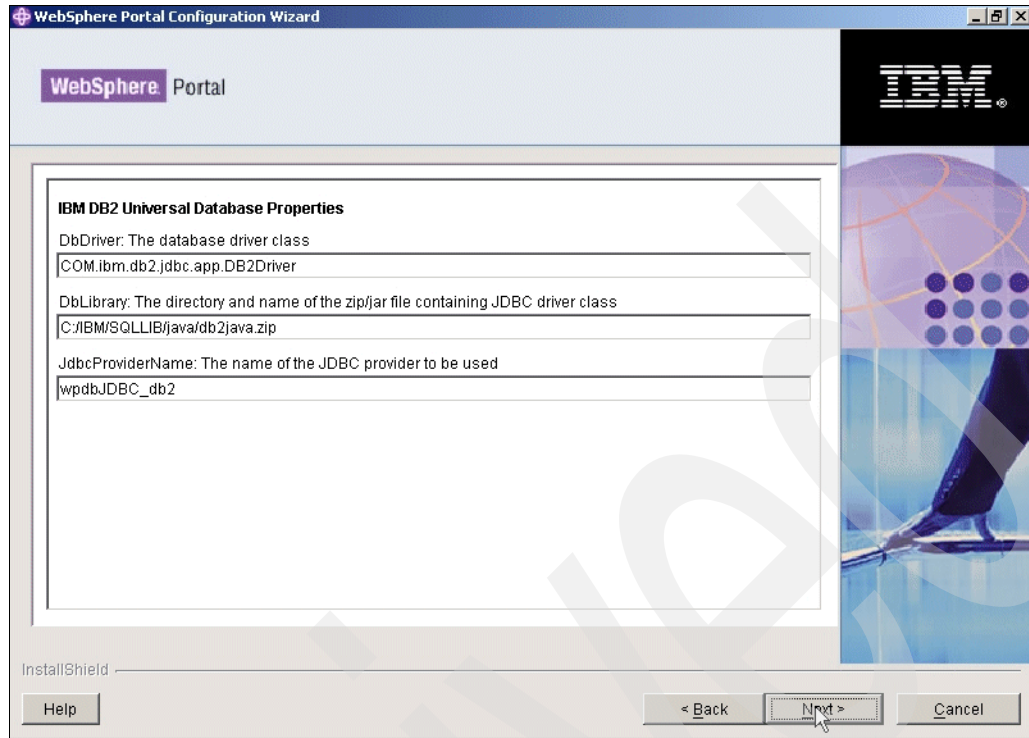


Figure 5-29 Specifying the DB2 database server properties

Figure 5-29 shows the DB2 properties that we specified in the `wpconfig_dbtype.properties` file in step 3 (Example 5-2 on page 246). These values define how portal server is going to connect to DB2 database servers. Ensure these values are correct before proceeding; otherwise, the database transfer process will not be successful.

Figure 5-30 on page 270 shows the Oracle properties that we specified in the `wpconfig_dbtype.properties` file in step 3 (Example 5-5 on page 262). These values define how portal server is going to connect to Oracle database servers.

Ensure that these values are correct before proceeding; otherwise, the database transfer process will not be successful.

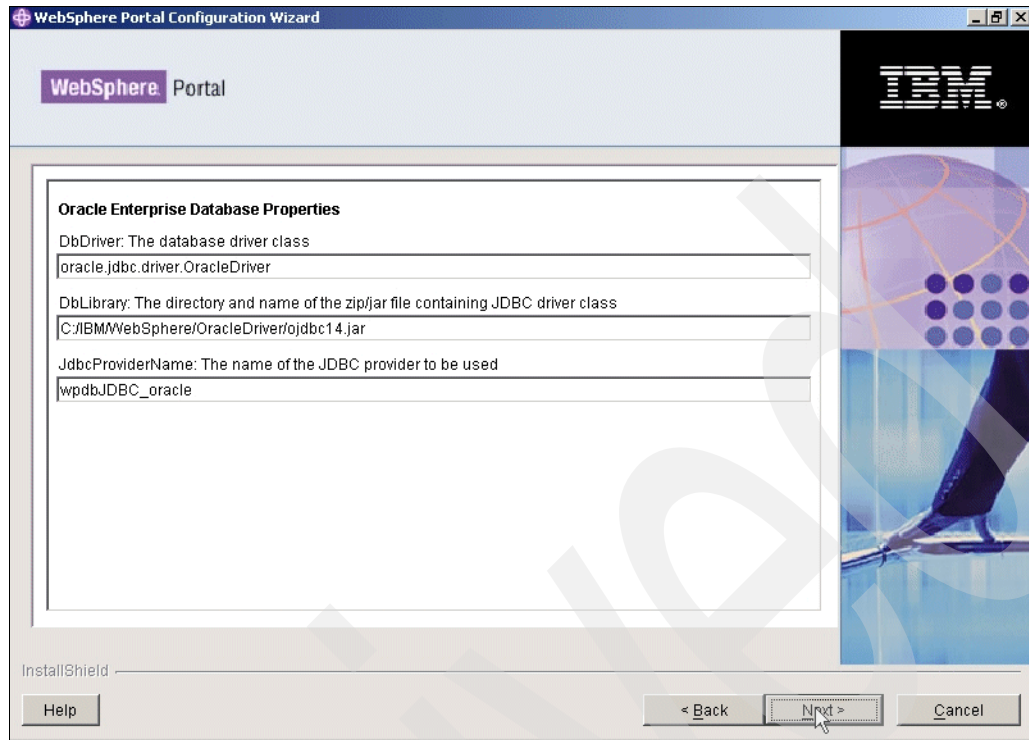
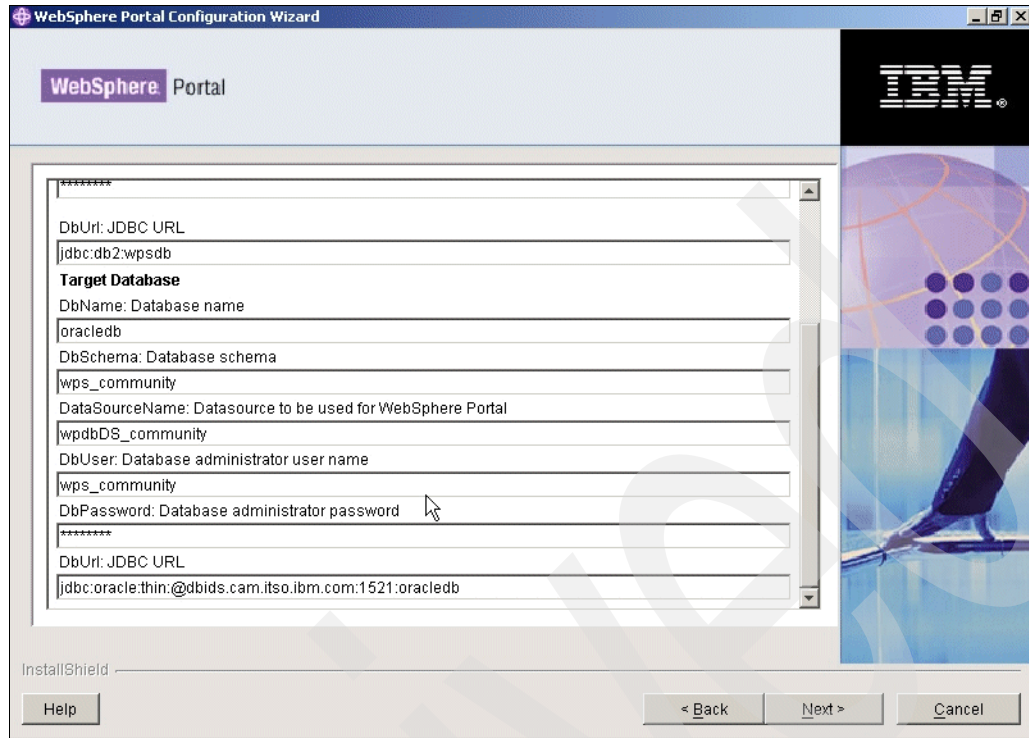


Figure 5-30 Specify the Oracle database server properties

Figure 5-31 on page 271 shows the details for the first database domain we want to move (in our case Community). Provide a valid user name and password to connect to the target Oracle database/database server; otherwise, the database transfer process will fail.

In our case this was the user **wps_community** and the password was **password**.

Tip: After the configuration wizard runs, for security reasons, any password values you specified in the properties files it reads from are replaced with the generic text string of `ReplaceWithYourDbAdminPwd`.



WebSphere Portal Configuration Wizard

WebSphere Portal

DbUrl: JDBC URL
jdbc:db2:wpbdb

Target Database

DbName: Database name
oracledb

DbSchema: Database schema
wps_community

DataSourceName: Datasource to be used for WebSphere Portal
wpdbDS_community

DbUser: Database administrator user name
wps_community

DbPassword: Database administrator password

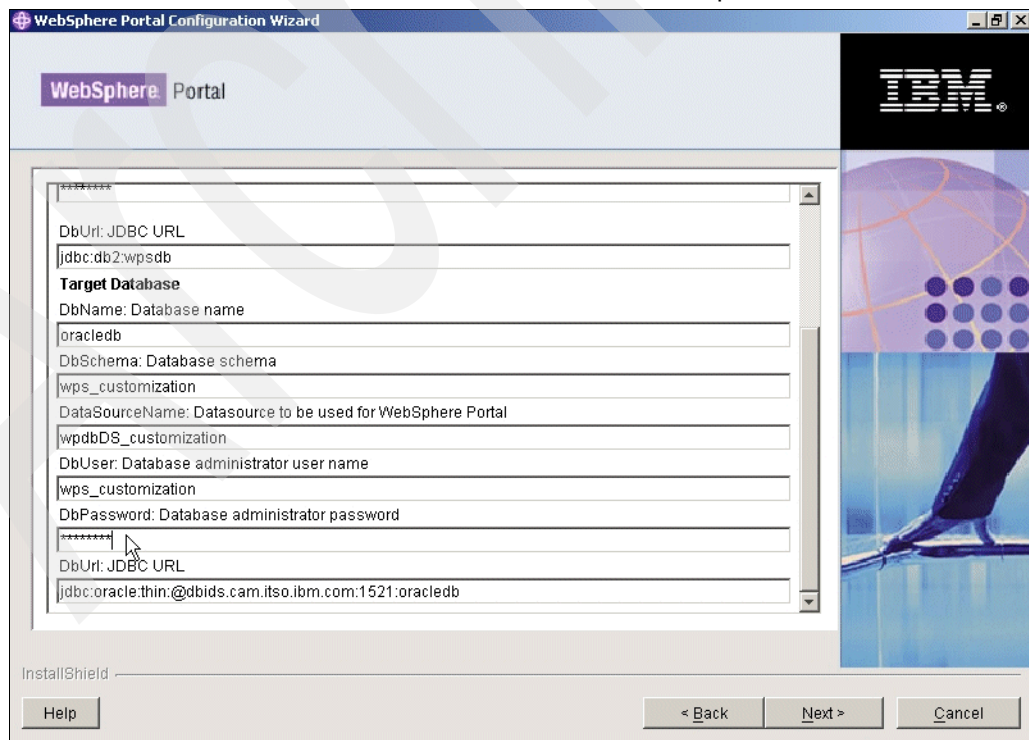
DbUrl: JDBC URL
jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb

InstallShield

Help < Back Next > Cancel

Figure 5-31 Specify the target Oracle database properties for the community domain

Figure 5-32 shows the details for the next database domain we want to move (in our case Customization). Provide a valid user name and password to connect to the target Oracle database/database server; otherwise, the database transfer process will fail.



WebSphere Portal Configuration Wizard

WebSphere Portal

DbUrl: JDBC URL
jdbc:db2:wpbdb

Target Database

DbName: Database name
oracledb

DbSchema: Database schema
wps_customization

DataSourceName: Datasource to be used for WebSphere Portal
wpdbDS_customization

DbUser: Database administrator user name
wps_customization

DbPassword: Database administrator password

DbUrl: JDBC URL
jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb

InstallShield

Help < Back Next > Cancel

Figure 5-32 Specify the target Oracle database properties for the Customization Domain

In our case the Oracle user was called **wps_customization** and the password was **password**.

Figure 5-33 shows the details for the next database domain we want to move (in our case release). Provide a valid user name and password to connect to the target Oracle database/ database server; otherwise, the database transfer process will fail.

In our case the Oracle user was called **wps_release** with a password of **password**.

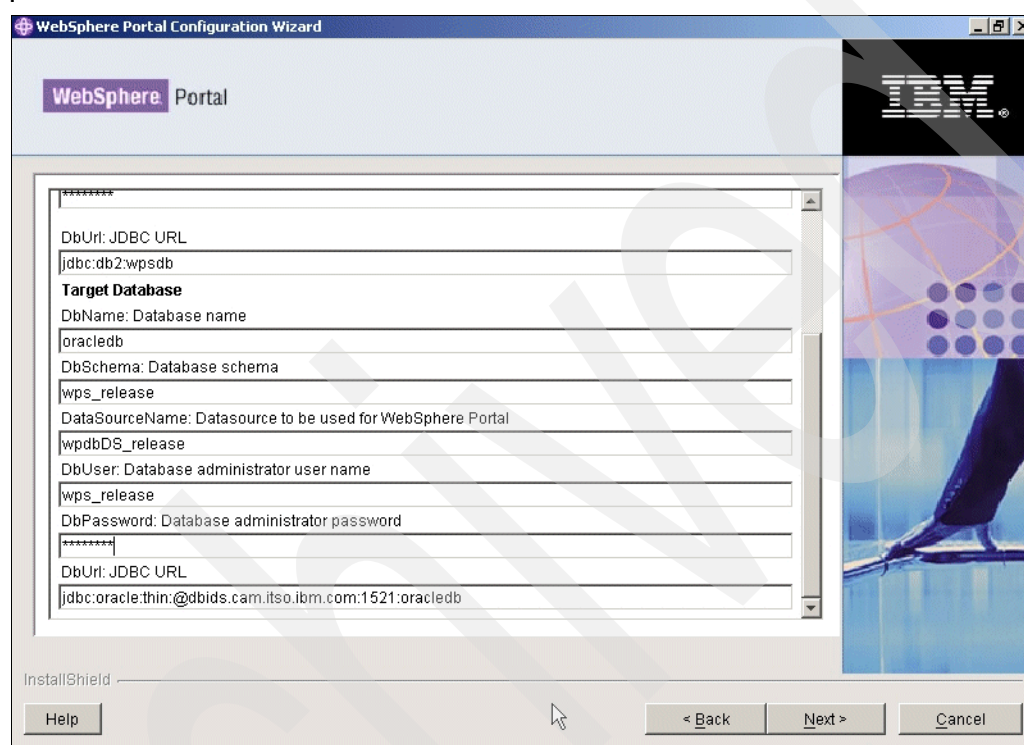
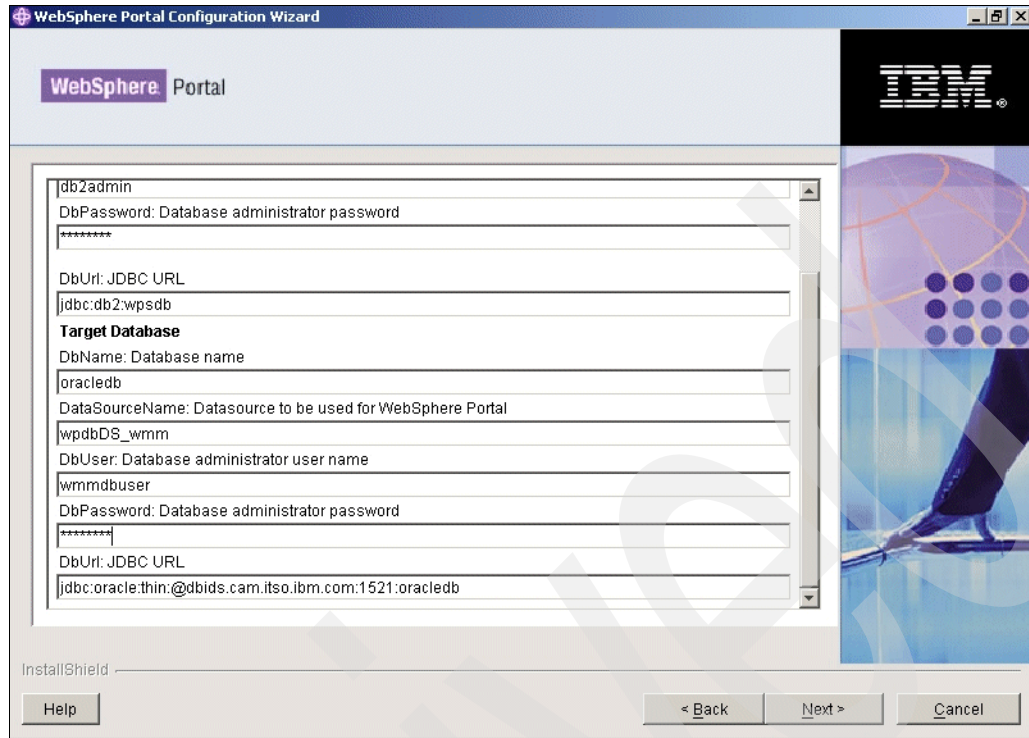
The image shows a screenshot of the 'WebSphere Portal Configuration Wizard' window. The title bar reads 'WebSphere Portal Configuration Wizard'. The main window has a header with the 'WebSphere Portal' logo and the IBM logo. The central area contains a list of configuration fields for a 'Target Database'. The fields are: 'DbUrl: JDBC URL' (value: jdbc:db2:wpsdb), 'DbName: Database name' (value: oracledb), 'DbSchema: Database schema' (value: wps_release), 'DataSourceName: Datasource to be used for WebSphere Portal' (value: wpsdbDS_release), 'DbUser: Database administrator user name' (value: wps_release), 'DbPassword: Database administrator password' (masked with asterisks), and 'DbUrl: JDBC URL' (value: jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb). At the bottom, there is an 'InstallShield' label, a 'Help' button, and navigation buttons '< Back', 'Next >', and 'Cancel'.

Figure 5-33 Specify the target Oracle database properties for the release Domain

Figure 5-34 on page 273 shows the details for the final database domain we wanted to move (in our case WMM). Provide a valid user name and password to connect to the target Oracle database/database server; otherwise, the database transfer process will fail.

In our case the Oracle user was called **wmmdbuser** with a password of **password**.



WebSphere Portal

db2admin
 DbPassword: Database administrator password

DbUrl: JDBC URL
 jdbc:db2:wpsdb

Target Database
 DbName: Database name
 oracledb
 DataSourceName: Datasource to be used for WebSphere Portal
 wpdbDS_wmm
 DbUser: Database administrator user name
 wmmdbuser
 DbPassword: Database administrator password

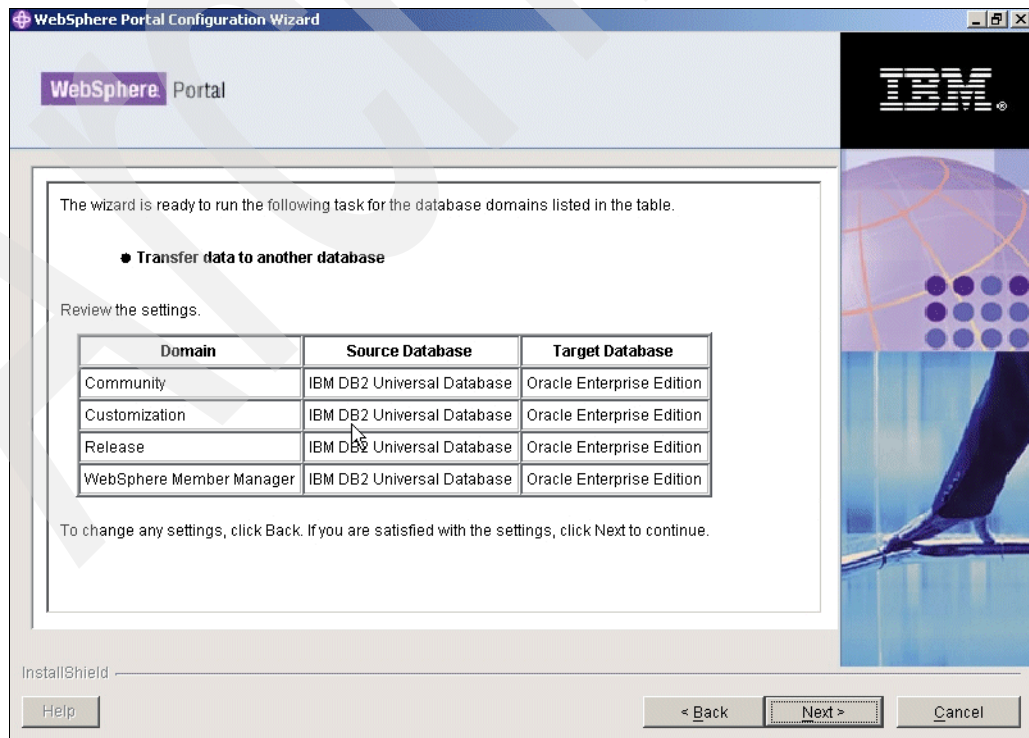
 DbUrl: JDBC URL
 jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb

InstallShield

Help < Back Next > Cancel

Figure 5-34 Specify the target Oracle database properties for the WMM domain

Figure 5-35 shows a summary of the database domains that will be transferred to the Oracle databases server. It lists where the database domains currently (DB2) reside and where they will be moved to (Oracle). Click **Next** to begin the transfer process.



WebSphere Portal

The wizard is ready to run the following task for the database domains listed in the table.

● **Transfer data to another database**

Review the settings.

Domain	Source Database	Target Database
Community	IBM DB2 Universal Database	Oracle Enterprise Edition
Customization	IBM DB2 Universal Database	Oracle Enterprise Edition
Release	IBM DB2 Universal Database	Oracle Enterprise Edition
WebSphere Member Manager	IBM DB2 Universal Database	Oracle Enterprise Edition

To change any settings, click Back. If you are satisfied with the settings, click Next to continue.

InstallShield

Help < Back Next > Cancel

Figure 5-35 Summary

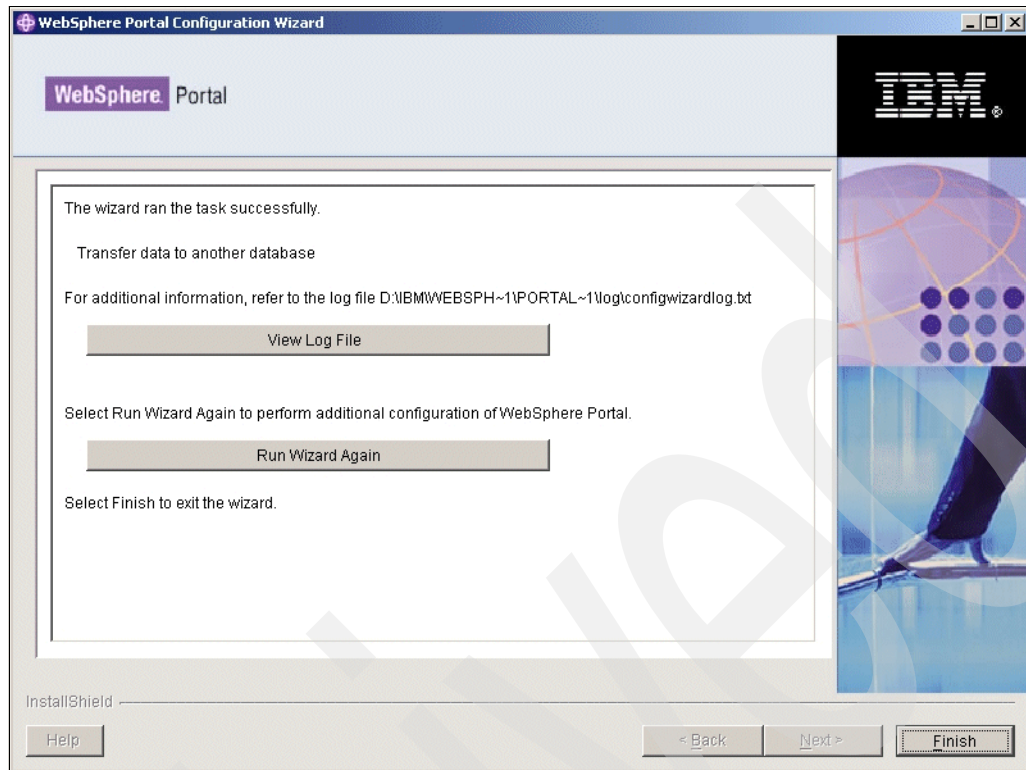


Figure 5-36 Successful transfer

In order to verify that our portal server was using both DB2 and Oracle as its data sources, we followed the instructions in the section entitled "Testing your JDBC data connection on the DMGR" on page 105.

Note: When following these instructions, make sure that you also create a WebSphere Variable for the `ORACLE_JDBC_DRIVER_CLASSPATH`, and not just for the `DB2_JDBC_DRIVER_CLASSPATH`.

Figure 5-37 on page 275 shows the results of a successful test of our JDBC data sources.

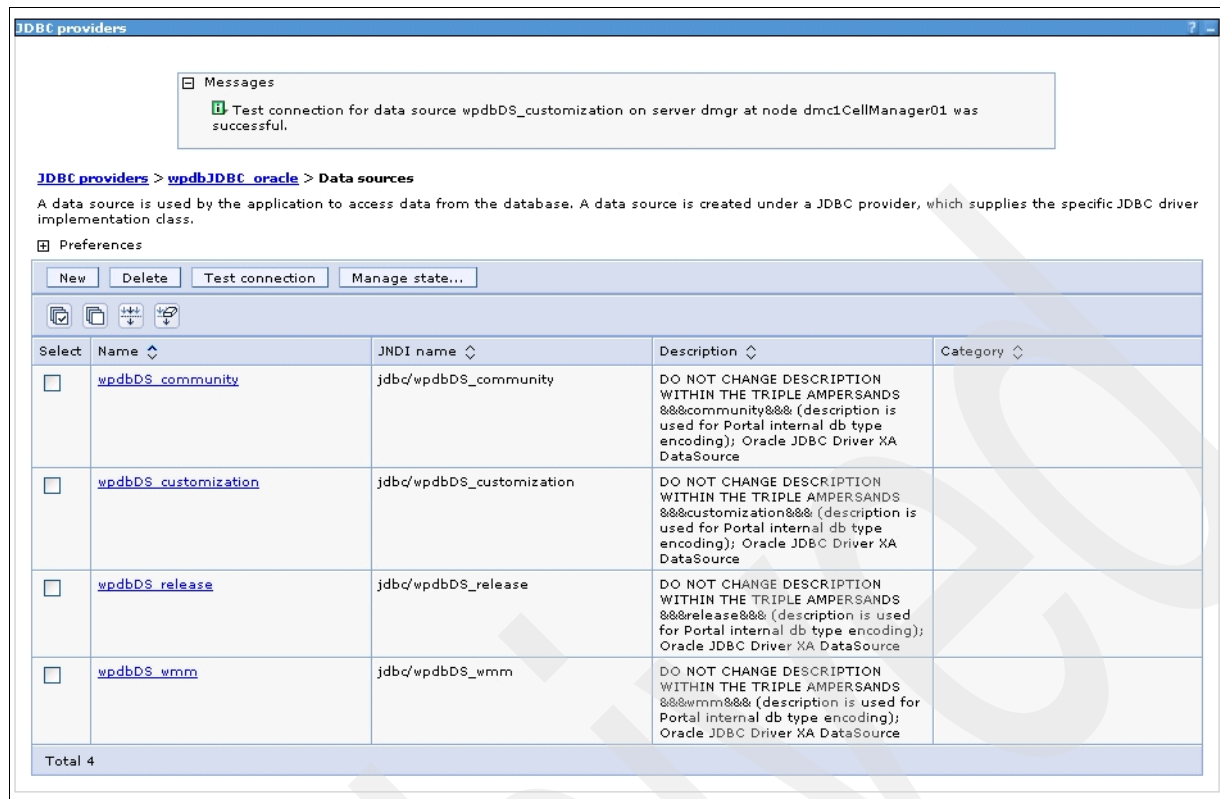


Figure 5-37 Successfully testing our JDBC connections to DB2 and Oracle

5.6 Example 3, sharing domains amongst Portal nodes

Example 3 differs somewhat from the previous two examples because rather than moving database domains, we are now going to get multiple portal server nodes to *share* a common set of database domains. This is particularly useful when designing high availability portal environments such as the one described earlier in this chapter in Figure 5-3 on page 239.

5.6.1 A quick recap of example 2

In 5.5, “Example 2, moving domains between database types” on page 259, we successfully moved our remaining database domains from our first DB2 database server to an Oracle database server. This resulted in our environment looking like Figure 5-38 on page 276.

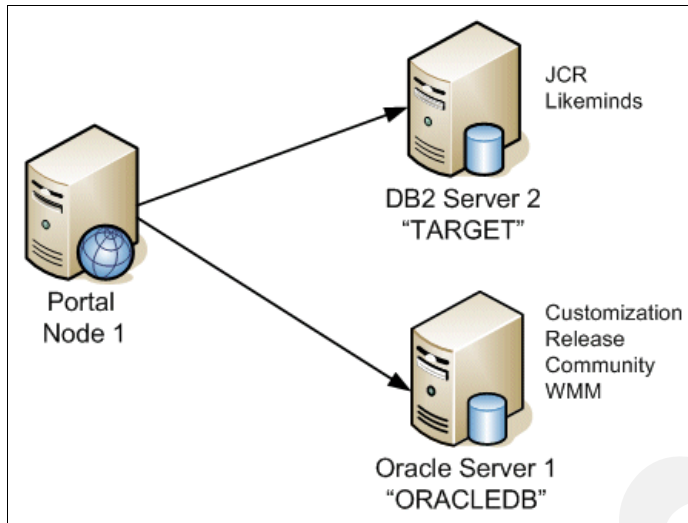


Figure 5-38 Example 2, Portal using DB2 and Oracle

As shown in Figure 5-38, our portal server is using two database domains (JCR and Likeminds) in DB2 and four database domains (customization, release, community and WMM) in Oracle.

5.6.2 A summary of example 3

In 5.6, "Example 3, sharing domains amongst Portal nodes" on page 275, we now want to show how individual Portal nodes can share and use the same Portal data.

To accomplish this we installed a new Portal server as an empty Portal. We then used XMLaccess export from the original Portal environment and imported into the new Portal environment to create identical Portal environments. We did this to ensure that both Portal environments' file system configurations for the Portlet applications are the same. We hoped to have a good baseline to build from.

Eventually, this new Portal environment will be connected to, and share, the existing DB2 and Oracle database domains used by the original Portal environment. (Note: The release and JCR domains will not be shared).

This procedure or architecture could be valuable when wanting to maintain identical Portal environments that will be used for pre-production activities. This approach and architecture can therefore be maintained without the need to constantly use XMLaccess to keep the environments identical.

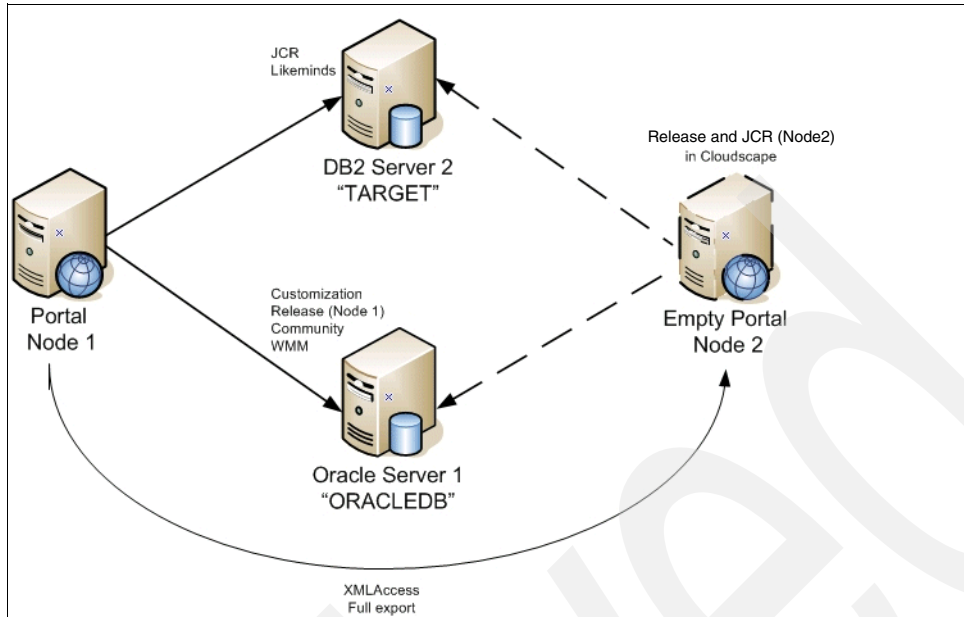


Figure 5-39 Example 3, multiple portals sharing the same database domains

Once configured, we should be able to log into either portal (as the WMM database domain would be common amongst both portals), and we should also be able to see exactly the same content.

Steps for sharing database domains between portal servers

1. We installed an empty portal server on node 2 (see the Infocenter document entitled "Advanced Installation Options" for details on using the -W flag to install an empty portal).
2. On our existing portal server (node 1), we did the following -
 - a. Created a test user (jdelaney), which is stored in the WMM database domain because we are using a WMM CUR.
 - b. Created some content in the form of portal page (see Figure 5-40).

Note: Release data cannot be shared amongst portal servers (see Table 5-1 on page 237); however, we created some Release content in the form of a page, so we could verify that our XMLAccess export from node 1 and subsequent import in to node 2 was successful.

- c. Created a document in the Portal Document Manager on node 1, which would be stored in the JCR database domain (see Figure 5-41 on page 278).



Figure 5-40 Some additional release content created on node 1

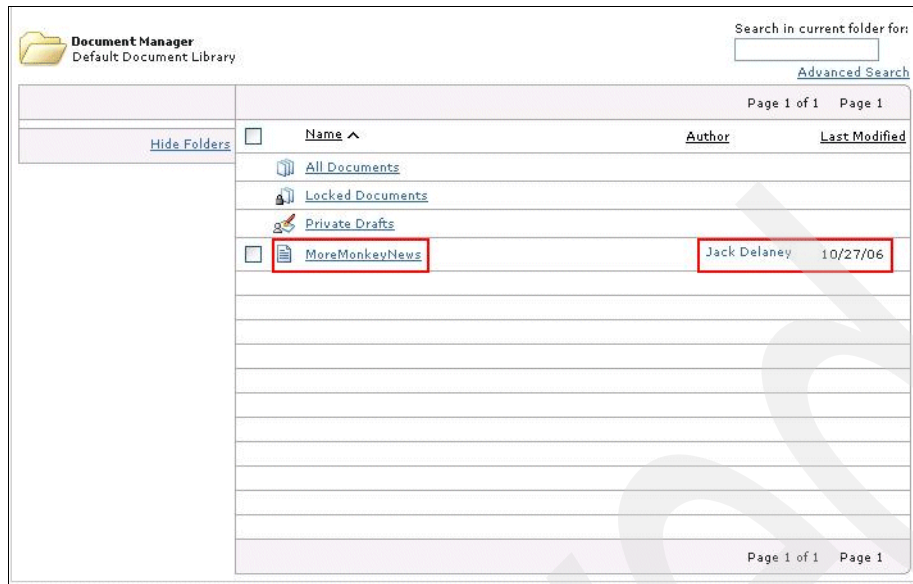


Figure 5-41 Some JCR content created on portal node 1

3. Having created some content in node 1, we did a full XMLAccess export, and then imported into node 2 so that both portal servers would have exactly the same content. We verified this by logging in to node 2 after the XMLAccess import and verifying that we could see the page we previously created on node 1 (Figure 5-40 on page 277).

See Chapter 7, "Configuration management: moving between environments" on page 337 and the Infocenter for full details on how to export and import a portal server's content with XMLAccess.

Note: In our example we built our empty portal server, carried out an XMLAccess import, and then connected the new portal server to its new data sources.

As you may recall from Table 5-1 on page 237, the release database domain cannot be shared between portal servers so in our example we ended up with node 1 having its release database domain stored in Oracle and node 2 keeping its release database domain in Cloudscape.

In a typical production environment, however, we would not want to keep our release database domain in Cloudscape so the steps we used would most likely be as follows:

- ▶ Run the database Transfer task from the empty portal to an external database that would include the release domain.
- ▶ Carry out the XMLAccess import on the empty portal.
- ▶ Connect the empty portal to the shared database domains.

Using these steps we avoid the additional overhead of importing all the Release data into Cloudscape (via XMLAccess) on the empty portal, and then having to transfer it again to an external database.

After the XMLAccess import successfully completed, we prepared node 2 for connection to DB2 and Oracle. This involved the following:

- Installing the DB2 client on node 2 and cataloguing the remote DB2 database. See Chapter 3 for more details on how to do this for DB2.

- Copying the Oracle JDBC driver to node 2. See step 2 of “Steps for moving database domains between database types” on page 262.
 - Editing the wpconfig.properties file with the relevant WebSphere Application Server and Portal Server credentials required for the database connect process.
4. Edit wpconfig_dbtype.properties. Similar to examples 1 and 2, on node 2 we edited the wpconfig_dbtype.properties file to provide information about how to connect to both the DB2 and Oracle database servers (see Example 5-8).

Important: Regardless of which operating system your portal server is using, make sure any file system paths use the forward slash /.

In our case we specified the path of the DB2 library to be used (the library is installed with the DB2 client). We also specified the path to the Oracle Directory that we previously created in the WebSphere Application Server root. This is also where we copied our Oracle JDBC driver to.

Tip: Unlike with DB2, Oracle uses a type 4 JDBC driver and therefore does not require any client software to be installed on the portal server in order to allow it to connect to an Oracle database server.

Example 5-8 Editing wpconfig_dbtype.properties for both DB2 and Oracle

```
#####
# DB2 Properties
#####

# DbDriver: The name of class SqlProcessor will use to import SQL files
db2.DbDriver=COM.ibm.db2.jdbc.app.DB2Driver

# DbLibrary: The directory and name of the zip/jar file containing JDBC driver
class
# Please use the system specific file separator names, e.g. for windows semicolon
and for unix colon.
db2.DbLibrary=C:/IBM/SQLLIB/java/db2java.zip

# JdbcProviderName: The name of jdbc provider to be used
db2.JdbcProviderName=wpdbJDBC_db2

#####
# END: DB2 Properties
#####

#####
# Oracle Properties
#####

# DbDriver: The name of class SqlProcessor will use to import SQL files
oracle.DbDriver=oracle.jdbc.driver.OracleDriver

# DbLibrary: The directory and name of the zip/jar file containing JDBC driver
class
# Please use the system specific file separator names, e.g. for windows semicolon
and for unix colon.
```

```
oracle.DbLibrary=C:/IBM/WebSphere/OracleDriver/ojdbc14.jar
```

```
# JdbcProviderName: The name of jdbc provider to be used
oracle.JdbcProviderName=wpdbJDBC_oracle
```

```
#####
# END: Oracle Properties
#####
```

5. Unlike the previous two examples, there is no need in example 3 to edit the wpconfig.dbsource.properties file. This is because we will not actually be transferring any *data* from CloudScape, rather we are simply pointing the Portal to an existing Database and re-building data sources through the WPCongfig connect-database task.
6. Edit wpconfig_dbdomain.properties. Even though in this example we are not actually transferring any data from Cloudscape to DB2 and Oracle (we are simply *connecting* node 2 to DB2 and Oracle so it can share the existing data with node 1), we still need to specify where are our database domains are located so the data sources can be built correctly.

You may recall from Figure 5-38 on page 276, that our database domains are split amongst DB2 and Oracle as follows in Table 5-3:

Table 5-3 Current location of our database domains

Database domain	Database server	Database name
JCR	DB2	TARGET
Likeminds	DB2	TARGET
Release	Oracle	ORACLEDB
Customization	Oracle	ORACLEDB
Community	Oracle	ORACLEDB
WMM	Oracle	ORACLEDB

Example 5-9 shows the values we specified for the customization database domain (in Oracle).

We updated the values in the wpconfig_dbdomain.properties files for all the database domains listed in Table 5-3 with the exception of the release and JCR database domains, which we elected to leave in Cloudscape on node 2.

Example 5-9 Editing wpconfig_dbdomain.properties for DB2 and Oracle

```
#####
# Customization Database Properties
#####

# DbType: The type of database to be used for WebSphere Portal Customization
# domain
customization.DbType=oracle

# DbName: The name of the WebSphere Portal Customization domain database
# Note: This value should also appear as the database element in DbUrl
# Note: Non-Windows platforms when using DB2 only. This value is the TCP/IP
# Alias for the database.
customization.DbName=oracledb
```

```
# DbSchema: The WebSphere Portal Customization domain database schema name
#           Follow the documentation of the target database management system
#           in order to define a valid schema name as restrictions apply for
#           some database management systems.
customization.DbSchema=wps_customization
# DataSourceName: The name of datasource to be used for WebSphere Portal
# Customization domain
customization.DataSourceName=wpsdbDS_customization

# DbUrl: The wp customization domain database URL
customization.DbUrl=jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb

# DbUser: The database administrator user ID
customization.DbUser=wps_customization

# DbPassword: The database administrator password
customization.DbPassword=password
#####
# END: Customization Database Properties
#####
```

7. Run the configuration wizard or WPSconfig connect-database task. For the database domains we want to connect and share, we specified which existing domains to connect to in order to use the existing database domains used by node 1 (for example, DB2 and Oracle).

As a result, we are now ready to connect to the database domain (in our case JCR and Likeminds in DB2 and customization, community, and WWM in Oracle) by running either the configuration wizard or WPSconfig connect-database task.

Note: We are not going to connect node 2 to the release database domain currently stored in Oracle or the JCR database domain stored in DB2. As previously mentioned in Table 5-1 on page 237, the release and JCR database domains cannot be shared (unless portal servers are in a cluster). Therefore we opted to leave the release and JCR database domains for node 2 in Cloudscape. In a clustered production environment however, you would typically have your release and JCR database domains in a non-Cloudscape database being shared by all cluster nodes.

The configuration wizard can be found in <portal_server_root>/config/wizard. Run the configwizard.bat file to start the wizard.

Alternatively you can run the following command from <portal_server_root>/config/ directory (substituting the names of the database domains you wish to connect to):

```
Windows: WPSconfig.bat connect-database
-DActDbDomainList=likeminds,customization,community,wmm
UNIX: ./WPSconfig.sh connect-database
-DActDbDomainList=likeminds,customization,community,wmm
```

See the Infocenter documented entitled “Connecting to existing database domains” for more information about the command line options.

Important: In our testing we found some problems with the configuration wizard. Specifically it appeared that the wizard did not correctly read some of the values we previously specified in the wpconfig_dbtype, dbdomain, and sourcedb properties files.

As a result, we re-entered some values by hand in the configuration wizard. You may therefore elect to run the wpconfig connect-database task instead.

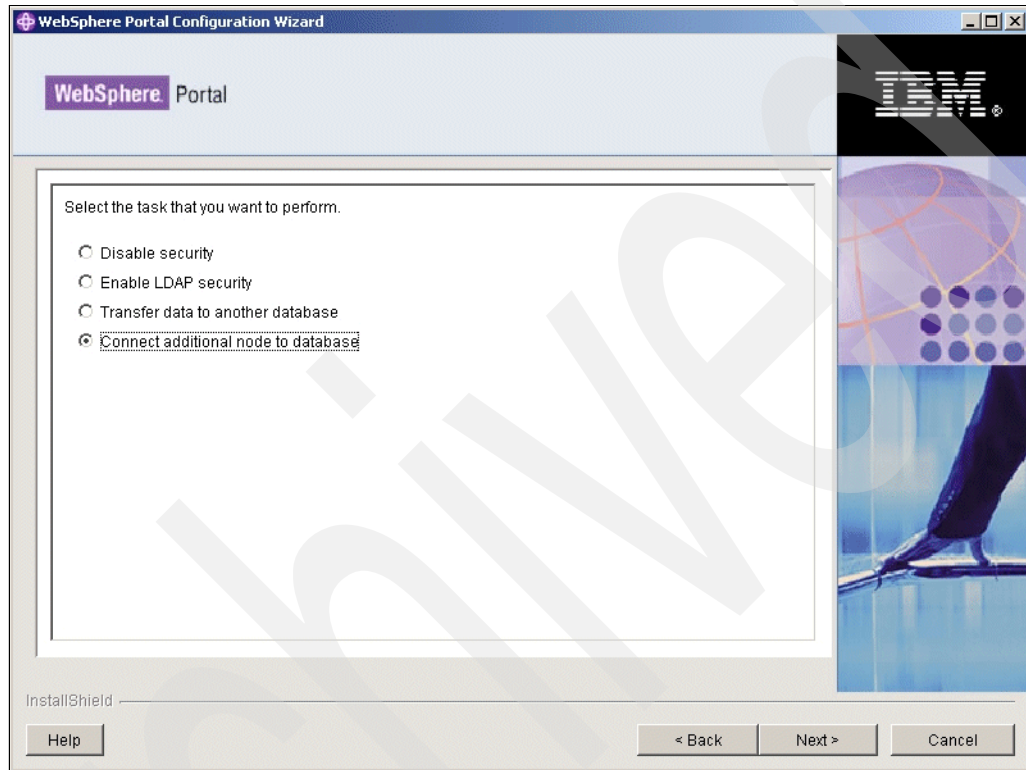


Figure 5-42 Selecting to connect an additional node to a database

As we are not transferring any data from node 2 in this example, we simply selected **Connect additional node to a database**.

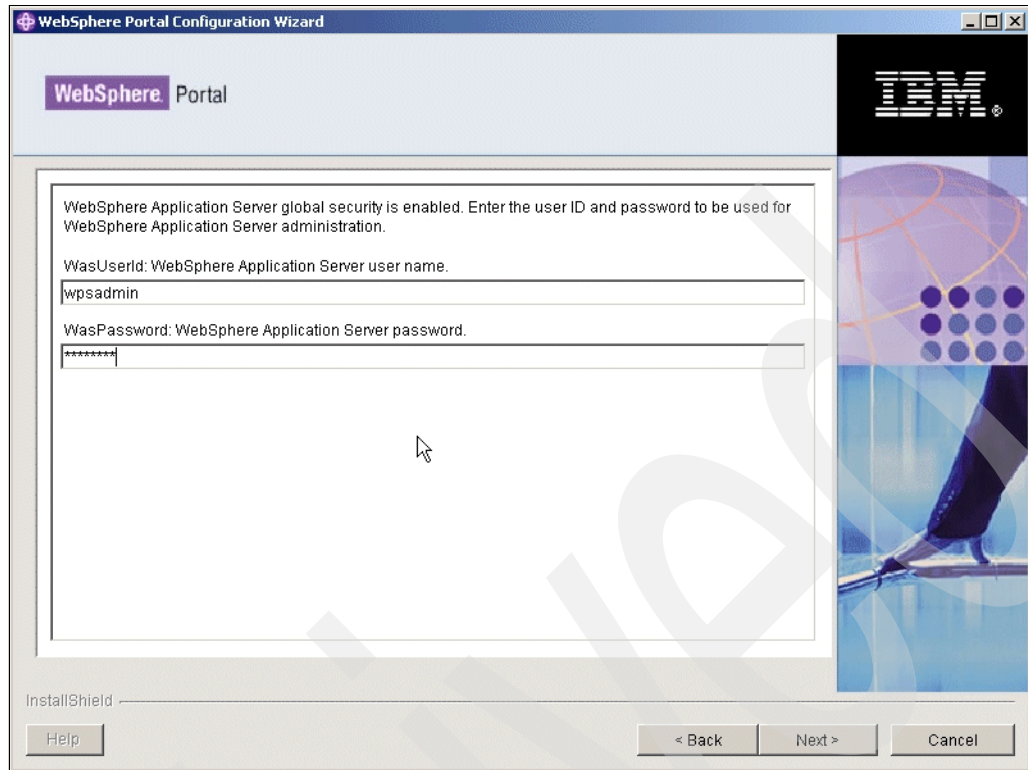


Figure 5-43 Specify the WebSphere Application Server credentials

We specified the WebSphere Application Server Administrator user name and password as shown in Figure 5-43.

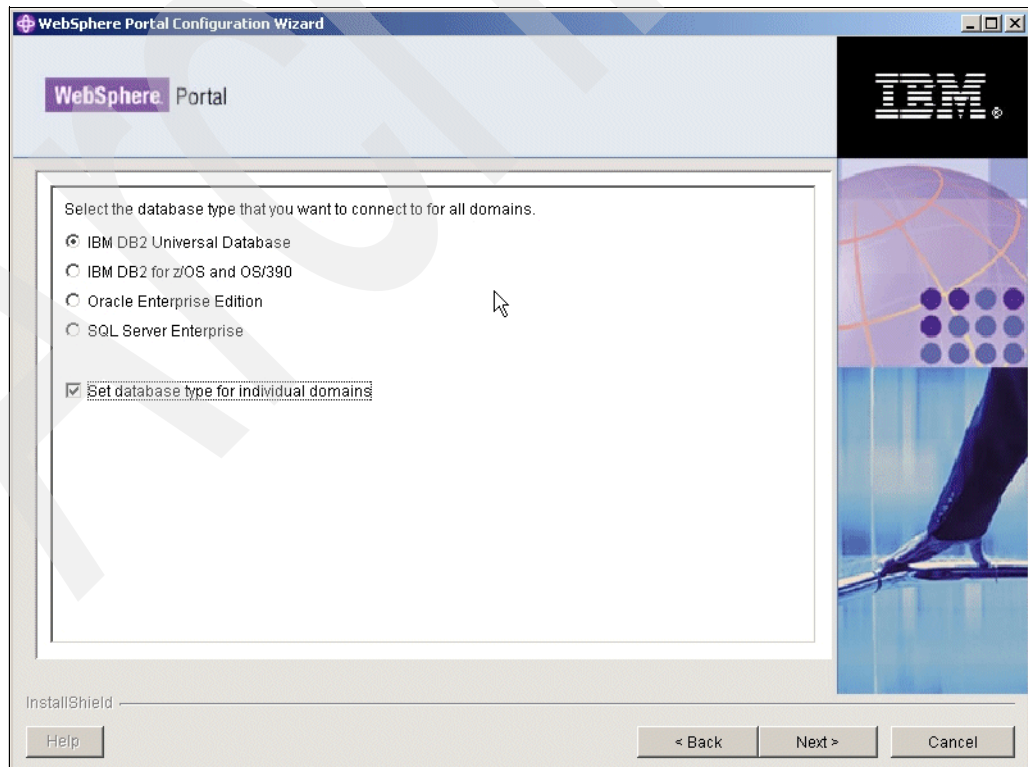


Figure 5-44 Select "Set database type for individual domains"

Select **Set database type for individual domains** as shown in Figure 5-44 on page 283. This option allows us to specify the individual database domains we want to connect to.

Important Tip: Figure 5-44 on page 283 may be the cause of some confusion. Specifically the radio buttons for the different types of database servers may lead you to believe that there is no way to specify both DB2 and Oracle at the same time.

Our initial thought when using the configuration wizard was that we may have to run it more than once (for example, once for DB2 and once for Oracle); however, what we discovered is that as long as you select **Set database type for individual domains**, the radio buttons are ignored and the configuration wizard lets you select different database servers for the various database domains (as shown in Figure 5-45).

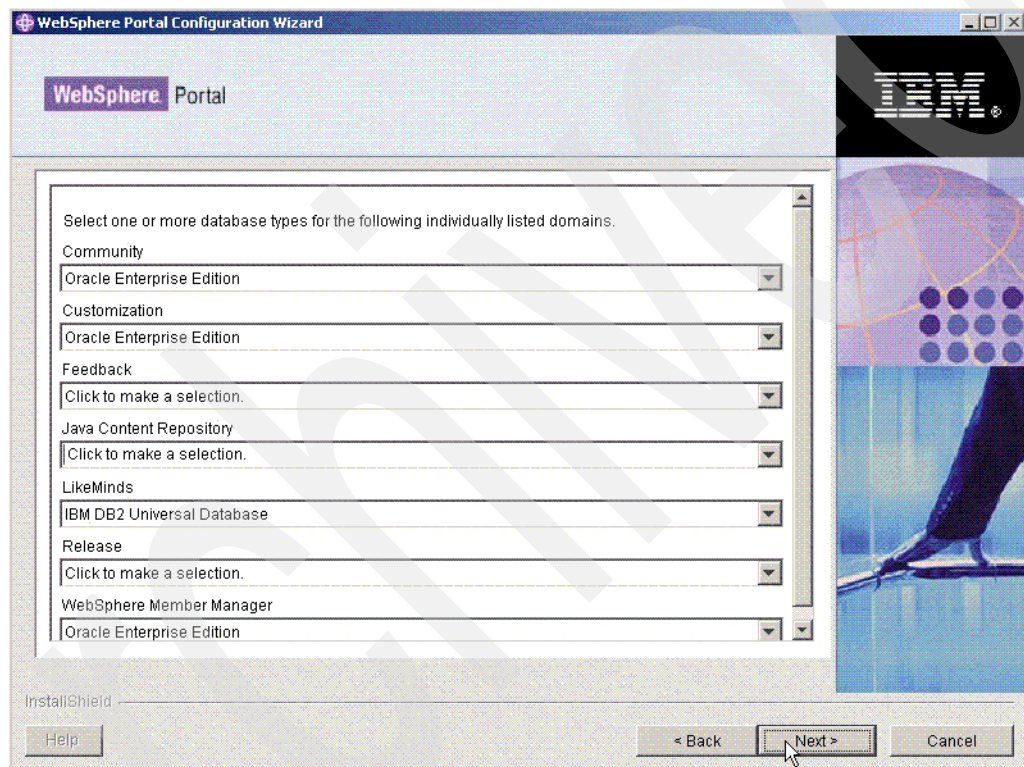


Figure 5-45 Select individual database domains to connect to

Figure 5-45 shows how we selected the locations of the database domains we wanted to connect to. Notice how the release database domain is left as the default “Click to make a selection”. This is because as previously mentioned in Table 5-1 on page 237, we cannot share release data amongst portal servers unless they are all in the same cluster. This is why we elected to leave node 2’s release database domain in Cloudscape and not connect it to node 1’s release database domain in Oracle.

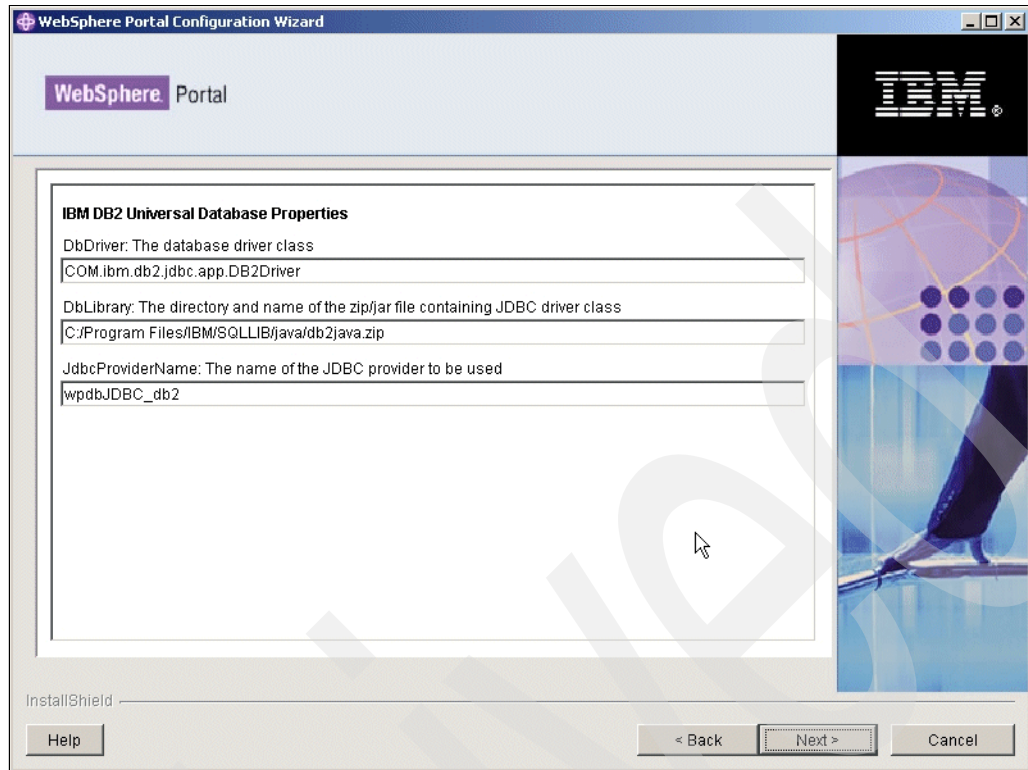


Figure 5-46 Specify connection properties for DB2

Figure 5-46 shows the DB2 connection properties we previously specified in the `wpconfig.dbytype.properties` file. As mentioned earlier, we found some inconsistencies with the configuration wizard and had to enter some of this information again manually.

We encountered the same issue with Figure 5-47 on page 286 for the Oracle connection properties. Again, we had to manually re-enter some of this information.

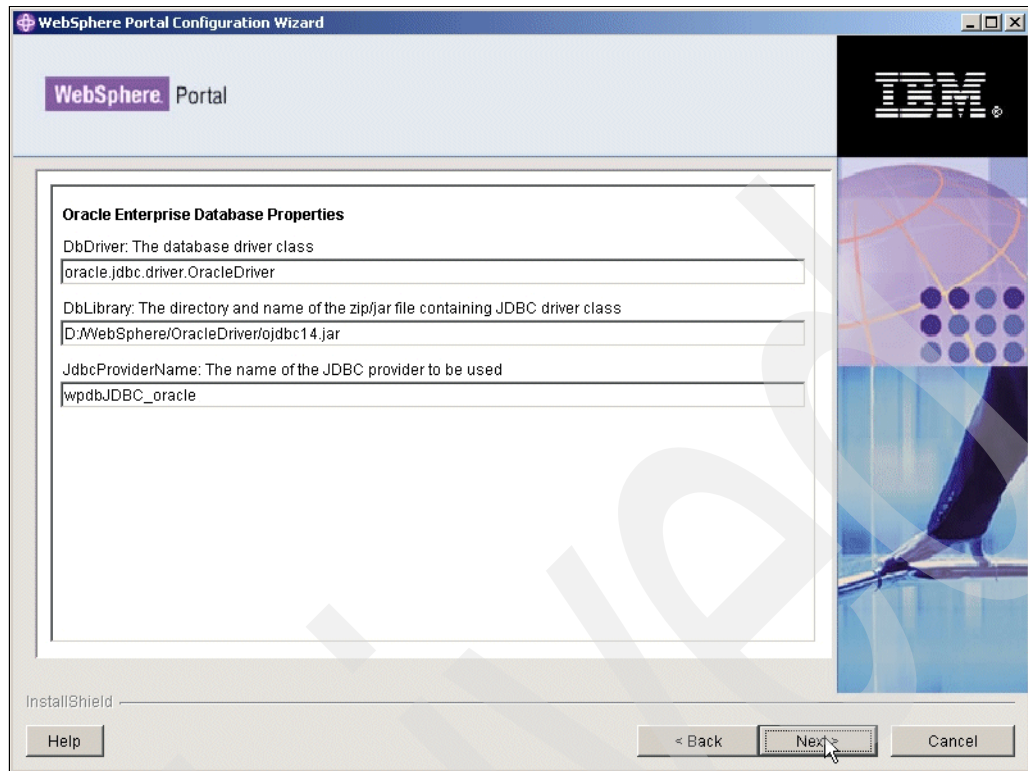
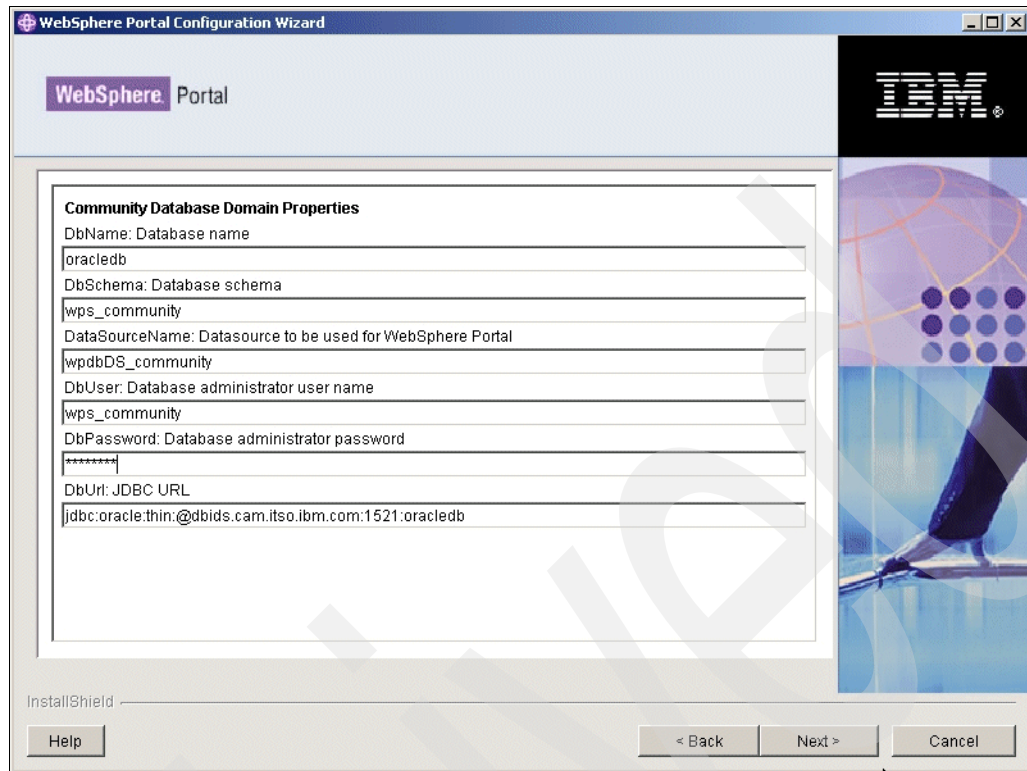


Figure 5-47 Specifying connection properties for Oracle

Figure 5-48 on page 287 shows the properties for the first database domain we want to connect node 2 to. In our case this was the community database domain currently stored in Oracle.

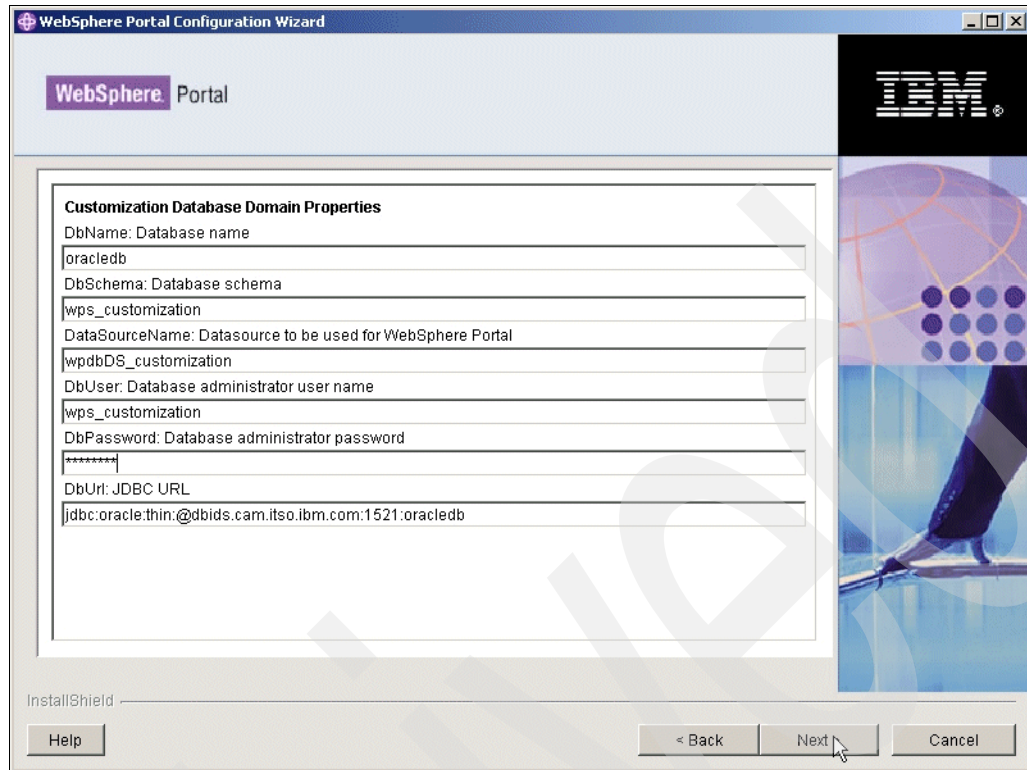


The image shows a screenshot of the 'WebSphere Portal Configuration Wizard' window. The title bar reads 'WebSphere Portal Configuration Wizard'. The main window has a header with the 'WebSphere Portal' logo and the IBM logo. The central area is titled 'Community Database Domain Properties' and contains several text input fields with labels: 'DbName: Database name' (value: 'oracledb'), 'DbSchema: Database schema' (value: 'wps_community'), 'DataSourceName: Datasource to be used for WebSphere Portal' (value: 'wpdbDS_community'), 'DbUser: Database administrator user name' (value: 'wps_community'), 'DbPassword: Database administrator password' (value: '*****'), and 'DbUrl: JDBC URL' (value: 'jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb'). At the bottom, there is an 'InstallShield' progress bar, a 'Help' button, and navigation buttons '< Back', 'Next >', and 'Cancel'.

Figure 5-48 Specify properties for the community database domain

Attention: We found some inconsistencies with the configuration wizard and had to enter some of this information again manually. This condition could continue to occur in future configuration wizard panels. Please ensure that all values are populated in the panels before clicking Next.

Make sure you specify a valid user name and password to connect to the Oracle database; otherwise, the database connect task will not be successful.



The image shows a screenshot of the 'WebSphere Portal Configuration Wizard' window. The title bar reads 'WebSphere Portal Configuration Wizard'. The window has a header with the 'WebSphere Portal' logo on the left and the IBM logo on the right. The main content area is titled 'Customization Database Domain Properties'. It contains several labeled text input fields: 'DbName: Database name' (value: 'oracledb'), 'DbSchema: Database schema' (value: 'wps_customization'), 'DataSourceName: Datasource to be used for WebSphere Portal' (value: 'wpdbDS_customization'), 'DbUser: Database administrator user name' (value: 'wps_customization'), 'DbPassword: Database administrator password' (value: '*****'), and 'DbUrl: JDBC URL' (value: 'jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb'). At the bottom of the window, there is an 'InstallShield' progress bar, a 'Help' button, and three navigation buttons: '< Back', 'Next >', and 'Cancel'. The 'Next >' button is highlighted with a mouse cursor.

Figure 5-49 Specify properties for the customization database domain

Figure 5-49 shows the properties for the next database domain we want to connect node 2 to. In our case this was the customization database domain, also currently stored in Oracle.

Make sure you specify a valid user name and password to connect to the Oracle database; otherwise, the database connect task will not be successful.

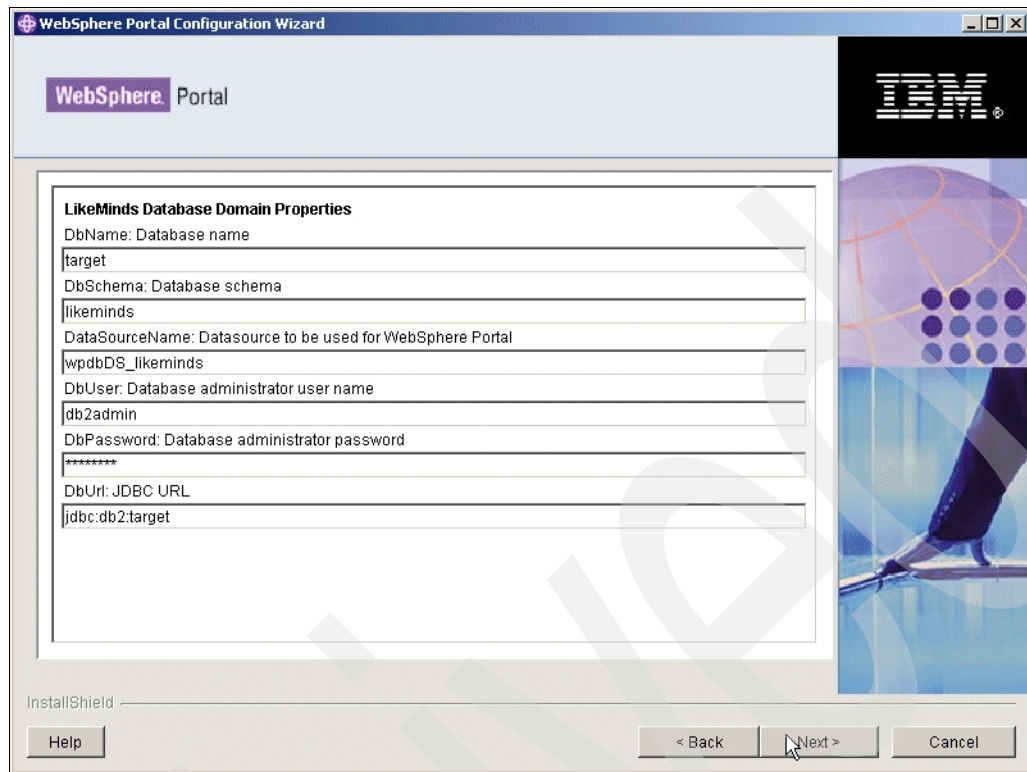


Figure 5-50 Specify properties for the Likeminds database domain

Figure 5-50 shows the properties for the next database domain we want to connect node 2 to. In our case this was the Likeminds database domain currently stored in DB2.

Make sure you specify a valid user name and password to connect to the DB2 database; otherwise, the database connect task will not be successful.

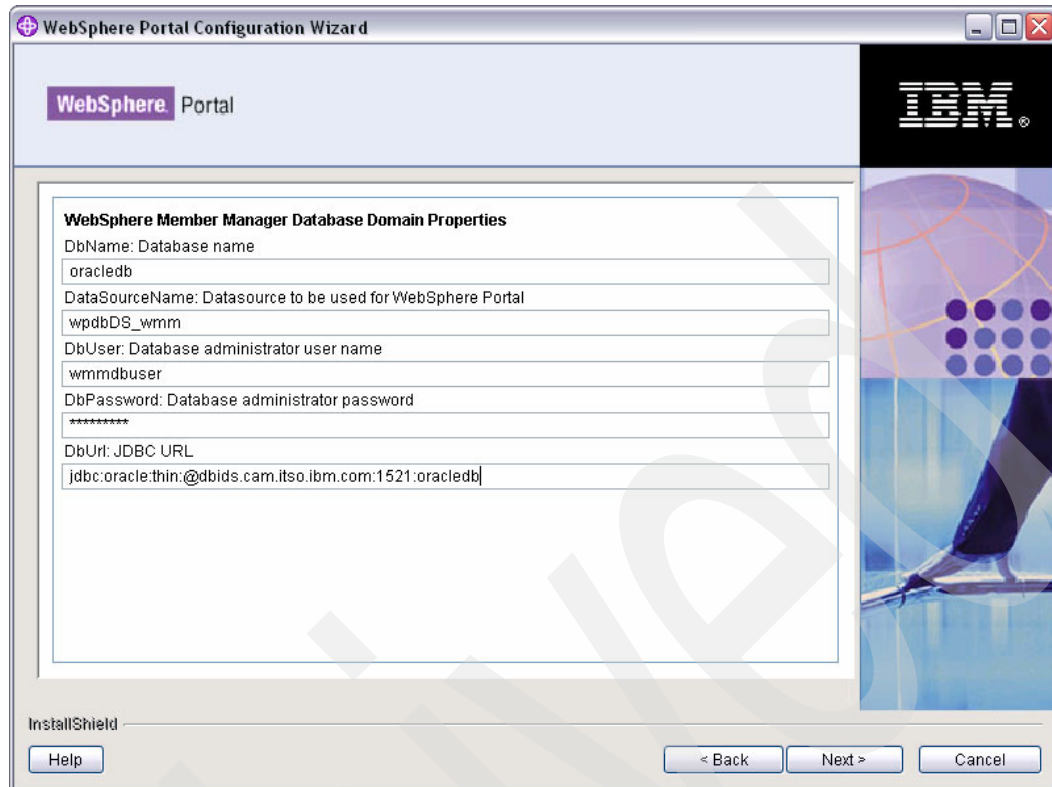


Figure 5-51 Specify properties for the WMM database domain

Figure 5-51 shows the properties for the final database domain we want to connect node 2 to. In our case this was the WMM database domain currently stored in Oracle.

As we were not using an external LDAP directory in our example, sharing the WMM database domain means that users should be able to log into any of the portal server nodes.

Make sure you specify a valid user name and password to connect to the DB2 database; otherwise, the database connect task will not be successful.

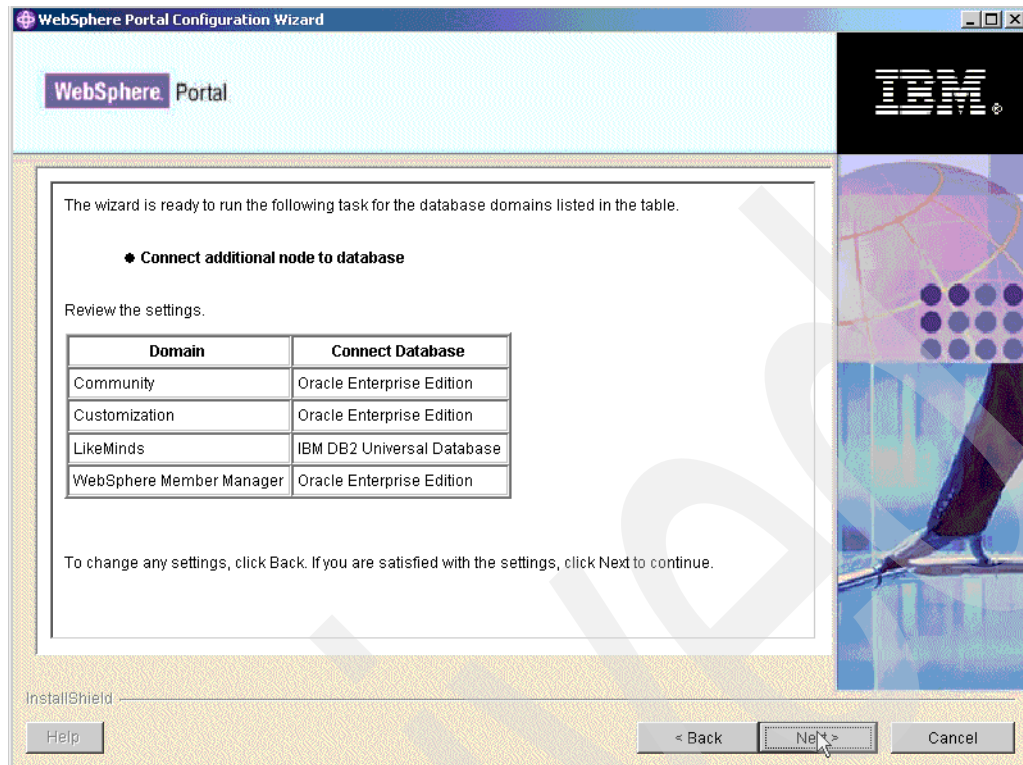


Figure 5-52 Summary of the database domains we wish to connect to

Finally, Figure 5-52 shows the Summary window of the database domains we want to connect node 2 to.

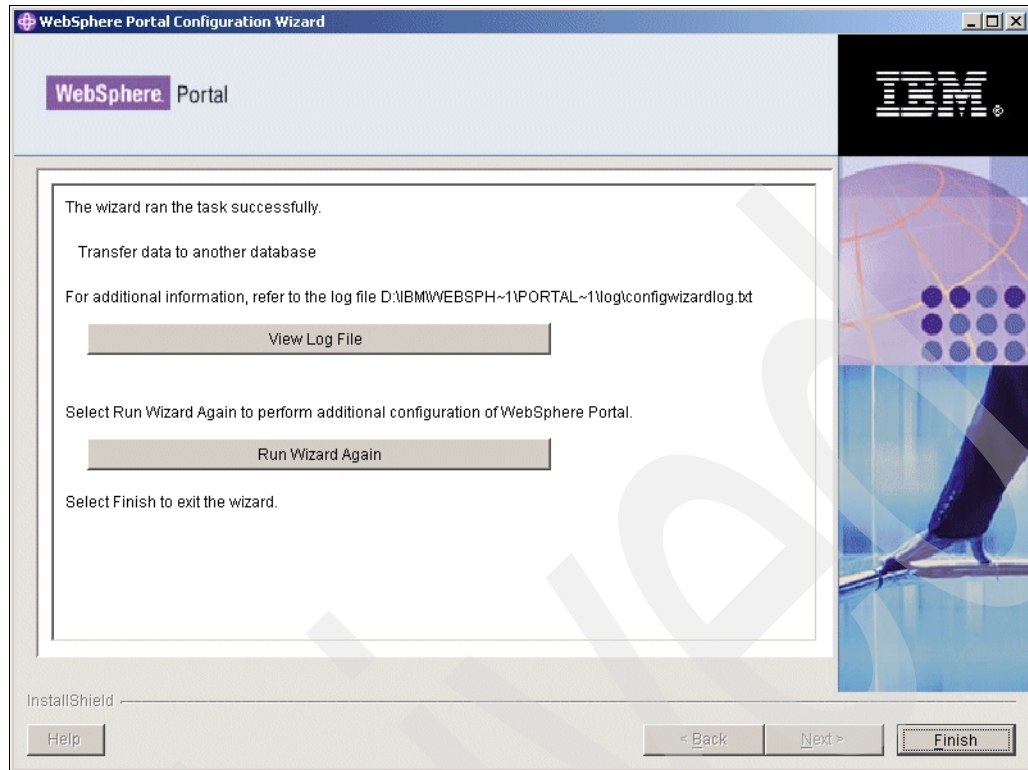


Figure 5-53 Successful database transfer

Testing our shared database domains

Having successfully completed the database domain connect task for node 2, our environment now looks like Figure 5-54.

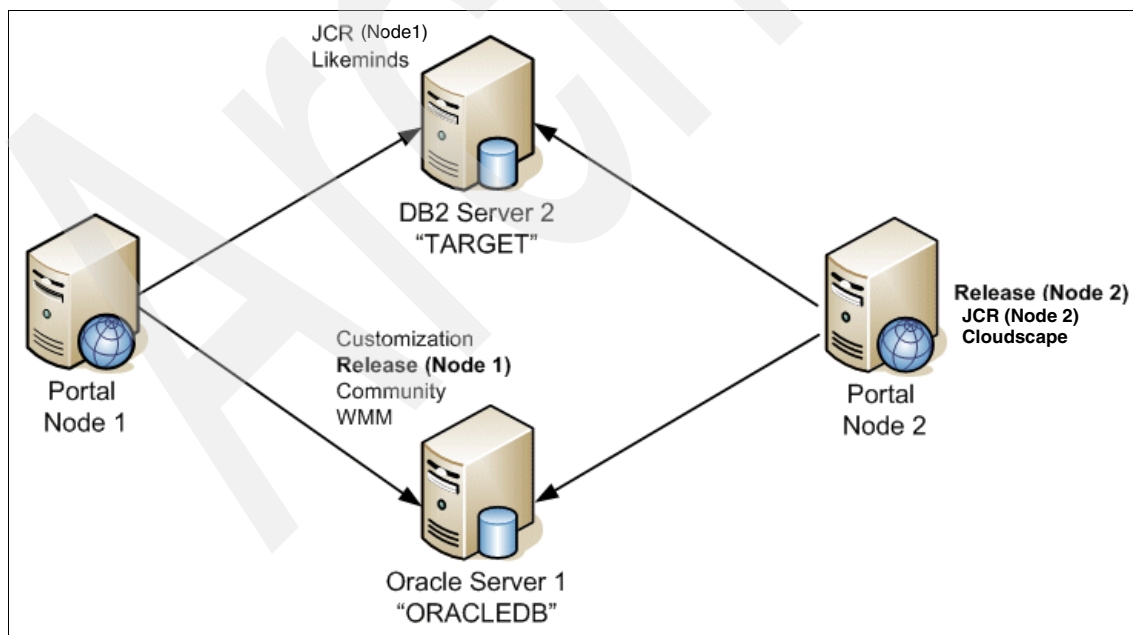


Figure 5-54 Example 3, sharing database domains amongst portal servers

We therefore conducted a number of simple test to ensure that node 2 was in fact successfully using the same content as node 1.

- **Test 1.** As previously mentioned, we created a test user called Jack Delaney (uid=jdelaney) on node 1.

Since the WMM database domain is shared between node 1 and node 2, we can successfully log into node 2 with our uid=jdelaney.

- **Test 2.** In order to test that our user customizations were being shared amongst both portal server nodes, we logged in as our test user on node 2 and modified some of his mail portlet settings.

We then logged out of node 2 and logged back in as our test user on node 1 and verified that we could still see our user customizations as shown in Figure 5-55 on page 293.

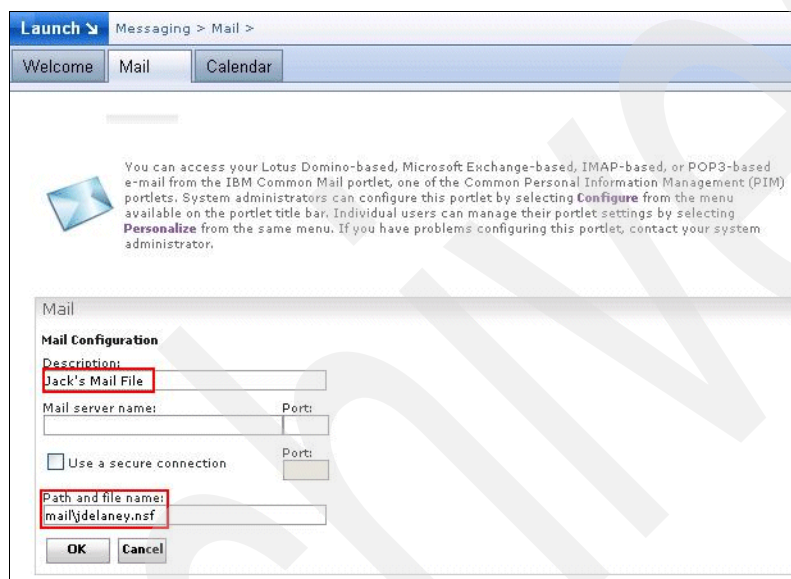


Figure 5-55 Testing our customization database domain

5.7 Example 4, sharing domains between separate clusters

Example 4 extends the ideas and steps discussed in example 3 in a very important way.

This time rather than getting multiple standalone portal server *nodes* to share a common set of database domains we are going to get multiple portal *clusters* to share a common set of database domains.

This allows us to operate a true 24x7 high-availability environment as described in Figure 5-3 on page 239 and Example 5-1 on page 240.

5.7.1 A quick recap of example 3

In 5.6, “Example 3, sharing domains amongst Portal nodes” on page 275, we successfully connected two standalone portal servers to both DB2 and Oracle as shown in Figure 5-56 on page 294.

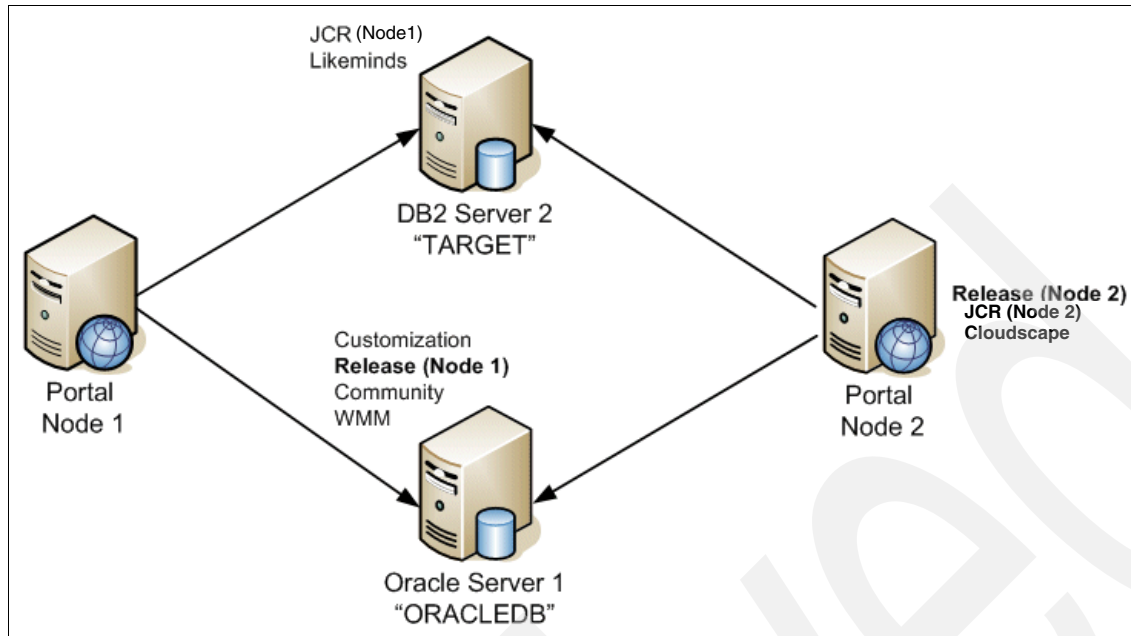


Figure 5-56 Summary of example 3, sharing database domains amongst portal servers

In example 4, we will perform a procedure similar to example 3, but instead with a Vertical and a Horizontal cluster.

Following is the sequence of steps we use in example 4:

1. Build a vertical cluster as per the instructions in Chapter 3, "Different deployment scenarios: building clustered environments" on page 51 (in particular, look at Section 3.8, "Vertical cluster installation and configuration steps" on page 176 for specific information about how to create a vertical cluster node).
2. We configured the domains of this Vertical cluster to DB2 and Oracle databases as shown in the environment shown in Figure 5-57 on page 295.

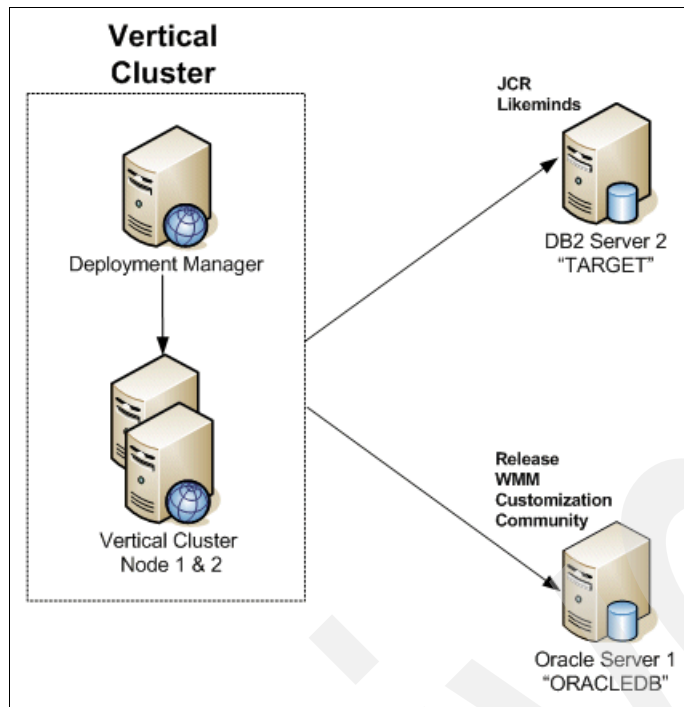


Figure 5-57 Connecting our Vertical cluster to DB2 to and Oracle

3. Finally we will take the Horizontal cluster that was built in Chapter 3, "Different deployment scenarios: building clustered environments" on page 51 and connect it to the same DB2 and Oracle databases that the Vertical cluster is currently using. This will result in our final environment looking like Figure 5-58 on page 296.

Important: The Horizontal cluster from Chapter 3 already had its own DB2 Database Server configured. This is depicted in Figure 5-58 on page 296 as "DB2 Server 3". Notice it has a portal database called WPDB.

In our example we left the release database domain for the Horizontal cluster on this existing DB2 Server 3, as release data cannot be shared.

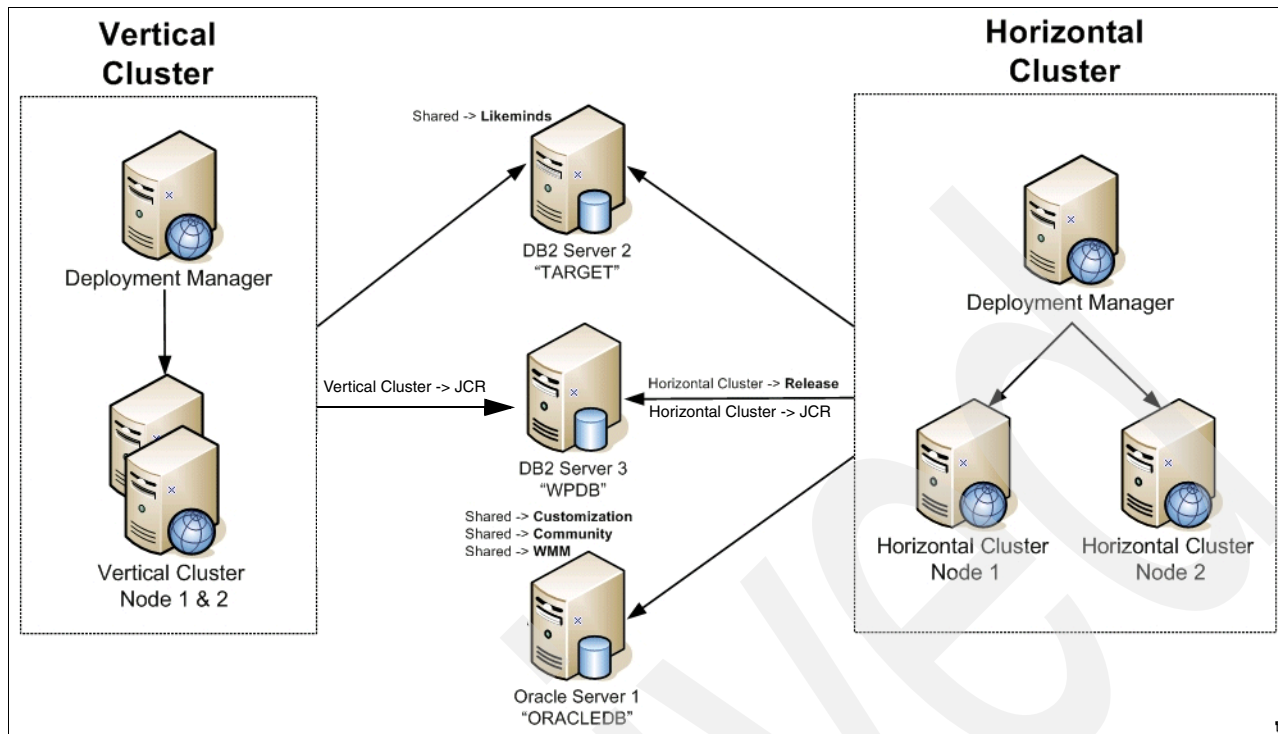


Figure 5-58 Connecting our Horizontal cluster to DB2 and Oracle

Note: In Figure 5-58 you will see that the Vertical cluster has a node 1 and a node 2. In reality there is only *one* physical node, with two JVMs running on it. The multiple nodes therefore refer to the individual JVMs that are running on the same physical server.

Steps for sharing database domains between portal clusters

The steps in this section describe how we connected the Horizontal cluster to the DB2 and Oracle Database Servers.

We will now reconfigure the Horizontal cluster to connect to the Vertical cluster's current database configuration as shown in Figure 5-57 on page 295. The only difference being that we will leave the Horizontal cluster configured to the Release Domain and JCR Domain on DB2 Server 3, because again, release data cannot be shared across separate clusters.

The connect-database task only needs to be run once since all cluster members (vertical or horizontal) share the same data source and driver via the WSAS configuration, which will get synced to all nodes in the cluster.

1. Shutdown WebSphere Portal on the primary node in the Horizontal cluster, but ensure the nodeagent is still running.
2. Make sure the Deployment Manager (DMGR) is running.
3. Prepare the primary node in the cluster, for both DB2 and Oracle. Although the primary node in the Horizontal cluster was already connected to a DB2 Database Server (DB2 Server 3), we had to prepare it for connecting to DB2 Server 2 and the Oracle Server 1. This involved the following steps:
 - a. On the primary node in the Horizontal cluster we cataloged the databases on DB2 Server 2 and ensured we could connect to the new TARGET database on DB2 Server 2 (See Chapter 3 for details on how to do this for DB2.). We repeated this for each

subsequent node in the Horizontal cluster. This ensured that each portal node in the cluster could connect to the new DB2 Server 2 and its TARGET database.

- b. On the primary node in the Horizontal cluster, we copied the relevant Oracle database driver and checked connectivity between portal and the Oracle database server and its database ORACLEDB (refer to your Oracle documentation for the steps on how to confirm database connectivity). Again, this was repeated for all the other portal nodes in the Horizontal cluster as well as the Deployment Manager to ensure that each portal node in the cluster could connect to the new Oracle Server 1 and its ORACLEDB database.
4. Edit the `wpconfig_dbtype.properties` on the primary node. We needed to edit this properties file in order to provide the configuration wizard (or connect-database task) with the location of the drivers that will allow it to connect to the destination DB2 and Oracle database servers/databases.

As the Horizontal cluster was already configured for DB2 Server 3, we only needed to specify details for how to connect to Oracle, as shown in **bold** in Example 5-10.

Tip: For completeness, it is a good idea to copy the `wpconfig_dbtype.properties` file to every portal node in the cluster. Back up the original `wpconfig_dbtype.properties` file before copying across the one from the primary node.

Example 5-10 Editing `wpconfig_dbtype.properties` for Oracle

```
#####
# Oracle Properties
#####

# DriverManager: The name of class SqlProcessor will use to import SQL files
oracle.DbDriver=oracle.jdbc.driver.OracleDriver

# DbLibrary: The directory and name of the zip/jar file containing JDBC driver
class
# Please use the system specific file separator names, e.g. for windows semicolon
and for unix colon.
oracle.DbLibrary=C:/IBM/WebSphere/OracleDriver/ojdbc14.jar

# JdbcProviderName: The name of jdbc provider to be used
oracle.JdbcProviderName=wpdbJDBC_oracle

#####
# END: Oracle Properties
#####
```

5. Unlike the previous examples, there was no need in this example to edit the `wpconfig.dbsource.properties` file. This is because we will not actually be transferring any *data* from CloudScape. We are simply pointing the Portal to an existing Database and re-building data sources via the WPCConfig connect-database task.
6. Edit `wpconfig_dbdomain.properties` file on the primary node.

In our case, we wanted to connect Likeminds to a different DB2 server (DB2 Server 2 with a database called TARGET). We also wanted to connect them to Community, Customization, and WMM to an Oracle Server (Oracle Server 1 with a database called "ORACLEDB").

The values we changed in the `wpconfig_dbdomain.properties` are as follows (and are shown in **bold** in Example 5-11 on page 298 for DB2 and Example 5-12 on page 299 for Oracle).

- ▶ **likeminds.DbType**. Specifies the type of database where the Domain(s) we want to connect are. In our case we wanted to connect to the Likeminds database domain in DB2
- ▶ **likeminds.DbName**. Specifies the name of the database where the Domain(s) we want to connect to are. In our case the Likeminds database domains is in a DB2 database called TARGET (see Figure 5-58 on page 296).
- ▶ **likeminds.DbSchema**. Specifies the unique name for the schema for this particular database domain.
- ▶ **likeminds.DataSourceName**. Specifies the unique name for the data source for this particular database domain.
- ▶ **likeminds.DbUrl**. Specifies how to connect to the database containing the database domains we want to connect to.
- ▶ **likeminds.DbUser**. Specifies the user name to use when connecting to the database where the database domains are.
- ▶ **likeminds.DbPassword**. Specifies the password to use for the previously specified user name.

Tip: Repeat step 6 for each of the database domains you want to connect to. Example 5-11 shows the values we used for the likeminds database domain.

Again, for completeness it is a good idea to copy the `wpconfig_dbdomain.properties` file to every portal node in the cluster. Back up the original `wpconfig_dbdomain.properties` file before copying across the one from the primary node.

Example 5-11 Editing `wpconfig_dbdomain.properties` for the LIKEMINDS domain

```
#####
# LIKEMINDS Database Properties
#####

# DbType: The type of database to be used for WebSphere Portal LIKEMINDS domain
likeminds.DbType=db2

# DbName: The name of the WebSphere Portal LIKEMINDS domain database
# Note: This value should also appear as the database element in DbUrl
# Note: Non-Windows platforms when using DB2 only. This value is the TCPIP Alias
# for the database.
likeminds.DbName=target

# DbSchema: The WebSphere Portal LIKEMINDS domain database schema name
# Follow the documentation of the target database management system
# in order to define a valid schema name as restrictions apply for
# some database management systems.
likeminds.DbSchema=likeminds

# DataSourceName: The name of datasource to be used for WebSphere Portal LIKEMINDS
domain
likeminds.DataSourceName=wpdbDS_likeminds

# DbUrl: The wp LIKEMINDS domain database URL
```

```

likeminds.DbUrl=jdbc:db2:target

# DbUser: The database administrator user ID
likeminds.DbUser=db2admin

# DbPassword: The database administrator password
likeminds.DbPassword=password

#####
# END: likeminds Database Properties
#####

```

Example 5-12 shows the values (in **bold**) we changed for the customization database domain.

- ▶ **customization.DbType**. Specifies the type of database where the Domain(s) we want to connect are. In our case we wanted to connect to the customization, community, and WMM database domains in Oracle.
- ▶ **customization.DbName**. Specifies the name of the database where the domain(s) we want to connect to are. In our case the community, customization, and WMM database domains are in an Oracle database called ORACLEDB (see Figure 5-58 on page 296).
- ▶ **customization.DbSchema**. Specifies the unique name for the schema for this particular database domain.
- ▶ **customization.DataSourceName**. Specifies the unique name for the data source for this particular database domain.
- ▶ **customization.DbUrl**. Specifies how to connect to the database domain.
- ▶ **customization.DbUser**. Specifies the user name to use when connecting to the database where the database domain is.
- ▶ **customization.DbPassword**. Specifies the password to use for the previously specified user name.

Tip: Repeat step 6 for each of the database domains you want to connect to. Example 5-12 shows the values we used for the customization database domain.

Remember, that it is good practice to copy the `wpconfig_dbdomain.properties` file to every portal node in the cluster. Back up the original `wpconfig_dbdomain.properties` file before copying across the one from the primary node.

Example 5-12 Editing the `wpconfig_dbdomain.properties` for the Customization Domain

```

#####
# Customization Database Properties
#####

# DbType: The type of database to be used for WebSphere Portal Customization
# domain
customization.DbType=oracle

# DbName: The name of the WebSphere Portal Customization domain database
# Note: This value should also appear as the database element in DbUrl
# Note: Non-Windows platforms when using DB2 only. This value is the TCP/IP
# Alias for the database.

```

```

customization.DbName=oracledb

# DbSchema: The WebSphere Portal Customization domain database schema name
#           Follow the documentation of the target database management system
#           in order to define a valid schema name as restrictions apply for
#           some database management systems.
customization.DbSchema=wps_customization

# DataSourceName: The name of datasource to be used for WebSphere Portal
# Customization domain
customization.DataSourceName=wpdbDS_customization

# DbUrl: The wp customization domain database URL
customization.DbUrl=jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb

# DbUser: The database administrator user ID
customization.DbUser=wps_customization

# DbPassword: The database administrator password
customization.DbPassword=password

#####
# END: Customization Database Properties
#####

```

7. Run the configuration wizard or WPConfig connect-database task.

For the database domains we want to connect to, we specified where we want to connect the Horizontal cluster to in order to use the existing database domains used by the Vertical cluster (for example, DB2 Server 2 and Oracle Server 1), except for the Release and JCR Domains.

As a result, we are now ready to connect to the database domains (in our case Likeminds in DB2 and customization, community and WMM in Oracle) by running either the configuration wizard or WPConfig connect-database task.

The configuration wizard can be found in <portal_server_root>/config/wizard. Run the configwizard.bat file to start the wizard.

Alternatively you can run the following command from <portal_server_root>/config/ directory (substituting the names of your database domains)

```

Windows: WPSconfig.bat connect-database
-DActDbDomainList=likeminds,community,customization,wmm
UNIX: ./WPSconfig.sh connect-database
-DActDbDomainList=likeminds,community,customization,wmm

```

See the Infocenter documented entitled “Connecting to existing database domains” for more information about the command line options.

Important: In our testing we found some problems with the configuration wizard. Specifically it appeared that the wizard did not correctly read some of the values we previously specified in the wpconfig_dbtype, dbdomain, and sourcedb properties files. As a result, we re-entered some values by hand in the configuration wizard. You may therefore elect to run the wpconfig connect-database instead.

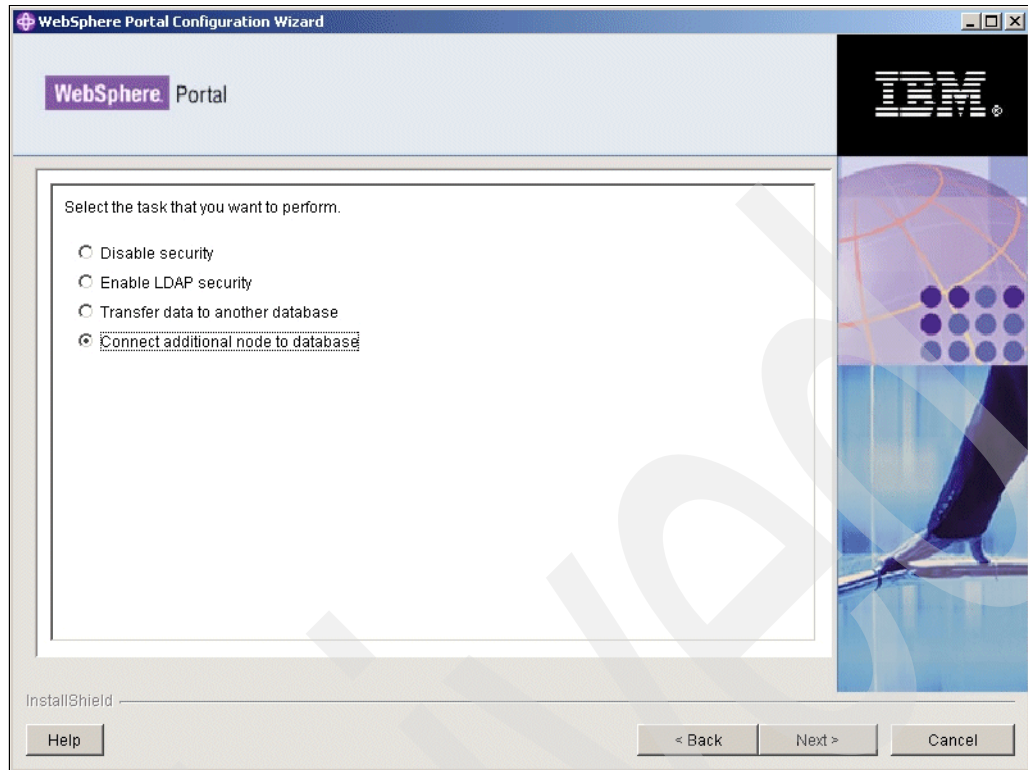


Figure 5-59 Select to connect an additional node to a database

As we are not transferring any data from the Horizontal cluster in this example, we simply selected **Connect additional node to a database**.

WebSphere Portal Configuration Wizard

WebSphere Portal

WebSphere Application Server global security is enabled. Enter the user ID and password to be used for WebSphere Application Server administration.

WasUserId: WebSphere Application Server user name.

wpsadmin

WasPassword: WebSphere Application Server password.

InstallShield

Help

< Back

Next >

Cancel

Figure 5-60 Specify the WebSphere Application Server credentials

We then specified the WebSphere Application Server Administrator user name and password as shown in Figure 5-60.

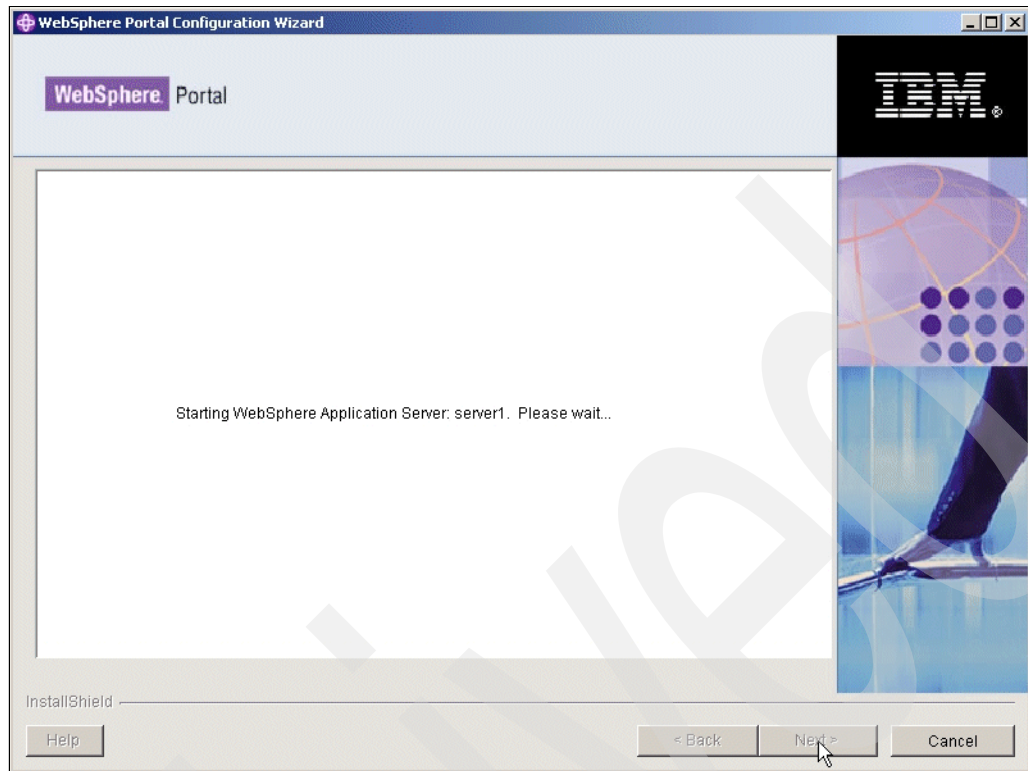


Figure 5-61 Validating WebSphere Administration Server credentials

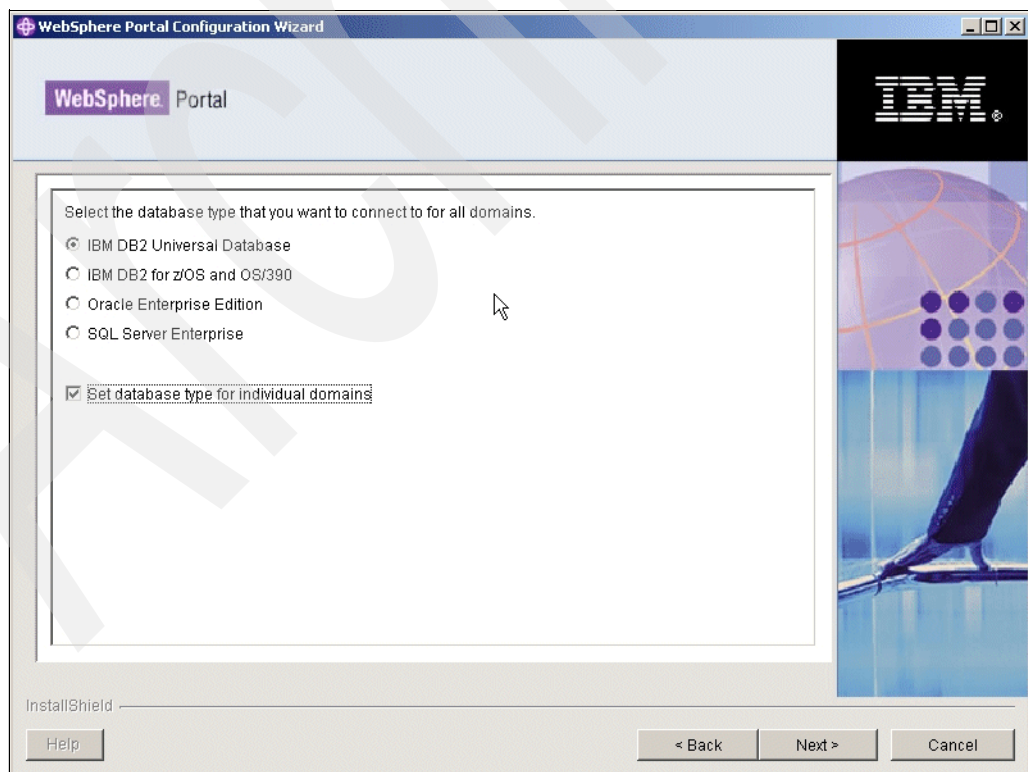


Figure 5-62 Select "Set database type for individual domains"

Select **Set database type for individual domains** as shown in Figure 5-62 on page 303. This option allows us to specify the individual database domains we want to connect to.

Important tip: Figure 5-62 on page 303 may be the cause of some confusion. Specifically the radio buttons for the different types of database servers may lead you to believe that there is no way to specify both DB2 and Oracle at the same time.

Our initial thought when using the configuration wizard was that we may have to run it more than once (for example, once for DB2 and Once for Oracle); however, what we discovered is that as long as you select **Set database type for individual domains**, the radio buttons are ignored and the configuration wizard lets you select different database servers for the various database domains (as shown in Figure 5-63).

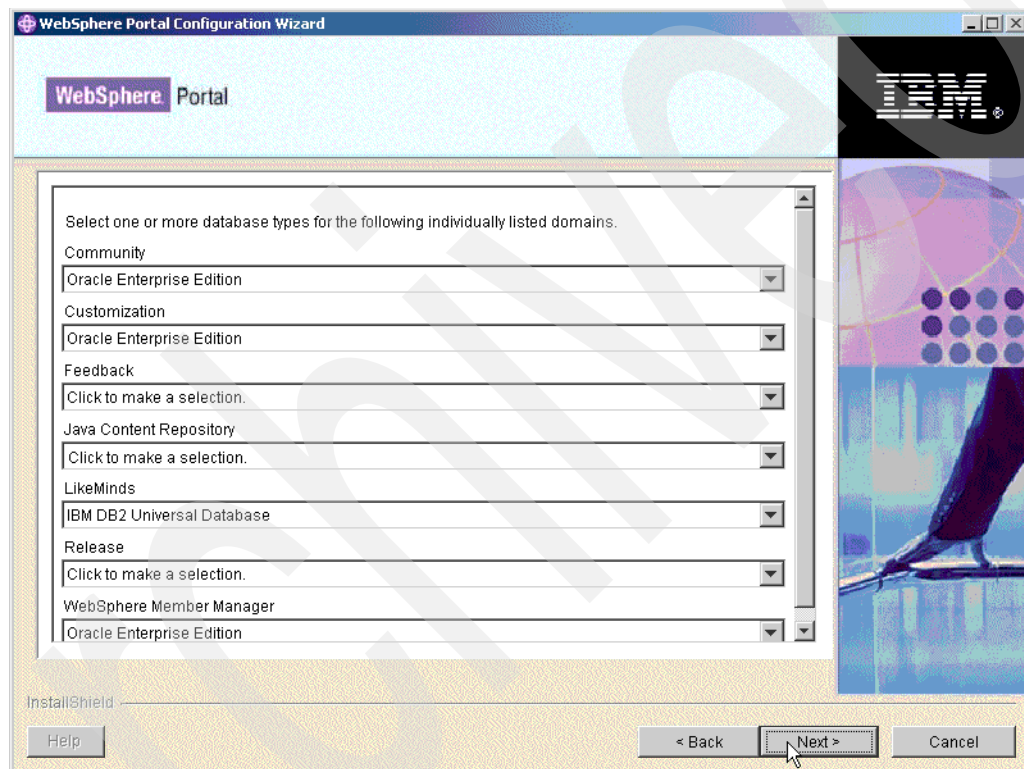


Figure 5-63 Select individual database domains to connect to

Figure 5-63 shows how we selected the locations of the database domains we wanted to connect to. Notice how the release database domain is left as the default option **Click to make a selection**. This is because as previously mentioned we elected to leave the release database domain in its original DB2 Server (DB 2 Server 3).

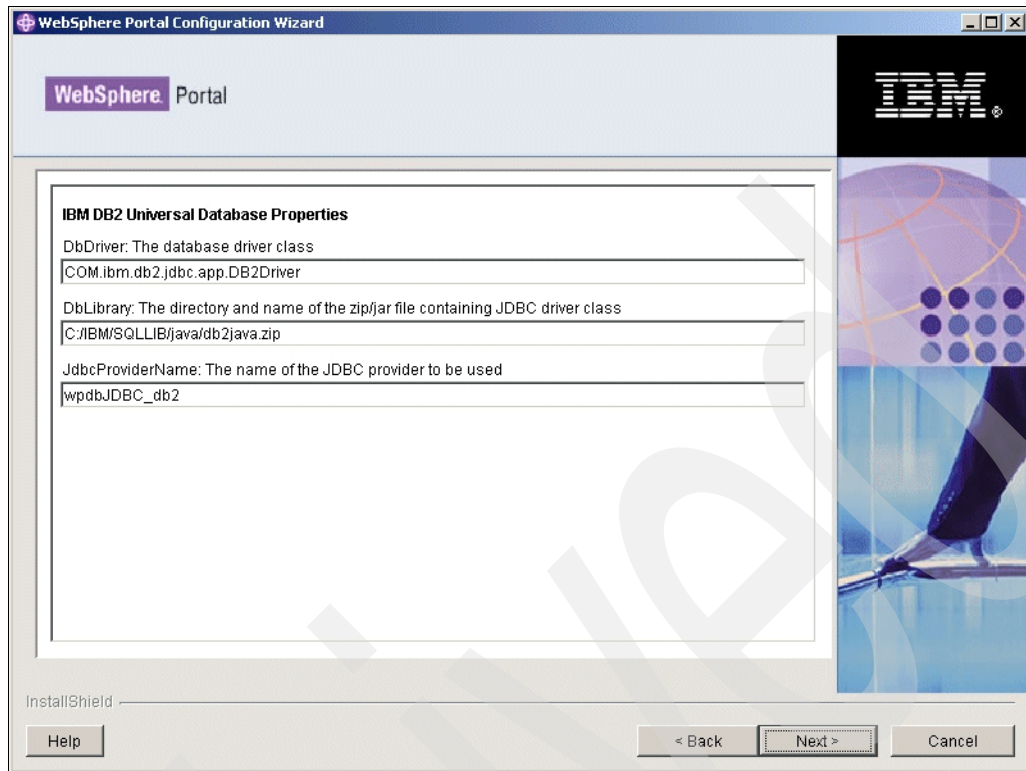


Figure 5-64 Specify connection properties for DB2

Figure 5-64 shows the DB2 connection properties we previously specified in the `wpconfig.dbytype.properties` file. As mentioned previously, we found some inconsistencies with the configuration wizard and had to enter some of this information again manually.

We encountered the same issue with Figure 5-65 on page 306 for the Oracle connection properties. We had to manually re-enter some of this information.

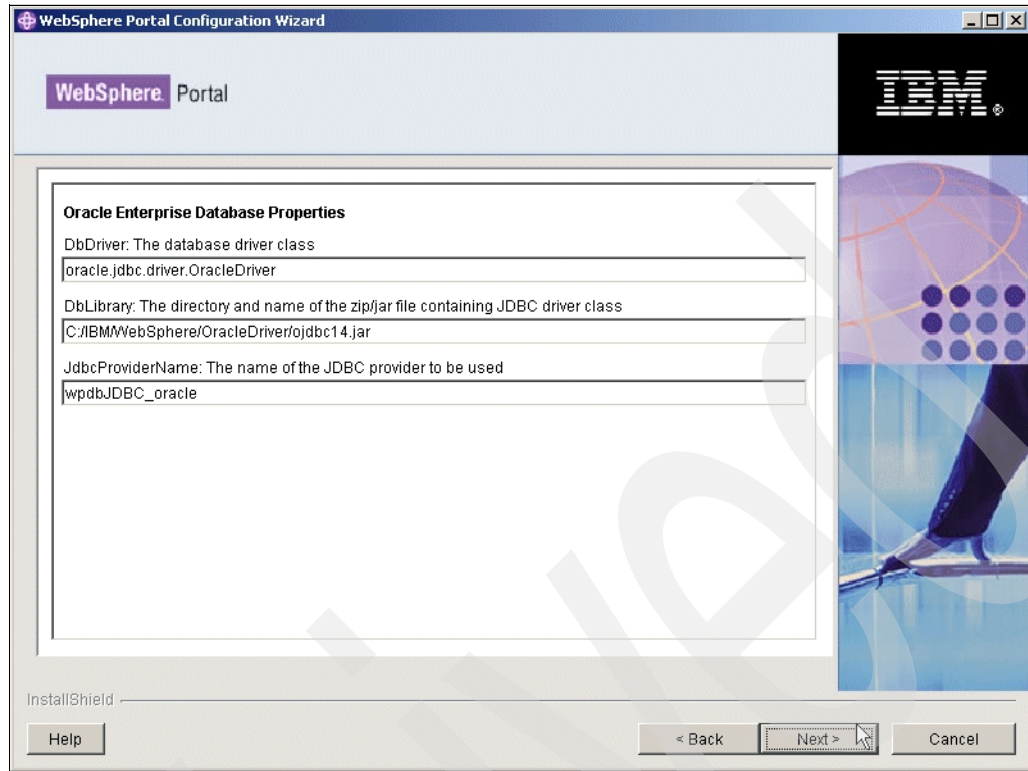
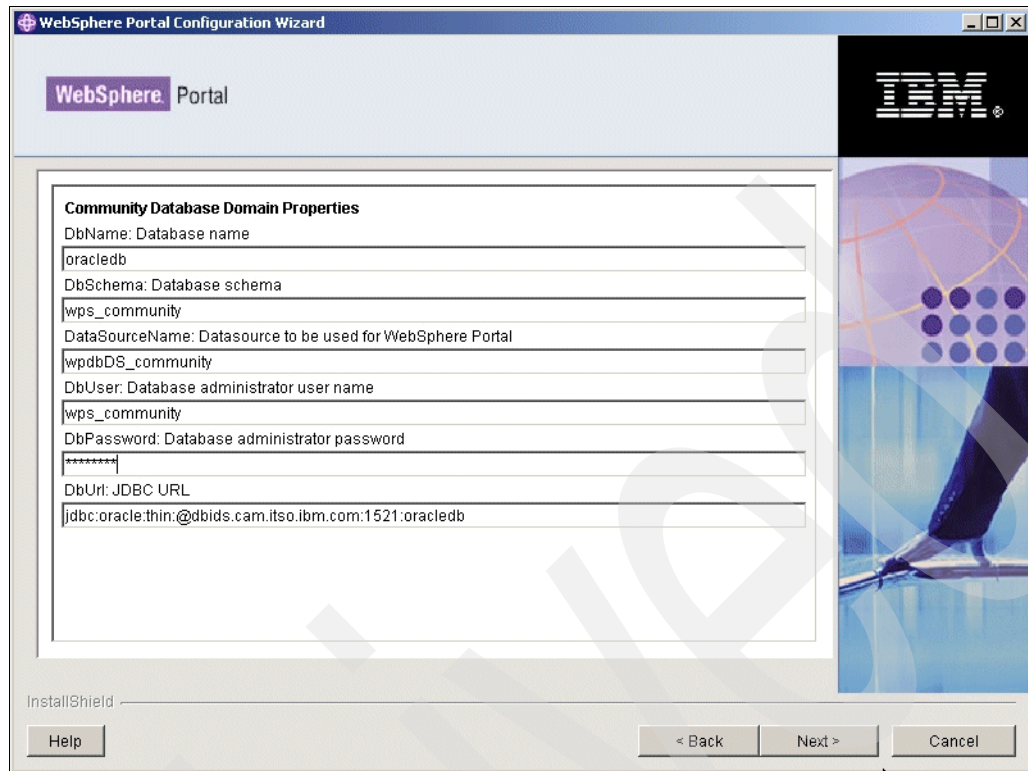


Figure 5-65 Specify connection properties for Oracle

Figure 5-66 on page 307 shows the properties for the first database domain we want to connect the Horizontal cluster to. In our case this was the community database domain, currently stored in Oracle.

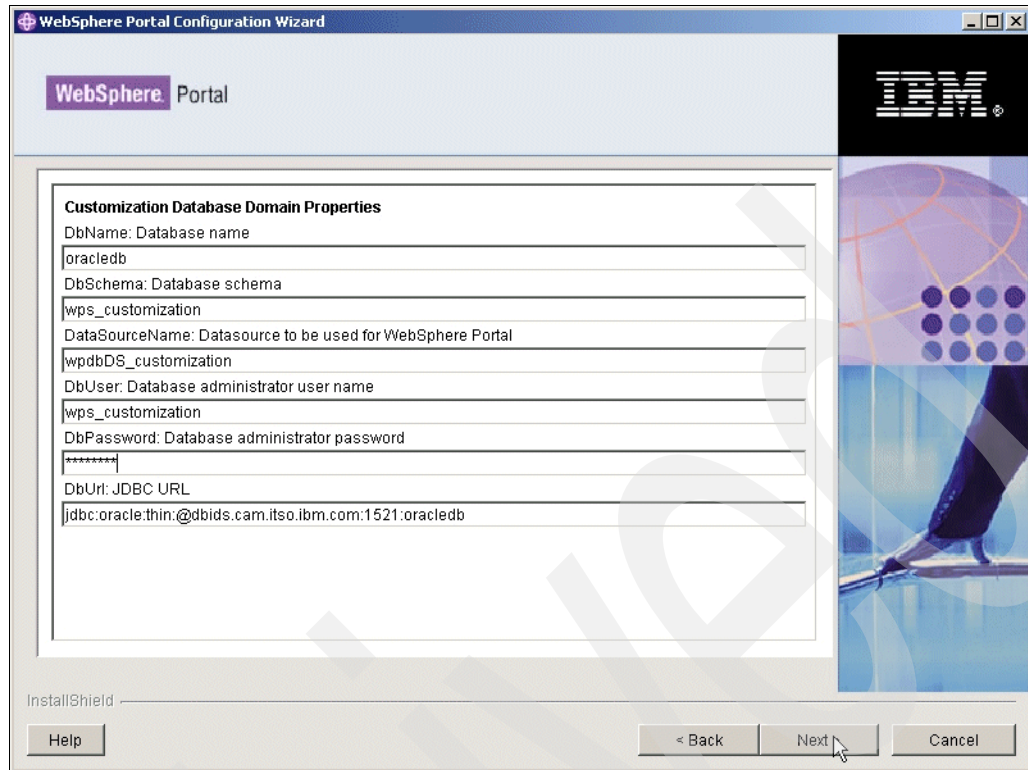


The image shows a screenshot of the 'WebSphere Portal Configuration Wizard' window. The title bar reads 'WebSphere Portal Configuration Wizard'. The main window has a header with the 'WebSphere Portal' logo on the left and the IBM logo on the right. Below the header, the title 'Community Database Domain Properties' is displayed. The form contains several labeled text input fields: 'DbName: Database name' with the value 'oracledb', 'DbSchema: Database schema' with the value 'wps_community', 'DataSourceName: Datasource to be used for WebSphere Portal' with the value 'wpdbDS_community', 'DbUser: Database administrator user name' with the value 'wps_community', 'DbPassword: Database administrator password' with masked characters '*****', and 'DbUri: JDBC URL' with the value 'jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb'. At the bottom left, there is an 'InstallShield' progress bar and a 'Help' button. At the bottom right, there are three buttons: '< Back', 'Next >', and 'Cancel'. A large, faint 'ARCHIVED' watermark is visible across the center of the image.

Property	Value
DbName: Database name	oracledb
DbSchema: Database schema	wps_community
DataSourceName: Datasource to be used for WebSphere Portal	wpdbDS_community
DbUser: Database administrator user name	wps_community
DbPassword: Database administrator password	*****
DbUri: JDBC URL	jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb

Figure 5-66 Specify properties for the community database domain

Make sure you specify a valid user name and password to connect to the Oracle database; otherwise, the database connect task will not be successful.



The image shows a screenshot of the 'WebSphere Portal Configuration Wizard' window. The title bar reads 'WebSphere Portal Configuration Wizard'. The main window has a header with the 'WebSphere Portal' logo and the IBM logo. The central area is titled 'Customization Database Domain Properties' and contains several text input fields with labels: 'DbName: Database name' (value: 'oracledb'), 'DbSchema: Database schema' (value: 'wps_customization'), 'DataSourceName: Datasource to be used for WebSphere Portal' (value: 'wpdbDS_customization'), 'DbUser: Database administrator user name' (value: 'wps_customization'), 'DbPassword: Database administrator password' (value: '*****'), and 'DbUrl: JDBC URL' (value: 'jdbc:oracle:thin:@dbids.cam.itso.ibm.com:1521:oracledb'). At the bottom, there is an 'InstallShield' progress bar, a 'Help' button, and navigation buttons labeled '< Back', 'Next >', and 'Cancel'. A mouse cursor is pointing at the 'Next >' button. The right side of the window features a decorative graphic with a blue and purple abstract design and the IBM logo.

Figure 5-67 Specify properties for the customization database domain

Figure 5-67 shows the properties for the next database domain we want to connect the Horizontal cluster to. In our case this was the customization database domain, also currently stored in Oracle.

Make sure you specify a valid user name and password to connect to the Oracle database; otherwise, the database connect task will not be successful.

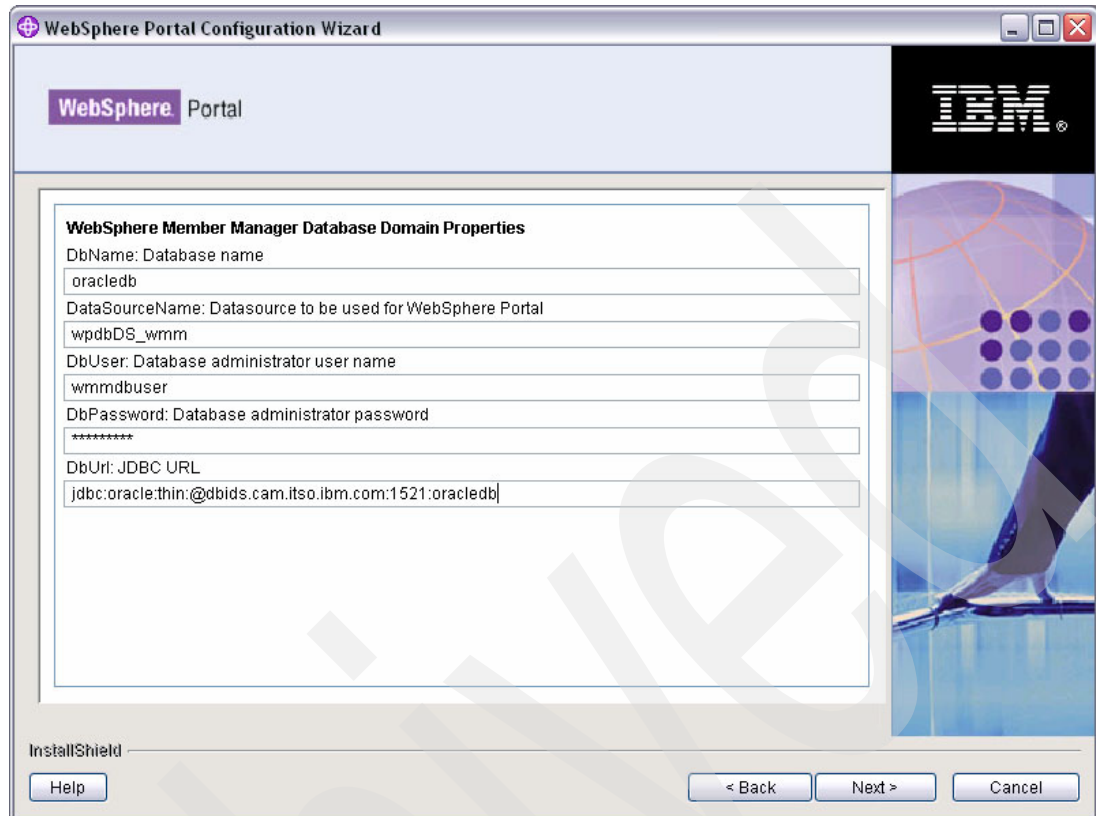
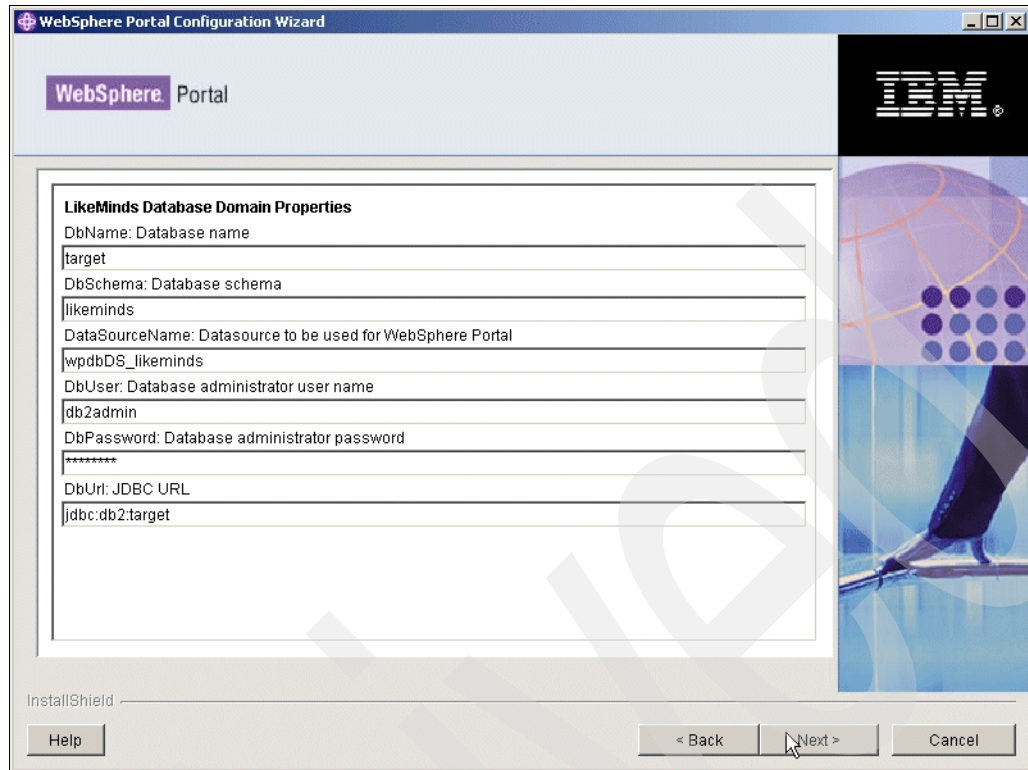


Figure 5-68 Specify properties for the WMM database domain

Figure 5-68 shows the properties for the next database domain we want to connect the Horizontal cluster to. In our case this was the WMM database domain, also currently stored in Oracle.

Make sure you specify a valid user name and password to connect to the Oracle database; otherwise, the database connect task will not be successful.



The image shows a screenshot of the 'WebSphere Portal Configuration Wizard' window. The title bar reads 'WebSphere Portal Configuration Wizard'. The main window has a header with the 'WebSphere Portal' logo and the IBM logo. The central area is titled 'LikeMinds Database Domain Properties' and contains several text input fields with labels: 'DbName: Database name' (value: target), 'DbSchema: Database schema' (value: likeminds), 'DataSourceName: Datasource to be used for WebSphere Portal' (value: wpdbDS_likeminds), 'DbUser: Database administrator user name' (value: db2admin), 'DbPassword: Database administrator password' (value: *****), and 'DbUrl: JDBC URL' (value: jdbc:db2:target). At the bottom, there is an 'InstallShield' progress bar, a 'Help' button, and navigation buttons '< Back', 'Next >', and 'Cancel'. A mouse cursor is pointing at the 'Next >' button. The right side of the window features a decorative graphic with a globe and a pen.

Property	Value
DbName: Database name	target
DbSchema: Database schema	likeminds
DataSourceName: Datasource to be used for WebSphere Portal	wpdbDS_likeminds
DbUser: Database administrator user name	db2admin
DbPassword: Database administrator password	*****
DbUrl: JDBC URL	jdbc:db2:target

Figure 5-69 Specify properties for the Likeminds database domain

Figure 5-69 shows the properties for the final database domain we want to connect the Horizontal cluster to. In our case this was the Likeminds database domain, currently stored in DB2.

Make sure you specify a valid user name and password to connect to the DB2 database; otherwise, the database connect task will not be successful.

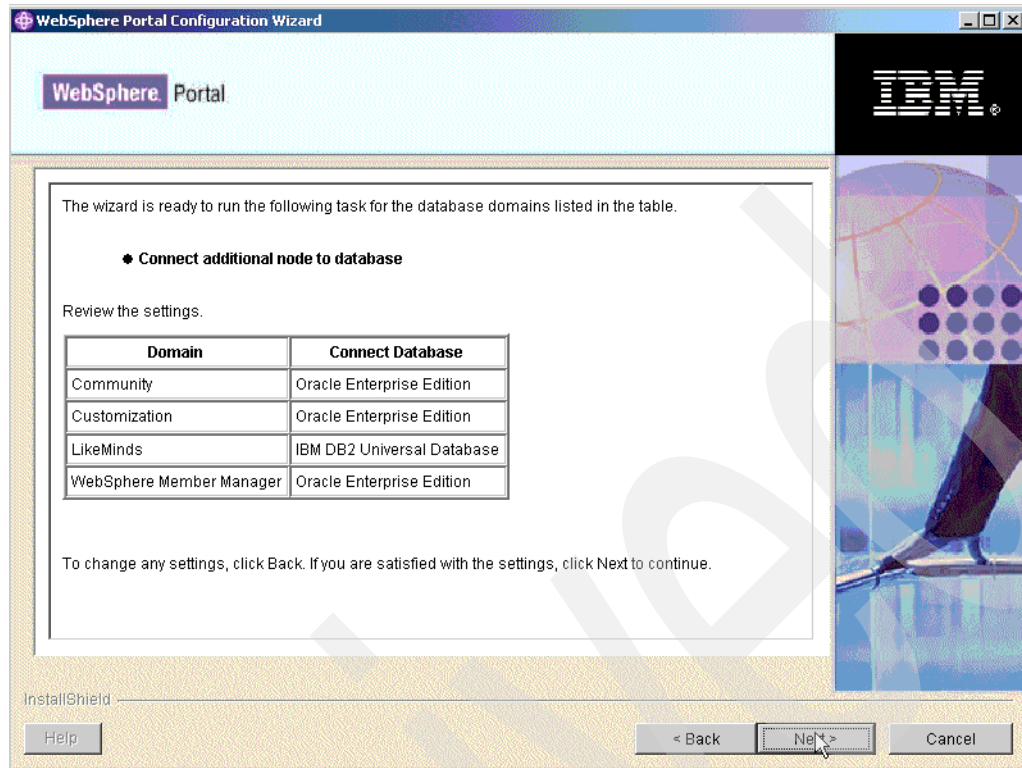


Figure 5-70 Summary of the database domains we want to connect to

Finally, Figure 5-70 shows the Summary window of all the database domains we want to connect the Horizontal cluster to.

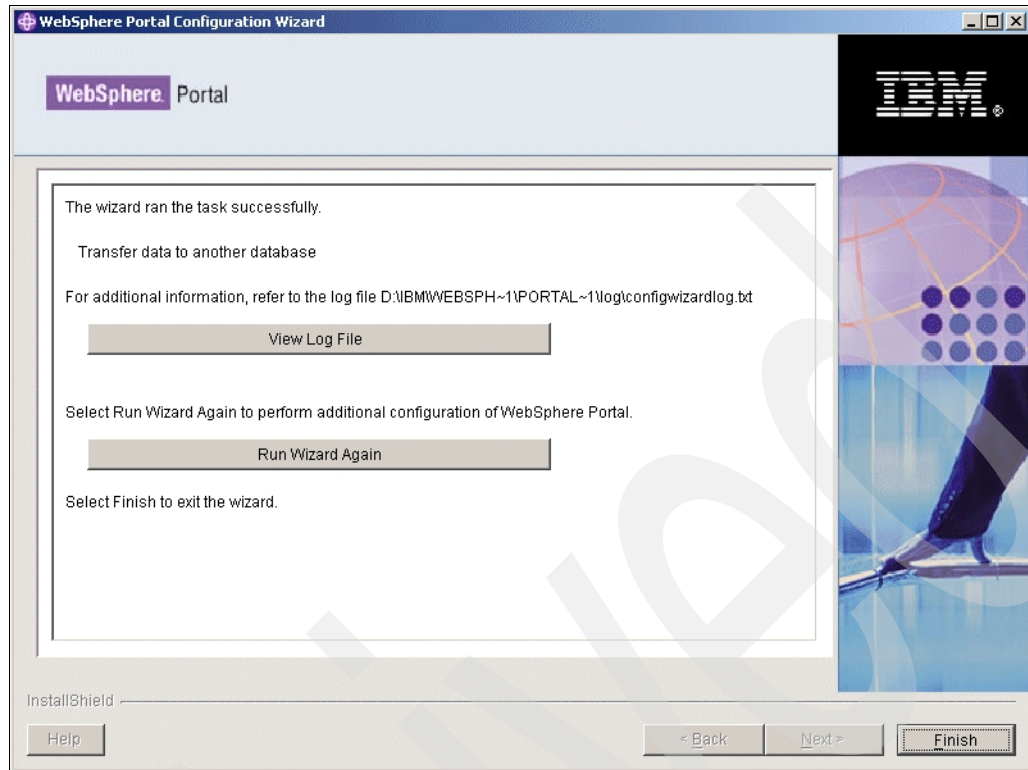


Figure 5-71 Successful database connect

Our environment now looks like Figure 5-72.

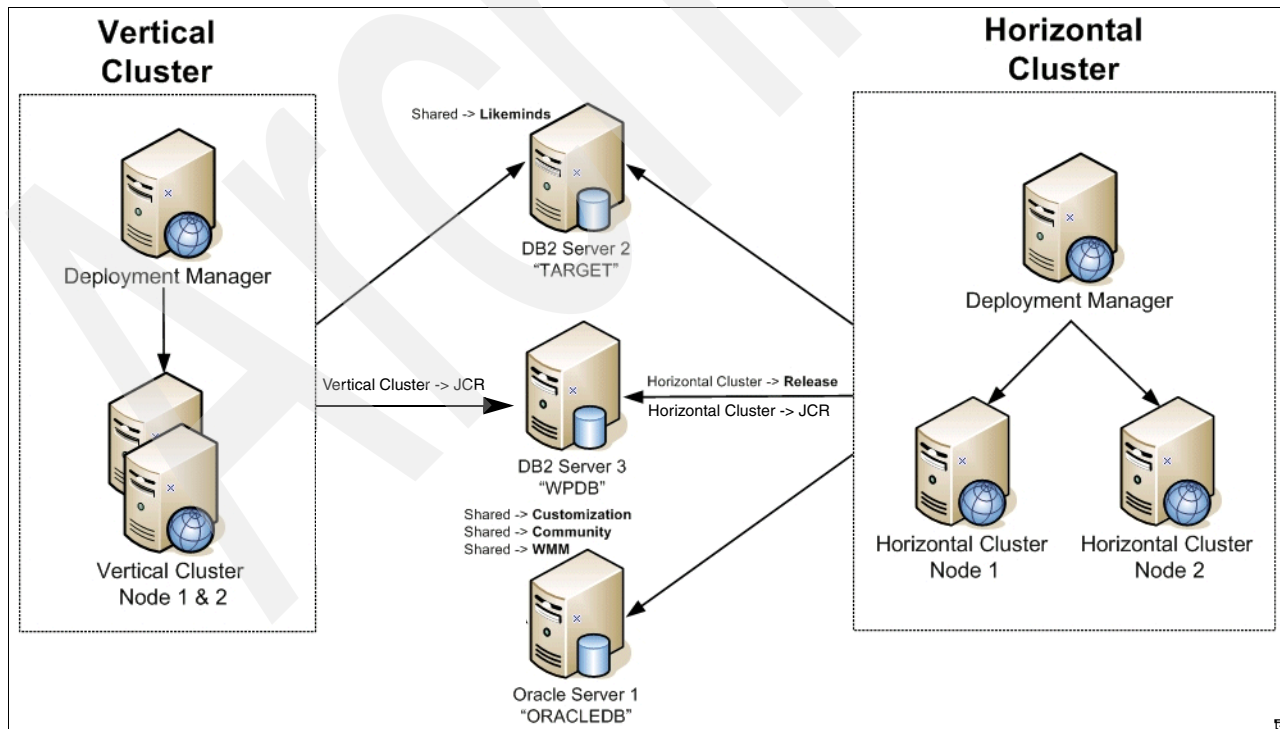


Figure 5-72 The final example 4 environment

5.8 Using database domains in a production environment

This section shows how we can take the techniques that we learned from the previous examples and apply them to building a typical high-availability production Portal environment. We outline each step involved and refer you back to the relevant section in this (and other) chapters for the particular configuration steps required.

The ultimate goal in this section is to build a multi-cluster, high-availability Portal environment that is sharing identical content. This will allow us to provide 24x7 operations and users will not be disrupted in the event of server maintenance or upgrades.

Building a multi-cluster high-availability portal environment

1. A typical starting point for building any portal environment is to install a Portal server (Portal node 1 in our case), using the default Cloudscape database. Portal node 1 is the primary node for Cluster A. Since we will be building a cluster you need to decide at this point on how to install Portal.
 - If you want to build a Process Server cluster, you will need to install Portal into a federated WSAS cell. See 3.4, “Horizontal cluster (includes WebSphere Process Server) installation and configuration steps” on page 54.
 - If you want to build a cluster without Process Server, you can install WSAS and Portal together as a standalone environment. See 3.5, “Horizontal cluster (without WebSphere Process Server) installation and configuration steps” on page 136.
2. After determining your cluster approach, the next step is to transfer our database domains (using either the configuration wizard or the database-transfer task) from Cloudscape to a supported Database Server Instance.

In our case we used DB2 for our database server as shown in Figure 5-73.

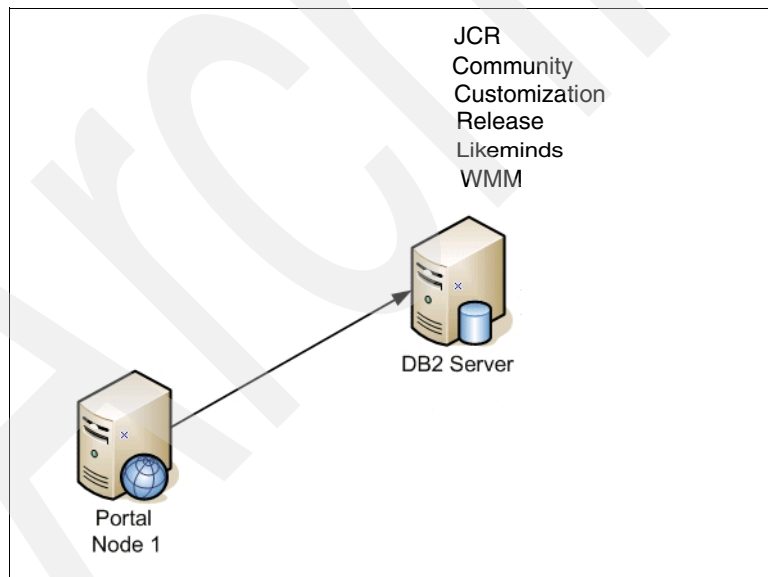


Figure 5-73 Transferring from Cloudscape to DB2

See Chapter 3, “Different deployment scenarios: building clustered environments” on page 51 for more information about how to transfer database domains from Cloudscape to DB2.

At this stage we can either transfer all or some of our database domains from Cloudscape to a supported database server. In a well designed Portal environment, where we want

high availability and good performance, we may want to isolate particular database domains. It is generally recommended and a best practice to isolate the JCR database domain. JCR data is a good candidate for isolation because it has unique usage patterns and capacity requirements in comparison to other database domains, and isolating it gives us far more flexibility for tuning and capacity management.

We can isolate the JCR domain in a number of ways:

- We can place the JCR domain on a physically separate database server. This may be applicable for an environment with large amounts of JCR data where we want a greater disaster recovery capability. *This is the method we chose in this scenario.*
- We can place the JCR domain on the same physical database server but isolate it by putting it in a database on a separate database instance as shown in Figure 5-74.
- We can place the JCR domain on the same physical database server but isolate it by putting it in a different database within the same DB2 instance as shown in Figure 5-75 on page 315.

Figure 5-74 shows one physical server with two instances of DB2 running. Each instance of DB2 has its own database containing our various schemas or domains.

We can see in Figure 5-74, that the JCR domain was isolated by being transferred from Cloudscape to its own DB2 instance and database (WPSDB2).

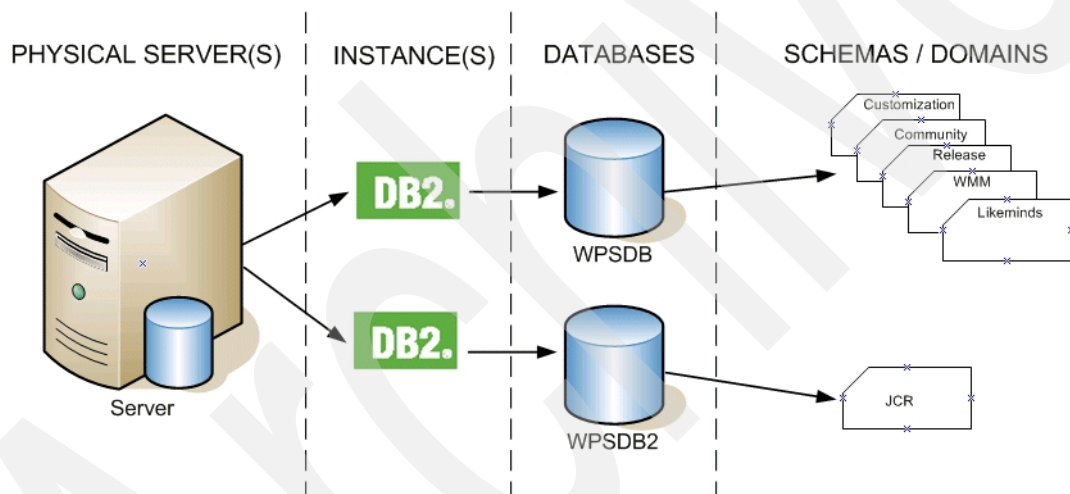


Figure 5-74 Isolating database domains by putting them in separate databases instances and separate databases

Alternatively we can place the JCR domain on the same physical database server and in the same database instance but have them in different databases as shown in Figure 5-75 on page 315.

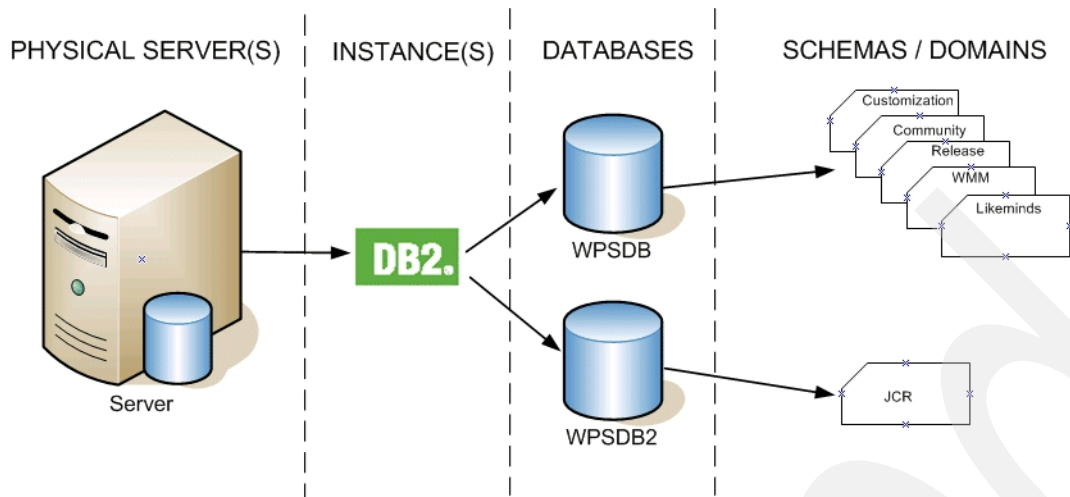


Figure 5-75 Isolating domains by putting them in the same database instance but separate databases

Figure 5-75 shows that the JCR domain was isolated by being transferred from Cloudscape to its own database (WPSDB2). Even though this database is being managed by the same DB2 instance as the other schemas or domains, it should provide for increased performance and capacity management.

We opted to put the JCR database domain on a separate physical server; however, one of the other options listed above may be more suitable for your environment and requirements.

3. Having decided on a method of isolating the JCR domain, the next step for building our high-availability Portal environment is to move the JCR Domain to its own database physical server (as shown in Figure 5-76 on page 316).

Note: Step 3 can also be conducted as part of the initial database transfer from Cloudscape to a supported database server (for example, as part of Step 2).

We opted to conduct this extra step of transferring the JCR domain from one database server to another to show you how this can be accomplished after the initial database transfer is performed. This approach may be required because business or technical considerations dictate the need to isolate domains after the initial database transfer occurs.

You may find it more convenient to isolate your domains as part of the initial database transfer from Cloudscape to multiple database servers by applying the techniques learned earlier in this chapter, specifically by editing the `wpconfig_dbDomain.properties` file to point the JCR domain to a different DB2 server than the other domains.

See the steps outlined in section 5.4, “Example 1, moving database domains” on page 243 for how to move the JCR Domain to a new database server.

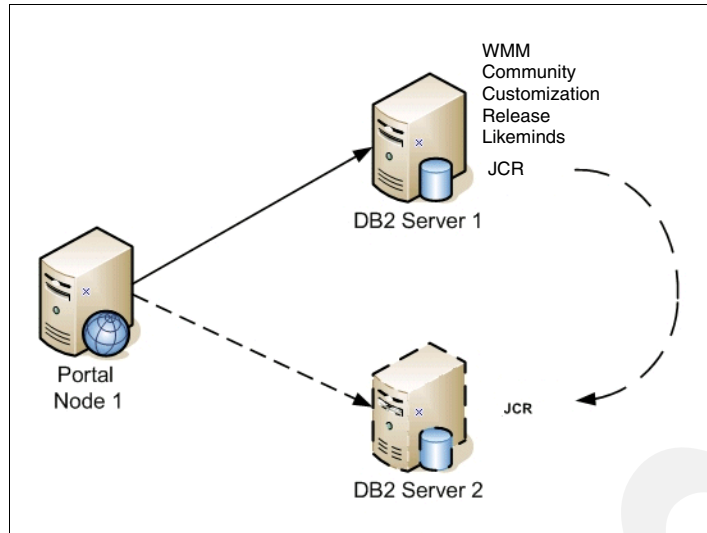


Figure 5-76 Moving the JCR Domain to a new DB2 server

4. We isolated the JCR data by having it on a different database server. Now we can install a second portal server (in our case portal node 2). Install portal node 2 with the same install approach as used for portal node 1, for example, with Process Server or without Process Server. Portal node 2 is the primary node for Cluster B, so it needs to be installed the same as portal node 1 was installed.

As with step 1, portal node 2 will initially have all its database domains in Cloudscape. In a production environment we do not want any data left in Cloudscape, so the next step for building our high-availability environment is to connect portal node 2 to our existing database domains on DB2 Server 1, which are being used by portal node 1.

We connect portal node 2 to all the existing database domains on DB2 Server 1 (customization, community, wmm, and Likeminds) with the exception of the release and JCR domains (see step 5). At this point the release and JCR domains for portal node 2 are still in Cloudscape.

See Section 5.3, “Examples of moving and sharing database domains” on page 241 for the steps on how to connect this additional Portal server node to our database servers.

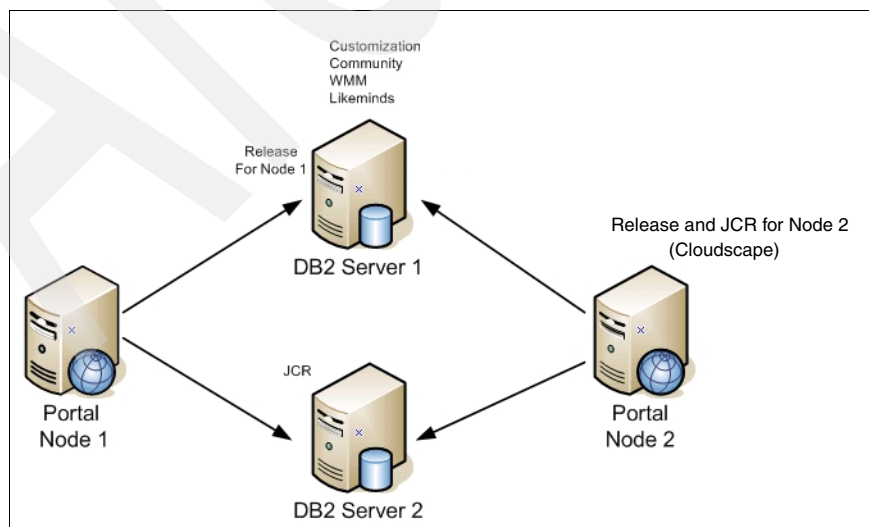


Figure 5-77 Connecting portal node 2 to the database servers

After completing step 4, we now have portal nodes 1 and 2 pointing to the same database domains; however, portal node 2 still has its Release and JCR Domains in Cloudscape.

As previously mentioned, in a production portal environment, we do not want to leave any data in Cloudscape; therefore, the next step in building our high-availability portal environment is to transfer portal node 2's Release and JCR Domains to database servers. We will transfer the Release domain for portal node 2 to DB2 Server 1. We will then transfer the JCR domain for portal node 2 to DB2 Server 2.

Section 5.4, “Example 1, moving database domains” on page 243 gives details on how to move the JCR Domain to a new database server. You can follow these same steps, but substitute the Release Domain values for the JCR Domain values as appropriate.

Figure 5-78 shows what our environment now looks like.

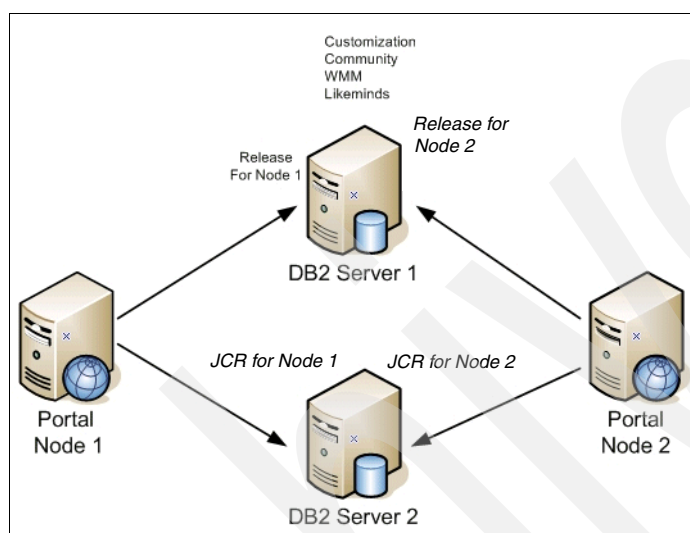


Figure 5-78 Two nodes connected to the same databases with their own release domains

5. The final step is to build two new clusters (Cluster A and Cluster B in our case) with portal node 1 serving as the primary node in Cluster A and portal node 2 serving as the primary node in Cluster B. To accomplish the cluster creation, see Chapter 3, “Different deployment scenarios: building clustered environments” on page 51.
6. At this point you have installed WSAS and Portal on both nodes using the install method determined by whether you want a Process Server cluster or a cluster without Process Server. You configured both primary nodes to external databases. Now you will move forward by following the cluster creation steps after the database configuration occurs on the primary node.
 - a. Cluster with Process Server, see section 3.4.6, “Create the cluster definition” on page 107.
 - b. Cluster without Process Server, see section 3.5.4, “Create cluster definition” on page 153.

All nodes in Cluster A will be configured and sharing the same database domains with Cluster B. Thus Cluster A and Cluster B will be sharing data between the clusters.

As you continue to follow the instructions in Chapter 3, “Different deployment scenarios: building clustered environments” on page 51, you will eventually configure security. For the clusters to share the same data, the security must be configured identically on both clusters by running the security task with the same values in the `wpconfig.properties` files.

Figure 5-79 shows how our final high-availability portal environment will look after the clusters are completed.

The key point to note here is that both clusters are serving up users with identical content and that we can now take down a cluster for maintenance or upgrade and direct users to the other cluster without any disruption.

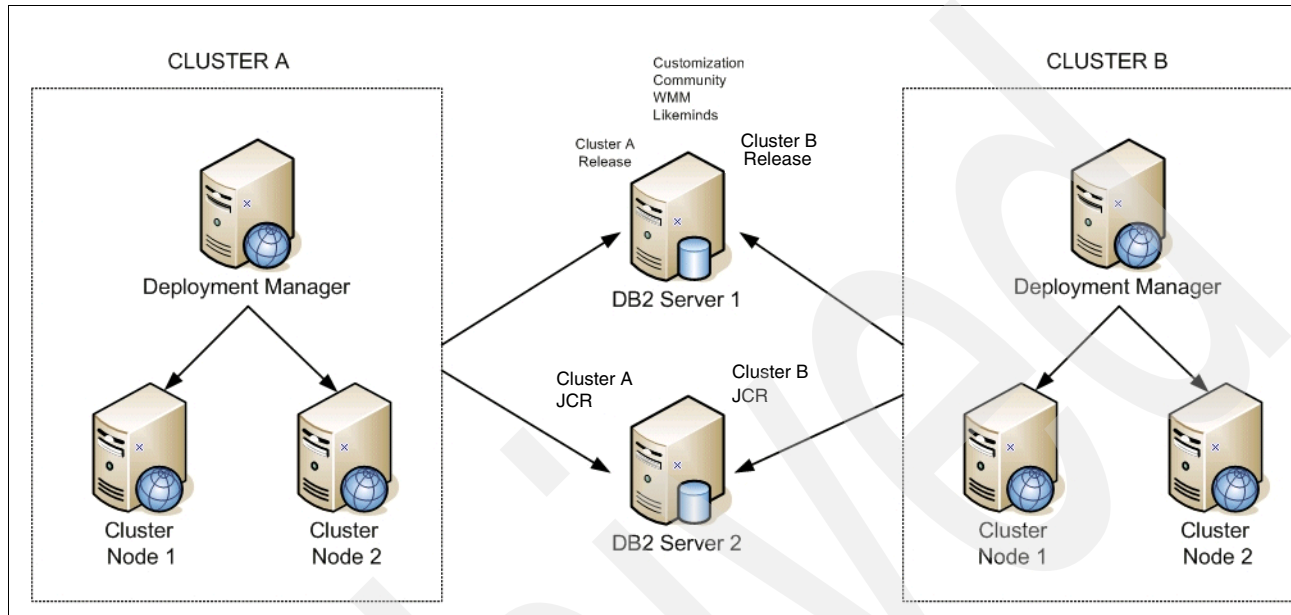


Figure 5-79 A high-availability multi-cluster Portal environment

Archived

Archived

Strategies for tuning and testing

WebSphere Portal Server is an enterprise-wide, integrated solution that touches on many components and teams. Any low-performing part of the integrated solution can cause overall portal performance degradation.

The goal of performance testing or benchmarking is the following:

- ▶ Evaluate the total system throughput and responsiveness against a predefined set of performance goals
- ▶ Identify the low-performing part of the system
- ▶ Execute tuning steps to bring overall satisfactory performance results

This chapter covers the following topics:

- ▶ Initial functional testing of a new portal environment
- ▶ Preparing your Performance testing environment
- ▶ Planning for Performance testing and tuning
- ▶ General strategies for executing performance tests

6.1 Functional test of a new installation

Before you deploy any custom code or configurations to the environment, you should perform a thorough functional check of the new installation.

1. Check that all necessary WAS and WPS fixes are installed (in case there are any WAS or Portal fixes that are necessary after install, or if WAS and WPS were installed separately).
2. Do all starting performance tuning according to the Performance Tuning Guide, especially basic things such as heap size.
3. Start the Deployment Manager (DMGR) and nodeagent if they were down, and check that the logs are free from exceptions.
4. Start portal server from the Deployment Manager console, and verify that there are no exceptions in the SystemOut.log.
5. Log on to Portal with your administrator user id, go to the administrative pages, add a new page, and then verify that the new page is visible.
6. Create new user, and verify that the new user can log in but cannot view the administrative pages.
7. Log on to portal with your administrator user id, assign the new user to the portal administrators group (wpsadmins), and verify that the new user can log onto Portal and now sees the administrative pages.
8. [HA] Bring down one node of Portal in the cluster, assuming there is more than one node in the cluster, and verify that the portal is still accessible.
9. [HA] Bring down one node of the database server, assuming that there is high availability on the database servers, and verify that the portal content is still accessible.
- 10.[usability] Bring down all portal nodes or all database nodes, and verify that the Web server returns something informational, such as that the site is down instead of simply displaying an unhelpful 404 error.

6.2 Planning for Performance testing and tuning

The most important thing about Performance tuning is to do it. Portal out of the box is tuned so that it runs in any environment—from a 32-way processor “Big Iron” UNIX® machine with 48 GB of RAM to an old mobile computer with 1GB of RAM and a single processor not worth mentioning. The initial tuning parameters are set so that, on any hardware, it will install, it can be configured, and you can log onto it. It is not tuned for large numbers of users, and it is not tuned to take advantage of the powerful hardware that you purchased for it to run on. So it is most important that you set aside several weeks in your deployment schedule for tuning.

It is best if you have a production-like environment set aside specifically for performance tuning. That way you can re-run performance tests for later releases.

You should have run performance tests of the default portal install as soon as it was configured, but without any portlet code or customizations. Before you do any tuning at all, run another performance test with all of your portlet code, page layouts, security, and customizations in place. This gives you an idea of the impact of your customizations to the portal environment and its impact to performance. Monitor every aspect of the system—the database servers, LDAP servers, Portal servers, Web servers, and policy servers for single sign-on. If your portlets access any other back-end servers, monitor those too. Any one of these could be a bottleneck to overall performance if not tuned properly.

The next step is to go through the WebSphere Portal Performance Tuning Guide published by IBM and apply the basic tuning parameters suggested. Not all of the suggested values are correct for your environment, but you can use them as a starting point and a reference guide of the most important parameters to look at. Run performance tests between each change to gauge the impact of each parameter as you learn how each one affects your environment. Keep an eye on all of your integration points for any spikes in resource utilization and resolve any bottlenecks as they appear.

The following sections outline key best practices in planning and implementing a successful testing strategy for your Portal implementation.

6.2.1 Planning your test

Successfully planning performance tuning and testing of your Portal solution requires thorough knowledge of all aspects of the environment: user base attributes, login

- Performance testing should have equal importance to development and functional testing when planning and scheduling release cycles.
- Performance testing should start as early as possible.
- Performance testing should be given enough time and resources.
- Designate a single person to be the focal point for performance testing.
- When there is a performance problem, open a defect and track it as though there were a functional problem.
- Designate a performance focal point person who is responsible for following up with Portal system maintenance after the Production portal starts operations.
- Do not use LPAR on performance test environment unless it is also used in the Production environment.
- Apply tuning parameters before running the first portal performance test.

- Use scripts to automate basic performance tuning changes to ensure consistency across environments.
- Check testing script to make sure it covers login, navigational, administrative, public and private pages, reload performance, with realistic think times and login patterns.
- Use an incremental testing approach to execute the performance tests.
- Monitor back end system utilization, and raise concerns if there is an alarming usage of back end resources—even if overall performance results are acceptable.

6.2.2 Establish consistent hardware and software set up

The goal of performance testing is to use load generation techniques and measurements to predict the performance behavior of the production environment under various loads. Therefore, the performance testing environment (PERF) should be an exact replica of the production environment. The hardware and software profiles must be consistent in order to have an accurate prediction of the behavior of your production environment. In particular, the following specifications should be as close as possible:

- ▶ Hardware type (RS/6000, Sun, iSeries, Intel)
- ▶ CPU speed and architecture (32big versus 64bit, LPAR versus non-LPAR)
- ▶ Memory size and speed
- ▶ Software levels (OS, middleware, WAS and WP version and patch levels)
- ▶ Cluster status of Portal (clustered versus standalone, horizontal versus vertical clones)

If you are planning to support a large number of users with a large horizontal cluster, it is acceptable to scale back the number of nodes in your performance testing environment. For example, if your infrastructure plan is for a 6-node portal cluster, you can build a 2-node portal cluster for the performance testing environment. A performance test run against the performance testing environment with 200 users would predict performance behavior of an actual production load of 600 users.

6.2.3 Apply base tuning parameters

Note: Do not forget to tune all parts of your environment—database, LDAP server, and HTTP server and firewalls—all of which can have significant impacts on your overall performance.

Portal server tuning parameters

For complete guidelines on tuning your WP 6.0 environment, refer to the *WP Performance Tuning Guide*, which is located at the following Web site:

<http://www-1.ibm.com/support/docview.wss?rs=688&uid=swg27008511>

Table 6-1 has some important values to consider.

Table 6-1 Portal server tuning parameters

Property	Impact
JVM Heap size	Too big will cause long Garbage Collection (GC) time. Too small will cause frequent GC or paging, or in some extreme cases, out of memory exception.

Property	Impact
Nursery size: MaxNewSize NewSize	Increase the nursery size to approximately 1/4 of the heap size, but not more than 1/2 the full heap. The nursery size is set using XX:MaxNewSize=128m, XX:NewSize=128m Too large will cause longer, less frequent GC time. Too small will increase time spent in memory allocation.
Session Timeout	Too long will cause too much memory usage, because too many concurrent sessions are both saved in memory. Too short will cause negative user experience.
Servlet thread pool size	Too large will overload the CPU. Too small will limit throughput.
Class GC	Whether to encourage class reuse or not.
Enable Pass-by-reference	Whether EJBs can be passed by reference.
WPS configuration datasource WPS release datasource WPS customization datasource WMMdb datasource JCRdb datasource FDBKdb datasource	preparedStatementCache size - too big will use a lot of memory, and too small will cause extra overhead for each statement execution. ConnectionPool size - too big will increase load on the database server, and too small will cause poor throughput. Also notice that each data source can be tuned independently. Allocate resources according to your usage of portal features: <ul style="list-style-type: none"> ▶ WPSdb contains portal configuration and security ▶ WMMdb contains user data, including lookaside or db-only security information ▶ JCRdb contains WCM and PDM content and stores personalization rules ▶ FDBKdb data source contains Personalization and LikeMinds usage data

It is also important to consider your cache strategy and how you can tune Portal's Cache Manager Service to reflect an appropriate amount of caching for the usage of resources in your environment. The available caches and how to configure them is well documented in the *WebSphere Portal 6 Tuning Guide*, which is available for download at the following Web address:

<http://www-1.ibm.com/support/docview.wss?rs=688&uid=swg27008511>

Web server tuning

Your Web server sits between the Portal server and the client (browser) requests. If not tuned for the expected user load, it can become a major bottleneck and impede performance. Table 6-2 highlights some of the most important parameters to consider.

Table 6-2 Web server tuning parameters

Property	Impact
ThreadsPerChild ThreadLimit	More threads allowed will mean more connections but also more CPU power and memory used. Set this to a reasonable value to find the balance between CPU utilization and best throughput. ThreadLimit is the max value that ThreadsPerChild can reach.

Property	Impact
MaxClients	The maximum number of concurrent requests that can be handled. Any concurrent connection request over this limit will be queued (see below for restriction).
ServerLimit	ServerLimit and ThreadLimit together set the maximum configured value for MaxClients.

Database tuning

WebSphere Portal server stores all runtime configurations in portal databases. Portal can generate a very large number of DB queries: if there is a bottleneck accessing any of the portal's underlying databases, it can have a significant performance impact.

LDAP server tuning

Connections are made into the LDAP server by the security components of portal frequently for authentication and authorization purposes, especially during the login sequences. The design and structure of your LDAP server can have a significant impact on performance, especially for group memberships.

In particular, enabling support for nested groups can impact performance, especially if your LDAP server has a lot of deeply nested groups. Using dynamic groups or a very large number of groups can also have a performance impact unless your LDAP supports (and you configure portal to use) a user attribute to determine group membership rather than spidering group memberships to find the current user (for example, setting the groupMembershipAttributeMap to ibm-allgroups: all in the case of Tivoli Directory Server LDAP).

6.2.4 Run baseline performance tests before deploying your applications

As soon as you complete the base portal installation and configuration of the performance testing environment and applied the initial tuning parameters as indicated for your environment and applications in the WP Performance tuning guide, but before you deploy any of your applications, run a baseline performance test of your environment.

6.3 Scripting performance tests

The goal of performance testing is to measure portal performance metrics under certain portal usage scenarios. The Web-based application performance test is usually executed using scripts on a load generator tool (such as Rational® Robot or LoadRunner) to simulate workload with virtual users (vusers).

6.3.1 Performance metrics

There are two major performance metrics that can be used to evaluate portal application performance: *response time* and *throughput*. Response time is the time required to complete a unit of work (such as a page load). Throughput is the total units of work per period of time (measured in seconds or minutes). Both metrics must be considered to set performance test scenarios. Portal applications share some of the common Web application characteristics, and it also has its own unique characteristics.

In a portal server context, throughput is commonly treated as how many page views per second can be served. The response time is the time required for portal server to completely render the portal page per user workload (an http request). Because the portal server is an aggregator of multiple applications on the same page, and it also integrates multiple enterprise services such as security (authentication and authorization), you need to consider the following scenarios:

- ▶ **Login performance:** The time it takes from when a user enters their user ID and password until the first portal page is fully rendered. Usually, this is the total time taken for user authentication and the first portal page rendering. The second part usually requires extensive computing. Before the portal server shows the first page, depending on the portal theme and portal access control model, it needs to decide what pages or links this current user is authorized to access and dynamically show or hide the links. This process includes multiple back-end calls such as LDAP and database queries, and it is likely to see worse performance when there is back-end service contention.
- ▶ **Navigational performance:** The time it takes for a regular portal user to navigate from one portal page to another page when the user clicks a page link. This activity usually involves a lot of database reads to load configuration data from the database. This performance depends heavily on database read throughput.
- ▶ **Administrative performance:** The performance measurement for a portal administrator to log in and navigate. By nature, a portal administrator is able to access more portal resources, and most of the operations of a portal administrator are to change portal configurations. This involves a lot of portal database writes and as a result of more access control computing and heavy database writes (which are more expensive than reads), it is possible that administrator performance is worse than normal user-viewing performance. You need to consider whether it is important to measure administrative performance.
- ▶ **Public pages versus private pages:** Portal server allows you to set pages to be publicly viewable (not requiring login), and these pages can take advantage of different levels of caching (at the Web server, proxy server, client, and so on). The performance measurement results can be very different for public or private pages.
- ▶ **Reload performance:** After a Portal Administrator makes configuration changes, it may take some time for regular users to see the changes. You may want to consider measuring how responsive your portal is in reloading portal resources.

6.3.2 Building scripts

Knowing what to measure is not enough. You also need to consider how a normal portal user is going to interact with the portal server. If the real world behavior is different from the test scenario, hardly any conclusions can be drawn from the results. Following are some best practices to simulate user behavior, which are further discussed in the next sections:

- ▶ Assign unique user login per virtual user
- ▶ First time log in or repeating user log in
- ▶ Do not use hard-coded URLs in your scripts
- ▶ Use an appropriate think time
- ▶ Script user logouts realistically
- ▶ Warm up the Portal

Assign unique user login per virtual user

We recommend that your load generator scripts assign a unique portal login user id to each virtual user (vuser) in the load generator. When two or more user sessions log in using the same user id and the two sessions both have configuration changes that need to be saved to

the database, it will cause a lock condition in the database. This will severely degrade portal server performance. To ensure such uniqueness, you can do one of the following:

1. Have a large pool of portal user ids so that each vuser can randomly pick one user id. We recommend having a user id pool that is 25 times the total number of vusers used by the load generator tool. For example, if you have 100 vusers, then prepare a pool of 2500 user ids in your LDAP server. Also, make sure that your load generator script always picks the id from the pool randomly (rather than sequentially from the top of the list).
2. Assign a dedicated unique portal user id to each vuser. For example, vuser1 will always use user id *wpsuser1* to log in. This is less realistic but will guarantee uniqueness.

First time log in or repeating user login

This is related to the point made in the previous section. The login performance measurement is different depending on whether this is the first time the user logged in or whether this is a repeating login of the same user. Your performance script should consider this. You might want to test two different scenarios depending on your business requirements.

Do not use hard-coded URLs in your scripts

Since version 5.1, Portal URLs can no longer be parsed or composed on the client side. The URL is always dynamically generated to include the encoded format of portal page state. Your script should always get the URL from the retrieved portal HTML page to determine the next link location. Test scripts should not contain any hard-coded URLs except the entry point (the first URL).

Use an appropriate think time

Think time is the time period in between simulated user clicks (between each generated load request). The suggested think time is 10-15 seconds. You can reduce the think time to generate higher load (requests per second), but there is a boundary to this approach. Notice the following:

$$\text{Requests per second} = \# \text{ of concurrent users} / (\text{avg response time} + \text{avg think time})$$

This means that there is an upper limit of the following:

$$\text{requests per second} = \# \text{ concurrent users} / \text{avg response time}$$

To achieve this, the average think time would have to be 0, which is not possible. Assume you have 100 concurrent users (vusers) and the average portal response time is 1. Then you cannot have more than 100 requests per second. This is another common pitfall in that people confuse requests per second with the number of concurrent users. The number of concurrent users is normally the size of your vuser pool. These users are concurrent over a long period of time (hours), while requests per second is a calculated value that represents the number of requests that can be handled by portal server within a short period of time (a second). When you make performance goals, remember that people are more likely to be concerned about the number of concurrent users instead of requests per second.

Script user logouts realistically

Considering your current user population, ask and answer the question: Does every user log off when they are done with the portal application, or do they never log off explicitly but just close the browser window? In the latter case, the server session for this client stays in memory until it reaches the session timeout value. If most users do not log off, you should consider reducing the session timeout value in the JVM settings for portal.

Warm up the portal

When the portal server (or any other application server) is restarted, most of the caches are rebuilt from scratch, and it takes some load to populate the caches with a reasonable amount of data before the cache is sufficiently populated to improve performance. It is recommended that before each performance test, you run some load on the server to populate the cache entries and gain realistic test results.

6.3.3 Testing scenarios

At a minimum, performance testing should address the following two scenarios: achieving maximum throughput and an endurance run.

Table 6-3 Testing scenarios

	Maximum throughput	Endurance run
Definition	Use the load generator to ramp up the load over predefined intervals until the server reaches the heaviest portal server traffic based on your business requirements during a relatively short period of time.	Use the load generator to simulate constant load on the portal server over a long period of time.
Load profile	Variable load: Rapidly increasing load. Stop when performance exceeds acceptable goals (for example, login time > 5 seconds).	Constant load: Expect the performance to stay constant. Stop when performance starts to degrade.
Length	Usually between 1 and 5 hours.	Usually several days to a week.
Goal	Test the maximum system capacity. Any performance enhancement or degradation can easily be seen.	Test the overall system stability to reveal problems such as memory leaks.

6.3.4 Setting goals

After you understand the unique performance metrics that relate to portal, and you understand the behavior patterns of your user population, you can set performance goals based on your business requirements and historic performance data.

The best practice is to keep your goals realistic and to accept average or percentile values. For example, the following is a typical realistic performance goal:

Achieve 50 requests per second with response times of the following:

<2 seconds in average for navigation

<4 seconds for login requests

And have 50 requests per second with the following:

average WebSphere Portal CPU utilization <50%

average DB2 server CPU utilization < 20%

With the following portal statistics:

1000 total page count

6.4 Executing your test

This section discusses how to execute your performance tests.

6.4.1 Establish baseline performance

This step is necessary to prepare tests on a new portal installation or when the portal system has undergone major changes (such as moving from WP 5.1 to WP 6.0 or changing the back-end database or LDAP server). The philosophy behind this test is to detect, fix, and base portal configuration problems as early as possible. If the base portal system has some issues, no matter how well-performing your custom portlets are, the overall load testing will not yield satisfactory results. The earlier you can find base portal system tuning problems and fix them, the greater savings you can gain overall in your performance testing cycle.

Start this test by ensuring that the bare-bones portal installation has good performance by itself. The portal server should be installed and configured. If you are using an SSO application such as Tivoli Access Manager or SiteMinder, security should be enabled so that the SSO login challenge is in use. Before installing any custom code, including themes and skins, create an empty portal page with no portlets as a landing page so that it becomes the default page that virtual users will see after logging on. Run a basic login performance load test to make sure that this system is able to render acceptable login response times under the desired load.

If portal performs poorly at this point, check to make sure that the basic tuning of all systems was completed, including portal, Web server, database, LDAP, and the SSO solution. If investigation shows very high CPU utilization under heavy load, you might need to evaluate the hardware specification. Rerun the test as necessary as you change tuning parameters until the base portal login time is satisfactory.

6.4.2 Deploy applications one at a time and retest

Rather than deploying your entire portal configuration and application suite into performance all at once, deploy application code in stages and retest. This approach is particularly valuable for a team who is not familiar with portal performance testing. Deploy custom content (themes, skins, portlets, and pages) in an iterative fashion—start small and then add one or two features or change one or two parameters at a time, run the test, and only add more if it passes the test.

TIP: Do not add too many new features or change too many parameters between tests. The more changes you have between tests, the more difficult it is to isolate the problem and resolve it.

One frequently asked question is “If I follow the portal deployment approach outlined in Chapter 7, “Configuration management: moving between environments” on page 337, the entire portal solution will be moved from the UAT environment using ReleaseBuilder. How do I achieve this incremental test strategy?” We recommend that when a new portal application is introduced to the performance testing environment (PERF), do not use ReleaseBuilder or a full XMLAccess export for the initial deployment. Instead, use XMLAccess to selectively export and import individual component resources into the PERF environment, run performance tests, and make tuning or code fixes if necessary. After each individual

deployment, if performance is satisfactory, then deploy more resources; otherwise, refer to section 6.4.4, “Monitoring: what to monitor?” on page 331 to troubleshoot. Because the scope of change between each incremental deployment is fairly small, it is straightforward to identify any performance issues as they come up.

After the entire system performs well, in the performance testing environment, perform the following steps to ensure that nothing was missed:

1. Apply to the source (UAT) portal environment all of the tuning and code fixes that were introduced for each performance tuning step.
2. Clean up the PERF environment portal. Use ReleaseBuilder to move all resources from the UAT portal environment.
3. Perform a final regression performance test on the integrated solution on the performance test server.

If custom code fixes were needed, ensure that the changed code is checked back into your version control system. If there were portal configuration changes, the corresponding XMLAccess or wsadmin scripts should be updated and checked back into the version control system so that changes are not lost in future environments.

6.4.3 Analysis and comparison to baseline

There are two levels of baseline performance statistics that can be used:

- ▶ Legacy application performance measurements. If the portal application was converted from an existing J2EE application, the natural expectation is to compare portal application performance with the traditional application performance. While this is an important metric due to the fact that your users will be switching from the traditional application to the portal application, the reality is that due to the added computing cost of the portal presentation layer, it is possible that the performance measurements of the portal application for both throughput and response time, may be worse than that of the legacy J2EE application. We suggest using this baseline as a loose reference rather than a strict rule.
- ▶ Initial performance measurements of the base portal. This baseline is more meaningful to performance testing and tuning. After each incremental deployment and adjustment step, take performance measurements and compare them to the baseline results. This will provide more direct feedback on the effects on performance of your custom code. We suggest that you use this baseline as your main one for comparison and analysis.

6.4.4 Monitoring: what to monitor?

As we mentioned earlier, portal performance tests measure the overall user experience as an integrated results of five major software components. Any contention of any of the five components prevents the entire portal solution from achieving optimum performance. It is important to monitor the following touch points and make sure that there is no contention of resources before approaching 100% of CPU utilization on the portal server. Monitoring provides key evidence used in troubleshooting portal performance problems.

Important! If your portal environment shares backend resources (such as LDAP or database servers) with other applications, run simultaneous load tests of all systems to determine the impact of full load on all systems together. If you need to scale those resources to handle the additional load, you want to know that before you go into production.

CPU monitoring involves monitoring the utilization in percentage during the load test. One factor often overlooked is the CPU utilization of the load generator. In general, if your CPU utilization reaches 60% or greater on your load generator machine, you may have a bottleneck on the load generator itself, rather than somewhere in your portal solution. If that happens, you may want to add another load generator. CPU utilization should be monitored on the Portal server, the Web server, the database server, the LDAP server, and on the load generator.

File I/O is measured in disk read/write in bytes, and can also be a factor for performance. If you see a lot of expensive disk I/O on the portal server, check to make sure that logging and tracing is turned off. SiteAnalyzer also logs an entry for each portlet access, and the Web server logs every http request in access.log. On the database and LDAP servers, disk I/O speed is mission critical. It is suggested that logs are written to separate hard drives. File I/O should be monitored on the Portal server, the Web server, the database server, the LDAP server, and on the load generator.

The network should also be monitored for problems. If packets are dropped between servers in your portal environment, it can have a significant negative impact on performance. Use the ping command to determine that there are no packets being dropped. Check the network between the following servers:

- ▶ Portal server and the database server
- ▶ Portal server and Web server
- ▶ Portal server and LDAP server
- ▶ Http server and load generator
- ▶ Http server and portal server
- ▶ Database server and portal server
- ▶ LDAP server and portal server
- ▶ Load generator and portal server

On the portal server, it is also important to monitor JVM memory utilization, thread status, and database connection pools. These can be monitored using JVM logs, java thread dumps, and Tivoli Performance Viewer.

When you turn on verbose garbage collection (GC) on the portal JVM, it generates GC event logging in the WebSphere/PortalServer/log/native_stderr.log.

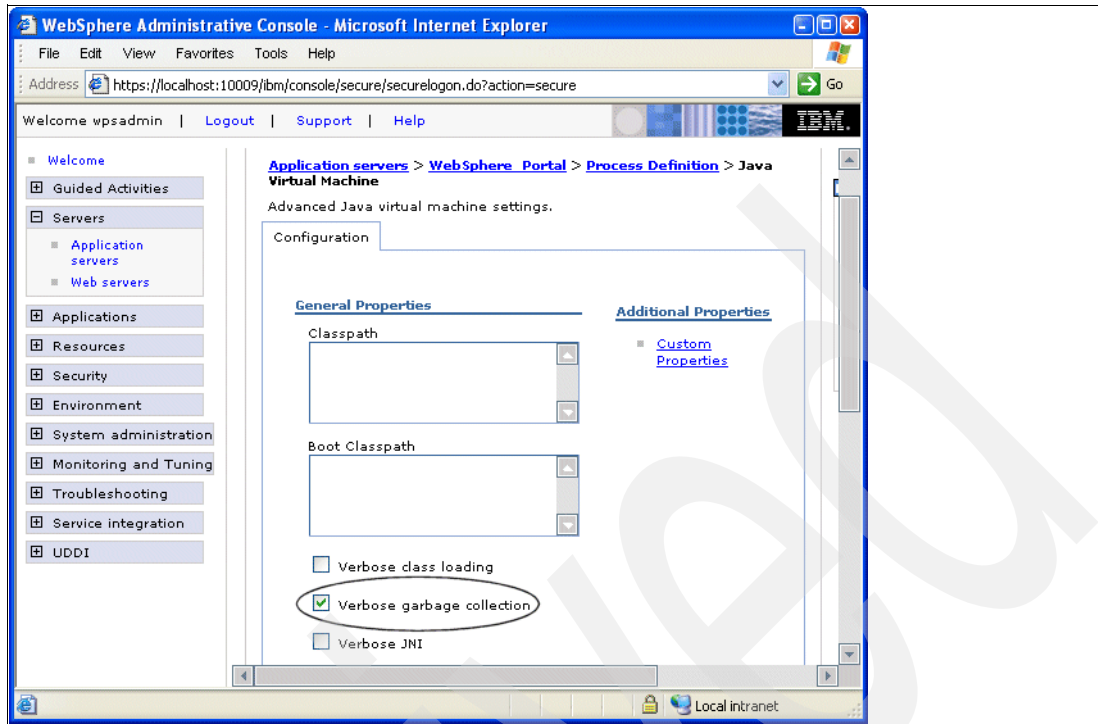


Figure 6-1 Enabling verbose Garbage Collection in WP 6.0.

6.4.5 Monitoring database performance

WebSphere Portal Server stores configuration, access control, such as user identities, credentials, and permissions for accessing portal resources and user data in a database. The performance of the underlying database(s) is critical to overall portal performance; therefore, monitoring and tuning the performance of the database is critical to WebSphere Portal performance testing.

There are two sides to monitoring database performance:

- ▶ Portal server side database performance
- ▶ Database side performance

Portal server side performance monitoring is done using Tivoli Performance Viewer, which is included with WAS or using a tool such as Wily Introscope or Wily Portal Manager. Documentation for Tivoli Performance Viewer can be found at the following Web address:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.webSphere.base.doc/info/aes/ae/tprf_tpvmonitor.html

Documentation for Wily Portal Manager can be found at the following Web address:

http://www.wilytech.com/solutions/products/PortalManager_IBM.html

When monitoring performance from the portal server side, the following three metrics are most important:

- ▶ Time spent executing database calls: This is the time required for issuing database queries. This value should normally be <35ms.

- ▶ Connection pool average wait time: This should be very close to 0. Otherwise, it means that the database pool is likely too small, causing many connections to wait.
- ▶ Connection pool Prepared statement cache discards: Prepared JDBC query statements are stored in JVM memory, reducing CPU load on the database server. Calculate the ratio for each data source (wpsdb, jrddb, wmmdb, and so forth):
 - Discard ratio = # of discards / total # of queries

If you see a data source with a very high ratio, you should increase the size of the prepared statement cache associated with that data source. Since wpsdb is the most critical to overall portal performance, make sure its discard ratio is low (<10%). If your portal implementation makes heavy usage of WCM or Personalization, pay special attention to the JCR data source. If your users are stored in the database, instead of LDAP, or if you have a lot of user data stored in the LookAside database, pay extra attention to the WMM data source.

When monitoring performance on the database server side, your DBA should make use of Snapshot™ monitors to quantify the number of queries coming out of Portal. Use snapshots to capture information about the database and any connected applications at a specific time. Taken at regular intervals, they are also useful for observing trends and foreseeing potential problems.

The following metrics should be looked at in particular:

- Total DBM Agents waiting for a token
- Total DBM Agents stolen
- Max Applications Connected to DB
- Avg Applications Connected to DB
- Application Statements Committed
- Application Rollback Requests
- Internal Statements Committed
- Internal Rollbacks
- Total Transactions
- Transactions Per Second
- Average Rows Read Per Transaction
- Rows Inserted
- Rows Updated
- Rows Deleted
- Rows Selected
- Total Rows Affected
- Average Rows Read Per Affected
- Total Lock Escalations
- Total Deadlocks Detected
- Total Lock Timeouts
- Total Lock Waits
- Percentage Lock Waits
- Average Lock Wait Time (sec.ms)
- Max Lock List Memory in Use (KB)

Monitor these metrics on all portal databases—wpsdb, wmmdb (if created separately), jrddb, lmdb, and fdbkdb.

6.4.6 Performance maintenance: keeping it fast

Once you deploy your portal solution into production for real life use, as more and more customizations, such as new pages, new portlets, new content are added or changed by either portal administrators or normal portal users if allowed, certain performance

maintenance tasks must be performed to keep portal performing the way it did when it left your performance lab.

Over time, as portal configurations are added to tables, and other information is deleted, there is an increase in fragments on the hard disks where the database stores data. To keep your portal implementation running in top shape, you should perform tasks such as rebuilding indexes, statistics, and reorganizing on a regular basis.

When Portal retrieves information from the database, it stores that information in cache so that later queries for the same information do not have to go back to the database again. Having the correct cache size setting is important. The larger the cache size, the more likely that entries are reused. But once memory is full, the JVM will start very expensive garbage collection processing that degrades overall system performance. Monitor the cache size and make sure that it is close to the reasonable upper limit of actual usage, for peak performance.

6.4.7 Final test should reflect user behavior

After all applications are deployed and tuned individually, run a performance test of the entire system using a script that reflects typical user behavior. If the portal is replacing an existing system with similar page layout and functionality, you should have statistics on user behavior that you can use to build your script. For example, if 70% of your users do not log out of the system, then your script should not either; otherwise, you will not see the effect on load of all of those inactive sessions waiting to time out.

Archived

Configuration management: moving between environments

This chapter describes the processes necessary to move portal artifacts and configurations between your environments. It describes the development release cycle and the procedures necessary to deploy those releases through your environments into Production.

This chapter will cover the following:

- ▶ Tools for configuration management
 - Transferring Portal artifacts using XMLAccess
 - Moving a differential release using ReleaseBuilder
- ▶ A step-by-step guide to moving a release between environments
- ▶ Maintenance procedures related to Configuration Management
 - Handling orphan data
 - Delayed cleanup of deleted portal resources

7.1 The Portal staging process

After you complete all of your portal installations and configurations, your developers have written wonderful custom portlets, your portal designer has laid out the pages, you applied appropriate security to everything, and you are ready to move the entire configuration to the next environment in your release cycle. This chapter discusses the tools available to you to achieve this.

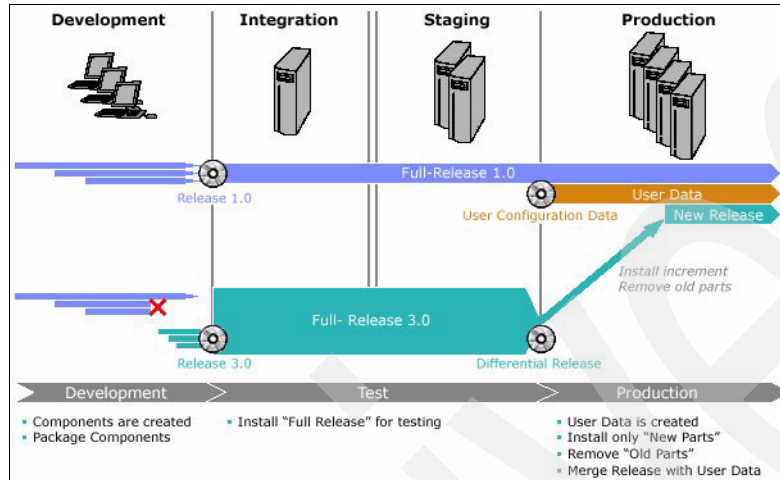


Figure 7-1 The release cycle

7.1.1 Terminology

Before we look at configuration management in more depth, it is important to define what we mean by some of the terminology we use. Every client environment seems to have their own terms, but to have a meaningful discussion, we all need to use the same ones.

The terminology we use in this chapter is as follows:

- **Migration** is the word used to describe the process of moving and converting Portal artifacts and configurations from a previous version of Portal to the current one. This topic is not discussed in this book, as there is another IBM book devoted to it: *IBM WebSphere Portal V6: Best Practices for Migrating from V5.1*, REDP-42270, at the following Web address:

<http://www.redbooks.ibm.com/redpieces/abstracts/redp4227.html?Open>

This word *migration* is commonly but incorrectly used interchangeably with terms such as Configuration Management.

- **Configuration Management** is the process of moving a portal release to the next environment in the chain. Many companies have their own terms for this, such as Elevation, Promotion, Release, Uplift, and Deployment.
- A **Release** is a complete portal configuration, including content and code, perfected and tested at a point in time and ready to be moved to the next server in the cycle.
- **Staging** is the process of building the final, tested release in an environment other than production with the intention of deploying that release to production. This environment is commonly called the staging environment. It gives you and your testing team the opportunity to see what your portal will look like in production before you release it to your customers or employees.

7.2 Deployment and build process

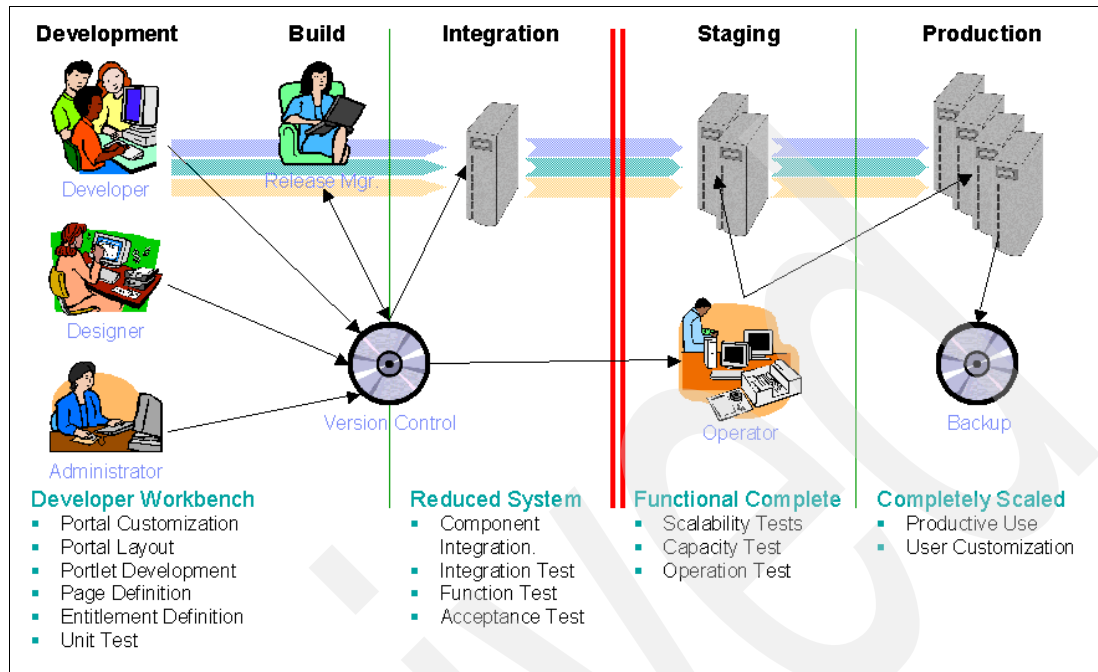


Figure 7-2 The build and staging process

Following is an example of how you could implement a Portal build process across environments:

1. A developer implements portlets, servlets, Enterprise JavaBeans™, and other J2EE artifacts using either Rational Application Developer or source code that is delivered into a version control system.
2. A designer creates themes, skins, HTML pages, portlet JSPs, and other design elements using any editor, and delivers the results into a version control system.
3. An administrator creates the content tree (labels, URLs, and pages) using the Portal admin user interface of a development Portal.
4. The resulting content trees and portlet instances are exported using XMLAccess or a script and then delivered into a version control system.
5. The release manager assembles a consistent release in the version control system and creates the delivery. The release manager executes scripts (for example, ANT) to extract Java sources, design elements, and configurations from the version control system and then runs a build (compile and package).
6. The operator takes delivery and deploys it onto the staging and production systems. The operator executes ready made config tasks (for example, ANT, XMLAccess configurations, ReleaseBuilder, and wsadmin scripts) to deploy the delivery.

There are many tools provided to help you perform these tasks. This guide will focus on the following four: XMLAccess, ReleaseBuilder, SLChecker, and the cleanup scheduler task.

Note: All of the tasks discussed in this chapter can be automated using scripting tools such as ANT or Maven. This chapter is intended to cover the steps an administrator would perform manually if the deployment was not scripted for automation. Appendix A, “Deployment automation options” on page 407 covers some of the options that can be used to script the tasks outlined in this chapter.

7.2.1 Object IDs

All resources in the Portal (except for the *portal* and the *settings* resources) have an object ID that uniquely identifies them in the Portal. The Portal generates Object IDs when resources are created. Object IDs are globally unique. Two object IDs that are automatically generated by different Portal installations can never be the same.

You can exchange resources between different Portal installations using XML exports and imports without worrying about possible object ID conflicts. Object IDs are represented by the `objectid` attribute in an XML export.

Object IDs are used to express references from one resource to another. For example, the following example references a portlet and puts it on a page using a symbolic object ID:

```
<portlet action="update" ... objectid="_3_609EVJDBI1S5CD0I_97 Welcome Portlet" ...>
...
<portletinstance action="update" ... portletref="_3_609EVJDBI1S5CD0I_97 Welcome Portlet"
...>
```

Note: You cannot simply invent Object IDs for new resources, because they must conform to a correct internal representation. To specify your own unique identifiers for portal resources, use Custom Unique Names.

7.2.2 Custom unique names

A resource that has an object ID can optionally also have a unique name. You can use the unique name to identify the resource unambiguously. Unique names are useful if you need a symbolic way to identify certain resources.

In contrast to object IDs, it is possible to modify unique names of resources, which may be an advantage in certain situations. Meaningful unique names are necessary to implement any scripting automation of the deployment process.

If you execute an XML import that assigns a unique name that is already used on the system, the execution fails. You can delete the unique name for a resource by setting it to the undefined value. To set a unique name for a resource, use the Custom Unique Names portlet under Administration → Portal Settings. Before you export with the XMLAccess tool, ensure that you defined naming conventions for unique names to prevent name clashes.

It is important to assign unique names as early as possible in the release cycle, preferably in development. Administrators may not see the need to assign unique names to all portal resources in the beginning, especially in early development, but a more thoroughly defined unique name practice will give maximum flexibility for future upgrades and migration. For example, for one portlet WAR containing multiple portlet applications, the initial `deployment.xml` does not need a unique name for the individual portlet application. Specifying a unique name for the WAR module is sufficient for deployment. But if later on there is a need to update just one portlet application (for example, changing initial portlet configuration), you

will have to update the entire WAR module since the unique name is defined only at the WAR module level.

7.2.3 The XML configuration interface

XMLAccess is the portal utility tool for administrators to export portal configurations and artifacts from the source server (Integration or Staging) for import into the destination server (Staging or Production). XMLAccess will not generate delete instructions for you, although you can create them by hand if you are very familiar with the structure of the XML files. For complete differential releases between servers, use ReleaseBuilder, which is discussed in section 7.2.4, “ReleaseBuilder” on page 342.

Using the XMLAccess tool for moving

Although there is no automated method to move Portal applications from one environment to another, there are two options:

- ▶ Completely replace the old release with a new release. The drawbacks of this option is that any data that was customized by the user is lost. While this option works, we do not recommend it.
- ▶ Use the XMLAccess tool in combination with ReleaseBuilder to load incremental or differential release as shown in Figure 7-3.

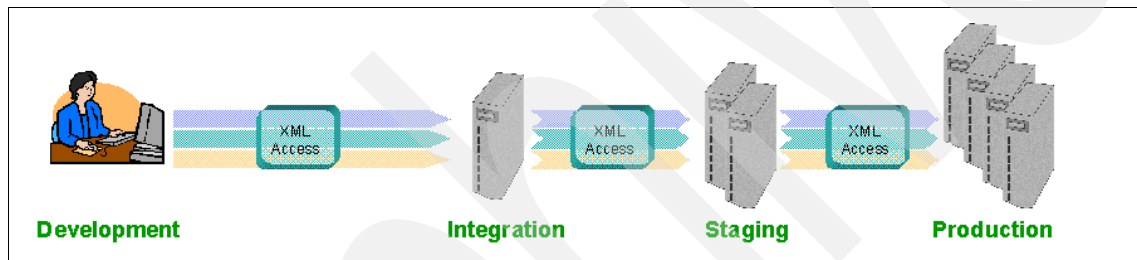


Figure 7-3 The XMLAccess tool

The XML configuration interface

The XML configuration interface is more commonly referred to as the XMLAccess tool, because `xmlaccess` is the command that is executed. The tool provides a batch processing interface for Portal configuration updates and allows you to export an entire Portal configuration or parts of a configuration. For example, you can process and export specific pages to an XML file. You can then recreate the exported configuration from a file on another Portal.

While XMLAccess is a simplified bulk transfer utility, there is no magic button that automatically moves all the components of your Portal to the next environment. The XMLAccess tool interface was developed in WebSphere Portal V2 to help clients move Portal artifacts from one machine to another.

The XMLAccess tool is best for the initial loading of Portal application and for doing incremental releases. However, it is increasingly being used as a command line Portal administration tool. When combined with ReleaseBuilder, it can be your administrator's primary tool for migrating portal configurations between environments.

Why use XMLAccess?

The major benefit of XMLAccess is its ability to update complete configurations of pages and portlets along with their security and configuration settings, without losing user customization.

It allows you to set up your complete portal solution, and move it in its entirety to the next server in your release cycle, minimizing the possibility of human error. You also have the ability to export just part of a portal configuration—just a page, just a portlet, or an entire subtree of your portal navigation with all the pages and portlets below it. If you perform your updates via XMLAccess, any user customization to a page or a portlet is retained because the object IDs are retained.

When exporting and importing via the XMLAccess tool, the object IDs of the portlet application are maintained. When a user customizes a portlet or a page, these customizations are stored in your back-end database. All these customizations are stored relative to the actual portlet application. The key that ties all of these customized versions of a portlet or page back to its parent is the object ID.

7.2.4 ReleaseBuilder

ReleaseBuilder enables management of release configurations independent of user configurations. Release configuration data can be exported into an XML configuration interface configuration file. During staging of follow-on releases it is now possible to stage release configurations between two releases using the XML configuration interface. This allows you to track which configuration entities were removed, added, or changed compared to the previous release, and apply differential updates. Differential updates are created by detecting the differences between one configuration and another. A third configuration can then be generated to represent the difference and used to apply not only addition and update modifications, but also deletions to the production server. This allows the staging and production servers to remain in sync and eliminates the problem of configuration bloat on the production server.

Why use ReleaseBuilder?

Like XMLAccess, ReleaseBuilder also allows you to move complete configurations from one environment to another, but you cannot use it to move partial configurations. Where it really makes a difference is in the later environments of the release cycle—performance testing, staging, and production. You may not want to use ReleaseBuilder to move configurations from the development environment into QA, for example, as your development environment may not always be in a state that you want to move to other environments, especially production. A common practice is to use XMLAccess to move partial configurations from Dev to the early testing environments, and then use ReleaseBuilder to move full configurations up from there through the rest of the environments in the release cycle.

7.3 Transferring Portal artifacts using XMLAccess

The process of moving Portal artifacts from one environment to another is a repeatable process that, after you complete it, you can effectively duplicate the process to transfer onto subsequent environments.

Note: The exception is a clustered environment where the export process remains the same but the import process requires extra steps.

You can transfer the following using the XMLAccess tool:

- ▶ Portal Web application configurations (portlet applications)
- ▶ Portal skin definitions
- ▶ Portal theme definitions
- ▶ Portal portlet configurations

- ▶ Portal site map (pages, labels, and links)
- ▶ Portal URL mappings
- ▶ Portal supported Languages
- ▶ User resources

How XMLAccess works

The XMLAccess command line client is a small separate program that connects to the server using an HTTP connection, which allows you to configure the Portal remotely. The XMLAccess command is located in the *wp root/bin* directory of the Portal server and is **xmlaccess.bat** (on Windows) or **xmlaccess.sh**.

The syntax for the command is as follows:

```
xmlaccess -user wpsadmin -pwd itso -in file.xml -out result.xml -url
myhost:9082/wps/config
```

In the command line, use the following file names:

- ▶ **file.xml**: The name of a file containing the XML request (configuration export or update) that should be processed.
- ▶ **result.xml**: The name of the result file containing the XML output (configuration export). You can later use that file to re-import the exported configuration.
- ▶ **url**: The URL to access the Portal configuration servlet. This URL consists of the Portal host name, the base Uniform Resource Identifier for the Portal as specified during installation (for example /wps), and the servlet extension /config.

You can use the XMLAccess tool to transfer a complete configuration, including the following:

- ▶ Pages
- ▶ Navigation
- ▶ Portlets
- ▶ Access Control List

Note: You need to transfer portlet .WAR files separately. Copy .WAR files to the PortalServer\installableApps\ directory on the target server. Themes and skins also need to be deployed separately, as discussed in section 7.6.3, “Deploying themes and skins” on page 362, although they are defined and assigned to pages using XMLAccess.

With the XMLAccess tool, you can also do the following:

- ▶ Load the WebSphere Portal default portlets configuration during the initial install and transfer parts of Portal configuration.
- ▶ Create and modify existing Portal artifacts in incremental releases.
- ▶ Delete Portal artifacts. However, you must manually add delete commands to the input file.

For a complete list of Portal artifacts that can be moved and for a complete list of commands, visit the following InfoCenter Web site:

<http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/topic/com.ibm.wp.ent.doc/wps/adxmltsk.html>

7.3.1 Transfer process

Transferring your Portal artifacts from one environment to another requires you to export your artifacts from the source environment and then transfer those artifacts to the target environment when you must import them.

The process involves the following steps:

1. Exporting the source configuration using XMLAccess commands.
2. Bundling the supporting files from the source Portal.
3. Transferring the bundled files to the target Portal.
4. Distributing the supporting files to the correct locations on the target Portal.
5. Updating the configuration of the target Portal using the XMLAccess tool.
6. Following up with post-transfer activities.

7.3.2 Exporting a sample page using XMLAccess

There are a number of sample scripts available from the WebSphere Portal InfoCenter. In the following example, we modified the sample file ExportPage.xml to export the Portal Welcome Page.

1. Copy and paste the following sample into a text editor and save it as ExportPage.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="PortalConfig_1.2.xsd" type="export">
  <portal action="locate">
    <content-node action="export" uniqueness="wps.My Portal.Welcome"/>
  </portal>
</request>
```

2. Check that the XML file is formatted correctly and that you have not copied over any unwanted characters by opening the file.
3. Run XMLAccess against this export script using the following command format:

```
[xmlaccess script] -user [PortalAdministrator] -pwd [PortalAdminPassword] -url [portal
config url] -in [xml input file] -out [out put file exported xml]
```

In a Windows environment, the command line should look similar to the following:

```
c:\WebSphere\PortalServer\bin\xmlaccess.bat -user wpsadmin -pwd itso -url
http://devportal.redbook.ibm.com:10038/wps/config -in ExportWelcomePage.xml -out
ExportedWelcomePage.xml
```

In a UNIX environment, the command line should look similar to the following:

```
/WebSphere/PortalServer/bin/xmlaccess.sh -user wpsadmin -pwd itso -url
http://devportal.redbook.ibm.com:10038/wps/config -in ExportWelcomePage.xml -out
ExportedWelcomePage.xml
```

Regardless of whether the XML export works correctly, you will see an output file called ExportedWelcomePage.xml. If you open this file, the last line should display **<status element="all" result="ok" />**. If the export was unsuccessful, a error message appears to tell you what went wrong with the export.

7.3.3 Exporting and importing a new page

This section demonstrates very simply what is involved in moving a custom page from a staging to production environment.

Exporting

Using the same process described in section 7.3.2, “Exporting a sample page using XMLAccess” on page 344, you can export and then import a custom page from one Portal into another. In a staging or production environment, follow these steps:

1. Create a new blank page in your Portal.
2. Assign the blank page a unique name. You assign unique names to a Page via the Portal Administrative interface by selecting Portal Settings → Custom Unique Names → Pages. Find the page that you just created. Edit it, and assign the page a meaningful custom unique name, such as **my.TestPage**.

Note: Do not add any portlets to this page.

3. Create an XMLAccess input file following the instructions section in 7.3.2, “Exporting a sample page using XMLAccess” on page 344. Save the file as **ExportTestPage.xml**, and set the `uniqueName` attribute of the page to export to the unique name you set in step 2.
4. Ensure that the **-out** parameter in the command line points to a new export file, for example **ExportedTestPage.xml**.

If you are working on Windows, your command line may look something like this:

```
c:\WebSphere\PortalServer\bin\xmlaccess.bat -user wpsadmin -pwd itso -url  
http://devportal.redbook.ibm.com:9081/wps/config -in ExportTestPage.xml -out  
ExportedTestPage.xml
```

5. Run the export and ensure that you see the status `element="all" result="ok"/` message at the end of **ExportedTestPage.xml**.

Transferring

To transfer the file, copy (or FTP) the exported XML file (**ExportedTestPage.xml**) to the Production Portal Environment.

Importing

Use the following steps to import the file:

1. Prepare the XMLAccess import. The import is basically the same as the export except that you pass the exported output from the staging environment as the input for the production environment.

The XMLAccess input command line on UNIX may look similar to the following:

```
/WebSphere/PortalServer/bin/xmlaccess.sh -user wpsadmin -pwd itso -url  
http://aixportal1.redbook.ibm.com:9081/wps/config -in ExportedTestPage.xml -out  
Results_ExportedTestPage.xml
```

2. Check the XML results by opening **Results_ExportedTestPage.xml** and check for the `<status element="all" result="ok"/>` message. You may see warnings that the users and groups could not be retrieved from the Portal data store if your LDAP environments are different between the staging and production environments.
3. Check that the page imported correctly by accessing the page in the Portal user interface via a Web browser.

7.4 Moving a differential release using ReleaseBuilder

ReleaseBuilder is the portal utility tool for administrators to manage and maintain their XMLAccess XML files for each release. Before ReleaseBuilder, administrators were faced with a choice of much work using cut-and-paste, merging multiple export files, and creating instructions to delete pages by hand or wiping out the production environment and doing a full import with each release cycle. The first option exposes serious risk of human error, the second requires significant downtime of the portal. ReleaseBuilder simplifies the process by automating the task of finding the differences between the old release and the new. The resulting XML file then contains only the differences between the two configurations.

7.4.1 ReleaseBuilder process overview

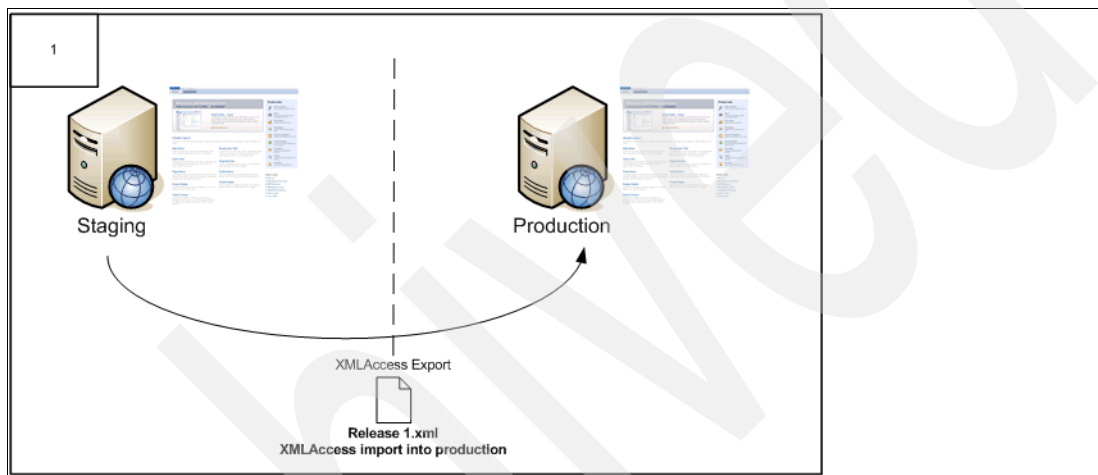


Figure 7-4 Initial release to production

For your first release, you run a full XMLAccess export and import it into your empty Production portal.

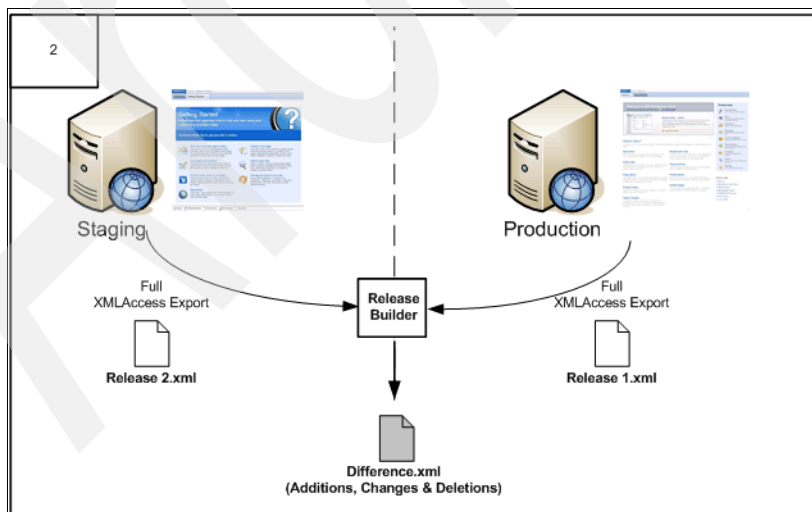


Figure 7-5 Building the differential release file

For subsequent releases, you will use ReleaseBuilder to automate the handling of the XMLAccess files.

1. Take a full export of your staging environment.
2. Take a full export from the production environment (which was originally configured from a full export of the staging environment), or use the previous release file, which should be in your version control system.
3. Use ReleaseBuilder to create the differential XML file.

The resulting XML file will contain only the differences between the environments. New pages and portlets are in the file, as well as changes to existing pages or portlets. The file will also contain instructions to delete artifacts that no longer exist in the new release.

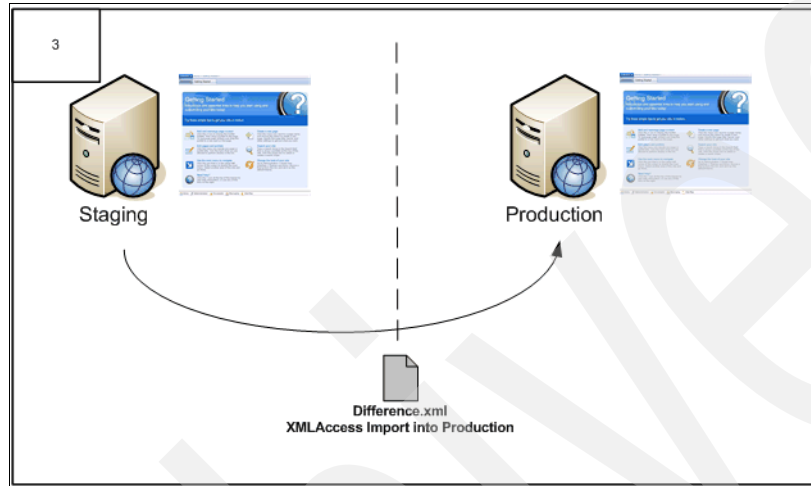


Figure 7-6 Import the differential release file into Production

Once you produce the differential XML file with ReleaseBuilder, you can then copy updated WAR files, themes, and skins and import the configuration into Production. ReleaseBuilder will detect whether or not WAR files are updated, and remove or leave in the <URL> tag pointing to the WAR file as necessary. You still need to copy the updated WAR file to a location that can be reached from the server that you are running the import from, and edit the <URL> tag if necessary to point to the updated file.

7.4.2 Using the ReleaseBuilder tool

ReleaseBuilder is the portal utility tool for administrators to manage and maintain their XMLAccess XML files for each release. The syntax for the command is as follows:

```
[release builder cmd] -inOld FullRelease0.xml -inNew FullRelease1.xml -out Release1.xml
```

On the command line, use the following values:

- ▶ **release builder cmd:** On UNIX, releasebuilder.sh, on Windows, releasebuilder.bat
- ▶ **inOld:** A full XMLAccess export of the old configuration (the current configuration of the Production server)
- ▶ **inNew:** A full XMLAccess export of the new configuration (the current configuration of the Integration server)
- ▶ **out:** The filename you wish it to write the ReleaseBuilder output file to (the release file)

The output of the ReleaseBuilder file is an XMLAccess file containing the difference between the old and new configurations and is used to apply not only addition and update modifications, but also deletions to the production server. This allows the staging and

production servers to remain in sync and eliminates the problem of configuration bloat on the production server.

Note: For performance reasons, we recommend that you run ReleaseBuilder on a separate, standalone machine. This system can be the staging system or a completely separate system on which WebSphere Portal is installed. To maximize use of system resources, WebSphere Portal should not be running during ReleaseBuilder execution.

Preparing to build a release

If you already have an existing production system, but without a defined configuration management scheme in place, the following is how you take control:

Export your current production configuration

Use XMLAccess to do a full XMLAccess export from your production server.

```
/WebSphere/PortalServer/bin/xmlaccess.bat -user wpsadmin -pwd wpsadmin -url  
http://production.portal.ibm.com:9081/wps/config -in ExportRelease.xml -out  
ExportedProdRelease.xml
```

Note: ExportRelease.xml is found under PortalServer/doc/xml-samples.

Prepare your staging environment

If you have not built your staging environment yet, do the install with the flag set to install an empty portal. This environment does not necessarily have to be your *Stage* environment. It should be the lowest environment that you want to keep under configuration management control, which might be Development.

UNIX: `./install.sh -W emptyPortal.active="True"`

Windows: `install.bat -W emptyPortal.active="True"`

If you have an existing staging environment with configurations in it already, empty it by running the empty portal task:

UNIX: `WPSconfig.sh action-empty-portal`

Windows: `WPSconfig.bat action-empty-portal`

Import your production configuration into the staging environment

Use XMLAccess to import the production configuration into the now empty staging environment:

```
/WebSphere/PortalServer/bin/xmlaccess.bat -user wpsadmin -pwd wpsadmin -url  
http://staging.portal.ibm.com:9081/wps/config -in ExportedProdRelease.xml -out  
ExportedProdRelease.out
```

Now your two environments have the same configuration, right down to the object IDs. If you have content ready in your development environment, you may use XMLAccess to transfer it, or you can use the Portal Administrative portlets to set up the staging environment for the next release to production.

7.5 A step-by-step guide to moving a release between environments

In this section we lay out, in a step-by-step fashion, all of the tasks needed to export your portal release from one environment and import it into another. In this example, we assume that you are ready to move Release1 from your staging environment to your production environment, which contains configurations and content from Release0. If your production environment is installed with the option for an Empty Portal, simply skip the steps for exporting the release information from the Production environment and building the differential release XML file.

In Figure 7-7, we show an outline of the steps we walk you through as part of the configuration management process to move your portal solution from your staging environment to your production environment.

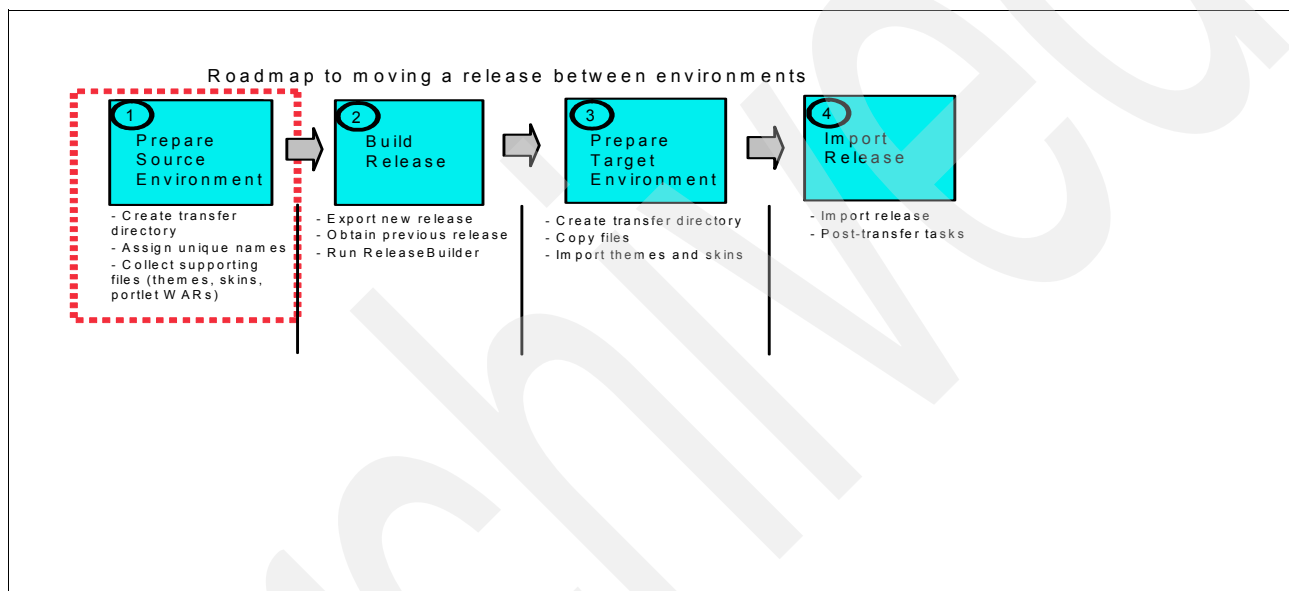


Figure 7-7 Configuration management—step one

7.5.1 Prepare source environment

Before you export or import Portal configurations, we recommend that you prepare the transfer environment to reduce the chances of mistakes and to ensure that the Portal artifacts and configuration information are exported to an easily identifiable location. Preparing the environments is not strictly required, but it is a best practice that we find works in a larger Portal deployment. Preparing the environment allows you to easily identify when artifacts were exported, so that you can revert back to the original files if needed.

Set up transfer directory

You may want to export from the development or staging environments frequently. Unless you first set up the transfer environment correctly, you may easily lose an exported configuration or overwrite them. To set up the environment correctly, create the directory structure shown in Figure 7-8 on page 350 in the `wp_root` directory.



Figure 7-8 Transfer directory

Create a transfer directory under wp_root that will contain a config directory and a directory that represents the current date in mmddyyyy format.

The transfer directory contains two scripts, one for exporting from the current server and one for importing into the current server. These can help mitigate the risk of accidentally typing in the URL for the production server some late night when you are intending to rebuild your development environment from the integration configuration. The config directory contains the XML files that you will use to perform actions using the XMLAccess tool. Copy the file ExportRelease.xml from the PortalServer/doc/xml-samples directory into your transfer/config directory.

The directory that represents the current date is where all the export scripts are run from on that date. All the exported XML files are written to this directory, allowing you to identify it easily in the future.

The import and export scripts and batch files shown in the following examples allow you to control how and where the exported XML files are written.

Example 7-1 Windows: Export.bat

```
C:\WebSphere6\PortalServer\bin\xmlaccess.bat -url  
http://ibm-5ab249a6f6c:10044/wps/config -user wpsadmin -pwd wpsadmin -in  
C:\WebSphere6\PortalServer\transfer\config\%* -out exported_%*
```

Example 7-2 Windows: Import.bat

```
C:\WebSphere6\PortalServer\bin\xmlaccess.bat -url  
http://ibm-5ab249a6f6c:10044/wps/config -user wpsadmin -pwd wpsadmin -in %* -out  
%*.out
```

Example 7-3 UNIX: export.sh

```
/opt/WebSphere/PortalServer/bin/XMLAccess.sh -url  
http://ibm-5ab249a6f6c:10044/wps/config -user wpsadmin -pwd wpsadmin -in $* -out  
exported_$*  
tail -n 3 exported_$*
```

Example 7-4 UNIX: import.sh

```
/opt/WebSphere/PortalServer/bin/XMLAccess.sh -url  
http://ibm-5ab249a6f6c:10044/wps/config -user wpsadmin -pwd wpsadmin -in $* -out  
$*.out  
tail -n 3 $*.out
```

Note that the XMLAccess command should be all on one line, so the Windows batch files have a single line, and the UNIX files have two. The UNIX shell scripts also tail the resulting exported XML file, so that you can determine if the XMLAccess tool was successful. Windows users need to open the file.

Assign custom unique names

Assign custom unique names to pages and portlets. Your organization should have a defined naming standard for all Portal artifacts, similar to the one shown in Table 7-1.

Table 7-1 Example UniqueNames

	Default install	HR	Accounting
Pages	wps.PageName	HR.pageName	Acc.pageName
Portlet applications	None	HR.portletName	Acc.portletName
Portlets	None	HR.Portlet	Acc.Portlet
URL mapping context	NA	HR.Url1	Acc.Url2
User groups	None	HR.GroupName	Acc.GroupName
Web modules	None	HR.AppName	Acc.AppName
Skins	wps.skin.skinName	HR.skin.skinName	Acc.skin.skinName
Themes	wps.theme.themeName	HR.theme.themeName	Acc.theme.themeName

You cannot assign unique names to themes or skins via the Portal Administration Interface. However, you can assign them using the XMLAccess tool. There is a sample file called DeployTheme.xml available to help you set custom unique names for your themes and skins.

To set a custom unique name for your themes and skins, use the following steps:

1. Create the file ExportThemesAndSkins.xml in your transfer/config directory as follows:

Example 7-5 ExportThemesAndSkins.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="PortalConfig_1.4.xsd" type="export"
  create-oids="true">
  <portal action="locate">
    <skin action="export" objectid="" />
    <theme action="export" objectid="" />
  </portal>
</request>
```

2. Open a command prompt in the directory representing today's date.
3. Run the command **export.bat ExportThemesAndSkins.xml**. This should result in a file called exported_ExportThemesAndSkins.xml that looks similar to Example 7-6:

Example 7-6 Output of ExportThemesAndSkins.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- IBM WebSphere Portal/6.0 build wp600_244 exported on Mon Oct 23 14:59:46 EDT
2006 from IBM-5AB249A6F6C/9.33.85.78 -->
<!-- 1 [skin K_N02UF4I1186E1026H4BLVI00E6] -->
<!-- 2 [skin K_N02UF4I1186E1026H4BLVI00E1] -->
<!-- 3 [skin K_N02UF4I1186E1026H4BLVI00E5] -->
<!-- 4 [skin K_N02UF4I1186E1026H4BLVI00E3] -->
<!-- 5 [skin J_8000CB1A08VRD02EETLAMB0004] -->
```

```

<!-- 6 [theme J_N02UF4I1186E1026H4BLVI00E7] -->
<!-- 7 [theme J_8000CB1A08VRD02EETLAMB0000] -->
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" build="wp600_244"
type="update" version="6.0.0.0"
xsi:noNamespaceSchemaLocation="PortalConfig_1.4.xsd">
  <portal action="locate">
    <skin action="update" active="true" default="true" domain="rel"
objectid="K_N02UF4I1186E1026H4BLVI00E6" resourceroot="IBM" type="default"
uniquename="ibm.portal.skin.IBM">
      <localedata locale="en">
        <title>IBM</title>
        <description></description>
      </localedata>
      ...
    </skin>
    <skin action="update" active="true" default="false" domain="rel"
objectid="K_N02UF4I1186E1026H4BLVI00E1" resourceroot="ThinSkin" type="default"
uniquename="wps.skin.thinSkin">
      <localedata locale="en">
        <title>Thin Skin</title>
        <description></description>
      </localedata>
    </skin>
    <skin action="update" active="true" default="false" domain="rel"
objectid="K_N02UF4I1186E1026H4BLVI00E5" resourceroot="NoSkin" type="default"
uniquename="wps.skin.noSkin">
      <localedata locale="en">
        <title>NoSkin</title>
        <description></description>
      </localedata>
      ...
    </skin>
    <skin action="update" active="true" default="false" domain="rel"
objectid="K_N02UF4I1186E1026H4BLVI00E3" resourceroot="IFrame" type="default"
uniquename="wps.iframe.album">
      <localedata locale="en">
        <title>IFrame</title>
        <description></description>
      </localedata>
      ...
    </skin>
    <skin action="update" active="true" default="false" domain="rel"
objectid="J_8000CB1A08VRD02EETLAMB0004" resourceroot="MySkin" type="default"
uniquename="undefined">
      <localedata locale="en">
        <title>MySkin</title>
        <description></description>
      </localedata>
    </skin>
    <theme action="update" active="true" default="true"
defaultskinref="K_N02UF4I1186E1026H4BLVI00E6" domain="rel"
objectid="J_N02UF4I1186E1026H4BLVI00E7" resourceroot="IBM"
uniquename="ibm.portal.theme.IBM">
      <localedata locale="en">
        <title>IBM</title>

```



```

        <description></description>
    </localedata>
    ...
    <allowed-skin skin="K_N02UF4I1186E1026H4BLVI00E1" update="set"/>
    <allowed-skin skin="K_N02UF4I1186E1026H4BLVI00E5" update="set"/>
    <allowed-skin skin="K_N02UF4I1186E1026H4BLVI00E6" update="set"/>
</theme>
<theme action="update" active="true" default="false"
defaultskinref="K_N02UF4I1186E1026H4BLVI00E6" domain="rel"
objectid="J_8000CB1A08VRD02EETLAMB0000" resourceroot="MyTheme"
uniquename="undefined">
    <localedata locale="en">
        <title>MyTheme</title>
    </localedata>
    <allowed-skin skin="K_N02UF4I1186E1026H4BLVI00E1" update="set"/>
    <allowed-skin skin="K_N02UF4I1186E1026H4BLVI00E3" update="set"/>
    <allowed-skin skin="K_N02UF4I1186E1026H4BLVI00E5" update="set"/>
    <allowed-skin skin="K_N02UF4I1186E1026H4BLVI00E6" update="set"/>
</theme>
</portal>
<status element="all" result="ok"/>
</request>

```

Your export will contain several more localedata stanzas for the default languages supported by WebSphere Portal Server, but for brevity, I have stripped out all but the english ones. Notice the theme and skin “MySkin” and “MyTheme”, which both have a uniqueness assignment of “undefined”. This is what we want to change.

4. Create a new file called SetUniqueNames.xml

Example 7-7 SetUniqueNames.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="PortalConfig_1.4.xsd" type="update"
create-oids="false">
    <portal action="locate">
        <skin action="update" objectid="J_8000CB1A08VRD02EETLAMB0004"
uniquename="my.MySkin"/>
        <theme action="update" objectid="J_8000CB1A08VRD02EETLAMB0000"
uniquename="my.MyTheme"/>
    </portal>
</request>

```

5. Compare this file to Example 7-6 on page 351. Notice that we pulled the Object ID references from the previous export to reference the theme and skin definition that we want to assign our custom unique names to. Notice that create-oids is set to false. If you leave this value at true, XMLAccess will reference Object IDs as internal references while processing the script, but create a new instance of a theme and skin with a new Object ID and the custom unique names you want to use, but without the resource root or other configuration information.

This example serves as an object lesson on why we recommend setting custom unique names on all of your resources. With custom unique names, you can avoid running through this exercise again

6. Run this file using the command **import.bat SetUniqueNames.xml**. The output is created in the file **SetUniqueNames.xml.out**, and should look similar to Example 7-8:

Example 7-8 Output from SetUniqueNames.xml

```
<!-- 1 [theme J_8000CB1A08VRD02EETLAMB0004 uniqueness=my.MySkin] -->
<!-- 2 [theme J_8000CB1A08VRD02EETLAMB0000 uniqueness=my.MyTheme] -->
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" build="wp600_244"
type="update" version="6.0.0.0"
xsi:noNamespaceSchemaLocation="PortalConfig_1.4.xsd">
  <status element="all" result="ok"/>
</request>
```

Alternatively, if you want, you can create your themes and skins directly using XMLAccess, or if you have already created them using the GUI, delete the theme and skin definitions through the GUI interface, and define your themes and skins through the XML interface using a script similar to Example 7-9:

Example 7-9 Script to create new theme and skin definitions

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" build="wp600_244"
type="update" version="6.0.0.0"
xsi:noNamespaceSchemaLocation="PortalConfig_1.4.xsd" create-oids="true">
  <portal action="locate">
    <skin action="update" active="true" default="false" domain="rel"
objectId="myskin-oid" resourceroot="MySkin" type="default" uniqueness="my.MySkin">
      <localedata locale="en">
        <title>MySkin</title>
        <description></description>
      </localedata>
    </skin>
    <theme action="update" active="true" default="false"
defaultskinref="myskin" domain="rel" resourceroot="MyTheme"
uniquename="my.MyTheme">
      <localedata locale="en">
        <title>MyTheme</title>
      </localedata>
      <allowed-skin skin="myskin-oid" update="set"/>
    </theme>
  </portal>
</request>
```

Notice, in this case, we set create-oids to true, so that we can use an internal reference to assign the new skin to the new theme, neither one of which has an Object ID assigned to it yet. For more information about how Portal and XMLAccess use Object IDs, please refer to section 7.2.1, "Object IDs" on page 340.

Gather required files

Run build from your version control system of all portlet .WAR files and place them in the transfer directory with today's date. If you have any new or changed themes and skins, place

them in the directory also. Check the entire contents of this directory into version control as your release.

7.5.2 Build the release

Now that you have set up a working environment on the source servers and set up your custom unique names, you are ready to start building your release.

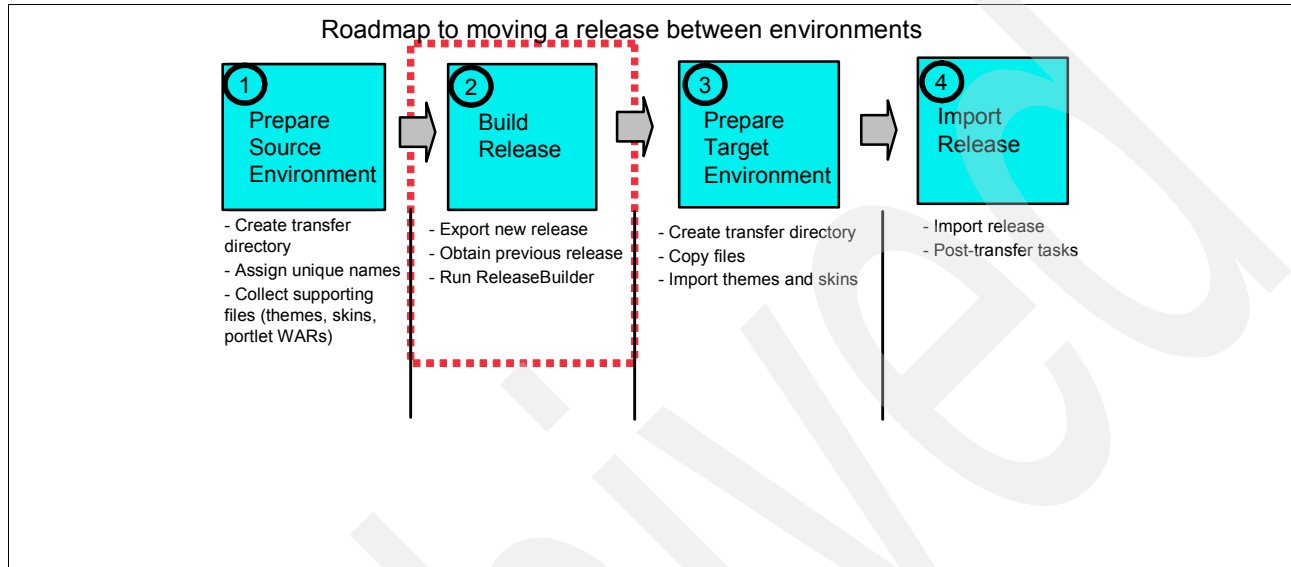


Figure 7-9 Configuration management: step 2

Export new release from the source server

You should have copied the `ExportRelease.xml` file from the `PortalServer/docs/xml-samples` directory as part of the preparing your environment exercise on page 349.

From the transfer directory representing today's date, run the following command:

Windows:

```
..\Export.bat ExportRelease.xml
```

UNIX:

```
../Export.sh ExportRelease.xml
```

This creates an `exported_ExportRelease.xml` file in the current directory.

If you have an `exported_ExportRelease.xml` file from the previous release that is currently running on the target server, you may skip the next section. If there is nothing in the target server that you want to keep (for example, the target server is empty or a new, default configuration from the install process), you need to perform the steps in the next section to make sure that your releases are consistent going forward.

Obtain previous release data

If you still have the `XMLAccess` file stored in your version control system that was used for your initial Portal deployment and have not made any configuration changes through `XMLAccess` or the portal Admin GUI, you can reuse that file for this step. Otherwise, you need to obtain an `XMLAccess` export of the current configuration as follows:

1. If it does not already exist, set up the same transfer directory structure on the target server as in section 7.5.1, “Prepare source environment” on page 349.
2. Create the current date directory representing the date that the current release data was placed into production, NOT today’s date.
3. Run the ExportRelease.xml script as in section 7.5.2, “Build the release” on page 355 from that directory.
4. Copy the release date directory that now contains an exported_ExportRelease.xml file over to the source server on which you will run ReleaseBuilder.

Run Release Builder

From the transfer directory, run the following command:

Windows:

```
..\bin\releasebuilder.bat -inNew todays_release_dir\exported_ExportRelease.xml  
-inOld previous_release_dir\exported_ExportRelease.xml -out  
todays_release_dir\Release.xml
```

UNIX:

```
../bin/releasebuilder.sh -inNew todays_release_dir/exported_ExportRelease.xml  
-inOld previous_release_dir/exported_ExportRelease.xml -out  
todays_release_dir/Release.xml
```

In our example, the *todays_release_dir* was 10232006 and *previous_release_dir* was 10022006. This creates the file necessary to duplicate the configuration from the source server on the target server.

Note: We do not recommend that you run the ReleaseBuilder task on your Production server. Depending on your configuration, it may consume significant CPU and memory resources and have an adverse impact on system performance. Run ReleaseBuilder on a Development or Integration server or on a machine specifically dedicated for this purpose.

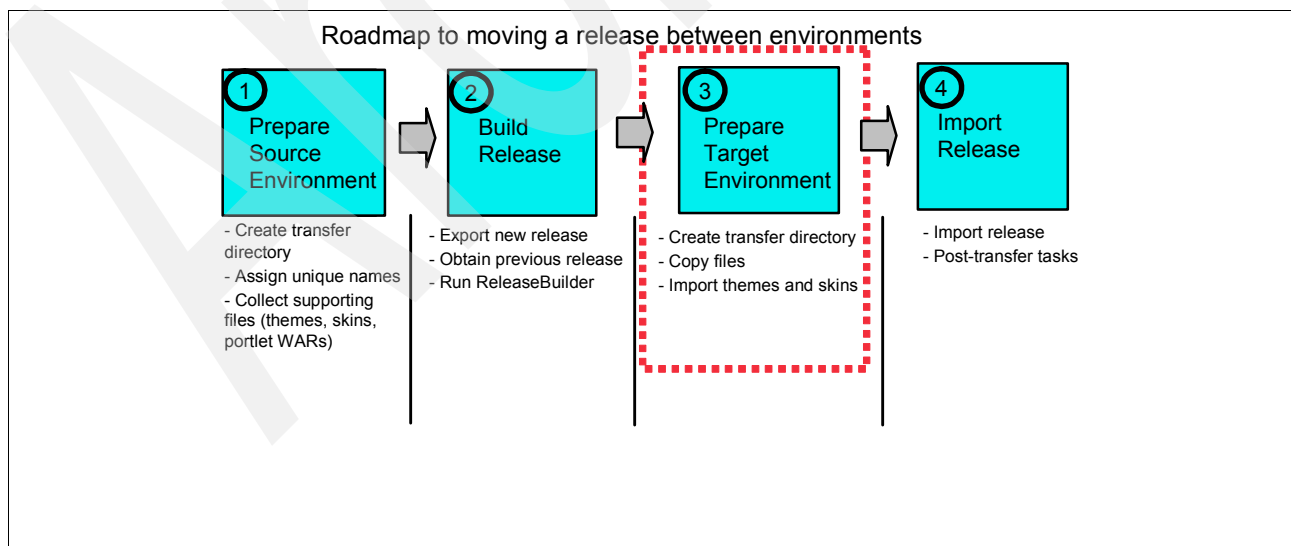


Figure 7-10 Configuration management: step 3

7.5.3 Prepare the target environment

If you are using WCM or Personalization, and you did an empty portal install on the target server, you must configure them before you can run a successful import.

Create transfer directory

1. If it does not already exist, set up the same transfer directory structure on the target server as in section 7.5.1, “Prepare source environment” on page 349.
2. Create the current date directory representing today’s date.
3. Run the ExportRelease.xml script as in section “Export new release from the source server” on page 355 from that directory.

Copy the files to target server

1. Move the entire directory over to the target server.
2. Copy all portlet WAR files into the PortalServer/installableApps directory.
3. Modify the Release1.xml file to change all URL tags to point to the installableApps directory instead of the deployed or archive directory.

Deploy themes and skins

If you have made changes to themes and skins that need to be included in this release, deploy them into the target environment before running XMLAccess. As this requires a portal application restart, it should be done during a scheduled outage. For full instructions on how to deploy themes and skins into a clustered environment, refer to section 7.6.3, “Deploying themes and skins” on page 362.

7.5.4 Import release

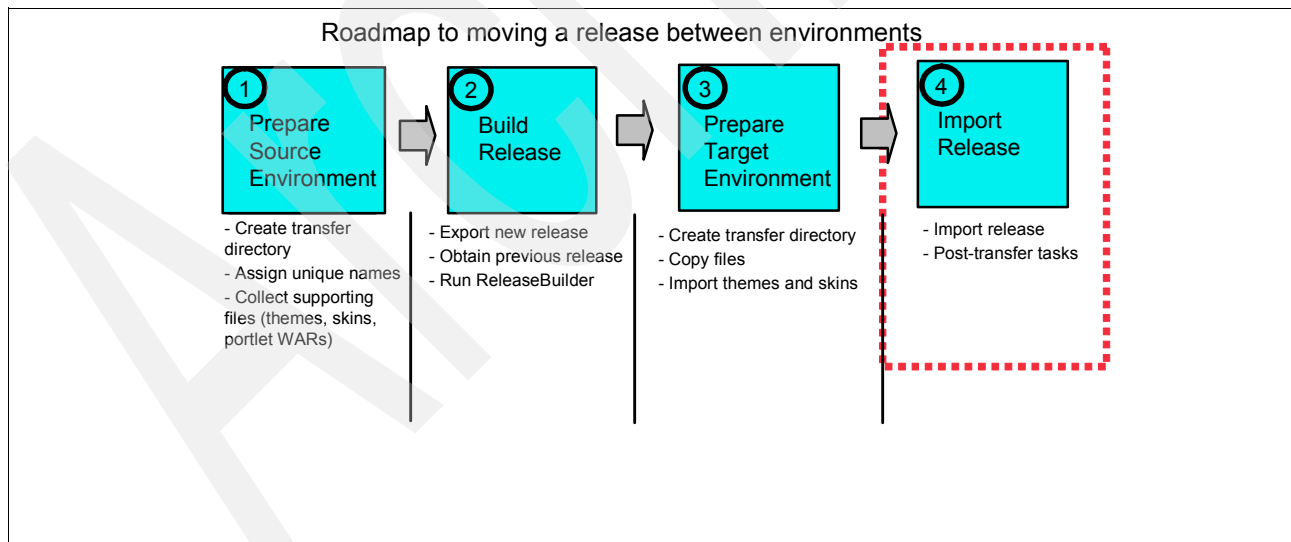


Figure 7-11 Configuration management: step 4

7.5.5 Import the release into the target server

If you updated themes, skins, or screens, deploy them to the server according to the instructions detailed in section 7.6.3, “Deploying themes and skins” on page 362.

From the current date directory on the target server, run the following command:

Windows:

```
..\Import.bat Release.xml
```

UNIX:

```
../Import.sh Release.xml
```

7.5.6 Post transfer actions

If you are deploying into a clustered environment, any updated portlets cannot be activated until the updated code is synchronized down to all the nodes. Perform the following task from the `portal_server_root/config` directory to synchronize the changes across cluster members and to activate all the portlets:

Windows: `WPSconfig.bat activate-portlets`

UNIX: `./WPSconfig.sh activate-portlets`

If any of your portlets have custom configurations that are different between your development, test, and staging environments and the production environment—for example, URLs pointing to the development instance that need to be changed to the production instance, then consider either pulling these from properties files on the file system of the respective environments or scripting your deployments as described in Appendix A, “Deployment automation options” on page 407

7.6 Configuration maintenance tasks

This section covers the configuring maintenance tasks.

7.6.1 Handling orphan data

When portal resources are deleted, dependent resources that are stored in a different database domain are not deleted at the same time. These remaining resources are still available for backup scenarios or other production lines that might share the database domain. For example, when an administrator deletes pages, the user customization to the deleted pages are not deleted. The SLCheckerTool allows you to detect orphaned data that is not needed any more by any of the production lines that share the domain. You can use the SLCheckerTool to prepare an XML script for later deletion of the orphaned data by the XML configuration interface.

The SLCheckerTool is included in the file `wp.db.slchecker.jar` in the following directory:

- ▶ UNIX: `portal_server_root/bin`
- ▶ i5/OS: `portal_server_root/bin`
- ▶ Windows: `portal_server_root\bin`

You invoke the SLCheckerTool by using the following shell scripts:

- ▶ UNIX: `./slcheckertool.sh`
- ▶ i5/OS: `slcheckertool.sh`
- ▶ Windows: `slcheckertool.bat`

To prepare and complete deleting orphaned data you also perform some steps by using the XML configuration interface. For details about the XML configuration interface refer to section 7.2.3, “The XML configuration interface” on page 341.

For details about how to use the XML configuration interface refer to “Using the XMLAccess tool for moving” on page 341. A sample XML script `ExportIncludingOrphanedData.xml` can be found under `PortalServer/doc/xml-samples` that allows you to prepare the orphaned data for work with the SLCheckerTool.

Note: Before deleting the orphaned data by using the SLCheckerTool, secure your databases by making a backup.

To delete all orphaned data, you need to include every production line that shares the database domain. Otherwise, resources that are still valid in a skipped production line might be unintentionally removed.

To delete the orphaned data, proceed with the following steps. For each step, use the target file from the previous step as the source file.

1. Create a full export including orphaned data of each production line that shares the particular domain. To do this, use the XML configuration interface and the XML sample file `ExportIncludingOrphanedData.xml`.
2. Use the SLCheckerTool to create an XML script file to delete the orphaned data. To do this, proceed by the steps given below. Observe the following rules:
 - a. For each step, start the SLCheckerTool with the appropriate parameters as described under that step.
 - b. Replace all file name variables (given in italics) by the complete directory path location and file name.
 - c. For each step, use the target file(s) from the previous step as the source file(s).

3. Invoke the SLCheckerTool by using the following shell scripts:

- UNIX: `./slcheckertool.sh`
- i5/OS: `slcheckertool.sh`
- Windows: `slcheckertool.bat`

4. Find candidates for orphaned data in each production line. Repeat this step for each production line, but specify a different output file for each iteration; otherwise, the results are overwritten. Use the following parameters:

```
[SLCheckerCmd] -find-candidates -s xml_source_file -d cand_target_file -domain domain_identifier
```

Use the following parameters to find the candidates for orphaned data within one production line. Replace the variables as follows:

- **xml_source_file** Specify a full XML export file that you created in the step for creating the XML export including orphaned data.
- **cand_target_file** Specify the target file. The orphaned data candidates are saved to that file.

- **domain_identifier** Specify the database domain in which you want candidates to be searched. Valid values are *comm* for the community database domain, *cust* for the customization database domain, or *all* for both database domains.
5. Identify the orphaned data. Perform this step only once, but with all the candidate files generated by previous steps at once, as this step determines the intersection of all result files, that is, the data that are orphaned in all production lines. Use the following parameters:

```
[SLCheckerCmd] -identify-orphans -s cand_source_files_and_directories -d orph_target_file
```

Use these parameters to identify the orphaned data by matching the information from all the candidate files. Replace the variables as follows:

- **cand_source_files_and_directories** Specify all files that were generated as *cand_target_files* by the substep for finding the candidates. If you specify one or more directories with the files, make sure that these directories contain only candidate files and no other files.
 - **orph_target_file** Specify the target file. The identified orphaned data is saved to that file.
6. Generate an XML script file. You can later use that script for deleting the orphaned data. Use the following parameters:

```
[SLCheckerCmd] -delete-orphans -s orph_source_file -d xml_target_file
```

Use these parameters to generate an XML script file for deleting the orphaned data. Replace the variables as follows:

- **orph_source_file** Specify the file that was generated as the *orph_target_file* in the step for identifying the orphaned data.
- **xml_target_file** Specify the target file. This is the XML script file that contains the information about the orphaned data. You can later use this file to delete the orphaned data.

Note: You can check whether all production lines were considered during the creation of the XML script file. To do this, review the comment in the file header. The header contains information about all full exports that were used.

7. Delete the orphaned data. To do this, invoke the import command that you created in section 7.5.1, “Prepare source environment” on page 349 with the XML script that you obtained as the *xml_target_file* with the orphaned data in the step for generating the XML script. You need to invoke the XML configuration interface with the XML script only on one production line that shares the database domain. This deletes the orphaned data for all production lines that share that database domain.

```
./import.sh xml_target_file
```

or

```
/WebSphere/PortalServer/bin/xmlaccess.sh -user wpsadmin -pwd itso -url  
http://aixportal1.redbook.ibm.com:9081/wps/config -in xml_target_file -out  
DeleteOrphanResults.xml
```

You have removed all orphaned data from your portal database.

7.6.2 Delayed cleanup of deleted portal resources

A new performance enhancement in Portal 6.0 that was introduced in 5.1 is the delayed cleanup of portal resources. When portal resources such as pages, components, or portlet instances are deleted from the portal server, the actual deletion can take a considerable amount of time and have a noticeable performance impact on the portal. Therefore, the actual deletion is scheduled for a later time. You can configure the frequency or disable this feature using the configuration interface. In a test or staging environment, you might want to disable this completely for administrative convenience, although you should leave it on in production for performance reasons.

Running the cleanup task

Use the following steps to run the cleanup task immediately:

1. Copy the file Task.xml from PortalServer/doc/xml-samples or create a new file with the following contents:

Example 7-10 XMLAccess file to perform immediate cleanup of portal resources

```
<?xml version="1.0" encoding="UTF-8"?>
<request type="update" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="PortalConfig_1.4.xsd">
  <portal action="locate">
    <task action="create" bean="ejb/wpsSchedulerTask"/>
  </portal>
</request>
```

2. Run the command using XMLAccess:

```
/WebSphere/PortalServer/bin/xmlaccess.sh -user wpsadmin -pwd itso -url
http://aixportal1.redbook.ibm.com:9081/wps/config -in Task.xml -out
TaskResults.xml
```

Disabling scheduled cleanup

This forces immediate deletions of portal resources, and can be useful in a development or Integration environment. If your portal is running standalone, use the local console. If your portal is installed in a cluster, use the console of the Deployment Manager (DMGR).

Use the following steps to disable scheduled cleanup completely:

1. Start the administrative console by entering the following in the location bar of a Web browser:

`http://example.com:admin_port/ibm/console`

Where *example.com* is the name of your server and *admin_port* is the port assigned to the administrative console.

2. In the left frame, open the Resource section by clicking the plus (+) sign.
3. Click **Resource Environment Providers**.
4. If you are running a clustered environment, set the scope to your portal cluster.
5. In the right frame, select the WP DataStoreService to which you want to make changes.
6. Click Custom Properties.

7. Create a new property called `scheduler.cleanup.enabled` and set the value to `false`. Use `java.lang.String` as its type and do not mark the property as required. Otherwise you will not be able to delete it later.
8. When you are done, click Save at the top of the window under Message(s).
9. Click Save again when prompted to confirm your changes.

If you are running WebSphere Portal in a cluster configuration, replicate your changes to the cluster. (In the Administration Console, select System Administration and then Nodes. Select all of the nodes in your cluster, and click Synchronize).

Restart the portal to make the changes become effective.

Note: This is NOT a recommended setting for a Production environment!

Altering the schedule of the cleanup service

By its default schedule configuration, the cleanup service runs weekly, on Saturdays at 8 pm. You can specify daily, weekly, or monthly cleanup at a specified time. You cannot schedule both a monthly and a weekly task, or a daily and a monthly.

1. Copy the file Task.xml from PortalServer/doc/xml-samples or create a new file with the following contents:

Example 7-11 XMLAccess file to alter scheduled cleanup of portal resources

```
<?xml version="1.0" encoding="UTF-8"?>
<request type="update" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="PortalConfig_1.4.xsd">
  <portal action="locate">
    <task action="create" bean="ejb/wpsSchedulerTask">
      <dayOfMonth>31</dayOfMonth>
      <startTime>23:59</startTime>
    </task>
  </portal>
</request>
```

2. Run the following command using XMLAccess:

```
/WebSphere/PortalServer/bin/xmlaccess.sh -user wpsadmin -pwd itso -url
http://aixportal1.redbook.ibm.com:9081/wps/config -in Task.xml -out
TaskResults.xml
```

Setting a dayOfMonth value of 31 causes the cleanup task to be done on the last day of the month, for example on February 28, September 30, or October 31.

To schedule weekly cleanup, substitute dayOfWeek for dayOfMonth where, 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday, 7=Sunday.

To schedule daily cleanup, simply remove the dayOfMonth tag completely, leaving only the startTime.

7.6.3 Deploying themes and skins

This approach does not require the enabling of automatic JSP reloading and does not affect the portal's run-time performance. Because of this, it is the recommended way to deploy themes and skins in a production environment.

Note: Before you start this task, make sure that all of the following is true:

The administrative server of the WebSphere Application Server is started:

- ▶ If your portal runs in a base application server node, make sure that the administration server is running. The default name is server1.
- ▶ If your portal runs in a Network Deployment (ND) cell, make sure that the Deployment Manager (DMGR) and all node agents are running.

To deploy themes and skins in a production environment, proceed as follows:

1. Export the WebSphere Portal EAR file, `wps.ear`, according to your network configuration. If you have a cluster environment, the WebSphere Portal EAR must be exported from the WebSphere Application Server Network Deployment machine.
2. At a command line, change to the `was_profile_root/bin` directory.
3. Invoke the `wsadmin` command to export the `wps.ear` to a temporary directory `wps_expanded` (make sure that all commands are entered on one line):

- **Windows:** `wsadmin.bat -user admin_user_id -password admin_password -c "$AdminApp export wps directory/wps.ear"`
- **UNIX:** `./wsadmin.sh -user admin_user_id -password admin_password -c '$AdminApp export wps directory/wps.ear'`

where:

- `was_profile_root` is the name of the WebSphere Application Server profile where WebSphere Portal is installed; for example, `/opt/WebSphere/AppServer/profiles/wp_profile` or `C:\IBM\websphere\profiles\wp_profile`.
- `admin_user_id` is the administrator's user ID
- `admin_password` is the administrator's password
- `directory` is the temporary directory; for example, `c:\temp` or `/tmp`

4. Create the `/wps_expanded` subdirectory. Use the `EARExpander` tool to expand the contents of the exported EAR file (make sure that all commands are entered on one line):

- **Windows:** `EARExpander.bat -ear directory\wps.ear -operationDir directory\wps_expanded -operation expand`
- **UNIX:** `./EARExpander.sh -ear directory/wps.ear -operationDir directory/wps_expanded -operation expand`

5. Place the updated themes and skins' JSPs into the correct directory within the expanded EAR. For example:

- HTML themes go in `directory/wps_expanded/wps.war/themes/html`
- HTML skins go in `directory/wps_expanded/wps.war/skins/html`

6. Delete the original the `wps.ear` file from the directory where you initially exported it.

7. Use the `EARExpander` command to collapse the EAR directory back into an EAR file:

- **Windows:** `EARExpander.bat -ear directory\wps.ear -operationDir directory\wps_expanded -operation collapse`
- **UNIX:** `./EARExpander.sh -ear directory/wps.ear -operationDir directory/wps_expanded -operation collapse`

8. Use the `wsadmin` command to update the WebSphere Portal EAR.

Note: If you have a managed cell (with or without a cluster), perform this step on the DMGR machine.

- **Windows:** `wsadmin.bat -user admin_user_id -password admin_password -c "$AdminApp install directory/wps.ear {-update -appname wps -nodeployejb}"`
- **UNIX:** `./wsadmin.sh -user admin_user_id -password admin_password -c '$AdminApp install directory/wps.ear {-update -appname wps -nodeployejb}'`

where:

- *admin_user_id* is the administrator's user ID
- *admin_password* is the administrator's password
- *directory* is the temporary directory

9. Restart the wps application using the administrative console. In a managed node or cluster configuration, perform this restart using the administrative console for the DMGR.
10. Log in to the administrative console.
11. Select Applications → Enterprise Applications, and scroll through the list until you locate the wps application.
12. Select the wps application, and click Stop.
13. After the application is fully stopped, click Start.

Note: Updates to the configuration of a cluster must occur on the DMGR and be resynchronized with the other nodes in the cluster. If updates are made to individual nodes in the cluster, the updates will be lost when the master configuration on the DMGR resynchronizes with the nodes again as changes on a node are overwritten. When you run configuration tasks on the nodes, however, you do initiate them on the nodes but must modify the master configuration (remote connection from the node to the DMGR).

7.7 Troubleshooting

This section provides a few hints for discovering and solving some common issues, problems, and mistakes that can occur when using XMLAccess to transfer Portal configurations between environments.

XMLAccess imports can be an iterative adventure. There is no utility to scan the import file for invalid paths, correct syntax, or missing references. The process will run and perform any actions that it understands and fail when it reaches a section it does not understand. It is not transactional, so nothing is rolled back after a failure. If you have `create-oids="true"` as an attribute on the request stanza, it will create additional copies of each object that it created already when you re-run the XMLAccess script—unless you manually cut out the sections that were already performed.

7.7.1 Problems importing portlets

There are two common problems when working with XMLAccess to deploy and update portlets:

Improper path to the portlet WAR file

XMLAccess exports XML files with paths to the WAR file for update, but it only creates pseudo-references in the form of file URLs that rely on the assumption that the file resides under the deployed/archive subdirectory of the WPS installation, as in the following example:

```
<url>file://localhost/C:/WebSphere6/PortalServer/deployed/archive/com.ibm.wps.portlets.resourceview.ResourceView/ResourceView.war</url>
```

If these assumptions are not met, an exported portal configuration cannot be successfully re-created without editing the generated URLs manually. Fortunately, there are two simple solutions to this:

1. Copy the WAR files from the deployed or archive directory on the source server to the target server.
2. Change all the URL references in the XML import file to point to the actual locations of the files. You can set up a Web server somewhere to serve up your portlet WAR files and change the <URL> targets to use a referenceable URL instead of a file:/// tag, as in **http://webserver/ResourceView.war**. This could be particularly useful if you want to be able to deploy to many different portal servers.

Failure to activate portlets after XMLAccess deploys portlets in a cluster

When portlets are updated in a cluster, they need to be synchronized down to the nodes from the DMGR before they can be activated. Monitor the nodeagent log files on the cluster members to determine whether they are all synchronized, and then run the config task `WPSCfg.sh activate-portlets`.

TIP: If you have a large number of portlets listed in your XMLAccess import file that are already installed on the target server, and you only wish to update configuration information and you do not need to redeploy the code, you may remove the entire <URL> tag from the XML file and save a significant amount of time on the XMLAccess import by not redeploying the portlet code.

7.7.2 Problems importing pages

If you export just pages, and not the full export, the configurations of portlets on those pages are not exported. For example, if you export the Welcome page from the default install configuration, you will see entries such as in Example 7-12:

Example 7-12

```
<web-app action="locate" domain="rel" objectId="1_N02UF4I1186E1026H4BLVI0034"
uid="com.ibm.wps.portlets.welcome">
  <servlet action="locate" domain="rel"
objectId="V_N02UF4I1186E1026H4BLVI0032" referenceid="portletidwelcome"/>
  <portlet-app action="locate" domain="rel" name="Welcome"
objectId="2_N02UF4I1186E1026H4BLVI0036" uid="com.ibm.wps.portlets.welcome.1">
    <portlet action="locate" domain="rel" name="Welcome Portlet"
objectId="3_N02UF4I1186E1026H4BLVI0031" uniquenessname="wps.p.Welcome"/>
  </portlet-app>
</web-app>
```

This is necessary for XMLAccess to find the portlets.

7.7.3 Problems importing policies

A new feature in WP6.0 is the ability to specify and apply common and specialized settings that determine how portal resources function for different classes of users.

A policy is a collection of settings that influences the behavior of a portal resource and the experience that users have when working with the resource. Policies simplify the management of portal resources because the policy settings for a resource type can control

the behavior of the resource for different classes of users. For sites that comprise large numbers of users and resources, specifying and applying policies to portal resources eliminates much time-consuming effort that would otherwise require administrators to manage a vast array of discrete settings for users and resources.

For more information about using Policies, refer to the InfoCenter at the following Web address:

http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/topic/com.ibm.wp.ent.doc/policies/i_pol_c_intro.html

Like working with portlets, using XMLAccess to configure policies takes a little additional work on the operators part. Policy configurations are stored in a separate file, so that file must be copied over from the source server to the target server. By default, it is found in the PortalServer root directory. It can be copied over to the same directory on the target server, or you can adjust the URL referenced in the XML file to the location of the file on the target server.

Troubleshooting and monitoring

This chapter describes approaches to problem determination and troubleshooting. It provides the necessary steps to take in order to determine causes of failures and other error messages. It also provides tips on how to spot problems early on, and how to avoid issues altogether.

In this chapter, the following topics are discussed:

- ▶ Problem determination
 - Configuration problems
 - Installation problems
 - Upgrade problems
 - Runtime problems
- ▶ Tools for Problem determination
 - IBM Support Assistant (ISA)
 - Automatic Problem Determination (AutoPD)
- ▶ Self-help and monitoring tips

8.1 Problem determination approaches

This section is an overview of several different Portal error scenarios and procedures on how to properly troubleshoot and debug failures. The information provided is intended to help you understand the different types of WebSphere Portal failures you may encounter and the appropriate steps you need to take to begin problem determination.

8.1.1 WPSconfig task failure

Definition of a configuration task failure

WebSphere Portal uses configuration tasks to configure Portal to LDAPs, databases, and so forth. The configuration tasks are a series of ANT scripts that are invoked by running the WPSconfig script from the *portal_root/config* directory. When invoking a configuration task you are simply invoking a parent task, WPSconfig.bat database-transfer for example, that is made up of a series of sub-tasks.

Failures during the configuration tasks are common and this information is intended to help you understand how the tasks work and how you can begin problem determination when a configuration task fails.

First it is important to understand the architecture behind the WPSconfig script. When running the WPSconfig script the script reads values defined in properties files located in the *portal_root/config* directory:

- ▶ Portal 5.1.x
 - wpconfig.properties
- ▶ Portal 6.0
 - wpconfig.properties
 - wpconfig_dbdomain.properties
 - wpconfig_dbtype.properties
 - wpconfig_sourceDb.properties

The values are defined by the user prior to running the configuration task by manually editing the appropriate property file, for example:

```
WasUserId=uid=knussbaum,ou=test,dc=ibm,dc=com
```

Portal also provides two alternative ways to edit the property files by using the helper files located in the *portal_root/config/helpers* directory, and the Config Wizard located in the *portal_root/config/wizard* directory. However, the specifics on using these tools are beyond the scope of these instructions. Both tools ultimately edit the same property files listed above, but just provide a more user friendly alternative than manually editing a file by hand. So the architecture is no different when it comes to troubleshooting.

Incorrect values defined in the property files are the most common reason for configuration task failures. The WPSconfig task output is written to a file located at *portal_root/log/ConfigTrace.log*. This log contains a history of all configuration tasks that were executed in the Portal environment. This log is never overwritten and is appended to each time another configuration task is executed.

Important: Please **never** remove the ConfigTrace.log. It contains the configuration history of the Portal and is vital for support to be able to troubleshoot configuration problems.

Analyze failures with configuration tasks

You have executed a WPSconfig task and have received a **BUILD FAILED** message. How do you begin problem determination?

Step 1

Review the ConfigTrace.log to determine the nature and timing of the failure. When opening the ConfigTrace.log, you should immediately scroll to the bottom of the file to review the most recent failure. At the bottom of the log, you should see something similar to the following:

```
wmm.XDbName=wps6TCP
work.dir=C:/PROGRA~1/IBM/WEBSPH~1/PORTAL~1/config/work
wps.classes.dir=C:/PROGRA~1/IBM/WEBSPH~1/PORTAL~1/bin
ws.ext.dirs=C:/Program Files/IBM/WebSphere/AppServer/java/lib;C:/Program
Files/IBM/WebSphere/AppServer/classes;C:/Program
Files/IBM/WebSphere/AppServer/lib;C:/Program
Files/IBM/WebSphere/AppServer/installedChannels;C:/Program
Files/IBM/WebSphere/AppServer/lib/ext;C:/Program
Files/IBM/WebSphere/AppServer/web/help;C:/Program
Files/IBM/WebSphere/AppServer/deploytool/itp/plugins/com.ibm.etools.ejbdeploy/r
untime;../bin;../shared/app;
wsadminConnType=NONE
---- End dump of properties ----

BUILD FAILED
file:../config/actions/jcr_cfg.xml:1787: Java returned: 1

Total time: 13 minutes 26 seconds
```

There are a few important things to note here:

1. The **BUILD FAILED** indicates that the task has not completed successfully. You should stop, determine the cause of the failure, correct the failure, and re-run the task. This should be done **before** moving forward with any further configurations to the environment.
2. The line **file:../config/actions/jcr_cfg.xml:1787: Java returned: 1** usually does **not** contain any initial meaningful information about the failure, but does contain valuable information needed for deeper problem determination if required. The `/config/actions/jcr_cfg.xml:1787` tells us the specific location (`/config/actions/`), XML file name (`jcr_cfg.xml`), and line number (1787) where the failure occurred. Again, this can be used for deeper problem determination and will be discussed later.
3. Notice the `---- End dump of properties ----` line. Each time a WPSconfig task finishes its dumps the property values that were used (defined in the `wpsconfig` files) when this specific task was executed. This dump of properties is quite long and is bookended by the `---- Begin dump of properties ----` and `---- End dump of properties ----` lines.

At this point, you should be at the bottom of the ConfigTrace.log and looking at the **BUILD FAILED** message as noted above. To continue with problem determination you should now scroll up the ConfigTrace.log until you reach the beginning of the property dump section. Directly above the `---- Begin dump of properties ----` is where the specific problem occurred and problem determination can begin, for example:

```
Target started: action-configure-content-security

action-configure-content-security:
[xmlaccess] EJPXB0006I: Connecting to URL http://localhost:10038/wps/config/
[xmlaccess] EJPXB0002I: Reading input file
C:\PROGRA~1\IBM\WEBSPH~1\PORTAL~1\config\work\ContentAdminGroupsPAC.xml
Error 404: Initialization of one or more services failed.
```

```

[xmlaccess] EJPXB0015E: Server response indicates an error.
[xmlaccess] EJPXB0015E: Server response indicates an error.
[xmlaccess] EJPXB0006I: Connecting to URL http://localhost:10038/wps/config/
[xmlaccess] EJPXB0002I: Reading input file
C:\PROGRA~1\IBM\WEBSPH~1\PORTAL~1\config\work\ContentUserGroupsPAC.xml
Error 404: Initialization of one or more services failed.
[xmlaccess] EJPXB0015E: Server response indicates an error.
[xmlaccess] EJPXB0015E: Server response indicates an error.
Wed Aug 23 13:01:24 PDT 2006
Target started: action-init-accesscontrol

action-init-accesscontrol:
    [echo] Calling ContentModelInitializer
Target finished: action-init-accesscontrol
Target finished: action-configure-content-security
Target finished: enable-security-ldap
---- Begin dump of properties ----

```

Here you can see specific details about when the error occurred (what sub-task) and some error codes that may provide clues on the nature of the error.

1. Notice the **action-configure-content-security**. This tells us what sub-task the error occurred.
2. The lines under action-configure-content-security shows us what actions were being attempted by the sub-task. In this case, you can see that an XMLAccess command is attempted.

Step 2

After identifying the relevant error, begin problem analysis. Sometimes failures lead to previous sub-tasks that indicate the root cause. In many cases, WPSconfig tasks contain sub-tasks that stop and start WebSphere_Portal. Many times the root cause of a WPSconfig task failure can be traced back to a Portal startup problem that occurred during the WPSconfig task. The example below shows such a scenario.

1. The ConfigTrace.log shows this specific error:

```

[xmlaccess] EJPXB0006I: Connecting to URL http://localhost:10038/wps/config/
[xmlaccess] EJPXB0002I: Reading input file
C:\PROGRA~1\IBM\WEBSPH~1\PORTAL~1\config\work\ContentAdminGroupsPAC.xml
Error 404: Initialization of one or more services failed.
[xmlaccess] EJPXB0015E: Server response indicates an error.

```

Here you can see that XMLAccess is attempting to contact Portal via the URL *http://localhost:10038/wps/config/*; however, the attempt results in the following:

```

Error 404: Initialization of one or more services failed.
[xmlaccess] EJPXB0015E: Server response indicates an error.

```

2. Knowing that XMLAccess requires Portal to be started, the error gives you clues that Portal did not start properly. Reviewing the previous sub-task shows the following:

```

start-portal-server:
[logmsg] 2006.08.23 12:57:08.332 start-portal-server
[logmsg] EJPCA3163I: Starting Server "WebSphere_Portal"

[echo] 'WebSphere_Portal' seems to be stopped.
[echo] Starting 'WebSphere_Portal'
[exec] ADMU0116I: Tool information is being logged in file

```

```

[exec]
C:\ibm\WebSphere1\profiles\wp_profile\logs\WebSphere_Portal\startServer.log
[exec] ADMU0128I: Starting tool with the wp_profile profile
[exec] ADMU3100I: Reading configuration for server: WebSphere_Portal
[exec] ADMU3200I: Server launched. Waiting for initialization status.
[exec] ADMU3000I: Server WebSphere_Portal open for e-business; process id
is 2764
Target finished: start-portal-server
Wed Aug 23 13:00:16 PDT 2006
Target started: action-check-cm

```

```

action-check-cm:
Target finished: action-check-cm
Wed Aug 23 13:00:16 PDT 2006
Target started: action-configure-content-security

```

```

action-configure-content-security:
[xmlaccess] EJPXB0006I: Connecting to URL http://localhost:10038/wps/config/
[xmlaccess] EJPXB0002I: Reading input file
C:\PROGRA~1\IBM\WEBSPH~1\PORTAL~1\config\work\ContentAdminGroupsPAC.xml
Error 404: Initialization of one or more services failed.
[xmlaccess] EJPXB0015E: Server response indicates an error.
[xmlaccess] EJPXB0015E: Server response indicates an error.
[xmlaccess] EJPXB0006I: Connecting to URL http://localhost:10038/wps/config/
[xmlaccess] EJPXB0002I: Reading input file
C:\PROGRA~1\IBM\WEBSPH~1\PORTAL~1\config\work\ContentUserGroupsPAC.xml
Error 404: Initialization of one or more services failed.
[xmlaccess] EJPXB0015E: Server response indicates an error.
[xmlaccess] EJPXB0015E: Server response indicates an error.
Wed Aug 23 13:01:24 PDT 2006
Target started: action-init-accesscontrol

```

```

action-init-accesscontrol:
[echo] Calling ContentModelInitializer
Target finished: action-init-accesscontrol
Target finished: action-configure-content-security
Target finished: enable-security-ldap

```

As you can see, the sub-task reports that WebSphere_Portal started and is opened for e-business. However, we should review the SystemOut.log to see if any startup problems actually occurred. You can match the time stamp from above to the SystemOut.log to ensure you are reviewing the correct startup:

```

start-portal-server:
[logmsg] 2006.08.23 12:57:08.332 start-portal-server
[logmsg] EJPCA3163I: Starting Server "WebSphere_Portal"

```

The corresponding WebSphere_Portal startup from the *portal_root/log/SystemOut.log* shows the following:

```

[8/23/06 12:58:46:736 PDT] 0000000a CacheServiceI I DYN1001I: WebSphere
Dynamic Cache instance named ws/com.ibm.wps.ac.ApplicationRoleChildrenCache
initialized successfully.
[8/23/06 12:58:46:736 PDT] 0000000a CacheServiceI I DYN1001I: WebSphere
Dynamic Cache instance named ws/com.ibm.wps.ac.ContainedRolesCache initialized
successfully.
[8/23/06 12:58:47:080 PDT] 0000000a Servlet E com.ibm.wps.engine.Servlet
init EJPFD0016E: Initialization of service failed.

```

```

com.ibm.wps.ac.DomainAdministratorNotFoundException: EJPSB0107E: Exception
occurred while retrieving the identity of the domain adminuser/admingroup
cn=r9wpsadmin.
    at
com.ibm.wps.ac.impl.AccessControlDataManagementServiceImpl.convertDNtoObjectID(
AccessControlDataManagementServiceImpl.java:728)
    at
com.ibm.wps.ac.impl.AccessControlDataManagementServiceImpl.initializeDomainConf
ig(AccessControlDataManagementServiceImpl.java:662)
    at
com.ibm.wps.ac.impl.AccessControlDataManagementServiceImpl.reinit(AccessControl
DataManagementServiceImpl.java:599)
    at
com.ibm.wps.ac.impl.AccessControlDataManagementServiceImpl.init(AccessControlDa
taManagementServiceImpl.java:361)
    at com.ibm.wps.services.ServiceManager.createService(Unknown Source)

```

The startup errors above are the root cause of the WPSconfig task failure. Specifically the **EJPSB0107E** error code.

Step 3

As you can see in the SystemOut.log file at the end of Step 2, following is the error:

```

[8/23/06 12:58:47:080 PDT] 0000000a Servlet      E com.ibm.wps.engine.Servlet
init EJPF0016E: Initialization of service failed.

```

```

com.ibm.wps.ac.DomainAdministratorNotFoundException:
EJPSB0107E: Exception occurred while retrieving the identity of the domain
adminuser/admingroup cn=r9wpsadmin.

```

This is the error that appears to be the root cause. Now that you determined the root cause, you can begin searching for a solution to the **EJPSB0107E** error code using the IBM Support Assistant (ISA), located at the following Web site.

<http://www-306.ibm.com/software/support/isa/>

Step 4

If the problem determination approach previously mentioned does not reveal anything, you can use a more advanced method by reviewing the actual XML configuration file. Previously we spoke about examining the file: `../config/actions/jcr_cfg.xml:1787`: Java returned: 1 line that appears at the BUILD FAILED output:

```

BUILD FAILED
file:../config/actions/jcr_cfg.xml:1787: Java returned: 1

```

Total time: 13 minutes 26 seconds

This file can be found at `portal_root/config/actions/jcr_cfg.xml`. Reviewing this file shows the following lines of code. The example below contains lines 1785 - 1798. Line 1787 is highlighted below:

```

<property name="JcrDbLibrary" value="${jcr.DbType}.DbLibrary"/>
    <echo message="Calling ContentModelInitializer" />
    <java classname="com.ibm.content.init.ContentModelInitializer" fork="yes"
failonerror="yes" dir="${WpsInstallLocation}/shared/app"
output="${WpsInstallLocation}/log/cmInit.log">
        <jvmarg value="-Dfile.encoding=${file.encoding}" />

```

```

        <jvmarg value="\${jvmArgForZos}" />
        <sysproperty key="db2j.system.home"
value="\${WpsInstallLocation}/cloudscape" />
        <arg value="-userName" />
        <arg value="\${PortalAdminId}" />
        <arg value="-password" />
        <arg value="\${PortalAdminPwd}" />
        <arg value="-url" />
        <arg value="http://\${WpsHostName}:\${WpsHostPort}/contentapi/init" />
        <arg value="-adminingroup" />
        <arg value="\${WpsContentAdministrators}" />

```

There are a few important things to note here:

1. Line 1787 points us to another log file named `cmInit.log`. Reviewing this log shows the following:

```

Aug 23, 2006 1:01:25 PM com.ibm.content.init.ContentModelInitializer main
INFO: Initializing Nodes and Node Types
entry - userName = cn=r9wpsadmin, strUrl =
http://<hostname>:10038/contentapi/init
Making remote connection...
Returning NULL
Finished connection. Response code: 401
Exception in thread "main" com.ibm.content.exception.ServiceException:
EJPVJ0036E: Initialization failed.
at
com.ibm.content.init.ContentModelInitializer.remoteInitialize(ContentModelIniti
alizer.java:829)
at
com.ibm.content.init.ContentModelInitializer.main(ContentModelInitializer.java:
674)

```

This log provides further insight into what is going on during WebSphere_Portal startup when the error occurs.

2. Notice the variables that are included in the code, example:

```

\${jcr.DbType}.DbLibrary}
\${PortalAdminId}

```

These variables indicate the corresponding properties that are being used from the `wpconfig` property files. Many times you can use this method to determine the property that is incorrect in the `wpconfig` property files.

Conclusion

By following these approaches, you should be able to determine the failure for many of the **BUILD FAILED** messages when running a WPSconfig task within IBM WebSphere Portal. By implementing this troubleshooting approach, you should be able to increase your self-sufficiency and might avoid the need for creating a PMR with IBM support. If the suggestions do not provide the relief you need however, do contact IBM support and provide the information that is required by using ISA to collect your logs.

8.1.2 Portal install failure

Definition of an installation failure

During WebSphere Portal 6.0 installation, you may receive messages indicating that the installation is not successful. The following information is intended to help you understand

what happens during installation, and how you can begin problem determination and analysis when installation fails.

It is important to understand the events that take place during the WebSphere Portal 6.0 installation. There are two different types of Portal installations: a typical installation or a custom installation.

- Installation order for a **Typical installation** (standalone):
 - a. WebSphere Application Server base is installed and upgraded to version 6.0.2.9.
 - b. WebSphere Process Server base is installed and upgraded to version 6.0.1.1 (optional).
 - c. Required WebSphere Application interim fixes are applied.
 - d. Required WebSphere Process server interim fixes are applied (if Process Server was installed).
 - e. Application Server/Process Server profile is created.
 - f. WebSphere Portal 6.0 is installed.
 - g. Configuration tasks are executed that include setting up the Cloudscape database and enabling security.

In a Typical installation, WebSphere Application Server and WebSphere Process Server are installed and upgraded for you. If any of these steps fail, the Portal installation will fail as well.

- Installation order for a **Custom installation** (standalone or managed):
 - a. WebSphere Portal 6.0 is installed.
 - b. Configuration tasks are executed that include creating the Application Server profile (if standalone), setting up the Cloudscape database, and enabling security.

In a Custom installation, WebSphere Application Server must already be installed and at the required version level for WebSphere Portal. The Custom installation also gives you an option to install to a managed node. This installs Portal into a pre-existing profile on a node that was federated into a Deployment Manager (DMGR). If you select this option, the WebSphere Portal installer will not create a WSAS profile because the WSAS profile is already required to exist on the federated node.

Installation failures commonly come from the configuration tasks that are executed during installation. The WebSphere Portal installation generates a number of logs that are useful in problem determination. Most of these logs can be found in the system defined **temp** directory. After the portal_root/log directory is created, some of the log files in the /tmp directory are copied to the portal_root/log directory. There are several log files created during install. The following table discusses some of the more important log files and their locations:

Table 8-1 A list of important installation logs

Log	Location	Description
installmessages.txt	/tmp	Contains messages that are generated during installation. The messages in this file are translated for the language that is specified during installation.
wpinstalllog.txt	/tmp /portal_root/log	Contains trace information that is generated by the installation program.

Log	Location	Description
LocalizerTrace.archive5.log	/portal_root/log	Contains portal archive install and fixup messages.
ConfigTrace.log	/portal_root/log	Contains additional information for configuration tasks executed during installation.
log.txt	/tmp /app_server_root/log	Contains trace information generated during the installation of WebSphere Application Server.

Analyze wpsinstalllog.txt

The wpsinstalllog.txt file is one of the more important log files generated during installation. It contains trace information that is generated by the installation program for every piece of the installation. Here you will review the wpsinstalllog.txt from the portal_root/log directory for a successful Typical installation so that you can understand the events that take place. Knowing this will be helpful in debugging installation issues where the problem is not obvious.

The wpsinstalllog.txt file for a Typical installation can logically be divided into the following sections:

- Validation
- WebSphere Application Server installation
- WebSphere Process Server installation
- WebSphere Application Server fixes installation
- Profile creation
- WebSphere Portal installation
- Enabling WebSphere Application Server Security
- Configure BPE
- Stop and Start WebSphere_Portal server

Validation

You should be at the top of the file and scrolling down. The first several lines are similar to the following:

Example 8-1 Validation

```
(Oct 30, 2006 3:01:19 PM), MultiPlatform.install,
com.ibm.wps.install.DeleteFilesAction, dbg, Not found:
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\wpsinstalllog.txt
(Oct 30, 2006 3:01:19 PM), MultiPlatform.install,
com.ibm.wps.install.DeleteFilesAction, dbg, Not found:
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\log.txt
(Oct 30, 2006 3:01:19 PM), MultiPlatform.install,
com.ibm.wps.install.DeleteFilesAction, dbg, Not found:
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\wpulimit.sh
(Oct 30, 2006 3:01:19 PM), MultiPlatform.install,
com.ibm.wps.install.SetTitleAction, msg2, About to set wizard title to: WebSphere
Portal Version 6.0 Installer
```

```
(Oct 30, 2006 3:01:19 PM), MultiPlatform.install,  
com.ibm.wps.install.LogCommandLineArgs, msg1, Command line parameters: -W  
config.arguments=-DskipWTP=true -W defaults.cdLocation=Y:\WPS60\Windows\W-Setup\
```

The format of each line is the same throughout the entire log. Look at the first line:

```
(Oct 30, 2006 3:01:19 PM), MultiPlatform.install,  
com.ibm.wps.install.DeleteFilesAction, dbg, Not found:  
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\wpsinstalllog.txt
```

Each section of the line is separated by commas (,).

- ▶ The first section is the date and time.
- ▶ The second section names the process running.
- ▶ The third section shows you what component of the process is being executed at that time.
- ▶ The final section shows you the exact action being taken.

In Example 8-1 on page 375, the Portal installer is cleaning up old log files and cannot find the wpsinstalllog.txt file, so no action is taken.

Note: For the remaining examples from the wpsinstalllog.txt, the date, time, and the name of the process running are removed from each line in order to make the examples easier to read and understand.

From Example 8-1 on page 375, the following line is very important for debugging purposes:

```
com.ibm.wps.install.LogCommandLineArgs, msg1, Command line parameters: -W  
config.arguments=-DskipWTP=true -W defaults.cdLocation=Y:\WPS60\Windows\W-Setup\
```

This tells you what command line parameters were used when the installation program was executed.

Scroll down the log, and you will see several lines similar to the following:

```
com.ibm.wps.install.LogEntryAction, msg1, Logging: archiveInstall.choice=archive
```

These logging statements print out various environment variables and variables used during installation. For troubleshooting and debugging purposes, you can ignore these.

Continue to scroll down the log. You will find several PortalValidationAction statements similar to the following:

```
com.ibm.wps.install.PortalValidationAction, msg2, VersionInfo object: wpsmp  
6.0.0.0  
com.ibm.wps.install.PortalValidationAction, msg2, Created PortalValidation object  
using default rules files  
com.ibm.wps.install.PortalValidationAction, msg2, Detected no versions of WAS  
installed  
com.ibm.wps.install.PortalValidationAction, msg1, Space required for WAS = 1650000  
KB  
com.ibm.wps.install.PortalValidationAction, msg1, Space required for WAS = 1611 MB  
com.ibm.wps.install.PortalValidationAction, msg1, Space required for WPS = 1550000  
KB  
com.ibm.wps.install.PortalValidationAction, msg1, Space required for WPS = 1513 MB  
com.ibm.wps.install.PortalValidationAction, msg1, Space required for IHS = 30000  
KB
```

```
com.ibm.wps.install.PortalValidationAction, msg1, Space required for TEMPWASFP =
600000 KB
com.ibm.wps.install.PortalValidationAction, msg2, useValidation=true
com.ibm.wps.install.PortalValidationAction, msg1, Checking prereqs..
com.ibm.wps.install.PortalValidationAction, msg1, PortalValidation version info
for OS: Windows 2003 SP1
com.ibm.wps.install.PortalValidationAction, msg2, osCurrent: true
```

These lines contain several valuable pieces of information.

- Detects any currently installed versions of WAS.
- Lists space requirements for WAS (in KB and MB).
- Lists space requirements for Process Server (in KB and MB).
- Lists space requirements for IHS and Temp directories.
- Validates the operating system.

The next important set of lines from the wpinstalllog.txt comes from the DiskSpaceValidator component:

```
com.ibm.wps.install.DiskSpaceValidator, msg1, DiskSpaceValidator: Starting check.
Product: WAS, Destination: C:\WebSphere\AppServer
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Checking disk space
for product: WAS, destination: C:\WebSphere\AppServer
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: partition = C:
com.ibm.wps.install.DiskSpaceValidator, msg1, checking writability, parent=C:\
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: destination is
writeable
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: free space available
= 14711930
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: adding partition for
first time
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Space required =
1650000
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Remaining available
space: 13061930
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Success
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Checking temp disk
space for product: TEMPWASFP, destination: C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: partition = C:
com.ibm.wps.install.DiskSpaceValidator, msg1, checking writability,
parent=C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: destination is
writeable
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: free space available
= 14711930
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: re-calc partition
required space: product installed on same partition = WAS
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: re-calc partition
required space = 1650000
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Actual calculated
Free Space available = 13061930
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Pre-recorded Free
Space available = 13061930
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Space required =
600000
```

```
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Remaining available  
space: 12461930  
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Success  
com.ibm.wps.install.DiskSpaceValidator, msg1, DiskSpaceValidator: check successful
```

The DiskSpaceValidator checks the remaining free space on your system and determines if you meet the requirements for the WebSphere Application Server installation. Notice the last two lines:

```
com.ibm.wps.install.DiskSpaceValidator, msg1, checkDiskSpace: Success  
com.ibm.wps.install.DiskSpaceValidator, msg1, DiskSpaceValidator: check successful
```

This will tell you if you successfully met the requirements.

Not all operating systems supported by WebSphere Portal are supported by WebSphere Process Server. Because of that, the next noteworthy line makes this check:

```
com.ibm.wps.install.WbiSupportedOsCondition, msg2, Detected WBI supported OS = true
```

This line tells you whether or not your operating system is supported by Process Server.

Scrolling down further, you will come to the `DetectPortBlockAction` component for the WebSphere Application Server, and it will look similar to the following:

```
com.ibm.wps.install.DetectPortBlockAction, msg1, Detect ports for server1
com.ibm.wps.install.DetectPortBlockAction, msg1, Logging values...
com.ibm.wps.install.DetectPortBlockAction, msg1,      startingPort      = 10000
com.ibm.wps.install.DetectPortBlockAction, msg1,      endingPort        = 10024
com.ibm.wps.install.DetectPortBlockAction, msg1,      endingPortPlusOne = 10025
com.ibm.wps.install.DetectPortBlockAction, msg1,      startingPortToScan = 10000
com.ibm.wps.install.DetectPortBlockAction, msg1,      endingPortToScan   = 65000
com.ibm.wps.install.DetectPortBlockAction, msg1,      portBlockSize     = 25
com.ibm.wps.install.DetectPortBlockAction, msg1, Found port block at: 10000 - 10024
com.ibm.wps.install.DetectPortBlockAction, msg1, Logging values...
com.ibm.wps.install.DetectPortBlockAction, msg1,      startingPort      = 10000
com.ibm.wps.install.DetectPortBlockAction, msg1,      endingPort        = 10024
com.ibm.wps.install.DetectPortBlockAction, msg1,      endingPortPlusOne = 10025
com.ibm.wps.install.DetectPortBlockAction, msg1,      startingPortToScan = 10000
com.ibm.wps.install.DetectPortBlockAction, msg1,      endingPortToScan   = 65000
com.ibm.wps.install.DetectPortBlockAction, msg1,      portBlockSize     = 25
com.ibm.wps.install.DetectPortBlockAction, msg1, profileParam: -startingPort 10000
```

Here you can see what ports are being set aside for use with WebSphere Application Server.

As you continue to scroll down, you will come to the `PortalLocationValidator` section. This section is very important as it reveals several key variables for the Portal installation. That section will look similar to the following:

```
com.ibm.wps.install.PortalLocationValidator, msg2, Checking Portal destination:
C:\WebSphere\PortalServer
com.ibm.wps.install.PortalLocationValidator, msg2, Number of currently/previously
installed Portals:0
com.ibm.wps.install.PortalLocationValidator, msg2, Number of currently/previously
installed Portals:0
com.ibm.wps.install.PortalLocationValidator, msg1, PortalLocationValidator:
Starting check. Product: WPS, Destination: C:\WebSphere\PortalServer
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: Checking disk
space for product: WPS, destination: C:\WebSphere\PortalServer
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: partition = C:
com.ibm.wps.install.PortalLocationValidator, msg1, checking writability,
parent=C:\
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: destination is
writeable
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: free space
available = 14711595
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: re-calc
partition required space: product installed on same partition = TEMPWASFP
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: re-calc
partition required space = 600000
```

```

com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: re-calc
partition required space: product installed on same partition = WAS
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: re-calc
partition required space = 2250000
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: Actual
calculated Free Space available = 12461595
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: Pre-recorded
Free Space available = 12461930
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: Change detected
updating pre-recorded value = 12461595
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: Space required
= 1550000
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: Remaining
available space: 10911595
com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: Success
com.ibm.wps.install.PortalLocationValidator, msg1, PortalLocationValidator:
DiskSpaceValidator: check successful
com.ibm.wps.install.PortalLocationValidator, msg1, PortalLocationValidator: check
successful

```

The following three important checks are made here:

- Checks to see if there are any previously installed instances of Portal.
- Checks installation location to ensure it is valid.
- Checks disk space for location to ensure it meets Portal requirements.

The last three lines tell you if you met all of these requirements:

```

com.ibm.wps.install.PortalLocationValidator, msg1, checkDiskSpace: Success
com.ibm.wps.install.PortalLocationValidator, msg1, PortalLocationValidator:
DiskSpaceValidator: check successful
com.ibm.wps.install.PortalLocationValidator, msg1, PortalLocationValidator: check
successful

```

The installer also runs the DetectPortBlockAction component again. This time for WebSphere Portal. It will resemble the previous example of DetectPortBlockAction.

WebSphere Application Server installation

After validation completes, the actual installation begins. The first step of installation is WebSphere Application Server. In the wpinstalllog.txt, you will see the following:

```

com.ibm.wps.install.ExternalCommandAction, msg2, Beginning install step:
Installing WebSphere Application Server. This step will take several minutes.
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command:
"Y:\WPS60\Windows\W-1\windows\ia32\ifpackage\WAS\Install.exe" -silent -options
"C:\WebSphere\PortalServer\package\was60responsefile.nd.txt" -is:silent
com.ibm.wps.install.ExternalCommandAction, msg2, Working directory:
com.ibm.wps.install.ExternalCommandAction$CancelWatcher, msg2, Created
Cancelwatcher
com.ibm.wps.install.ExternalCommandAction$ProgressWatcherRunner, msg2, Created
ProgressWatcherRunner, ProgressWatcher class:
com.ibm.wps.install.LogProgressWatcher
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, Created StdOut
OutputWatcher
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, Created StdErr
OutputWatcher

```



```
com.ibm.wps.install.ExternalCommandAction, msg2, Return code = 0
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: completed
com.ibm.wps.install.ExternalCommandAction, msg2, Completed install step:
Installing WebSphere Application Server. This step will take several minutes.
com.ibm.wps.install.ScanFileAction, msg1,
fileName=C:\WebSphere\AppServer\logs\log.txt
```

There are several important items to discuss from this section:

- Note that this section begins with the following line:
Beginning install step: Installing WebSphere Application Server.
This section ends with the following line:
Completed install step: Installing WebSphere Application Server.
- The following line shows us exactly how the WAS installer was executed:
Executing command:
"Y:\WPS60\Windows\W-1\windows\ia32\ifpackage\WAS\Install.exe" -silent -options
"C:\WebSphere\PortalServer\package\was60responsefile.nd.txt" -is:silent
- When the installation finishes, we see the following line:
Return code = 0
0 means that no errors were encountered. Anything else here means the installation failed.
- Finally, this line shows us where the installation was logged:
fileName=C:\WebSphere\AppServer\logs\log.txt
In the event of any failures, you would check this log to determine the cause.

WebSphere Process Server installation

After WebSphere Application Server installation completes, the installer moves on to install WebSphere Process Server. Scrolling down the wpinstalllog.txt, you can see that with the following lines:

```
com.ibm.wps.install.ExternalCommandAction, msg2, Beginning install step:
Installing WebSphere Process Server. This step will take several minutes.
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command:
"Y:\WPS60\Windows\W-2\windows\ia32\WBI\Install.exe" -silent -options
"C:\WebSphere\PortalServer\package\wps60.bpc.responsefile.txt" -W
unsupportedJDKFoundSeq.active=false -is:silent
com.ibm.wps.install.ExternalCommandAction, msg2, Working directory:
"Y:\WPS60\Windows\W-2\windows\ia32\WBI"
com.ibm.wps.install.ExternalCommandAction$CancelWatcher, msg2, Created
Cancelwatcher
com.ibm.wps.install.ExternalCommandAction$ProgressWatcherRunner, msg2, Created
ProgressWatcherRunner, ProgressWatcher class:
com.ibm.wps.install.LogProgressWatcher
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, Created StdOut
OutputWatcher
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, Created StdErr
OutputWatcher
com.ibm.wps.install.ExternalCommandAction, msg2, Return code = 0
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: completed
com.ibm.wps.install.ExternalCommandAction, msg2, Completed install step:
Installing WebSphere Process Server. This step will take several minutes.
```

```
com.ibm.wps.install.ScanFileAction, msg1,  
fileName=C:\WebSphere\AppServer\logs\wbi\log.txt
```

You can make the same observations from here as we did for the WAS installation. You can see the following:

- ▶ Where the installation begins and ends.
- ▶ The exact command executed to install WebSphere Process Server.
- ▶ The return code.
- ▶ The file where the installation was logged.

WebSphere Application Server fixes installation

The next section of the installation installs the required WebSphere Application Server interim fixes. That section will look similar to the following:

```
com.ibm.wps.install.ExternalCommandAction, msg2, Beginning install step:  
Installing WebSphere Application Server Fixes  
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: cmd /c  
""C:\WebSphere\PortalServer\package\was6fixes.bat""  
com.ibm.wps.install.ExternalCommandAction, msg2, Working directory:  
com.ibm.wps.install.ExternalCommandAction$CancelWatcher, msg2, Created  
Cancelwatcher  
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, Created StdOut  
OutputWatcher  
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, Created StdErr  
OutputWatcher  
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut:  
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut:  
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\wplaunch>"C:\WebSphere\AppServer\updateinstall  
er\update" -is:javahome "C:\WebSphere\AppServer\updateinstaller\java" -W  
maintenance.package="C:\WebSphere\AppServer\updateinstaller\maintenance\6.0.1.0-WS  
-WBI-IFIY82199.pak" -silent  
com.ibm.wps.install.ExternalCommandAction, msg2, Return code = 0  
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: completed  
com.ibm.wps.install.ExternalCommandAction, msg2, Completed install step:  
Installing WebSphere Application Server Fixes  
com.ibm.wps.install.ScanFileAction, msg1,  
fileName=C:\WebSphere\AppServer\logs\update\6.0.1.0-WS-WBI-IFIY82199.install\updat  
ellog.txt
```

Like the previous two sections, we can make several important observations here. You can see the following:

- ▶ Where the installation begins and ends.
- ▶ The exact command being used to install the fixes.
- ▶ The return code.
- ▶ The file where the installation is logged.

Profile creation

After WebSphere Application Server and WebSphere Process Server are installed and upgraded, the next section you come to in wpinstalllog.txt is profile creation:

```
com.ibm.wps.install.LogProgressRendererSwingImpl, msg1, Log file is:  
$LONGPATH($N($W(was.location)/logs/wasprofile/wasprofile_create_$W(detectProfileAc  
tion.profileName).log))
```

```

com.ibm.wps.install.ExternalCommandAction, msg2, Beginning install step:
Installing WebSphere Application Server Profile. This step will take several
minutes.
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command:
"C:\WebSphere\AppServer\bin\manageprofiles.bat" -create -profileName "wp_profile"
-profilePath "C:\ibm\WebSphere\profiles\wp_profile" -templatePath
"C:\WebSphere\AppServer\profileTemplates\default.bfm" -federateLater "true"
-nodeName "nodename" -cellName "cellname" -hostName "hostname.ibm.com"
-startingPort 10000 -winerviceCheck false -dbJDBCClasspath "null" -dbType "null"
-ceiSampleJmsUser "wpsadmin" -ceiSampleJmsPwd "PASSWORD_REMOVED"
-ceiSampleServerName "server1" -ceiDbProduct "CLOUDSCAPE_V51_1" -ceiDbName ""
-ceiDbUser "wpsadmin" -ceiDbPwd "PASSWORD_REMOVED" -ceiDbSysUser "wpsadmin"
-ceiDbSysPwd "PASSWORD_REMOVED" -ceiDbJdbcDriverClasspath "" -ceiDbJdbcDriverType
"" -ceiDbServerName "" -ceiDbServerPort "" -ceiDbNodeName "" -ceiDbExecuteScripts
"" -configureScaSecurity "true" -scaSecurityUserId "wasadmin" -scaSecurityPassword
"PASSWORD_REMOVED"
com.ibm.wps.install.ExternalCommandAction, msg2, Working directory:
com.ibm.wps.install.ExternalCommandAction$CancelWatcher, msg2, Created
CancelWatcher
com.ibm.wps.install.ExternalCommandAction$ProgressWatcherRunner, msg2, Created
ProgressWatcherRunner, ProgressWatcher class:
com.ibm.wps.install.LogProgressWatcher
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, Created StdOut
OutputWatcher
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, Created StdErr
OutputWatcher
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut:
INSTCONFSUCCESS: Success: The profile now exists.
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut:
INSTCONFSUCCESS: Profile augmentation succeeded.
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut:
INSTCONFSUCCESS: Profile augmentation succeeded.
com.ibm.wps.install.ExternalCommandAction, msg2, Return code = 0
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: completed
com.ibm.wps.install.ExternalCommandAction, msg2, Completed install step:
Installing WebSphere Application Server Profile. This step will take several
minutes.

```

This section is slightly different from the previous sections. You can still see markers for the beginning and the end of the profile creation. You can still see the command being executed. You can still see the return code returned from the profile creation. However, the log file line comes at the beginning rather than at the end.

```

com.ibm.wps.install.LogProgressRendererSwingImpl, msg1, Log file is:
$LONGPATH($N($W(was.location)/logs/wasprofile/wasprofile_create_$W(detectProfileAc
tion.profileName).log))

```

If there are any problems with the profile creation, you need to check that log.

WebSphere Portal installation

The WebSphere Portal installation and configuration begins and continues from this point on. There are numerous logging statements written here. Most of these statements are only for information purposes and are not relevant for troubleshooting and debugging. They are outside the scope of this document. A majority of the WebSphere Portal installation and configuration debugging statements are written to LocalizerTrace.archive5.log.

Enabling WebSphere Application Server security

If you continue to scroll down wpinstalllog.txt you will come to the next section for enabling security. This section is headed with the following lines:

```
com.ibm.wps.install.ExternalCommandAction, msg2, Beginning install step: Enabling WebSphere Application Server Security
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: cmd /c
"C:\WebSphere\PortalServer\config\WPSconfig.bat" enable-security-wmmur-db-initial
-DPortalAdminPwd=PASSWORD_REMOVED -DWasPassword=PASSWORD_REMOVED
-DLTPAPassword=PASSWORD_REMOVED -DskipWTP=true"
```

Notice the command that is executed. It is a WPSconfig command executing the task enable-security-wmmur-db-initial. Because this is a WPSconfig command, the information from this security task is logged to ConfigTrace.log in the portal_root/log directory. Knowing this, in the event of any failures here, you can check this log for valuable information.

From this point, if you continue to scroll down the wpinstalllog.txt, you will see that the output of the task is also written here in wpinstalllog.txt and continues for several lines. Eventually, you will come to lines similar to the following:

```
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut: BUILD SUCCESSFUL
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut: Total time: 12 minutes 36 seconds
com.ibm.wps.install.ExternalCommandAction, msg2, Return code = 0
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: completed
com.ibm.wps.install.ExternalCommandAction, msg2, Completed install step: Enabling WebSphere Application Server Security
com.ibm.wps.install.ScanFileAction, msg1,
fileName=C:\WebSphere\PortalServer\log\ConfigTrace.log
```

Here you can note the following important items:

- ▶ **BUILD SUCCESSFUL** indicates that the configuration task was successful and that no errors were encountered.
- ▶ **Return code = 0** also indicates no errors were encountered.
- ▶ **fileName=C:\WebSphere\PortalServer\log\ConfigTrace.log** shows us where this information was logged to in the event of an error.

Business Process Choreographer configured

The next section is another WPSconfig task: setup-bpe. The first couple of lines from this section look similar to the following:

```
com.ibm.wps.install.ExternalCommandAction, msg2, Beginning install step: Running setup-bpe
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: cmd /c
"C:\WebSphere\PortalServer\config\WPSconfig.bat" setup-bpe
-DPortalAdminPwd=PASSWORD_REMOVED -DWasPassword=PASSWORD_REMOVED
-Dbpe.jmsBFMRUNAsPwd=PASSWORD_REMOVED -Dbpe.jmsHTMRUNAsPwd=PASSWORD_REMOVED
-Dbpe.mqPwd=PASSWORD_REMOVED -DskipWTP=true"
```

If you continue to scroll past this in the wpinstalllog.txt, you will see that the output of the task is written here. Eventually you will come to the following:

```
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut: BUILD SUCCESSFUL
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut: Total time: 10 minutes 20 seconds
```

```
com.ibm.wps.install.ExternalCommandAction, msg2, Return code = 0
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: completed
com.ibm.wps.install.ExternalCommandAction, msg2, Completed install step: Running
setup-bpe
com.ibm.wps.install.ScanFileAction, msg1,
fileName=C:\WebSphere\PortalServer\log\ConfigTrace.log
```

Again we see BUILD SUCCESSFUL, the return code, and the location of the log.

Stop and start WebSphere Portal server

The final two actions in the WebSphere Portal installation are to stop and start the WebSphere Portal server, using WPSconfig again. First, you will come to the stop server, as follows:

```
com.ibm.wps.install.ExternalCommandAction, msg2, Beginning install step: Stopping
WebSphere Application Server
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: cmd /c
""C:\WebSphere\PortalServer/config/WPSconfig.bat" stop-admin-server"
```

You can make the same notes here as before. WPSconfig is executing the task stop-admin-server. Scrolling forward once again the contents are written to the wpinstallog.txt. Eventually you come to the following:

```
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut: BUILD
SUCCESSFUL
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut: Total time:
43 seconds
com.ibm.wps.install.ExternalCommandAction, msg2, Return code = 0
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: completed
com.ibm.wps.install.ExternalCommandAction, msg2, Completed install step: Stopping
WebSphere Application Server
```

You should see BUILD SUCCESSFUL here. Immediately after this, you will see the following:

```
com.ibm.wps.install.ExternalCommandAction, msg2, Beginning install step: Starting
WebSphere Portal
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: cmd /c
""C:\WebSphere\PortalServer/config/WPSconfig.bat" start-portal-server"
```

The WPSconfig tool executes the start-portal-server task. The output is printed out to the wpinstallog.txt, and if you scroll down you will eventually come to the following:

```
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut: BUILD
SUCCESSFUL
com.ibm.wps.install.ExternalCommandAction$OutputWatcher, msg2, StdOut: Total time:
12 minutes 6 seconds
com.ibm.wps.install.ExternalCommandAction, msg2, Return code = 0
com.ibm.wps.install.ExternalCommandAction, msg2, Executing command: completed
com.ibm.wps.install.ExternalCommandAction, msg2, Completed install step: Starting
WebSphere Portal
```

You should see BUILD SUCCESSFUL here to indicate the server startup completed with no errors.

Conclusion

By understanding the contents of the wpinstallog.txt, you should be more familiar with the installation process and the order of events that take place during WebSphere Portal

installation. Knowledge and understanding of this log will increase your self-sufficiency and might avoid the need for creating a PMR with IBM support.

Analyze failures with Portal installation

You have launched the installation of WebSphere Portal. During installation, the following message appears:

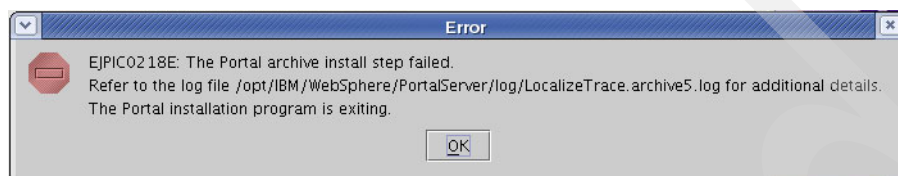


Figure 8-1 Installation failure

How do you begin problem determination?

Step 1

The error message tells you where you need to begin troubleshooting. Review the LocalizeTrace.archive5.log file. When opening the LocalizeTrace.archive5.log file, you should scroll to the bottom of the file to see where the failure occurred. At the bottom of the log you may see something similar to the following:

```
BUILD FAILED
file:../config/includes/fixup_was_cfg.xml:177: Java returned: 105
Total time: 2 minutes 37 seconds
```

This is very similar to what you may find in a ConfigTrace.log as discussed in section 9.1.1. There are a few important items to note, however:

1. The **BUILD FAILED** indicates that the task running during installation has not completed successfully. In most cases, this will require an uninstall and then reinstall. You should stop here, determine the cause of failure, and correct before moving on.
2. The line **file:../config/includes/fixup_was_cfg.xml:177: Java returned: 105** usually does NOT contain any initial meaningful information about the failure, but does contain valuable information needed for deeper problem determination if required.
3. Unlike the ConfigTrace.log file, the LocalizerTrace.archive5.log does not include a properties dump.

At this point you should be at the bottom of the LocalizerTrace.archive5.log and looking at the **BUILD FAILED** messages as noted above. To continue problem determination, you should now scroll up the LocalizerTrace.archive5.log until you see where the specific problem occurred. For example,

```
[wsadmin] Found ServerTemplate defaultProcessServer
[wsadmin] Attrs-----
[wsadmin] -name WebSphere_Portal -genUniquePorts -templateName
defaultProcessServer
[wsadmin] WASX7017E: Exception received while running file
"/opt/IBM/WebSphere/PortalServer/config/gather/was/fixup_was_cfg.jacl";
exception information:
com.ibm.websphere.management.cmdframework.CommandValidationException:
ADMG0253E:Matching template defaultProcessServer could not be found.

BUILD FAILED
file:../config/includes/fixup_was_cfg.xml:177: Java returned: 105
```

If you scroll up a bit further you can see the sub-task in which the error occurred.

```
action-consolidated-was-cfg:
```

```
Tue Oct 31 20:12:30 EST 2006
```

```
[wsadmin] WASX7357I: By request, this scripting client is not connected to
any server process. Certain configuration and application operations will be
available in local mode.
```

```
[wsadmin] WASX7303I: The following options are passed to the scripting
environment and are available as argument that is stored in the argv variable:
"/opt/IBM/WebSphere/PortalServer"
```

Here we can get specific details about the error.

1. Notice the action-consolidated-was-cfg. This tells you the name of the sub-task that failed.
2. The lines under action-consolidated-was-cfg shows us what actions were being attempted by the sub-task. In this case, you can see that a wsadmin command is being attempted.

Step 2

After identifying the relevant error, begin problem analysis.

1. The LocalizeTrace.archive5.log shows this specific error:

```
[wsadmin] WASX7017E: Exception received while running file
"/opt/IBM/WebSphere/PortalServer/config/gather/was/fixup_was_cfg.jacl";
exception information:
com.ibm.websphere.management.cmdframework.CommandValidationException:
ADMG0253E:Matching template defaultProcessServer could not be found.
```

Here you can see that wsadmin is attempting to execute a jacl file and it fails with the following error:

```
ADMG0253E:Matching template defaultProcessServer could not be found.
```

2. The wsadmin tool generates its own log called wsadmin.traceout. This log can be found in the *was_profile_root/log* directory. Knowing this, you may find additional information about the failure in that log. For example, if you examine the wsadmin.traceout for the previous failure, you will find the following:

```
[10/31/06 20:13:03:619 EST] 0000000a AbstractShell A WASX7091I: Executing
script: "/opt/IBM/WebSphere/PortalServer/config/gather/was/fixup_was_cfg.jacl"
[10/31/06 20:13:14:281 EST] 0000000a AbstractShell E WASX7120E: Diagnostic
information from exception with text
"com.ibm.websphere.management.cmdframework.CommandValidationException:
ADMG0253E:Matching template defaultProcessServer could not be found.
" follows:
com.ibm.websphere.management.cmdframework.CommandValidationException:
ADMG0253E:Matching template defaultProcessServer could not be found.
    at
com.ibm.wsspi.management.commands.server.CreateServer.validate(CreateServer.jav
a:352)
    at
com.ibm.ws.management.commands.server.CreateApplicationServer.validate(CreateAp
plicationServer.java:100)
    at
com.ibm.websphere.management.cmdframework.provider.AbstractTaskCommand.execute(
AbstractTaskCommand.java:526)
```



```
at  
com.ibm.ws.scripting.adminCommand.AdminCmdController.executeCmd(AdminCmdControl  
ler.java:1209)
```

Unfortunately in this example, the only new information you get from the wsadmin.traceout is a stacktrace associated with the error.

From this, you can determine that the error in the wsadmin.traceout is the cause of the failure. Specifically the **ADMG0253E** error code.

Step 3

After determining the root cause you can begin searching for a solution to the ADMG0253E error code using the IBM Support Assistant (ISA).

<http://www-306.ibm.com/software/support/isa/>

Conclusion

By following these approaches, you should be able to determine the failure for many of the error messages when installing IBM WebSphere Portal. By implementing this troubleshooting approach, you should be able to increase your self-sufficiency and avoid the need for creating a PMR with IBM support. If the suggestions do not provide the relief you need however, do contact IBM support and provide the information that is required by using ISA to collect your logs. The instructions to collect logs using ISA are included in section 9.1.6.

8.1.3 Portal upgrade failure

The following sections provides information about Portal upgrade failure.

Definition of a Portal upgrade failure

WebSphere Portal uses an update installer to apply maintenance fixes or fixpacks to the Portal environment. Failures during upgrades are common. This section helps you understand how the upgrade works and how you can troubleshoot an upgrade failure.

A WebSphere Portal interim fix upgrade consists of a single JAR file (for example, *PK32501.jar*). An interim fix is meant to update a specific component of Portal and usually does not require any additional configuration.

A Portal fixpack upgrade consists of a single JAR file as well (for example, *WP_PTF_6001.jar*). A fixpack is meant to update several components and features of Portal and requires additional configuration once the components are upgraded. Failures during upgrades commonly occur during this additional configuration.

With fixpack upgrades in WebSphere Portal 5.1.0.x, you were required to first run the Portal Update Installer (PUI) to update the Portal components and then manually run a configuration task to configure Portal with these newly updated components. In WebSphere Portal 6.0.x, the PUI was enhanced to take care of both the component upgrade and the additional configuration automatically for you.

The relevant logs for the PUI are located in the *portal_root/version/log* directory.

Analyze failures with the Portal Update Installer

You attempt to upgrade WebSphere Portal to version 6.0.0.1 using the Portal Update Installer GUI. The upgrade fails with the following message:

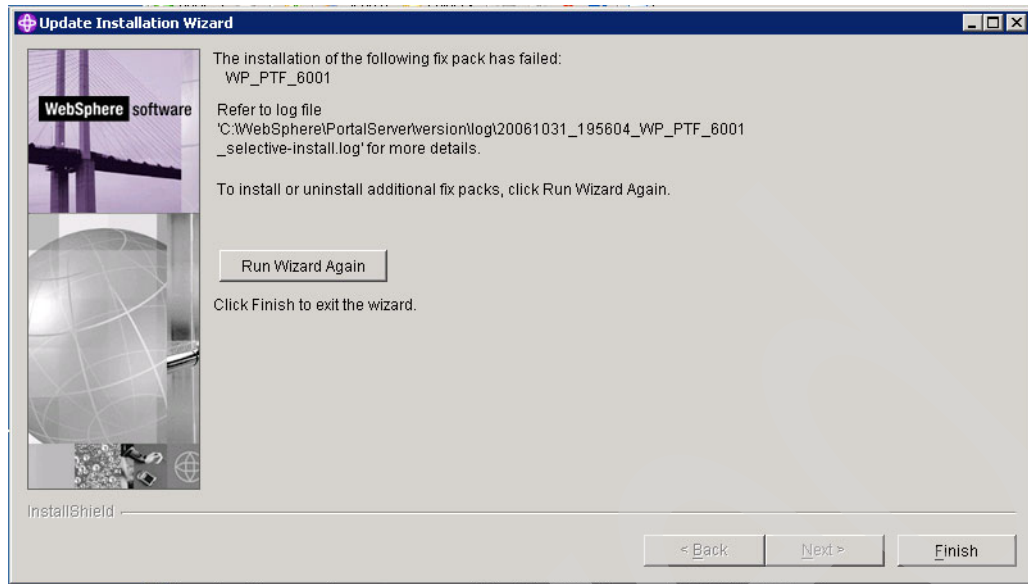


Figure 8-2 Upgrade failure

How do you begin problem determination?

Step 1

From the error message, you are referred to a log:

`portal_root/version/log/20061031_195604_WP_PTF_6001_selective-install.log`.

Since the name of the log is not the same across all environments, this log is referred to as `<timestamp>_WP_PTF_6001_selective-install.log`.

Review the `<timestamp>_WP_PTF_6001_selective-install.log` to determine the cause of the failure. After you open the log, you should scroll to the bottom of the file and start from there to find the error message. At the bottom you may see something similar to the following:

PTF Results:

```
=====
Time Stamp (End)      : 2006-10-31T14:15:43-05:00
PTF Result            : failed
PTF Result Message:
=====
```

WUPD0218E: Fix pack installation failure: The component wp.ptf.config failed to install.

```
=====
Notification: Completing fix pack 'WP_PTF_6001'
```

In this case, the `wp.ptf.config` component failed to install. If you scroll up a little further, you will see more information about the failure:

```
WUPD0248E: Fix pack update failure: The processing of fix pack WP_PTF_6001,
component wp.ptf.config failed. See the log file
C:\WebSphere\PortalServer\version\log\20061031_184235_WP_PTF_6001_wp.ptf.config
_install.log for processing details.
```

This line points you to a log file for the `wp.ptf.config` component:

`<timestamp>_WP_PTF_6001_wp.ptf.config_install.log`. When you open this log file, you should once again scroll to bottom, where you can see the following:

2006-10-31T15:27:25-05:00 Result: [wsadmin] WASX7017E: Exception received while running file "C:\WEBSPH~1\PORTAL~1/config/was/wp_SetTargetMappings.jacl"; exception information:
com.ibm.websphere.management.exception.ConfigServiceException
2006-10-31T15:27:25-05:00 Result: [wsadmin]
com.ibm.ws.sm.workspace.WorkSpaceException
2006-10-31T15:27:25-05:00 Result: [wsadmin]
com.ibm.ws.sm.workspace.WorkSpaceException
2006-10-31T15:27:25-05:00 Result: [wsadmin] java.io.IOException:
java.io.IOException: The system cannot find the specified file, either the filename is too long on Windows system or run out of file descriptor on UNIX platform. java.io.FileNotFoundException:
C:\ibm\WebSphere\profiles\wp_profile\wstemp\Script10ea00c683a\workspace\cells\nodename\applications\BPEContainer_nodename\WebSphere_Portal.ear\deployments\BPEContainer_nodename\WebSphere_Portal\compensate_ejb.jar\META-INF\ibm-ejb-jar-ext-pme.xml.workspace_save (The system cannot find the file specified)
2006-10-31T15:27:25-05:00 Result:
2006-10-31T15:27:25-05:00 Result: [wsadmin] WASX7309W: No "save" was performed before the script
"C:\WEBSPH~1\PORTAL~1/config/was/wp_SetTargetMappings.jacl" exited;
configuration changes will not be saved.
2006-10-31T15:27:25-05:00 Result: StdErr:
2006-10-31T15:27:25-05:00 Result: StdErr: BUILD FAILED
2006-10-31T15:27:25-05:00 Result: StdErr:
file:../config/includes/wp_ptf_6001.xml:1646: Java returned: 105
2006-10-31T15:27:25-05:00 Result: StdErr:
2006-10-31T15:27:25-05:00 Result: StdErr: Total time: 30 minutes 45 seconds
2006-10-31T15:27:25-05:00 Error 112 -- Return code (1) differs from the expected code (0).
2006-10-31T15:27:25-05:00 Validating platform for (Linux, Solaris, SunOS, AIX, HP-UX, s390)
2006-10-31T15:27:25-05:00 platform check is negative.
2006-10-31T15:27:25-05:00 Validating platform for (OS/400)
2006-10-31T15:27:25-05:00 platform check is negative.
2006-10-31T15:27:25-05:00 Product File Update is not active; skipping update step.
2006-10-31T15:27:25-05:00 Input Jar File :
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\1\ptfs\WP_PTF_6001\components\wp.ptf.config\update.jar
2006-10-31T15:27:25-05:00 Target Directory : C:\WebSphere\PortalServer
2006-10-31T15:27:25-05:00 Backup Jar File :
C:\WebSphere\PortalServer\version\backup\20061031_195612_WP_PTF_6001_wp.ptf.config_undo.jar
2006-10-31T15:27:25-05:00 Warnings Issued : 0
2006-10-31T15:27:25-05:00 Log File :
C:\WebSphere\PortalServer\version\log\20061031_195612_WP_PTF_6001_wp.ptf.config_install.log
2006-10-31T15:27:25-05:00
2006-10-31T15:27:25-05:00 Errors were noted: 1
2006-10-31T15:27:25-05:00 Extractor functionality may be compromised!

Starting from the bottom of this file, you move up and search for the error message. You then see the following set of lines:

```
2006-10-31T15:27:25-05:00 Result: StdErr: BUILD FAILED
2006-10-31T15:27:25-05:00 Result: StdErr:
file:../config/includes/wp_ptf_6001.xml:1646: Java returned: 105
2006-10-31T15:27:25-05:00 Result: StdErr:
2006-10-31T15:27:25-05:00 Result: StdErr: Total time: 30 minutes 45 seconds
2006-10-31T15:27:25-05:00 Error 112 -- Return code (1) differs from the
expected code (0).
```

The **BUILD FAILED** message is very similar to what you may see from a Configuration Task failure. Scrolling up a bit further reveals to us why you received a **BUILD FAILED** message:

```
2006-10-31T15:27:25-05:00 Result: [wsadmin] java.io.IOException:
java.io.IOException: The system cannot find the specified file, either the
filename is too long on Windows system or run out of file descriptor on UNIX
platform. java.io.FileNotFoundException:
C:\ibm\WebSphere\profiles\wp_profile\wstemp\Script10ea00c683a\workspace\cells\n
odename\applications\BPEContainer_nodename_WebSphere_Portal.ear\deployments\BPE
Container_nodename_WebSphere_Portal\compensate_ejb.jar\META-INF\ibm-ejb-jar-ext
-pme.xmi.workspace_save (The system cannot find the file specified)
```

Step 2

After identifying the relevant error, begin problem analysis.

1. The `<timestamp>_WP_PTF_6001_wp.ptf.config_install.log` shows the specific error:

```
2006-10-31T15:27:25-05:00 Result: [wsadmin] java.io.IOException:
java.io.IOException: The system cannot find the specified file, either the
filename is too long on Windows system or run out of file descriptor on UNIX
platform. java.io.FileNotFoundException:
C:\ibm\WebSphere\profiles\wp_profile\wstemp\Script10ea00c683a\workspace\cells\n
odename\applications\BPEContainer_nodename_WebSphere_Portal.ear\deployments\BPE
Container_nodename_WebSphere_Portal\compensate_ejb.jar\META-INF\ibm-ejb-jar-ext
-pme.xmi.workspace_save (The system cannot find the file specified)
```

There are a few items to note from this error:

- The 2006-10-31T15:27:25 tells us the time this error occurred.
 - The [wsadmin] tells you that the update installer is using the wsadmin tool.
 - The java.io.FileNotFoundException line tells you the type of error you received.
 - There is no error code associated with this error.
2. The wsadmin tool generates its own log called wsadmin.traceout. This log file can be found in the `wsas_profile_root\log` directory. Knowing this, you may find additional information about the failure in the wsadmin.traceout. Open this log and scroll to the bottom. In the example above, you will find the following error:

Note: In the following error message, the stacktrace associated with the error message was omitted to save space.

```
[10/31/06 15:27:09:688 EST] 0000000a AbstractShell E WASX7120E: Diagnostic
information from exception with text
"com.ibm.websphere.management.exception.ConfigServiceException
```

```
com.ibm.ws.sm.workspace.WorkSpaceException
com.ibm.ws.sm.workspace.WorkSpaceException
java.io.IOException: java.io.IOException: The system cannot find the specified
file, either the filename is too long on Windows system or run out of file
descriptor on UNIX platform. java.io.FileNotFoundException:
C:\ibm\WebSphere\profiles\wp_profile\wstemp\Script10ea00c683a\workspace\cells\n
odename\applications\BPEContainer_nodename_WebSphere_Portal.ear\deployments\BPE
Container_nodename_WebSphere_Portal\compensate_ejb.jar\META-INF\ibm-ejb-jar-ext
-pme.xml.workspace_save (The system cannot find the file specified)
" follows:
```

```
com.ibm.websphere.management.exception.ConfigServiceException: WKSP0008E
RepositoryException while checking the state of
cells/nodename/applications/BPEContainer_nodename_WebSphere_Portal.ear/deployme
nts/BPEContainer_nodename_WebSphere_Portal/compensate_ejb.jar/META-INF/ibm-ejb-
jar-ext-pme.xml in the master repository
--com.ibm.ws.sm.workspace.WorkSpaceException: WKSP0016E Error get digest for
cells/nodename/applications/BPEContainer_nodename_WebSphere_Portal.ear/deployme
nts/BPEContainer_nodename_WebSphere_Portal/compensate_ejb.jar/META-INF/ibm-ejb-
jar-ext-pme.xml.workspace_save --java.io.IOException: The system cannot find
the specified file, either the filename is too long on Windows system or run
out of file descriptor on UNIX platform. java.io.FileNotFoundException:
C:\ibm\WebSphere\profiles\wp_profile\wstemp\Script10ea00c683a\workspace\cells\n
odename\applications\BPEContainer_nodename_WebSphere_Portal.ear\deployments\BPE
Container_nodename_WebSphere_Portal\compensate_ejb.jar\META-INF\ibm-ejb-jar-ext
-pme.xml.workspace_save (The system cannot find the file specified)
```

Caused by: com.ibm.ws.sm.workspace.WorkSpaceException: WKSP0008E
RepositoryException while checking the state of
cells/nodename/applications/BPEContainer_nodename_WebSphere_Portal.ear/deployme
nts/BPEContainer_nodename_WebSphere_Portal/compensate_ejb.jar/META-INF/ibm-ejb-
jar-ext-pme.xml in the master repository
--com.ibm.ws.sm.workspace.WorkSpaceException: WKSP0016E Error get digest for
cells/nodename/applications/BPEContainer_nodename_WebSphere_Portal.ear/deployme
nts/BPEContainer_nodename_WebSphere_Portal/compensate_ejb.jar/META-INF/ibm-ejb-
jar-ext-pme.xml.workspace_save --java.io.IOException: The system cannot find
the specified file, either the filename is too long on Windows system or run
out of file descriptor on UNIX platform. java.io.FileNotFoundException:
C:\ibm\WebSphere\profiles\wp_profile\wstemp\Script10ea00c683a\workspace\cells\n
odename\applications\BPEContainer_nodename_WebSphere_Portal.ear\deployments\BPE
Container_nodename_WebSphere_Portal\compensate_ejb.jar\META-INF\ibm-ejb-jar-ext
-pme.xml.workspace_save (The system cannot find the file specified)

There are several key items to note about this error:

- Notice the time stamp from the wsadmin traceout. This is at the same time from the error in the <timestamp>_WP_PTF_6001_wp.ptf.config_install.log
- The same error you saw earlier appears here as well:

```
java.io.IOException: The system cannot find the specified file, either the
filename is too long on Windows system or run out of file descriptor on UNIX
platform. java.io.FileNotFoundException:
C:\ibm\WebSphere\profiles\wp_profile\wstemp\Script10ea00c683a\workspace\cell
s\nodename\applications\BPEContainer_nodename_WebSphere_Portal.ear\deploymen
ts\BPEContainer_nodename_WebSphere_Portal\compensate_ejb.jar\META-INF\ibm-ej
b-jar-ext-pme.xml.workspace_save (The system cannot find the file specified)
```

- Scrolling down the error further reveals the cause and provides error codes:

```
Caused by: com.ibm.ws.sm.workspace.WorkSpaceException: WKSP0008E
RepositoryException while checking the state of
cells/nodename/applications/BPEContainer_nodename_WebSphere_Portal.ear/deplo
yments/BPEContainer_nodename_WebSphere_Portal/compensate_ejb.jar/META-INF/ib
m-ejb-jar-ext-pme.xmi in the master repository
--com.ibm.ws.sm.workspace.WorkSpaceException: WKSP0016E Error get digest for
cells/nodename/applications/BPEContainer_nodename_WebSphere_Portal.ear/deplo
yments/BPEContainer_nodename_WebSphere_Portal/compensate_ejb.jar/META-INF/ib
m-ejb-jar-ext-pme.xmi.workspace_save --java.io.IOException: The system
cannot find the specified file, either the filename is too long on Windows
system or run out of file descriptor on UNIX platform.
java.io.FileNotFoundException:
C:\ibm\WebSphere\profiles\wp_profile\wstemp\Script10ea00c683a\workspace\cell
s\nodename\applications\BPEContainer_nodename_WebSphere_Portal.ear\deploymen
ts\BPEContainer_nodename_WebSphere_Portal\compensate_ejb.jar\META-INF\ibm-ej
b-jar-ext-pme.xmi.workspace_save (The system cannot find the file specified)
```

These errors are the cause of the upgrade failure. Specifically the **WKSP0008E** and **WKSP0016E** error codes.

Step 3

After determining the root cause you can begin searching for a solution to the **WKSP0008E** and **WKSP0016E** error code using the IBM Support Assistant (ISA).

<http://www-306.ibm.com/software/support/isa/>

Conclusion

By following these approaches, you should be able to determine how to troubleshoot any errors encountered when upgrading IBM WebSphere Portal. By implementing this troubleshooting approach, you should be able to increase your self-sufficiency and might avoid the need for creating a PMR with IBM support. If the suggestions do not provide the relief you need however, do contact IBM support and provide the information that is required by using ISA to collect your logs.

8.1.4 Portal JVM crash

The following sections provides troubleshooting information for Portal JVM crashes.

Definition of a WebSphere Portal crash

WebSphere Portal is considered to have crashed if its process identifier (PID) disappeared or its PID changed over time without an intentional stopping of the server, such as the **stopServer** command being issued.

To obtain what the current PID is for your WebSphere Portal's application server, you can look at the file `<server>.PID` under

`<WSAS_install_root>/logs/<server>` (for WebSphere Application Server V5.1)

or

`<WSAS_install_root>/profiles/<profile>/logs/<server>` (for WebSphere Application Server V6.0).

Note: If the application server's PID is still running, you are not experiencing a crash but rather a hang, which is beyond the scope of this section.

The following diagnostic logs and files are essential to troubleshooting a crash:

1. Javacore (also known as a thread dump). This file is generated when the Java Virtual Machine (JVM) terminates unexpectedly. It is a text file that contains information about the JVM and Java application captured at some point during execution. For example, it would contain information about the operating system, thread, native stack, lock, and memory. As a note, a javacore can also be generated by sending a specific signal command to the JVM.

You can look for the location for javacore within the file `native_stderr.log`.

The location could be in any of the following:

- The location specified by the `IBM_JAVACOREDIRENvironment` variable if it is set.
 - `<WSAS_install_root>` (for WebSphere Application Server V5.1)
or `<WSAS_install_root>/profiles/<profile>` (for WebSphere Application Server V6.0)
 - The location that is specified by the `TMPDIR` environment variable, if set.
 - The `/tmp` directory or on Microsoft Windows the location that is specified by the `TEMP` environment variable, if configured.
2. Core Dump or User Dump. This file contains a complete dump of your computer's memory, and therefore, it can grow large.

Analyze a Portal crash

You were using Portal when the JVM suddenly crashed. How do you begin problem determination?

Step 1

Search for a javacore at the locations documented above. If there is a javacore present, proceed to Step 2. If a javacore is not generated, you may need to upgrade your IBM Java SDK with the latest service release because the JVM is probably hung.

If no javacore is present, you need to determine if a core file is generated. The core file can be used to investigate crash related issues. As a note, if there were no javacore or core file available, make sure a **stopServer** command was not issued. The nodeagent can also do a stop without showing anything in the portal logs.

Step 2

To analyze the javacore that was generated with the crash, look into the `TITLE` within the file to see what signal triggered the creation of javacore. The signals 1, 0, `OUTOFMEMORY`, and `SIGNONE` are usually related to memory issues.

A signal 11 (`SIGSEGV`) indicates an application server crash. A signal 3 indicates that it was a user-generated javacore.

The javacore signal appears in this section:

```
NULL -----
OSECTION    TITLE subcomponent dump routine
NULL =====
1TISIGINFO   OUTOFMEMORY received
1TIDATETIME  Date:                2006/07/25 at 10:01:56
1TIFILENAME  Javacore filename:    /data/work/javacore450680.1153839716.txt
```



```

NULL -----
or
NULL -----
OSECTION      TITLE subcomponent dump routine
NULL          =====
1TISIGINFO    signal 11 received
1TIDATETIME   Date:                2006/07/28 at 11:02:35
1TIFILENAME   Javacore filename:   /coredump/javacore336050.1154077355.txt
NULL          -----

```

Step 3

Check the Java SDK version from the 1XHFULLVERSION within the javacore. As a note, sometimes it may show up as 1CJAVAVERSION depending on what version of WebSphere Application Server you are using, for example:

```
1XHFULLVERSION J2RE 1.4.2 IBM AIX build ca1420-20040626
```

or

```
1CIJAVAVERSION J2RE 1.4.2 IBM Windows 32 build cn142-20041202
```

Step 4

Look at the cause of the crash from the line beginning with 1XHSIGRECV. This line shows the library involved in the crash. As a note, sometimes it will show *Fault Modules* within the javacore.

If the *Fault Module* or 1XHSIGRECV shows the following, then upgrade the IBM Java SDK as discussed in Step 1 (above) to resolve this problem:

- ▶ The JVM Modules
 - Windows: JVM.dll
 - AIX: libjvm.a
 - Linux: libjvm.so
- ▶ The Just In Time (JIT) Modules
 - Windows: JITC.dll
 - AIX: libjitc.a
 - Linux: libjitc.so

The 1XHSIGRECV and *Fault Module* entry will look similar to the following:

```

OSECTION      XHPI subcomponent dump routine
NULL          =====
1XHTIME       Fri Jul 28 11:02:35 2006
1XHSIGRECV    SIGSEGV received at 0xd15e58f0 in
<WSAS_root>/java/jre/bin/libjitc.a. Processing terminated.

```

or

```

OSECTION XHPI subcomponent dump routine
NULL =====
1XHEXCPCODE Exception code: C0000005 Access Violation
1XHEXCADDRESS Fault address: 100D15B0 01:000D05B0
1XHEXCPMODULE Fault module: <Java_root>\jre\bin\classic\jvm.dll

```

Work-around for JIT Problems

For a crash that is related to the Just In Time (JIT) modules, you may not want to upgrade the Java SDK yet, or you may need a work-around while IBM is working on a solution for you. Read the following to determine what work-around is most acceptable for your environment. You can work around the problem by either skipping a method or by disabling JIT completely.

► Skipping a method

To determine what method caused the JIT to crash, either look at the Current Method in the javacore, or enable COMPILING trace within JITC_COMPILEOPT to isolate the trouble even further.

For more information about the enable COMPILING option, refer to the technote *Selectively disabling JIT options for WebSphere Application Server V3.5, V4.0, V5.0, V5.1 and V6.0*, at the following Web address:

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21162255>

To determine what method you need to skip within a javacore, find the Current Thread entry where it shows the method that caused the crash.

It will look similar to the following:

```
1XMCURTHDINFO Current Thread Details
NULL -----
3XMTHREADINFO "Servlet.Engine.Transports : 2" (TID:0x10759F18,
sys_thread_t:0x39CCF340, state:R, native ID:0x8F4) prio=5
4XESTACKTRACE at
com.ibm.workplace.wcm.app.ui.portlet.widget.HTMLPopupMenuButtonRenderer.render (Unknown Source)
4XESTACKTRACE at
com.ibm.workplace.wcm.app.ui.portlet.widget.HTMLPopupMenuButtonRenderer.render(Unknown Source)
4XESTACKTRACE at
com.ibm.psw.wcl.core.ARendererFactory.performRender(ARendererFactory.java(Compiled Code))
```

From the above example, you can see that `com.ibm.workplace.wcm.app.ui.portlet.widget.HTMLPopupMenuButtonRenderer.render` caused the crash. To skip that method's JIT compilation and avoid the crash, alter the JVM's command line options to include the following:

```
JITC_COMPILEOPT=COMPILING:SKIP{
com/ibm/workplace/wcm/app/ui/portlet/widget/renderer/HTMLPopupMenuButtonRender
er}{render}
```

Note that there are no spaces. It is all entered as one line. You can substitute * for the method if there are different methods within the same package and class that caused the crash.

The JITC_COMPILEOPT options uses a semicolon (;) as a separator on Windows and a colon (:) for Unix.

► Disable JIT

An alternate way to skip the failing method is to disable JIT entirely. This is the easiest way to work around your issue without having to determine which method to skip. But, be advised that disabling JIT can cause noticeable performance degradation, which may not be an acceptable option for you.

To disable JIT, follow the steps outlined in one of the following technotes:

- *Disabling the Just-In-Time (JIT) Compiler in WebSphere Application Server V5.0 and V5.1 Releases*

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21084185>

- *Disabling the Just-In-Time (JIT) compiler in WebSphere Application Server V6.0*

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21194787>

Conclusion

By following this guide, you should be able to determine the failure for the majority of crashes within IBM WebSphere Portal. By implementing this troubleshooting approach, you should be able to increase your self-sufficiency and might avoid the need for creating a PMR with IBM support. If the suggestions do not provide the relief you need however, do contact IBM support and provide the information that is required by using ISA to collect your logs.

8.1.5 Portal Runtime failure

The following section provides troubleshooting information for Portal Runtime failures.

Definition of a Portal Runtime failure

Any failure during general operation and administration of WebSphere Portal is considered a runtime failure. This could include, but is not limited to, the following:

- ▶ Unable to login
- ▶ Administration tab unavailable
- ▶ Portlet deployment failures
- ▶ Portlets unavailable
- ▶ Page does not display

This information is intended to help you understand the WebSphere Portal runtime logs, and how to begin problem determination when you encounter an error during runtime.

WebSphere Portal has two runtime JVM log files: SystemOut.log and SystemErr.log. Both of these files are located in the `portal_root/log` directory. All runtime information for both Portal and the WebSphere_Portal application server are logged into these files. In addition, WebSphere Portal supports the redirection of its trace and message logging into separate log files. This can help you separate the log output of portal components from the output of other applications and WebSphere Application Server itself. The redirection is disabled by default. If you enable it, portal creates its own log file. It writes all trace and message logs to this new log file and no longer to the log files of WebSphere Application Server. You enable the redirection into separate log files by setting the log configuration key `useAppServerLog` to `false` in the file `log.properties`. This file is located in the `portal_root/shared/app/config` directory.

If tracing is enabled, a third runtime WebSphere Portal log file is generated that will contain additional messages and trace information. By default, this file is named `trace.log` file and is located in the `portal_root/log` directory. Steps to enable tracing and generate this log file will be discussed later.

Analyze failures during Portal Runtime

You started Portal and opened it in a Web browser. When you try to log in, you receive the following error message:

EJPAK0004W: Login failed. Please enter a valid user ID and password.

How do you begin problem determination?

Step 1

Review the SystemOut.log. In most cases, it is helpful to create a new, clean log and recreate the error. To do this, take the following steps:

1. Stop the WebSphere_Portal server.
2. Move SystemOut.log and SystemErr.log to a temporary directory.
3. Start the WebSphere_Portal server.
4. Recreate the problem and stop there.

This will generate brand new SystemOut.log and SystemErr.log files. New log files can be helpful because they are easier to read and should only contain the error or errors that you are looking for.

At this point you should have SystemOut.log open. Before moving forward with troubleshooting, there are a few important points to note about this file. At the top of the file, you should see something similar to the following:

```
***** Start Display Current Environment *****
WebSphere Platform 6.0 [ND 6.0.2.9 cf90614.22] [BPC 6.0.1.1 m0612.02] running
with process name nodename\nodename\WebSphere_Portal and process id 2384
Host Operating System is Windows 2003, version 5.2
Java version = J2RE 1.4.2 IBM Windows 32 build cn142ifx-20060209 (SR4-1) (JIT
enabled: jitc), Java Compiler = jitc, Java VM name = Classic VM
was.install.root = C:\WebSphere\AppServer
user.install.root = C:\ibm\WebSphere\profiles\wp_profile
Java Home = C:\WebSphere\AppServer\java\jre
ws.ext.dirs =
C:\WebSphere\AppServer\CEI\lib;C:\WebSphere\AppServer\CEI\client;C:\WebSphere\AppS
erver\java\lib;C:\ibm\WebSphere\profiles\wp_profile\classes;C:\WebSphere\AppServer
\classes;C:\WebSphere\AppServer\lib;C:\WebSphere\AppServer\installedChannels;C:\We
bSphere\AppServer\lib\ext;C:\WebSphere\AppServer\web\help;C:\WebSphere\AppServer\d
eploytool\itp\plugins\com.ibm.etools.ejbdeploy\runtime;C:\WEBSPH~1\PORTAL~1\shared
\ext;C:\WEBSPH~1\PORTAL~1\shared\ext\wp.jndi.jar
Classpath =
C:\ibm\WebSphere\profiles\wp_profile\properties;C:\WebSphere\AppServer\properties;
C:\WebSphere\AppServer\lib\bootstrap.jar;C:\WebSphere\AppServer\lib\j2ee.jar;C:\We
bSphere\AppServer\lib\lmpoxy.jar;C:\WebSphere\AppServer\lib\urlprotocols.jar
Java Library path =
C:\WebSphere\AppServer\java\bin;. ;C:\WINNT\system32;C:\WINNT;C:\WEBSPH~1\PORTAL~1/
shared/app/olexport;C:\WebSphere\AppServer\bin;C:\WebSphere\AppServer\java\bin;C:\
WebSphere\AppServer\java\jre\bin;C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem
;"C:\Program Files\Symantec\Norton Ghost 2003\"
***** End Display Current Environment *****
```

This block of text appears in the log at the very top each time you start the WebSphere_Portal server. It contains information regarding your environment at the time of the server startup. There are several important items to note:

1. Notice the `***** Start Display Current Environment *****` line and the `***** End Display Current Environment *****` line. These two lines bookend the environment variables.
2. The line `WebSphere Platform 6.0 [ND 6.0.2.9 cf90614.22] [BPC 6.0.1.1 m0612.02]` tells you what version of WebSphere Application Server and WebSphere Process Server you are using.
3. The line `Java version = J2RE 1.4.2 IBM Windows 32 build cn142ifx-20060209 (SR4-1) (JIT enabled: jitc), Java Compiler = jitc, Java VM displays what Java Runtime environment version you are currently running.`

After the `***** End Display Current Environment *****` line, diagnostic information for the startup of WebSphere_Portal is displayed. This continues until the line `[11/3/06 10:27:19:891 EST] 0000000a WsServerImpl A WSVR0001I: Server WebSphere_Portal open for e-business.` This line signals that WebSphere_Portal startup has completed and was successful.

To begin troubleshooting, scroll to the bottom of SystemOut.log. Since the last thing you did to Portal was recreate the error, the information you are looking for will have been the last thing logged to this file. At the bottom of the file, you may see something similar to the following:

```
[11/3/06 10:28:35:500 EST] 00000065 CacheServiceI I DYN1001I: WebSphere Dynamic
Cache instance named
ws/com.ibm.wps.propertybroker.common.cluster.ClusterEventManagerImpl.DistributedMa
p initialized successfully.
[11/3/06 10:28:35:750 EST] 00000065 ServletWrapper A SRVE0242I: [wps] [/wps]
[login]: Initialization successful.
[11/3/06 10:28:36:188 EST] 00000065 LTPAServerObj E SECJ0369E: Authentication
failed when using LTPA. The exception is WMM-UR: The password check for user
security name "wpadmin" failed. Root cause is: "Invalid password"..
[11/3/06 10:28:36:344 EST] 00000065 Authenticatio E
com.ibm.wps.auth.AuthenticationServlet doLoginWithExceptions
WASAuthenticationFailedException occurred:
com.ibm.wps.services.authentication.exceptions.WASAuthenticationFailedException:
EJPSD0001E: Authentication against WebSphere Application Server failed for user
wpadmin.
```

The errors highlighted above are the cause of our login failure. Specifically, the **SECJ0369E** and **EJPSD0001E** error codes.

Step 2

In the example above, it may be fairly obvious what the cause of the failure is. However, this will not always be the case. If you determine what the root cause of the failure is, but are unsure why you are receiving errors, you can begin searching for a solution to the **SECJ0369E** and **EJPSD0001E** error codes using the IBM Support Assistant (ISA).

<http://www-306.ibm.com/software/support/isa/>

Step 3

If the above problem determination approach did not resolve your issue, you can use a more advanced troubleshooting method by enabling additional tracing.

Note: Enabling the trace loggers can affect WebSphere Portal performance. We recommend that you only enable tracing for debugging purposes; otherwise, disable tracing.

To enable WebSphere Portal tracing, use the following steps:

1. Determine what tracing you need to enable. Refer to the following page in the WebSphere Portal Information Center for a list of trace strings for specific WebSphere Portal components:

http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp?topic=/com.ibm.wps.ent.doc/wpf/run_logs.html

2. Once you determine what trace strings you will need, proceed with the following steps:

- a. Access the WebSphere Application Server Administrative Console by using the following Web address:

http://hostname:port_number/ibm/console

- b. Go to the Server → Application servers section.
- c. Select the application server of your portal.
- d. Click Troubleshooting → Change Log Detail Levels.
- e. Specify the required trace settings. For example, this can be `com.ibm.wps.command.credentialvault.*=finest`
- f. Save your updates.
- g. Restart the portal.
- h. Recreate the issue.
- i. To disable tracing, specify tracestring: `*=info`, and restart the portal.

Tracing can provide more thorough debugging statements that might make it easier to determine the cause of your failure.

Conclusion

By following these approaches, you should be able to determine the failure for many issues when administering and operating IBM WebSphere Portal. By implementing this troubleshooting approach, you should be able to increase your self-sufficiency and might avoid the need for creating a PMR with IBM support. If the suggestions do not provide the relief you need, you can contact IBM support and provide the information that is required by using ISA to collect your logs.

8.1.6 Providing logs to support

If you already contacted support or feel that you may need to collect data to determine the nature of a problem regarding a WebSphere Portal failure, review the following information about how to use the ISA tool to collect logs.

The Automatic Problem Determination (AutoPD) log collection tool is fully integrated into the IBM Support Assistant (ISA). The ISA tool provides quick access to support-related information, plus service tools for problem determination and log collection so that you can provide IBM support with the information we need to effectively resolve your problem in a timely fashion. Additional information about ISA can be found at the following Web address:

<http://www.ibm.com/support/docview.wss?rs=688&uid=swg21235976>

You can also view the ISA home page at the following Web address:

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

Following are instructions for retrieving, installing, and using the ISA tool, with additional instructions on providing support with the output of the tool.

1. Retrieving ISA:

- a. To download ISA directly, go to the following Web address:

<https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=isa>

Note: You need an IBM ID (free registration) to sign in.

- b. After signing in, select the most current version of ISA for your WebSphere Portal platform and select **Continue**. Accept the license agreement, and click **I confirm**.
- c. On the next page, select the download file appropriate for your WebSphere Portal platform, and download ISA.

2. Installing ISA and the WebSphere Portal plug-in:

Note: This is a one time procedure only required for initial install and set up of the tool. If you already have the ISA tool installed, upgraded, and configured with the appropriate plugins, then you can skip to Step 3 - Using ISA.

- a. After downloading, unzip/untar the package into its own directory anywhere on the WebSphere Portal server machine. Launch the installer by executing the setup program (**setupwin32.exe**, **setupLinux.sh**) that is located in the root directory or folder.
- b. After ISA is installed, launch it from the install directory by executing the **startisa.sh** script (non-Windows) or **startisa.cmd** script (Windows).
- c. Once the ISA interface is displayed, you have a yellow message box instructing you to install appropriate plug-ins for your environment. Begin the install of the WebSphere Portal plug-in by clicking the **Updater** link in the yellow box or the **Updater** tab on the top right.
- d. On the Updater interface, you should select the **New Products and Tools** tab to locate new product plug-ins. Expand the WebSphere folder and scroll down to the appropriate WebSphere Portal plug-in (5.0, 5.1, 6.0) for your product version. Then select the plug-in by clicking the check box to the left of the plug-in name.

Note: If you get an error or do not see any new products, you may need to configure ISA to work with your proxy server. To do so, click **Preferences** → **Proxy Settings**, and input your proxy server settings. For more information, visit the ISA FAQ page: <http://www.ibm.com/software/support/isa/faq.html>

- e. Click the Install button, and agree to the installation terms that are displayed on the license window that appears.
- f. After the installation of the plug-in is completed, restart the ISA tool. Note: This not only installs the plugins requested, but also checks for ISA upgrades and automatically upgrades if required.

3. Using ISA:

- a. To use the ISA to collect logs for this problem, click the **Service** tab, and then choose to **Collect Data** on the left navigation pane.
- b. Choose the appropriate version of WebSphere Portal.
- c. Choose the **Portal Configuration Problem** from the drop-down, and click the Collect button. Later select to only **Proceed to Data Collection** to simply collect the diagnostic information.

- d. When the data collection has completed, a message is displayed indicating where the collected jar file resides on the file system.
4. Providing the output:
 - a. Follow the instructions documented in the technote *How to Transfer Files to WebSphere Portal Technical Support* at the following Web address:
<http://www.ibm.com/support/docview.wss?rs=688&uid=swg21201571>

8.2 The IBM serviceability tools strategy

The IBM Support Assistant (ISA) tool is the premiere strategic serviceability tool for IBM software. We recommend that all Portal customers take the time to download and install the ISA tool following the instructions in section 8.1.6. In future releases of Portal, the ISA tool will be part of the Portal install package.

With ISA, you can get to the information you need quickly. ISA provides this quick access through its concurrent Search tool that spans across the bulk of IBM documentation and returns the results categorized by source for easy review.

In addition to the Search, ISA provides a product information feature that has key product information links that are essential to self-help. These include the following:

- Product support pages
- Product education road maps and the IBM Education Assistant
- Product home pages
- Product recommended updates
- Product troubleshooting guides
- Product news groups and forums

Included in ISA is a new Tool workbench that provides you with the problem determination tools that IBM Support uses to resolve issues. Ongoing, different product teams make more and more tools available for ISA's tools framework in order to enable you to perform problem determination on your desktop.

ISA also provides a Service feature with an automated system and symptom based collector. The system collector gathers general information from your operating system, registry, and so on. The symptom based collection provides the unique ability to collect specific information relating to a particular problem that you are having. It can also provide you with the ability to automatically enable tracing that is helpful to IBM support as part of the data gathering process.

Another aspect of the Service feature is the problem submission tool. This allows you to enter your entitlement information once and have it saved for future sessions. You can then easily create a problem report for IBM and attach the collector file at the same time. Simple to do and yet extremely helpful for expediting a solution back from IBM.

So whether you need to find information about a software fix, collect key logs, or want to build your skills on a particular product, the IBM Support Assistant can help you get that done.

8.3 Self service resources

The information provided in this section gives you additional resources for Portal product documentation.

8.3.1 WebSphere Portal Product Library

<http://www-128.ibm.com/developerworks/websphere/zones/portal/proddoc.html>

The WebSphere Portal Product Library site contains information about many aspects of WebSphere Portal from version 4.1 to 6.0. You can find references to the following information:

- WebSphere Portal Information Center
- Release Notes
- Hardware and Software Requirements
- Javadoc™
- Performance Guides
- Quick Start Guide
- Articles and Tutorials

8.3.2 WebSphere Portal version 6.0 Information Center

<http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp>

The WebSphere Portal version 6.0 Information Center is a quick reference guide for everything you need to know about WebSphere Portal. It contains information about how to do the following tasks, as well as many other topics:

- Install and configure Portal
- Migrate between versions
- Administer Portal
- Integrate Lotus Domino
- Design portlets, pages, themes and skins
- Tune your Portal for better performance

8.3.3 WebSphere Support Pages

The WebSphere support pages are self-help Web sites that contain useful information regarding important fixes, technical tips, solutions and work-arounds for various issues, and additional product documentation.

- ▶ WebSphere Portal Support Page
<http://www-306.ibm.com/software/genservers/portal/support/>
- ▶ WebSphere Application Server Support Page
<http://www-306.ibm.com/software/webservers/appserv/was/support/>
- ▶ WebSphere Process Server Support Page
<http://www-306.ibm.com/software/integration/wps/support/>

These support pages are valuable tools for troubleshooting any issue you may encounter. From this site, you can search for relevant fixes or work-arounds for the errors that you receive. In addition, you can obtain the latest updates, fixpacks, flashes, and other information from here.

For example, if you examine the WebSphere Portal Support page, you may see something similar to the following:

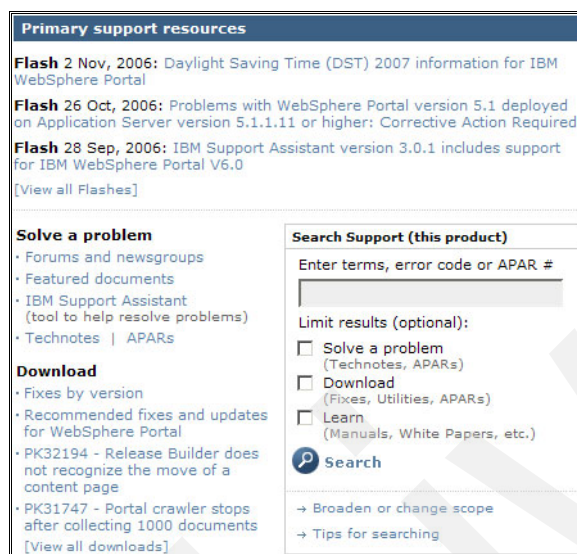


Figure 8-3 Section of WebSphere Portal Support page

The section *Primary support resources* appears on most WebSphere support sites and looks similar to Figure 8-2 on page 389. At the top of this section, you can see the most up to date Flashes. These flashes contain the most important Portal information such as mandatory fixes, product releases, and so forth.

At the bottom left, you see two lists. The first is *Solve a problem*. Here you will find links to resources that might be helpful in troubleshooting. The next section is *Download*. Here you will find links to fixes and fixpacks that you can download and apply to your Portal environment.

To the right side, there is a section to search the Portal support site. This is useful if you are trying to do the following:

- ▶ Find technotes or solutions for errors you are receiving. For example, you can search for the error code **SECJ0369E**.
- ▶ Find specific interim fixes or fixpacks. For example, you can search for **PK30928** or **Cumulative Fix 4**.
- ▶ Find additional documentation for any topic. For example, you can search for **Portal Cluster**.

8.4 Monitoring WebSphere Portal

This section will provide an introduction to monitoring techniques as well as additional steps you can take to help keep your Portal environment running smoothly.

8.4.1 Monitoring techniques

Monitoring your system is very important for maintaining a healthy Portal environment. WebSphere Application Server provides a monitoring strategy to help you understand the following aspects:

- ▶ The response times perceived by end users.
- ▶ The basic health of the systems that participate in an end-to-end user request.
- ▶ The resource usage of applications.

Monitoring user response time

Monitoring user response time is an external perspective of how the overall Web site performs from a user view, and identifies how long the response time is for a user. From this perspective, it is important to understand the load and response time on your site. To monitor at this level, many industry monitoring tools, for example, Tivoli Monitoring for Transaction Performance, support you to inject and monitor synthetic transactions, helping you identify when your Web site experiences a problem.

More details can be found at the following Web site:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/tprf_monitoringresp.html

Monitoring basic health of systems

Monitoring overall system health is fundamentally important to understanding the health of every system involved with your system. This includes Web servers, application servers, databases, back-end systems, and any other systems critical to running your Web site.

WebSphere Application Server provides Performance Monitoring Infrastructure (PMI) data to help monitor the overall health of the WebSphere Application Server environment. PMI provides average statistics on WebSphere Application Server resources, application resources, and system metrics. Some of the more important statistics are as follows:

- Average response time
- Number of requests
- Number of live HTTP sessions
- Web server thread pools
- The Web and Enterprise JavaBeans (EJB™) thread pools
- Database and connection pool size
- Java virtual memory (JVM)
- CPU
- I/O
- System Paging

To monitor several of these statistics, WebSphere Application Server provides the Performance Monitoring Infrastructure to obtain the data, and provides the Tivoli Performance Viewer (TPV) in the administrative console to view this data.

More details can be found at the following Web address:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/tprf_monitoringhealth.html

Manage application resource usage

Understanding the application flow that satisfies the user request is important for the in-depth internal understanding of who is using specific resources. WebSphere Application Server provides request metrics to help trace each individual transaction as it flows through the application server. It records the response time at different stages of the transaction flow, for example, request metrics records the response times for the Web server, the Web container, the Enterprise JavaBeans container, and the back-end database. Additionally, several IBM development and monitoring tools that are based on the request metrics technology (for example, Tivoli Monitoring for Transaction Performance) are available to help view the transaction flow.

More details can be found at the following Web address:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/tpmf_monitoringappflow.html

8.4.2 Portal update strategy

Keeping WebSphere Portal up-to-date helps ensure that you maintain a more stable environment. By applying routine Portal maintenance you can avoid hitting issues that were already encountered, documented, and fixed.

The following Web site contains the most recent update strategy for WebSphere Portal.

<http://www-1.ibm.com/support/docview.wss?uid=swg21248245>

The above link covers the following information:

- Update Strategy Overview
- Recommended Update Path
- Tentative Release Schedule
- Maintenance Planning
- Delivering Updates

Deployment automation options

Deployment automation refers to the use of available tools or technologies to reduce the manual steps required by the portal server installation and configuration process. This chapter discusses various options that the large enterprise can leverage to automate the portal deployment to reduce overall deployment time. We first introduce the underlying technologies that can be used to enable automating portal deployment, then we discuss some sample scenarios utilizing these technologies.

A.1 Motivation of deployment automation

The enterprise scale portal deployment typically requires a large number of repeating portal server deployments on multiple physical machines. It is desirable to use tools or scripts to automate the repeating deployment steps so that the system administrator does not have to repeat the labor extensive work on every machine, multiple times. Although requiring extra effort to develop, the automation capability will result in tremendous benefits to the overall deployment process, which includes the following:

- ▶ Reduced labor cost because overall shorter time is required for the deployment.
- ▶ Increased productivity because the automated process eliminates a lot of user involvement.
- ▶ Improved product quality because fewer chances of human errors can be introduced in the process.

Creating automated solution requires time, commitment, and resources knowledgeable of related technologies. This is no doubt a non-trivial investment that will bring long-term benefits overtime. We suggest that you carefully review the solutions presented in this chapter, and make long term plans before immediately adopting one or more solutions. Careful planning of an automated process goes a long way to reduce the time to automate the process, and it will also create quality repeatable assets that can be leveraged many times. The main concerns this chapter tries to address to create repeatable quality automation assets include the following:

- ▶ The tool or technology must be simple to use. It does not make sense for the portal administrator to learn a new programming language in order to write this solution.
- ▶ The solution must be easy to extend. Many enterprises can only afford writing small solutions over time, and this requires that the solution allow incremental development: start simple and grow overtime.
- ▶ The solution should be reusable over time. It does not make sense to rewrite everything when new releases of WebSphere Portal or Application Server need to be adopted.
- ▶ The solution should be cross platform. This maximizes your return of investment (ROI) by supporting that the same solution runs on multiple environments.

Implementing some of the suggestions discussed in this chapter may require additional development skills. We recommend that you explore these options to streamline your deployment process. You can find related tutorials on the IBM DeveloperWorks Web site, or you can engage IBM Software Services team to help you develop these automation solutions.

<http://www-128.ibm.com/developerworks/websphere/zones/portal/>

A.2 Tools and technologies used

This section introduces the most relevant tools or technologies that can be used for automation. In the next section, we present some automation examples that use these technologies.

A.2.1 ANT/Maven

Typically an automation is implemented using one or more scripting tools, for example, batch scripting on MS-DOS®, and shell scripting on Unix or Linux. However the main drawback of these scripting languages are the following:

- ▶ They are platform dependent.

- ▶ They are difficult to implement reusable components.
- ▶ There are very little available integration samples to facilitate routine operation on J2EE application servers.

For example, it is very complex to write a shell script to automate a very typical deployment scenario on a portal server:

1. Copy the Portlet war file to the target file system.
2. Deploy the Portlet war file to portal server.
3. Synchronize the Portlet files across multiple nodes in a cluster.
4. If the previous step is successful, add portal pages that have references to the Portlet.
5. If the previous step is successful, add a URL mapping context for the newly created page.

Fortunately, there is an alternative in the J2EE world to make this kind of automation easier. That answer is ANT and Maven.

Developed by system administrators and J2EE developers, ANT and Maven are very popular open source (so it is free!) tools for building, packaging, and deploying J2EE projects. You can download ANT from the Apache Web site:

<http://ant.apache.org>

Alternately, you can use the Portal server bundled ANT package. ANT requires the operations to be written in one or more ANT XML files with the default name of the XML being *build.xml* for ANT. ANT can be easily extended as the scripting tool to integrate multiple configuration and deployment operations in a portal environment.

ANT and Maven are very similar to each other in terms of the way users interact with them, Maven makes up for some deficiencies that ANT has, such as project management. ANT alone is sufficient to serve our automation needs. We will demonstrate the examples using ANT for the most part in this section. You can always achieve the similar goal using Maven.

Similar to any other scripting tools, before you can execute ANT, you have to somehow create a script (either write a new script or copy from some examples) to describe how the machine should perform each task and to specify the logical connection between individual tasks. Unlike typical scripting languages that require writing shell commands, in ANT or Maven, the instructions are written in a structured XML file. The file references a target tree in which various tasks are run.

Before we review any details, let us check an ANT XML example showing an ANT task in action. The sample is used to automate the task 3 scenario described earlier in the section, for example, the scenario to deploy a portal page.

Example: A-1 An ANT XML example

```
<?xml version="1.0" encoding="UTF-8"?>

<project name="PortalDeploy" default="default" basedir=".">

  <taskdef name="xmlaccess" classname="com.ibm.wps.xmlaccess.XmlAccessTask"
    classpath="WPS/bin/wp.xml.client.jar"/>

  <!-- project properties contains reusable variables
  <property name="PageUniqueName" value="ITS0.myPage"/>
  <property name="targetPortalUser" value="wpsadmin"/>
  <property name="targetPortalUrl" value="http://localhost:9081/wps/config"/>
```



```

<!-- Target starting the page deployment,
<target name="deploy-page" depends="deploy-war">
<!--"depends" specify the WAR deployment must be successful first
    <xmlaccess
<!-- xmlaccess is a built in ANT task to call xmlaccess command
        user="${targetPortalUser}"
        password="${targetPortalPwd}"
        url="${targetPortalUrl}"
        srcfile="${PageUniqueName}.source.xml"
        destfile="${PageUniqueName}.target.xml" />
</target>
</project>

```

In Example A-1 on page 409, first you see some definition of a custom task *XMLAccess*, which is a custom ANT task developed by IBM to facilitate XMLAccess commands invoking in ANT. Any task not built into standard ANT must be declared using *taskdef*. Next you see definitions of variables and their default value. Finally you see that the *deploy-page* task was defined to import an XMLAccess file that describes a page to the portal server. At runtime, when the system administrator executes the *deploy-page* task (the actual *deploy-war* task is not listed here) because of the *depends* clause, ANT always invokes the *deploy-war* task first to make sure all the referenced Portlet WAR files are deployed first. If for some reason the *deploy-war* task failed, the *deploy-page* task will not execute.

After you create the ANT xml file and save it as *build.xml*, you can execute the task in the command line as the following: (Use *ant.sh* to replace *ant.bat* on Unix, remember the same ANT will run on both all the OS that Java runs.)

```
ant.bat -DtargetPortalPwd=wpsadmin deploy-page
```

In addition to using the standard ANT command, you can also use a customized ANT command—*WPSconfig* as follows. We will go through more details on how to prepare ANT scripts in order to use *WPSconfig* in section A.3.3. In a nutshell, you can invoke *WPSconfig* just as you normally invoke ANT. For example, Example A-1 on page 409 can also be invoked as the following:

```
WPSconfig.bat -DtargetPortalPwd=wpsadmin deploy-page
```

Important: Notice here that you can specify the password in the command line instead of storing it in the XML file for security reasons. *Deploy-page* is the task name to be invoked. If a task is not specified in the command, ANT will automatically invoke a task named *default*.

As you can see from the example, when you write new ANT XML scripts, the unit operation of ANT is task, defined as *target* in ANT XML. One target can invoke one or more targets to form a single-entry point for the complex tasks. Consequently, there are two types of ANT task:

- ▶ *unit level ANT task* that does actual operations, for example, XMLAccess task to import a file, copy task to copy files, and so on.
- ▶ *composite ANT task* that integrates other unit level ANT tasks or other composite ANT tasks to complete a more complex job.

There are many built-in ANT tasks available to perform the unit level tasks. You can refer to the following Web site for available general purpose unit level tasks.

<http://ant.apache.org/manual>

Additionally, for the tasks that are specific to WebSphere Application Server and WebSphere Portal, they are available using the ANT WP extension. Only in some rare cases, you may need to write ANT custom tasks to perform customized unit-level tasks.

The unit level tasks are the fundamental building blocks of automation. Once you have all unit level ANT tasks identified, you can create composite ANT tasks to aggregate pieces together. In the composite ANT task definition, you can implement the dependency using the *depends* clause. We suggest you adopt a top-down approach to break the complex deployment process into simpler tasks, which can be further simplified until it reaches the unit-level task. Ideally, you want to build multiple levels of tasks so that each task itself does not contain an overwhelmingly large number of actions. Smaller composite tasks promote reusability and are easier to test. As you move along the process of building XML to automate the deployment operations, you will realize that XML is more like a tree model, where the top level node contains the high-level user interface and the bottom level node contains the unit-level task.

Figure A-1 shows the top-down approach we used to create the XML structure in the previous example. We first break down the Portal deployment into two sub tasks, deploy Portlet, and then deploy pages. Then we further break down Portlet deployment into unit-level tasks: file copy action to move WAR file, XMLAccess to deploy WAR, then WAS command to activate portlet.

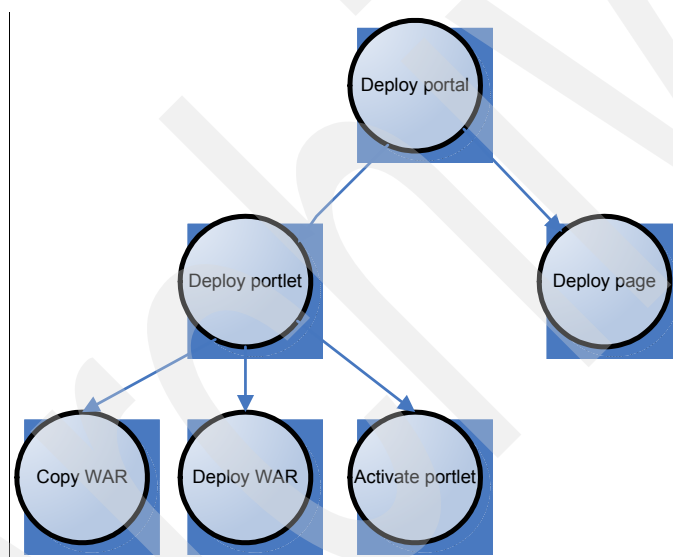


Figure A-1 Top down approach to create the XML structure

In order to leverage a WebSphere Portal provided ANT task extension (for example, wsadmin ANT task and XMLaccess ANT task) or to use existing a WAS specific ANT task (for example, synchronizing multiple portal nodes in a cluster) you need to use a portal server bundled ANT package—WPSconfig. WPSconfig.sh/bat is a WebSphere Portal provided add-on extension of standard ANT package that simplifies the portal deployment process. WPSconfig.sh is the ANT front-end user interface that has all the Portal extensions automatically loaded. WPSconfig.sh is referred to as a Configuration Task in the Portal Infocenter. The Portal administrator should be very familiar with the WPSconfig tool, as it is

the standard tool to configure various Portal server settings (for example, create virtual portals and configure portal security).

Notice: WAS also provides an integrated ANT interface—`ws_ant`—that only contains WAS specific extensions. `wpsconfig` is a super-set of `ws_ant`. You should always use `wpsconfig` as the ANT interface as it provides access to the built-in custom tasks used for both WAS and WP.

Existing WAS/WP specific ANT tasks cover many standard server operations, such as the following:

- ▶ XMLAccess task to import or export portal server configuration
- ▶ Install and uninstall applications
- ▶ Start and stop servers in a base configuration
- ▶ Run `wsadmin` (administrative scripts or commands)
- ▶ Run the Enterprise JavaBeans (EJB) deployment tool

In addition to using `WPSconfig.sh` to execute existing portal configuration tasks, the system administrator can write custom task definitions that will be loaded so that the task can be executed from `WPSconfig.sh`. `WPSconfig.sh` automatically retrieves custom-task definition XML from `$WPS_HOME/config/includes/`.

`WPSconfig.sh` is executed the same way as standard ANT.

```
./WPSconfig.sh -Dproperty1=value1 -Dproperty2=value2 task-name1 task-name2
```

Similar to ANT, `WPSconfig.sh` should be executed with a list of tasks. Reusable variable values should be specified in a properties files, `wpconfig.properties`. `WPSconfig.sh` also allows user overwriting default parameter values by specifying them as the command line options. This provides a handy way to quickly test the ANT script without modifying the XML or properties files.

Now that you have learned the basics of automation using ANT and the WP customized version of ANT, `WPSconfig`, you will see real life scenarios in section A.3, and we will demonstrate how to write composite ANT tasks to automate complex deployment scenarios.

A.2.2 WSAdmin and Jython

ANT is a great tool to integrate multiple tasks together, but as an integration tool, it does not directly interact with WebSphere Application Server's system management infrastructure. To use scripting interface to perform administrative operations, you need to use `wsadmin`, the powerful, non-graphical command interpreter environment enabling you to administrate application servers in a scripting language.

Out-of-the box, WAS administrator can leverage the administrative console to manage and control WAS in a graphical user interface (GUI). The GUI is easy to use, but it is simply unmanageable if you have to repeat the same WAS configuration change using the GUI on large scale (>10) application servers deployments. It is very difficult to automate the GUI operations. The ideal way of automation is to create reusable scripts and then have the system administrator run the script on each application server. Additionally, once you have an individual script as the building block to automate unit level operation, you can use ANT to “glue” multiple functions together to achieve high-level automation.

`WSadmin` is the command line environment equivalent to the administrative console GUI for the system administrator to perform the system management operations. The scripting

capability allows using either Jython or Jacl to automate WAS-related configuration tasks and controlling (start and stop). Since the WebSphere Application Server v5.1 release, the WebSphere administration scripting tool (wsadmin) supported both Jacl (Java TCL) and Jython (Java Python) as the scripting languages. When selecting a scripting language, Jython is the strategic direction as the administration scripting language for WebSphere Application Server because Jython is more intuitive to object oriented (such as Java) developers. Future enhancements of the scripting language are focused on the use of Jython.

Jython is a pure Java implementation of Python, an object oriented scripting, programming language. Software developers who are familiar with object oriented programming will find that it is intuitive to use Jython. It is seamlessly integrated with WAS system management. You do not need any additional Java libraries in order to execute Jython scripts that interact with WAS.

The recommended development approach for a first time Jython/Jacl script developer is to start with trying the individual command in the interpreter environment provided by wsadmin. In the interpreter environment, wsadmin immediately responds to your command so you get feedback on your input without the typical long cycle of code - save - test - code. So you can spend more time coding.

We will use one frequently used task as our example to see wsadmin in action. The scenario is that we want to change portal JVM heap size to 1024. Of course you can achieve the goal by looking at the WAS admin console, but doing the same thing over and over at every portal node is a lot of tedious work. Instead, we will see how we can achieve the goal in wsadmin using Jython.

1. Start wsadmin in interactive mode. (In the listing below, the bold text is the actual command typed.)

```
C:\ibm\WebSphere\profiles\wp_profile\bin>wsadmin -conntype NONE -lang jython  
WASX7357I: By request, this scripting client is not connected to any server  
process. Certain configuration and application operations will be available in  
local mode.  
WASX7031I: For help, enter: "print Help.help()"
```

2. Locate the WAS configuration object that stores the JVM settings. WAS stores objects in a hierarchical tree. You can always get the handle of the configuration object using AdminConfig interface. In our case, we use the route Server, JVM.

```
wsadmin>server1 = AdminConfig.getId('/Cell:wpsiut/Node:wpsiut/Server:WebSphere_Portal/')  
wsadmin>print server1  
WebSphere_Portal(cells/wpsiut/nodes/wpsiut/servers/WebSphere_Portal|server.xml#  
Server_1154005406891)  
wsadmin>jvm = AdminConfig.list('JavaVirtualMachine', server1)  
wsadmin>print jvm  
(cells/wpsiut/nodes/wpsiut/servers/WebSphere_Portal|server.xml#JavaVirtualMachi  
ne_1154005406906)
```

3. Now that we have the object of JVM, we can modify it.

```
wsadmin>AdminConfig.modify(jvm, [['maximumHeapSize', '1024']])  
''
```

4. Empty quotes mean success. Save the change.

```
wsadmin>AdminConfig.save()  
''
```

5. Again, an empty quote means the save is successful. Now that we verified that the procedure works, we can create a new Jython script. Name it `updateWasHeap.py`. Copy and paste the commands we used during steps 2-4.

```
#updateWasHeap.py
server1 =
AdminConfig.getid('/Cell:wpsiut/Node:wpsiut/Server:WebSphere_Portal/')
jvm = AdminConfig.list('JavaVirtualMachine', server1)
AdminConfig.modify(jvm, [['maximumHeapSize', '1024']])
AdminConfig.save()
```

6. Congratulations! You just wrote your first Jython automation script. You can run this script on any portal server node as the following:

```
wsadmin.bat -conntype NONE -lang jython -f updateWasHeap.py
```

There are sample Jython scripts for almost all the typical management operations, so you do not have to reinvent the wheels every time. Chances are that you will spend more time modifying existing scripts than writing new scripts. You can get many sample scripts at the WAS Infocenter by visiting the following Web site:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/txml_configserver.html

As you can see, WSadmin scripting using Jython is an ideal method to replace the repeating steps that are required in the Portal deployment procedures, such as modifying WebSphere_Portal app server configuration for basic performance tuning, redeploying `wps.ear` and synchronizing all portal nodes in a cluster. With this scripting, you can further leverage System Management tools such as Tivoli Provisioning Manager to automatically push the change in your enterprise without manual work.

A full introduction to WAS scripting tool is beyond the scope of this book; however, you can find detailed tutorials for WAS scripting tool in the IBM book *WebSphere Application Server V6 System Management & Configuration Handbook*, SG24-6451.

<http://www.redbooks.ibm.com/abstracts/SG246451.html?Open>

A.2.3 WebSphere Application Server Installation Factory

In certain situations, you may want to install WebSphere Application Server separately before you install WebSphere Portal Server. For example, when you want to install a different version of WAS than the version shipped with WP V6.0 instead of using portal server embedded WAS install media to install both WAS and Portal server. Typically in this situation, there are many steps to follow: After the WAS install, you need to manually install WAS fixpacks and maybe individual fixes afterwards. Then you may also want to make customizations to the application server install instance, for example, installing custom EJBs to connect to the back-end service or adding datasource definitions for your custom application data. This work multiplied many times (one on each node) means a lot of labor extensive manual work for the system administrator. Remember this is just the preparation for portal server install.

Fortunately, WebSphere Installation Factory tooling was introduced in WAS 6.0.2 or newer to allow you take a snapshot of the customized WAS instance, then create a custom install image using the snapshot so that you can deploy the rest of WAS nodes with the snapshot. This new capability allows you to build your customized WAS install image, referred to as CIP(customized installation package) in WAS infocenter. This enables the administrator to deploy a WAS with specific fixpack/efix/configuration in one single step. The system administrator can also customize the WAS install image with the required WAS fix level and

configuration update (change session timeout value etc) on the default WAS application server (server1). After the custom image is created using installfactory, the administrator can deploy the customized WAS using the new image in one single step instead of repeating the steps on every machine.

The key enabling feature here is configuration archive. A configuration archive (CAR) file captures the configuration of a stand-alone application server profile for later restoration. The CAR can help clone the original profile to another machine or system. The configuration archive can contain any configuration updates made to the application server, including installing EAR, server updates, and so forth. For more information about configuration archive, visit the Infocenter at the following Web site:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/cins_if_config_archive.html

You can also use the IBM Education Assistant:

ftp://ftp.software.ibm.com/software/eod/was/6.0/SystemManagement/WASv6_SM_Configuration_Archives.pdf

Important: WAS Installation Factory only supports the configuration updates on a single-node mode. It will not support the configuration changes that related to distributed environments, such as the WAS server with security enabled using LDAP server or a WAS node that has been federated.

Installation Factory provides the long term strategy that WebSphere family products use to simplify the install procedure to tailor clients' need. The Installation Factory makes it possible to *clone* a configured WAS install so that the future installed WAS contains the exact same maintenance level and configuration update. Unfortunately, WebSphere Portal has not implemented Installation Factory in V6. As a long term direction, WebSphere Portal may fully implement WPS Installation Factory in future releases.

Because of the large amount of time required to install and configure portal server, it is tempting to have a similar clone solution to replicate your existing portal install to new Portal server. The closest Portal clone solution may be achieved through a portal instance copy and localization, for example, copy a completely configured portal server, and move it to a 2nd box, run a set of localization script to fix up the incorrect reference to hostname/ip/portal unique ID, and so forth, so that it can run on the local portal server. Although possible in theory, the clone solution is not tested fully in the product, and there is no product level support for this portal clone solution. In other words, your portal instance deployed in this matter will not be supported if there is any problem. However, if you are interested in this portal clone solution, you need to engage the IBM Software Services team. IBM Software Service team interacts with clients to make multiple successful portal server clones and provides support on a per client basis during portal server V5 time frame.

A.3 Sample scenarios of automation

This section provides four sample scenarios to illustrate how to use the tools and technologies, discussed in the previous sections, to automate various steps of the deployment process. The steps we will automate come from previous chapters of this book.

A.3.1 Use Installation Factory to build custom WAS install image

WAS Installation Factory was a new tooling feature added since WAS 6.0.2. It provides the *clone* function for you to limit the pain of preparing the WAS stack to the first time install experience, and once the custom install image is created, you can deploy the full WAS stack in one step. This full custom install image includes the following:

1. WAS Base
2. WAS maintenance (refresh pack, fixpack and ifixes)
3. WAS profile (this is achieved via WAS configuration archive functions)*
4. Applications installed to the default profile*

*These two parts can be achieved by either using configuration archive or using additional profile creation scripts (jython).

To create custom install images using WAS Installation Factory for the first time, we need to prepare the WAS stack as documented in the install document. Notice the following:

1. The Installation Factory image is not cross-platform. It must be created on the operating system that the install image is targeted to support, for example, we can only create the AIX install image on AIX box.
2. Install factory can only contain one profile saved as “configuration archive”. To generate the configuration archive, run the following wsadmin command:

```
./wsadmin.bat -conntype none -c '$AdminTask exportWasprofile {-archive /defaultprofile.car}'
```
3. DO NOT install Portal before creating the WAS Installation Factory image.
4. The custom install image cannot be modified. After the custom install image is created, recreate the template image again if you want to change the configuration in the template. There is no way to update the install template image directly.
5. After the WAS stack is complete, run WAS Installation Factory tooling. Download Installation Factory tooling from the following Web site:
<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg24009108>
6. Follow the instructions to unzip it to any folder that the current logged on user has write access. If the current user cannot write to the current folder, Installation Factory throws Java OutOfMemory exceptions and exits abnormally.

WAS Installation Factory was packaged as an Eclipse plugin, and it supports both GUI mode and CLI mode. CLI mode is easier to execute as it does not require X client connection. In order to use CLI, you must create the configuration XML BuildDefinition.xml, and specify the input value in the file. The easiest way is to modify the sample Installation Factory build XML to reflect the correct media location. Installation Factory must have write access to the output directory and read access to the following directories:

- ▶ Full WAS where was installed completely as the pre-req
- ▶ Install media of base WAS install image
- ▶ Maintenance package files downloaded to local directory
- ▶ Configuration archive file (defaultprofile.car in the example) generated by wsadmin command

First, specify the output director in the XML. Assume that the target directory is /was_staging_area/if

```
<buildOptions>
```



```
<targetLocation>/was6_staging_area/if</targetLocation>
```

Specify refreshpack directory and filename in the XML. In this example, we specified the XML to use configuration archive defaultprofile.car:

```
<sourceMaintenanceInstallPackages installOrder="1" maintenanceType="refreshPack">
  <rootFolder>
<rootFolder
permissions="644">/opt/WebSphere60/AppServer/updateinstaller/maintenance</rootFold
er>
<fileNamePattern
isRegex="false">6.0-WAS-WAS-AixPPC32-RP0000002.pak</fileNamePattern>
Specify configuration archive location in the XML
  <configurationInfo>
    <profileCreationActions>
      <configurationArchive executionOrder="1"
fatalErrorWhenExecutionFailed="true">
        <rootFolder>
          <whichFolderToUse>literalRootProvided</whichFolderToUse>
          <rootFolder>/was6_staging_area</rootFolder>
        </rootFolder>
        <relativeFolder>.</relativeFolder>
        <fileNamePattern
isRegex="false">defaultprofile.car</fileNamePattern>
      </configurationArchive>
    </profileCreationActions>
  </configurationInfo>
```

After you complete the XML update, launch the Installation Factory CLI:

```
Ifcli.sh -buildDef BuildDefinition.xml
```

The process takes about 40 minutes to complete. After the creation process finishes, the out folder structure should look like the following:

```
/$OUTPUT_DIR/ifpackage
Copyright.txt
JDK
WAS
componentmaps
docs
framework
install
lafiles
lib
maintenance.xml
media.inf
readme
readme.html
repository
repository.zip
responsefile.nd.txt
responsefile.pct.NDmgrProfile.txt
responsefile.pct.NDmanagedProfile.txt
responsefile.pct.NDstandAloneProfile.txt
setup.jar
custom
```


Now that you have created your custom WAS install image, you can use it just like any regular WAS install image to deploy WAS. The only exception is that now after the install it contains customizations you have made on your snapshot WAS instance.

The following shows how to use the command line interface to invoke the created install image for the new WAS install. Here, the response file provides all necessary user input (as usual, you can modify the response file that comes with WAS install image):

```
# nohup ./install -options /opt/WebSphere60/responsefile.nd.txt -silent 2>&1  
>/opt/WebSphere60/install.nd.out &
```

After the install completes on the new machine, we suggest that you verify the WAS version information using the version Info tool provided by WAS.

```
# /opt/WebSphere60/AppServer/bin/versionInfo.sh -maintenancePackages  
Installed Product
```

```
-----  
Name                IBM WebSphere Application Server - ND  
Version             6.0.2.3  
ID                  ND  
Build Level         cf30542.05  
Build Date          10/18/05  
Installed Maintenance Package  
-----  
Maintenance Package ID 6.0.2-WS-WASND-FR000003-2006-02-22_18_52_40  
Description            This is the install factory install image.  
Build Date             2005/01/20  
Installed Maintenance Package  
-----  
Maintenance Package ID PK11623  
Description            NPE in FileTransferClient for standalone client.  
Build Date             2005/09/29
```

The results above shows that our installed image contains all the customizations we made to a standard WAS instance. The custom install package was successfully created.

A.3.2 Use WSADMIN to automate application server configuration change

WebSphere Portal, in essence, is a custom application server running on top of WebSphere Application Server. It utilizes WAS provided services to provide enterprise level information integration. Not surprisingly, a lot of portal operations related to portal deployment must be performed with WAS provided tools—wsadmin. Section A2.2 introduced the wsadmin tooling and methodology to develop script to automate server configuration changes (the example shown earlier was for changing portal server heap size).

In this section, we will demonstrate how to extend the earlier example to implement a more complete performance tuning script that you can run on each server. This section assumes you visited Chapter 6, “Strategies for tuning and testing” on page 321 for basic performance tuning concepts and Appendix A.2.2, “WSAdmin and Jython” on page 412. It also assumes that you are familiar with the basic concept of using wsadmin scripting interface and development methods (for example, starting scripting with using the command in wsadmin interpreter environment, after verifying that the script is working, then integrating them together into a script file. The final script can be invoked using wsadmin command -f option, for example, wsadmin -f your.script).

Having introduced Jython in the previous section, in this section we will demonstrate how to implement the similar function in Jacl. You will see Jacl scripting is very similar to Jython scripting. You can always achieve the same functions using either tool.

In summary, the application server setting changes that we wanted to automate include the following (assume this is Windows):

```
JVM setting:Initial and maximum heap size: 1280
JVM setting:Session timeout: 10 minutes
AppServer setting:Minimum and Maximum Thread pool size: 50
AppServer setting:Enable pass by reference: true.
```

In jacl, you always need to get a handle on the WAS configuration object that you want to modify before you can access any attributes of that object. This is very similar to Jython. If the object is embedded in another object, you must recursively navigate the structure until you reach the desired configuration object, for example, heap size and session timeout settings are attributes of the JVM setting object. The JVM setting object is embedded under the application server setting object. You must go through Application Server -> JavaVirtualMachine to get the handle of the JVM setting object. This is shown in the following:

```
set wps [$AdminConfig getid /Cell:cellName/Node:nodeName/Server:WebSphere_Portal/]
set jvm [$AdminConfig list JavaVirtualMachine $wps]
```

After you get the handle of the JVM setting object, we can modify the attributes:

```
$AdminConfig modify $jvm {{ initialHeapSize 1280}}
$AdminConfig modify $jvm {{ maximumHeapSize 1280}}
```

Similarly, we can traverse the hierarchy to get access to the configuration object WebContainer - SessionManager - TuningParams, which stores the session timeout attribute (invalidationTimeout). After you get the handle of TuningParams, we can modify the attributes:

```
set wc [$AdminConfig list WebContainer $wps]
set sm [$AdminConfig list SessionManager $wc]
set tp [$AdminConfig list TuningParams $sm]
$AdminConfig modify $tp {{invalidationTimeout 10}}
```

Using the same method, we can enable the “Pass by Reference” attribute of ORB service:

```
set orb [$AdminConfig list ObjectRequestBroker $wps]
$AdminConfig modify $orb {{noLocalCopies true}}
```

After the changes are made, save to commit the changes.

```
$AdminConfig save
```

After you commit the changes, launch WAS administrative console to verify these changes.

The power of the scripting effort is more visible if you can add some programming logic to the script to handle more general cases. For example, typically you have multiple WebSphere_Portal app server instances managed by a single Deployment Manager (DMGR). Now you can use wsadmin to connect to the DMGR. In the script add a loop to go through all the nodes and get a list of all portal application servers in the cell, then modify all the application servers in the cell in a single step. This reduces your workload dramatically, compared to the otherwise labor extensive manual work to go through WAS administration console for each portal server.

The loop to go through all portal server nodes is as follows:

First we need to get a list of servers, and store it into variable *servers*.

```
set cell [$AdminConfig getid /Cell:mycell_name/]
set servers [$AdminConfig list Server $cell]
```

Then we can loop through all entries in the list:

```
foreach serverName $servers {
    set searchIndex [string first portal $serverName ]
    if {$searchIndex !=-1} {
        #process you app server, update JVM, session management etc
        #this will process all the nodes }
    else {    puts "nothing to do for this server"  }
}
```

To put all of these pieces together, the following listing shows a complete script to automate the WAS server performance tuning step.

```
#update this before running
set cellName=mycell_name

#First get a list of all servers
set cell [$AdminConfig getid /Cell:mycell_name/]
set servers [$AdminConfig list Server $cell]

foreach serverName $servers {
    #we only process the server if the server name starts with "portal"
    set searchIndex [string first portal $serverName ]

    if {$searchIndex !=-1} {
        #update JVM setting
        set wps [$AdminConfig getid /Cell:cellName/Node:nodeName/Server:WebSphere_Portal/]
        set jvm [$AdminConfig list JavaVirtualMachine $wps]
        $AdminConfig modify $jvm {{ initialHeapSize 1280}}
        $AdminConfig modify $jvm {{ maximumHeapSize 1280}}

        #update session timeout
        set wc [$AdminConfig list WebContainer $wps]
        set sm [$AdminConfig list SessionManager $wc]
        set tp [$AdminConfig list TuningParams $sm]
        $AdminConfig modify $tp {{invalidationTimeout 10}}

        #update ORB service property
        set orb [$AdminConfig list ObjectRequestBroker $wps]
        $AdminConfig modify $orb {{noLocalCopies true}}

    else {    puts "nothing to do for this server"  }
}
```

Save this script as updateWASPerf.jacl, and you can run this script as the following:

```
wsadmin -f updateWASPerf.jacl
```

References are available for writing wsadmin task scripts using either Jython or Jacl to create new scripts that automate WebSphere Application Server changes. The references can be found on WebSphere Infocenter, located at the following Web site:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/txml_reference.html

In addition, to search the online resources for help, you can always engage the IBM Software Services (ISS) team to help you write custom scripts to improve productivity of portal administrators.

A.3.3 Use ANT script to automate portal server operations

The value of ANT allows you to integrate multiple tasks to create a single-entry point of a complex work. Ideally, after you create the ANT task, all you need to do is complete your complex portal deployment operation, similar to the following:

```
ant.bat deploy-portal
```

or

```
ant.sh deploy-portal
```

In this section, we demonstrate how to use ANT to automate complex operation scenarios so that you can create simple, high-level interfaces. Let us use a typical portal update scenario that involves deploying a new portlet war file in to a portal server. At a high level, we can use the top down approach to break up the deployment process into the following three steps:

1. Run XMLAccess to export portlet configuration from source server.
2. Copy portlet WAR file from staging directory to WP/installableApps directory.
3. Run XMLAccess to import the exported portlet configuration xml to target portal.

We will create ANT tasks in the ANT script `itso.deploy.xml` to automate these three steps. First we need to identify the right unit level task to perform each step: Step 1 and 3 are XMLAccess procedures, so we can use `<xmlaccess task>`. Step 2, file copy, is a standard ANT procedure, so we can use `<copy>` task.

After we decide the unit level task, we can start writing the ANT integration script using a bottom up approach: for example, we write the unit level task first, then we integrate them into a high-level task. The benefits of this bottom-up approach of writing ANT script is that you can start testing the script as soon as you write one unit task.

We already demonstrated an XMLAccess task usage in the previous section, so you can follow that example and create the corresponding XML for step 1 and 3.

Step 1: create XMLAccess export task

The XMLAccess export task first loads a `portlet.export.xml` file that specifies resource names that need to be exported. It then connects to the source portal server to execute the export process. The export XML is very similar to our previous example. First you create the ANT xml file `itso.deploy.xml` with the following unit task. You need to update `sourceHost` in the following sample with your actual host name.

```
<?xml version="1.0" encoding="UTF-8"?>
<target name="portal-xml-export" >
  <xmlaccess
    user="wpsadmin"
    password="wpsadmin"
    url="http://sourceHost:10038/wps/config"
    srcfile="c:\portlet.export.xml"
    destfile="c:\portlet.source.xml" />
</target>
```

Here c:\portlet.export.xml is specified as "srcFile". It is modified from the sample portlet export XML from WebSphere Portal infocenter. Update the string portlet.uniquename with your actual portlet war unique name:

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="PortalConfig_1.4.xsd" type="export">
  <portal action="locate">
    <web-app action="export" uniquename="portlet.uniquename"/>
  </portal>
</request>
```

After you create these two XML files, you can unit test your script in ANT to verify that it works. Because here we used the portal custom ANT task <xmlaccess>, you should either use standard ANT.bat/sh with the jars file containing <xmlaccess> task definition or use WPSconfig.bat/sh, which automatically loads the right custom task jars. Because WPSconfig.bat/sh always loads XML under \${WPSinstallLocation}/config/includes directory, the most convenient way to develop and test ANT tasks is to use the WPSconfig command instead of the standard ANT command. This will require you to save the ANT XML file (here it is itso.deploy.xml) under the "includes" directory. For example, \${WPSinstallLocation}/config/includes/itso.deploy.xml

Make sure that the referred export XMLAccess file is stored as c:\portlet.export.xml. Then you can try the task using WPSconfig.bat/sh as such:

```
WPSconfig.bat portal-xml-export
```

WPSconfig always invokes some initialization tasks prior to executing our portal-xml-export task. Our task execution trace string should be seen toward the end of the execution as follows.

```
portal-xml-export:
Mon Nov 06 14:59:55 PST 2006
[xmlaccess] EJPXB0006I: Connecting to URL http://localhost:10038/wps/config
[xmlaccess] EJPXB0004I: Writing output file C:\portlet.source.xml
[xmlaccess] EJPXB0002I: Reading input file C:\portlet.export.xml
[xmlaccess] EJPXB0020I: The request was processed successfully on the server.
```

```
BUILD SUCCESSFUL
Total time: 14 seconds
```

The expected result of portl-xml-export task is the output file portlet.source.xml, which we can use during step 3. You can inspect this file to see whether there is any error.

Now that you have verified that our export script works, let us move on to copy files and do the import.

Step 2: create copy task

In this file copy step, because the portal WAR deployment procedure requires that the WAR file be located under \${srcDir}/installableApps, we need to copy the war file from a staging directory, which is typically pushed by the build process automatically to the default location. The file copying process can be achieved using <copy> task.

```
<target name="copy-war">
  <copy todir="${WpsInstallLocation}/installableApps" overwrite="true">
    <fileset dir="${srcDir}/installableApps"/>
  </copy>
```

```
</target>
```

You can add this to the XML file `itso.deploy.xml`, and do another unit test. We show the second test until after step 3.

Step 3: create XMLAccess import task

Step 1 created an XMLAccess import xml file through the export process. Step 2 placed the WAR file in the right location. Now in step 3, you use the XMLAccess import process to load the generated XMLAccess file from step 1. You do not need to write additional XMLAccess srcfiles. You just specify srcfile with the output file from step1 `c:\portlet.source.xml`. The destfile is not too useful other than for reporting the status of the import. The key difference between step 1 and step 3 is that the URL points to a different server: step 1 uses src server, while step 3 uses tgt server.

```
<target name="portal-xml-import" >
    <xmlaccess
        user="wpsadmin"
        password="wpsadmin"
        url="http://targetHost:10038/wps/config"
        srcfile="c:\portlet.source.xml"
        destfile="c:\portlet.importresult.xml" />
</target>
```

Add the above to `itso.deploy.xml`, and save the change. Now it is the time to do another unit test to verify that this task works or not. To that end, we execute the following:

```
wpsconfig.bat portal-xml-import
```

If your task runs successfully, the command output should read as follows (toward the end of it). The import process takes longer time than export:

```
portal-xml-import:
Mon Nov 06 15:58:11 PST 2006
[xmlaccess] EJPXB0006I: Connecting to URL http://localhost:10038/wps/config
[xmlaccess] EJPXB0004I: Writing output file C:\portlet.importresult.xml
[xmlaccess] EJPXB0002I: Reading input file C:\portlet.source.xml
[xmlaccess] EJPXB0020I: The request was processed successfully on the server.
```

```
BUILD SUCCESSFUL
Total time: 1 minute 25 seconds
```

Now that you have all the individual unit task as your building blocks and verified that they all work, the last step is to integrate them in a single, high level deploy-war task. The following listing shows the completely integrated XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<target name="deploy-war" >
    <antcall target="portal-xml-export">
    </antcall>
    <antcall target="copy-war">
    </antcall>
    <antcall target="portal-xml-import">
    </antcall>
</target>
```

```

<target name="copy-war" >
    <copy todir="${WpsInstallLocation}/installableApps" overwrite="true">
        <fileset dir="${stageDir}/installableApps"/>
    </copy>

</target>

<target name="portal-xml-export" >
    <xmlaccess
        user="wpsadmin"
        password="wpsadmin"
        url="http://sourceHost:10038/wps/config"
        srcfile="portlet.export.xml"
        destfile="portlet.source.xml" />

</target>

<target name="portal-xml-import" >
    <xmlaccess
        user="wpsadmin"
        password="wpsadmin"
        url="http://targetHost:10038/wps/config"
        srcfile="portlet.source.xml"
        destfile="portlet.importresult.xml" />

</target>

```

You can run an integrated task by specifying *deploy-war* as your target name when you run *wpsconfig*. It will always connect to your source portal server, export to a local file, copy the necessary portlet war files, and then import to the local target portal server. All of these tasks are now achievable via a single ANT call.

This example demonstrates a case where ANT scripting can be used to simplify the complex deployment process. You can further customize the script to use variables replacing the hard-coded values in the ANT xml file so that the script is reusable.

What's more, because *wpsconfig* is also the main configuration tool for WebSphere Portal, there are many existing configuration tasks that you can embed in your ANT script so that you do not have to write lots of new ANT tasks. For example, in order to fully synchronize the portal cluster, you can use the following task directly.

```
<antcall target="action-full-sync-nodes"/>
```

For a complete reference to existing configuration tasks that you can invoke directly, you can go to the following Web site:

http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/topic/com.ibm.wp.ent.doc/wpf/inst_best.htm



Portal 6 Update Installer

Appendix B describes the IBM WebSphere Portal update installer application.

B.1 Portal 6 Update Installer

This section describes the IBM WebSphere Portal update installer application that you can use to install interim or cumulative fixes and fix packs on the WebSphere Portal Version 6.0 product. The application can also be used to uninstall fixes.

This section provides examples of the command syntax used by the update installer and a step-by-step example of a fix installation. Please note that for the latest information about this utility, always refer to the fix readme and the Portal Update Installer (PUI) readme files. This section is simply an overview of this utility.

B.1.1 Downloading the Portal Update Installer

Download the PortalUpdateInstaller.zip file that contains the Portal Update Installer application.

<http://www-1.ibm.com/support/docview.wss?rs=899&uid=swg24006942>

The default location from which to unpack the update installer file is the portal_server_root/update directory. Interim fixes can be downloaded to the /update/fixes directory, and fix packs can be downloaded to the portal_server_root/update/fixpacks directory. The location of these directories is arbitrary.

Note: On Microsoft Windows platforms, the pkunzip utility might not decompress the download image correctly. Use another utility (such as WinZip) to unzip the image.

Important: This section is a reference only for the Portal Update Installer. For detailed instructions on using the update installer, always refer to the README file provided with the interim or cumulative fix or fix pack.

B.1.2 Clustering considerations

Installing a fix in a clustered Network Deployment environment requires an administrator to perform the following steps:

1. Shut down one node. The node you choose to install the fix on is arbitrary.
2. Install the fix the same way as a single-server installation
3. Bring up the node on which you just installed the fix.
4. Shut down the next node.
5. Bring up the node on which you just installed the fix.

For special considerations please refer to the fix readme instructions.

B.1.3 Sample fix installation

In the following example we show installation of a Portal 6 WMM fix PK31592:

Following is the link to the fix:

<http://www-1.ibm.com/support/docview.wss?rs=688&uid=swg24013740>

Use the following steps to install the sample fix:

1. Download the update installer tool in order to install this fix. Please see section 2.1.1 for detailed steps on downloading this installer.

Important: The version of PUI used varies based on the version of WebSphere Portal. Ensure the correct version is in use.

2. Create a temporary *fix* directory to store the jar file. In a typical installation you may create this directory under the update directory. Please note that the directory name is arbitrary.
3. Copy the jar file to this *fix* directory. Please note that the download from the IBM support Web site typically comes in the form of a zip file that includes a readme. You must extract the zip file to get access to the jar file.

Important: Go over the readme included for the fix to see if there are special considerations.

4. Shutdown WebSphere Portal.
5. Set up the Java environment for the update installer:
 - a. Open a command line window.
 - b. Change the directory to the app_server_root/bin directory, where app_server_root is the installation directory of the WebSphere Application Server that is associated with WebSphere Portal.

Issue the appropriate command:

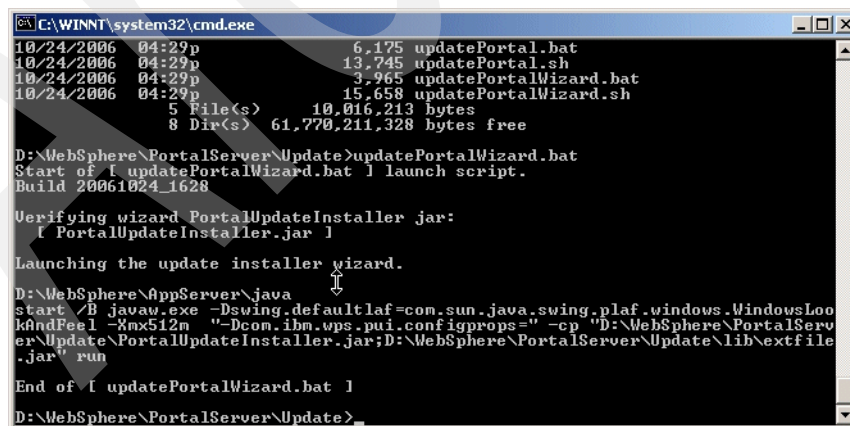
UNIX: `.(space)./setupCmdLine.sh`

Note: When running this command in a UNIX shell, be sure to use the syntax `.(space)./setupCmdLine.sh`. If you do not precede the command with the period and space, the Java environment will not be properly set for the active shell.

Linux: `source setupCmdLine.sh`

Windows: `setupCmdLine.bat`

6. Launch the `updatePortalWizard.bat`. The Portal Update Installer welcome window appears.



```
C:\WINNT\system32\cmd.exe
10/24/2006  04:29p             6,175 updatePortal.bat
10/24/2006  04:29p          13,745 updatePortal.sh
10/24/2006  04:29p           3,965 updatePortalWizard.bat
10/24/2006  04:29p          15,658 updatePortalWizard.sh
          5 File(s)      10,016,213 bytes
          8 Dir(s)  61,770,211,328 bytes free

D:\WebSphere\PortalServer\Update>updatePortalWizard.bat
Start of [ updatePortalWizard.bat ] launch script.
Build 20061024_1628

Verifying wizard PortalUpdateInstaller.jar:
[ PortalUpdateInstaller.jar ]

Launching the update installer wizard.

D:\WebSphere\AppServer\java
start /B javaw.exe -Dswing.defaultlaf=com.sun.java.swing.plaf.windows.WindowsLookAndFeel -Xmx512m -Dcom.ibm.wps.pui.configprops="-cp "D:\WebSphere\PortalServer\Update\PortalUpdateInstaller.jar;D:\WebSphere\PortalServer\Update\lib\extfile.jar" run
End of [ updatePortalWizard.bat ]

D:\WebSphere\PortalServer\Update>
```

Figure B-1 Launch `updatePortalWizard.bat` from DOS command line

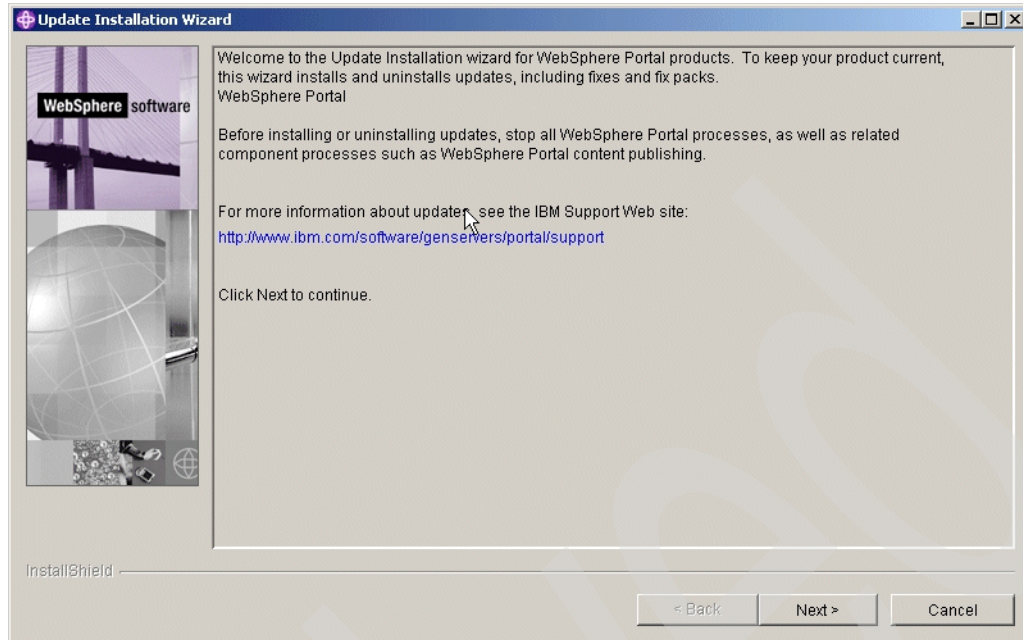


Figure B-2 Welcome window

7. Select your Portal Installation, as shown in Figure B-3.

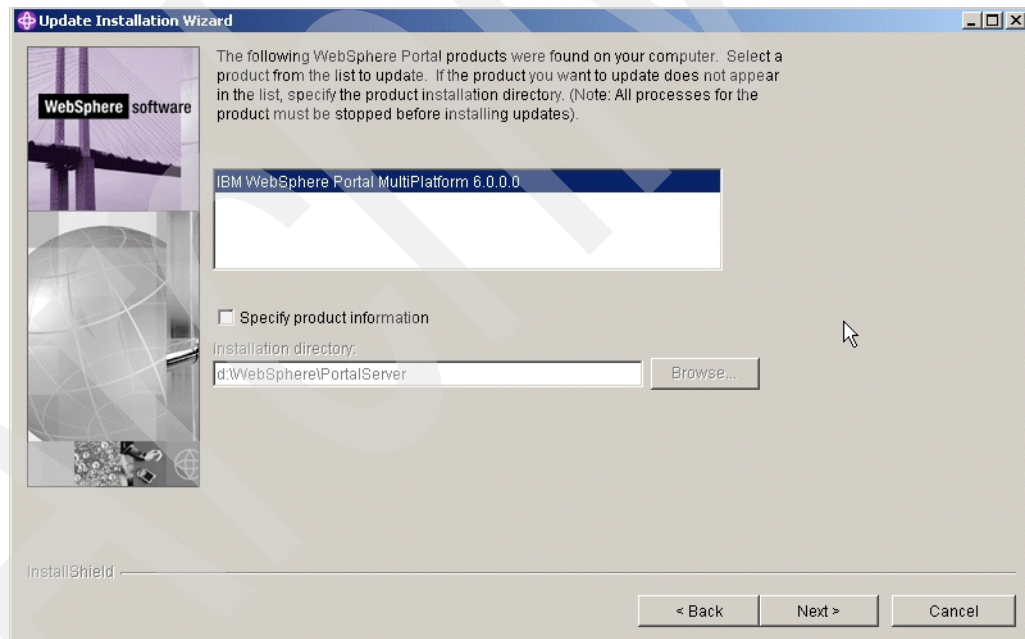


Figure B-3 Portal selection window

8. At this point the Portal Update Installer does a check to make sure all Application Servers are stopped.

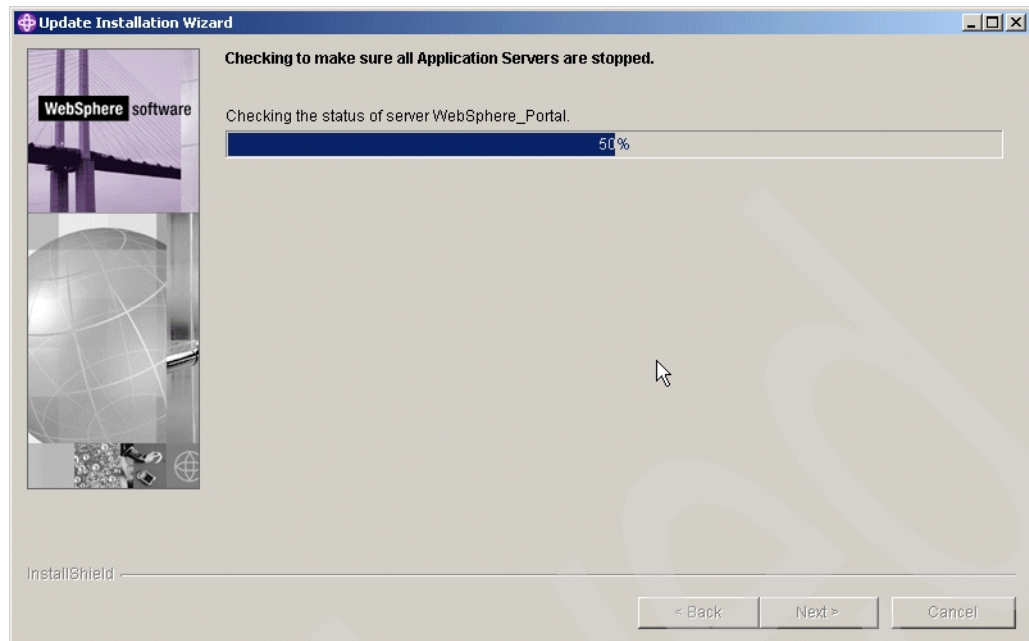


Figure B-4 Application server check window

9. Pick the location of your fix, as shown in Figure B-5.

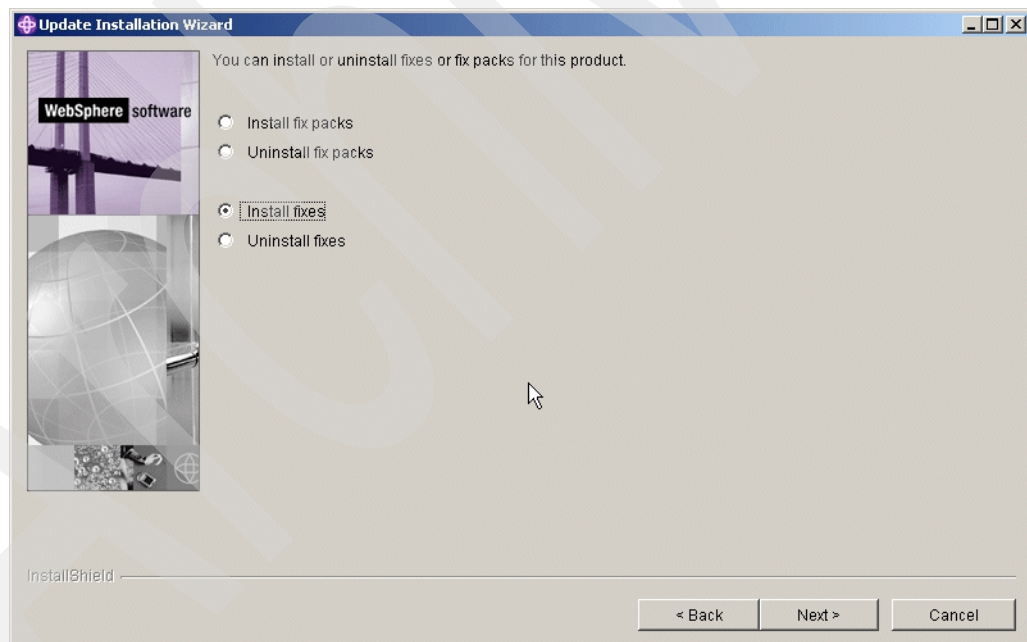


Figure B-5 Install fixes selection window

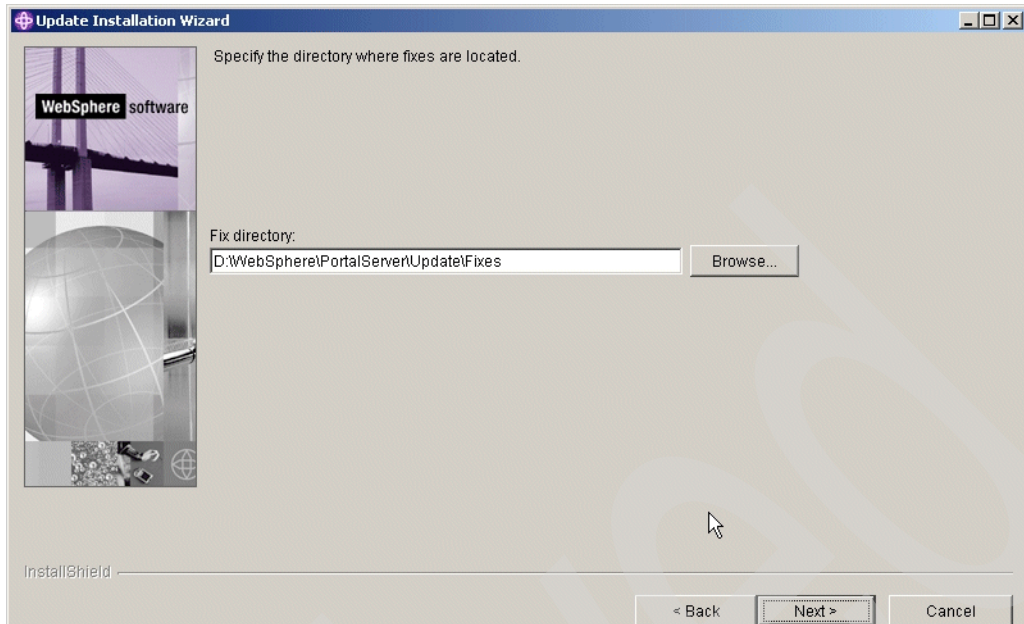


Figure B-6 Fix location window

10. Pick the fix you want to install, as shown in the following figure.

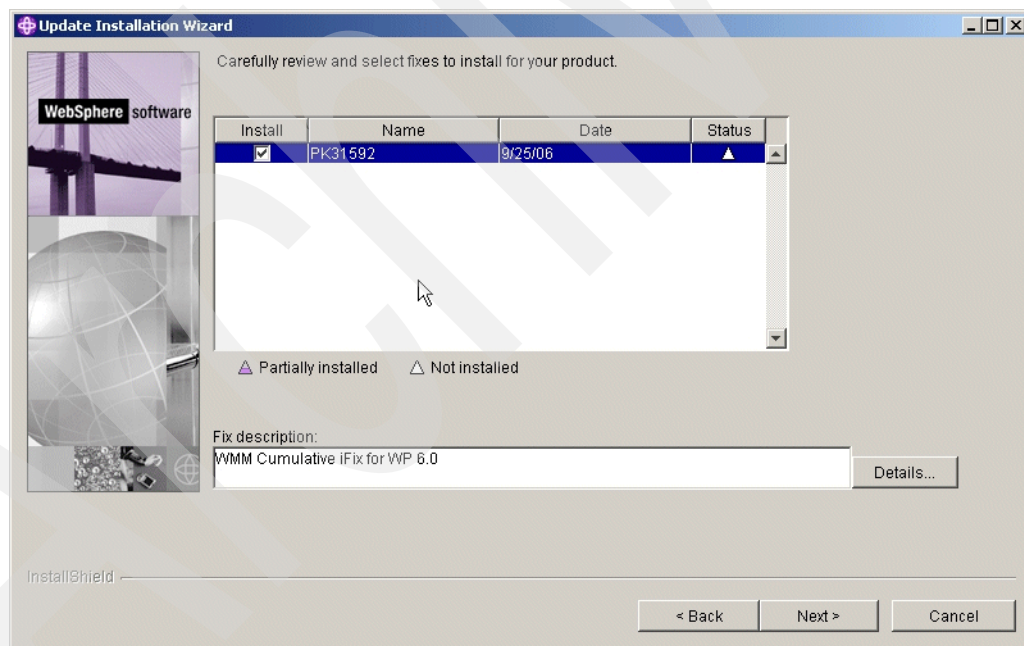


Figure B-7 Select the fix to install

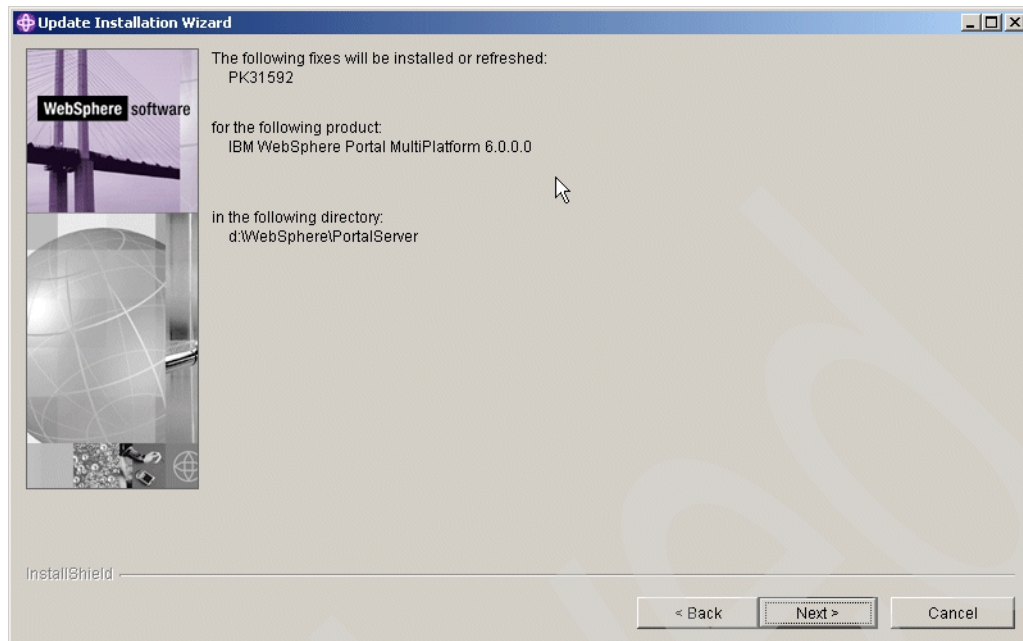


Figure B-8 Fix summary window

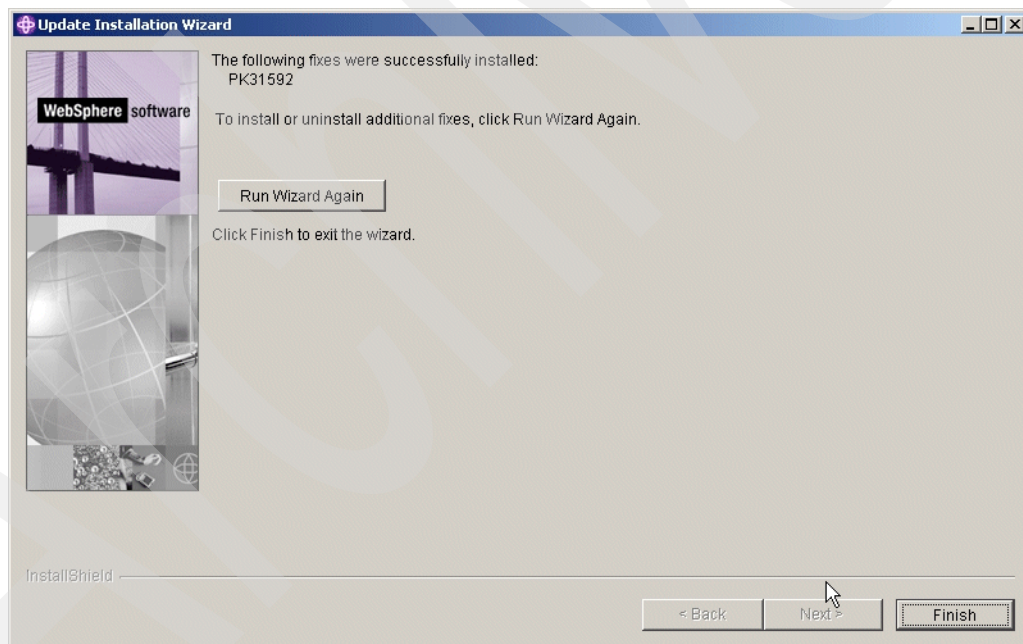


Figure B-9 Fix installed successfully

11. Restart WebSphere Portal.

The temporary directory may be removed.

Un-Installation:

Important: Fixes must be removed in the order they were applied. Do not remove a fix unless all fixes applied after it have first been removed. You may reapply any removed fix.

1. Shutdown WebSphere Portal.
2. Follow the instructions that are packaged with the Portal Update Installer on how to uninstall the fix.
3. Restart WebSphere Portal.

B.1.4 Update Portal Wizard Graphical User Interface overview

This section describes the `updatePortalWizard` command and its command-line parameters. After launching the `updatePortalWizard`, refer to the instructions in the panels for installing or uninstalling fixes and fix packs.

Important: Do not launch multiple copies of the update installer at one time. The update installer cannot be launched concurrently with itself. Performing more than one update at the same time can lead to a failed or faulty installation.

Graphical User Interface command name

To run the Portal Update Installer use the following command and follow the instructions in the panels of the `updatePortalWizard` to install or uninstall fixes and fix packs.

`updatePortalWizard.{sh|bat}`

Note: Follow the instructions in the panels of the `updatePortalWizard` to install or uninstall fixes and fix packs.

Installation roots

`Portal_server_root` represents the directory where WebSphere Portal is installed. By default, this varies per product offering and operating system.

IBM AIX: `/usr/WebSphere/PortalServer`

UNIX and Linux: `/opt/WebSphere/PortalServer`

Windows: `drive:\Program Files\WebSphere\PortalServer`

Space requirements

Space requirements vary depending on what you are installing. The size of each download is available on the Support site. After unpacking the zip file you download, delete the zip file to free space. Space is also required for backup files in the `portal_server_root /version/backup` directory and your system's temp directory.

Prerequisite Java environment setting

Use the following steps to set up the Java environment for the update installer:

1. Open a command line window.
2. Change the directory to the `app_server_root/bin` directory, where `app_server_root` is the installation directory of the WebSphere Application Server that is associated with WebSphere Portal.
3. Issue the appropriate command:

UNIX: `.(space) ./setupCmdLine.sh`

Note: When running this command in a UNIX shell, be sure to use the syntax `.(space)./setupCmdLine.sh`. If you do not precede the command with the period and space, the Java environment will not be properly set for the active shell.

Linux: `source setupCmdLine.sh`

Windows: `setupCmdLine.bat`

B.1.5 updatePortal command line overview

This section describes the `updatePortal` command and its command-line parameters.

Installation Root

`Portal_server_root` represents the directory where WebSphere Portal is installed. By default, this varies per product offering and operating system:

AIX: `/usr/WebSphere/PortalServer`

UNIX and Linux: `/opt/WebSphere/PortalServer`

Windows: `drive:\Program Files\WebSphere\PortalServer`

Command parameters

Given certain command parameters, the `updatePortal` command can:

- ▶ Install or uninstall interim or cumulative fixes or fix packs.
- ▶ Provide information about the update state of applied interim or cumulative fixes or fix packs.

The following table describes the parameters that can be used with the `updatePortal` command. Variables in the parameter column are shown italicized, as follows.

Table B-1 updatePortal parameters

Parameter	Description
<code>-?</code>	Shows command usage.
<code>/?</code>	Shows command usage on Windows platforms only. Not supported for Linux and UNIX-based platforms.

Parameter	Description
-configProperties propertyFile.properties	<p>Specifies an externally supplied properties file containing WebSphere Portal properties and values. Using a properties file enables you to set property values to be used during the installation of the interim or cumulative fix or fix pack. You can also share the properties file across multiple installations of WebSphere Portal, if this is appropriate for your environment. You can use the following properties in this file:</p> <p>Any property in the wpconfig.properties file</p> <p>WasDM. Use this property when applying interim fixes or fix packs in a clustered environment with WebSphere Application Server's Deployment Manager (DMGR).</p> <p>When specifying properties in a file, the following conventions should be observed:</p> <p>Values identified for properties should not have trailing spaces</p> <p>Do not enclose values in quotation marks</p> <p>When typing directories, use a forward slash (/) instead of a backward slash (\), regardless of the operating system used.</p>
-fix	Interim fix only. Identifies the update as an interim fix update.
-fixDetails	Interim fix only. Displays interim fix detail information.
-fixDir	Interim fix only. Specifies the fully qualified directory where you download interim fixes. This directory is usually the portal_server_root/update/fixes directory.
-fixes	Interim fix only. Specifies a list of space-delimited interim fixes to install or uninstall.
-fixJars	Interim fix only. Specifies a list of space-delimited interim fix JAR files to install or uninstall. Each JAR file has one or more interim fixes.
-fixpack	Fix pack only. Identifies the update as a fix pack update.
-fixpackDetails	Fix pack only. Displays fix pack detail information.
-fixpackDir	Fix pack only. Specifies the fully qualified directory where you download and unpack fix packs. By default, this directory is the portal_server_root/update/fixpacks directory.
-fixpackID	<p>Fix pack only. Specifies the ID of a fix pack to install or uninstall. The value you specify does not include the .jar extension. The value is not the fully qualified package file name, but is the name of the individual fix pack within the JAR file. The current WebSphere Portal strategy for fix pack JAR files is to use one JAR file per fix pack. The fix pack ID is the name of the JAR file before the Jar extension. For example:</p> <p>fix pack ID: WP_PTF_601</p> <p>fix pack JAR file name: WP_PTF_601.jar</p> <p>fix pack ZIP file name: WP_PTF_601.zip</p>
-help	Shows command usage.
/help	Shows command usage.
-includeOptional	Fix pack only. Specifies a space-delimited list of features. The installer applies any service for the components, if present in the fix pack. Otherwise, the installer does not apply the service.

Parameter	Description
-install	Installs the update, either interim fix or fixpack.
-installDir	Specifies the fully qualified installation root of the WebSphere Portal product.
-prereqOverride	Interim fix only. Overrides any installation and uninstallation prerequisite checking. The update installer does not log missing prerequisites.
-skipWPCP	Fix pack only. Skips any optional service for WebSphere Portal content publishing that might exist in the fix pack.
-uninstall	Uninstalls the identified fix.
-uninstallAll	Interim fix only. Specifies to uninstall all applied interim fixes.
-usage	Shows command usage.
-wpcplInstallDir	Specifies the fully qualified installation root of WebSphere Portal content publishing.
-wpcpOnly	Fix pack only. Skips the installation of all service except WebSphere Portal content publishing, and applies any service for WebSphere Portal content publishing that might exist in the fix pack. Note: Requires the -wpcplInstallDir parameter.

B.1.5.1 Examples of useful commands with parameters

The following examples assume the followings:

- ▶ The installation root is the C:\Program Files\WebSphere\PortalServer directory.
- ▶ The fix repository is the C:\Program Files\WebSphere\PortalServer\update\fixes directory.
- ▶ The fix pack repository is the C:\Program Files\WebSphere\PortalServer\update\fixpacks directory.

Note: The examples are split onto more than one line for visual clarity, but the command and associated parameters are entered on one line during actual usage.

Installing a collection of interim fixes:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix
    -installDir "C:\Program Files\WebSphere\PortalServer"
    -fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"
    -install
    -fixes Fix1 Fix2
```

Installing a collection of interim fixes and displaying interim fix details:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix
    -installDir "C:\Program Files\WebSphere\PortalServer"
    -fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"
    -install
```

-fixes Fix1 Fix2

-fixDetails

Installing a collection of interim fixes and using a separate properties file:

C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix

-installDir "C:\Program Files\WebSphere\PortalServer"

-fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"

-install

-fixes Fix1 Fix2

-configProperties .\myProp.properties

Installing a collection of fixes and overriding prerequisite checking:

C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix

-installDir "C:\Program Files\WebSphere\PortalServer"

-fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"

-install

-fixes Fix1 Fix2

-prereqOverride

Installing interim fixes from a Java archive (JAR) file:

C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix

-installDir "C:\Program Files\WebSphere\PortalServer"

-fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"

-install

-fixJar Fix1

Installing interim fixes from a Java archive (JAR) file and displaying interim fix details:

C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix

-installDir "C:\Program Files\WebSphere\PortalServer"

-fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"

-install

-fixJar Fix1

-fixDetails

Installing interim fixes from a Java archive (JAR) file and overriding prerequisite checking:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix
    -installDir "C:\Program Files\WebSphere\PortalServer"
    -fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"
    -install
    -fixJar Fix1
    -fixDetails
```

Uninstalling a collection of interim fixes:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix
    -installDir "C:\Program Files\WebSphere\PortalServer"
    -fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"
    -uninstall
    -fixes Fix1 Fix2
```

Uninstalling a collection of interim fixes and displaying interim fix details:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix
    -installDir "C:\Program Files\WebSphere\PortalServer"
    -fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"
    -uninstall
    -fixes Fix1 Fix2
    -fixDetails
```

Uninstalling a collection of interim fixes using a separate properties file:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix
    -installDir "C:\Program Files\WebSphere\PortalServer"
    -fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"
    -uninstall
    -fixes Fix1 Fix2
    -configProperties .\myProp.properties
```

Uninstalling a collection of interim fixes and overriding prerequisite checking:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix
    -installDir "C:\Program Files\WebSphere\PortalServer"
    -fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"
```

```
-uninstall  
-fixes Fix1 Fix2  
-prereqOverride
```

Uninstalling interim fixes in a Java archive (JAR) file:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix  
  
-installDir "C:\Program Files\WebSphere\PortalServer"  
  
-fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"  
  
-uninstall  
  
-fixJar Fix1
```

Uninstalling interim fixes in a Java archive (JAR) file and displaying interim fix details:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix  
  
-installDir "C:\Program Files\WebSphere\PortalServer"  
  
-fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"  
  
-uninstall  
  
-fixJar Fix1  
  
-fixDetails
```

Uninstalling interim fixes in a Java archive (JAR) file and overriding prerequisite checking:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix  
  
-installDir "C:\Program Files\WebSphere\PortalServer"  
  
-fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"  
  
-uninstall  
  
-fixJar Fix1  
  
-fixDetails
```

Viewing a list of installed interim fixes :

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix  
  
-installDir "C:\Program Files\WebSphere\PortalServer"
```

Viewing a list of interim fixes available in the repository:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fix  
  
-installDir "C:\Program Files\WebSphere\PortalServer"  
  
-fixDir "C:\Program Files\WebSphere\PortalServer\update\fixes"
```

Installing a fix pack:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fixpack  
  
-installDir "C:\Program Files\WebSphere\PortalServer"  
  
-fixpackDir "C:\Program Files\WebSphere\PortalServer\update\fixpacks"  
  
-install  
  
-fixpackID Fixpack1
```

Installing a fix pack and displaying fix pack details:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fixpack  
  
-installDir "C:\Program Files\WebSphere\PortalServer"  
  
-fixpackDir "C:\Program Files\WebSphere\PortalServer\update\fixpacks"  
  
-install  
  
-fixpackID Fixpack1  
  
-fixpackDetails
```

Installing a fix pack and using a separate properties file:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fixpack  
  
-installDir "C:\Program Files\WebSphere\PortalServer"  
  
-fixpackDir "C:\Program Files\WebSphere\PortalServer\update\fixpacks"  
  
-install  
  
-fixpackID Fixpack1  
  
-configProperties .\myProp.properties
```

Performing a partial installation of a fix pack by choosing to skip the installation of optional service to the WebSphere Portal content publishing feature:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fixpack  
  
-installDir "C:\Program Files\WebSphere\PortalServer"  
  
-fixpackDir "C:\Program Files\WebSphere\PortalServer\update\fixpacks"  
  
-wpcpInstallDir "C:\Program Files\WebSphere\PortalServer\wpcp"  
  
-install  
  
-fixpackID Fixpack1  
  
-skipWPCP
```

Note: The fix pack status shows partial installation.

Uninstalling a fix pack:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fixpack  
-installDir "C:\Program Files\WebSphere\PortalServer"  
-uninstall  
-fixpackID Fixpack1
```

Uninstalling a fix pack and displaying fix pack details:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fixpack  
-installDir "C:\Program Files\WebSphere\PortalServer"  
-uninstall  
-fixpackID Fixpack1  
-fixpackDetails
```

Uninstalling a fix pack and using a separate properties file:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fixpack  
-installDir "C:\Program Files\WebSphere\PortalServer"  
-uninstall  
-fixpackID Fixpack1  
-configProperties .\myProp.properties
```

Viewing a list of installed fix packs:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fixpack  
-installDir "C:\Program Files\WebSphere\PortalServer"
```

Viewing a list of fix packs available in the repository for the WebSphere Portal product:

```
C:\Program Files\WebSphere\PortalServer\update> updatePortal -fixpack  
-installDir "C:\Program Files\WebSphere\PortalServer"  
-fixpackDir "C:\Program Files\WebSphere\PortalServer\update\fixpacks"
```

Services Oriented Architecture and Portal

Appendix C discusses Services Oriented Architecture (SOA) within the context of Portal.

C.1 Introduction

Service-oriented architecture (SOA) is a concept specifying that an application can be made up from a set of independent but cooperating subsystems or services. Such a framework isolates each service and exposes only the necessary declared interfaces to other services. This not only allows architects to organize and reduce dependencies in their products, but also provides for a tailored mix of services in the deployed environment. This approach can be used to support existing requirements, enterprise application integration, for example, as well as provide a foundation for extending the platform to meet specific business demands, such as rapid solution delivery for e-business.

C.2 SOA Definition

A *service* is a discoverable software resource that has a service description. The service description is available for searching, binding, and invoking by a service consumer. The service description implementation is realized through a service provider who delivers quality of service requirements for the service consumer. Services can be governed by declarative policies.

Any service-oriented architecture contains three roles: a service requestor, a service provider, and a service registry (see Figure A-1 on page 411).

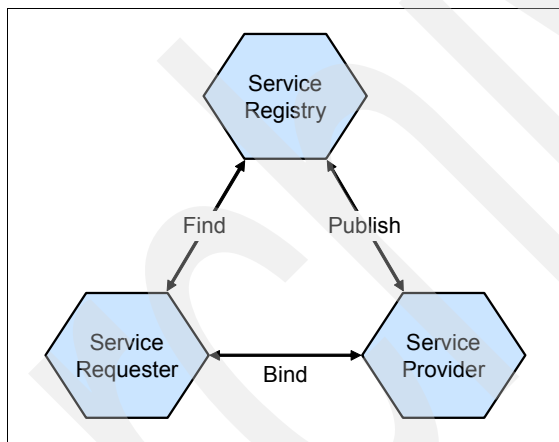


Figure C-1 Three roles contained within a Services Oriented Architecture

- ▶ A **service provider** is responsible for building a useful service, creating a service description for it, publishing that service description to one or more service registries, and receiving service invocation messages from one or more service requestors.
- ▶ A **service requestor** is responsible for finding a service description published to one or more service registries, and for using service descriptions to bind to or invoke services hosted by service providers. Any consumer of a service can be considered a service requestor.
- ▶ The **service registry** is responsible for advertising service descriptions published to it by service providers, and for allowing service requestors to search the collection of service descriptions contained within the service registry. After the service registry makes a match between the service requestor and the service provider, the service registry is no longer needed for the interaction.

Any program or network node can play each of these roles. In certain scenarios a single program might fulfill multiple roles; for example, a program could be a service provider

providing a service to downstream consumers, as well as a service requestor itself consuming services provided by others.

SOA also includes three operations: publish, find, and bind. These operations define the contracts between the SOA roles.

- ▶ The **publish** operation is an act of service registration or service advertisement. It acts as the contract between the service registry and the service provider.
- ▶ With the **find** operation, the service requestor states one or more search criteria, such as type of service, quality of service, and so forth. The result of the find operation is a list of service descriptions that match the find criteria.
- ▶ The **bind** operation embodies the client-server relationship between the service requestor and the service provider. The bind operation can be dynamic (or late), such as on-the-fly generation of a client-side proxy based on the service description used to invoke the service. Alternately, it can be very static (or early), such as a developer hand coding the way a client application invokes a service (that is, during application construction time).

C.3 SOA life cycle

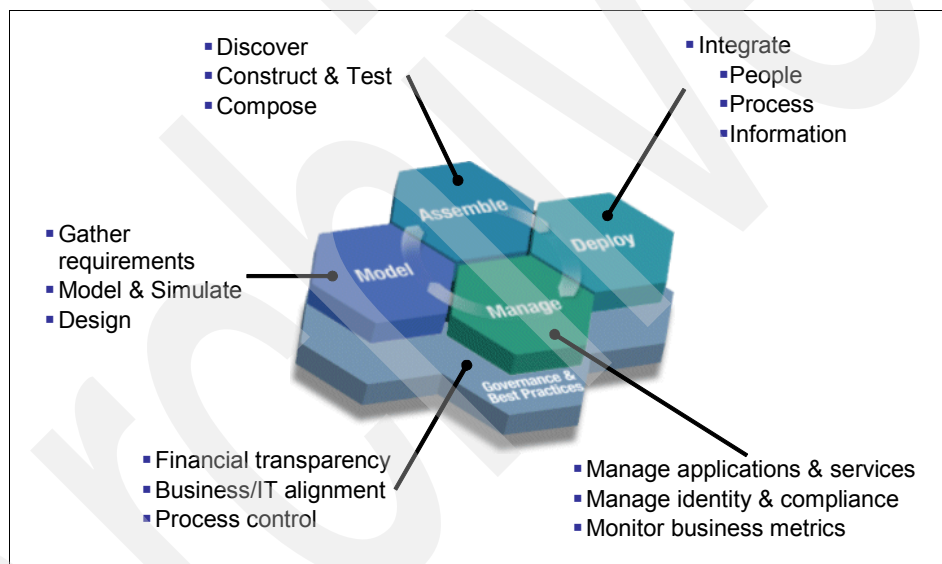


Figure C-2 SOA life cycle

IBM clients indicated that they think about SOA in terms of a life cycle. The life cycle includes the following:

- ▶ Starting the **Model** phase by gathering business requirements and designing their business processes.
- ▶ Optimizing the processes and implementing them by assembling new and existing services to form these business processes.
- ▶ Deploying these assets into a highly secure and integrated services environment.

After the business processes are deployed, IBM clients manage and monitor these business processes from both an IT and a business perspective. Information gathered during the **Manage** phase is fed back into the life cycle to enable continuous process improvement. Underpinning all of these life-cycle stages are governance and processes that provide guidance and oversight for the SOA project.

C.3.0.1 Model phase

You begin the Model phase by gathering and analyzing business requirements that you then use to model, simulate, and optimize your business processes. The resulting business processes are used to design associated software services and service levels to support these processes. During this phase a model is used to establish a common understanding between business and IT of the following:

- ▶ Business processes
- ▶ Objectives
- ▶ Outcomes

The modeling phase helps to ensure that the resulting application meets your defined business requirements. This model also provides a base line from which to measure business performance.

C.3.0.2 Assemble phase

During the Assemble phase, you create services out of existing assets, such as enterprise resource planning (ERP) and financial systems, IBM CICS® applications, and other solutions that run your business. In many cases, you can search a library of existing services for this functionality. If no functionality exists, you can create and test a service to deliver the functionality required for a particular business process. When the required services are available they are orchestrated to implement a business process.

C.3.0.3 Deploy phase

During the Deploy phase, you configure and scale the runtime environment to meet the service levels required by your business processes. After a business process is configured, you can deploy it into a robust, scalable, and highly-secure services environment. This services environment is optimized to reliably run mission-critical business processes while providing the flexibility to make updates in response to changing business requirements. This service-oriented approach also reduces the cost and complexity associated with maintaining numerous point-to-point integrations.

C.3.0.4 Manage phase

The Manage phase involves establishing and maintaining service availability and response times, as well as managing underlying services assets. You can monitor key performance indicators (KPIs) in real time to get the information you need to prevent, isolate, diagnose, and fix problems. Understanding the real-time performance of your business processes enables you to provide vital feedback to the business-process model to enable continuous improvement. This phase also involves managing and maintaining version control over the services that make up your business processes. The management phase ultimately enables you to make better business decisions sooner than previously possible.

C.3.0.5 Governance and processes

Governance and processes are critical to the success of any SOA project. SOA governance is the set of solutions, policies, and practices that enable companies to implement and manage an enterprise SOA. Governance defines the processes for directing and controlling the SOA components in order to achieve the enterprise's goals by adding value, while balancing risk versus return. To help ensure success, you might choose to create a center of excellence within your business to implement governance policies and to follow proven international governance standards. Implementing strong governance policies can result in successful SOA projects-and also improve the potential for higher profits and increased shareholder value.

The governance model defines the following:

- ▶ What has to be done.
- ▶ How it is done.
- ▶ Who has the authority to do it.
- ▶ How it is measured.

C.3.0.6 SOA reference architecture

The SOA reference architecture is a way of looking at the set of services that go into building an SOA. This architecture is not unique to IBM. These are things that you need to consider when approaching SOA regardless of what products and services are used. These capabilities can be implemented on a build-as-you-go basis allowing capabilities and project level solutions to be easily added as new requirements are addressed over time. You can see that these services organized along the same life cycle that we discussed. On the left is Development Services which is model and assemble, in the middle are the elements of the deployment run-time environment you use, and on the right is management. The backbone of the reference architecture is the enterprise service bus that facilitates communication between services. The SOA reference architecture outlines the key capabilities that are required for comprehensive, enterprise-wide SOA solutions.

The reference architecture is a great tool for laying out road maps for pursuing SOA. Regardless of what kind of project you undertake, it makes sense to lay it out on a reference architecture to see how the various services you design are going to interact with each other.

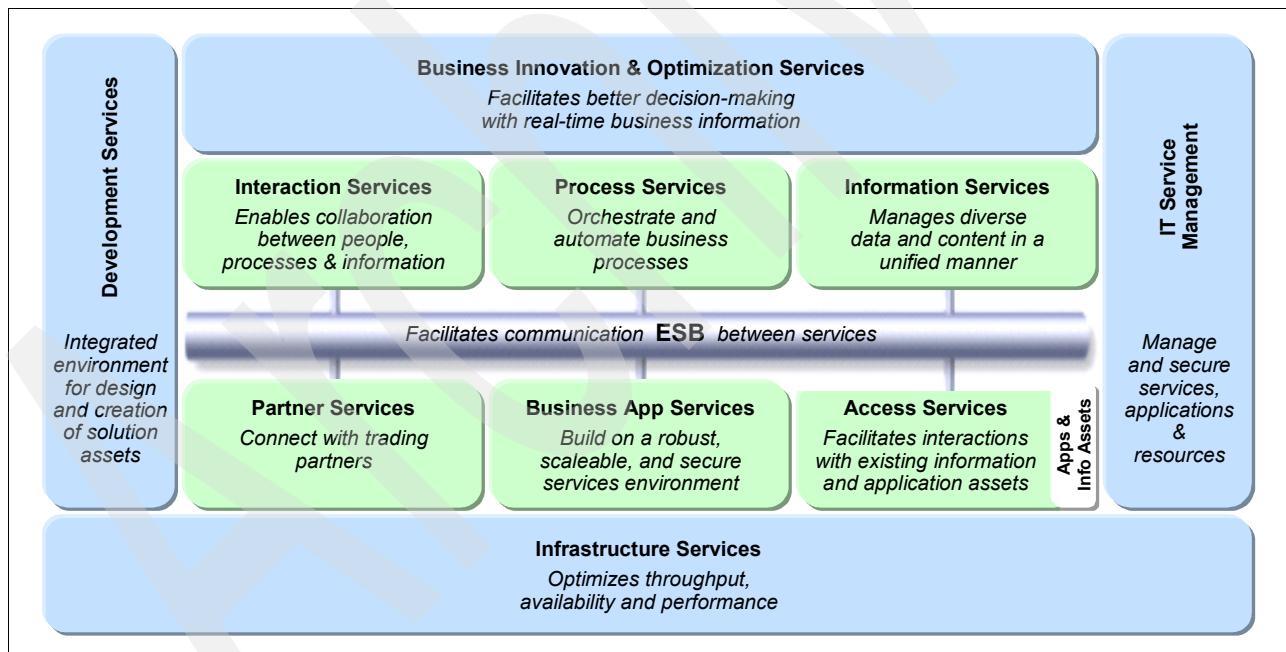


Figure C-3 SOA reference architecture

Tools are an essential component of any comprehensive integration architecture. The SOA Architecture includes both Development Services, which are used to implement custom artifacts that leverage the infrastructure capabilities, and Business Innovation & Optimization Services, which are used to monitor and manage the runtime implementations at both the IT and business process levels.

At the core of the SOA reference architecture is the Enterprise Service Bus (ESB). This delivers all of the inter-connectivity capabilities required to leverage the services implemented

across the entire architecture. Transport services, event services, and mediation services are all provided through the ESB.

The following set of services are oriented toward the integration of people, processes, and information, and are part of the SOA reference architecture:

- ▶ Interaction Services provide the capabilities required to deliver IT functions and data to end users, meeting the end-user's specific usage preferences.
- ▶ Process Services provide the control services required to manage the flow and interactions of multiple services in ways that implement business processes.
- ▶ Information Services provide the capabilities required to federate, replicate, and transform data sources that may be implemented in a variety of ways.

Many of the services in an SOA are provided through existing applications; however, others are provided in newly implemented components, and others are provided through external connections to third party systems.

Existing enterprise applications and enterprise data are accessible from the ESB through a set of Access Services that provide the bridging capabilities between traditional applications, pre-packaged applications, enterprise data stores, and the ESB.

The SOA reference architecture also contains a set of Partner Services that provide the document, protocol, and partner management capabilities required for business processes that involve inter-actions with outside partners and suppliers.

Business Application Services provide runtime services required for new application components to be included in the integrated system.

Underlying all these capabilities of the SOA reference architecture is a set of Infrastructure Services that optimize throughput, availability, and performance.

IT Services Management Services include capabilities that relate to scale and performance, for example edge services, clustering services, and virtualization capabilities allow efficient use of computing resources based on load patterns.

The SOA Reference Architecture is a complete and comprehensive architecture that covers all the integration needs of an enterprise. Its services are well integrated and are delivered in a modular way, allowing SOA implementations to start at a small project level. As each additional project is addressed, new functions can be easily added, incrementally enhancing the scope of integration across the enterprise.

C.3.0.7 Entry points to SOA

Determining where to start can be challenging. Approaching today's critical needs with strategic business goals in mind is what SOA is all about. SOA entry points help businesses pursue SOA incrementally.

They take a project-based approach so each project can deliver real business value. IBM has found that many companies approach SOA from entry points of integrating people, processes, information, or a combination of all three. IBM offers focused software, services, and expertise to help utilize these entry points effectively. SOA entry points provide a business-centric and IT-centric approach to the life cycle. This enables companies to start an SOA project that targets specific business needs to help achieve benefits more rapidly.

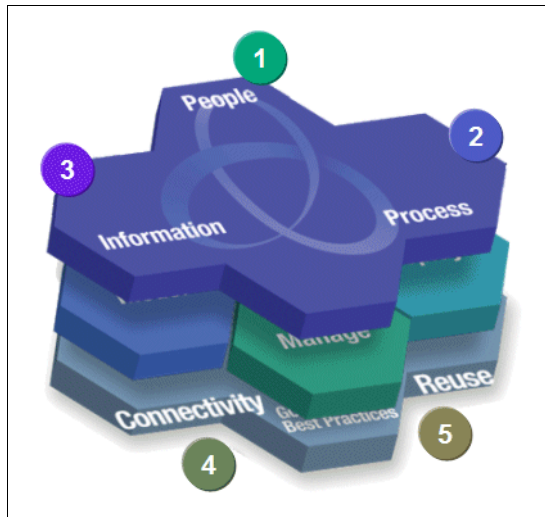


Figure C-4 SOA entry points

C.3.1 People entry point

People-centric collaboration is about enabling human and process interaction with consistent levels of service. Companies that focus on collaboration among people improve their productivity by giving their employees and trading partners the ability to create a personalized and consolidated way to interact with other people and information in the context of business processes. They need a role-based, intuitive, and adaptive user experience. SOA enables this user experience and ensures consistent levels of service are met each step of the way. SOA allows customers to create, deploy, and update composite applications faster with SOA portlets. As application portlets are created or converted to *service-based* portlets, you can aggregate and integrate these portlets faster and more economically to get the collaborative user interface and view into your business that you need.

C.3.2 Process entry point

Companies are interested in the ability to optimize their processes, deploy them on-the-fly, and monitor how effective the enhanced processes are. This faculty drives the ability to create and deploy innovative business models. In order to do this effectively, companies should consider how to *reuse* the components that make up these business processes so that they can be reconfigured quickly and economically (rather than created from scratch for every new project) in response to rapidly changing opportunities and threats. For many years companies wanted to fix, improve, and modify process activity. But IT has been the biggest inhibitor. SOA changes the game in this space by allowing processes to be represented as services and components. As such, the service-based process now becomes re-usable, flexible, and can be modified and re-deployed more quickly. SOA turns processes into parts, called *services* so the specific part of the process can be improved, optimized, and then reassembled with other parts of the process

C.3.3 Information entry point

Information delivered as a service provides extensive options to a company to allow for packaging, transformation, and distribution of the information to the right people at the right time. Information as a service, as a part of an SOA does the following:

- ▶ Ensures consistency in sources and data rules
- ▶ Aligns information with business processes

- Provides business context to information
- Uncovers insightful relationships hidden within information
- Provides a basis for trust in information
- Enables tighter control over information

By implementing the information entry point, SOA lowers costs by supporting a single federated version of the information. By making existing business functions available to new users or channels, new business opportunities are created that extend the value of existing assets through reuse.

C.3.4 Connectivity and reuse entry point

Each of the three entry points discussed is dependent on connectivity that allows services to communicate and interact with other elements of people, processes, and information. Connectivity also allows a SOA implementation to connect to existing applications, partners and customers, as well as the new services that are being created to be shared across the enterprise. In order to get the value from the business-focused entry points, one needs an IT infrastructure that delivers this foundation of connectivity, and that enables one to easily create & reuse services. These are capabilities that need to be considered before moving on to the more advanced states. Also taken into consideration is to build security into every SOA project and to adhere to established SOA governance best practices, ensure oversight and control for the SOA projects.

C.3.5 People entry point

To enable efficient real-time decision making, people need to easily interact and collaborate with other people. They also need to have instant access to information and data from multiple sources. SOA can help people do all of these things easily—from within the context of their business processes. Companies that focus on people as the entry point to SOA are taking a business-centric approach. This approach maximizes people productivity to enhance business results. IBM Workplace, Portal, and Collaboration software from the Lotus family of products enables the people entry point. It delivers the essential people-focused capabilities of the IBM SOA strategy and provides significant value to organizations large and small worldwide. To make use of this entry point, many businesses start at the front end of SOA using IBM WebSphere Portal.

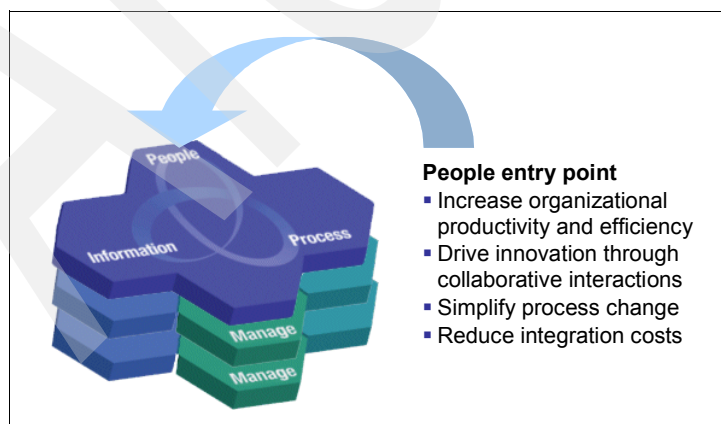


Figure C-5 People entry point

WebSphere Portal allows one to build a view of a key business process by aggregating information in a way that makes sense for people. This creates an intuitive, role-based user

experience-to help people make better decisions. After the portal foundation of SOA is established, businesses can easily expand the portal environment and extend people productivity with other Lotus software for SOA. This software can range from pre-built collaborative capabilities and composite applications to alert-driven dashboards, from products that help accelerate composite application development to products that enable people to access those applications from mobile devices. Determining the next step usually depends on whether the objective is to achieve immediate business goals, or the objective may be to position your business to extend the SOA application environment to respond to changing businesses needs in the future.

C.4 Composite applications

Portals have evolved over recent years as the needs of businesses have become more demanding: starting with personalized access to content and applications, and then to deeper integration with back-end systems. When one considers how applications have evolved within most organizations it is very easy to see a common trend where multiple silos predominate and result in a confusing interface with its own site structure and layout that users must learn to navigate. These are usually designed by lines of businesses within an organization to expose the departmental model and procedures to users. The context of the user is totally lost as this is multiplied across the number of different sites that the user needs to access. Unfortunately this undermines the success of the systems as it places a significant usability challenge on the user. This results in lack of productivity and lack of adoption. Figure C-6 identifies an example of what this scenario looks like.

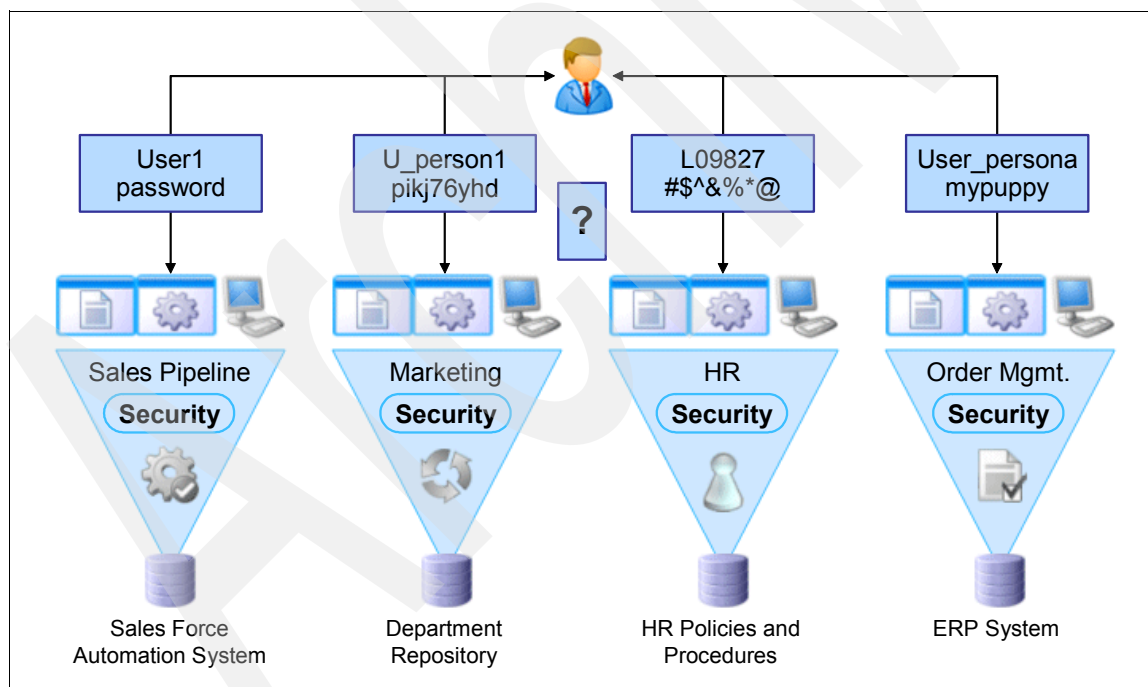


Figure C-6 Multiple interfaces for applications defined by lines of business

The right approach to solving this problem would be to design a single interface with the user in mind. Understanding the audience means providing the right context within the interface or application and delivering the right content, applications, and processes based on the user's role. This approach reduces the burden on the user by hiding the underlying complexity. It adapts to the user's needs and delivers a composite view that allows the user to interact with these multiple back-end systems as though they were one. By consolidating access to the

content, applications, and processes relevant to the user—all in one view—the common simplifies the users tasks and provides audience value, which results in improved productivity and adoption of the portal.

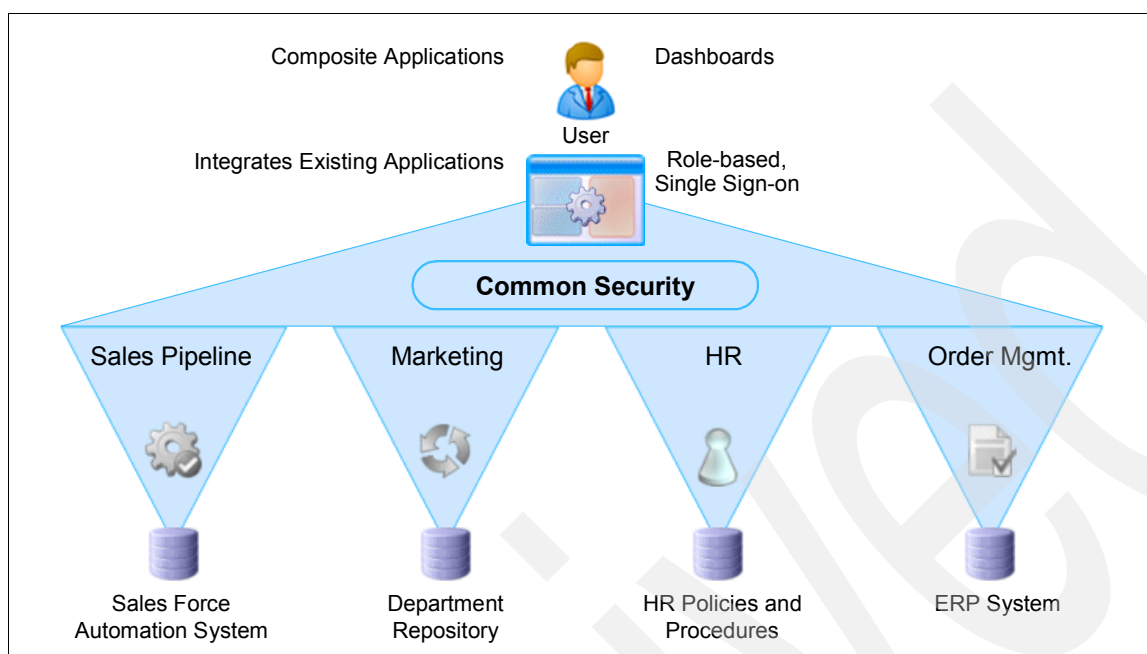


Figure C-7 Single interface for applications defined by lines of business

The original Composite Portal Pattern, borrowed from the e-Business Pattern library, allows for combining information and applications from multiple sources on the page. Although combining multiple applications and information is a good concept, it is imperative that reusability be addressed for composite applications to be successful. Although, the IBM best practice recommendation is to design and implement solutions that are reusable, in reality, many customers find it very difficult to implement a reusable solution that attempts to conform already implemented solutions to standards. Technically this might be a good practice but in reality most customers do not see business value in implementing this approach.

Hence, a more sophisticated approach along the same lines is a composite application. In this case, we assume that the user has to interact with a number of applications simultaneously and by providing a separate layer whose role it is to render a coherent user interface, the user can access a number of disparate back-end applications or services. Although the single access pane is implemented as an application, it allows for other items to be *plugged in* and separates the user interface completely from the different back-end systems. A composite application is often viewed as a key part of a service-oriented architecture where new applications are composed of existing services. This requires, however, a solid services infrastructure and a good handle on how to expose your application's business functions as services. Figure C-8 on page 451 illustrates the single interface for a composite application.

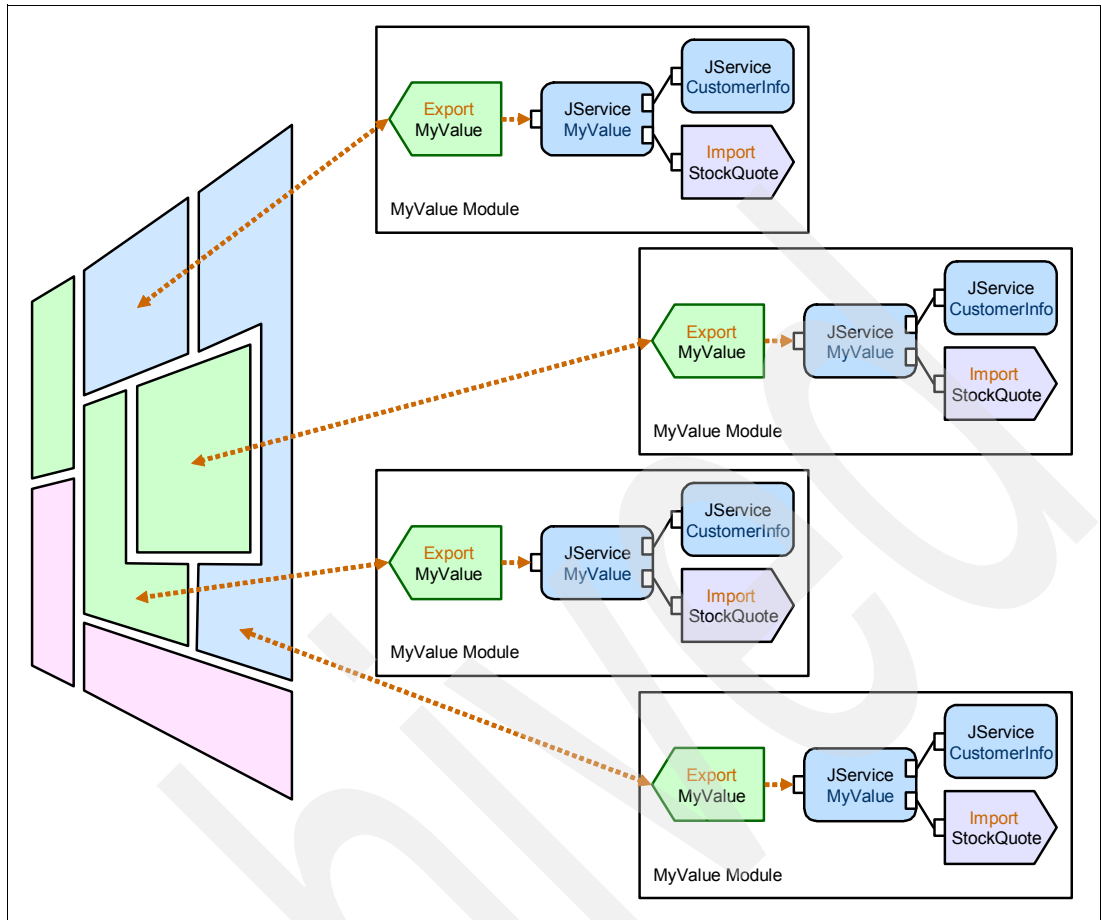


Figure C-8 Composite application

The following are the principles of composite applications:

- ▶ Composite applications are more than just combining applications on a page.
- ▶ The goal is to compose applications, and not to develop applications.
- ▶ Composite applications should work together and be aware of one another even if developed independently.
- ▶ Dashboards, collaboration, and industry solutions can achieve some of this goal but are fairly limited in flexibility.
- ▶ Custom solutions require good design and strong standards to ensure interoperability between components.

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 453. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *WebSphere Portal V5.0 Production Deployment and Operations Guide*, SG24-6391
- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392

Online resources

These Web sites are also relevant as further information sources:

- ▶ IBM WebSphere Portal, Version 6.0 Information CenterLink:
<http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp>
- ▶ WebSphere Portal Documentation
<http://www-128.ibm.com/developerworks/websphere/zones/portal/proddoc.html>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Index

A

- A high availability multi cluster Portal environment 318
- A simple example of templates and applications 8
- A single LDAP directory with multiple realms 16
- A single LDAP directory with multiple virtual portals and realms in WebSphere Portal Server 5.1 190
- A step by step guide to moving a release between environments 349
- Above is a diagram that illustrates a Configuration Split 32
- access control list 343
- active configuration 37–38
- Active Directory Application Mode (ADAM) 26
- adapterclassname for additional LDAP repository 221
- Add Node 2 and subsequent Nodes to the Cluster Definition 160
- Add Node2 and subsequent nodes to the cluster definition 119
- Adding a Horizontal Node to a Cluster (with WebSphere Process Server) 168
- Adding a Horizontal Node to a Cluster (without WebSphere Process Server) 172
- Adding a value the selected attribute must have to apply the current role 12
- Adding a vertical cluster member on the primary node 177, 182
- Adding additional realms in the wwmur.xml file 225
- Adding and additional LDAP repository in wmm.xml 219
- Adding the Federation Repository configuration for Cloudscape 217
- Adding the Federation Repository configuration for Oracle 218
- Adding the short name for the WebSphere Application Server Administrator 214
- AdminConfig getid 419–420
- Advanced LDAP Configuration 209
- Advantages of Database Domains 238
- Advantages of multiple LDAP directories 188
- All database domains stored in one database called “WPSDB” 244
- allowed-skin skin 353–354
- Analysis and comparison to baseline 331
- Analyze a Portal crash 394
- Analyze Failures during Portal Runtime 398
- Analyze Failures with Configuration Tasks 369
- Analyze Failures with Portal Installation 386
- Analyze Failures with the Portal Update Installer 388
- Analyze wpinstalllog.txt 375
- ANT 339
- ANT task 409–410
- antcall target 423
- app server 414–415
- Application roles 9
- Application Server
 - need 408

- application server 25, 27, 409, 412, 428
 - identical copies 29
- Application Server Update Installer welcome window 66
- Applications 8
- Apply base tuning parameters 324
- Assign Custom Unique Names 351
- Assign unique user login per virtual user 327
- Asynchronous JavaScript® and XML (AJAX) 6
- AutoSync Disabled 84
- Availability Gold Standard 37–38

B

- Backing up of user data 240
- Backup of user data 241
- base model 13
- batch processing interface 341
- Before you begin configuring multiple LDAP directories 192
- Build the release 355
- build-as-you-go basis 445
- Building a multi cluster high availability portal environment 313
- Building scripts 327
- Building the differential release file 346
- business process 1, 3, 30, 443–444
 - integration support 29
 - level 445
 - management 29–30
- Business Process Choreographer configured 384

C

- Catalog Databases command 95, 119, 161, 168, 172
- Check Cluster Definition 109, 122, 160, 165
- Choose the AppServer Directory name window 66
- Classpath Settings 156
- CLI mode 416
- Cluster members view 177, 182
- Cluster properties in wpconfig.properties 108, 120, 159
- cluster, elaborated 36
- Clustering Considerations for High Availability 29
- Clusterproperties in wpconfig.properties 164, 170, 173
- command line 19, 27, 341, 343, 410, 412
- Commonly used Environments 40
- Community DB 31
- Composing business logic from components 7
- composite application 4, 6–7, 447, 449
 - abstract definition 7
 - new instance 8
- Composite applications and templates 6
- Conclusion 373, 385, 388, 393, 397, 400
- Config wizard welcome page 203
- configuration
 - active and passive 37–38

- configuration archive 415–416
 - defaultprofile.car 417
 - location 417
- Configuration maintenance tasks 358
- Configuration Management 41–42, 337–338
 - Moving between environments 337
- Configuration management - Step Four 357
- Configuration management - Step One 349
- Configuration management - Step Three 356
- Configuration management - Step Two 355
- Configure Portal Nodes and the DMGR for LDAP security with Realm Support 122, 166
- Configure Portal to use a remote Web server 131
- Configure Portal to use remote Web Servers 166
- Configure Primary Node 1 to an external database 152
- Configure Primary Node1 to an external database 93
- Configure the database using the ConfigWizard 95
- Configuring an additional LDAP directory and enabling realms 214
- Configuring multiple LDAP directories 200
- Configuring portal servers in a cluster 19
- Configuring supported entry types for an additional Domino LDAP directory 222
- Confirmation window 68
- Connecting our Horizontal Cluster to DB2 and Oracle 296
- Connecting our Vertical Cluster to DB2 to and Oracle 295
- Connecting Portal Node 2 to the Database servers 316
- conntype None 413, 416
- Content & Web content management 15
- content trees 339
- control system 339, 347
 - consistent release 339
- Copy ExportRelease.xml 350
- Copy the files to target server 357
- Create cluster definition 153
- Create transfer directory 357
- Creating a composite application with the Template Builder 8
- Creating a hierarchical group in the Domino LDAP directory 229
- Creating a new rule for a specific portlet 10
- Creating a new rule with the Personalization Picker 10
- Creating a profile with the LDAP browser tool 194
- Creating a virtual portal for customers 226
- Creating a virtual portal for employees 226
- Creating two virtual portals to test multiple LDAP directories 225
- Creation of the cluster definition 107
- Current location of our Database Domains 280
- Currently supported LDAP directories 200
- custom task
 - definition 412
 - definition XML 412
- Custom Unique Names 340
- Customers in the Domino LDAP Directory 216

D

- Database Domain 17–18, 44–45, 358

- Horizontal cluster 45–46
- horizontal cluster 44
- Vertical cluster 47–48
- Database Domain (database split)Overview 30
- Database Domains 17–18, 233
- Database properties 100–104
- Database tuning 326
- databaseType for the Member Manager database 218
- db client 35
- DB Settings in wpconfig_dbdomain.properties 97
- DB Settings in wpconfig_dbtype.properties 96
- DB2 command line processor 94, 119, 160, 168, 172
- DB2_JDBC_Driver_Classpath variable view 106
- Default realm with multiple LDAP directories (one to many) 198
- Defining our terminology 235
- Definition of a configuration task failure 368
- Definition of a Portal Runtime failure 397
- Definition of a Portal upgrade failure 388
- Definition of a WebSphere Portal crash 393
- Definition of an installation failure 373
- Delayed cleanup of deleted portal resources 361
- Deploy applications one at a time and retest 330
- Deploy themes and skins 357
- Deploying themes and skins 362
- Deployment and build process 339
- Deployment Manager
 - master configuration 364
- deployment manager 29, 33, 361, 363, 434
- Deployment manager welcome/login window 70
- deployment process 408, 411
- Deployment Scenario 21, 27
- deploy-page task 410
- DeployTheme.xml 351
- Detected WebSphere Application Server 74
- Development/Unit Test (DEV) 41
- Different Deployment Scenarios
 - Building Clustered Environments 51
- Disabling security 203
- Disabling security with the Config Wizard 204
- Disabling WebSphere Application Server security 202
- DMGR started 69
- Do not use hardcoded URLs in your scripts 328
- Domino LDAP & groups 227
- Domino LDAP groups under the Root DSE 228
- Drag and Drop
 - Adding portlets to a page using the Portlet Palette. 6
- Drag, drop and rearrange portlets 6
- Duplicate UID's across multiple LDAP directories 202

E

- Editing the default realm in the wmmur.xml file 224
- Editing the wmmWasAdmin.xml 213
- Editing the wpconfig_dbdomain.properties for the Customization Domain 264, 299
- Editing wpconfig_dbdomain.properties for DB2 and Oracle 280
- Editing wpconfig_dbdomain.properties for JCR and Like-minds 248
- Editing wpconfig_dbdomain.properties for the JCR Data-

- base Domain 298
- Editing wpconfig_dbtype.properties for both DB2 and Oracle 279
- Editing wpconfig_dbtype.properties for DB2 246
- Editing wpconfig_dbtype.properties for Oracle 262, 297
- Editing wpconfig_sourceDB.properties for DB2 263
- Editing wpconfig_sourceDB.properties for the JCR and Likeminds Domains 247
- elaborated cluster 36
- Employees in the IBM Tivoli Directory Server 216
- Enabling horizontal partitioning in wmm.xml 217
- Enabling multiple LDAP directories in a cluster 230
- Enabling security with realm support 207
- Enabling security with realms 213
- Enabling verbose Garbage Collection in WP 6.0. 333
- Enabling WebSphere Application Server Security 384
- Enter basic cluster member information 178, 183
- Enter the WebSphere Application Server Administrator user name and password 204
- Enterprise Applications view in the DM AdminConsole 133
- Enterprise JavaBeans 339
- Establish baseline performance 330
- Establish consistent hardware/software setup 324
- Example 1, Moving Database Domains 243
- Example 1, Portal connected to two DB2 servers 260
- Example 2, Moving Domains between database types 259
- Example 2, Moving the remaining Database Domains to Oracle 261
- Example 2, Portal using DB2 and Oracle 276
- Example 2, Retiring the original DB2 server and portal database 261
- Example 3 - Sharing Domains amongst Portal Nodes 275
- Example 3, Multiple portals sharing the same Database Domains 277
- Example 3, Sharing Database Domains amongst portal servers 292
- Example 4 - Sharing Domains between separate clusters 293
- Example architectures in operation 36
- Example UniqueNames 351
- Examples of moving and sharing Database Domains 241
- Executing your test 330
- export
 - custom 345
 - sample page 344
- Export new release from the source server 355
- Export your current production configuration 348
- Exporting 345
- Exporting a sample page using XMLAccess 344
- Exporting and importing a new page 345
- exporting via the XMLAccess tool 342
- ExportThemesAndSkins.xml 351

F

- failover 28, 33
- Failover and lost data 34
- Failure to activate portlets after using XMLAccess to de-

- ploy portlets in a cluster. 365
- Familiarizing yourself with the LDAP infrastructure 192
- Fault tolerant Portal cluster (with full redundancy and security) 36
- Fault tolerant Portal cluster (with full redundancy) 36
- Fault tolerant Portal cluster illustrates how this Portal architecture avoids a single-point-of-failure and makes use of caching 36
- Fault tolerant Portal cluster illustrates how this Portal architecture avoids a single-point-of-failure and makes use of caching and enhanced security 37
- Federation 76
- Federation screen 114
- file.xml 343
- final environment 48
- Final rule. Only display the portlet if the user's hobby is "sports" 12
- Final test should reflect user behavior 335
- First time login or repeating user login 328
- fix pack 426, 434
 - detail 439–440
 - Id 434
 - JAR file 434
 - partial installation 439
 - repository 435
 - status 439
 - update 434
- fix PK13784 25
- Fix Selection screen 67
- Fix successfully installed window 68
- fixJar Fix1 436, 438
- fixpackID Fixpack1 439
- fixpacks 426, 434
- Flow Chart about Installation Steps
 - Step 2 152
- Flow Chart about Installationssteps
 - Step 4 166
- Flow Chart about the Installation Steps
 - Step 1 138
- Flow Chart for Installation and Configuration Steps 137
- Flow Chart Installation Steps
 - Step 1 56
 - Step 2 93
 - Step 3 107
 - Step 4 122
- Flow Chart of Installation Steps
 - Step 3 153
- Flow Chart of the Installation Steps Overview 55
- Free LDAP browser tool used in our testing 195
- Functional test of a new installation 322

G

- Gather required files 354
- Global Deployment 240
- GUI 354–355, 412, 416

H

- Handling orphan data 358
- Hardware Requirements 22

- Hardware requirements for High Availability 35
- Helper file location screen 125, 205
- Horizontal cluster 28, 44
- Horizontal Cluster (includes WebSphere Process Server)-
Installation and Configuration steps 54
- Horizontal Cluster (without WebSphere Process Server)-
Installation and Configuration steps 136
- Horizontal cluster with database domain 44–45
- Horizontal cluster with database domain and multiple
LDAP 45–46
- Horizontal cluster with single database and single LDAP
44
- Horizontal Cluster with WebSphere Process Server built
in Chapter 3.2 54
- Horizontal Scaling topology 28
- Host Aliases view 179, 184
- host name 27
- How is portal data organized? 236
- How our terminology maps to DB2 236
- How realms have been extended in WebSphere Portal
Version 6 190
- How XMLAccess works 343
- HP-UX system 22
- HTML pages 339
- HTTP
 - connection 343
- HTTP Inbound Channel 82
- HTTP Session 33
- HTTP Session Failover 33
- Hypertext Markup Language (HTML) 339

I

- IBM Lotus
 - Domino 25
- IBM Lotus Domino
 - 6.5.4 26
 - 6.5.5I 26
 - 7.0.1 26
- IBM Tivoli Directory Server
 - V5.2 26
 - V6.0 26
- IBM WebSphere Portal
 - 6 22
 - Development 2
- IBM Workplace Web Content Management Properties
211
- IBM's Serviceability Tools Strategy 402
- Illustrates the WebSphere Application Server clustering
capabilities. You can have an arbitrary number of nodes.
This diagram shows three WebSphere Portal nodes. 28
- import
 - custom page 345
 - via the XMLAccess tool 342
- import process 342, 423
- Import release 357
- Import the differential release file into Production 347
- Import the release into the target server 358
- Import your production configuration into the staging envi-
ronment 348
- Important Installation Consideration with WebSphere Pro-

- cess Server 53
- Importing 345
- Improper path to the portlet WAR file 364
- Initial release to production 346
- Install Portal onto the managed node, Node1 84
- Install Portal onto the managed node, Node2 and subse-
quent nodes 117
- Installation and upgrade WSAS and WPS on the Deploy-
ment Manager (DMGR). 56
- Installation Completed 91, 141
- Installation Directory 72
- Installation directory window 57, 139
- Installation failure 386
- Installation is Complete 73, 111
- Installation is complete window 59–60
- Installation is complete with profile wizard launch option
screen 112
- Installation of WebSphere Portal on Node2 and subse-
quent nodes 160
- Installation of WebSphere Portal on the Primary Cluster
Node, Node 1 146
- Installation of WSAS and WPS on future cluster node,
Node2 and subsequent Nodes 109
- Installation of WSAS and WPS on the Primary cluster
node, Node1 70
- Installation Steps 55, 137
- Installation Summary 58, 140
- Installation Type 86
- Installation type screen 147
- Installation WSAS on the Deployment Manager 138
- Integration Testing (UAT) 41
- Interactive form 5
- Interactive forms in portal interfaces 5
- Interim fix 426, 434
- interim fix
 - detail 435–436
 - update 434
- Introduction 1, 52
- Introduction to the deployment scenarios used in this Red-
book 44
- Isolating database domains by putting them in separate
databases instances and separate databases 314
- Isolating domains by putting them in the same database
instance but separate databases 315

J

- J2EE artifacts 339
- J2EE developer 409
- jar file 427, 434
 - individual fix pack 434
- Java
 - Enterprise JavaBeans 339
- Java archive 436, 438
 - interim fixes 436
- Java Content Repository (JCR) 15
- Java Management Properties 83
- JDBC Provider 155
- JVM setting 413
- Jython script 413–414

L

- label 339
- Launch Profile Wizard 75
- LDAP
 - Using Ldapsearch.exe 195
- LDAP directory
 - redundance 16
 - replicas 16
- LDAP Properties Configuration 209
- LDAP server tuning 326
- ldapsearch query to confirm that we can connect to and search our LDAP directory 196
- Ldapsearch.exe 195
- Leveraging points of variability in a template 9
- Lotus Domino
 - 7.0.2 45

M

- Maintenance package to install window 67
- Manage application resource usage 406
- Map Modules to server 134
- Map modules to servers 133
- Master Configuration Repository 33
- Member Manager LDAP attribute files 224
- Membership 9
- meta information 7, 10
- Minimal environment set 43
- Mixed horizontal and vertical scaling topology 29
- Module WebSphere Portal Server (wps.war) is now mapped to both servers. 134
- Monitoring
 - what to monitor? 331
- Monitoring basic health of systems 405
- Monitoring database performance 333
- Monitoring end user response time 405
- Monitoring Techniques 405
- Monitoring WebSphere Portal 404
- Most Customizations 35
- Moving a differential release using ReleaseBuilder 346
- Moving JCR & Likeminds to a second DB2 Server and database called "TARGET" 244
- Moving the JCR Domain to a new DB2 server 316
- Multiple LDAP
 - Overview 45
 - support 15
- multiple LDAP 30, 45
 - directory 16–17, 30
 - feature 17
- Multiple LDAP directories with multiple realms 17
- Multiple LDAP directories with multiple virtual portals and realms in WebSphere Portal Version 6 191
- Multiple LDAP directory scenarios 197
- Multiple LDAP Directory Support 187
- Multiple LDAP Overview 30
- Multiple LDAP support 15
- Multiple lines of production with shared Community & Customization domains 239
- Multiple realms with multiple LDAP directories (many to many) 200

Multiple realms with multiple LDAP directories (one to one) 199

N

- navigation 343
- Network Connectivity Requirements 26
- Network Deployment (ND) 25, 363
- New in the v6.0 programming model 13
- No group container or group suffix in Domino LDAP 228
- Node agents restart 158
- Node and host names selection screen 115
- Node and Hostname 78
- Node input 88
- Node maps configuration for Domino LDAP 222
- Node, host and cell names screen 144
- Node, host, and cell names selection window 62

O

- object ID 340, 342
- object Id 340, 342
- Object IDs 340
- Obtain previous release data 355
- One DB server 44
- Operations & Deployment 15
- Optional Steps after Cluster Creation 167
 - 135
- orphan data 358
- orphaned data 358–359
- Our portal environment after splitting the JCR and Likeminds domains 245
- output file
 - portlet.source.xml 422
- Output from SetUniqueNames.xml 354
- Output of ExportThemesAndSkins.xml 351
- Overview 52
- Overview of virtual portals and realms 188

P

- pages 343
- passive configuration 37
- password admin_password 363–364
- Performance maintenance
 - keeping it fast 334
- Performance metrics 326
- Performance Testing (PERF) 41
- Personalizing the Portal 9
- Physical memory 22–23
- Planning for Multiple LDAP directories 197
- Planning for Performance testing and tuning 323
- Planning for your Portal 6.0 High Availability Deployment 21
- Planning your test 323
- Points of variability (Parameters) 8
- Port Value Assignment 78
- Port Values 144
- Portal 5.1 Availability Gold Standard architecture above illustrates two separate WebSphere Portal infrastructures. One is active, and the other one is in standby mode. 38

- Portal 6.0 Availability Gold Standard architecture above illustrates two separate WebSphere Portal infrastructures where two separate portal infrastructures (cells) service user requests 39
- Portal configuration properties 208
- Portal Install failure 373
- Portal Installer 85
- Portal JVM Crash 393
- Portal Runtime Failure 397
- Portal Screen 92, 151
- Portal Server tuning parameters 324
- Portal server tuning parameters 324
- Portal Update Installer xi, 68, 80, 388, 425–426, 468
 - application 426
 - Download 426
 - use 432
 - welcome screen 428
- Portal Update Strategy 406
- Portal Upgrade failure 388
- PortalServer 432–433
- portlet 4, 6, 9, 31, 34, 340, 342, 410
 - configurations 342
 - Custom Unique Names 340
 - instances 339
 - JSPs 339
 - new rule 10
 - XMLAccess tool 343
- portlet war
 - file 364–365, 421
 - unique name 422
- portlets 3–4, 34, 338–339, 447
- Ports selection window 63
- Possible workarounds 229
- Post transfer actions 358
- Prepare Source Environment 349
- Prepare the DMGR and Node1 for the Portal install 80
- Prepare the target environment 357
- Prepare your staging environment 348
- Preparing to build a release 348
- Pre-requisites 201
- print jvm 413
- Problem Determination Approaches 368
- Problems importing pages 365
- Problems importing policies 365
- Problems importing portlets 364
- production
 - system 339
- Production (PROD) 42
- production environment 22–23, 345
 - full export 347
 - recommended setting 362
 - release information 349
- production line 358
 - orphaned data 359
- production server 42–43, 342, 347
 - configuration bloat 342, 348
 - current configuration 347
- Profile Creation 382
- Profile Creation is complete 146
- Profile Creation Summary 145

- Profile creation window 65
- Profile Directory 77, 143
- Profile directory screen 115
- Profile Name 77, 143
- Profile name screen 114
- Profile name window 61
- Profile summary 64
- Profile type selection 75, 142
- Profile type selection screen 113
- Profile type selection window 61
- Profiles 32
- programming model 1–2
- Properties files associated with Database Domains 242
- Properties files used for Database Domains 242
- Providing Logs to Support 400
- Puma SPI 14
- Purposes of each environment 41
- pwd itso 343–344
- pwd wpsadmin 348, 350

R

- Recommendations 42
- Recommendations and Best Practices
 - How many environments do I need? 39
- Recommendations for Navigating this chapter 52
- Recommended process to configure remote external Web server (Overview) 35
- Red Hat
 - Enterprise Linux 25
 - Enterprise Linux ES 25
- Redbooks Web site 453
 - Contact us xiv
- reference architecture 445
- Regenerate the plug-in 135
- release manager 339
- ReleaseBuilder 342
- ReleaseBuilder process overview 346
- Remote database before we moved data 249
- request xmlns
 - xsi 344, 351, 422
- result.xml 343
- Results from a typical ldapsearch query 196
- Run as Service 145
- Run baseline performance tests before deploying your applications 326
- Run Release Builder 356
- Running the Config Wizard 250

S

- Sample wmm.xml files for different LDAP directory combinations 221
- Save changes 82
- Scaled down environment set 42
- Scenario 1, Default realm with multiple LDAP directories (one to many) 197
- Scenario 2, One realm per LDAP directory (one to one) 198
- Scenario 3, Multiple realms & multiple LDAP directories (many to many) 199

- Scenarios that we discuss in this chapter 53
- Script to create new theme and skin definitions 354
- Script user logouts realistically 328
- Scripting performance tests 326
- search API 14
- Secondary Node 118
- Section of WebSphere Portal Support Page 404
- security helper file properties 126
- Select "Transfer data to another database" 251, 266
- Select Instance 87
- Select Source Database Type 267
- Select WebSphere Portal user id and password 150
- Selecting "Set database type for individual domains" 283, 303
- Selecting an attribute for the rule 11
- Selecting individual Database Domains to connect to 284, 304
- Selecting the Database Domains we want to move to Oracle 268
- Selecting the source Database Domains 267
- Selecting the source database type & individual Database Domains 252
- Selecting to connect an additional node to a database 282, 301
- Self service resources 403
- service description 442–443
- service oriented architecture (SOA) 1, 3, 29
- service provider 442–443
- service registry 442–443
- service requestor 442–443
 - client-server relationship 443
- service-based process 447
- service-oriented approach 444
- service-oriented architecture 442, 450
- Services Oriented Architecture (SOA) 441–442
- servlet 339
- Set up transfer directory 349
- Setting goals 329
- SetUniqueNames.xml 353
- Sharing data amongst Portal servers 238
- Simplified portlet creation 4
- single LDAP 16, 44, 46–47, 188–189
 - directory 16
- Single server installation 27
- Single Server topology 27
- single sign-on (SSO) 13
- SiteMinder 36
- skin action 351–352
- skins 339
- SLCheckerTool 358–359
- SOA approach 3
- SOA integrates information and people as well as business processes 3
- SOA project 443–444
- SOA reference architecture 445
- Software Requirements 24
- Some additional Release content created on Node 1 277
- Some JCR content created on portal Node 1 278
- source code 339
- Source Database selection window 99
- source server 341, 355, 421
 - portlet configuration 421
 - working environment 355
- Specify cell name, node name and host name information 148
- Specify user id and password 148
- Specifying a user name and password for the Database Domains we wish to move 255–256
- Specifying connection properties for DB2 285, 305
- Specifying connection properties for Oracle 286, 306
- Specifying DB2 database properties 254
- Specifying group values for Domino LDAP 229
- Specifying properties for the Community Database Domain 287, 307
- Specifying properties for the Customization Database Domain 288, 308
- Specifying properties for the Likeminds Database Domain 289, 310
- Specifying properties for the WMM Database Domain 290, 309
- Specifying target databases 253
- Specifying the DB2 database server properties 269
- Specifying the JAVA_HOME variable in lbe.bat 194
- Specifying the Oracle database server properties 270
- Specifying the source database type 252
- Specifying the target Oracle database properties for the Community Domain 271
- Specifying the target Oracle database properties for the Customization Domain 271
- Specifying the target Oracle database properties for the Release Domain 272
- Specifying the target Oracle database properties for the WMM Domain 273
- Specifying the WebSphere Application Server Administrator user name and password 251
- Specifying the WebSphere Application Server credentials 283, 302
- staging 339
- Staging (STAGE) 41
- staging environment 42, 338, 349, 358
 - exported output 345
 - full export 347
- standard ANT
 - command 410, 422
 - package 411
 - procedure 421
- Step 1 369, 386, 389, 394, 398
- Step 2 370, 387, 391, 394, 399
- Step 3 372, 388, 393, 395, 399
- Step 4 372, 395
- Steps for adding an additional LDAP directory 216
- Steps for moving Database Domains between database types 262
- Steps for moving Database Domains between the same database type 245
- Steps for sharing Database Domains between portal clusters 296
- Steps for sharing Database Domains between portal servers 277
- Stop and Start WebSphere Portal Server 385

- Strategies for Tuning and Testing 321
- Successful database connect 312
- Successful database transfer 258, 292
- Successful transfer 274
- Successfully testing our JDBC connections to DB2 and Oracle 275
- Summary 273
- Summary of example 3, Sharing Database Domains amongst portal servers 294
- Summary of the Database Domains we wish to connect to 291, 311
- Summary screen 257
- Summary screen view 178, 183
- Sync with Node 82
- Sync with nodes was successful 83
- system administrator 7, 408, 410
- System Requirements 22

T

Table 9-1

- A list of important installation logs 374
- Target Database selection window 100
- target name 410, 421
- target server 343, 355–357
 - current date directory 358
 - same directory 366
 - same transfer directory structure 356–357
 - source server 356
- Task selection screen 124
- Temp directory 22–23
- Templates 7
- Terminology 234, 338
- Testing (QA) 41
- Testing our Customization database domain 293
- Testing our shared Database Domains 292
- Testing scenarios 329
- testing team 41, 43, 338
- Testing your JDBC Data Connection on the DMGR 105
- The Availability Gold Standard in Portal 5.1 37
- The base model 13
- The build and staging process 339
- The final example 4 environment 312
- The ideal set of environments and data flow between them 40
- The main drop down menu and contextual menus 5
- The New Availability Gold Standard in Portal 6.0 38
- The Portal Staging Process 338
- The relationship between virtual portals and realms 189
- The release cycle 338
- The XML Configuration Interface 341
- The XML configuration interface 341
- The XMLAccess tool 341
- theme J_NO2UF411186E1026H4BLVI00E7 352
- themes 339
- Tivoli Access Manager 36
- To alter the schedule of the cleanup service 362
- To disable scheduled cleanup completely 361
- To run the cleanup task immediately, follow these steps

- 361
- Tools to help you understand your LDAP directory 193
- Top down approach to create the XML Structure 411
- Transfer data to another database window 98
- Transfer directory 350
- transfer directory 349–350
- Transfer process 257, 344
- transfer process 344
- Transfer success window 105
- Transferring 345
- Transferring from Cloudscape to DB2 313
- Transferring Portal artifacts using XMLAccess 342
- Transport Chain 81
- Troubleshooting 364
- Troubleshooting and Monitoring 367
- Two Nodes connected to the same databases with their own Release domains 317
- Two realms and two LDAP directories 215
- Typical Deployment Scenarios 27

U

- UI logic 6
- Uniform Resource Identifier 343
- unique name 340, 345
- UniqueNames 351
- UNIX
 - export.sh 350
 - import.sh 350
- update installer 425–426
 - file 426
 - Java environment 427, 432
 - multiple copies 432
 - tool 427
- Updating Portal 238
- Upgrade failure 389
- Use an appropriate think time 328
- user admin_user_id 363–364
- User group 31, 351
- User interface enhancements 5
- user wpsadmin 343–344
- Username and password window 98
- Using an LDAP browser tool 193
- Using Database Domains in a production environment 313
- Using ldapsearch.exe 195
- Using the ReleaseBuilder tool 347
- Using the XMLAccess tool for moving 341

V

- Validating credentials for realms 212
- Validating WebSphere Administration Server credentials 303
- Validation 375
- Values to update in the wmm.xml file for an additional LDAP directory 220
- Verify Federation 79
- Verify the Profile directory selection window 62
- Verifying the results of the Database Domain moves 259
- version control system 339

- Vertical cluster 46–47
- Vertical Cluster (includes WebSphere Process Server)-
Installation and Configuration steps 176
- Vertical Cluster (without WebSphere Process Server)-
Installation and Configuration steps 181
- Vertical Cluster Deployment Architectur 176, 181
- Vertical cluster with database domain 47–48
- Vertical cluster with database domain and multiple LDAP
48–49
- Vertical cluster with single database and single LDAP
46–47
- Vertical Scaling topology 27
- Virtual memory/swap space 22–23

W

- WAR file 347, 354, 410, 424
- Warm up the portal 329
- WAS V6 Web Server Changes 34
- Wasadmin input 89
- Web Server
 - Change 34
 - Consideration 33
- Web Server Considerations 33
- Web Server tuning 325
- Web Server tuning parameters 325
- WebSphere Application Server
 - administrative server 363
 - clustered environment 434
 - installation directory 427, 432
- WebSphere Application Server Fixes Installation 382
- WebSphere Application Server Installation 380
- WebSphere Application Server Installation on
DMGR(WSAS) 56
- WebSphere Portal
 - 6 1, 27
 - 6.0 1
 - access control management 15
 - Administration Interface 351
 - artifacts 341–342
 - cluster 28
 - content publishing 435
 - content publishing feature 439
 - default portlets configuration 343
 - Document Manager 15
 - Ear 363
 - EAR file 363
 - environment 40
 - infocenter 344, 422
 - Information Center 14
 - Infrastructure 39
 - infrastructure inside 17
 - main configuration tool 424
 - main functional areas 5
 - multiple installations 434
 - new releases 408
 - node 28
 - offering 4, 18
 - platform capability 12
 - portlet configurations 342
 - product 435, 440
 - programming model 2, 12–13
 - property 434
 - Server 4, 15, 25, 34, 353, 414
 - server instance 33
 - Server Version 6 3, 16
 - Setup CD 25
 - site map 343
 - skin definitions 342
 - strategy 434
 - theme definitions 342
 - URL mappings 343
 - V 6.0 14
 - V 6.0 programming model 14
 - V5.1 13–14, 18
 - V5.1.0.1 13
 - V5.1.0.1 Programming Model 13
 - V6 5–6
 - V6 help operator 19
 - V6 Infocenter 19
 - Version 6 1–2
 - Version 6 programming model 1
 - Version 6.0 product 426
 - Web application configurations 342
 - zone 14
- WebSphere Portal (WP) 1–2, 22–23, 348, 408, 411,
425–426
- WebSphere Portal 6 as the “Front End” to SOA 4
- WebSphere Portal and SOA 3
- WebSphere Portal Installation 383
- WebSphere Portal Product Library 403
- WebSphere Portal programming model 12
- WebSphere Portal Security LTPA and SSO configuration
208
- WebSphere Portal Security LTPA and SSO Configuration
(Optional) 212
- WebSphere Portal version 6.0 Information Center 403
- WebSphere Process Server
 - runtime 30
 - Version 6.0.1.1 25
- WebSphere Process Server Considerations 29
- WebSphere Process Server Installation 381
- WebSphere Process Server Installation on DMGR(WPS)
59
- WebSphere Support Pages 403
- WebSphere variable defined in the Class Path text box
106
- WebSphere Variables Scope 155
- Welcome to IBM WebSphere Application Server 71
- Welcome to Profile Creation Wizard 142
- Welcome to WebSphere Process Server Installation 74
- Welcome window 57, 139
- What are Database Domains? 234
- What’s new in WebSphere Portal Version 6 3
- Which portal data can be shared 237
- Why use ReleaseBuilder? 342
- Why use XMLAccess? 341
- Window service selection screen 150
- Windows
 - Export.bat 350
 - Import.bat 350

- Windows service definition configuration window 63
- WMM database ID and password screen 125, 206
- Workaround for JIT Problems 396
- Workflow 9
 - workflow builder 5, 9
- Workflow Support 5
- wpconfig_domain.properties properties 212
- wps application 364
- wpsadmin input window 90
- WPSconfig task failure 368
- wsadmin command 363, 416
- wsadmin script 339
- WSAS detection screen 112
- WSAS detection window 60

X

- XML configuration interface. See XMLAccess tool.
- XML export 340
 - objectid attribute 340
- XML file 7–8, 341, 346, 409–410
- XML structure 411
- xml version 344, 351, 409, 421
- xmlaccess export 346–347
- XMLAccess file to alter scheduled cleanup of portal resources 362
- XMLAccess file to perform immediate cleanup of portal resources 361
- XMLAccess import 345, 365
- XMLAccess tool 340–344, 350
- xmlaccess.bat 343
- xmlaccess.sh 343



WebSphere Portal V6 Enterprise Scale Deployment Best Practices

(0.5" spine)
0.475" <-> 0.873"
250 <-> 459 pages



Redbooks

WebSphere Portal Version 6 Enterprise Scale Deployment Best Practices

Planning for Portal deployment

Building and configuring high-availability clusters

Troubleshooting and monitoring the environment

This IBM book introduces new key technical features in WebSphere Portal Version 6.0 and discusses how these new features have a strong bearing on best practices for deploying WebSphere Portal. Specifically, it focuses on the following areas:

- ▶ Planning for deployment with an overview of several typical deployment scenarios.
- ▶ High-availability deployment strategies by installing IBM WebSphere Portal V6.0 in a clustered deployment/Network Deployment configuration.
- ▶ Configuring WebSphere Portal Version 6 to support multiple different LDAP directories.
- ▶ Considerations and implementation details for configuring Portal to share data between different database domains.
- ▶ Strategies for tuning and testing your WebSphere Portal 6 environment.
- ▶ Configuration Management—procedures necessary to move your WebSphere Portal environment through pre-production release phases and deploy into the Production environment.
- ▶ Best practice approaches to problem determination and troubleshooting.
- ▶ Available tools for deployment automation, reducing the manual steps required by the portal server installation and configuration process.
- ▶ Discussion of how to use the IBM WebSphere Portal update installer application to install interim or cumulative fixes and fix packs in WebSphere Portal Version 6.0.
- ▶ A discussion of Services Oriented Architecture (SOA) and what this means within the context of Portal.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7387-00

ISBN 0738486027