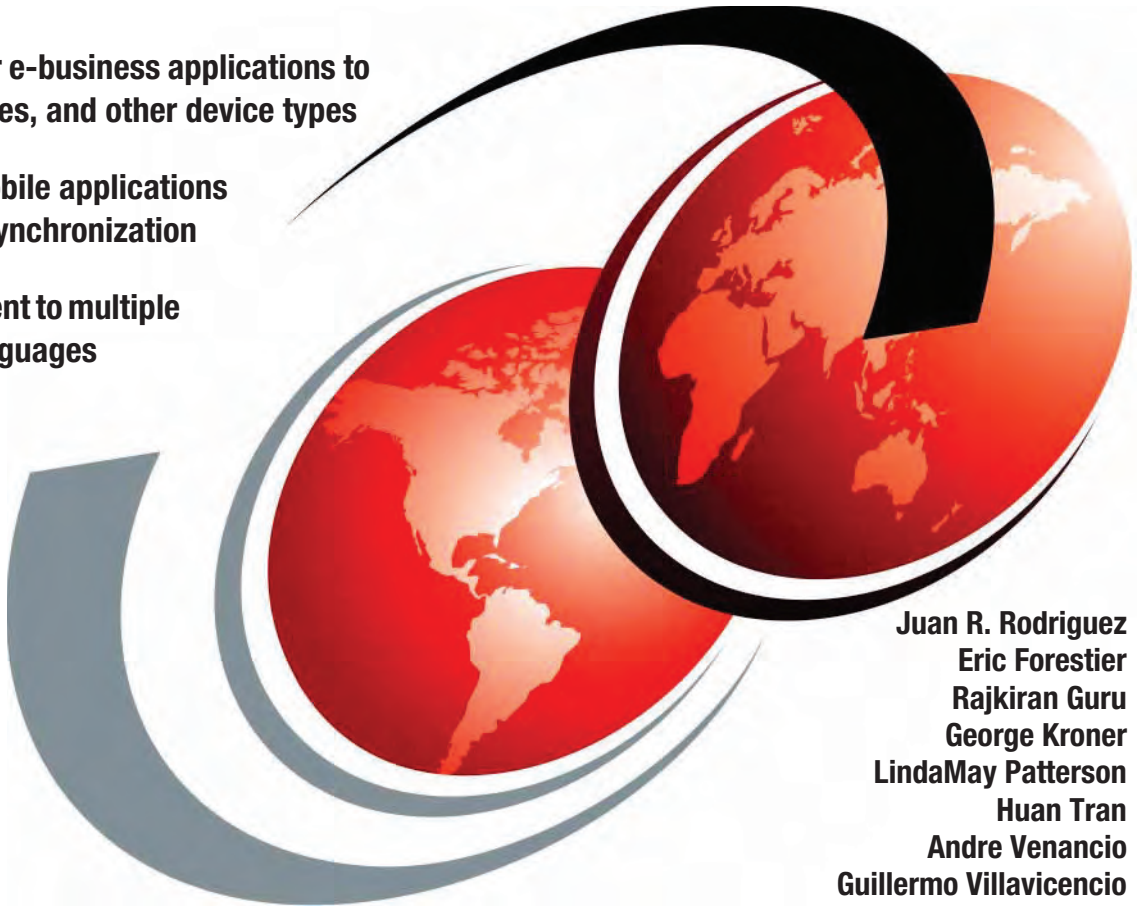
WebSphere. software

IBM

WebSphere Everyplace Access Version 4.3 Handbook for Developers

- Extend your e-business applications to PDAs, phones, and other device types
- Develop mobile applications with data synchronization
- Adapt content to multiple markup languages



Juan R. Rodriguez
Eric Forestier
Rajkiran Guru
George Kroner
LindaMay Patterson
Huan Tran
Andre Venancio
Guillermo Villavicencio

Redbooks

ibm.com/redbooks

Petitioner has added letters to pages (a)-(c) and numbers to pages 1299-1300. Otherwise, it leaves the original page numbering.

SAP AMERICA, INC.
EXHIBIT 1012

(a)



International Technical Support Organization

WebSphere Everyplace Access Version 4.3 Handbook for Developers

November 2003

SG24-7015-01

(b)

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

Second Edition (November 2003)

This edition applies to Version 4, Release 3 of IBM WebSphere Everyplace Access for multiplatforms.

This document was updated on November 14, 2003.

© Copyright International Business Machines Corporation 2003. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices xv
Trademarks xvi

Preface xvii
The team that wrote this redbook. xvii
Become a published author xix
Comments welcome. xx

Chapter 1. Overview 1
1.1 The Big Picture 2
1.2 Inside the product 4
1.2.1 WebSphere Everyplace Access 4
1.2.2 Portlets 6
1.2.3 WebSphere Everyplace Access services 7
1.2.4 Everyplace Client 14
1.2.5 Everyplace Toolkit Version V4.3 19
1.2.6 Component products. 20
1.2.7 Complementary products 23

Chapter 2. Administration 25
2.1 Getting started. 26
2.2 Managing users and groups 27
2.3 Install portlets 34
2.4 Manage places and pages 38
2.5 Managing access control. 47
2.6 Changing themes and skins 56

Chapter 3. Enhanced portlets 63
3.1 Introduction 64
3.1.1 Supported devices 64
3.2 Enhanced portlets provided by Everyplace Access 65
3.2.1 Lotus Notes PIM portlets. 67
3.2.2 Microsoft Exchange PIM portlets 71
3.2.3 Productivity portlets. 74

Chapter 4. Everyplace Client 81
4.1 Overview 82
4.2 Everyplace Client for Palm OS 82
4.2.1 Everyplace Client installation 82

© Copyright IBM Corp. 2003. All rights reserved.

4.2.2 Everyplace Client configuration for Palm OS 5.2 92
4.3 Everyplace Client for Pocket PC 2002 95
4.3.1 Everyplace Client installation 95
4.3.2 Everyplace Client configuration. 101
4.4 Everyplace Client for Zaurus 105
4.4.1 Desktop Pass-through Proxy installation 106
4.4.2 Desktop Pass-through Proxy configuration. 106
4.4.3 Everyplace Client installation 107
4.4.4 Everyplace Client configuration. 109

Chapter 5. Everyplace Client secure connections 115
5.1 Introduction 116
5.2 Enabling SSL on IBM HTTP Server 116
5.2.1 Creating a key database 116
5.2.2 Create a self-signed key file 118
5.2.3 Setting up IBM HTTP Server. 120
5.2.4 Verifying if security is enabled on IBM HTTP Server 121
5.3 Enabling SSL in WebSphere Application Server. 121
5.3.1 Configuring WebSphere Application Server 121
5.3.2 Verifying if security is enabled on WebSphere Application Server . 122
5.4 Enabling SSL in Everyplace Client 122
5.4.1 Enabling SSL in Pocket PC devices 122
5.4.2 Enabling SSL in Palm devices 124

Chapter 6. Mobile application development using portlets 129
6.1 Portlets 130
6.1.1 Portlet terminology 132
6.2 How portlets work 137
6.2.1 The Model-View-Controller (MVC) architecture 137
6.2.2 The portlet life cycle 138
6.3 Portlets and WebSphere Everyplace Access 140
6.4 Making a portlet project 140
6.4.1 Defining a portlet project 141
6.4.2 Types of portlet projects 143
6.4.3 Parameters of a portlet project 144
6.4.4 Contents of a portlet project 146
6.5 Testing and debugging portlets 154
6.5.1 Setting up a test environment and running a portlet 155
6.5.2 Testing portlet projects on Pocket PC, Palm, and WAP devices . . 161
6.5.3 Debugging with the test environment 171

Chapter 7. WebSphere Studio Site Developer and Everyplace Toolkit . 179
7.1 WebSphere Studio Site Developer V5.0 180
7.1.1 WebSphere Studio Site Developer 180

7.1.2 Site Developer and Application Developer 181
7.1.3 Everyplace Toolkit 182
7.1.4 Everyplace Toolkit and the Portal Toolkit 183
7.1.5 Multiple Device Authoring Technology 184
7.1.6 Other tools of interest 185
7.2 The WebSphere Studio Site Developer workbench 187
7.2.1 Starting Site Developer 187
7.2.2 The workbench user interface 188
7.2.3 How to get help 190

Chapter 8. My first portlet applications 193
8.1 Create a MVC portlet application supporting HTML 194
8.1.1 Instructions 194
8.1.2 Examining the HelloWorldMVC project 207
8.1.3 Changing a portlet application 209
8.2 Create a JSP portlet that supports WML 215
8.2.1 Modifying the HelloWorldJSP portlet 225

Chapter 9. Portlet action event handling 229
9.1 Overview 230
9.2 Create the ActionEvent portlet 232
9.3 Update the portlet code 237
9.4 Look inside the ActionEventLab project 238
9.5 WML and the ActionEvent portlet 247
9.5.1 Examine the code that renders the content 247

Chapter 10. Portlet messaging 253
10.1 Overview 254
10.2 Using a portlet to send a message 255
10.3 Creating the target portlet 257
10.4 Receiving a message 268
10.5 Displaying the message in View mode 270
10.6 Running the portlet application 270

Chapter 11. Portlet National Language Support (NLS) 275
11.1 Overview 276
11.2 Creating the NLS bundles 277
11.3 Accessing NLS bundles from JSPs 280
11.4 Running the portlet 281

Chapter 12. Portlet Credential Vault 287
12.1 Overview 288
12.2 Creating the portlet project 292
12.3 Updating the active Credential Vault project 298

12.4 Reviewing the portlet code 299
12.5 Running the portlet project 302

Chapter 13. Offline Portal Content 307
13.1 Overview 308
13.2 How it works 308
13.3 Limitations 309
13.4 Configuration 309
13.4.1 Server configuration 309
13.4.2 Administrator configuration 310
13.4.3 User configuration 312
13.5 Usage 317
13.5.1 Pocket PC devices 317
13.5.2 Palm OS V5.2 devices 320
13.6 Development guidelines 324
13.6.1 Enable support for PDA markup 324
13.6.2 Adhere to XML “well-formedness” 324
13.6.3 Do not use Form GET method 325
13.6.4 Do not use PortletActions 325
13.6.5 Plan for dynamic content 325
13.6.6 Avoid action buttons 325
13.6.7 Avoid cascading forms 326
13.7 Scenarios 326
13.7.1 Scenario 1: Offline browsing 326
13.7.2 Scenario 2: Offline forms 344
13.7.3 Scenario 3: Converting an online portlet for offline usage 356
13.8 Hints and tips 372
13.8.1 Deleting users 372
13.8.2 Changing PDA icon 373
13.9 Resources 376

Chapter 14. Transcoding Technology 377
14.1 Overview 378
14.1.1 Architecture 379
14.1.2 Preference profiles 381
14.1.3 XML stylesheets 384
14.1.4 Annotators 384
14.1.5 Transcoding plug-ins 385
14.2 XML configuration utility 386
14.3 Request Viewer 387
14.3.1 How to start Request Viewer 390
14.4 Logging and tracing 394
14.4.1 Message files 395

14.4.2 Trace files 396
14.4.3 Gather troubleshooting data 396
14.5 A simple portlet using Transcoding Technology 396
14.5.1 Enable transcoding 396
14.5.2 Use Request Viewer to monitor the process 401

Chapter 15. Using annotation to clip HTML documents 403
15.1 Annotation overview 404
15.1.1 Annotation processing 405
15.2 Internal annotation 405
15.2.1 Page Designer in WebSphere Studio 406
15.2.2 Sample application: MyRedbookNews 407
15.3 External annotation 416
15.3.1 The external annotation language 418
15.3.2 Sample scenario: YourRedbookNews 419
15.3.3 Using the HTML Annotation Editor 420
15.3.4 Running our portlet with external annotations 431
15.3.5 Viewing the results of external annotation 433

Chapter 16. Using XSL transformations and XML tools 441
16.1 Overview 442
16.1.1 Configuring stylesheets 442
16.2 Configure Portal for stylesheet processing 443
16.3 Sample scenario 1: RSSNewsFeed 446
16.3.1 Configuring the RSSNewsFeed portlet 448
16.4 Sample scenario 2: ITSONewsBrief 449
16.4.1 Sample stylesheets 451
16.5 XML tools 456
16.5.1 DTD editor 457
16.5.2 XML editor 470
16.5.3 XSL editor 479
16.5.4 XML to XML mapping editor 497
16.5.5 Summary 508

Chapter 17. Portal-level transcoding 511
17.1 Overview 512
17.1.1 How it works 513
17.1.2 Fragmentable elements 516
17.1.3 Common problems 516
17.1.4 Example 516
17.2 WML fragmentation 518
17.2.1 Scenario 1: Using a WAP reverse proxy 519
17.2.2 Scenario 2: Using a forward proxy 525
17.2.3 Scenario 3: Using a forward proxy and reverse proxy 527

Chapter 18. DB2 Everyplace applications with WebSphere Studio Device Developer 529
18.1 Overview 530
18.1.1 Introduction 530
18.1.2 Java 2 Micro Edition 530
18.1.3 WebSphere Micro Environment 532
18.2 Installing WebSphere Studio Device Developer 532
18.3 Working with WebSphere Studio Device Developer 536
18.3.1 Using the workbench 536
18.3.2 Using the Update Manager 537
18.4 Sample scenario 545
18.4.1 Define the database tables for the application 545
18.4.2 Configure the Pocket PC device for development 545
18.4.3 Create the project on WebSphere Studio Device Developer 548
18.4.4 Set up the build and test environment 552
18.4.5 Write the Java classes of the application 569
18.4.6 Test the application on the device 597

Chapter 19. DB2 Everyplace applications with Mobile Application Builder (MAB) 603
19.1 Overview 604
19.2 Installation 604
19.3 Required components for Palm development 606
19.4 Windows CE development environment 608
19.5 DB2 Everyplace runtime environment 610
19.6 Scenario: A sales force automation application 610
19.6.1 Table definitions for the application 610
19.6.2 Planning the flow of the application 614
19.6.3 Create a project in the Mobile Application Builder 614
19.6.4 Configure the preferences 617
19.6.5 Load the table definitions 620
19.6.6 Start creating the forms 622

Chapter 20. Synchronizing with DB2 databases 645
20.1 Architecture overview 646
20.1.1 DB2 Everyplace 647
20.1.2 IBM Everyplace Client 648
20.1.3 DB2 Everyplace Sync Server 648
20.2 Before you start 648
20.3 Server configuration 650
20.3.1 Creating users and groups 650
20.3.2 Creating subscription and subscription set 653
20.4 Binding LDAP and MDAC 663
20.5 DB2 Everyplace Client configuration 665

20.6 Sample application synchronization 666
20.7 Verify the synchronization 668
20.8 Synchronization using SSL 671
20.8.1 Enable server security 671
20.8.2 Enable client security 671
20.9 Synchronization with remote DB2 databases 671
20.10 Types of subscription 673
20.10.1 DataPropagator subscription 673
20.10.2 Upload subscription 684
20.11 Filtering data from data sources 689
20.12 Debug and tracing 690
20.12.1 Enable tracing 690
20.12.2 Trace files 690
20.12.3 DB2 Everyplace control database 690
20.13 Hints and tips 691

Chapter 21. Synchronizing with Oracle databases 693
21.1 Common grounds with DB2 data source 694
21.2 Create a subscription with Oracle data source 695
21.2.1 Add Oracle JDBC driver 695
21.2.2 Create a JDBC subscription 696
21.2.3 Create an upload subscription 705
21.3 Sample dsysetjavahome.bat file 709
21.4 Synchronize with remote Oracle database 709
21.5 Hints and tips 710

Chapter 22. PIM and e-mail synchronization with Domino Server 711
22.1 Overview 712
22.2 How it works 712
22.3 Configuration 714
22.3.1 Lotus Domino Server configuration 714
22.3.2 Lotus Notes client installation and configuration 729
22.3.3 Lotus Notes User configuration 733
22.3.4 WebSphere Everyplace Access server configuration 736
22.3.5 WebSphere Everyplace Access user configuration 749
22.3.6 WebSphere Everyplace Access Client configuration 754
22.4 Usage 761
22.4.1 Online PIM portlets 761
22.4.2 Setting up synchronization scenarios 766
22.4.3 Performing synchronization 779
22.4.4 Verifying Synchronization 804
22.5 Hints and tips 806
22.5.1 Limitations 806

22.5.2 Troubleshooting 806
22.5.3 Log files 806
22.5.4 Tuning 806
22.5.5 Order does matter 806
22.6 Resources 807

Chapter 23. PIM and e-mail synchronization with Exchange 2000 809
23.1 Overview 810
23.2 WP portlets for ESS 811
23.3 Synchronizing e-mail and PIM information 811
23.3.1 Exchange 2000 Server configuration 811
23.3.2 Everyplace Synchronization Server configuration 812
23.3.3 Synchronization users and groups 814
23.3.4 Device profiles 816
23.3.5 Client configuration 822
23.4 Hints and tips 828

Chapter 24. Instant messaging 829
24.1 Overview 830
24.1.1 Sample scenario 830

Chapter 25. Intelligent Notification Services (INS) 839
25.1 Overview 840
25.1.1 Concepts 840
25.1.2 Components 841
25.1.3 INS application programming interfaces 844
25.2 Intelligent Notification Services enablement 845
25.2.1 Update portlet configuration (distributed environment only) 846
25.2.2 Starting INS administration servers 846
25.2.3 INS server configuration 849
25.2.4 Starting INS servers 849
25.2.5 Enable INS users 851
25.3 INS implementation 852
25.3.1 Defining the needs 852
25.3.2 Analyzing the needs 852
25.4 INS development: Custom Delivery Channel 853
25.4.1 Custom gateway adapter 853
25.4.2 Delivery channel portlet 865
25.5 INS development: Custom subscription 880
25.5.1 Custom content adapter 880
25.5.2 Custom trigger handler 883
25.5.3 Custom subscription portlet 892
25.6 INS server configuration 908
25.6.1 E-mail subscription configuration 908

25.6.2 Content adapters configuration 910
25.6.3 Gateway adapters configuration 917
25.7 INS user configuration 921
25.8 Using the message center 938
25.8.1 Create a user group 939
25.8.2 Create a message rule 941
25.8.3 Enabling the delivery channels 942
25.8.4 Triggering message rules 943
25.9 INS enablement and configuration check list 946
25.10 Viewing INS LDAP settings 946
25.11 Troubleshooting 953
25.11.1 INS log files 953
25.11.2 Suggested actions 956

Chapter 26. Server Initiated Actions (SIA) 957
26.1 Overview 958
26.1.1 Description 958
26.1.2 Prerequisite settings 962
26.1.3 User defined server-initiated actions 965
26.1.4 Administrator-defined server-initiated actions 968
26.2 Troubleshooting 969

Chapter 27. Location Aware Services 971
27.1 Introduction 972
27.2 Architecture 974
27.2.1 Service adapters 975
27.2.2 LAS portlet service 975
27.2.3 Administration portlets 977
27.3 Installation, configuration and verification 977
27.3.1 Installation 977
27.3.2 Configuration 977
27.3.3 Verifying the installation 977
27.4 Administration portlets 979
27.4.1 Adding a custom service provider 979
27.4.2 Modifying the properties of a service and service provider 983
27.4.3 Enabling and disabling the service 984
27.5 Sample Location Aware Services application 985
27.5.1 Getting a map of a location 985
27.5.2 Adding locations for persistence 987
27.5.3 Getting directions 989
27.5.4 Getting points of interest near location 992
27.5.5 Location Aware Services on devices 994
27.6 Application development 996

27.6.1 Location Aware Services API for application development 996
27.6.2 Developing Location Aware Java application 996
27.6.3 Developing Location Aware portlet application 1000
27.7 Adapter development 1008
27.7.1 Development 1010
27.7.2 Registering the custom adapter 1012
27.8 Troubleshooting 1013

Chapter 28. Device Services 1015
28.1 Device Services overview 1016
28.1.1 Architecture 1016
28.1.2 Device Services 1017
28.1.3 Device Services Agent 1017
28.1.4 Device Services database 1017
28.1.5 Device Management portlets 1017
28.1.6 Device Services console 1018
28.2 Setting the stage 1019
28.2.1 Creating a device management user group 1019
28.2.2 Starting the Device Services console 1020
28.3 Scenarios 1022
28.3.1 Bootstrapping and configuring Everyplace Access Client 1022
28.3.2 Software distribution 1036
28.3.3 Configuring SSL on a Palm device 1046
28.4 Device management for Sharp Zaurus 1054
28.5 Troubleshooting 1054

Chapter 29. Everyplace Client API for Pocket PC 2002 1057
29.1 Overview 1058
29.2 Usage 1058
29.2.1 Commands available 1058
29.3 Developing a custom synchronization client 1060
29.3.1 Java classes for the customized client 1060
29.3.2 Running the sample 1073
29.4 Hints and tips 1075

Appendix A. Everyplace Access sample installation instructions 1077
System requirements 1078
Hardware requirements 1078
Software requirements 1078
Planning for installation 1078
Setup Manager 1078
Installation instructions 1079
Installing Lotus Notes client 1080
Installing Microsoft Outlook 1080

Making sure that port 80 is not used 1082
Setting the domain suffix 1083
Starting the install and selecting the features to be installed 1083
Installation begins 1125
Configuring the admin role 1128
Verifying the install 1138
Troubleshooting 1142

Appendix B. Sample Oracle Enterprise Edition installation 1145
Oracle installation 1146
Create a simple database using wizards 1160
Create simple table using wizards 1166
Populate table with data 1176

Appendix C. Palm Simulator installation 1177
Overview 1178
Installation 1178
Configuration 1180
Usage 1181

Appendix D. Palm Emulator installation 1185
Overview 1186
Installation 1186
Downloading ROM from Palm device 1186
Usage and skins 1189

Appendix E. Palm Desktop installation 1193
Overview 1194
Installation 1194

Appendix F. Pocket PC development tools 1201
Overview 1202
Microsoft ActiveSync 3.7 1202
Installation 1202
Microsoft Remote Display Control 1206
Installation 1206
Usage 1208

Appendix G. Portlet development platform installation 1211
G.1 Prerequisites 1212
G.2 Windows configuration (before installation) 1213
G.3 Configuring TCP/IP 1219
G.4 DB2 installation 1220
G.4.1 DB2 installation 1220

G.4.2 DB2 fix pack installation 1227
G.5 WebSphere Studio Site Developer V5.0 1233
G.6 Application Server Single Server V4.0 FP4 1235
G.6.1 Manually starting WebSphere Application Server 1241
G.7 WebSphere Everyplace Access/Portal Toolkit V4.3. 1242
G.8 Configure WebSphere Studio Site Developer and the Toolkit 1250

Appendix H. Lotus Domino Server and Sametime installation. 1257
Installing Lotus Domino Server 1258
Installing the Domino Administration Client 1265
Installing Lotus Sametime Server. 1268
Installing Sametime V3.0 Server. 1269
Installing Sametime V3.0 Service Pack 1 1270
Installing Sametime Mobile Extension V3.0 1270

Appendix I. Additional material 1271
Locating the Web material 1271
Using the Web material 1271
System requirements for downloading the Web material 1272
How to use the Web material 1272

Related publications 1273
IBM Redbooks 1273
Other publications 1273
Online resources 1273
How to get IBM Redbooks 1274
Help from IBM 1274

Index 1275

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 illustrates 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. 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 IBM's application programming interfaces.

© Copyright IBM Corp. 2003. All rights reserved.

Trademarks

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

@server™	DB2®	Rational®
Redbooks (logo) 	Everyplace®	Redbooks™
ibm.com®	IBM®	Sametime®
iNotes™	Lotus Notes®	SecureWay®
AIX®	Lotus®	Tivoli®
DataPropagator™	Maestro™	WebSphere®
Domino™	Notes®	WorkPad®
DB2 Universal Database™	PartnerWorld®	

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Preface

This IBM Redbook helps you plan and develop mobile applications to run in a WebSphere® Everyplace® Access Version 4.3 environment. The information provided in this redbook targets business-to-employee (B2E) enterprise applications, but most of the scenarios presented will apply to business-to-consumer (B2C) applications as well. In this redbook, you will find step-by-step examples and scenarios showing ways to integrate your enterprise applications into a WebSphere Everyplace Access environment using the WebSphere Studio Site Developer and the Everyplace Toolkit as well as extending your portlet capabilities to use other advanced functions such as Transcoding Technology. You will also find numerous scenarios describing recommended ways to develop and implement server initiated actions, everyplace client security and others.

This redbook includes sample scenarios describing ways to implement Intelligent Notification Services to notify Lotus® Sametime®, SMTP e-mail, WAP devices and others. It also includes scenarios illustrating the new Location Aware Services (LAS) function to provide access to location-based services from multiple vendors using the available APIs. Sample scenarios will also include offline applications using Offline Portal Content and Forms, DB2e applications and Relational Database Synchronization with JDBC back-end databases, PIM and e-mail synchronization with Domino™ and Microsoft Exchange servers.

A basic knowledge of Java technologies such as servlets, JavaBeans, EJBs, JavaServer Pages (JSPs), as well as XML applications and the terminology used in Web publishing, is assumed.

The team that wrote this redbook

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



Juan R. Rodriguez is a Consulting IT professional at the IBM® ITSO Center, Raleigh. He received his Master of Science degree in Computer Science from Iowa State University. He writes extensively and teaches IBM classes worldwide on such topics as networking, Web technologies, and information security. Before joining the IBM ITSO, he worked at the IBM laboratory in Raleigh, North Carolina as a designer and developer of networking products.

© Copyright IBM Corp. 2003. All rights reserved.

Eric Forestier is an IT Architect and works at the e-business Solutions Center in IBM La Gaude, France. He currently is working with the IBM Pervasive Computing Division and provides EMEA advanced technical support to Independent Software Vendors, assessing and enabling IBM Partners to include the pervasive computing assets into their solutions. He has also been a system sofware developer, working in various areas such as networking, telephony and Internet.



Rajkiran Guru is a software engineer at IBM Software Labs, India, where he has worked since 1999. He is in the pervasive computing group, focusing on SyncML Data synchronization and Device Management protocols. His current focus is on server side technology in the pervasive computing area. He holds a Master of Science degree in Software Systems.



George Kroner is a Co-op IT Specialist at the IBM ITSO Center in Raleigh, North Carolina. He is currently pursuing a Bachelors of Science degree in Information Sciences and Technology at Pennylvania State University. His interests include Web applications, pervasive computing, intelligent interfaces, and business process refinement.



LindaMay Patterson is an IT Consultant for the IBM @server Custom Technology Center in Rochester, MN. She currently is working with the IBM Pervasive Computing Division and provides support for the Product Management Team. She has worked in a wide variety of organizations including PartnerWorld®, Information Systems and Advanced Technology. She has written various papers and articles on XML and XML-related technologies and has contributed to various Redbooks™.



Huan Tran is an IT Specialist for the Adelaide Application Center in IBM Global Services Australia. He has worked on a number of commercial projects, providing him with experience on various programming languages and technology. His areas of expertise includes J2EE Web Application Development on WebSphere Application Server, CORBA and Web Services. He has had roles as a lead developer and technical architect on a complex J2EE ordering and provisioning system.





Andre Venancio is an Advisory IT Specialist with the WebSphere Software Platform (Software Group) in IBM Brazil. He has four years of experience with WebSphere products developing solutions with Host Integration, Edge Server and Portal solutions. He holds a Bachelor's Degree in Mathematics from the Fundação Santo André in Brazil and a post-graduate degree from Faculdade de Informática e Administração Paulista (FIAP), Sao Paulo Brazil on enterprise solutions using distributed object technologies with Java.



Guillermo Villavicencio holds a degree in Informatics Engineering from the Pontifical Catholic University of Peru and works as a Software Architect for Avatar e-Business Solutions, an IBM Business Partner. He has been the architect for several e-business projects including wireless and Portal solutions. His current area of expertise is centered around Web technologies and pervasive computing.

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

Margaret Ticknor
International Technical Support Organization, Raleigh Center

Greg C. Smith, James Thrasher, Michael Wiles, Chuck Proffer, J Smith Doss
Jim Brancato, Jobin John, Mohit Jain, Renee Kovales, Mary Curran
Nichelle Hopson, Sung-Ik Son, Bill Trautman
IBM Research Triangle Park, North Carolina, USA

Mary Fisher
IBM Boca Raton, Florida, USA

Become a published author

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

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

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

`ibm.com/redbooks/residencies.html`

Comments welcome

Your comments are important to us!

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

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

  `ibm.com/redbooks`

- Send your comments in an Internet note to:

  `redbook@us.ibm.com`

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195

1

Overview

As the popularity of mobile computing grows, mobile workers are realizing the value of having access to information and resources held by the enterprise. Access to this information and the enterprise resources allows them to work more effectively and stay up to date with changing events and new information while away from the office. Likewise, enterprises are seeing real benefit from providing their mobile work force with mobile devices and giving access to enterprise information and applications. With advances in WebSphere Everyplace Access, the enterprise can also push important information to the mobile worker.

In this chapter, we introduce you to WebSphere Everyplace Access, the IBM mobile enablement platform. WebSphere Everyplace Access is designed to address the complexity and diversity of providing and managing mobile computing solutions for the enterprise. This chapter provides a high-level overview of WebSphere Everyplace Access, including:

- The product architecture and how it supports the enterprise
- An overview of the main components of the product
- An introduction to the various services that are provided by WebSphere Everyplace Access

© Copyright IBM Corp. 2003. All rights reserved.

1.1 The Big Picture

WebSphere Everyplace Access (subsequently referred to as Everyplace Access) is a comprehensive product that provides end-to-end coverage for the enterprise mobile computing needs. Everyplace Access is middleware that enables the enterprise and business partners to create robust mobile computing solutions that extend enterprise resources such as business applications, business data and business information to the mobile worker. Everyplace Access consists of both the *infrastructure* intended to reduce the complexity of providing mobile computing solutions and the *services* needed to create the right mobile solution for the mobile worker and their particular needs.

Everyplace Access provides the connection between the mobile client and the enterprise environment. Figure 1-1 on page 3 provides the logical view of the overall mobile computing runtime environment. The figure shows the following major areas:

- **Mobile client devices**: Everyplace Access supports a variety of mobile devices including various personal digital assistants (PDAs) and cell phones. Everyplace Client, included with Everyplace Access, provides a device environment for creating robust mobile solutions.
- **Network (connectivity)**: Everyplace Access provides service across the majority of mobile computing network models, such as a wired and wireless infrastructure and local or wide area wireless networks.
- **Load Balancing**: An optional complementary product provided by IBM (WebSphere Edge Server) or a third party that improves performance and scalability of the overall mobile computing environment. WebSphere Application Server (part of the infrastructure of Everyplace Access) provides additional capabilities to create multiple instances of the components to improve availability and scalability.
- **WebSphere Everyplace Access**: The server-based portion of the product that provides the access point to the enterprise resources, such as PIM and e-mail, data (through connected access or data synchronization), offline browsing and offline forms entry, and business data synchronization.
- **Enterprise Resources**: These include business applications, Web content, and business data and information that are used to run the business and support business processes. Access to these resources can help the mobile worker perform their business activities more efficiently and effectively.

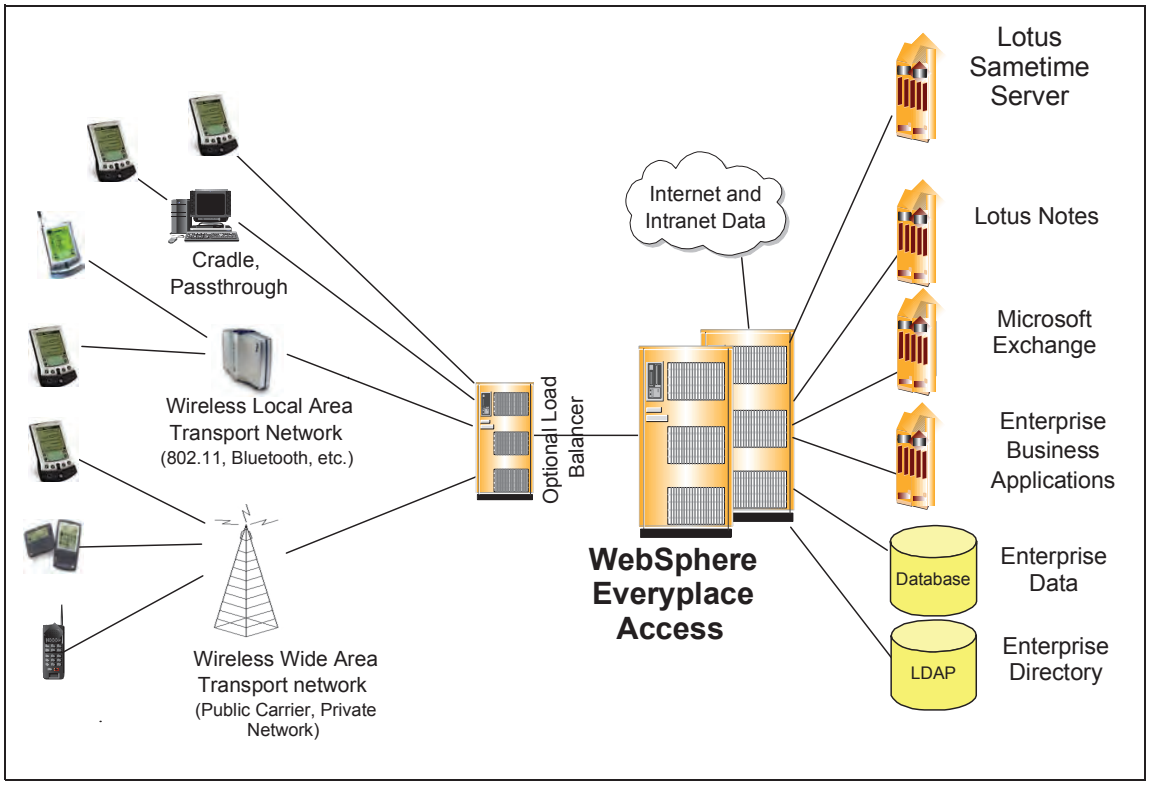


Figure 1-1 Logical view of mobile computing with WebSphere Everyplace Access

By being the access point to the enterprise, Everyplace Access addresses the complexity of mobile computing within the Enterprise environment and simplifies the overall runtime environment. The logical view in Figure 1-1 shows Everyplace Access in support of the enterprise and identifies it as providing the consistent access point for the mobile users regardless of the mobile device, the network or the type of request or interaction with the enterprise that is needed.

Everyplace Access follows the IBM software development ground rules of incorporating open standards-based technology into the product and reusing existing technology wherever appropriate. Both IBM's WebSphere Application Server and WebSphere Portal technologies provide the foundation for Everyplace Access. These products are built using open standards and open technology, such as XML and Java. Mobile computing technologies are changing fast and quickly evolving, which makes Everyplace Access the stabilizing factor in a very dynamic world. Everyplace Access accomplishes this by allowing the solution developer to write to its services, which minimizes the need to understand the details of the underlying technologies.

1.2 Inside the product

Everyplace Access comprises three major components necessary for delivering robust mobile computing solutions for the enterprise. The three components are:

1. **WebSphere Everyplace Access**: The server-based product that provides the infrastructure for interacting with the enterprise (from a mobile device) and the services which allow the enterprise to provide mobile solutions for mobile workers. It is the access point to the enterprise from the mobile device.
2. **Everyplace Client**: An optional mobile client environment which runs on the mobile device (PDA class devices) and extends the capabilities of the device. Everyplace Client supports the mobile device when operating in a connected or occasionally connected mode.
3. **Everyplace Toolkit**: A plug-in for WebSphere Studio (WebSphere Studio Site Developer Advanced or WebSphere Studio Application Developer) that provides the developer with the tools needed to create mobile solutions. Associated with the Everyplace Toolkit and WebSphere Studio is WebSphere Studio Device Developer with tools necessary to build a mobile device-based application and Mobile Application Builder used to create DB2e-based applications.

These three components give Everyplace Access the ability to provide an end-to-end mobile computing solution environment by providing the runtime elements both at the server and at a mobile device and the development tools necessary to create mobile solutions. Each of these components will be discussed in the following sections.

1.2.1 WebSphere Everyplace Access

As stated above, WebSphere Everyplace Access provides the access point to the enterprise for mobile devices. Figure 1-2 on page 5 shows the high-level architecture of WebSphere Everyplace Access (server side) and the services provided. Everyplace Access builds on the services provided by both WebSphere Application Server and WebSphere Portal technology and provides the additional services needed for mobile computing solutions. Everyplace Access uses many common services provided by the Portal technology as shown in the Page Aggregation and Portlet Container & Services boxes within Figure 1-2 on page 5.

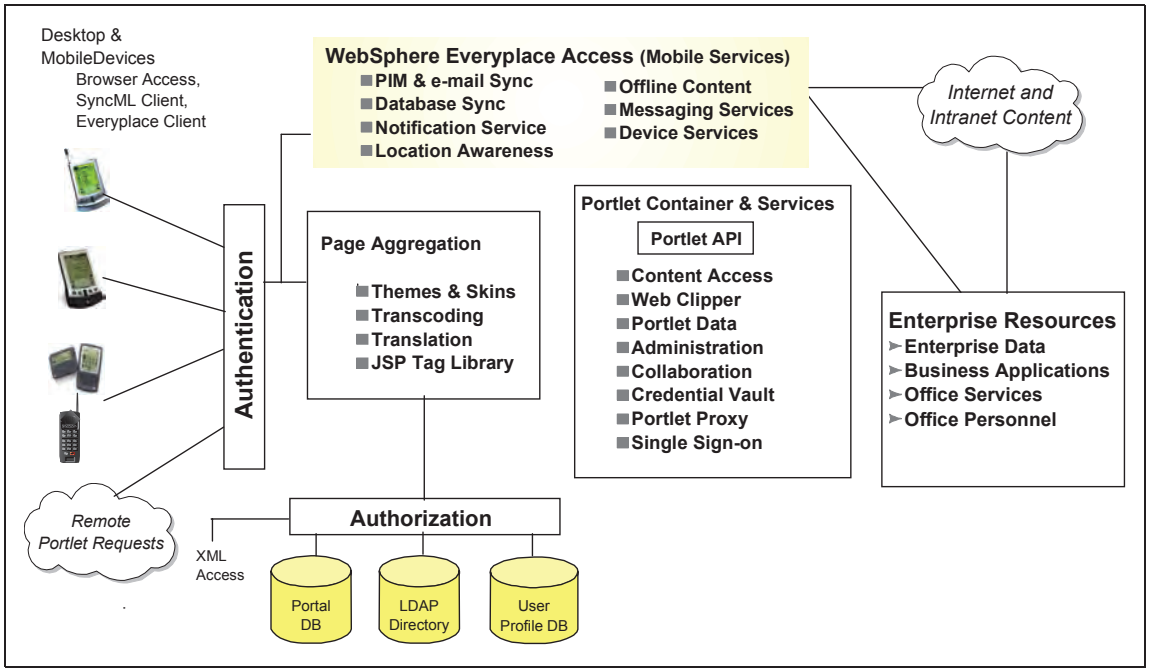


Figure 1-2 WebSphere Everyplace Access server

Everyplace Access provides a wide range of server-side functionality that includes:

- Basic Portal services for execution of mobile-enabled applications, including content access, Web clipper, portlet data, collaboration, Credential Vault, portlet proxy and single sign-on.
- Administration that allows the management of mobile users (including their profiles and preferences), mobile devices and the mobile solutions that make up the mobile computing environment.
- Services for connected or occasionally connected access to the enterprise.
- Browser-based access that is customized to the target device.
- Synchronization of PIM, e-mail, JDBC database data, Web content and simple forms.
- Fundamental mobile enablement services that enhance the ability to deliver mobile computing solutions, such as:
  - Transcoding: Provides dynamic adaptation of Web and XML content to fit the device needs or user preferences
  - Themes and skins: Enhances the look and feel of the content and applications delivered to the device

  - Messaging Services: Support for mobile Sametime and other messaging mechanisms
  - Device Services: Support the management and maintenance of the mobile devices used within an enterprise environment
- Advanced mobile enablement services:
  - Intelligent Notification Services: Provides server-generated notifications to the mobile device when events occur that meet established criteria
  - Location Aware Services: An extension that can be incorporated into enterprise applications that provides device location and location-based service provider information access.

Everyplace Access relies on the authentication provided via the integrated Portal technology and the WebSphere Application Server (which is included in the software bundle). Tivoli® Access Manager and Netegrity Siteminder can be used in conjunction with Everyplace Access for additional authentication support. The integrated Portal technology provides authorization services such as users and groups management.

1.2.2 Portlets

Everyplace Access such as WebSphere Portal uses portlets as a fundamental building block for applications. Portlets are applications that are based on the J2EE container model. They run inside the Portal Container of the Portal, similar to the way servlets run within the Servlet Container for WebSphere Application Server.

Portlets are a special subclass of HTTPServlet that includes properties and functionality that allows them to run within the Portal Container. Even though portlets run as servlets, they do not interact with the browser directly. All communications back to the user from a portlet is accomplished through the aggregation modules.

Many of the capabilities within Everyplace Access are implemented as portlets. Everyplace Access provides various productivity portlets that provide commonly used capabilities. Also, various portlets are available in the portlet catalog, which contains various portlets that can be used in building your enterprise solution.

For more details on portlets, see Chapter 6, “Mobile application development using portlets” on page 129.

1.2.3 WebSphere Everyplace Access services

WebSphere Everyplace Access provides a wide variety of services that are available for creating robust mobile solutions that fit the enterprise and the mobile user's needs.

Administration

WebSphere Everyplace Access Administration provides centralized administration of information about the enterprise mobile environment. Administration is used to manage the overall environment, including users and groups, user customization, product administration, Portal content (online and offline Portal pages), Portal settings, security and access control, Web Clipping and portlets.

Administration provides user and group definitions that include portlets for users to register and manage their own account information. User profile information includes a users's name and user ID, preference information such as news topics of interest, preferred language, etc. A user may be a member of one of more groups. Group membership is used to determine the access rights to underlying services. For example, a user must belong to a group with permission to perform data synchronization, in order to be authorized to use data synchronization on their PDA.

For more details about administration see Chapter 2, "Administration" on page 25.

PIM and e-mail access

Everyplace Access supports both synchronization of PIM and e-mail information to PDAs and SyncML-based devices and support for online (connected) viewing of e-mail and PIM information. Everyplace Access provides access to Lotus Domino Server 5.0.10 or later and Microsoft Exchange Servers 5.5 and 2000. For browser-based access, the information is tailored for the target device to improve the user experience.

Everyplace Access provides all the fundamental e-mail capabilities. With the latest release, the following have been added:

- ***Filtering:*** Allows the user to specify what e-mail gets delivered to the mobile device by selection criteria such as From, Subject, and Priority. Filtering helps limit the e-mail sent to the mobile device to those items that fit the selection criteria.
- ***Attachments***: Allows the user to view and manage attachments on their mobile device. This includes adding attachments of any file type to new e-mail documents created on the mobile device. E-mail with attachments can also be forwarded from the mobile device.

- ***Invitations***: Allows the user to accept, decline, or reschedule meeting requests and initiate meeting requests on the mobile device.

PIM support includes access to contacts, tasks, to-do lists, calendars and memos. PIM and e-mail are also available using native device mail and PIM services and is enabled on the Everyplace Client and other syncML compliant clients.

PIM and e-mail synchronization

The Everyplace Synchronization Service (ESS) provides synchronization of PIM and e-mail data between the mobile device and the mail server. ESS interfaces with both Microsoft Exchange 2000 and Microsoft Exchange 5.5 servers and Lotus Domino Version 5.0.10 and later.

ESS relies on SyncML data synchronization (DS), which is the standard for synchronization of remote data and personal information across different networks, platforms and devices. SyncML is sponsored by the leaders in the mobile computing industry, including Ericsson, IBM, Lotus, Matsushita, Motorola, Nokia, Openwave, Starfish Software, and Symbian, and is supported by the leading wireless providers. SyncML, an XML-based protocol, defines the structure and format of SyncML messages exchanged between the mobile device and the server. Each SyncML message is a well-formed XML document that contains the business information and a set of related SyncML commands (add, delete, replace, and so on).

ESS supports the synchronization of e-mail data and PIM information, such as contacts, calendars, tasks, to-do lists, and memos with the IBM Everyplace Client and other SyncML clients. Synchronization is profile driven. The profile defines the mail server the user intends to exchange data with, the user credentials to access that server, details on how conflicts are to be resolved during the synchronization process, data filtering, and any user synchronization preferences. A user may have multiple unique profiles for each device they wish to synchronize to the mail server.

E-Mail filters allow the user to determine which e-mail items will be delivered to their mobile device. The user can configure synchronization preferences to define the synchronization criteria. These preferences are used by ESS to determine what mail will be synchronized to your mobile device. By using the filters, a user can better control and manage the mail that is delivered to his or her mobile device.

ESS and SyncML DS are covered in some detail in Chapter 22, “PIM and e-mail synchronization with Domino Server” on page 711 and Chapter 23, “PIM and e-mail synchronization with Exchange 2000” on page 809.

Data synchronization

Everyplace Access enables data synchronization of JDBC-compliant database data to the DB2® Everyplace database, which is part of the Everyplace Client located on the mobile device. DB2 Everyplace has been incorporated into Everyplace Access. DB2 Everyplace is composed of two runtime components:

- DB2 Everyplace Sync Server: Supports the synchronization of data between the mobile device and any server-based JDBC-compliant database.
- DB2 Everyplace database: Provides a small footprint relational database located on the mobile device. Refer to “Everyplace Client” on page 14 for more details.

Data synchronization supports WBXML, the DB2 Everyplace-defined XML-based data representation of the data, which is not currently a SyncML standards-based markup language. The data can be compressed and encrypted before the data stream is transmitted between the mobile device and the server.

For more details on data synchronization, refer to Chapter 20, “Synchronizing with DB2 databases” on page 645.

Intelligent Notification Services (INS)

The Intelligent Notification Services are used to deliver urgent or late-breaking information to users on their preferred mobile device. This service allows the enterprise to proactively notify mobile workers (based on their subscription to particular notification applications) of important information and events. The notifications can be generated (when criteria is met) from a wide variety of sources such as e-mail, news feeds, data management events and directly from business applications. There are two general types of notifications:

1. Simple notifications: Messages that originate from other users or that come directly from applications. Examples would be personal messages or reminders.
2. Subscription-based notifications: Messages that are triggered by events to which users subscribe. For example, a subscriber could be notified when their favorite stock shifted 5 points.

To receive messages from the subscription-based notification, the user must subscribe to a notification by using the Everyplace Access Administration. During the subscription process, the user specifies the information source, their notification criteria, any preferences, and subscription settings.

Figure 1-3 on page 10 shows the subscription-based notification process. The figure is divided into three segments:

- **Information source:** The originating sources of information that may cause the generation of a notification. This information may come from a variety of

sources, such as Internet news services, the enterprise database which is triggered when specific data changes occur, or business applications when certain criteria is met.

- **Criteria and selection**: Once the notification manager has received information (from the information source), it determines if that information meets the criteria established by the subscriber(s) and issues a message to the dispatcher for further processing. Included in this section is the Administration that is used to define the user subscriptions, user preferences, and configuration information. This information is stored in the profile database. The context shown depicts the user context such as their online availability, location, or preferred device for receiving this notification.
- **Delivery**: The dispatcher is responsible for message formatting, using the subscriber's preferences and the context to send the notification using the right delivery channel. A wide variety of delivery channels are supported including Lotus Sametime, SMTP e-mail, Short Message Service (SMS) and the Message Center portlet (which is part of Everyplace Access) or a custom-defined channel.

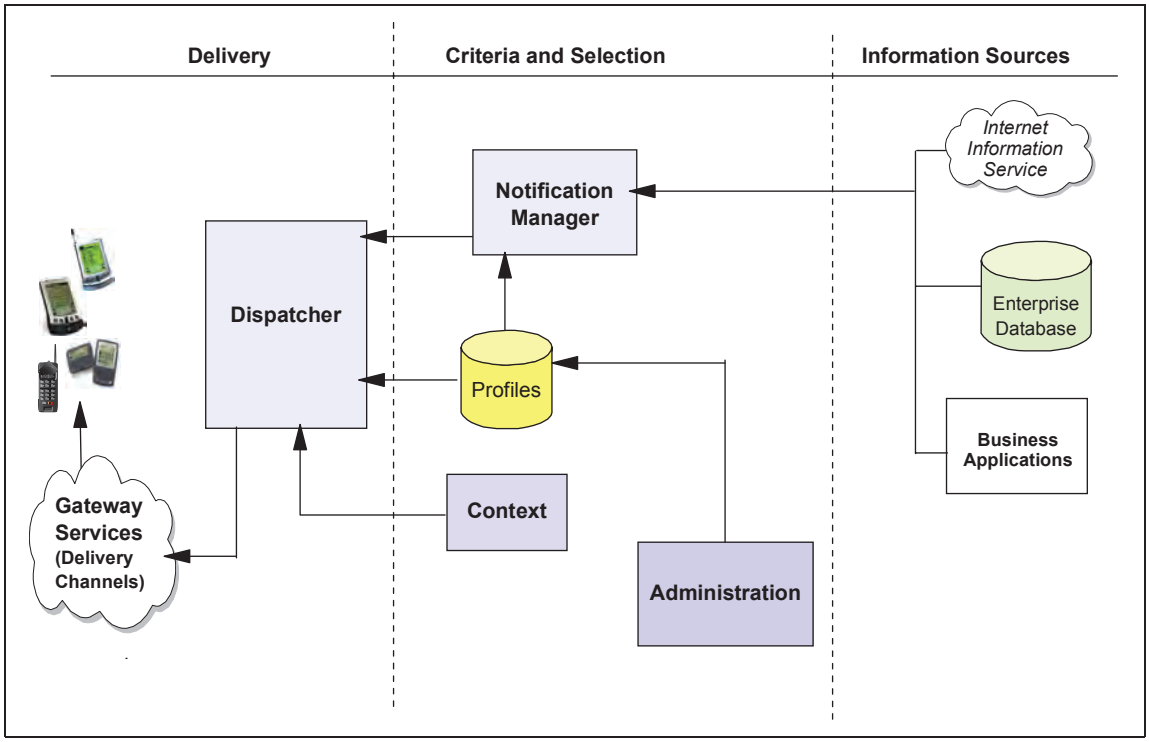


Figure 1-3 Notification Services

Developers can extend the INS framework by adding more delivery channels and services.

The Intelligent Notification Services are detailed in Chapter 25, “Intelligent Notification Services (INS)” on page 839.

Location Aware Services

The Location Aware Services (LAS) enables the enterprise to take advantage of the location information provided by mobile devices and interpreted by the service provider of location information. This service consists of a set of mobile device-enabled portlets and a rich set of Java APIs for accessing location-based services from enterprise or business partner-written applications. These APIs isolate the developer from the underlying service provider details that may vary between providers.

LAS provides support for:

- Geocoding: Using an address to determine its geographical coordinates
- Reverse geocoding: Converting geographical coordinates into the nearest address
- Mapping: Returning a map of a given location
- Routing: Returning directions in a road network from one point to another
- Directory: Finding points of interest and other entities around a given location
- Device position: Returning the location of a given subscriber’s device

LAS enables dynamic balancing among several providers at runtime, helping to ensure the applications get the most accurate information available. Location awareness includes the ability to select the best fit location provider at runtime based on the current situation, the ability to detect service availability, and the ability to roll over to another service provider if the primary provider fails. The service provides support for the leading worldwide service providers: Webraska, Go2map, Mapinfo, and Location Interoperability Forum (LIF).

The Location Aware Services is detailed in Chapter 27, “Location Aware Services” on page 971.

Server Initiated Actions

Server Initiated Actions (SIA) is a means to notify the client device of events that drive client actions, such as e-mail synchronization. Pocket PC Phone Edition devices can support SAI by accepting SMS messages that inform the client that synchronization should be performed. The SIA synchronization may be used for e-mail, PIM applications, databases, and offline Portal content and offline forms.

Pocket PC 2002 devices can receive these notifications via the Sametime Mobile Client.

Server Initiated Actions is detailed in Chapter 26, “Server Initiated Actions (SIA)” on page 957.

Device Services

Device Services provide the ability to maintain and deploy the Everyplace Client and software to the device. This server-based function provides support for IBM Everyplace Client. Device Services provides:

- Initial client deployment: Automating the deployment of Everyplace Client on mobile devices
- Device configuration: Maintains and manages the device configurations
- Software distribution: Ensures the mobile device has the right level of the application code by managing the software inventory on the device and delivering any new versions
- Inventory collection: Retrieves the current device inventory for review and reporting purposes

Also, the Everyplace Access device services can integrate with Tivoli Configuration Manager for an enterprise-wide management solution.

Device Services is covered in detail in Chapter 28, “Device Services” on page 1015.

Offline Portal browsing and forms

Everyplace Access provides the ability to preselect Web content provided by the Portal technology for download to the mobile device. The Web content can be viewed on the device while disconnected from the network. The Everyplace Access Administration provides the ability to create a specific offline Portal page that contains user-selected portlets that generate Web content and Web forms. The Everyplace Access content adaptation (via Transcoding Technology) capabilities can be used to dynamically adapt the selected content for the target device. The content associated with an offline Portal page is synchronized to the user’s mobile device, where the user can perform:

- Offline viewing of the Web content
- Offline entry and submission of simple Web forms.

Once the mobile device is connected to the server, the Web form data is sent to the server for processing by the server-based application.

For more details on offline content processing, refer to Chapter 13, “Offline Portal Content” on page 307.

Transcoding Technology

The Transcoding Technology included in Everyplace Access provides the ability to dynamically tailor and customize Web-based information to the needs of the mobile device and the mobile users. Transcoding is used to transform content based on the device's characteristics and constraints, user preferences, and network bandwidth. Content adaptation may be as simple as automatically transforming an image to an alternate format. or as involved as selecting specific Web content and converting that content to the markup required by the device. If the Transcoding Technology is used without any additional instruction, it will perform a "best-can-do" content adaptation of the original Web content.

To further customize the content sent to the mobile device, developers can use annotation commands that direct the modification of the incoming HTML document. The annotation commands can be stored in a separate .ann file or embedded within the HTML document. The eXtensible Stylesheet Language (XSL) can be used to format XML documents for a mobile device. Multiple transcoders (each with a specific task) can be invoked during the course of transforming a particular HTML or XML document.

Web Clipping

Web Clipping is the process of selecting specific content from an existing Web page for display on a mobile device. Web Clipping is a form of content adaptation that uses Transcoding Technology. The administrator or user creates a Web Clipping portlet and identifies the Web page associated with this particular clipper. As part of the creation process, the Web content is marked indicating the content to be removed from the content displayed on the mobile device. Web Clipping uses Transcoding Technology to provide device-aware content tailoring and support for the major device markup languages.

Fragmentation

Fragmentation is another transcoding mechanism used to support pervasive devices. Large units of content may need to be broken down into smaller units for transmission to the mobile device, because the mobile device does not have the capacity to receive large chunks of data. The fragmentation support within the Transcoding Technology automatically splits the content, manages the generated decks, and delivers these decks to the mobile device. The first deck is delivered to the device with the remaining decks cached on the server. Each deck is delivered to the device as needed.

Transcoding Technology fragmentation support is detailed in Chapter 17, "Portal-level transcoding" on page 511.

Security

Everyplace Access authentication is provided by WebSphere Application Server. Another alternative is to use a third-party authentication server that has a trusted association with the application server. Everyplace Access uses the Java Authentication Authorization Service (JAAS) as implemented by WebSphere Application Server.

For a third-party authentication server, trust between that proxy and WebSphere Application Server is accomplished by using a Trust Association Interceptors (TAI) to convert security information specific to the authentication proxy into a format that can be handled by WebSphere Application Server. Various TAI modules are included in WebSphere Application Server, such as Netegrity Siteminder and IBM Tivoli Access Manager (formerly Policy Directory).

1.2.4 Everyplace Client

Everyplace Client is an integrated complementary software package for the occasionally connected personal digital assistants (PDAs).

The Everyplace Access mobile environment in relation to the Everyplace Access server environment is shown in Figure 1-4 on page 15. The mobile client shows three possible client environments: a SyncML client (which in this case is shown using TrueSync Plus), the Everyplace Client, and a device browser (which allows browsing while in connected mode). The server environment reflects the Everyplace Access server. The enterprise includes the enterprise resources that are made available by using Everyplace Access services.

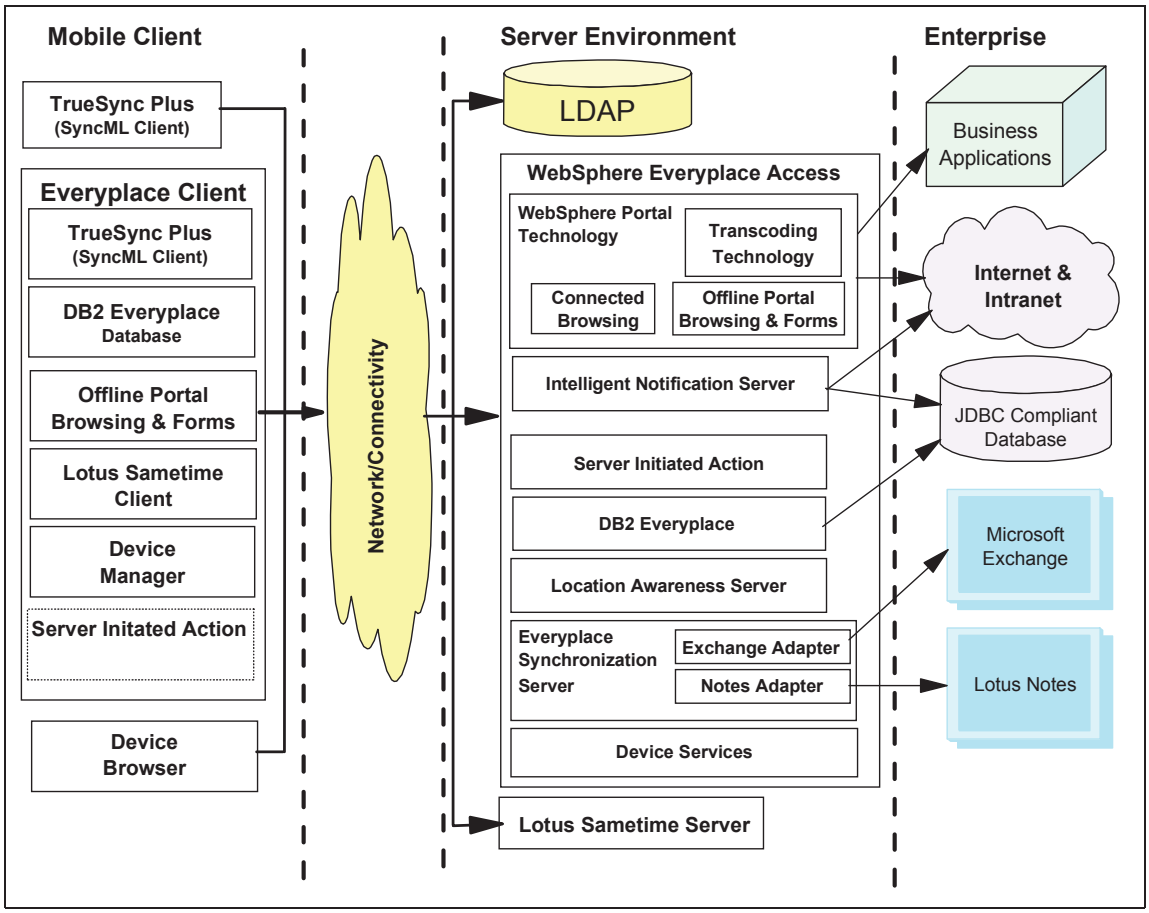


Figure 1-4 Mobile environment

Everyplace Client provides the following capabilities:

- **Connected browsing**: The native or installed device browser can be used to access the Internet or enterprise intranet Web content surfaced through portlets. These portlets can provide unique views for each device type.
- **DB2 Everyplace database**: Provides support for accessing the data within the local database by applications running on the device. Modifications and additions to the data in the local database can be synchronized with the enterprise database when a connection is available.
- **PIM and e-mail functions (SyncML client):** Allows the mobile worker to view and manage their e-mail and use the PIM applications on the local device. When a connection is available, synchronization is used to reflect any changes that have occurred both on the mobile device and on the server.

- **Offline Portal browsing**: The ability to view Portal content that has been downloaded to the mobile device. This allows the user to view Web content while disconnected from the enterprise.
- **Offline Form submission**: The ability to enter data and submit simple forms while disconnected from the network. The form data is held locally until the device can connect to the enterprise and transmit the form data for processing on the server.
- **Sametime Connect:** Provides mobile access to the server-based Lotus Sametime service. This allows mobile workers to communicate with colleagues in the office who are active on Sametime. This provides a transport-independent messaging capability and is available as a technology preview in Everyplace Access Version 4.3. Sametime Server Version 3.0 is a separately available product and is not included in the Everyplace Access product. Sametime Server configuration and administration is provided by the Sametime product. The Sametime Server Extensions for Mobile Access are also required.
- **IBM Device Manager**: Provides device management services. The IBM device agent is used to report, inventory, carry out configuration activities, and perform installation activities on the mobile device.

Tested devices

To ensure a particular device works with the Everyplace Client per the product's specification, the mobile devices are tested. As of this writing, the list of tested and certified devices are:

- Pocket PC 2002 Professional and Phone Edition (English) with Pocket Internet Explorer
  - Compaq iPAQ 3600*/3700/3800/3900
  - HP Jornado 56x, iPAQ 5400
  - Casio E200, NEC MobilePro 300, Audiovox Maestro™, Toshiba e740, Dell AXIM X5
  - Audiovox Thera (Verizon) and Siemens SX56 (ATTW)
- Pocket PC 2002 Professional and Phone Edition (Japan) with Pocket Internet Explorer
  - Compaq iPAQ 3800, HP Jornado 56x, Toshiba GENIO e550
- NTT DoCoMo 504i and 504is (Japan-only client) with limited e-mail/PIM clients
- Palm devices
  - Palm OS 3.5: Palm IIIc/Vx/m105, IBM WorkPad® c3
  - Palm OS 4.0: Palm m500/505, IBM WorkPad c500/505

  - Palm OS 4.1: Palm m130/m515/ Tungsten W
  - Palm OS 5.0: Palm Tungsten T
  - Palm OS 5.2: Palm Tungsten C
  - Palm devices with these browsers:
    - Eudora and Web Pro (text only)
    - Au-Systems (WML)
    - Palm Web Pro Browser
    - PalmSource Web Browser 2.0
- Sharp Zaurus devices:
  - Zaurus 5600

Important: To ensure you have the latest code, you should visit the WebSphere Everyplace Access support Web site at `http://www.ibm.com/software/pervasive/ws_everyplace_access/support/` to download the latest fix packs and code fixes. Apply these fixes prior to using the samples within this redbook.

Everyplace Client features per platform OS

Table 1-1 details the Everyplace Client features available on the various platforms that are supported as of this writing.

Table 1-1 Everyplace Clients feature support per device

Operating System	PPC 2002	PalmOS 3.5/4.0	PalmOS 4.1	PalmOS 5.0	PalmOS 5.2	NT DoCoMo CLDC/DoJa	Zaurus SL-5600
Integrated Shell (New browser-based UI)	Yes	No	No	No	Yes	No	Yes
DB2 Everyplace	Yes	Yes	Yes	Yes	Yes	No	Yes
SyncML Client	Yes	Yes	Yes	Yes	Yes	Yes[3]	Yes
e-Mail sync with attachment	Yes[6]	Yes (no attach)	Yes[6] (VersaMail 2.0 Support)	Yes[6] (VersaMail 2.0 Support)	Yes[6] (VersaMail 2.0 Support)	Yes (no attach)	Yes
PIM sync[1]	Yes	Yes	Yes	Yes	Yes	Yes (calendar & contacts)	Yes

Operating System	PPC 2002	PalmOS 3.5/4.0	PalmOS 4.1	PalmOS 5.0	PalmOS 5.2	NT DoCoMo CLDC/DoJa	Zaurus SL-5600
Invitation support [6]	Yes	No	No	No	No	Yes (view only)	No
Tivoli Device Agent	Yes	Yes	Yes	Yes	Yes	No	Yes
Lotus Sametime Client	Yes	Yes	Yes	Yes	Yes	No	Yes
Server Initiated Action [2]	Yes	No	No	No	No	No	No
Offline Portal Pages (offline portal browsing / offline Web forms	Yes	No	No	No	Yes (Browsing Only)	No	No
Service Oriented Runtime (also called Edgelet Platform)	Yes	No	No	No	No	No	Yes
NLS [4 and 5]	Grp 1 except KO	Grp 1 except PB,KO,ZH,TW	Grp 1 except PB,KO,ZH,TW	Grp 1 except PB,KO,ZH,TW	Grp 1 except KO,ZH,TW	Japan Only	Grp 1 except KO,ZH,TW

Notes:

1. PIM services: Calendar, contacts, to do list, memo
2. Server Initiated Action: Real time e-mail/PIM sync Sametime for PPC2002, SMS for Siemens SX56(HTC), PPC2002 Phone Edition, Sametime for others TBD
3. SyncML client runs on server and communicates with DoJa using proprietary protocol.
4. The languages that are not supported by the device platform cannot be supported by IBM, which are listed here as exceptions.

5. Customer has to supply Attachment views.
6. Invitation support such as accept, reject, calendar update

1.2.5 Everyplace Toolkit Version V4.3

The Everyplace Toolkit is a plug-in to WebSphere Studio Site Developer or WebSphere Studio Application Developer. The WebSphere Studio offering provides an integrated development environment (IDE) that supports WebSphere/J2EE application development such as Web services, Web applications, XML applications, portlets and much more. WebSphere Studio provides an end-to-end test environment for all the applications that can be created using the IDE.

The Everyplace Toolkit includes portlet development tools for the Everyplace Access environment. With the Everyplace Toolkit the developer can:

- Create a portlet and the associated JSP source
- Create the portlet directory tree appropriate to the type of portlet project selected
- Create and complete the portlet application descriptor
- Create “PDA” markup type content using the Portlet wizard
- Package the portlet application into a WAR file for deployment on the Portal Server
- Easily deploy portlets to the portlet container
- View the expected results using the appropriate device emulator when testing or debugging the application
- Debug portlets using the new debug environment, which includes simplified local Portal install and debug

The Everyplace Toolkit includes various samples to speed the developers understanding and use of the toolkit and the application artifacts it creates.

Associated with Everyplace Access are other tools that provide additional developer support. These tools include WebSphere Studio Device Developer (which can plug in to WebSphere Studio), which provides developers with an IDE for creating Java applications that run on the mobile device and use the WebSphere Micro Environment. Also included in this group is the DB2 Everyplace’s Mobile Application Builder, which is used to create form-based applications that access and update the local (on the mobile device) DB2 Everyplace database.

For details on the Everyplace Toolkit, see Chapter 7, “WebSphere Studio Site Developer and Everyplace Toolkit” on page 179.

1.2.6 Component products

In this section, we list the various products and components that comprise WebSphere Everyplace Access. There are three types of components:

1. Featured components: The primary services and components that make up Everyplace Access proper. This includes the server and client components.
2. Support components: Components that provide underlying support for the featured components.
3. Third-party components: Third-party software that is required to support some of the featured components. These components are not included in the packaging and must be purchased separately.

Featured components

The following products are featured in the product. Each component is briefly described below:

- Server components
  - WebSphere Everyplace Access Basic Services: Provides the core functionality and includes:
    - WebSphere Portal Version 4.2
    - Transcoding for Everyplace Access Version 4.3
    - Productivity and PIM portlets
    - Offline browsing and offline forms
    - Administration portlets for synchronization, INS, SIA, LAS
    - User portlets for INS
  - Everyplace Synchronization Server Version 1.3: Enables synchronizing data between the mobile device and the server, which includes data from Microsoft Exchange and Lotus Notes®.
  - DB2 Everyplace Version 7.2 (Fix Pack 7): Enables synchronizing relational database data stored in any JDBC-compliant database.
  - Everyplace Intelligent Notification Services Version 1.2.2: Allows users to subscribe to notification services and define the delivery channel, the selection criteria, and message sending criteria.
  - Device Manager Version 1.3.1: Device management service that helps the enterprise manage the various mobile devices connecting to the enterprise. Device Manager is used to identify, configure, inventory, and distribute software to any device supported by the enterprise.
  - Location Awareness Service Version 1.1: Provides a set of portlets that interact with location-based providers to obtain location information and provide a consistent interface for solution developers who need to incorporate this information into their mobile solutions.

- Client components
  - Everyplace Client Version V4.3: Provides device security and a single user interface to these client capabilities:
    - Lotus Notes Client Version 6
    - E-mail and PIM function (SyncML Client)
    - DB2 Everyplace (database)
    - Offline Portal browsing
    - Offline form submission
    - IBM device agent (device management)
  - TrueSync Plus Version 3.1 (included with Everyplace Client): A multi-point synchronization engine that provides synchronization of PIM applications (calendar, to-do lists and tasks, address book and contacts, and memos) between multiple information sources such as mobile devices and servers.
- Application Development Components
  - Everyplace Toolkit Version 4.3: A plug-in for WebSphere Studio Site Developer or WebSphere Studio Application Developer. The toolkit provides support for building mobile applications.
  - WebSphere Studio Site Developer Version 5.0: An integrated tool set that supports Web site development and management activities. It supports the J2EE specification and delivers integrated support for open Web standards including Java, JSP, servlets, XML, HTML, DHTML, JavaScript, rich media, and Web services tools.
  - WebSphere Studio Device Developer Version 5.5: An integrated development environment (IDE) for the creation and testing of applications deployed on mobile devices.
  - DB2 Everyplace Mobile Application Builder Version 8.1.2: Provides the tools necessary for developing mobile device applications using the DB2 Everyplace product.

Support components

The following products are support components included in the product. Each component is briefly described below:

- IBM Java Toolkit Version 1.3.1 Fix Pack 2: Required to complete product installation. The installation process will install it if it is not already present.
- DB2 Universal Database™ Version 7.2: A relational database management system used by Everyplace Access that stores component-specific data.
- IBM HTTP Server Version 1.3.19.3: An IBM-enhanced version of the Apache Web server that supports both the Secure Sockets Layer (SSL) Version 2 and SSL Version 3 protocols for secure connections.

- IBM Directory Services Version 4.1: A Lightweight Directory Access Protocol (LDAP) that runs as a stand-alone daemon. It is based on a client/server model that provides client access to an LDAP server. It provides an easy way to maintain directory information in a central location.
- WebSphere Application Server Advanced Edition Version 4.0.1 Fix Pack 4: Enables Web transactions and interactions with the deployment environment for e-business applications. It provides the foundation for WebSphere Portal technology and Everyplace Access.
- WebSphere Application Server Single Server Edition Version 4.0 Fix Pack 4: Provides the same core J2EE and Web services programming model with a simplified administration. In this case, it is used to support WebSphere Studio products and the Everyplace Toolkit in relation to portlet development and testing.

Third-party components

The following third-party products are associated with Everyplace Access and can be ordered separately as needed:

- Operating systems:
  - Windows 2000 Server with Service Pack 3
  - Windows 2000 Advance Server with Service Pack 3
  - AIX® Version 5.1 ML 1
- Web browser support:
  - Microsoft Internet Explorer Version 5.0, 5.5, and 6.0
  - Mozilla 5.0
  - Netscape version 6.1 and 6.2
  - Opera 5.0
- Microsoft Exchange 2000 for PIM and e-mail synchronization with Exchange data
- LDAP server (other than SecureWay® Directory that is provided), such as:
  - iPlanet Directory Server 5.0
  - Windows Active Directory 2000

System hardware requirements

The Everyplace Access server runs on these operating systems:

- Windows 2000 Server or Advanced Server with Service Pack 3
- AIX V5.1 with ML3 or later

1.2.7 Complementary products

IBM provides a set of complementary products that can be used in conjunction with Everyplace Access to fulfill the requirements for a particular enterprise environment. We have included a list of complementary IBM products. Some of these products could be replaced by third-party products that provide similar services, if so desired. The commonly used IBM complementary products are:

- **WebSphere Everyplace Connection Manager** (formerly known as the WebSphere Everyplace Gateway): Fortifies Everyplace Access by providing the underlying network connectivity for the enterprise. WebSphere Everyplace Connection Manager can be used to extend the number of network choices by providing secure data access to both WAP and non-WAP clients over a wide range of international wireless network technologies, as well as local area network (LAN) and wide area network (WAN) wire-line networks. WebSphere Everyplace Connection Manager is often used with Everyplace Access to provide a secure VPN-like connection to the enterprise. It also provides seamless roaming across different networks and different network types. It extends Everyplace Access's ability to handle large numbers of subscribers, improves response time on low bandwidth networks, and lowers network fees by providing data compression and header reduction. WebSphere Everyplace Connection Manager also protects the data with bi-directional user and server authentication and data encryption, ensuring that the mobile data is protected and secure.
- **WebSphere MQ Everyplace (MQe)**: Extends Everyplace Access by providing secure and dependable messaging between the mobile device and the enterprise. Applications written for the mobile device can create messages that are handled by MQe and sent to the server for processing. It is often used in conjunction with the DB2 Everyplace capability that is part of the Everyplace Client. MQe delivers once-only messaging so transactions are not lost and not duplicated between the mobile applications. MQe also provides peer-to-peer synchronous and asynchronous support, local and remote queue access, direct and indirect routing, encryption, nonrepudiation and authentication features that can be used to extend the existing capabilities within Everyplace Access. MQe interacts seamlessly with the WebSphere MQ family of products and provides any-to-any communication and business process integration. MQe is often used in situations where WebSphere MQ is already used as part of the enterprise solution architecture.
- **WebSphere Edge Server:** Can be used with Everyplace Access to distribute the application processing to the edge of the network while at the same time ensuring centralized administration and application control. It provides both load balancing and caching functions. With its enhanced load balancing features, it can improve server selection, load optimization, and fault tolerance.

- **IBM SecureWay Firewall:** For a production Everyplace Access deployment, firewalls are required. Usually two firewalls are included within the solution architecture. One is between the Internet (or third-party gateway) and all servers that communicate outside the enterprise (these are the edge of network servers). The second one is normally between these edge of network servers and all the back-end system servers in the enterprise that contain all the business databases, applications, directories, and so forth. The area between these two firewalls is called the demilitarized zone (DMZ).
- **WebSphere Portal offering:** Even though Everyplace Access relies on Portal technology to provide mobile solutions, to create a robust Portal for the enterprise a WebSphere Portal product should be used to support the connected users. There are three unique WebSphere Portal offerings, each of which can work in conjunction with Everyplace Access. The WebSphere Portal offerings deliver a single, universal point of access that is integrated, highly customizable and scalable. A Portal can be created to interact with various enterprise resources, such as business applications, existing content, people, and business processes. The WebSphere Portal offering can be used to support the always connected employees while Everyplace Access can be used to support mobile employees who may operate either in a connected or in an occasionally connection mode.

2

Administration

WebSphere Everyplace Access provides administration services that allow administrator and users to maintain the Everyplace Access runtime environment that supports the enterprise mobile environment. Administration consists of a centralized administration environment that supports such activities as managing users and groups, managing devices, managing portlet applications, and maintaining the mobile security environment.

In this chapter, we describe the main features of administration and show how to use it to accomplish many common activities. This chapter covers the following:

- Managing users and groups
- Creating places and pages
- Installing portlets
- Managing access control
- Selecting themes and skins

© Copyright IBM Corp. 2003. All rights reserved.

2.1 Getting started

We will be performing a variety of administration tasks. To log in, click the **log in** link shown in Figure 2-1. Log in with a user ID and password of `wpsadmin`.



Figure 2-1 Log in

WebSphere Everyplace Access Administration navigation is very similar to Portal Administration. Everyplace Access Administration (subsequently called Administration) provides a multi-tiered menu with the top-level tab menu bar controlling the overall navigation and selection process. This menu bar (top level) then determines the second-level menu options displayed.

The Portal Administration tab contains many of the options we will be using. Administration with the Portal Administration tab selected is shown in Figure 2-2.

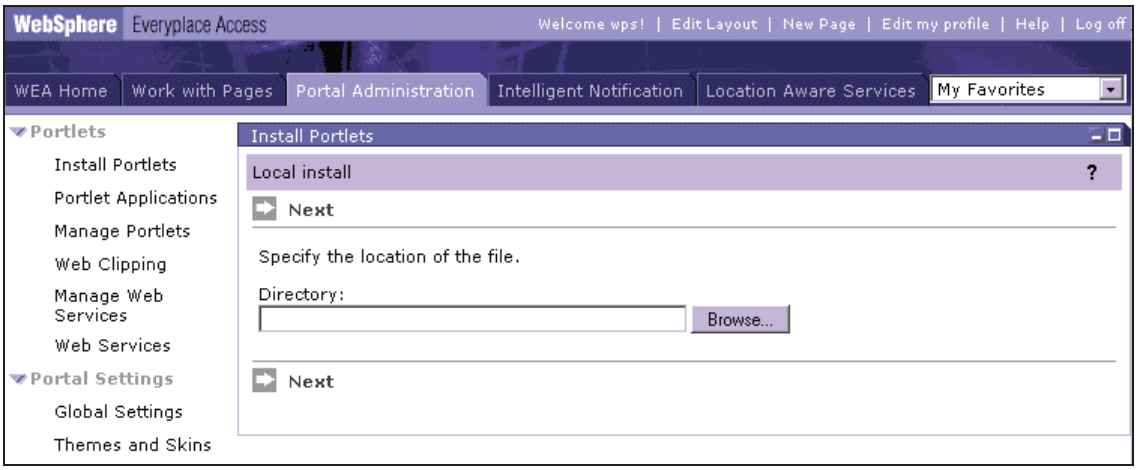


Figure 2-2 Portal Administration tab

The left pane (shown in Figure 2-2) contains the categories and the associated activities:

- **Portlets**: Install Portlets, Manage Portlet Applications, Manage Portlets, Web Clipping, Manage Web Services and Web Services.
- **Portal Settings**: Global Settings, Themes & Skins, Manage Clients, Manage Markups, Manage Search Index and Enable Tracing
- **Users & Groups**: Manage Users and Manage User Groups

- **Security**: Access Control List and Credential Vaults
- **Portal Content**: Manage Content Organizer and Content Organizer

2.2 Managing users and groups

Within the Portal Administration, select **Manage Users** (under Users and Groups), which allows you to create new users, edit user profiles, delete users and show ID information. The Manage Users window is shown in Figure 2-3.

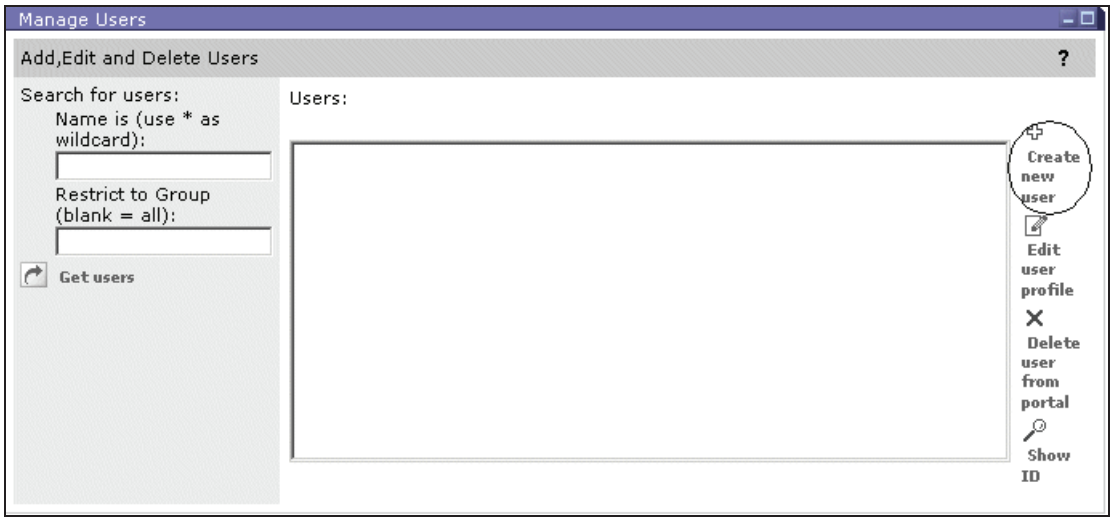


Figure 2-3 Manage Users window

The users we are creating are listed in Table 2-1 with the key information about each user. Notice that both Eric and LindaMay are part of the SalesGroup. Since the password must be at least five characters, we added the 01 to Eric and Huan. To be consistent, we added 01 to both Eric's and Huan's user IDs as well.

Table 2-1 Sample user list and groups

User ID & password	First Name	Last Name	Group
lindamay	LindaMay	Patterson	SalesGroup
eric01	Eric	Forestier	SalesGroup
huan01	Huan	Tran	

To create a new user select the **Create new user** option. Table 2-1 on page 27 shows the information for each new user. The Provide user information window is shown in Figure 2-4. Click **OK** after entering each user's information.Then click **OK** to finish this process.

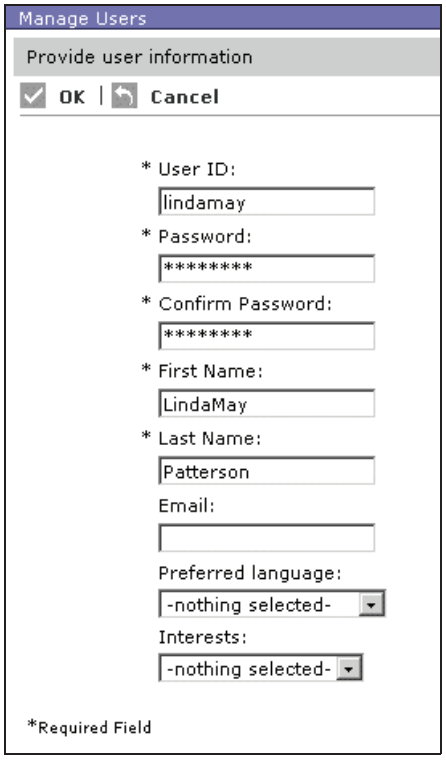


Figure 2-4 User information

Next we will create a group (SalesGroup) and add the users (Eric and LindaMay) to that group. Select **Manage User Groups** from the left pane. The Manage User Groups window is shown in Figure 2-5 on page 29. To create a user group, enter `SalesGroup` into the Group Name entry field, then click **Create Group**.

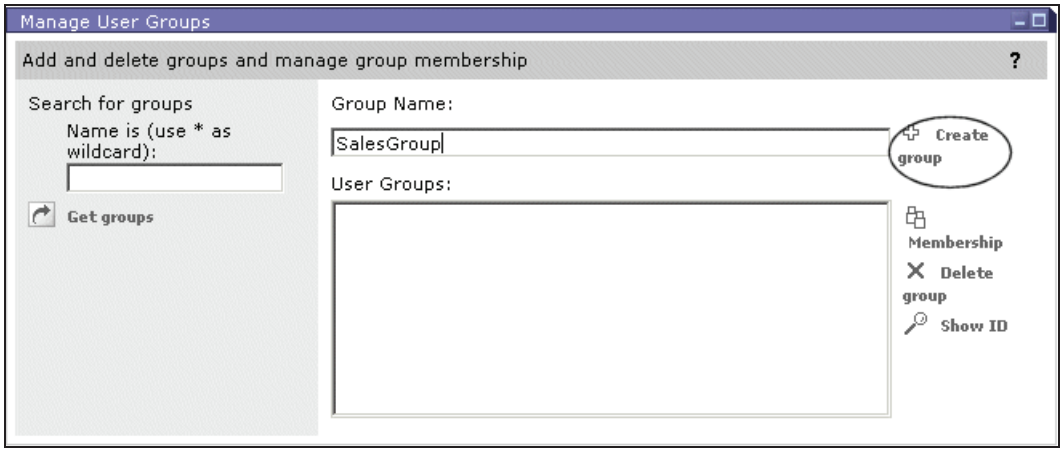


Figure 2-5 Manage User Groups window

The message `Group has been created` appears at the bottom of the screen as shown in Figure 2-6.

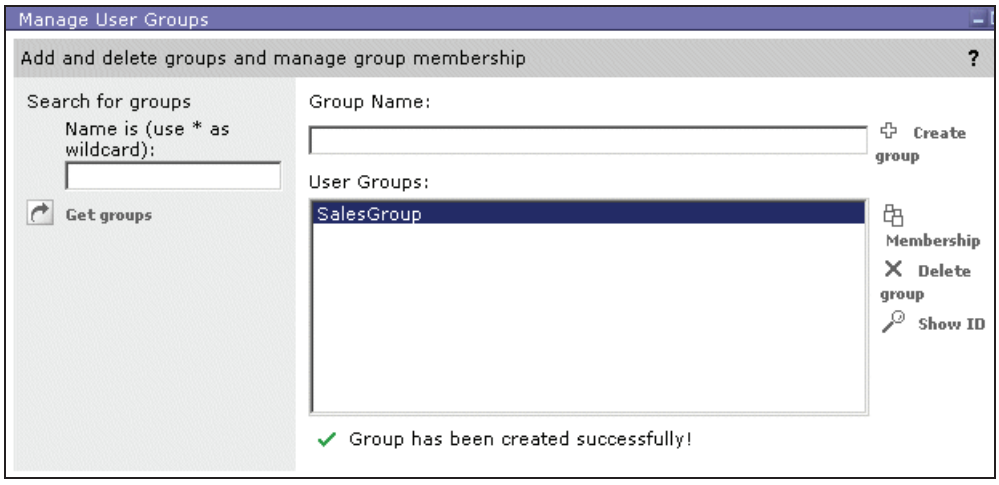


Figure 2-6 Group created

To add the users to the SalesGroup (group) click **Membership** as shown in Figure 2-7 on page 30.

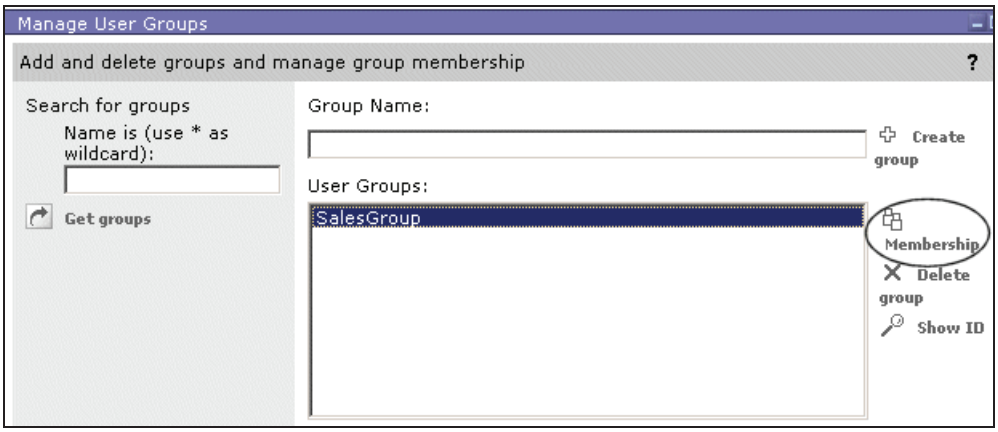


Figure 2-7 Group highlighted

Now, we need to find the users that we want to add to the group. This is done by selecting the **Add users to group,** with an asterisk (*) in the Name is entry field. This input is shown in Figure 2-8. Click **Go** to start the search process.

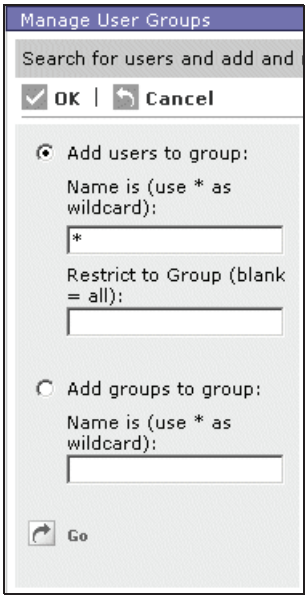


Figure 2-8 Find users

The search returns the users in the Search Results list as shown in Figure 2-9 on page 31. Highlight a user (in this case, Eric Forestier) and click **Add to Group**.

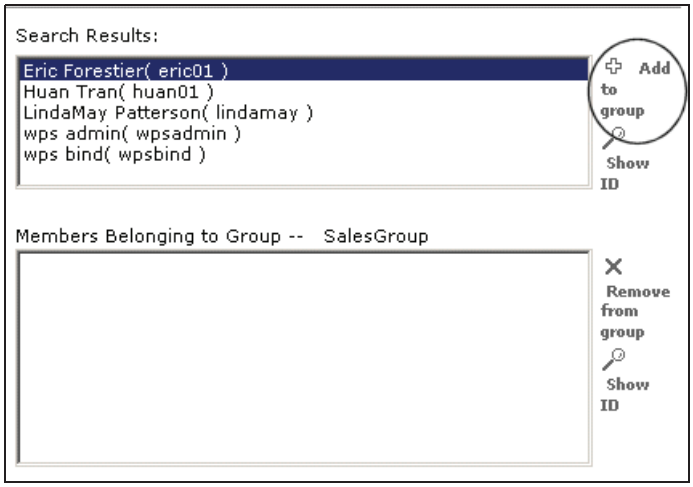


Figure 2-9 Search results

The user is added to the SalesGroup, as shown in Figure 2-10.

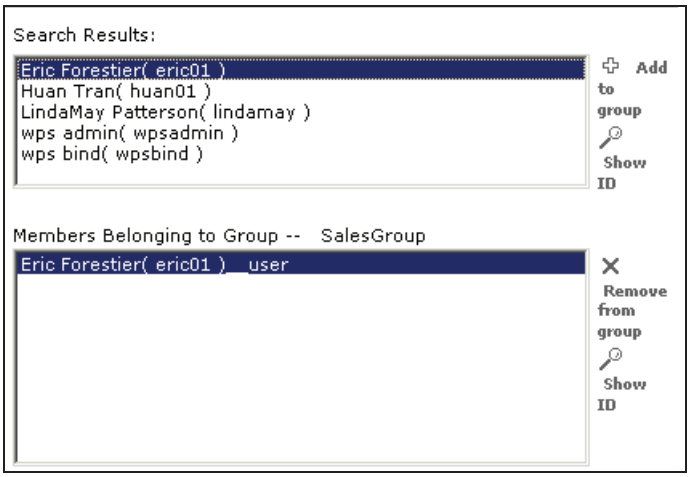


Figure 2-10 Member added to the group

Once Eric and LindaMay have been added to the SalesGroup, click **OK** to complete the activity.

To verify that the users have been added to LDAP, you can check the IBM SecureWay Directory (the directory service we installed with Everyplace Access). Select **Start -> Programs -> IBM Directory Server 4.1 -> Directory Manager Tool** to start the Directory Manager. Next, expand the Server folder to

display the Rebind entry, then click **Rebind**. Select the **Authenticated** radio button and enter the user ID and password of `wpsadmin`. The Directory Manager with the Rebind log is shown in Figure 2-11. Click **OK** to continue.

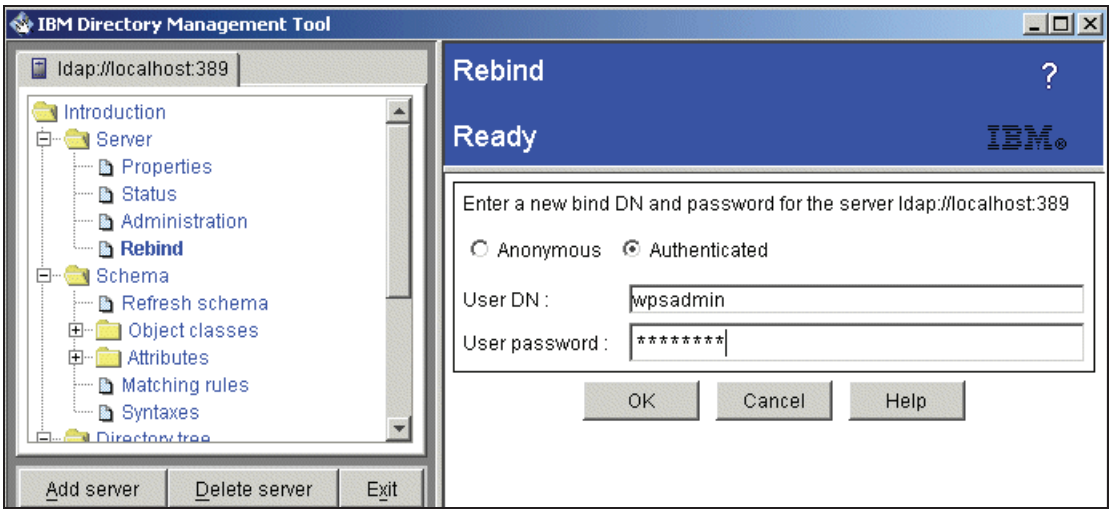


Figure 2-11 Rebind user

To view the users that we created, expand the Directory Tree folder and select the **Browse** tree entry. Under the ldap://localhost:389 entry, expand the tree dc=itso,dc=ral,dc=ibm,dc=com entry. Our users (lindamay, eric01 and huan01) and our group (salesgroup) are shown in Figure 2-12 on page 33.

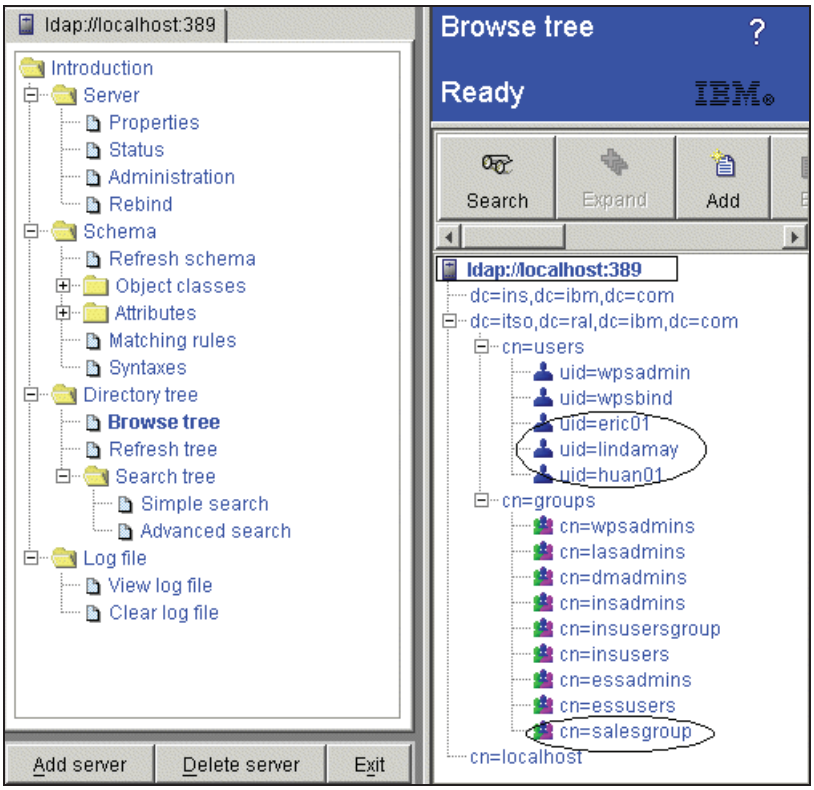


Figure 2-12 Browse tree

To view the details of a user or a group, double-click an entry. In this case we selected **eric01** as shown in Figure 2-13 on page 34.

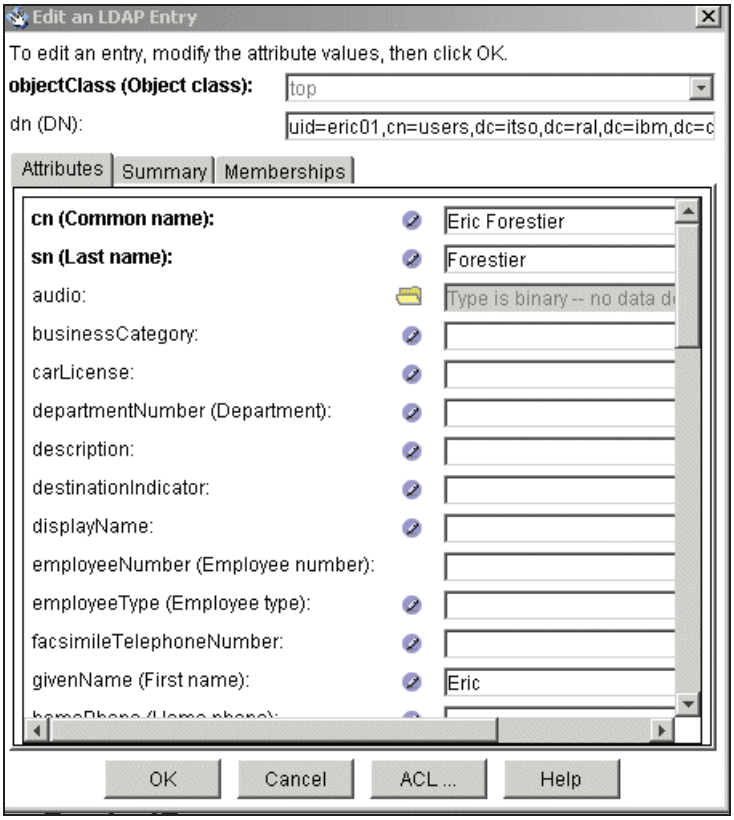


Figure 2-13 LDAP entry

2.3 Install portlets

A portlet is an application that displays content on a page. Portlets are the basic building blocks of a Portal. Each portlet application is developed, deployed, managed and displayed independently of each other. Everyplace Access provides a wide variety of portlets for use in your pages. You can obtain portlets in various ways, such as:

- Create your own portlets using WebSphere Studio with the Everyplace Toolkit plug-in.
- Include portlets from the portlet catalog found at the WebSphere Portal Web site:

  `http://www.ibm.com/software/info1/websphere/index.jsp?tab=products/portal`

  Select **WebSphere Portlet Catalog**.

- Select portlets for install from C:\WebSphere\PortalServer\install.

New portlets can be installed from either:

- A local file system by clicking **Install Portlets**
- A remote location by clicking **Manage Web Services**

We selected the **MyList** portlet from the portlet catalog and stored it on our server at c:\portletjar. The portlet file is named mylist.war. Because we are installing from the local file system, we are using the Install Portlet option. Select **Portal Administration -> Install Portlets** to see the Install Portlets window, shown in Figure 2-14.

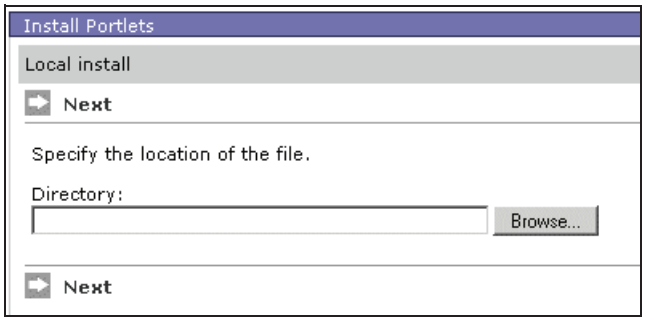


Figure 2-14 Install Portlets window

To find the WAR file, click **Browse** and navigate to C:\PortletJars\mylist.war as shown in Figure 2-15. With the WAR file displayed in the Directory entry field, click **Next**.

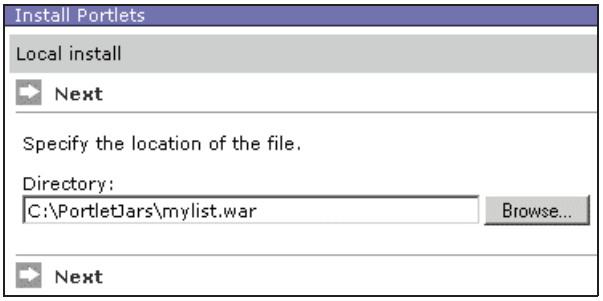


Figure 2-15 Install Portlets window with directory selected

Figure 2-16 on page 36 shows the portlet application and a list of the portlets that make up the application that was retrieved from the WAR file.

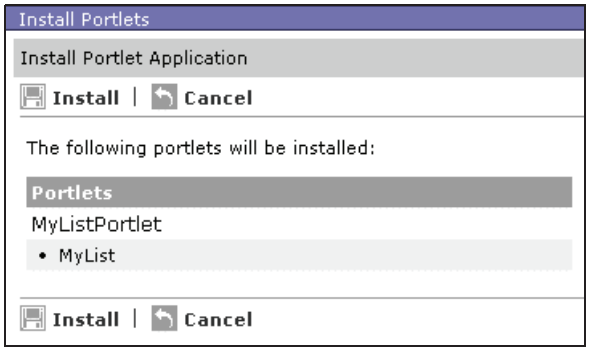


Figure 2-16 Install portlet list

To complete the install process, click **Install**. Both the installation of the portlet and activation of the portlet is accomplished at this time. Once the process is completed, you will see the message `Portlet successfully installed`, as shown in Figure 2-17.

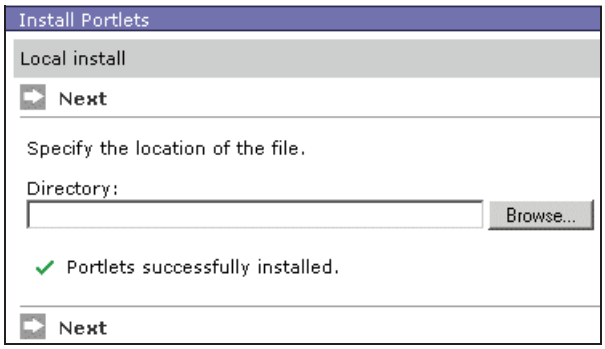


Figure 2-17 Portlet installed successfully

To verify that the portlet is now installed, click the **Portlets Applications** option under Portlets. The Manage Portlet Applications displays all the installed portlets in the Web Modules list. Highlight the **mylist.war** portlet (in the Web modules list) to determine the status of the portlet. The list of portlet applications belonging to the selected Web module shows mylist.war as active. This is shown in Figure 2-18 on page 37.

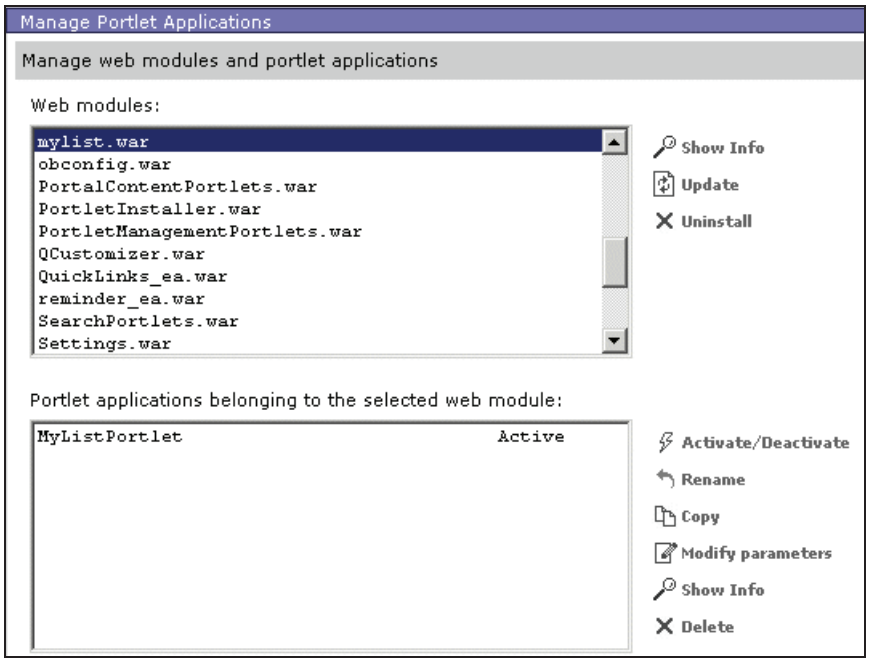


Figure 2-18 MyListPortlet active

Setting a PDA icon

You may be wondering how to change the icons associated with a portlet that is displayed on a PDA. The developer can determine the icon associated with a portlet by making an entry into the portlet.xml file that is located in the projects WEB-INF directory (visible in the WebSphere Studio development environment). Within the portlet's portlet.xml file, add a config-param with a param-name of `pda-icon` and a param-value of the icon (location and name). This is shown in Example 2-1.

Example 2-1 portlet.xml for icon

```
<config-param>
<param-name>pda-icon</param-name>
<param-value>/images/pda/myicon.gif</param-value>
</config-param>
```

When the portlet is installed, the administrator can define the portlet icon, if one has not been established by the developer. This is accomplished by doing the following steps with Portal Administration (once the portlet is installed):

1. Select the **Manage Portlets** option under the Portlets category.

2. Select the **Show all portlets** radio button and click **Go**.
3. The portlets are returned in the list. Select the portlet you want to modify (in this case the **MyList** portlet) and click **Modify parameters**, as shown in Figure 2-19.

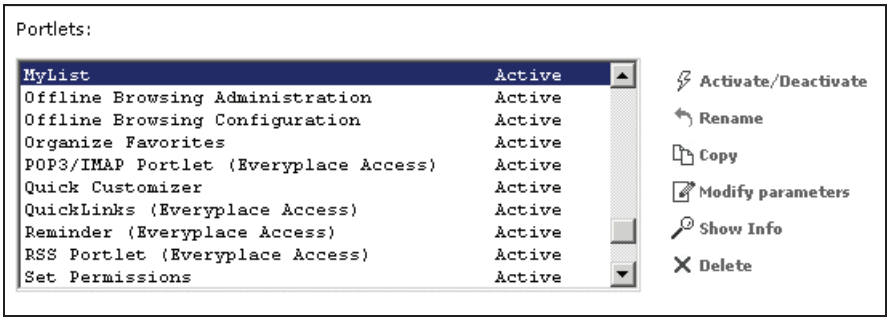


Figure 2-19 Modify parameters

4. Enter the Parameter value of `pda-icon` and the Value of `/images/pda/default.gif` (or your icon). Then click **Save**.

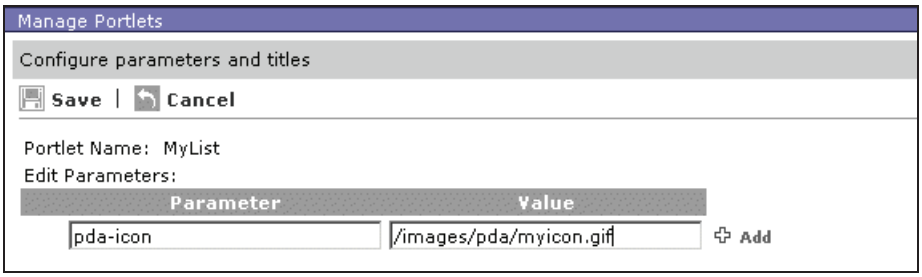


Figure 2-20 PDA icon

The icon must exist in the /images/pda directory as a .gif file.

2.4 Manage places and pages

A portal consists of places that have one or more pages, and each page contains one or more portlet applications. Think of a place as a collection of Portal pages. A Portal page displays content.

Both the administration and authorized users can construct portal places and pages. Places and pages are managed within the Work with Pages entry in the top-level menu bar. To see the options for creating and managing places and

pages, click the **Work with Pages** tab. The Work with Pages options are shown in Figure 2-21. There are two aspects to the page: the icons at the top (which are circled in Figure 2-21) and the main options, which are in a second-level menu. The icons from left to right provide these options:

- Create a place
- Create a label
- Create a page
- Create a URL

The list of options are:

- Edit My Pages
- Edit Layout
- Manage Places and Pages (which shows the existing places you can edit and manage places and pages)
- Set Permission
- Choose Skins
- Organize My Favorites

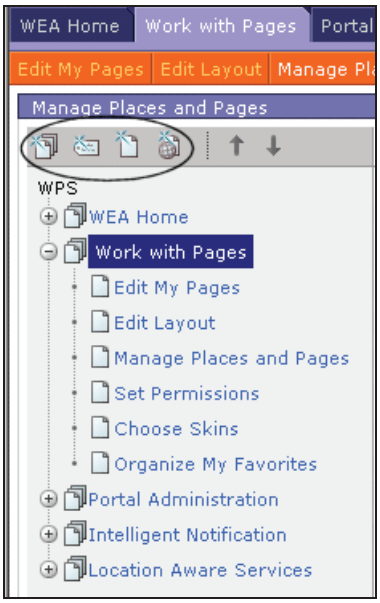


Figure 2-21 Manage Places and Pages window

Click the **Create a place** icon (the first icon in the list), which allows you to create a new place. To create a place, do the following steps, as shown in Figure 2-22 on page 40:

1. In the Administrative field, enter `MyPlace` as the name of your new place.

2. Select **Corporate** as the theme. You can click the eye icon to preview the theme.
3. Select all the markup languages. This indicates the markup language(s) your place will support.

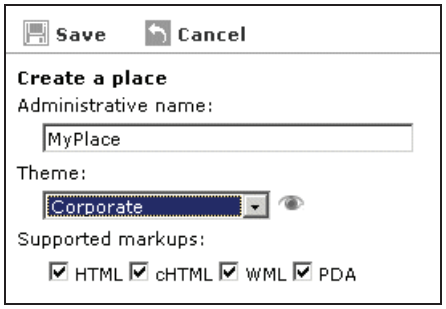


Figure 2-22 Create a place

To create MyPlace, click **Save**. The new place named MyPlace is automatically activated by default. You will see the new place, MyPlace, added to the Manage Places and Pages list (left pane), as shown in Figure 2-23.

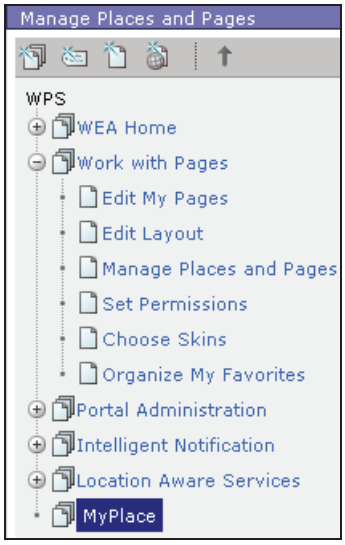


Figure 2-23 MyPlace

To create a new page for MyPlace, select **MyPlace** and click the **Create a page** icon (the third icon in the list). Perform the following steps to create a page, as shown in Figure 2-24:

1. Enter `Page1` in the Administrative name field to name the page
2. Select two columns for the number of columns.
3. Select all the markup languages. These will already be selected to match the Place markup language selections.

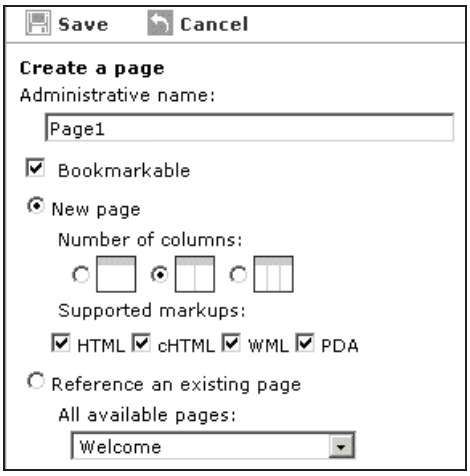


Figure 2-24 Create a page

To create the page, click **Save**. Now you have created a place MyPlace with a page Page1. Notice that the page is now included in the Manage Places and Pages list under MyPlace (this is in the left pane), as shown in Figure 2-25 on page 42.

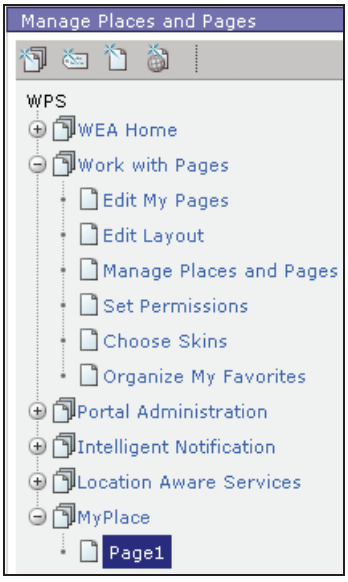


Figure 2-25 Page1 in list

The Edit Layout option is used to add the portlets to a page. Click the **Edit Layout** tab, expand the MyPlace list (click the + sign by MyPlace) and then click **Page1** to identify Page1 as the page to be edited. This is shown in Figure 2-26.

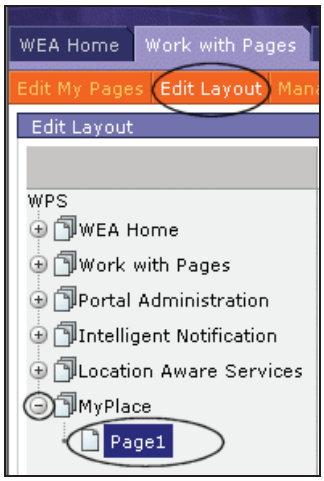


Figure 2-26 Edit layout

The previous steps identified Page1 for layout editing. Since we selected two columns for Page1, the layout reflects the two-column layout and is shown in Figure 2-27.

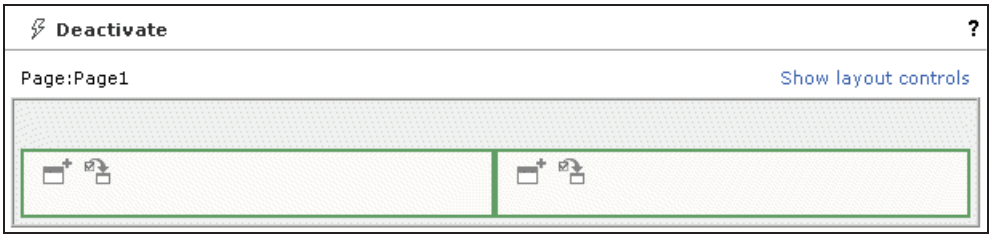


Figure 2-27 Page1 layout

In the first column of Page1, click the **Add Portlets** icon (shown in Figure 2-28).

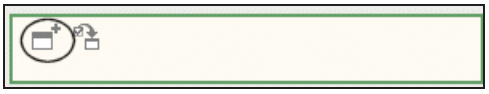


Figure 2-28 Page1 Add Portlets icon

The Edit Layout window (shown in Figure 2-29) allows you to determine the types of portlets you want to include in this page. Make sure **All Available** is selected and click **Go**.

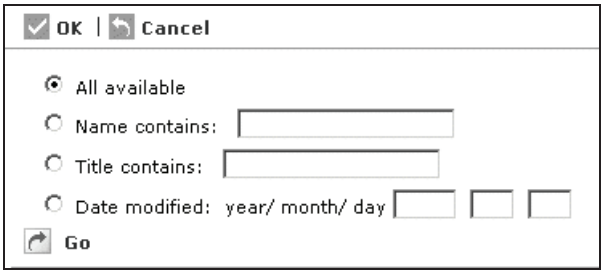


Figure 2-29 Determine portlets

A list of the installed portlets with their related information is provided in the result of the request list. For column 1 of Page1, we selected **QuickLinks** and **Reminder,** by selecting the check box beside them. This is shown in Figure 2-30 on page 44. To complete the selection process, click **OK**.

☐	Organize Favorites	Organize Favorites	favorites.	html wml
☐	Quick Customizer	Quick Customizer	Small medium business customizer	html
☑	QuickLinks	QuickLinks (Everyplace Access)	Simple QuickLinks portlet	html pda wml
☑	Reminder	Reminder (Everyplace Access)	Reminder portlet for recording brief text reminders	html pda wml
☐	Remove User Preferences	Intelligent Notification User Preference Administration	Remove Intelligent Notification user preferences	html
☐	RSS Portlet	RSS Portlet (Everyplace Access)	A portlet that displays RSS headlines	html pda wml

Figure 2-30 Select portlets for column 1

To add portlets to column 2 of Page1, perform the same steps selecting the **RSS Portlet**, **World Clock**, and **MyList** (remember we imported this portlet in 2.3, “Install portlets” on page 34) and then click **OK**. The completed Page1 layout is shown in Figure 2-31.

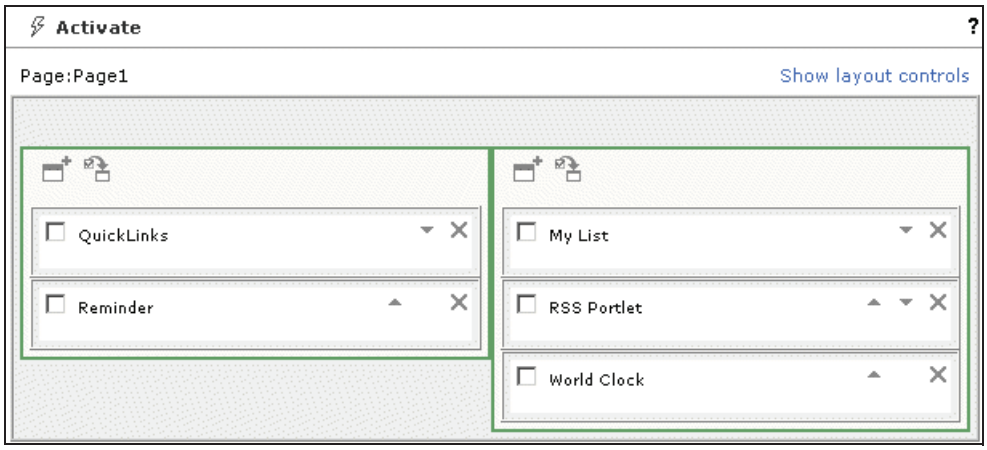


Figure 2-31 Complete Page1 layout

The portlets can be moved around in the columns by clicking the up or down arrow (depending on their current position), or deleted by clicking the **x** within each individual portlet container. The last step is to activate Page1. Click **Activate**.

You can look at Page1 by selecting the **MyPlace** tab. Page1 will appear as shown in Figure 2-32 on page 45.

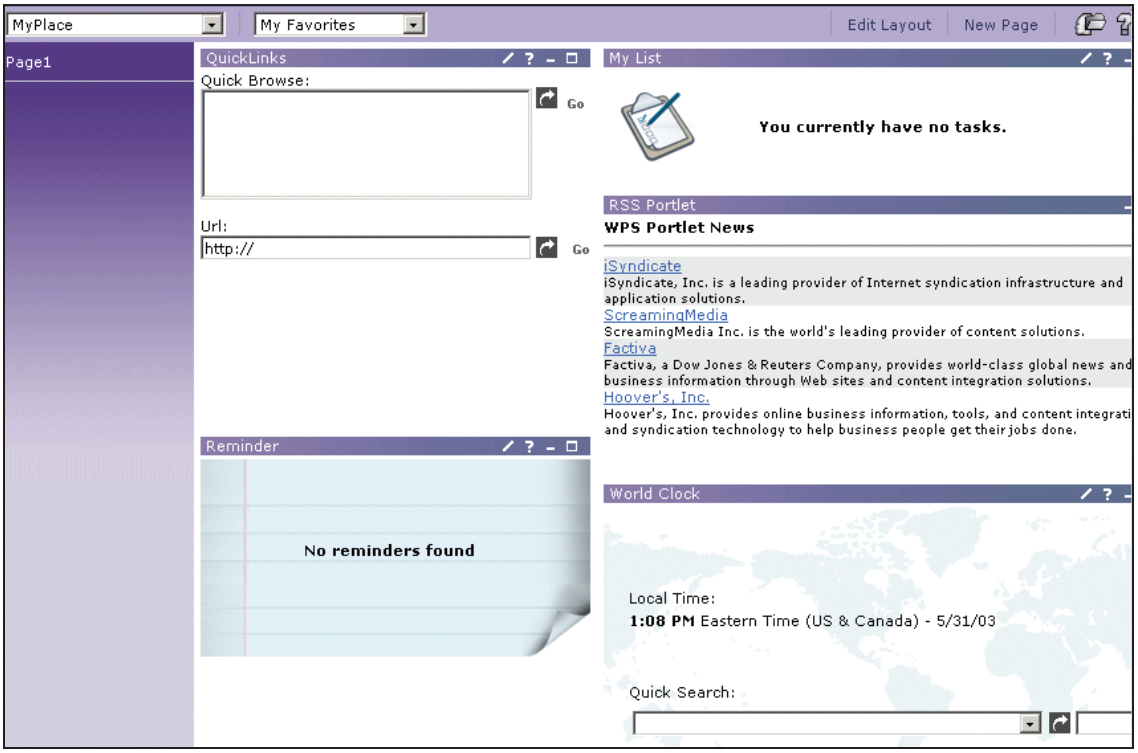


Figure 2-32 Page1 results

It is possible to reformat Page1 from a two-column page to a one-column or three-column format. Select the **Work with Pages** tab, then select **Edit Layout** and finally select **MyPlaces -> Page1**. Click the **Show Layout controls**, shown in Figure 2-33 on page 46.

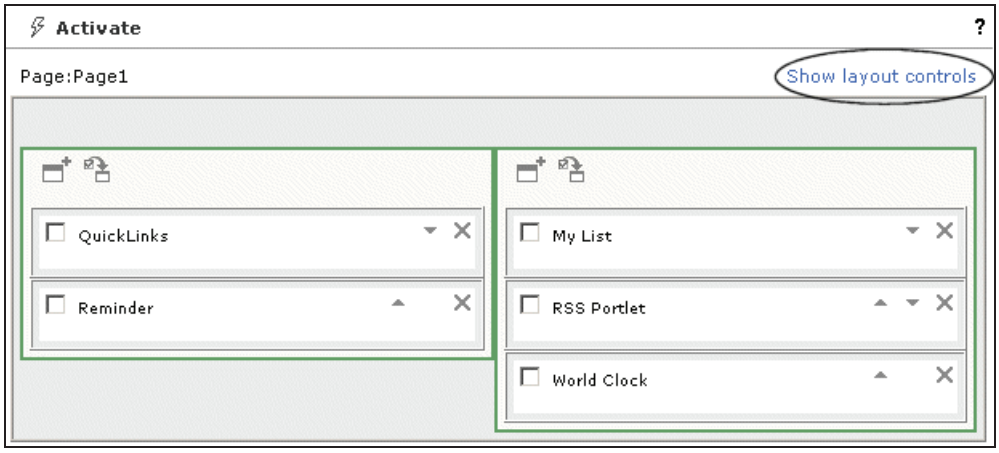


Figure 2-33 Show layout controls

To change from two columns to one column, do the following steps:

1. Move the portlets from column 2 to column 1. Check the box beside the portlets and click the **Move Portlets** icon in column 1.
2. Delete column 2. Click the **x** for the container and click **Yes** when asked if you want to delete the container.
3. Activate the page. Click **Activate**.

The new Page1 with only one column is shown in Figure 2-34 on page 47.

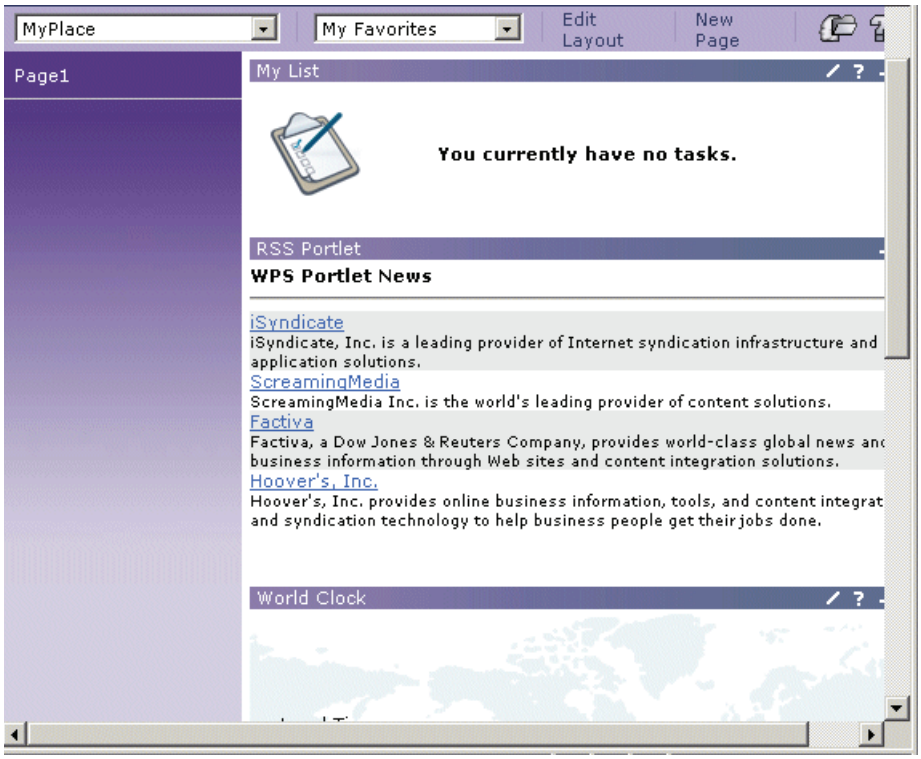


Figure 2-34 Page1 with one column

2.5 Managing access control

Access Control List (ACL) allows the administrator (or other designated users) to assign permissions to specific users and groups. Through ACL, these users and groups are given access (permission) to groups, portlets, places, and pages. Administration provides the tools for managing policies and resources by:

- Granting users and groups permission to places and pages
- Granting users and groups permission to specific portlets

Each Portal page is subdivided into one or more content areas. Each content area can contain one or more portlets. The administrator or an authorized user can control whether others have authority to manage and access pages and the portlets on a page. *Permissions* is the term used to define the control of these settings.

Access control can be used to create policy permissions that enable groups and users to access resources and even allow them to manage these resources.

Access control is part of Portal Administration. Click the **Portal Administration** tab, then within the Security category select **Access Control List**. The Access Control List is shown in Figure 2-35. We are going to allow our users access to MyPlace and Page1. Make sure the **Selected a group or user to assign permissions** radio button is clicked and click **Get groups and users**.

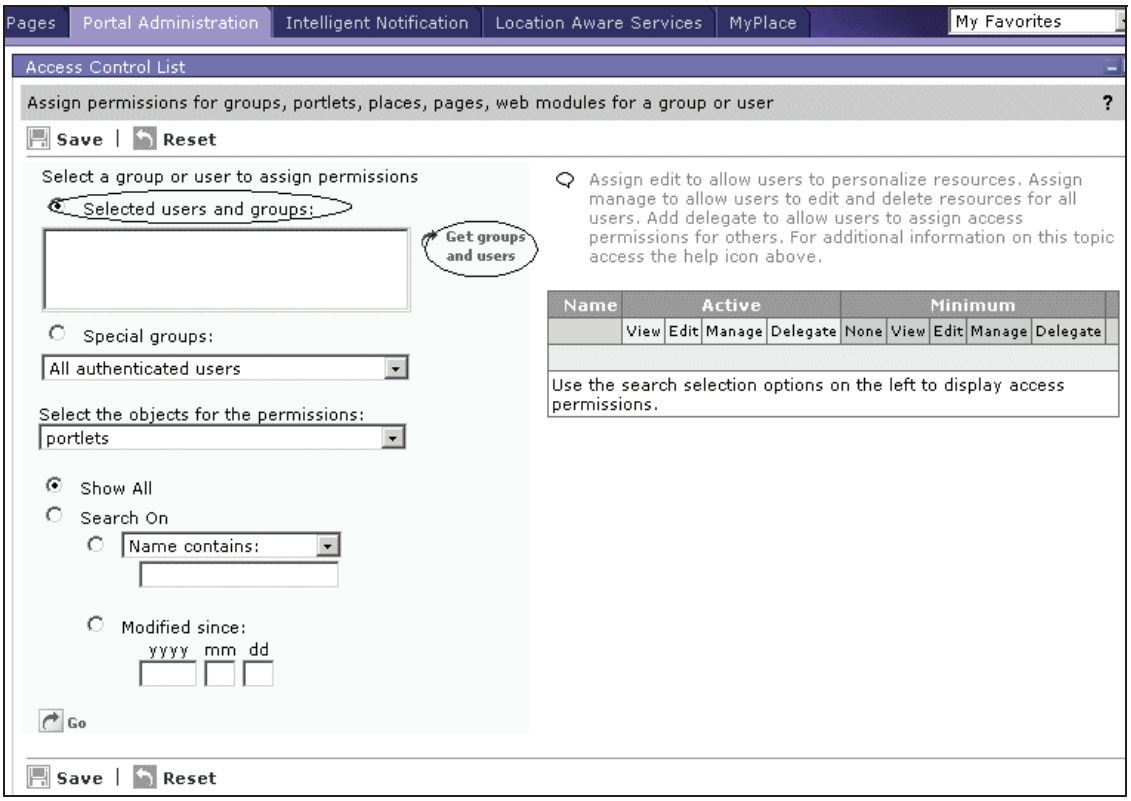


Figure 2-35 Access Control List

Next, you are presented with the Get Groups and users window, where you can find the user or group you want to work with. Make sure the **Search for groups** radio button is selected and enter `Sales*` in the Name entry field, then click **Go**. This is shown in Figure 2-36 on page 49.

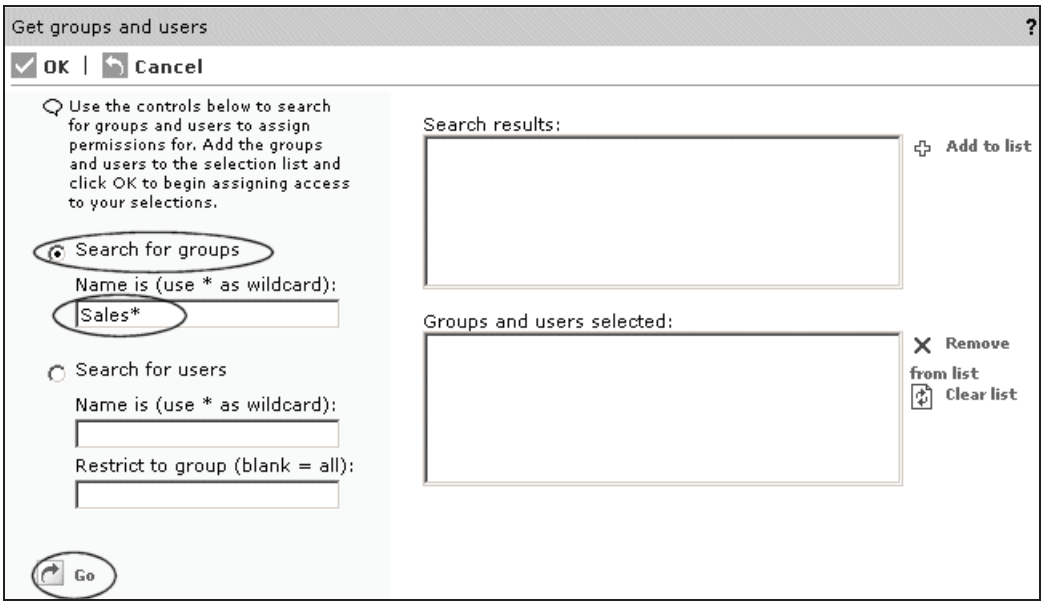


Figure 2-36 Get Groups and Users for SalesGroup

The Search results show the SalesGroup. Select **SalesGroup** and click the **Add to list** icon (shown in Figure 2-37).

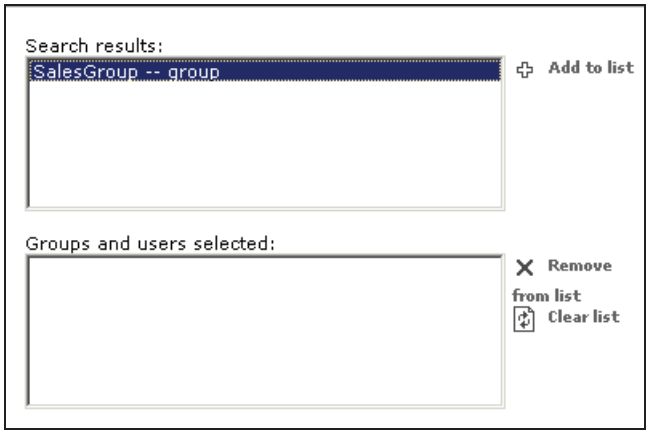


Figure 2-37 Add to list

The SalesGroup is then added to the list of groups and users selected. Next click **OK**, which returns you to the previous window, where the Access Control List has the SalesGroup identified, as shown in Figure 2-38 on page 50.

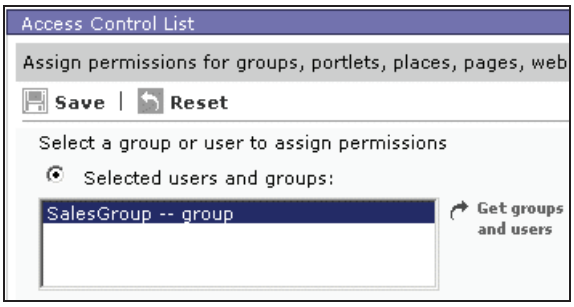


Figure 2-38 ACL with SalesGroup

Now we want to associate our group (SalesGroup) with MyPlace and Page1. To find the places do the following steps (as shown in Figure 2-39 on page 51):

1. Make sure **SalesGroup** is highlighted.
2. Select **places** in the Select the objects for the permission drop-down list.
3. Select **Search on** to limit the places returned.
4. Enter `place` in the Names contains entry field to identify the places we want returned.
5. Click **Go**.

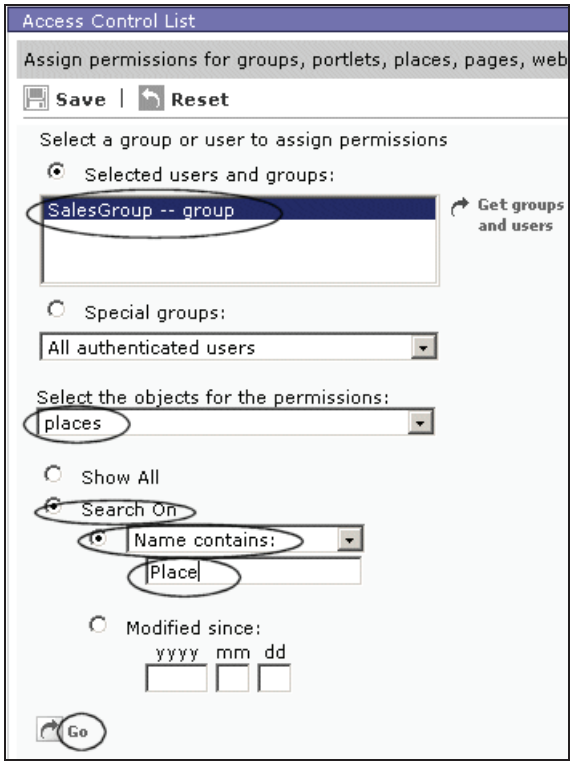


Figure 2-39 ACL select places for place

The search returns the places with the word `place` in the name. In this case, the search returned MyPlace and Page1 (because its is within the MyPlace page group). To give the SalesGroup permission, select the **View** radio button for both MyPlace and Page1, as shown in Figure 2-40 on page 52. Click **Save** to save these changes.

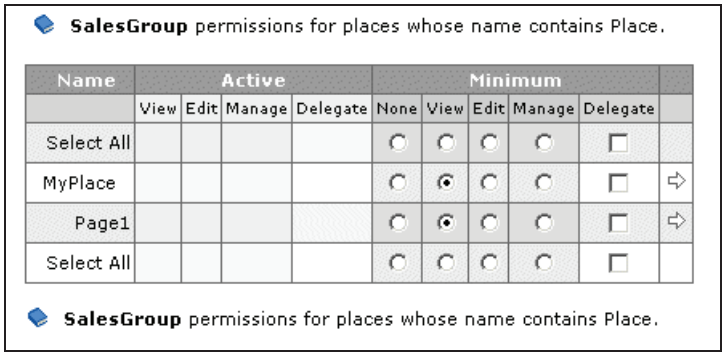


Figure 2-40 SalesGroup permission

Verify that the MyPlace and Page1 entries are checked in the View column under the Active column.

Now we must make sure the SalesGroup has access to the portlets on Page1. First we must find the portlets and associate them with the SalesGroup. This is accomplished by selecting **portlets** in the Select the objects for the permissions drop-down list. Click the **Show All** radio button and then click **Go**. This is shown in Figure 2-41 on page 53.

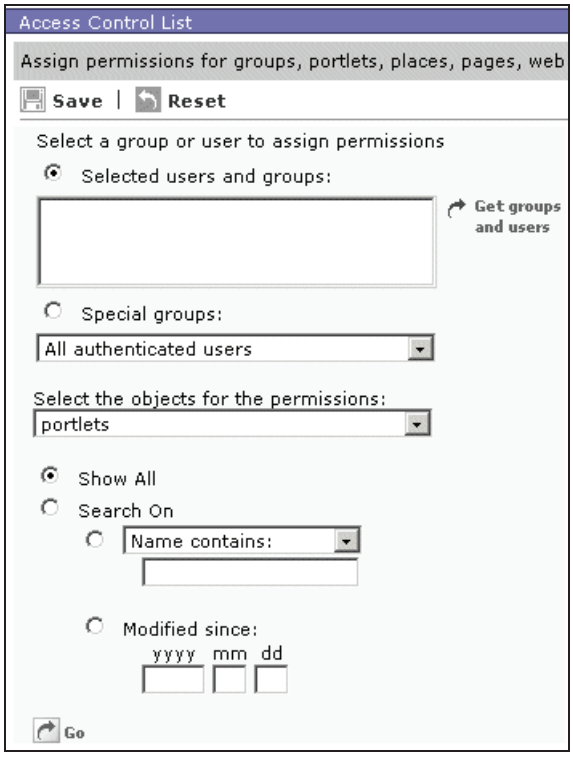


Figure 2-41 Find portlets for SalesGroup

The portlets are returned in the results list. Click the **View** radio button for the portlets associated with Page1, which are:

- My List
- RSS portlet
- World Clock
- QuickLinks
- Reminder

Figure 2-42 on page 54 shows a subset of the selected portlets (QuickLinks, Reminder, RSS portlet). Click **Save** to save these changes. As stated earlier, the view entry in the Active column will be checked for these particular portlets.

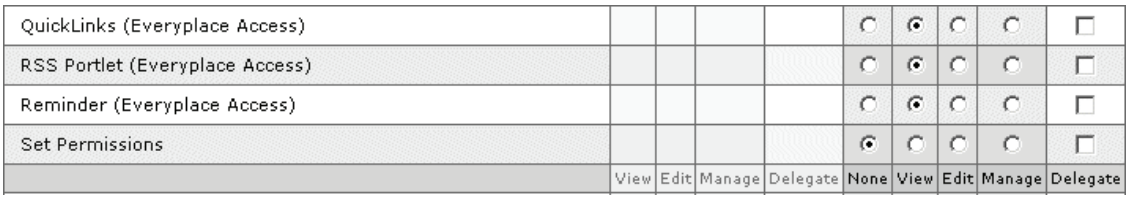


Figure 2-42 Select permission for portlets

Now log off the Portal as `wpsadmin` and log in as `lindamay` (password `lindamay`). Click **MyPlace -> Page1**. All the selected portlets should be available, as shown in Figure 2-43 on page 55.

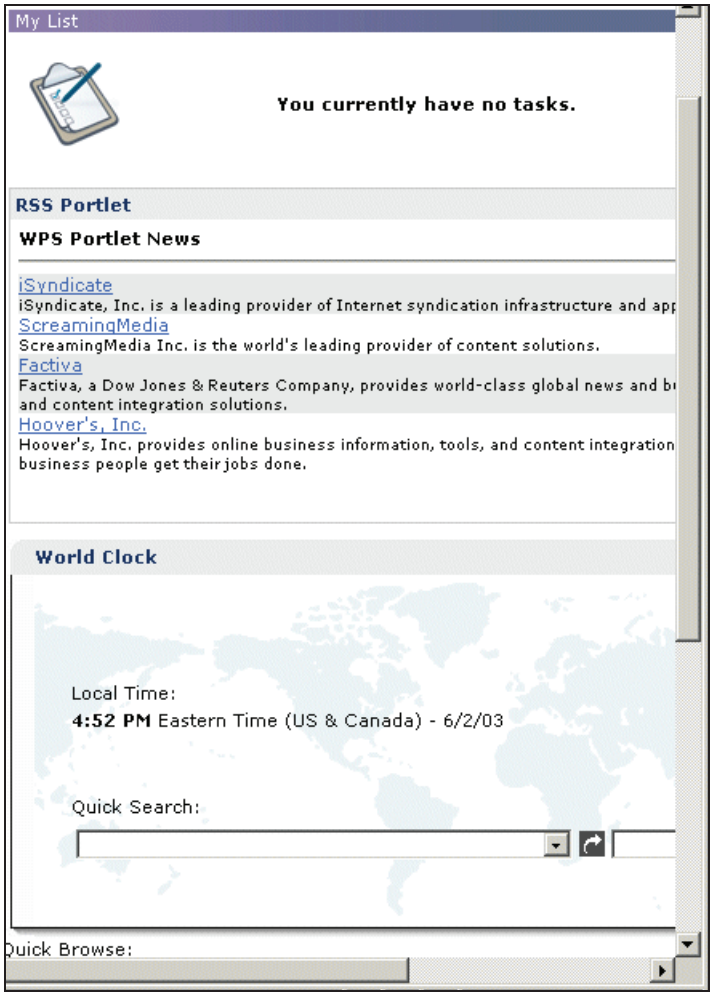


Figure 2-43 Permission results

Remember Huan was not part of the SalesGroup. Let's sign on as Huan (both user ID and password of `huan01`). Notice that he does not have access to MyPlace at all, as shown in Figure 2-44.



Figure 2-44 Sign on Huan

Access Control List details

The Access Control list displays the Active and Minimum access rights. Each user or user group can be assigned a different access control level for each resource. An access level may be inherited through membership in a user group. Active permissions can be inherited from the user groups to which a user or a nested user group belongs. The Minimum section contains the possible values that can be assigned which are:

- None: No rights to the resource.
- View: Rights to view the resource. This option or better is required to view any results.
- Edit: Rights to edit the resource.
- Manage: Allows the user or user group to manage the resource.
- Delegate: Allows the user or user group to give authority to other users and user groups access to an associated resource. Users may not assign an access level that is higher than their own for any resource.

2.6 Changing themes and skins

You can alter the look and feel of your Portal by selecting a different theme for a place and different skins for the portlets on a page. You can create your own custom themes and skins to further personalize the Portal (not covered in this chapter).

Themes represent the overall look and feel, including colors, images and fonts. There are several default themes provided with the standard installation. Each place in the Portal may have a different theme associated with it, thereby creating the appearance of virtual Portals.

The term *skin* refers to the visual appearance of the area surrounding an individual portlet. Each portlet can have its own skin. The skins that are available for use with a portlet are defined by the Portal theme that is associated with the place.

MyPlace was originally defined with the Corporate theme. Let's go back and change the theme to Finance by doing the following steps:

1. Select the **Work with Pages** tab and select **Myplace**.
2. Change the theme from Corporate to Finance.
3. Click **Save**.
4. Select the **MyPlace** tab and the new theme for MyPlace will appear. MyPlace with the Finance theme is shown in Figure 2-45 on page 57.

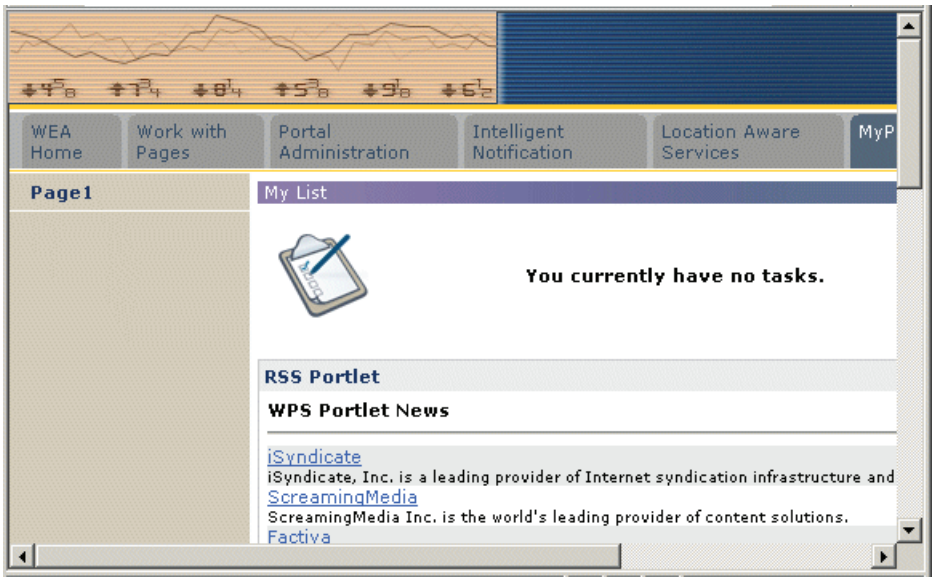


Figure 2-45 MyPlace Finance theme

The theme changes on the PDA as well. Figure 2-46 shows the Corporate theme while Figure 2-47 shows the Finance theme.

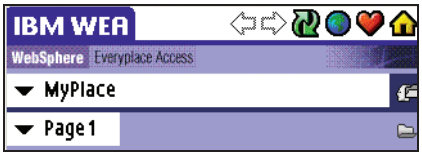


Figure 2-46 Palm Corporate theme

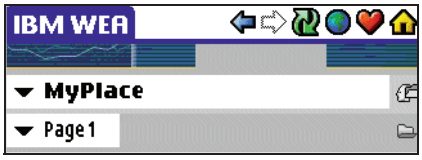


Figure 2-47 Palm Finance theme

Each portlet can have its own skin as well as the place having a unique theme. In order to change the skin for a portlet on a page, select **Choose Skins** (under Manage Places and Pages). In the left pane, expand **MyPlace** and select **Page1**.

Across from each portlet is a drop-down list that contains the skins currently available. You can select a skin from the list. If you want to preview the skin, click the eye icon and the skin will be displayed. As you can see, we have selected different skins for different portlets, as shown in Figure 2-48. They are in order:

- My List - Theme default
- RSS portlet - Outline
- World Clock - Shadow
- QuickLinks - NoSkin
- Reminder - No Border

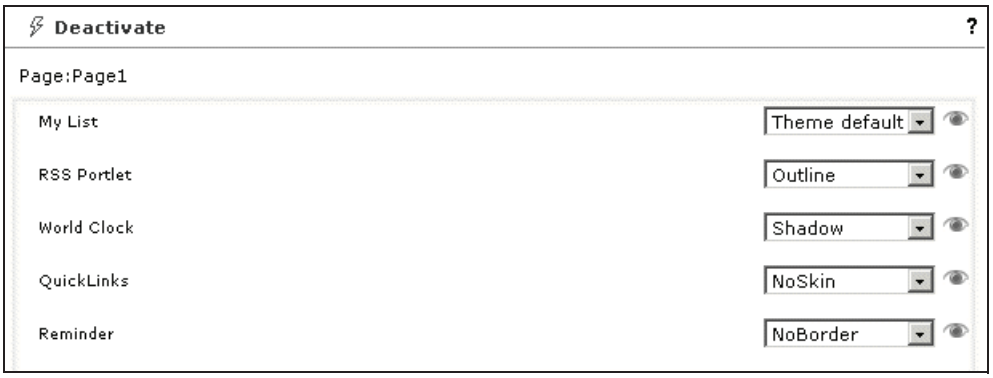


Figure 2-48 Choose skin

Once the skins have been selected, click **Activate** to activate the page. The results displayed on the desktop are shown in Figure 2-49 on page 59.

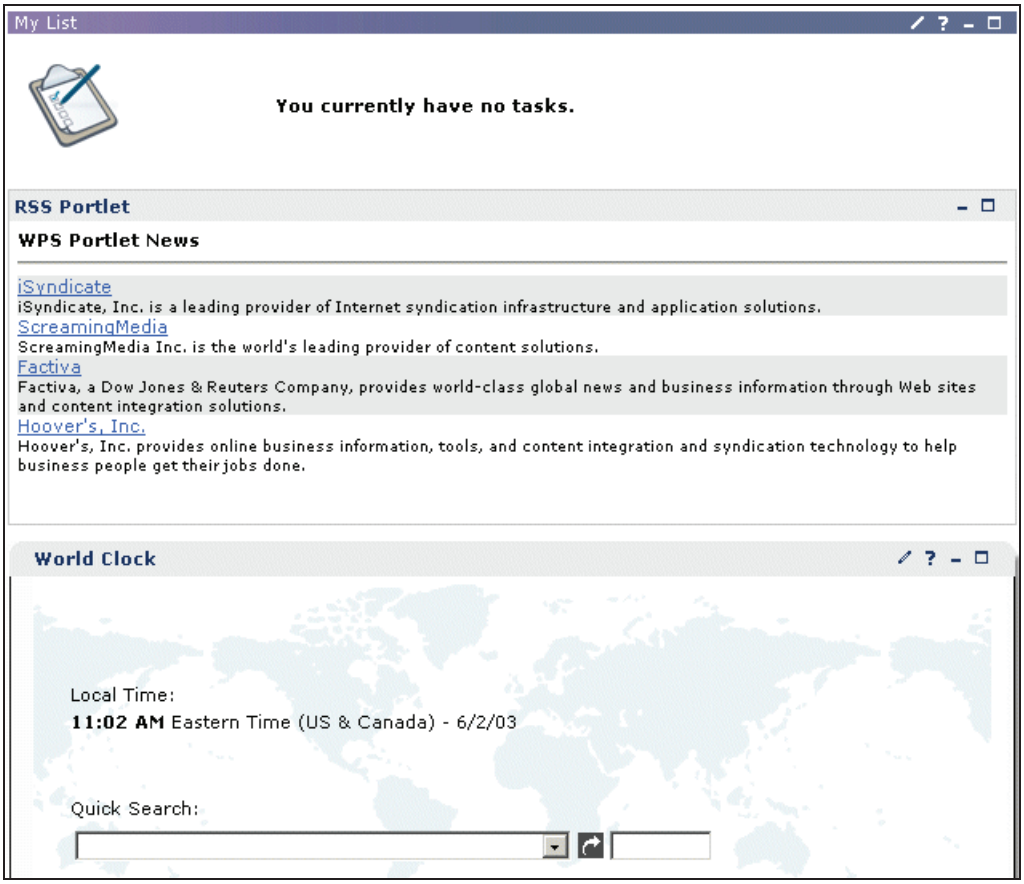


Figure 2-49 Skins viewed in desktop browser

Each portlet with their skin is reflected on the mobile device. The portlet and their skin is shown as follows:

- My List - not displayed
- RSS portlet - Figure 2-50 on page 60
- World Clock - Figure 2-51 on page 60
- QuickLinks - Figure 2-52 on page 61
- Reminder - Figure 2-53 on page 61



Figure 2-50 Palm with RSS portlet



Figure 2-51 Palm with World Clock portlet



Figure 2-52 Palm with QuickLinks portlet



Figure 2-53 Palm with Reminder portlet

When you look at the initial Palm panel for MyPlace Page1, as shown in Figure 2-54 on page 62, you do not see the My List portlet, even though it appears on the desktop as shown in Figure 2-45 on page 57. You may be wondering why this portlet does not appear even though both the place (MyPlace) and the page (Page1) are defined to present markup for HTML, cHTML, WML, and PDA. Not only do the place and page need to support the target devices markup, but the portlet must also support the device markup and in this case the MyList portlet only supports HTML, so it will not be included on the PDA.



Figure 2-54 Palm Page1 icons

3

Enhanced portlets

Everyplace Access extends the functionality of the Portal offering by providing specialized portlets for mobile devices. In this chapter we introduce you to the enhanced portlets provided in Everyplace Access.

© Copyright IBM Corp. 2003. All rights reserved.

3.1 Introduction

WebSphere Everyplace Access includes many enhancements that extend limited base Portal functions to mobile devices. WebSphere Everyplace Access provides device-viewing enhancements for online viewing, offline Portal content viewing, selected portlet enhancements, and a custom WebSphere Everyplace Access page group.

Portlets are visual windows that organize content from different sources, such as Web sites, e-mail, and business applications, into a single interface for individual or group access. Portlets enable highly customized and productive work environments for projects, tasks, or organizational communities.

Everyplace Access provides a set of productivity and personal information manager (PIM) portlets that have been enhanced for viewing on mobile devices using either WML or HTML-based browsers. Everyplace Access selected these particular portlets to focus on personal information management and enterprise-related functionality. These selected portlets have been enhanced specifically for Palm browsers, such as Eudora and AU Mobile Internet browser, and for Pocket PC browsers, such as Pocket IE. Other mobile device browsers are supported.

Everyplace Access provides synchronization functionality via Synchronization Server to further enhance the PIM portlets. Synchronization function is available for Microsoft Exchange and Lotus Notes using the Synchronization Settings portlet. These synchronization settings define what is synchronized to the native PIM application on the PDA. For example, you may only want to read your urgent mail on your PDA. Synchronization settings let you decide what information will be available on your device.

3.1.1 Supported devices

Everyplace Access includes a set of portlets that have been enhanced for viewing on the following supported mobile devices.

Pocket PC devices

Enhanced portlets look and work well on Pocket PC 2002 devices that use the Microsoft Pocket Internet Explorer (IE) browser. This means that the portlets conform to the specifications for Pocket IE, which are HTML 3.2, limited JavaScript (as defined by the Windows CE JS3.0 specification), and no Cascading Style Sheet support. Although Cascading Style Sheets are not supported by the device, the Everyplace Access will convert the stylesheet attributes on the server.

These enhanced portlets have been tested on the following Pocket PC devices:

- Compaq iPAQ 3870 3850 (Pocket PC 2002)
- HP Jornada 564
- NEC Mobile Pro
- Maestro PDA

Palm OS devices

Enhanced portlets also look and work well on Palm OS supported devices that use Eudora 2.1, AU Mobile Internet browser and Palm Web Pro Browser Version 1.0. Enhanced portlets conform to Palm OS specifications.

These enhanced portlets have been tested on the following Palm OS devices:

- Palm m130
- IBM WorkPad C505
- Palm Tungsten T

WML browser on mobile phones

Portlets that lend themselves to limited browser capabilities have been modified to make them available on WML browsers. These portlets look and work well from a WML browser. In some cases, portlet functionality is more limited when accessed from a WML browser. This is due to the limited user interface that is typically available on most mobile phones and other WML-browser devices. For example when viewing e-mail, a set of predefined replies is provided so the user does not need to type a complete message when replying.

WML browsers have been tested on the Ericsson T68.

3.2 Enhanced portlets provided by Everyplace Access

Table 3-1 shows the list of enhanced PIM portlets provided in the Everyplace Access package. Portlets may have additional functions beyond View, including Configure, Edit, Create, Delete, Forward, Reply, Reply All, etc. The table shows the functions available by portlet for the specific markup: PDA, PDA text-only, and WML. This function is provided for online browsing only.

Table 3-1 PIM portlets

Portlets	PDA markup[1]	PDA text-only markup[2]	WML markup[3]
Lotus Notes Mail	All functions available	All functions	View, Forward, Reply, ReplyAll

Portlets	PDA markup[1]	PDA text-only markup[2]	WML markup[3]
Lotus Notes Calendar	All functions available	View, Create, Delete	View only
Lotus Notes ToDo	All functions available	View, Create, Delete	View only
Lotus Notes Contacts	All functions available	View, Create, Delete	View only
Lotus Notes Journal	All functions available	View, Create, Delete	View only
Microsoft Exchange Mail	All functions available	All functions available	View, Forward, Reply, ReplyAll
Microsoft Exchange Calendar	All functions available	View, Create, Delete	View only
Microsoft Exchange Tasks	All functions available	View, Create, Delete	View only
Microsoft Exchange Contacts	All functions available	View, Create, Delete	View only
Microsoft Exchange Notes	All functions available	View, Create, Delete	View only

Notes:

1. Pocket PC, Opera for Linux OS and Nokia 9210i, 9290
2. Eudora and WebPro for Palm OS devices
3. Ericsson T68 AU Mobile Internet Browser

Table 3-2 shows the productivity portlets provided by Everyplace.

Table 3-2 Productivity portlets

Portlets	PDA markup[1]	PDA text-only markup[2]	WML markup[3]
Reminders	Edit, View	View only	View only
World Clock	Edit, View	View only	View only
QuickLinks	Edit, View	View only	View only
Internet Mailbox	All functions available	View, Create, Delete, Reply	View, Create, Delete, Reply

Portlets	PDA markup[1]	PDA text-only markup[2]	WML markup[3]
Banner Ad	Edit, View	View only	View only
Image Viewer	Edit, View	View only	View only

Notes:

1. Pocket PC, Opera for Linux OS and Nokia 9210i, 9290
2. Eudora and WebPro for Palm OS devices
3. Ericsson T68 AU Mobile Internet Browser

3.2.1 Lotus Notes PIM portlets

These portlets provide access to the PIM functions of Lotus Notes. You have to configure the Lotus Domino Server in order to access it from the PIM portlets. For information about configuring the Lotus Domino Server refer to 22.3.1, “Lotus Domino Server configuration” on page 714.

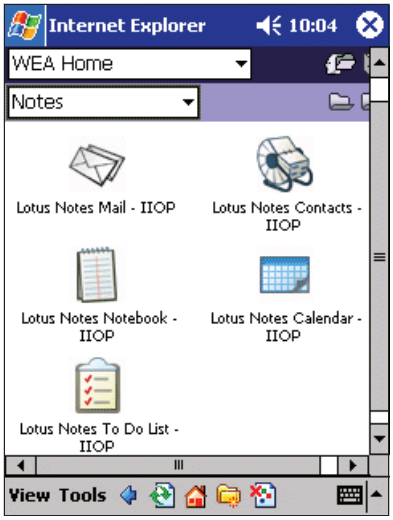


Figure 3-1 Lotus Notes PIM portlets

Lotus Notes Mail portlet

Lotus Notes Mail - IIOP provides access to your Lotus Notes mail database using a Web browser or Web-enabled wireless phone. Before you can access your mail database, you must first edit the settings and enter the name of your mail server, user name, and password. If you intend to access your mail using a wireless phone, you should also set the WAP Phone Options.

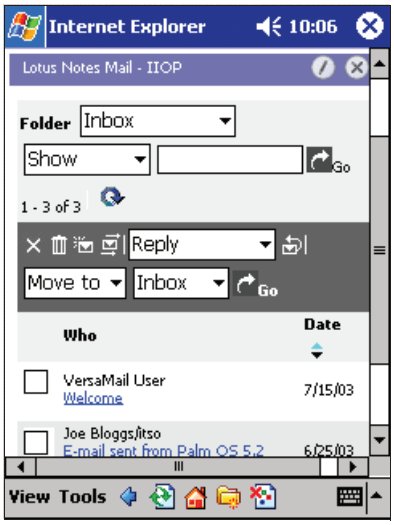


Figure 3-2 Lotus Notes Mail portlet

Lotus Notes Contacts portlet

Lotus Notes Contacts - IIOP provides access to your Lotus Notes contacts using a Web browser. Before you can access your contacts, you must first edit the settings and enter the name of your mail server, user name, and password.



Figure 3-3 Lotus Notes Contacts portlet

Lotus Notes Journal (Notebook) portlet

Lotus Notes Notebook - IIOP provides access to your Lotus Notes notebook using a Web browser. Before you can access your notebook, you must first edit the settings and enter the name of your mail server, user name, and password.



Figure 3-4 Lotus Notes Notebook portlet

Lotus Notes Calendar portlet

Lotus Notes Calendar - IIOP provides access to your Lotus Notes calendar using a Web browser. Before you can access your calendar, you must first edit the settings and enter the name of your mail server, user name, and password.

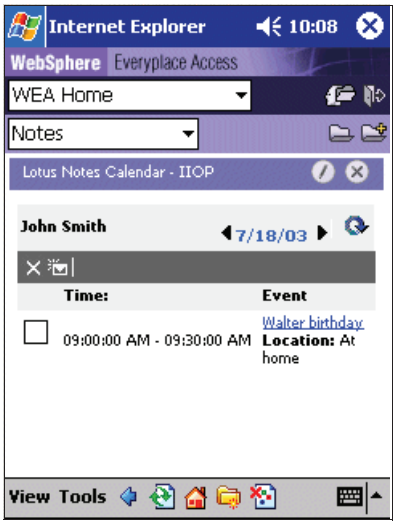


Figure 3-5 Lotus Notes Calendar portlet

Lotus Notes ToDo portlet

Lotus Notes ToDo - IIOP provides access to your Lotus Notes to-do list using a Web browser. Before you can access your to-do list, you must first edit the settings and enter the name of your mail server, user name, and password.

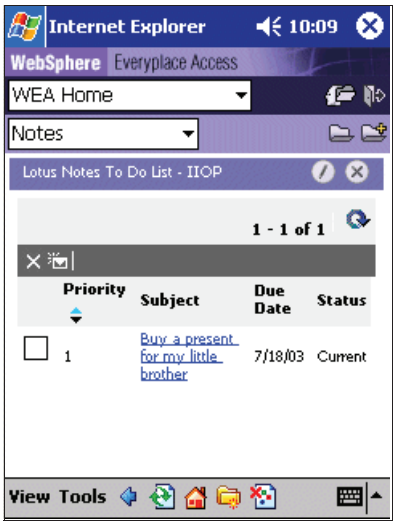


Figure 3-6 Lotus Notes ToDo portlet

3.2.2 Microsoft Exchange PIM portlets

The Microsoft Exchange PIM portlets provides access to Exchange 5.5/2000 servers. Before using these portlets, you have to configure them by setting a credential slot to store the users' passwords. The credential slot can be created using the Credential Vault Administration page in the WebSphere Everyplace Access server.

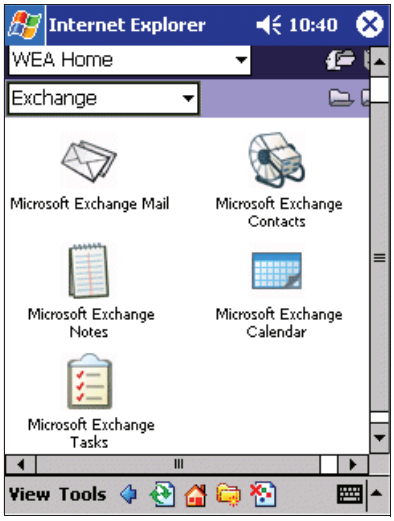


Figure 3-7 Microsoft Exchange PIM portlets

Microsoft Exchange Mail portlet

Exchange mail connects to an Exchange Server. You can perform standard mail functions, such as compose, reply, forward, and delete mail.



Figure 3-8 Microsoft Exchange Mail portlet

Microsoft Exchange Calendar portlet

Exchange Calendar connects to an Exchange Server. You can perform standard calendar functions, such as creating, editing, deleting, and viewing appointments.

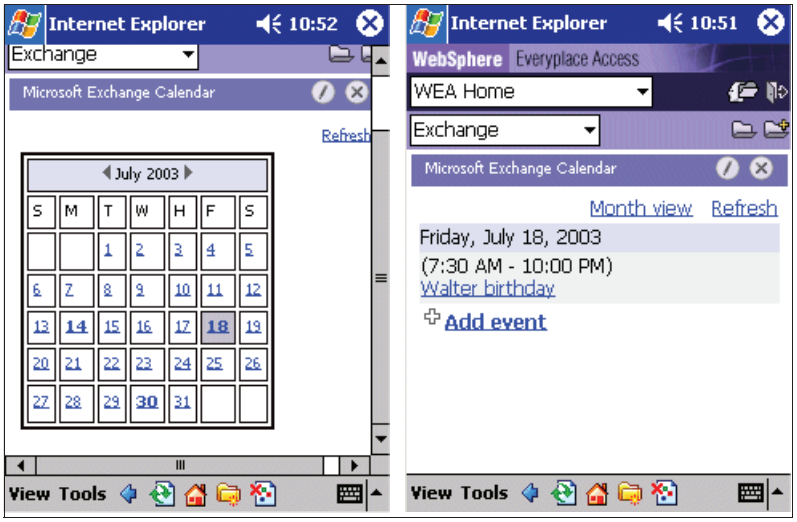


Figure 3-9 Microsoft Exchange Calendar portlet

Microsoft Exchange Tasks portlet

Exchange Tasks connects to an Exchange server. You can perform standard task functions, such as creating, editing, deleting, and viewing tasks.

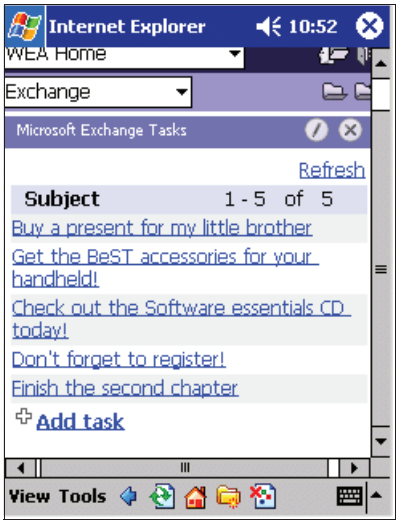


Figure 3-10 Microsoft Exchange Tasks portlet

Microsoft Exchange Contacts portlet

Exchange Contacts connects to an Exchange server. You can perform standard contact list functions, such as creating, editing, deleting, and viewing contacts.



Figure 3-11 Microsoft Exchange Contacts portlet

Microsoft Exchange Notes portlet

Exchange Notes connects to an Exchange server. You can perform standard note functions, such as creating, editing, deleting, and viewing notes.



Figure 3-12 Microsoft Exchange Notes portlet

3.2.3 Productivity portlets

WebSphere Everyplace Access provides several productivity portlets enhanced for mobile access. There are other enhanced portlets in the IBM WebSphere Portal Catalog at
`http://www7b.software.ibm.com/wsdd/zones/portal/catalog/`

Reminder portlet

Reminder allows you to view your reminders either separately or as a list. You can add, edit, or delete reminders. You can mark your reminders either “urgent” or “normal.”

Note: There is a 240-character limit for each reminder.

With the Reminder portlet, you can:

- Add a new reminder
- Edit an existing reminder
- Mark a reminder as “urgent” or “normal”
- Delete an existing reminder



Figure 3-13 Reminder portlet

World Clock portlet

World Clock allows you to set your local time zone and view the current time in that time zone. It also allows you to view the current local time in several time zones around the world at one time. You can also do a quick search for the current local time in an individual time zone.

With World Clock you can:

- Set a local time zone and view the current time
- Set a list of time zones around the world and view their current times
- Do a quick search for the current time in an individual time zone

Note: Be careful that you select the correct time zone. Even though a time zone has the same GMT, the actual local time may differ so you need to select the time zone with the correct cities. For example, GMT 7:00 has two options, Arizona and Mountain Time, because of Daylight Savings Time.

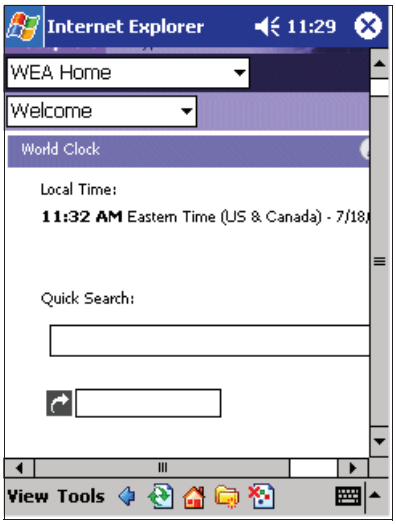


Figure 3-14 World Clock portlet

QuickLinks portlet

The QuickLinks window displays QuickLinks that you have defined in a browser. You can then use these QuickLinks to access regularly visited Web sites. You can add, delete, and edit your QuickLinks as well as import existing QuickLinks from your workstation.

Note: The maximum import size for QuickLinks is 64 KB.

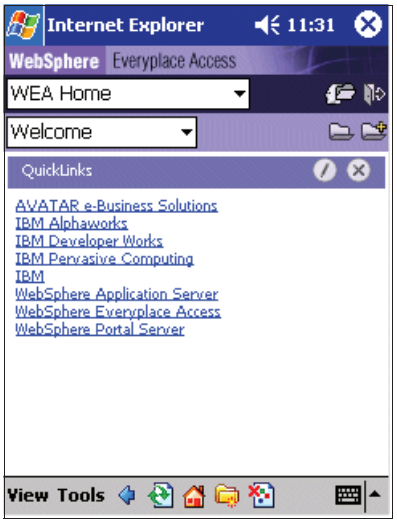


Figure 3-15 QuickLinks portlet

Internet Mailbox portlet

The IBM Internet Mail Box portlet allows you to receive, compose and send Internet e-mail using either a Web browser or wireless phone. It supports POP3 and IMAP protocols.

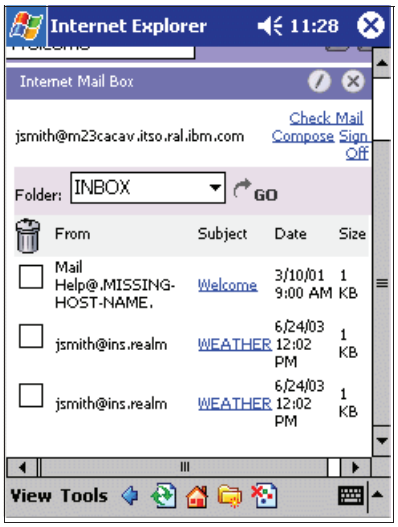


Figure 3-16 Internet Mailbox portlet

Banner Ad portlet

Banner Ad allows you to view advertisements by displaying a selectable image within the Banner Ad window. If the image is selected, then the browser is taken to the Web site for the advertisement. Both the image and the destination Web site are configurable.



Figure 3-17 Banner Ad portlet

Image Viewer portlet

The Image Viewer allows you to view images within the Image Viewer window.



Figure 3-18 Image Viewer portlet

4

Everyplace Client

In this chapter, we describe the installation and configuration of the Everyplace Client in different devices.

© Copyright IBM Corp. 2003. All rights reserved.

4.1 Overview

The Everyplace Client is the client application for personal digital assistants (PDAs). Everyplace Client provides a framework that supports information refreshment, security, device management, offline Portal pages, offline forms, and database synchronization.

For more information about the Everyplace Client and the supported devices refer to 1.2.4, “Everyplace Client” on page 14.

4.2 Everyplace Client for Palm OS

The Everyplace Client for Palm OS provides different characteristics depending on the Palm OS version. For a list of the characteristics supported on each version, refer to 1.2.4, “Everyplace Client” on page 14.

4.2.1 Everyplace Client installation

To install Everyplace Client for Palm OS, follow these steps:

1. Install the Palm desktop software to successfully install the Everyplace Client. Refer to Appendix E, “Palm Desktop installation” on page 1193 for more information.
2. Ensure your Palm device is placed on the cradle and connected to the desktop via the USB cable.
3. Insert the **IBM Everyplace Client CD** and run the setup.exe program under \client\PalmXX, where XX is the Palm OS version.
4. Select the setup language for this installation. For the purposes of this installation, select **English**. Click **OK**.
5. You will be presented now with the welcome window for the installation. Click **Next**.

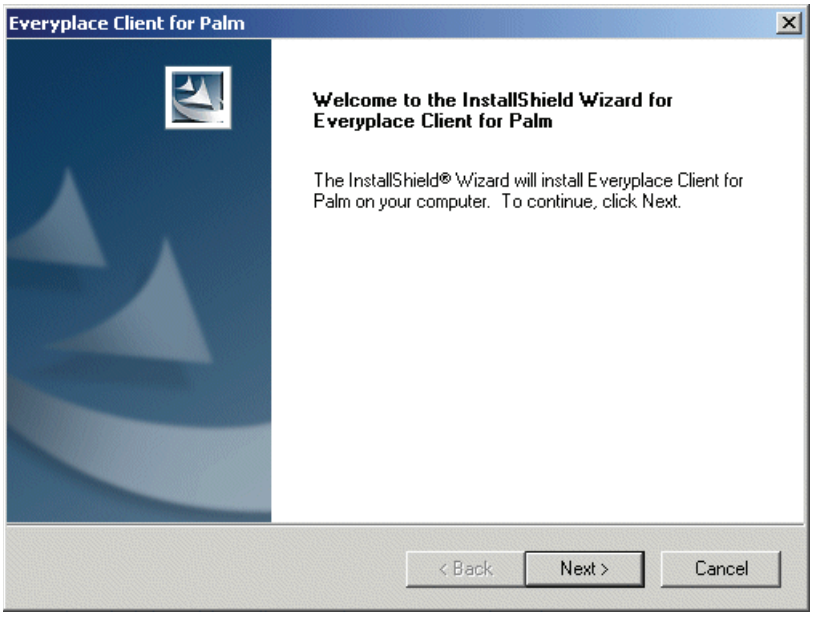


Figure 4-1 InstallShield welcome window

6. Read through the License Agreement and then click **Yes**. If you are also presented with the License Agreement in Notepad, you may want to close it.
7. Accept the default destination folder by clicking **Next**.

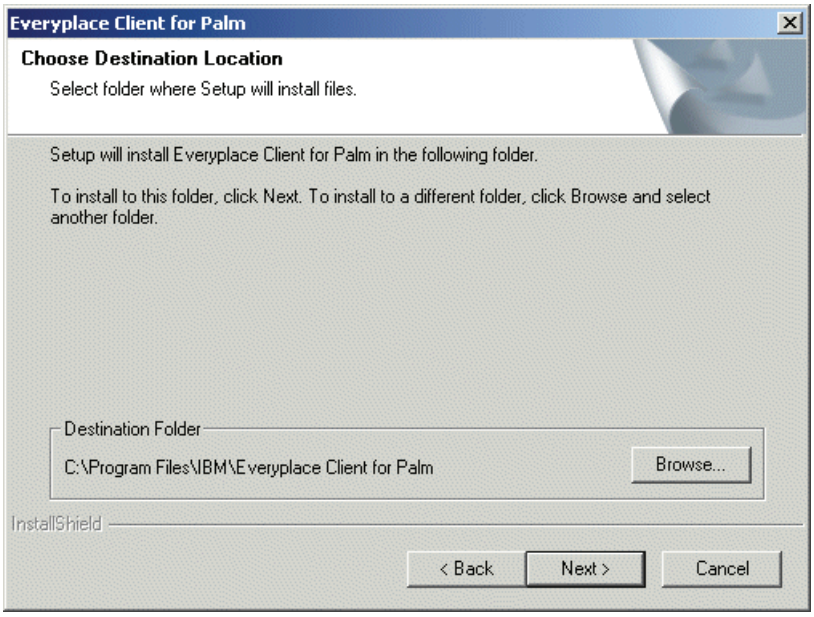


Figure 4-2 Select destination folder for Everyplace Client

8. On the Select Components window, select the components you wish to install and use. For the purposes of this installation, we will just accept the default selection and click **Next**.

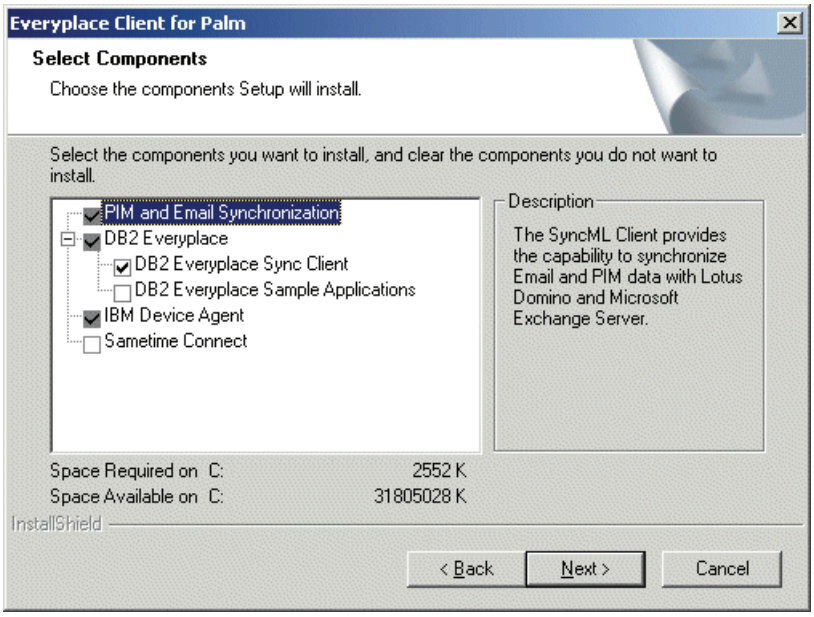


Figure 4-3 Select Everyplace Client components to install

If you are installing the Everyplace Client for PalmOS 5.2, you have a few more components to install, as shown in Figure 4-4 on page 86. Click **Next**.

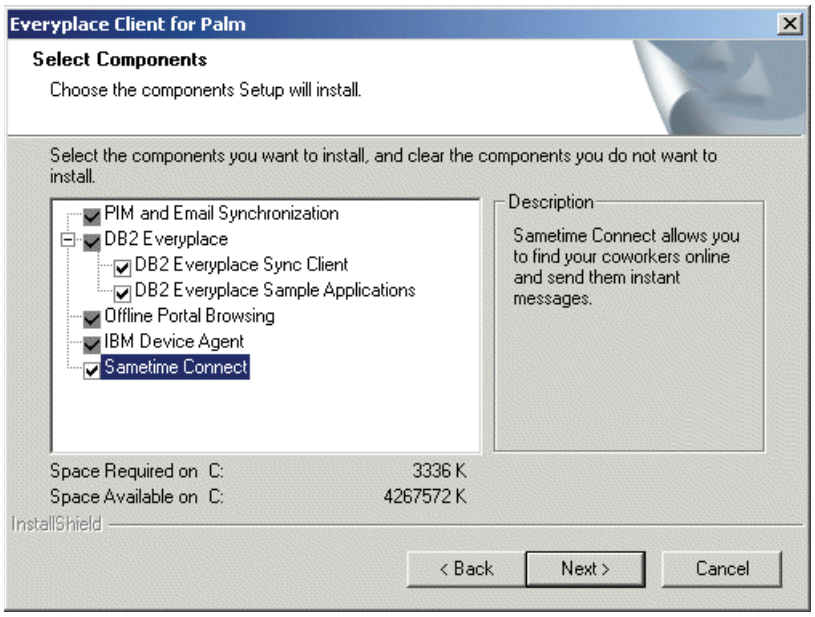


Figure 4-4 Everyplace Client for PalmOS 5.2

9. Review the settings you have selected. If you want to change anything, click **Back**. Otherwise, click **Next**.

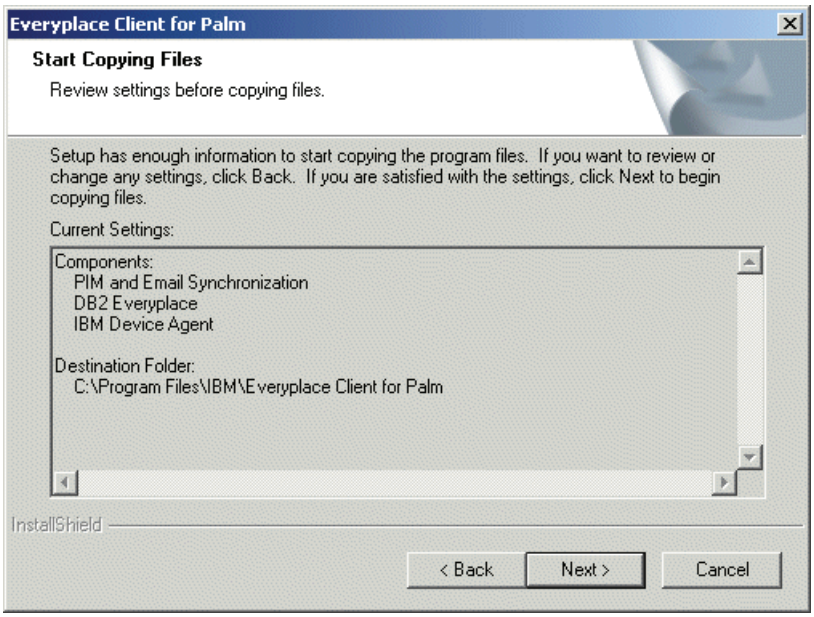


Figure 4-5 Installation selection summary

10. The **InstallShield Wizard Complete** window will now appear. Click **Finish**. This completes our installation as far as the desktop is concerned.

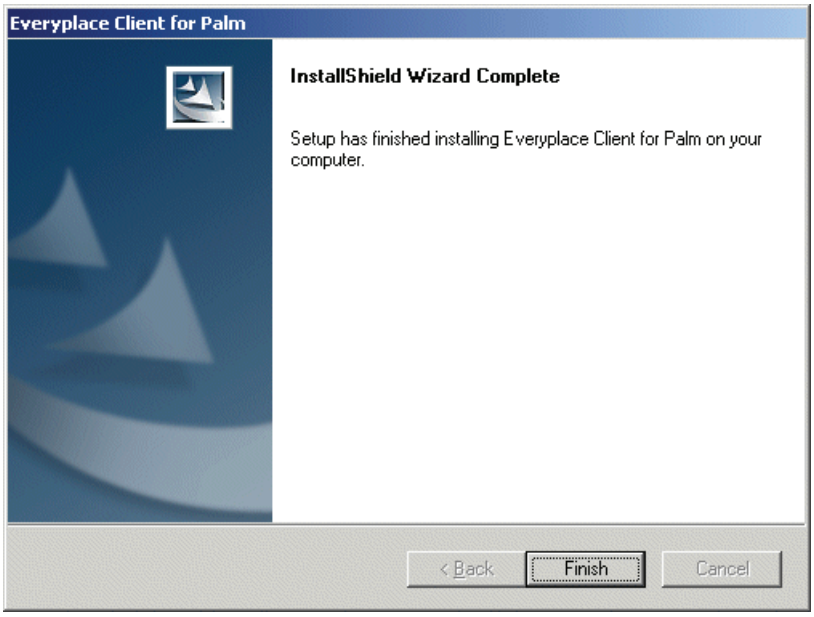


Figure 4-6 InstallShield Wizard Complete window

11. You will get a pop-up window telling you to use HotSync to install Everyplace Client on your Palm device. Click **OK**.

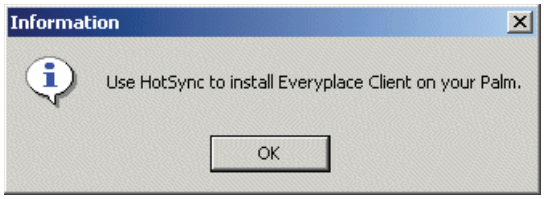


Figure 4-7 Use HotSync

Now, if you want to install the client on your device, simply perform a HotSync synchronization. But if you wish to install the client on the Palm Emulator, or on the Palm Simulator for development purposes, you have an alternative method, which is described in the following sections.

Installing on Palm Emulator (PalmOS 3.x and 4.x support)

Before you start the Everyplace Client installation, you may want to read Appendix D, “Palm Emulator installation” on page 1185 for instructions on installing the Palm Emulator.

Follow these steps to install the Everyplace Client on the Palm Emulator:

1. Start an Emulator session.
2. Right-click the Emulator and select **Install Application/Database -> Other**. Navigate to the folder C:\Program Files\IBM\Everyplace Client for Palm\palmfiles and select all .prc files. Do not select the CondInst directory. Click **Open**.

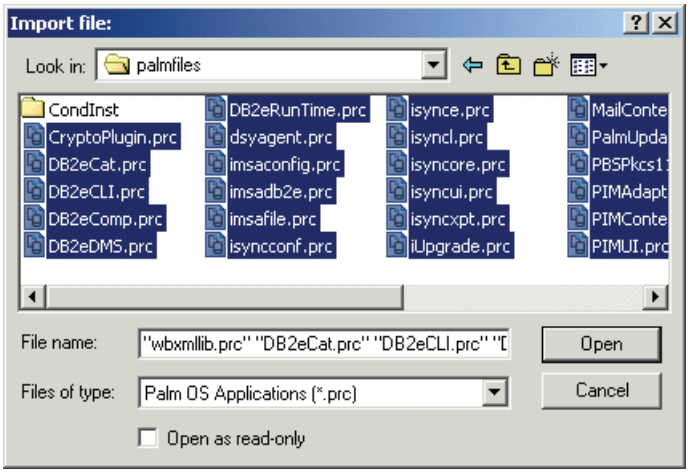


Figure 4-8 Select all .prc files

3. The installation should take a few seconds. After the installation, verify that the files have been installed properly by going to the All folder on the Emulator session. You should see the icons corresponding to the components you have selected during the install process.



Figure 4-9 Palm OS Emulator session with Everyplace Client installed

4. The Emulator session is now installed with Everyplace Client. To save the session, right-click the Emulator window and select **Save**.

Installing on Palm Simulator (PalmOS 5.x support)

Before you start the Everyplace Client installation, you may want to read Appendix C, “Palm Simulator installation” on page 1177 for instructions on installing and configuring the Palm Simulator.

The following steps illustrates a sample Everyplace Client installation:

1. Create a directory called AppsDLL in the Simulator program directory. Copy to this directory the PTLib.dll file, located in the Everyplace Client CD under the \client\Palm52\prcs\en\wox\Simulator directory.
2. Start a Simulator session.
3. Right-click in the Simulator window and choose **Install -> Database**. On the Open window, move to the Everyplace installation directory (c:\Program Files\IBM\Everyplace Client for Palm\palmfiles in the example) and select all files except the PTLib.prc file. There is a special version for Simulator of this file. Click **Open**.

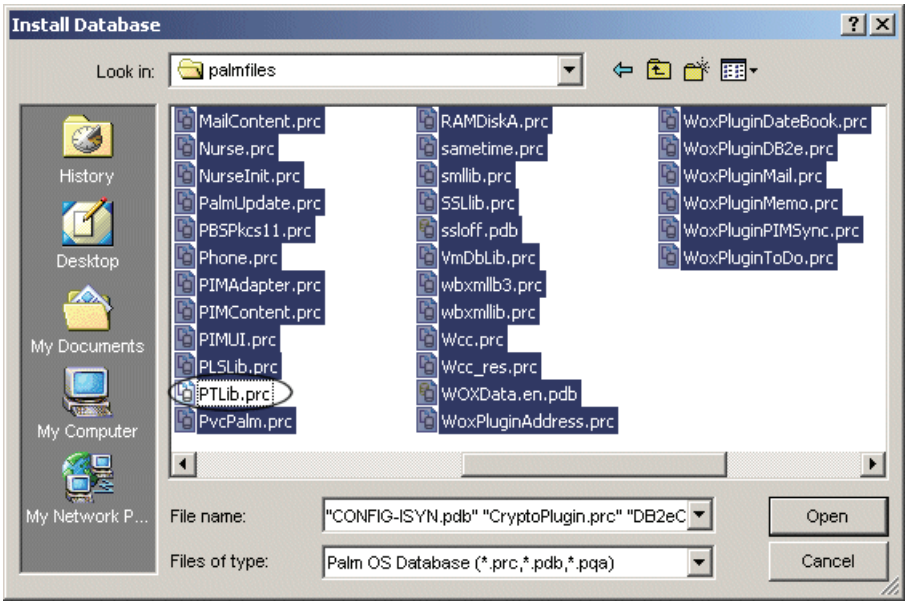


Figure 4-10 Select all the Everyplace Client files

4. The installer warns that the Simulator requires a soft reset. Click **OK**.

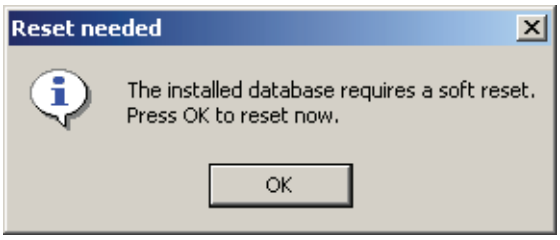


Figure 4-11 Soft reset required

5. When the reset is completed, if the Simulator warns about an unrecognized format on an external device, click **OK**.

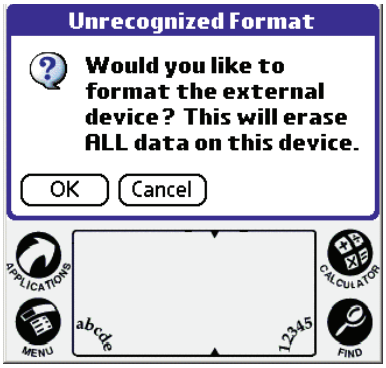


Figure 4-12 Unrecognized format

6. Install the PTLib.prc file from the Everyplace Client CD, located in \client\Palm52\prcs\en\wox\Simulator directory.
7. The Simulator is now properly configured and ready to use. Now you should save the current session by selecting **Storage -> Save** from the Simulator menu. When you want to restore the session, simply select **Storage -> Load** from the Simulator menu.

4.2.2 Everyplace Client configuration for Palm OS 5.2

For Palm OS Version 5.2, an integrated shell is provided. You can use this interface to synchronize, in a centralized manner, PIM and database information. For the other clients (Device Agent and offline browsing), you still have to perform the configuration in each client respectively. Refer to the appropriate chapters for these other clients.

Important: To ensure you have the latest code, you should visit the WebSphere Everyplace Access support Web site at `http://www.ibm.com/software/pervasive/ws_everyplace_access/support/` to download the latest fix packs and code fixes. Apply these fixes prior to using the samples within this redbook.

Before using the client, you have to perform some initial configuration. Essentially, there are two configuration tasks to perform:

- Setting up the user name and password for synchronization
- Create a network profile to connect to a WebSphere Everyplace Access server

Follow these steps to perform a basic configuration of the Everyplace Client for Palm OS 5.2:

1. Select the **IBM WEA Client** icon, as shown in Figure 4-13 on page 93.

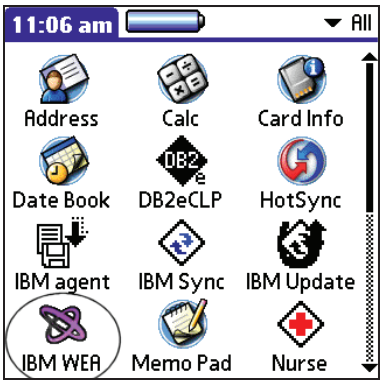


Figure 4-13 Starting the IBM WebSphere Everyplace Access Client

2. Enter the user information for synchronization, as shown in Figure 4-14. Select **Log in**.

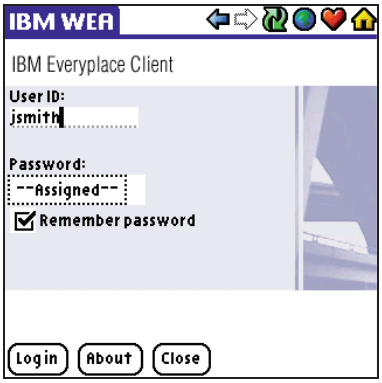


Figure 4-14 Entering user information

3. In the Everyplace Client main window, select the wrench icon to enter the Settings page, as shown in Figure 4-15 on page 94.



Figure 4-15 Entering the Settings page

4. In the Settings page, select the **Network profiles** icon to configure a network profile, as shown in Figure 4-16.

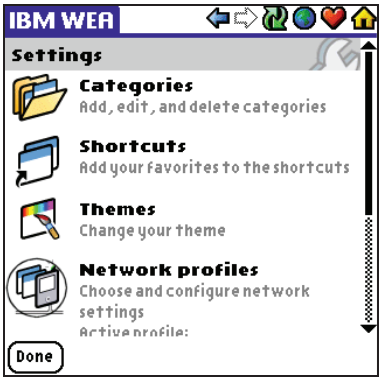


Figure 4-16 Configuring a network profile

5. By default, in Palm devices, you have three network profiles already created: Default, On the Road, and Work. When you synchronize, you can choose between these depending on the connection that you have in a particular location. In this case, we will configure the default profile. Enter the name for the PIM and database servers, as shown in Figure 4-17 on page 95. Also you can enable SSL support. When you finish, select **OK**.

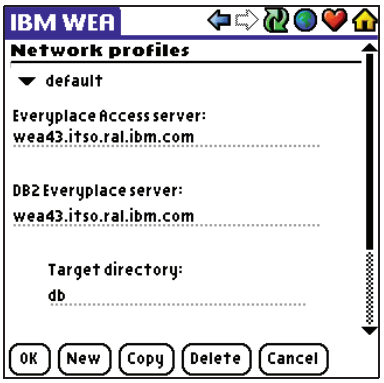


Figure 4-17 Setting the server names

6. Select **Done** to return to the Everyplace Client main page.

4.3 Everyplace Client for Pocket PC 2002

The Everyplace Client for Pocket PC 2002 has a unified browser-based interface for accessing the different characteristics of WebSphere Everyplace Access.

4.3.1 Everyplace Client installation

To install Everyplace Client for Pocket PC, perform the following steps:

1. Ensure your Pocket PC is placed on the cradle and connected to the desktop via the USB cable.
2. Insert the IBM Everyplace Client CD and run the setup.exe program under \client\PPC2002.
3. Select the setup language for this installation. For the purposes of this installation, we will select **English**. Click **OK**.
4. You will now be presented with the welcome window for the installation. Click **Next**.

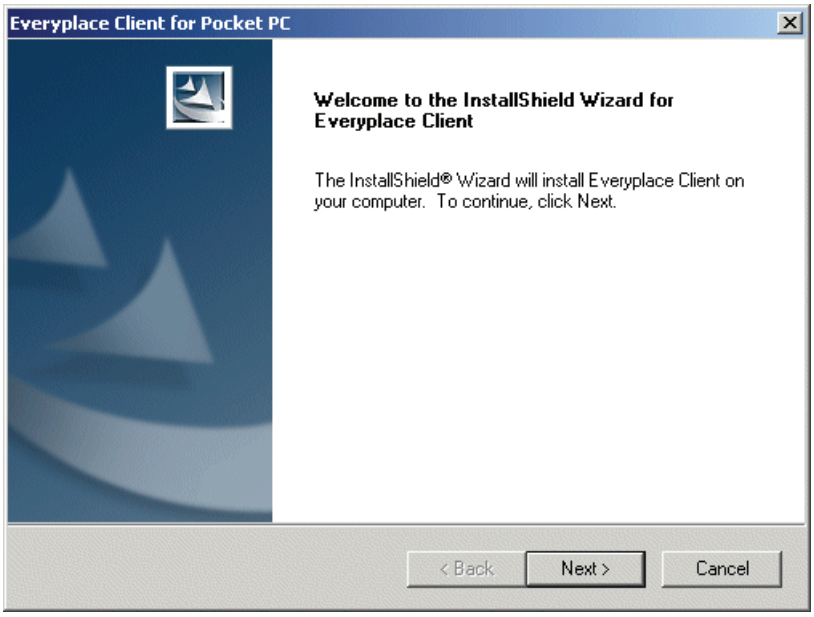


Figure 4-18 InstallShield Welcome window

5. Read through the License Agreement and then click **Yes**. If you are also presented with the License Agreement in Notepad, you may want to close it.
6. Accept the default destination folder by clicking **Next**.

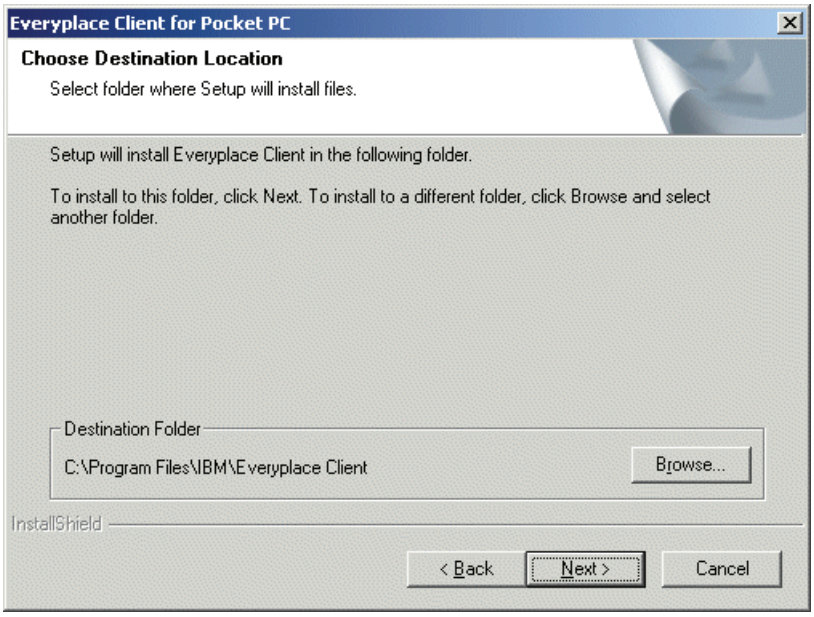


Figure 4-19 Select destination folder for Everyplace Client

7. On the Select Components window, select the components you wish to install and use. For the purposes of this installation, we will just accept the default selection and click **Next**.

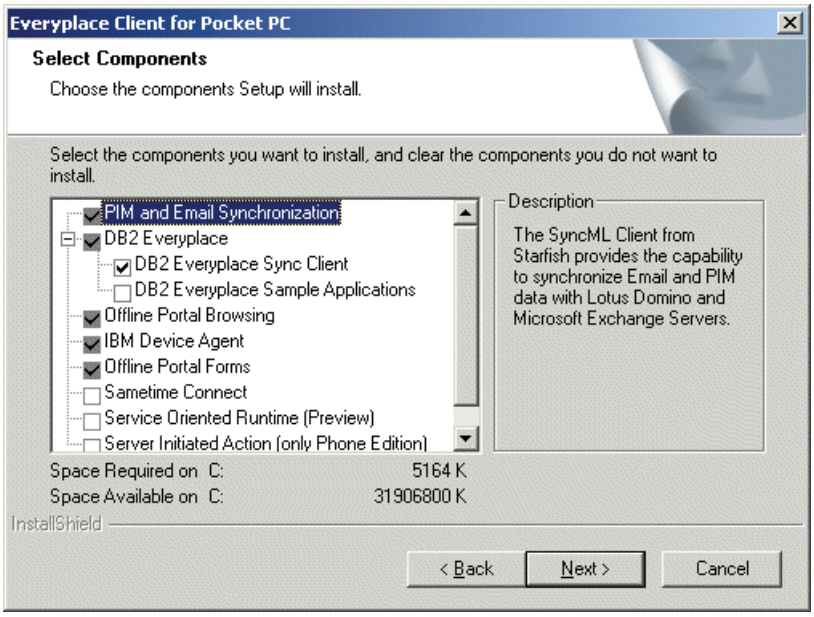


Figure 4-20 Select Everyplace Client components to install

8. Review the settings you have selected. If you want to change anything, click **Back**. Otherwise, click **Next**.

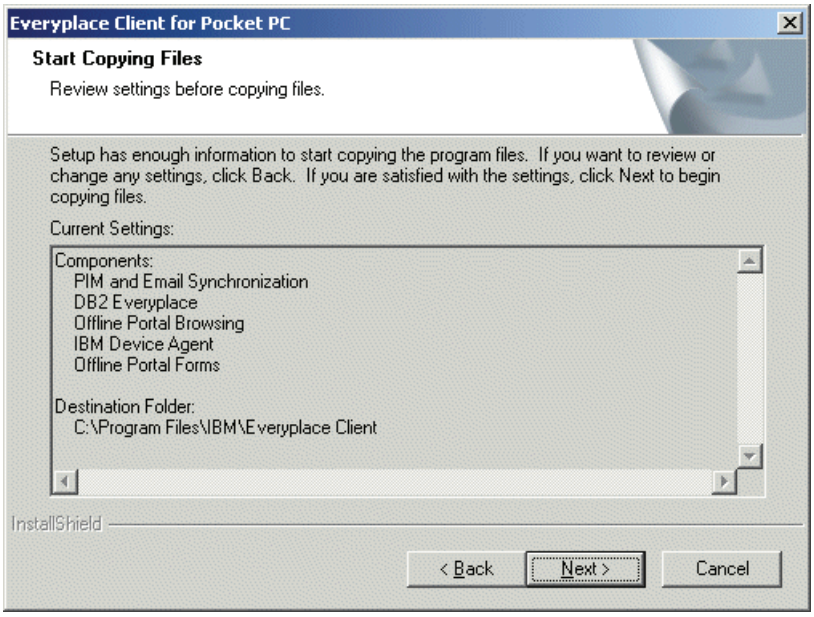


Figure 4-21 Installation selection summary

9. Click **Yes** on the next window.

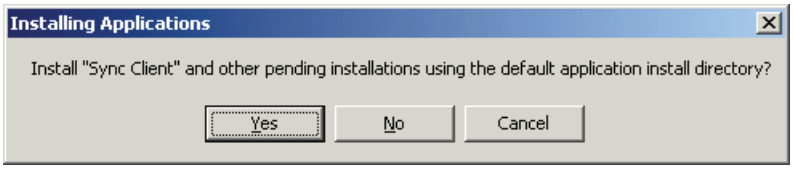


Figure 4-22 Install Sync Client

10. Click **OK**.



Figure 4-23 Check mobile device for additional steps

11. The **InstallShield Wizard Complete** window will now appear. Click **Finish**. This completes our installation as far as the desktop is concerned.

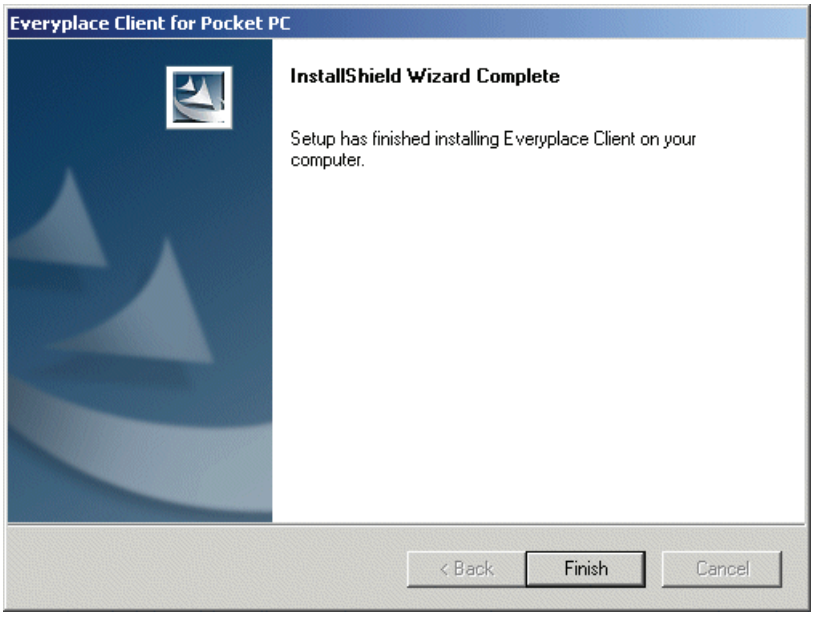


Figure 4-24 Installation complete

12. On your Pocket PC, you will be prompted to enter your user ID and password to log in to your Portal Server. The user ID and password should be for a user who already exists (or will exist) on your Portal Server. Enter the user ID and password and then select the **OK** button in the top-right corner.

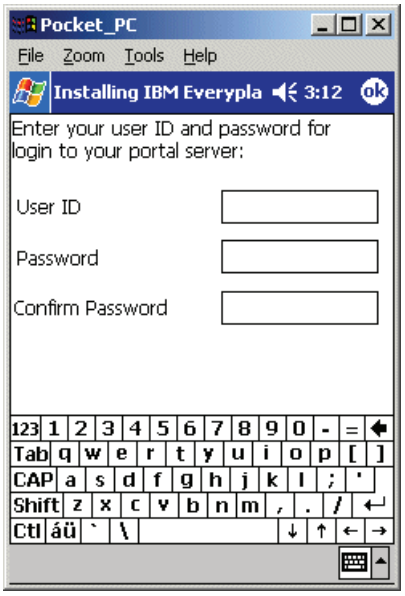


Figure 4-25 Log in user ID and password

13. The Install wizard will copy the appropriate files to the Pocket PC. Depending on the components you have selected, you will next be presented with a series of windows to configure these components. If you've selected the PIM and e-mail synchronization component, you will be prompted to enter your name and e-mail address. Enter your name and e-mail address and select the **OK** button in the top-right corner.
14. This completes the installation of Everyplace Client for Pocket PC.

4.3.2 Everyplace Client configuration

After completing the installation of Everyplace Client for Pocket PC, you still need to configure it to tell it where the component servers are. To configure Everyplace Client for Pocket PC, perform the following steps:

1. If you have not previously configured Everyplace Client:
   a. Launch Everyplace Client on your Pocket PC by clicking **Start -> Everyplace Client**.
   b. You will be prompted for a user name and password. You should enter the user name and password you entered in step 12 on page 100. Click **Log in**.



Figure 4-26 Everyplace Client Login

If you have previously configured Everyplace Client:

a. Launch Everyplace Client on your Pocket PC by selecting **Start -> Everyplace Client**.

b. You may be prompted for a user name and password. You should enter the user name and password you entered in step 12 on page 100. Click **Log in**.



Figure 4-27 Everyplace Client Login

c. The Everyplace Client Home page will appear. Select **Home** to open the drop-down list. Select **My Settings**.

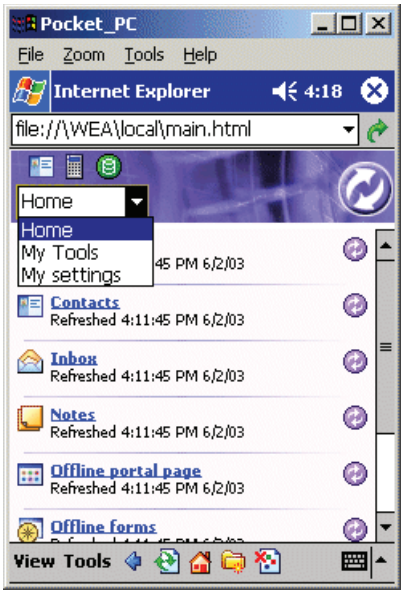


Figure 4-28 Everyplace Client Home

d. On the My settings page, select **Network Profiles**.

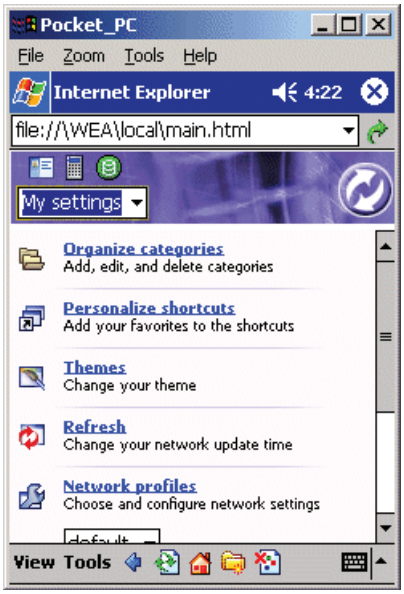


Figure 4-29 Everyplace Client settings

2. You will now be presented with the network profiles page. Depending on which components you have selected, you will need to enter the server name and settings for these components. The server name should be fully qualified. If you've installed all your components on the same box, then the server name should be the same for each component. Note the default network profile selection under the words Active network profile.

   Everyplace Client provides the ability to create multiple network profiles. This will allow you to connect different Everyplace Access servers simply by changing your network profile. For the purposes of this installation, we will stick with the default network profile. Enter the server and settings relevant to your component selection and then select the checkmark in the top-right corner of the Pocket PC browser window.

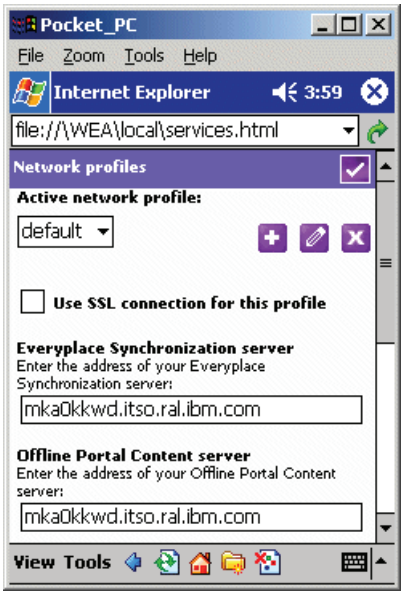


Figure 4-30 Network profile

3. This completes our configuration of Everyplace Client for Pocket PC.

4.4 Everyplace Client for Zaurus

The Everyplace Client for Zaurus devices is made up of two components:

- The Everyplace Client itself
- The Desktop Pass-through Proxy

The Everyplace Client component will be installed in the device to provide the synchronization capabilities. If you wish to connect the Zaurus to the WebSphere Everyplace Access server using the USB connection provided by the Zaurus Manager and the Zaurus USB driver, you will need to install the Desktop Pass-through Proxy. It is a special component developed by IBM only for Zaurus devices. If you are using another connection method, such as a network card or a Wi-Fi card, you won't need to use the Pass-through Proxy.

Important: To ensure you have the latest code, you should visit the WebSphere Everyplace Access support Web site at `http://www.ibm.com/software/pervasive/ws_everyplace_access/support/` to download the latest fix packs and code fixes. Apply these fixes prior to using the samples within this redbook.

4.4.1 Desktop Pass-through Proxy installation

To install the Desktop Pass-through Proxy component in the desktop PC, follow these steps:

1. Locate the installation program. It is located in the Everyplace Client CD in the \client\Zaurus\en\Desktop Passthru Proxy directory. Double-click the **setup.exe** file.
2. Click **Next** in the Welcome window.
3. Choose the destination location and click **Next**.
4. Click **Install** to begin the installation.
5. The installer creates an icon on the desktop to launch the program.

4.4.2 Desktop Pass-through Proxy configuration

You will need to configure the Desktop Pass-through Proxy only if you want to change the listening port. You would do this if you have another application using the pass-through port. By default it uses port 80, which is commonly used by HTTP servers. If you want to change the listening port, follow these steps:

1. Start the Desktop pass-through, if it is not started yet, by clicking its Desktop icon. The icon appears in the Windows tray bar, as shown in Figure 4-31.



Figure 4-31 Desktop pass-through tray icon

2. Right-click the **Desktop Pass-through** icon and select **Settings** from the menu.
3. In the Settings window, type an appropriate value for the port, as shown in Figure 4-32. Click **OK**.



Figure 4-32 Setting the Desktop Pass-through listening port

4.4.3 Everyplace Client installation

The Everyplace Client for Zaurus devices is packaged in the Zaurus Install Package (IPK) format. Therefore there are many methods to install the client. In this chapter, we use the Zaurus File Transfer program to copy the installation files to the device, and we use the Add/Remove Software program in the Zaurus to install the package.

To install the Everyplace Client in your Zaurus device, follow these steps:

1. Connect the Zaurus device to the cradle. Make sure that the Zaurus device has connectivity with the desktop PC.
2. Start the Zaurus File Transfer program. For a standard installation, select **Start -> Programs -> Sharp Zaurus 2 -> Zaurus File Transfer -> Zaurus File Transfer**. Wait while the File Transfer connects to the Zaurus device.

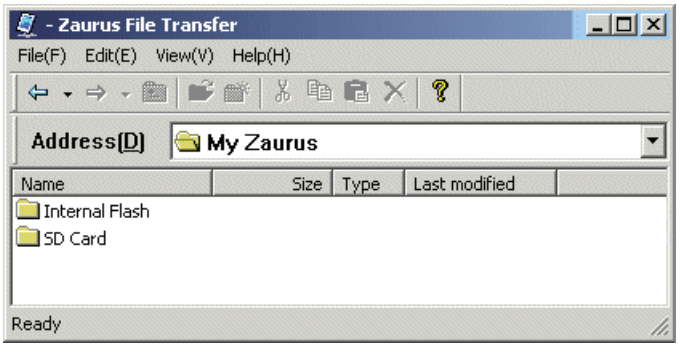


Figure 4-33 Zaurus File Transfer window

3. In the File Transfer window, navigate to the **My Zaurus -> Internal Flash -> Install Files** folder.
4. Locate the Zaurus installation packages for Everyplace Client. They are located in the Everyplace Client CD in the \client\Zaurus\en\runtime directory. The files to install are:
   - edgelet_1.2_arm.ipk
   - everyplaceclient_4.3-bm008_arm.ipk

Important: To ensure you have the latest code, you should visit the WebSphere Everyplace Access support Web site at `http://www.ibm.com/software/pervasive/ws_everyplace_access/support/` to download the latest fix packs and code fixes. Apply these fixes prior to using the samples within this redbook.

5. Drag the files from the Everyplace Client CD folder to the File Transfer window. Wait while the files are transferred to the device. At the end you will see the files in the File Transfer window, as shown in Figure 4-34 on page 108.

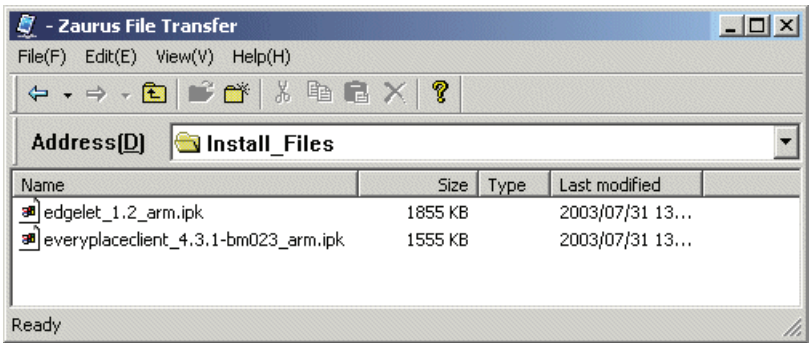


Figure 4-34 Everyplace Client package transferred to the Zaurus

6. In the Zaurus device, select **Settings -> Add/Remove Software**. In the Add/Remove Software window, select **Install packages.** You will see two packages in the list, as shown in Figure 4-35.

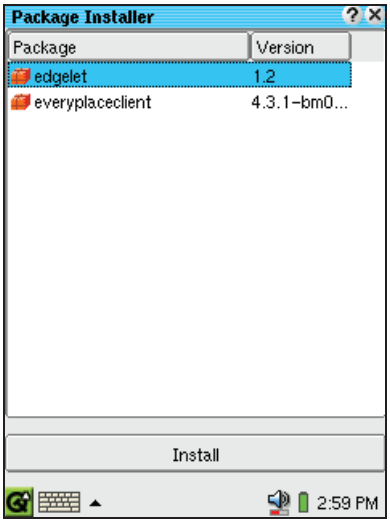


Figure 4-35 Package list for Everyplace Client

7. Select a package and select **Install,** or click directly in the package name. In the Install window, select the appropriate storage area and select **OK**. Wait while the package is installed. Select **OK** in the Finish install window. Close

the App/Remove Software application by selecting the **Close** button. Wait while the Zaurus updates the configuration.

8. Select the **Applications** tab. Verify that you have two new icons, one for Everyplace Client and one for Sametime Connect, as shown in Figure 4-36 on page 109.

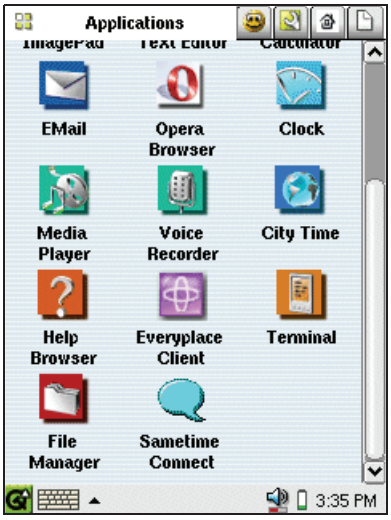


Figure 4-36 Everyplace Client installed

4.4.4 Everyplace Client configuration

When you first install the Everyplace Client in a Zaurus device, you have to perform some initial configuration for successfully connecting it to the WebSphere Everyplace Access server. Specifically, there are two basic configurations tasks to perform:

- Configure the user name and password for the synchronization session
- Configure the server or servers that have to be used for synchronization

The last step is to create network profiles. These profiles store the synchronization server names and ports and other necessary information. These profiles can be created using the Everyplace Client in the device, or they can be downloaded using Device Services. In this section, we show you how to use the Everyplace Client to create these profiles. For information about the use of Device Services, refer to Chapter 28, “Device Services” on page 1015.

To configure the Everyplace Client for the initial synchronization, follow these steps:

1. Start the Everyplace Client in the Zaurus by selection its icon. Wait while the program is launched.
2. On the Login page, enter the user information, as shown in Figure 4-37 on page 110. Select **Log in**.

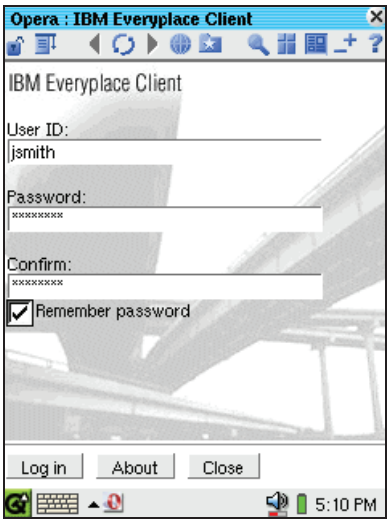


Figure 4-37 Enter user information

3. In the Everyplace Client main window, select the wrench icon to access the Settings page, as shown in Figure 4-38.

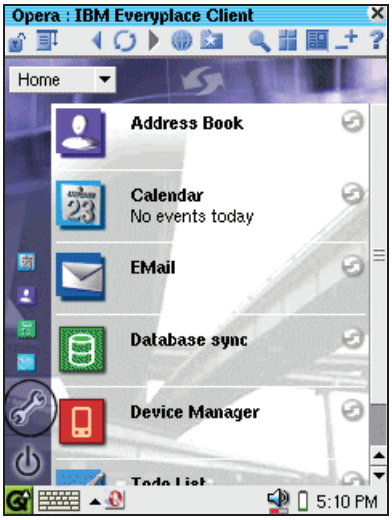


Figure 4-38 Entering the Settings page

4. In the Settings page, note a message (in red) that the network profiles are not yet configured. Select the **Network Profiles** icon, as shown in Figure 4-39 on page 111.

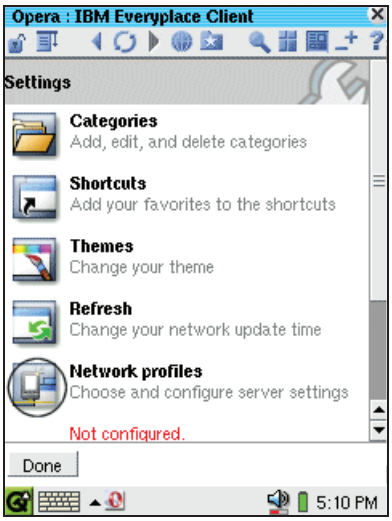


Figure 4-39 Configuring network profiles

5. In the Network profiles window, enter a name for the profile. Select **OK**.

6. Now you have to enter the information about the servers that you will use to synchronize. Note that you could have a server for each synchronization service or one for all the services. It depends on the architecture of the solution. In this case, we have all services running together in one server. Therefore, we need to introduce the same server name for all services, as shown in Figure 4-40 on page 112.

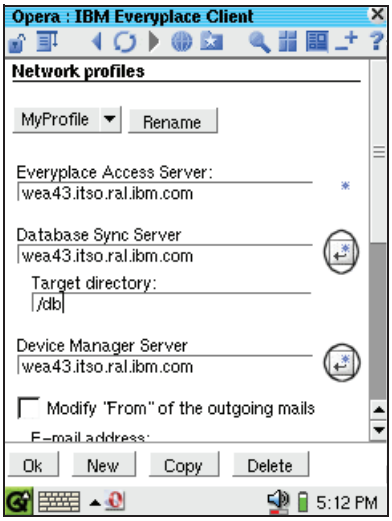


Figure 4-40 Entering the server information

Tip: You can use the icons near the server name field (as highlighted in Figure 4-40 on page 112) to copy the previously entered server name.

At the bottom of the Settings page, you can configure the IP address and port of the Desktop Pass-through Proxy. The port is the same one that you configured in 4.4.2, “Desktop Pass-through Proxy configuration” on page 106. The IP address is the one that you configured using the Zaurus Manager in the desktop and the PC Link application in the Zaurus. If you are using the defaults, leave these fields as is. Also you can configure the network timeout for the connection and choose if you want to use SSL, as shown in Figure 4-41 on page 113.

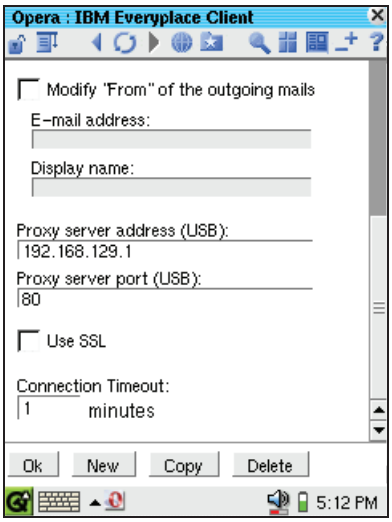


Figure 4-41 More settings

7. When you finish with the settings, select **OK** to save the changes. In the Settings window, note that the current profile being used is the newly created profile. Select **Done** to return to the Everyplace Client main window.

5

Everyplace Client secure connections

In this chapter, we describe how to enable Secure Sockets Layer (SSL) to secure access between the WebSphere Everyplace Access server and mobile devices using the Everyplace Client.

© Copyright IBM Corp. 2003. All rights reserved.

5.1 Introduction

To prevent exposing data exchanged between the client and the server to malicious agents, it is necessary to turn on a secure connection using Secure Sockets Layer (SSL). Not all the clients support SSL. Table 5-1 lists the SSL support by client and device.

Table 5-1 SSL support

Everyplace Client Component	PPC2002	Palm 4.x/5.x	Zaurus SL-5600
PIM Sync	Yes	Yes	Yes
Offline Browsing/Forms	Yes	Yes	NA
DB2e Sync	Yes	Yes	Yes
Device Manager Agent	Yes	Yes	No
Sametime Client	No	No	No

To have secure connections between the client and the server, the security mechanisms must be enabled on the WebSphere Everyplace Access Client, IBM HTTP Server, and WebSphere Application Server.

5.2 Enabling SSL on IBM HTTP Server

SSL uses key pairs (private and public keys) and certificates for ensuring security. You will need a Key database to store your keys and certificates. IBM HTTP Server comes with a Key Management Utility (IKEYMAN) to manage the key databases.

5.2.1 Creating a key database

1. Create a directory where the key database will be stored, for example C:\Program Files\IBM HTTP Server\Keys.
2. Start the Key Management Utility (IKEYMAN) by selecting **Start -> Programs -> IBM HTTP Server -> Key Management Utility**.
3. Click **Key Database File** and select **New**.

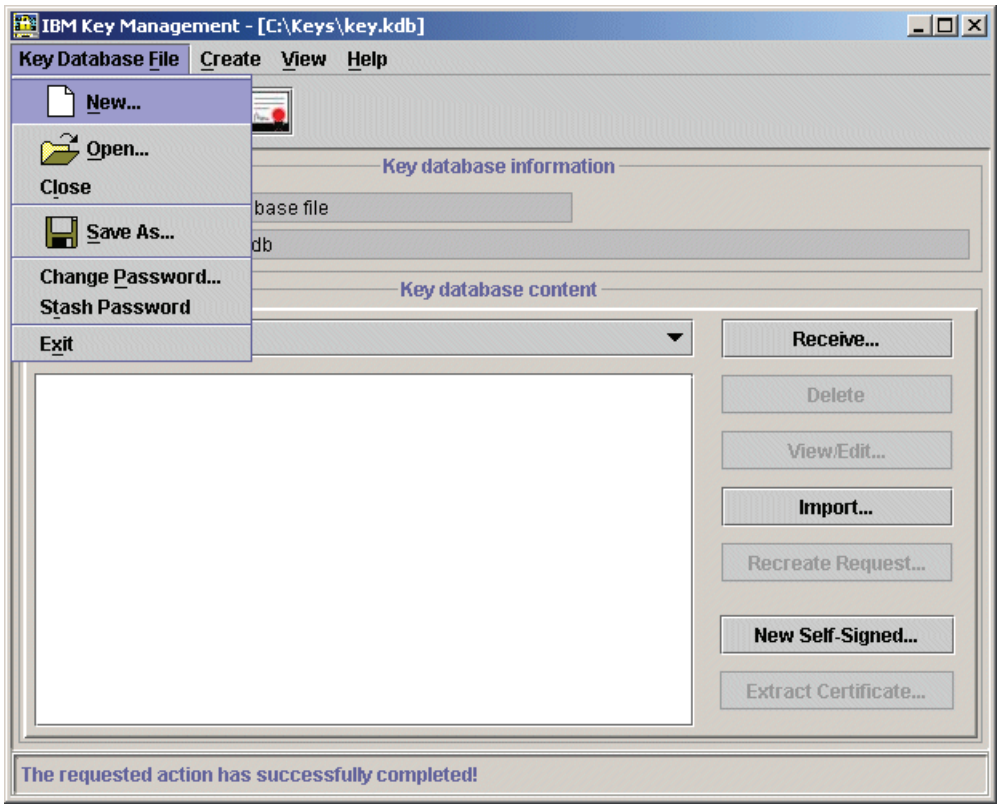


Figure 5-1 IBM HTTP Key Management Utility

4. Enter the name of the key database and select the location of the directory you created in the step 1 to store the key database. Click **OK**.

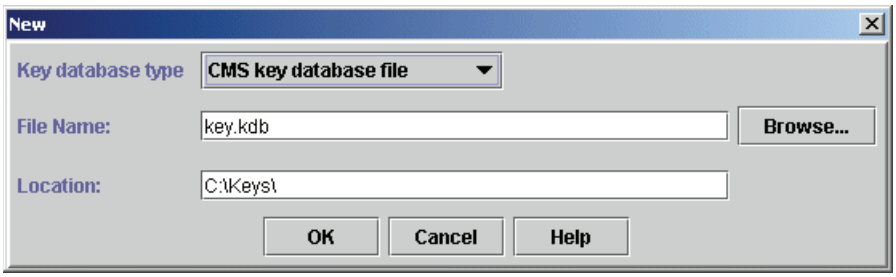


Figure 5-2 Create new key database

5. Enter the password. Check **Stash the password to a file** and click **OK**.

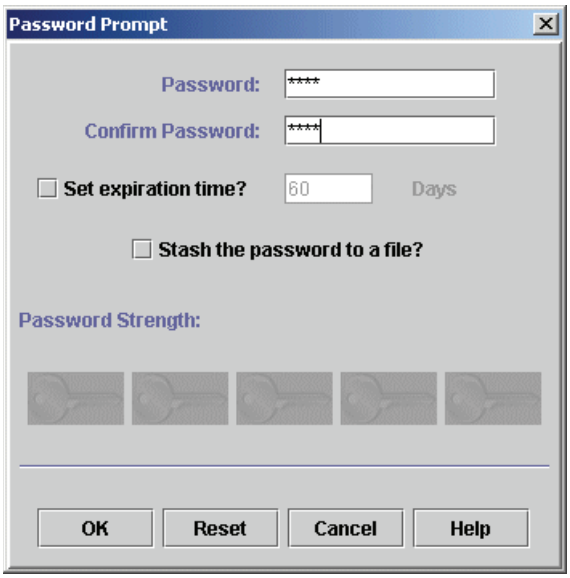


Figure 5-3 Set password for the key database file

5.2.2 Create a self-signed key file

You can either create a self-signed certificate or use a certificate from a well-known Certificate Authority (CA).

Follow these steps to create a self-signed certificate:

1. In IKEYMAN, select **Key Database File** from the main menu (see Figure 5-4 on page 119) and select **Open**.
2. Select the key database file just created and click **OK**.
3. Enter the correct password when prompted.
4. Select **Person Certificates** in the Key Database content frame and click the **New Self-Signed** button.

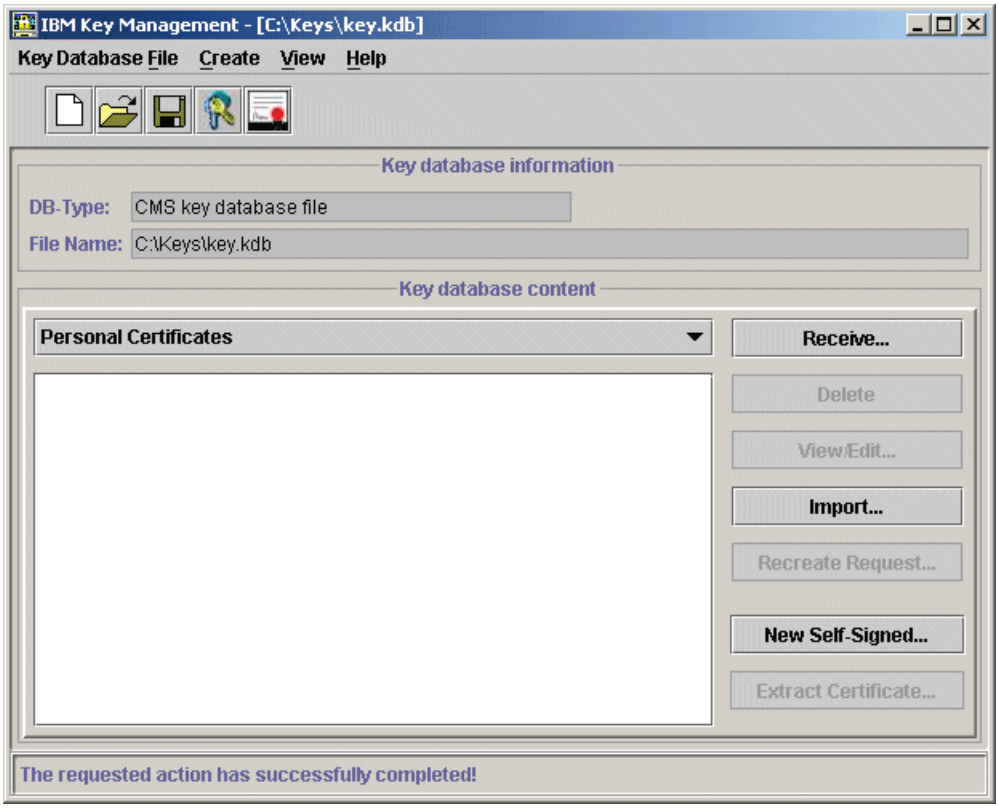


Figure 5-4 Create self-signed key

5. In the Create New Self-Signed Certificate window, enter the necessary information and click **OK**. Notice that some fields are optional.

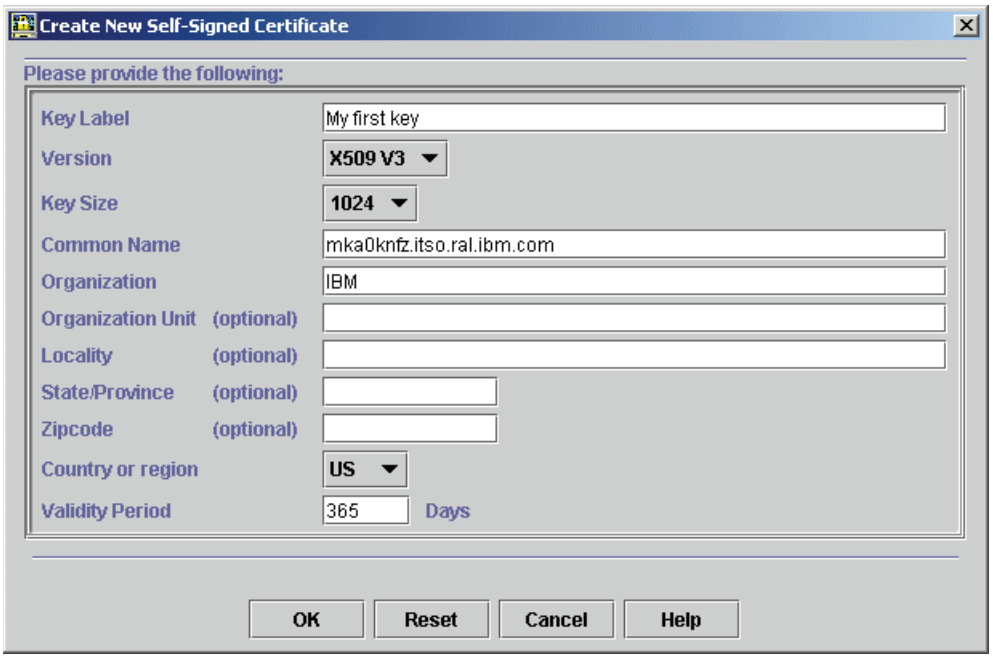


Figure 5-5 Create new self-signed key

6. Exit IBM HTTP Key Management Utility.

5.2.3 Setting up IBM HTTP Server

IBM HTTP Server configuration file must be changed in order to enable security. For example:

1. Open the IBM HTTP Server configuration file *Installation_Directory*\conf\httpd.conf.
2. Locate the following line:

   ```
   LoadModule ibm_app_server_http_module
   ```

3. After this line add the following line

   ```
   LoadModule ibm_ssl_module modules/IBMModuleSSL128.dll
   ```

4. At the end of the file add the following lines. The text shown in italics should be edited for your environment.

   ```
   Listen 443
   <VirtualHost yourservername:443>
   ServerName yourservername
   ServerPath "c:/program files/ibm http server"
   DocumentRoot "C:/Program Files/IBM HTTP Server/htdocs"
   ```

```
Keyfile C:\keys\key.kdb
SSLV2Timeout 100
SSLV3Timeout 1000
SSLEnable
SSLClientAuth none
</VirtualHost>

Keyfile C:\keys\key.kdb
SSLV2Timeout 100
SSLV3Timeout 1000
SSLClientAuth none
```

5. Save and close the httpd.conf configuration file.
6. Restart IBM HTTP Server.

5.2.4 Verifying if security is enabled on IBM HTTP Server

The following procedure may be used to verify if security has been properly enabled:

1. Open a browser.
2. Type `https://yourserver` in the address bar.
3. If using a self-signed certificate, accept the certificate if prompted.

5.3 Enabling SSL in WebSphere Application Server

In this section, we show how to enable SSL in WebSphere Application Server.

5.3.1 Configuring WebSphere Application Server

Follow this procedure to configure WebSphere Application Server for SSL:

1. Open the WebSphere Application Server Administrative Console by selecting **Start -> Programs -> IBM WebSphere -> Application Server 4.0 -> Administrator's Console**.
2. Click **Virtual Hosts**.
3. Click the **Add** button.
4. Add `*:443` to the Host Aliases list.
5. Click **Apply**.
6. Expand nodes by clicking the +.
7. Right-click your server.

8. Click **Regen Webserver Plugin**.
9. Restart your application server.

5.3.2 Verifying if security is enabled on WebSphere Application Server

Follow this procedure to verify if SSL has been properly configured in WebSphere Application Server:

1. Open a browser.
2. Access the Portal home page: `https://yourserver/wps/portal`.
3. Accept the certificate if prompted.

5.4 Enabling SSL in Everyplace Client

In this section, we show how to enable SSL in the Everyplace Client on Palm and Pocket PC devices.

5.4.1 Enabling SSL in Pocket PC devices

1. Open the WebSphere Everyplace Client on the Pocket PC and log in.
2. Select **MySettings** in the combo box.

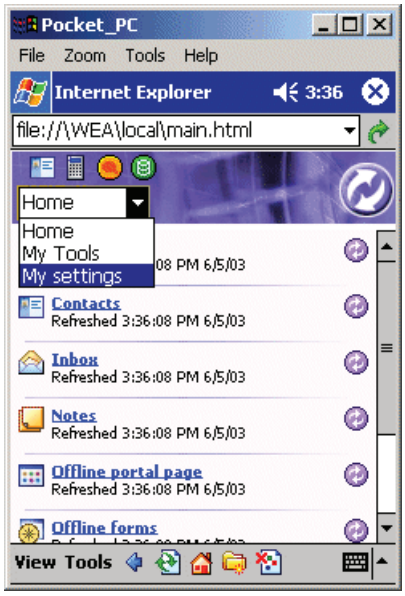


Figure 5-6 My Settings

3. Select the network profile being used to connect to the WebSphere Everyplace Access server.

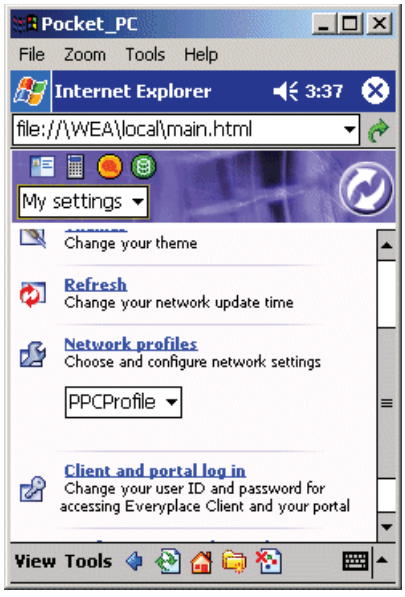


Figure 5-7 Network profiles

4. Enable **Use SSL connection for this profile**.

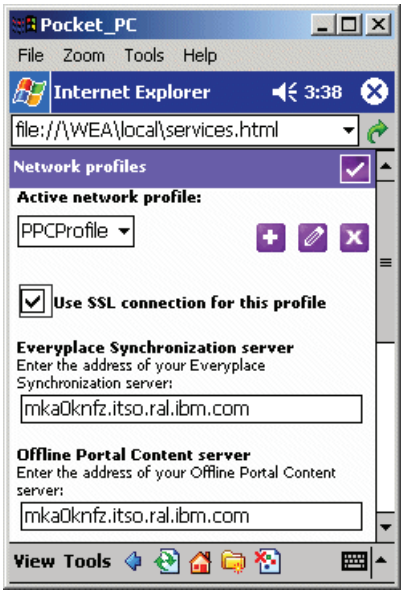


Figure 5-8 Use SSL Connection for this profile

The client will now use SSL when communicating with the server.

5.4.2 Enabling SSL in Palm devices

To enable SSL support on any Everyplace Client component, you first have to install your HTTP server certificate on your Palm device(s). Usually, the certificate is given to you by a Certificate Authority such as VeriSign or Thawte. Also you can use a self-signed certificate created with the IKEYMAN tool for development purposes, as explained in 5.2, “Enabling SSL on IBM HTTP Server” on page 116.

To install the server certificate on your Palm device, follow these steps:

1. Export the existing server certificate to a file named cacerts.bin using the DER format. For example, if you are using a self-signed certificate generated with the IBM Key Manager, follow these steps:
   a. Start the IKEYMAN tool by clicking **Start -> Programs -> IBM HTTP Server -> Start Key Management Utility** in your HTTP server machine.
   b. Open the generated key database for your server by selecting **Key Database File -> Open**. In the password window, enter the password for the database and click **OK**.

c. Select **Personal Certificates** in the combo-box to see the certificates contained in the key database, as shown in Figure 5-9. Select your server certificate and click **Extract Certificate**.

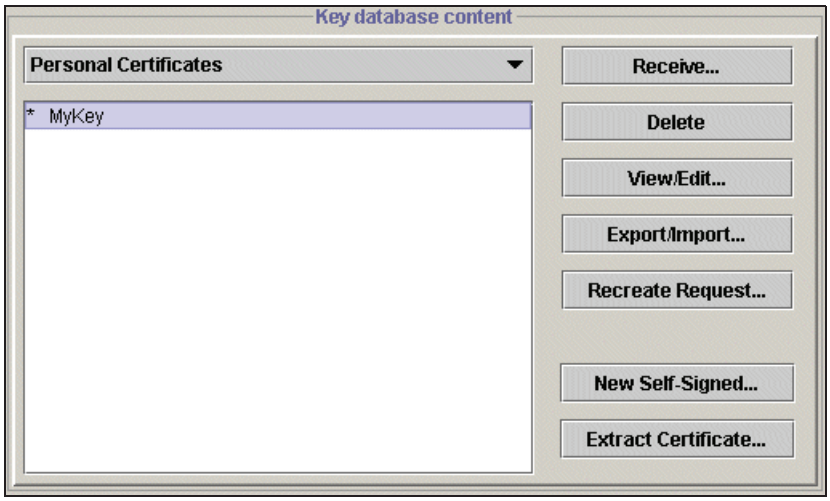


Figure 5-9 Self-signed certificates

d. In the Extract Certificate to a File window, select **Binary DER Data** as the data type, enter `cacerts.bin` as the certificate file name, and enter an existing directory name in the Location field, as shown in Figure 5-10. Click **OK**.

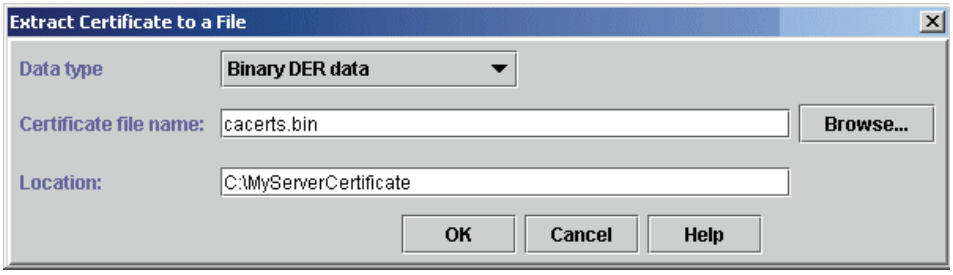


Figure 5-10 Extracting a certificate

e. Close the IKEYMAN tool.

2. Download the palmdb.exe utility from the WebSphere Everyplace Access Support page. Put the file in the same directory where you put the cacerts.bin file.

3. Run the palmdb.exe utility. You will see that the utility creates a new file named SSLCacerts.pdb. It contains the server certificate in the Palm Database format.
4. Install the SSLCacerts.pdb file in your Palm device by using HotSync synchronization, for a real device, or using the Install Database option from the emulator or simulator software.

To enable SSL in the supported clients, follow the instructions in the next sections.

For DB2 Everyplace Client

1. Open the IBM Sync client by selecting the corresponding icon.
2. Select **Options -> Settings** from the menu bar to open the Settings window.
3. On the Settings window, select **Advanced**.

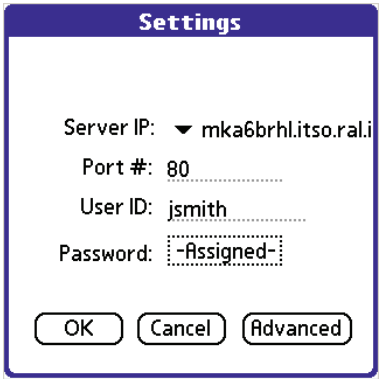


Figure 5-11 Settings window for DB2e client

4. Check **SSL (Secure Socket Layer)** and click **OK**.

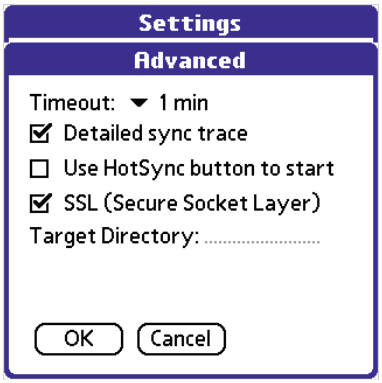


Figure 5-12 Selecting SSL on DB2e client

5. On the Settings window, make sure that the Port is set as 443. Click **OK**.

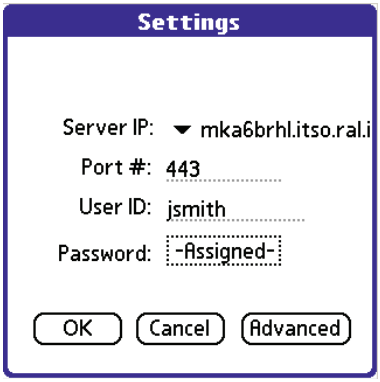


Figure 5-13 Checking the SSL server port

For Device Manager client (IBM Agent)

Refer to Chapter 28, “Device Services” on page 1015 for details about enabling SSL for Device Manager client.

For SyncML client

1. Open the SyncML client by selecting the corresponding icon.
2. Select **Preferences -> PIM Server** from the menu bar to open the PIM Server Preferences window.
3. Check **Enable SSL** and enter `443` in the Port field. Click **OK**.

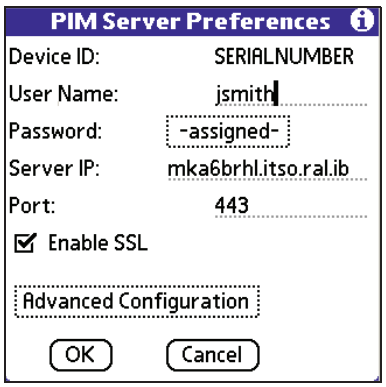


Figure 5-14 Enabling SSL on SyncML client

For Web Cache Client (WCC)

1. Open WCC by selecting the corresponding icon.
2. Select **Options -> Settings** from the menu bar to open the Settings window.
3. Check **SSL** and select **OK**.

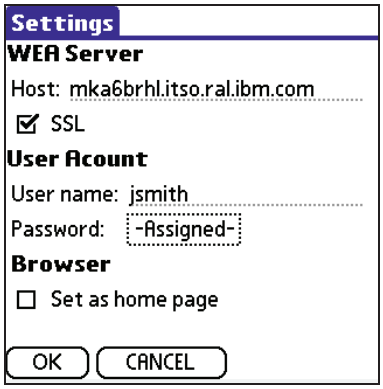


Figure 5-15 Enabling SSL on Web Cache Client

6

Mobile application development using portlets

This chapter explains the basics of portlets, portlet functionality, and portlet application projects. It will walk you through the steps of creating a project with the Portlet wizard and show you how to test and debug your portlet applications.

In this chapter, we cover:

- Portlet modes and states
- Portlets and WebSphere Everyplace Access
- Portlet projects
- Testing and debugging portlets

© Copyright IBM Corp. 2003. All rights reserved.

6.1 Portlets

Portlets are modularized Java programs that are used to encapsulate and organize content from various sources into a single user interface known as a portal, portal application, or portal page. The content can come from various sources including Web sites, e-mail, databases, and business applications. Portlets enable a user to organize a highly customized and productive work environment for projects, tasks, or group interaction.

A portlet is visible as one window of a Portal page. The portlet only provides the content inside the window and not the window itself.

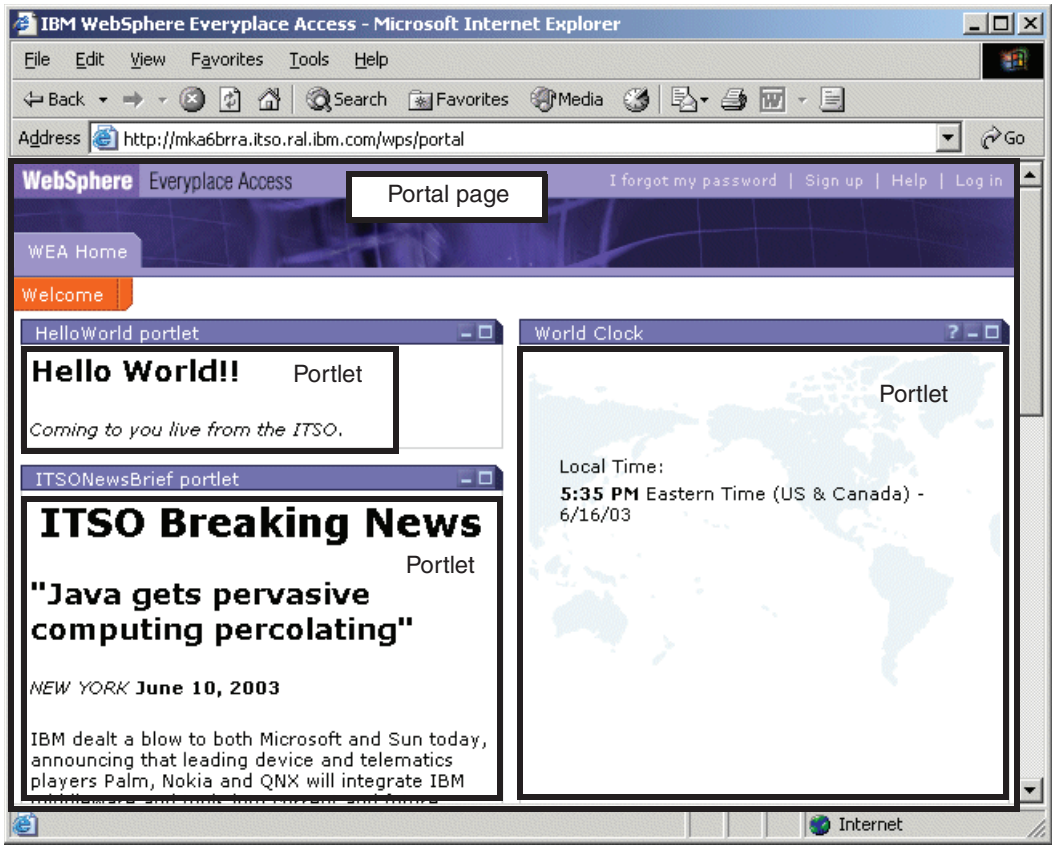


Figure 6-1 Portlets highlighted on a sample Portal page

To a programmer, a portlet is a server-side application that inherits from the javax.servlet.http.HttpServlet class and is treated as a servlet by the application

server. Portlets are executed within a portlet container and are managed by the application server.

Everyplace Access provides an assortment of pre-made productivity and personal information manager (PIM) portlets that have been enhanced specifically for Palm and Pocket PC devices using WML or HTML-based browsers. The included portlets will also work on other devices. A sampling of these portlets and their functionality is shown in Table 6-1. For more information on using these portlets, see Chapter 3, “Enhanced portlets” on page 63.

Tip: See the WebSphere Everyplace Access InfoCenter for a complete listing of included portlets and their functionality on various mobile devices.

You can find hundreds of additional pre-made portlets via the online Portal Catalog, found at:

`http://www-3.ibm.com/software/webservers/portal/portlet/catalog`

Portlets are available from multiple vendors, including IBM, at varying costs.

Table 6-1 Sample included portlets

Portlets	PDA markup (Pocket PC, Opera for Linux, Nokia 9210i, 9290)	PDA text-only (Eudora and WebPro for Palm OS)	WML markup (Ericsson T86, AU browser for Palm OS)
Lotus Notes Mail	All	All	View, Forward, Reply (All)
Lotus Notes Calendar	All	View, Create, Delete	View
Lotus Notes Contacts	All	View, Create, Delete	View
Microsoft Exchange 2000 Mail	All	All	View, Forward, Reply (All)
Microsoft Exchange 2000 Calendar	All	View, Create, Delete	View
Microsoft Exchange 2000 Contacts	All	View, Create, Delete	View
Bookmarks/Quick Links	Edit, View	View	View
POP3/IMAP Mail	All	View, Create, Delete, Reply	View, Create, Delete, Reply
Image Viewer	Edit, View	View	View

However, these portlets may not always provide the functionality you need or work the way you want. In these cases, it may be necessary to code your own portlets.

6.1.1 Portlet terminology

Before we go any further, let's review some portlet development terminology. You'll need to understand these terms to understand the following chapters. For more in-depth discussion on portlet terminology and concepts, refer to the online help or review the redbook *IBM WebSphere Portal V4 Developer's Handbook,* SG24-6897.

Portal application

A Portal application is a group of logically associated portlets. At minimum, a portlet application will define a single portlet, but many portlet applications will contain several portlets.

Portal page

A Portal page aggregates the content of one or more portlets and displays the content to the user. If the portlets are logically associated, the Portal page is referred to as a Portal application.

Portlet

A portlet is a Java application that displays the content for one portlet window.

Portlet API

The Portlet API (application programming interface) is a very useful document that describes the methods and technical specifications for interfacing with portlet applications.

The Portlet API can be found in the Site Developer online help.

The most current API information can be found at:
`http://www7b.software.ibm.com/wsdd/zones/portal/`

In-depth discussion of the Portlet API and accompanying code examples can be found in Chapter 2 of *IBM WebSphere Portal V4 Developer's Handbook*.

Portlet events

IBM WebSphere gives you the ability to develop portals that are dynamic in nature, displaying content based on the user's interaction with individual component portlets. When a user interacts with a portal, a portlet event is generated that contains information about the event.

There are three different types of events:

- **Action events** are generated when an HTTP request is received by the portlet container that is associated with an action. An example of this action is when a user clicks a Submit button on a form.
- **Message events** are generated when one portlet sends a message to communicate with other portlets inside a Portal application. For example, a customer number entered in one portlet of an invoice portlet application would enable another portlet to display the order history for that customer.
- **Window events** are generated when a user changes the state of the portlet window. For example, when a user maximizes a portlet to fill an entire screen, a window event is generated.

Events are relayed via their portlet containers to their respective listeners. It is important that individual portlet classes implement these listeners to respond to events.

It is also important to understand the sequence in which WebSphere handles portlet events:

1. **Action events are processed first**. During this process, the Portal application receives the request and decodes the action URI (Universal Resource Identifier) sent by the client and propagates the action event to the appropriate portlet. The receiving portlets action listener is then called to process the event.
2. **Message events are processed next**. Just as with action events, a portlet class must implement a message listener to process a message event. Portlets will continue to send, receive, and respond to messages until there are no more messaging events pending. Cyclical messaging is prevented by the WebSphere Portal architecture.
3. **Window events are processed last.** Operations to maximize, minimize, or restore a portlet's window size are processed until there are no more window events pending.
4. In the final step, the Portal aggregator invokes each portlet to render its contents and the resulting Portal page is displayed to the user.

Important: During the rendering process (step 4) of a Portal page, events are not processed. It is important that messages be sent *before* the portlet aggregation and rendering process for messages and events to be processed.

Portlet messaging

Portlets can communicate with each other using portlet actions and portlet messages. In portlet messaging, one portlet formats a message and sends this

message to the receiver of another portlet. The second portlet receives the message via the event handler and proceeds to process the message. Portlets can send, receive, and respond to messages during the second step of the event handling sequence detailed above.

Portlet messaging obeys the following rules:

- Portlets in different applications can only communicate through default portlet message objects. These objects can only carry strings.
- In order for portlets to implement custom messages, they must be component portlets of the same Portal application. This is because WebSphere Portal uses a unique class loader for each portlet application in order to provide security between applications. Since messaging portlets must share this custom message object, they must share the same class loader and therefore be part of the same application.
- For performance reasons, portlets that communicate through messaging must reside on the same page.

Portlet modes

When a portlet is called to render content, a mode indicator is passed in the portlet request object indicating how the portlet will be used. There are four modes:

1. **View**: The initial state of the portlet when created. This mode is how the user would normally view it. Think of it as the display mode.
2. **Edit**: The mode that allows the user to make a change to the portlet. In this mode, the user can enter, submit, and change data.
3. **Help**: Shows a help page to the user.
4. **Configure**: This mode is used by the Portal administrator to configure the portlet for its users.

The mode can be set by selecting the corresponding icon in the portlet window. Not all portlet modes may be available to all portlets. There may also be differences in functionality between markup languages. For example, a portlet may have view and edit functionality when rendered in HTML in a Web browser, but only view mode when viewed in WML on a WAP device.

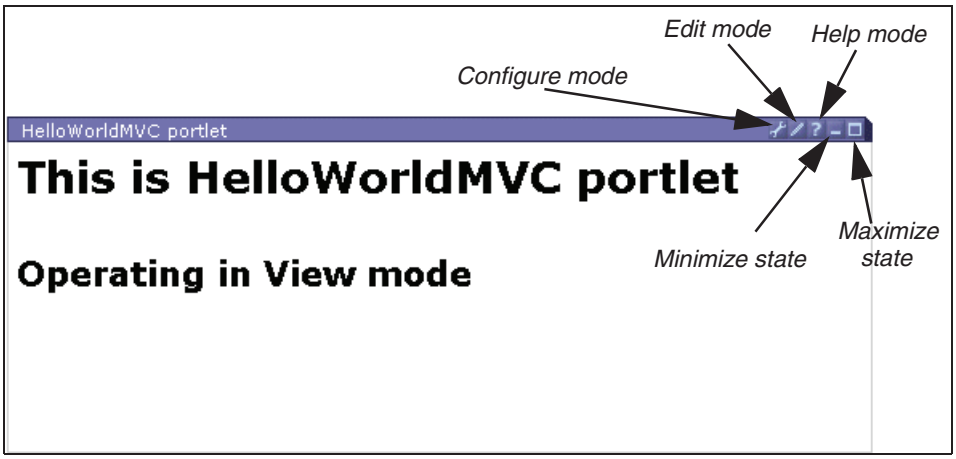


Figure 6-2 Some different portlet modes and states

Portlet states

Portlet states determine how the portlets will be displayed on a Portal page. There are three window states, which act in a way similar to application window controls in many popular operating systems. The three states of a portlet are:

1. **Normal**: The initial state the portlet is viewed in. A portlet can always be returned to its normal state from another state by clicking the Restore icon in the portlet window.
2. **Maximized**: A state in which one portlet fills the entire screen with its contents, hiding all other portlets on the portlet page. Portlets that can be maximized often have a special JavaServer Pages component to take advantage of the extra screen space to display more or more detailed content.
3. **Minimized**: Only the portlet title bar is visible.

The state of the portlet is stored in the PortletWindow.State object and can be used to display portlet content in different ways depending on its state.

Example 6-1 Portlet modes and window state

```
public void service (PortletRequest request, PortletResponse response) throws
PortletException, IOException
{  //Is the portlet running in View mode?
   if(request.getMode() == Portlet.Mode.VIEW)
   ...
   //Is the portlet window state set to Maximized?
   if((request.getWindow()).getWindowState() == PortletWindow.State.MAXIMIZED)
   {  //Set the maximized view JSP
      jsp = jspMaxView;
   }else
```

```
   {  //Set the normal view JSP
      jsp = jspNormalView
   }
   //Is the portlet running in Edit mode?
   if(request.getMode() == Portlet.Mode.EDIT)
   ...
   //Is the portlet running in Help mode?
   if(request.getMode() == Portlet.Mode.VIEW)
   ...
}
```

Just as certain modes may be made unavailable when rendered in certain markup languages, portlets can be configured so that only certain portlet states are available to a particular portlet. These options can be set in the portlet.xml file, which is part of every portlet application project.

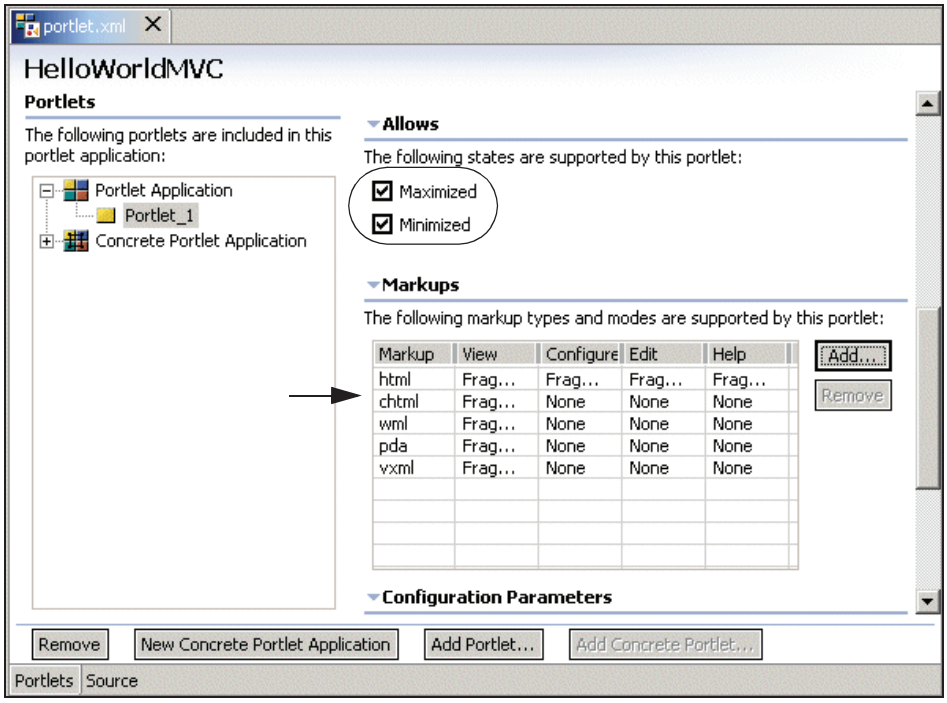


Figure 6-3 Portal.xml editor showing markups, modes, and states

6.2 How portlets work

WebSphere Portal is a J2EE application based on servlet technology. As previously mentioned, portlets inherit from the javax.servlet.http.HttpServlet class and are treated as servlets by the application server. However, portlets are instantiated by the portlet container, a thin layer implemented on top of the servlet container that is designed to reuse the functionality of the HttpServlet class. Portlet methods are accessed via the portlet API, which is very similar to the servlet API. There is more discussion on this later in this chapter, but first let us look into some portlet design methodology.

6.2.1 The Model-View-Controller (MVC) architecture

The Model-View-Controller (MVC) design pattern for portlet applications attempts to separate each portlet into separate, modularized, identifiable components that can be shared and reused with other portlets.

Having a well-thought-out design pattern will save time and effort in revising or updating later revisions of your portlet applications or performing maintenance tasks.

The Model

The Model is used to encapsulate all the business logic of a process and should not include any aspects of interpreting requests or displaying information. It should only be concerned with the business process going on. For example, in a banking portlet, the Model might receive an account balance, add a percentage of interest, and return the updated total. It would not, however, request the balance or display the results.

The data that the Model uses may be retrieved from an external data source and loaded into Java beans or arrive formatted in an XML document.

The View

The View displays output to the user. The View is usually either a JavaServer Page, when the data Model is implemented via Java beans, or an XSLT stylesheet, when the data is returned as a formatted XML document.

What the View returns will be aggregated into the portlet page, so it is important that any HTML the View returns must not contain page-level HTML tags.

The Portlet API provides tag libraries that aid in the creation of dynamic content.

The Controller

The Controller conducts the operation of the portlet. It must determine the requested mode (view, edit, help, or configure), execute the appropriate model, and select an appropriate view for the target device.

Controllers can be used to specify specific markup languages (such as WML for WAP-enabled devices or HTML for desktop PCs) to provide content formatted for the targeted device's Web browser.

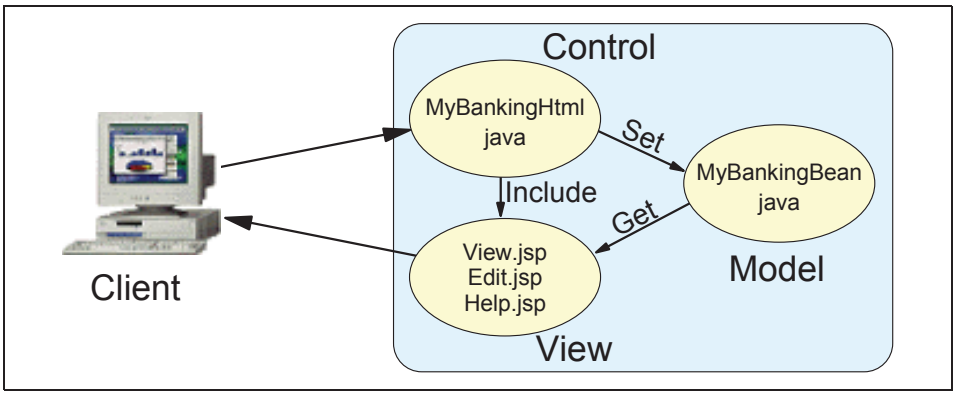


Figure 6-4 MVC sample scenario

6.2.2 The portlet life cycle

The portlet life cycle begins when the portlet container loads and creates an instance of the portlet class from the abstract portlet. The abstract portlet is used as a foundation for concrete portlets. Once you combine an abstract portlet with a set of configuration parameters, a concrete portlet is formed.

These settings are defined in the portlet.xml file, which is discussed in depth later in this chapter. Note that any number of concrete portlets may be defined from a single abstract portlet by using different configuration settings. When implementing all listener inferfaces that are linked with a portlet's life cycle, the following steps are executed:

1. init(PortletConfig config): After the portlet is constructed, it is initialized with PortletConfig (making it a concrete portlet). The Portal always shares only one instance of a portlet among all users, in exactly the same way a servlet is shared among all users on an application server.
2. initConcrete(PortletSettings settings): After being constructed but before being accessed for the first time, the portlet is initialized with PortletSettings.

The PortletSettings encapsulates the concrete portlet configuration parameter information.

3. login(PortletRequest request): The user portlet instance is initialized with PortletSession. Once the user logs in, the portlet container is able to associate a user with the portlet. You can define a default session for users who do not log in, but the customization provided by logging in will not be available.

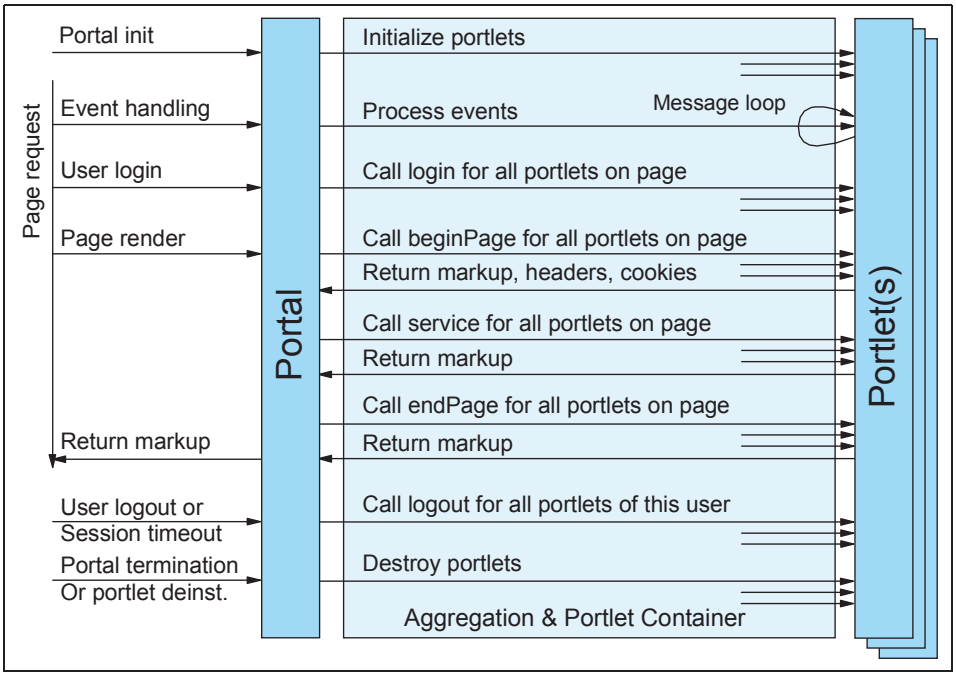


Figure 6-5 Portlet life cycle

4. beginPage(PortletRequest request. PortletResponse response): The portlet can render output at the beginning of each page for each request, but not in the portlet windows.
5. service(PortletRequest request. PortletResponse response): This method includes the doView(), doEdit(), doHelp(), and doConfigure() procedures that render the content for the corresponding portlet modes. It may be called many times to display portlet content, once for each request to the portlet.
6. endPage(PortletRequest request. PortletResponse response): The portlet may render content at the end of each page, for example a copyright notice or last-page-update information.
7. logout(PortletSettings settings): The user portlet instance is destroyed due to the user logging off or the user's session timing out. This method allows you

to manage user-specific information after the user has logged off and clean up user-specific resources.

8. destroyConcrete(PortletSettings settings): The concrete portlet instantiated earlier is taken out of service, such as would happen when an administrator decides to remove a concrete portlet during runtime or by halting the Portal Server.
9. destroy(PortletConfig config): This will destroy the individual abstract portlets that have been in use when they are taken out of service. This method includes garbage collection and finalization.

6.3 Portlets and WebSphere Everyplace Access

WebSphere Portal software allows companies to build, maintain, and administer custom Portal Web sites to serve the needs of their employees, business partners, and customers. The version of WebSphere Portal for Multiplatforms that is included with WebSphere Everyplace Access V4.3 is designed with Portal development for mobile devices in mind. Because of this, some features are not available via Everyplace Access delivery. The following features are not available in this release:

- **Portal Content Organizer**: A workspace for storing, navigating, viewing, and searching Portal documents and content
- **Live preconfigured content**: Preconfigured portlets that provide live weather, stock market, and news information
- **Click-to-Action**: A menu-driven method to transfer messages between portlets
- **WebSphere Content Publisher**: Web content management
- **Enterprise portlets**: Including SAP, Siebel, PeopleSoft, Oracle Financials, SQL Builder, and Lotus Domino Builder

6.4 Making a portlet project

When you install the Everyplace Toolkit, a new type of project is added to Site Developer. When you click **File -> New -> Other**, the category Portlet Development is seen in the list. The Portlet Application Project shown under this heading is used for developing portlet applications. Clicking **Next** starts the Portlet Application wizard.

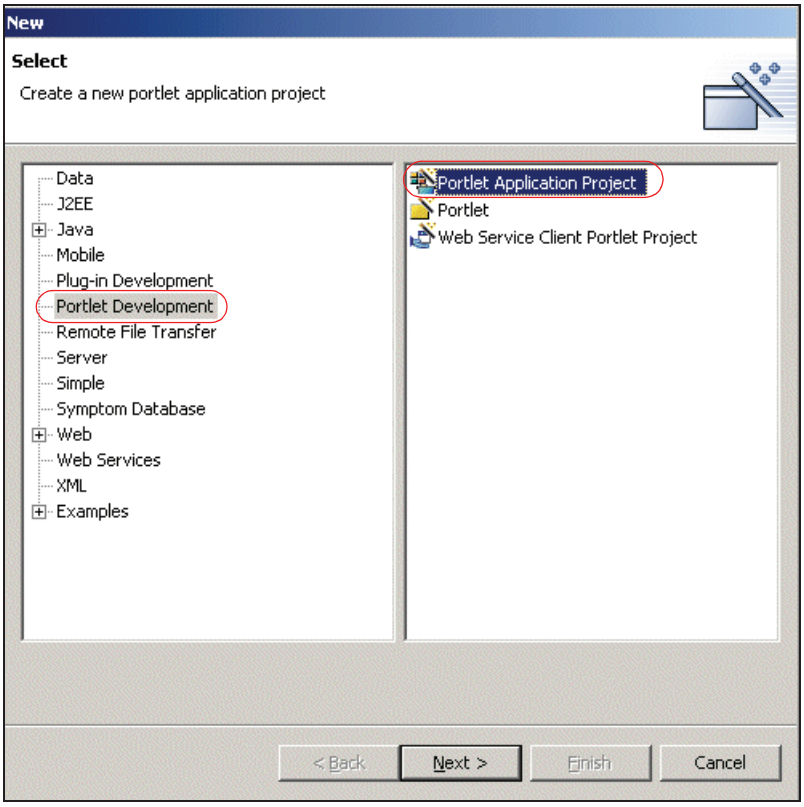


Figure 6-6 New application types

6.4.1 Defining a portlet project

The first step in the Portlet Application wizard is to define the portlet project.

You will need to supply the project name, project location, enterprise application project name, enterprise application location, and context root. These are:

- **Project name**: What you enter here will determine the names of the files and parameters used in all the components of the portlet project. It helps to make the project name descriptive of the project itself, for example NewsHeadlinePortlet or StockInfoPortlet.
- **Portlet location**: By default, your project is saved in the personal workspace you select when you run Site Developer for the first time. By default, this is in your \My Documents\IBM\wssd directory. If you would like to select a different location to store your files, uncheck the Use Default box and browse to a new location.

- **Enterprise project**: You may choose to either use a new or existing enterprise application project to contain all your project files. If you choose to use an existing project, you may click **Browse** to select the project. If you choose to use a new project, you must enter a name. The default is DefaultEAR. WebSphere Studio uses Enterprise Applications to contain portlet applications for debugging purposes; however, when deployed for actual use, portlet projects must be exported as Web archive (WAR) files, not deployed as EAR files.
- **Context root**: The context root is the Web application root, the top-level directory of your application when it is deployed to a Web server. Since this is the EAR context root, its role is limited to publication to a test environment for debugging purposes (remember that Portal Server does not recognize EAR files). The context root is used to maintain links as you move and rename files inside your project. By default, it takes the same name as the project name.

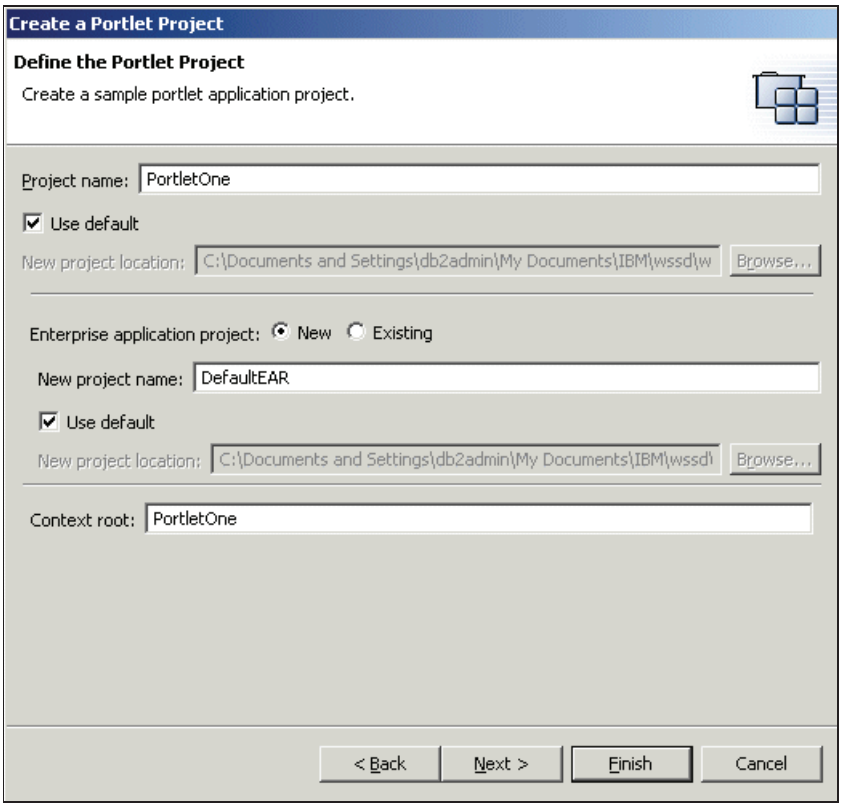


Figure 6-7 Defining a portlet project

6.4.2 Types of portlet projects

The next window will ask you to specify the type of portlet to generate. You have several selections:

- **None**: This is the default. It will create a blank portlet application, but not create a portlet or a concrete portlet. You will then have to manually add portlets to the application as well as the deployment descriptors. It is much easier to let the wizard set up a framework for your application than try to create one manually.
- **Basic portlet**: This is the most common. It will create a simple portlet class that extends the PortletAdapter class and contains implementations of the doView, doEdit, doHelp, and doConfigure mode adhering to the MVC approach. It also creates sample JSPs used to render portlet content and a sample Java bean that you can use to encapsulate business logic.
- **JSP portlet**: This will create a simple JSP file, a concrete portlet, and a concrete portlet application. Use this type of portlet if you intend for the entire application to be contained in a JavaServer Page.
- **MVC portlet:** This will create a MVC portlet, sample JSPs that are used in rendering the portlet, a sample Java bean, a concrete portlet, and a concrete portlet application. This choice utilizes the MVCPortlet provided as part of the com.ibm.wps.portlets package. The portlet relays calls to the controller classes dedicated to servicing a specific markup language. In other words, a separate controller bean will be constructed for each markup language you choose to support by selecting this option.
- **Servlet invoker portlet:** This creates a portlet that can easily include local or remote content into a portlet without writing any code, in addition to a concrete portlet and portlet application. If you use this option, you will be prompted to enter a URL pointing to the content you wish to serve. When this portlet is invoked, it will return unfiltered HTML.
- **XSL portlet:** This option creates a simple portlet that does basic XSLT transformation by specifying a stylesheet parameter and the given portlet URL. This wizard will also create XSL stylesheets, a simple XML document for the content provider, a concrete portlet, and portlet application.

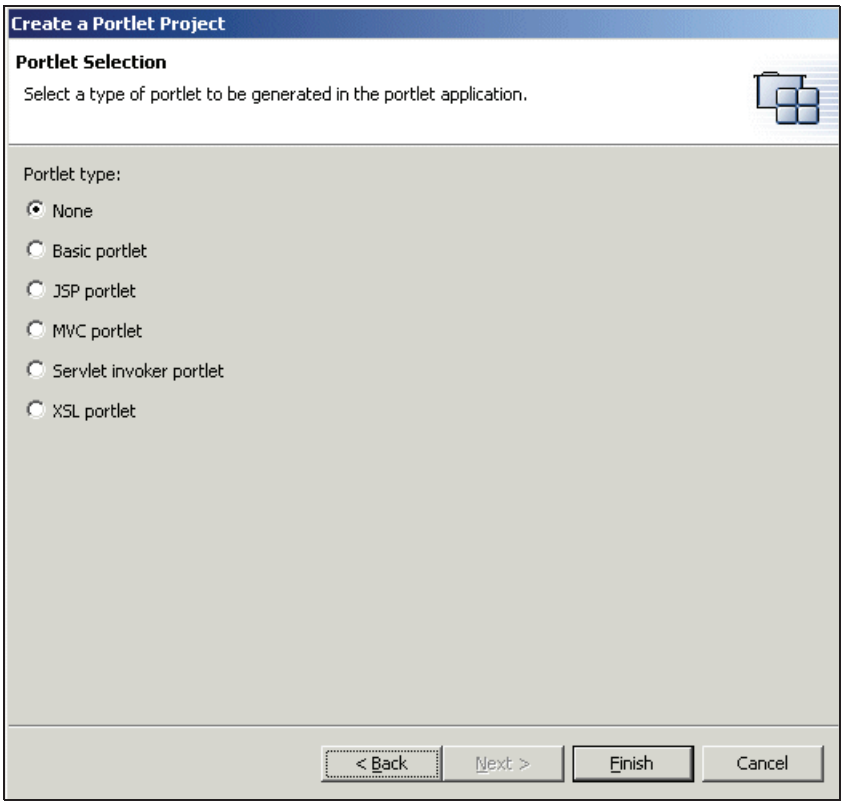


Figure 6-8 Types of portlet projects

6.4.3 Parameters of a portlet project

- **Portlet application name**: This is the name used by portlet.xml to specify the abstract portlet name and generally does not need to be modified.
- **Portlet name**: This value is used to identify the abstract portlet and generally does not need to be modified.
- **Concrete portlet application name**: This value is used by portlet.xml to specify the concrete application and generally does not need to be modified unless you intend to add more concrete applications to this portlet application. This name is used to identify the portlet to the administrator.
- **Concrete portlet name**: This name specifies the concrete portlet to be deployed. This name is used to identify the portlet to the portlet user.
- **Default locale**: Choose from the many country- and language-specific values to indicate the geographic location where the portlet will be used. This information is used for National Language Support (NLS), which is discussed

in depth in Chapter 11, “Portlet National Language Support (NLS)” on page 275.

- **Concrete portlet title**: Shown in the title bar of the portlet when displayed on a portlet page.
- **Portlet class name**: This name will be used as the name of the portlet created for you. Adjust this name to reflect the package name you would like yo use. By default, the wizard places the portlet in the default portlet package.
- **Markups**: You are given the option of supporting up to five markup languages for your target devices. Each selection results in the creation of a separate set of JSPs specifically for that markup.
  - **HTML**: For normal Web pages.
  - **cHTML** (compact HTML, sometimes also called iHTML): A subset of HTML that does not include complex HTML features such as layers, tables, image maps, and frames. It is typically used for i-mode enabled devices. cHTML is a subset of standard HTML but includes additional features such as special i-mode-only tags.
  - **WML** (Wireless Markup Language) for WAP-enabled devices (typically mobile phones): WML is a markup language based on XML that takes into account the constraints of limited bandwidth and small screen sizes.
  - **PDA markup**: A subset of HTML for small devices (PDAs). Selecting PDA markup will designate the portlet as an offline portlet and enable it to be used while not connected to the Internet as well as to synchronize data when it is connected to the Internet.
  - **VoiceXML** (VXML): This enables a portlet project to conform to VoiceXML specifications. Voice XML enables Web applications to respond to and with voice commands. It is useful in situations where it is not desirable for the user to use their eyes or hands to navigate a Web application, such as while driving.

    You may want to consider downloading and installing the Voice Application Access Toolkit for WebSphere Studio available at:

    `http://www-3.ibm.com/software/pervasive/products/voice/ws_vaat.shtml`

    Note: WebSphere Voice Server V3.1 is needed in order to configure a voice server running IBM speech recognition and text-to-speech technology. In addition, JSP and XSL portlets do not support VoiceXML and PDA markups.

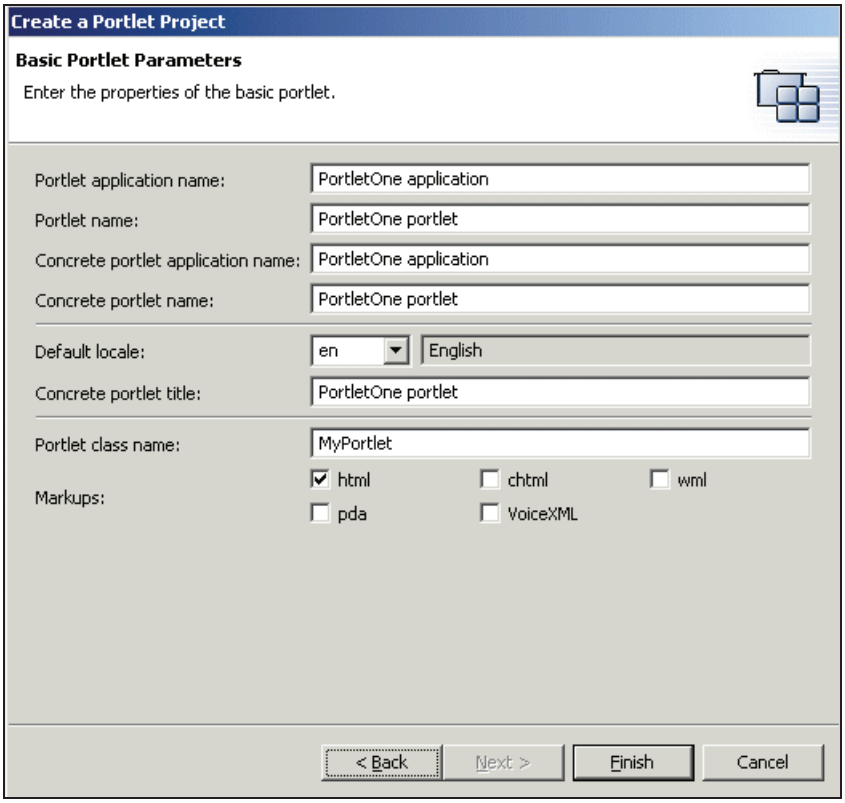


Figure 6-9 Portlet project parameters

6.4.4 Contents of a portlet project

Once a portlet is created, you can view its contents in the Navigator panel located in the upper-left corner of the Web perspective in the workbench.

The files are organized as follows:

In the Enterprise archive (EAR) file

/ The root directory of the EAR file structure.

/META-INF This directory contains the application.xml file. The application.xml file allows you to change settings for your portlet project, including changing security options, adding utility JARs and adding or removing WAR modules from the project. Double-clicking the application.xml opens it in its own editor.

Note the Module tab along the bottom of the editor (see Figure 6-10). You can use this to add or remove WAR modules (portlets) to and from your portlet application.

Example 6-2 application.xml content

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application
1.2//EN" "http://java.sun.com/j2ee/dtds/application_1_2.dtd">
<application id="Application_ID">
   <display-name>DefaultEAR</display-name>
   <module id="WebModule_1056388627695">
      <web>
         <web-uri>HelloWorldMVC.war</web-uri>
         <context-root>/HelloWorldMVC</context-root>
      </web>
   </module>
</application>
```

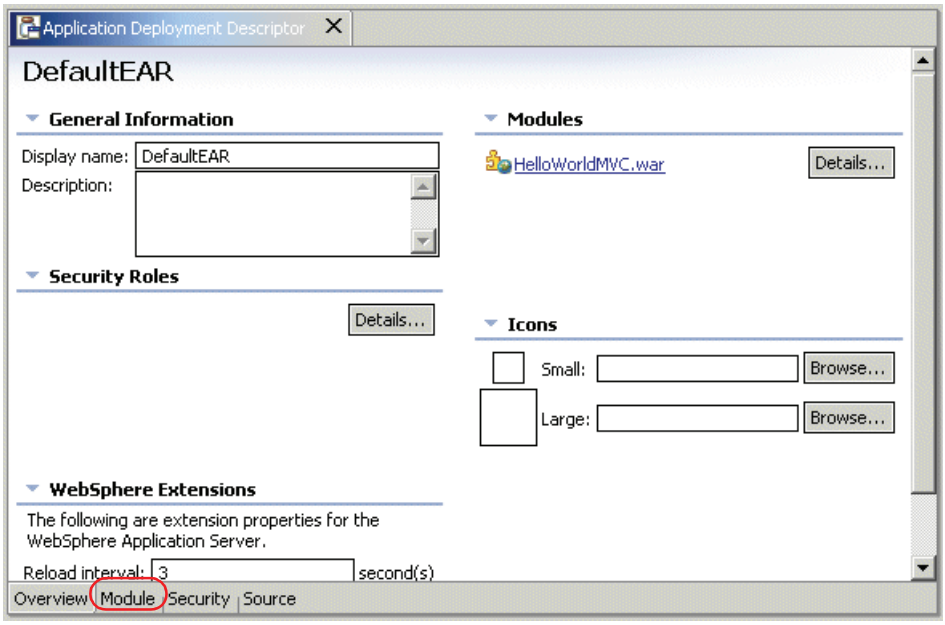


Figure 6-10 Application.xml editor

In the portlet project

/ The root directory of the portlet file structure.

/Java Source Depending on the type of portlet being developed, the Java Source directory will contain the Java source files for the portlet

and its Controller (corresponding to the Model and Controller of the MVC architecture).

Example 6-3 Sample MyControllerHtml.java code

```
package controller;

//*************************************************************************
//*
//* MyControllerHtml - A sample portlet controller based on
AbstractPortletController
//*
//*************************************************************************

import java.io.*;
import org.apache.jetspeed.portlet.*;
import com.ibm.wps.portlets.*;

public class MyControllerHtml extends AbstractPortletController {

   public void init(PortletConfig portletConfig) throws UnavailableException {
      super.init(portletConfig);
   }

   public void doView(PortletRequest request, PortletResponse response) throws
PortletException, IOException {
      // Make a bean
      MyControllerBean bean = new MyControllerBean();
      // Save name in bean
      bean.setPortletName("HelloWorldMVC portlet");
      // Save bean in request
      request.setAttribute("MyControllerBean", bean);

      // Invoke the JSP to render
      getPortletConfig().getContext().include("/jsp/View.jsp", request,
response);
   }

   public void doHelp(PortletRequest request, PortletResponse response) throws
PortletException, IOException {
      ...
   }

   public void doEdit(PortletRequest request, PortletResponse response) throws
PortletException, IOException {
      ...
   }

   public void doConfigure(PortletRequest request, PortletResponse response)
```

```
        ...
    }
}
```

/Web Content The Web Content folder contains all of the files used in the delivery of the portlet to its viewer.

-> /jsp The JSP files contained in the /jsp directory correspond to the view of the MVC architecture.

Example 6-4 Sample View.jsp code

```
<%@ page contentType="text/html" %>

<jsp:useBean id="MyControllerBean" class="controller.MyControllerBean"
scope="request" />

<h1>This is <%=MyControllerBean.getPortletName()%></h1>
<br>
<h2>Operating in View mode</h2>
```

-> -> /html The JSP files that display the content of the portlet are stored here. You may see more than one markup language directory. For instance, if your portlet also supports WML, you will see a /wml directory in addition to a /html directory. If you are using NLS, you will also see subdirectories under each markup language directory for each language supported, for example /html/de for HTML portlets and the German language, or /wml/ko for WML portlets and the Korean language.

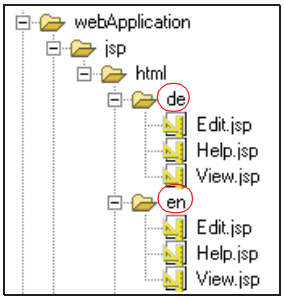


Figure 6-11 HTML support for German and English

-> /META-INF The MANIFEST.MF file is stored here. Double-clicking this file opens the JAR dependency editor, where you can update the class path for a utility JAR or module.

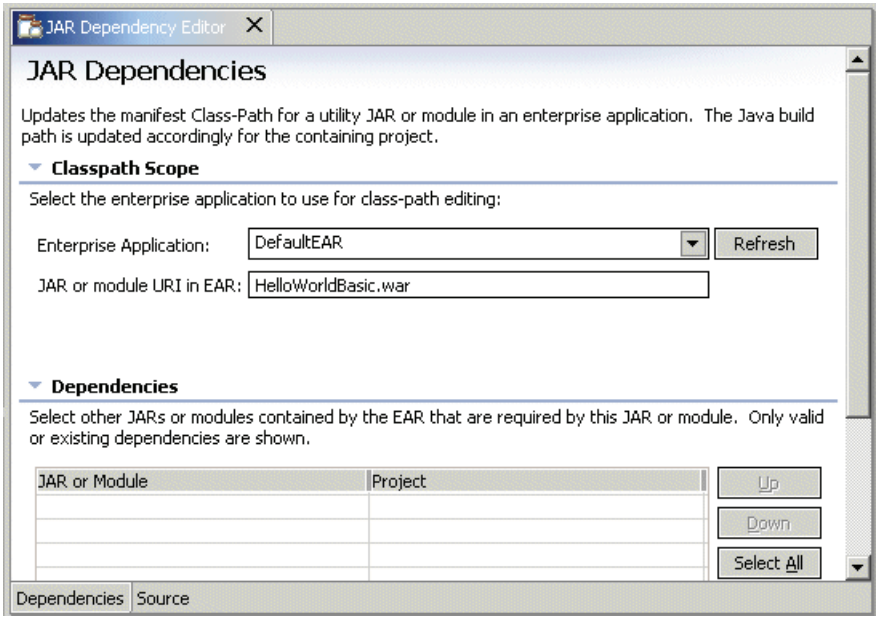


Figure 6-12 JAR dependency editor

-> /WEB-INF The WEB-INF folder is used to store the portlet descriptor and Web deployment descriptor, which contain configuration information for WebSphere Portal Server and WebSphere Application Server respectively. It also stores all of the runtime executable JAR files and classes that the portlet requires. Files in the WEB-INF folder are not served directly to the client.

-> -> /classes For storing compiled class files used by the portlet.

-> -> /lib For storing the portlet executable JAR files.

-> -> /tld For storing the TagLib declarations file for the Portlet API. It is used if you have included TagLib functions in your JSP file. You should copy the portlet.tld into this directory from /Site Developer installation directory/plugins/com.ibm.wps/.

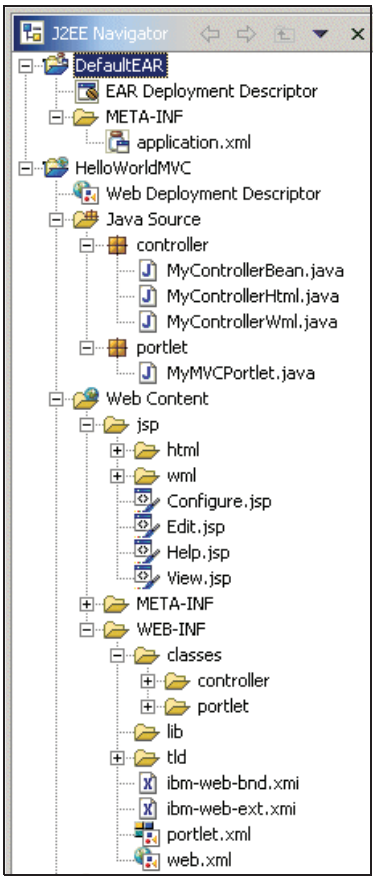


Figure 6-13 Navigator pane showing portlet project contents

Portlet.xml Portlet.xml contains deployment information for the portlet application. This information will be used by the Portal Server to set the initial configuration and retrieve parameters. Double-clicking portlet.xml will open it in its own portal.xml editor.

Example 6-5 portlet.xml content

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portlet-app-def PUBLIC "-//IBM//DTD Portlet Application 1.1//EN"
"portlet_1.1.dtd">
<portlet-app-def>
   <portlet-app uid="HelloWorldMVC.002482919ea5001719d6f3d19b875c9f"
major-version="1" minor-version="0">
      <portlet-app-name>HelloWorldMVC application</portlet-app-name>
      <portlet id="Portlet_1" href="WEB-INF/web.xml#Servlet_1"
major-version="1" minor-version="0">
```

```
            <portlet-name>HelloWorldMVC portlet</portlet-name>
            <cache>
               <expires>0</expires>
               <shared>NO</shared>
            </cache>
            <allows>
               <maximized/>
               <minimized/>
            </allows>
            <supports>
               <markup name="html">
                  <view/>
               </markup>
            </supports>
         </portlet>
      </portlet-app>
      <concrete-portlet-app
uid="HelloWorldMVC.002482919ea5001719d6f3d19b875c9f.1">
         <portlet-app-name>HelloWorldMVC application</portlet-app-name>
         <concrete-portlet href="#Portlet_1">
            <portlet-name>HelloWorldMVC portlet</portlet-name>
            <default-locale>en</default-locale>
            <language locale="en">
               <title>HelloWorldMVC portlet</title>
               <title-short></title-short>
               <description></description>
               <keywords></keywords>
            </language>
         </concrete-portlet>
      </concrete-portlet-app>
</portlet-app-def>
```

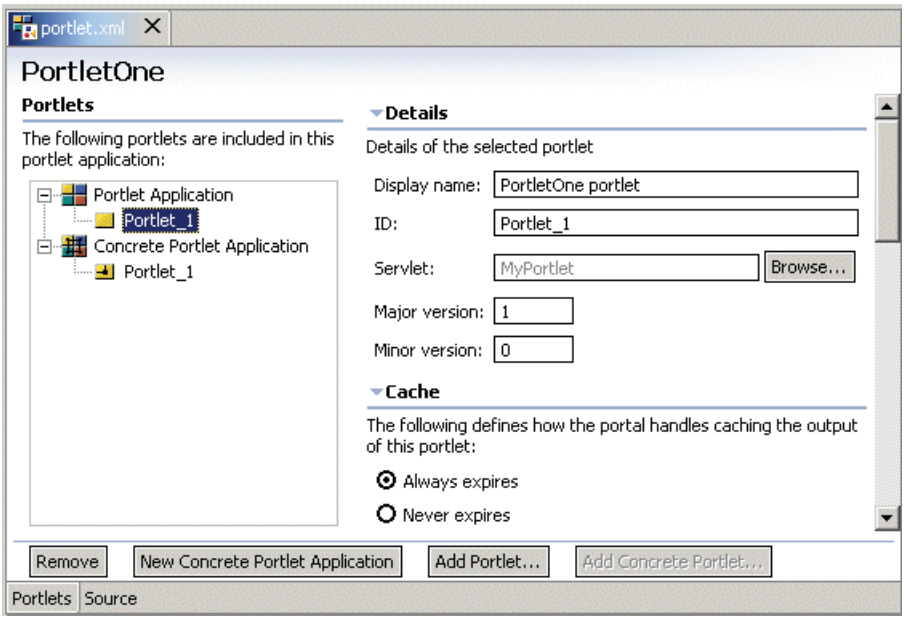


Figure 6-14 Portlet.xml editor

Web.xml Web.xml contains the deployment information used by WebSphere Application Server to register the portlets being deployed. WebSphere Application Server will not be aware of portlet configurations and only recognizes the installed portlet applications as Web applications containing servlets. Double-clicking this file will open it in its own Web deployment descriptor editor.

Example 6-6 web.xml sample content

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.2//EN" "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app id="WebApp">
   <display-name>HelloWorldMVC</display-name>
   <servlet id="Servlet_1">
      <servlet-name>MyMVCPortlet</servlet-name>
      <display-name>MyMVCPortlet</display-name>
      <servlet-class>portlet.MyMVCPortlet</servlet-class>
      <init-param>
         <param-name>controller.html</param-name>
         <param-value>controller.MyControllerHtml</param-value>
      </init-param>
   </servlet>
```

```
    <servlet-mapping>
        <servlet-name>MyMVCPortlet</servlet-name>
        <url-pattern>/MyMVCPortlet/*</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.htm</welcome-file>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

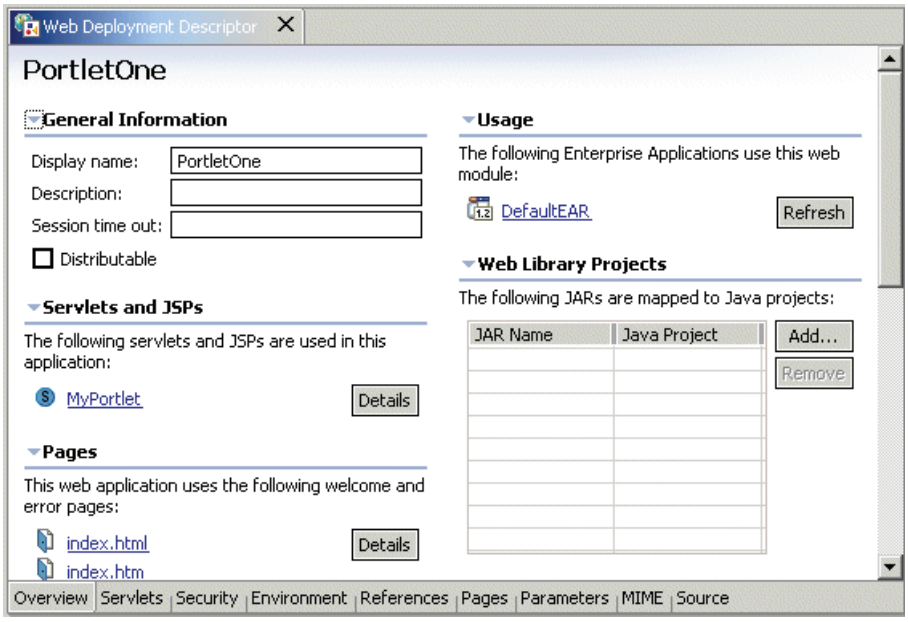


Figure 6-15 Web.xml - Web deployment descriptor editor

6.5 Testing and debugging portlets

As discussed in Chapter 7, “WebSphere Studio Site Developer and Everyplace Toolkit” on page 179, Site Developer has the following options with which to debug your portlets:

- **Remote Attach**: Useful for debugging PDA markup portlets and offline forms on a complex WebSphere Everyplace Access installation.

- **Remote Single Server**: Useful for debugging a portlet on either a local (outside of the development environment) or remote installation of WebSphere Portal Server, especially when multiple configurations or versions are being used.
- **Local debug**: Useful for debugging simple portlet applications within the WebSphere Studio development environment.

6.5.1 Setting up a test environment and running a portlet

In order to debug your portlet applications, you need to set up the environment that you want to use to debug your application. The easiest way to do this is to use the Server and Server Configuration wizard. To get there, select **File -> New -> Other -> Server -> Server and Server Configuration** from the WebSphere Studio Site Developer workbench.

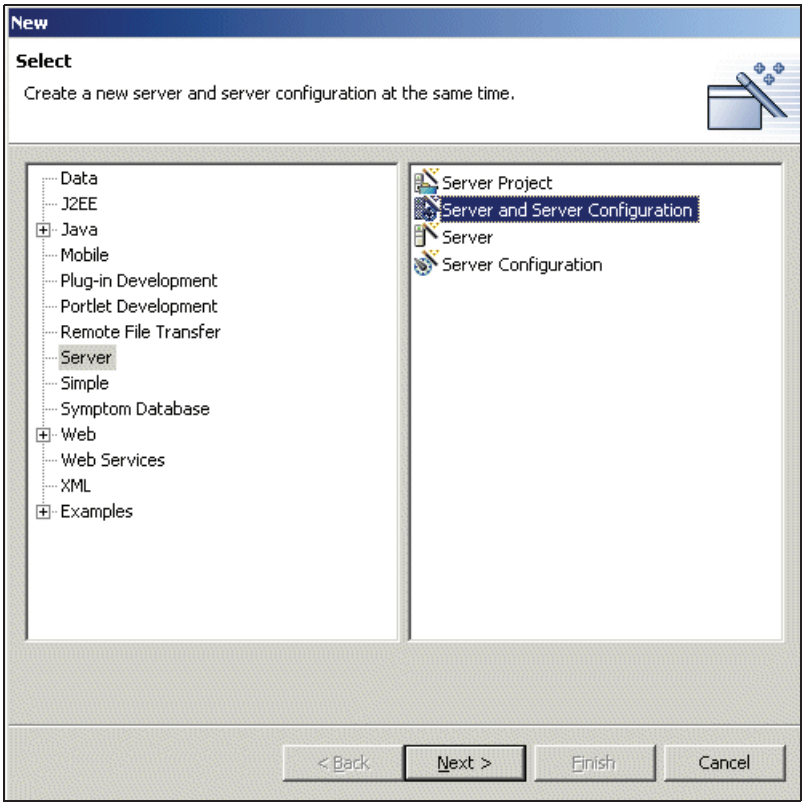


Figure 6-16 Setting up a new server and configuration

This starts the wizard. On the following window, you will select if you want to debug on a remote server, via remote server attach (for an instance of WebSphere that is already running), or in a local test environment (no external server needed). You have many server configurations to choose from, including the open source Apache Tomcat.

Note: WebSphere Studio does not support remote debugging on the Apache Tomcat server. Tomcat must be installed locally to use this feature.

You will be prompted for configuration information, including host name if you choose to use a remote server for testing purposes. The remote server will also have to be configured to allow remote testing and debugging.

See the online help for help setting up a remote server for testing and debugging.

Setting up a local test environment

The most convenient testing and debugging environment is within Site Developer itself. The following scenario will walk you through the steps of setting up a local test environment for WebSphere Portal V4.2:

1. Select **File -> New -> Other.** Then select **Server -> Server and Server Configuration** from within the WebSphere Studio Site Developer workbench.
2. Click **Next** to continue. You will be shown a configuration window like the one in Figure 6-17 on page 157. Enter `Test Environment WPS4.2` for the server name and click **Test Environment** under WebSphere Portal Version 4.2. Click **Next**.

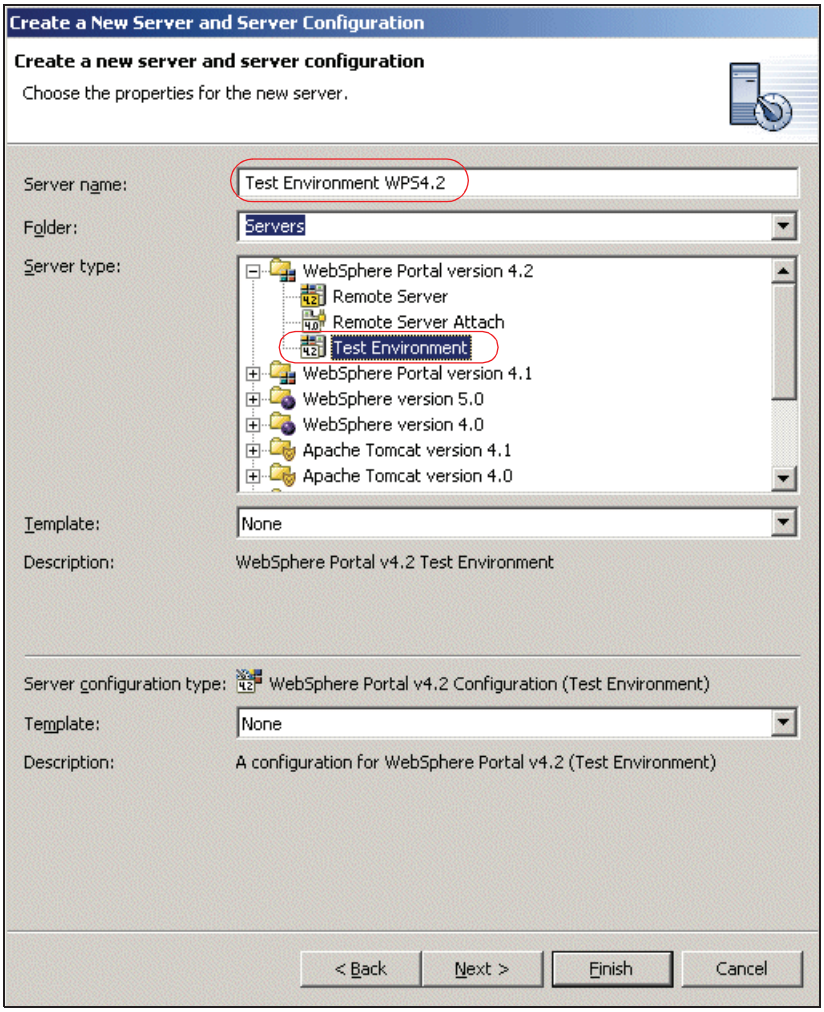


Figure 6-17 Test Environment wizard

3. Accept the default port of 9080, and click **Finish** to build the local test environment.

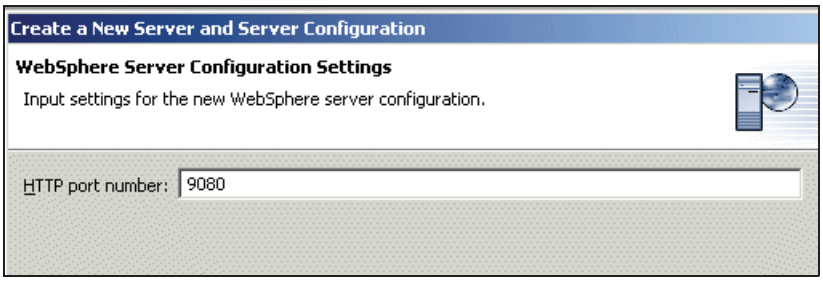


Figure 6-18 Default port

4. Test Environment will now be displayed on the Server Configuration tab of the Navigator pane.

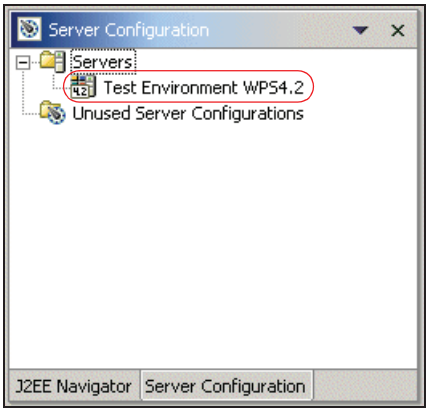


Figure 6-19 Server Configuration tab

5. To change the options for the test environment, double-click **Test Environment WPS4.2** to open the server configuration editor. This editor allows you to change the settings of the test server, add class paths, change port settings, and change Portal Server settings. See online help for reference.

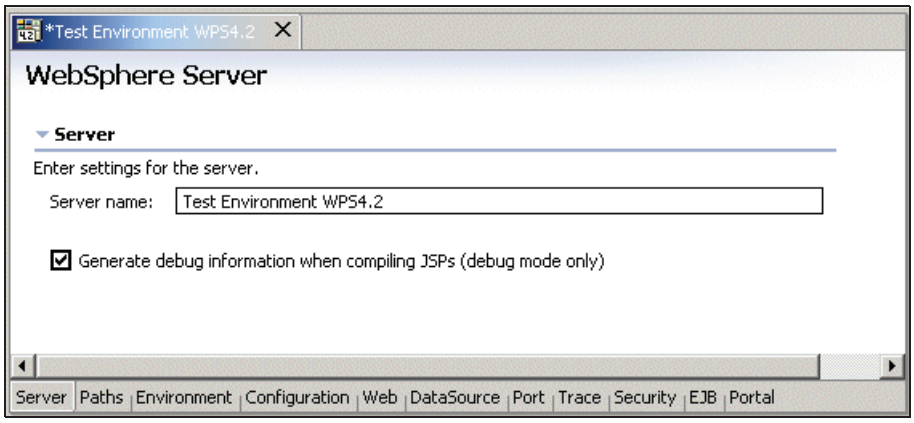


Figure 6-20 Test environment settings

6. You will need to add a Portal project to the test environment to run it. You can do this by right-clicking **Test Environment WPS4.2**, clicking **Add**, and then clicking **DefaultEAR** (or the name of the portlet project to add).

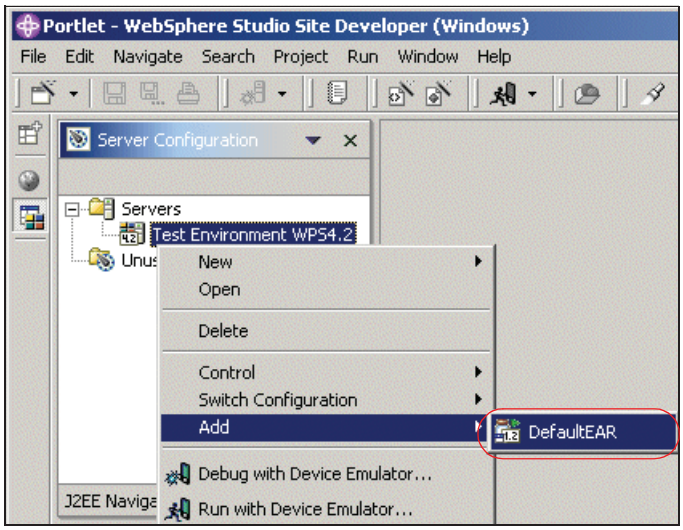


Figure 6-21 Adding a project

7. To run a project, just right-click the project name in the Navigator pane and click **Run on Server**.

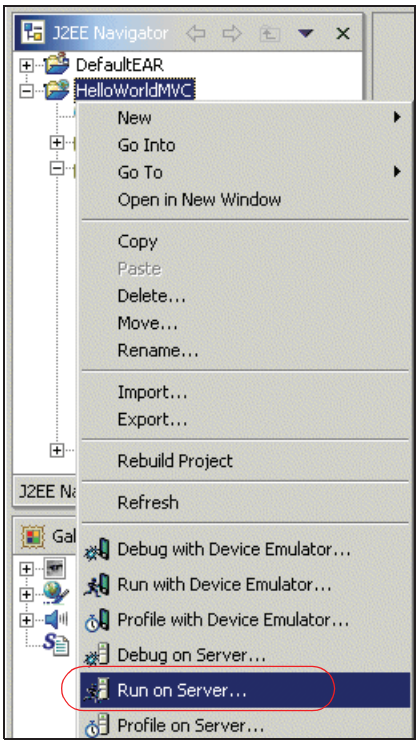


Figure 6-22 Running on Server

8. It may take some time to publish and load your portlet the first time you attempt this operation. Site Developer must start WebSphere Application Server for your project to run. Once this process is complete, your portlet will load in the Web browser.

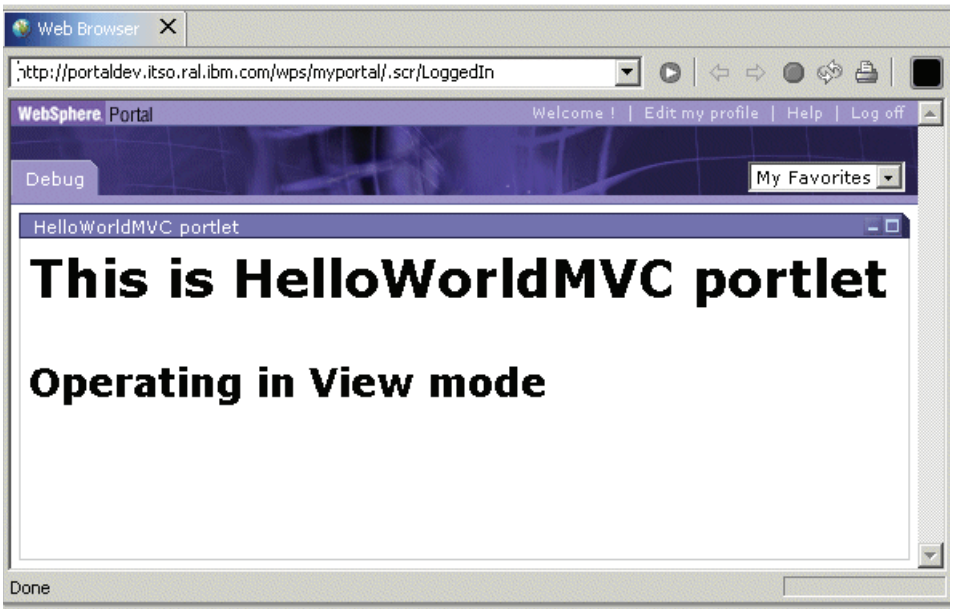


Figure 6-23 A running portlet

6.5.2 Testing portlet projects on Pocket PC, Palm, and WAP devices

In addition to the above testing methods, you may also find it extremely useful to test your portlet applications for each device it will support.

To do this, you have three options:

1. **Test the portlet on the device itself**. To do this, you will need to achieve connectivity between the device and the test computer. For example, if you are using a PDA, use the cradle. If you are using a Bluetooth or wireless Ethernet (802.11x) device, make sure you can access the test computer via the wireless connection.
2. **Test the portlet on a device emulator**. A device emulator uses the actual ROM of the device to execute the portlet. Using an emulator will give you the most accurate representation of how your portlet will perform on the actual device. An emulator is a piece of software run on the test machine.
3. **Test the portlet using a simulator**. In some cases, a simulator is necessary to run the project due to differences in processor or other hardware architecture between the device and development machine. A simulator can be used to verify the functionality of a program and in many cases will accurately portray the execution of the portlet application. A simulator is also software run on the test machine.

In some cases, such as when working with portlets that only support WML, testing with the device itself or with a simulator may be the only ways to view how the portlet will look and function.

When you install the Everyplace Toolkit, a new option is added to WebSphere Studio Site Developer called Run with Device Emulator. This option allows you to test your portlets directly with a simulator or emulator.

The following scenario details how to test a portlet that supports only WML with the mobile phone simulator included with the Openwave SDK 6.1. You can get more information about Openwave development and installation instructions for the Openwave SDK at http://developer.openwave.com/

1. To begin, click **Window** from the file menu, then click **Preferences.**

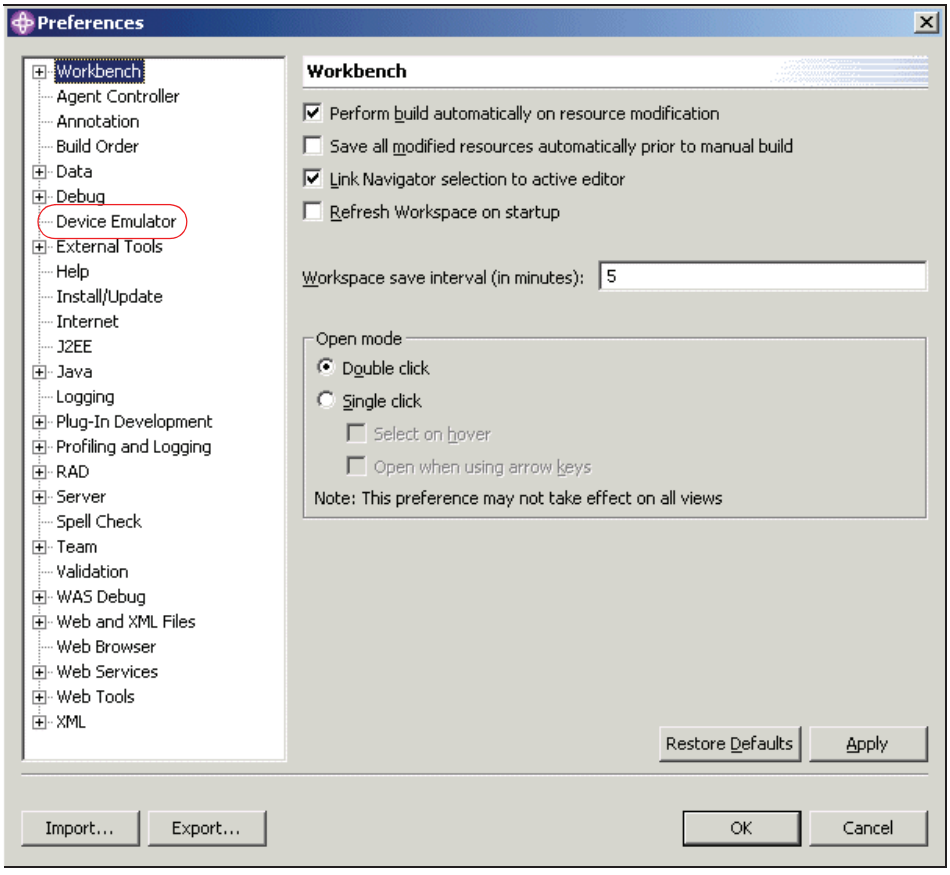


Figure 6-24 Preferences window

2. Select **Device Emulator** from the list on the left, then click the **Add** button. When prompted, add the following information:
   - Name: What you want to call this emulator (Note that Openwave is a simulator, but is used in the same way). For this example, we called it “Openwave”.
   - Location: The location of the executable of the emulator, in this case C:\Program Files\Openwave\SDK 6.1\program\http\OSDK61http.exe.
   - Parameters: The command-line arguments used to load the emulator and browse to the running portlet. Emulators vary greatly in their ability to handle command-line arguments, so check with the emulator’s documentation. Site Developer uses `%URL%` to specify a URL for these parameters, in this case:

     ```
     -direct -pho slimjim-mini-lime.pho -sethome %URL%
     ```

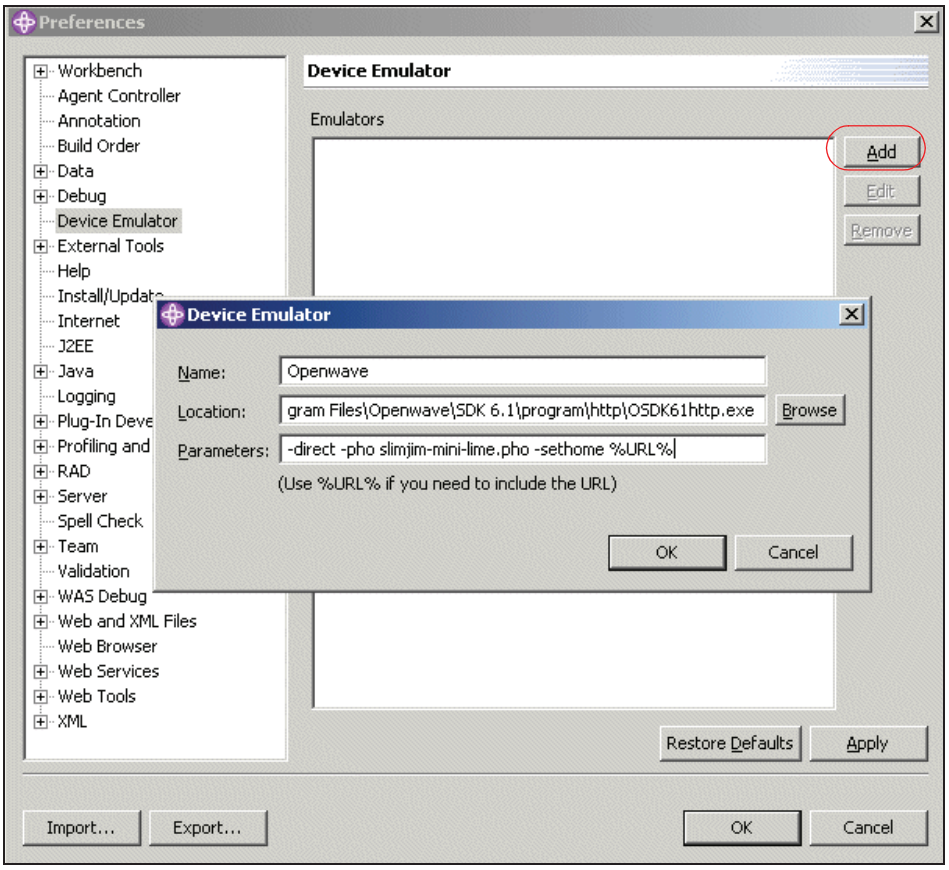


Figure 6-25 Adding Openwave

3. Click **OK,** and then **OK** again to save your changes.
4. For this example, we will use a portlet that only supports the WML markup language.

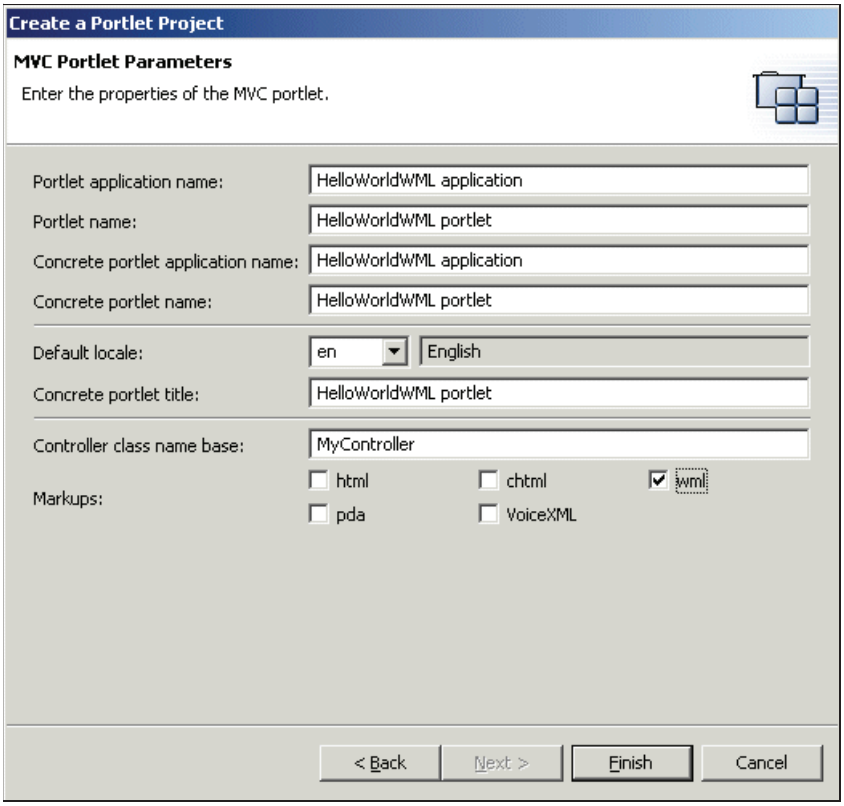


Figure 6-26 WML support only

5. To run it, right-click the project name and click **Run with Device Emulator**.

Important: You must add this project to a test environment before running it with a device emulator. The portlet still needs to be run on WebSphere Application Server to be accessed via the device emulator.

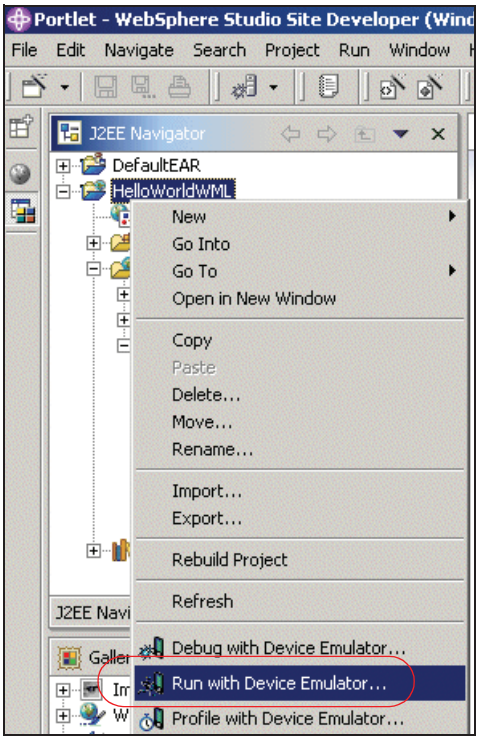


Figure 6-27 Run with Device Emulator

6. The portlet application will load in the device emulator and display its content. On the Openwave simulator, you can click the #1 button on the phone keypad to access the portlet.



Figure 6-28 WML portlet

7. Just to see what happens, right-click your project and click **Run on Server.** Since the portlet does not support HTML, you will receive the error message "`There is no content available. Check if there is content defined for the markup of your client device`" when you try to access it via a Web browser.

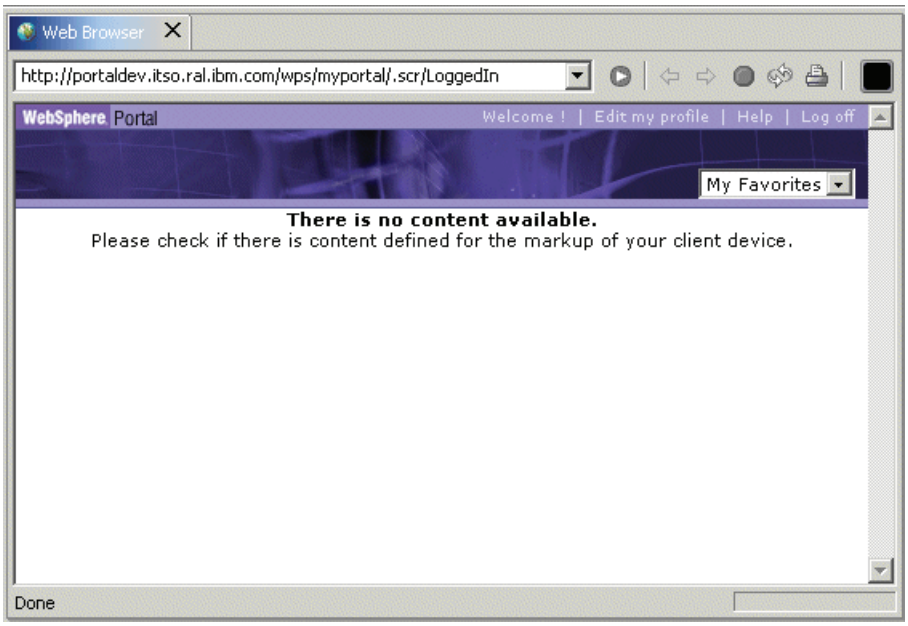


Figure 6-29 WML error

An alternate way to access your portlet from an emulator

When Site Developer starts WebSphere Application Server during the portlet testing process, your portlet is actually accessible to any application that can access your machine.

WebSphere Studio actually loads an instance of the WebSphere Application Server to serve the portlet content. You can load the portlet applications in an external Web browser (as opposed to the integrated browser), external simulators and emulators not integrated into the workbench, and mobile devices that can access the test machine through a cabled or wireless connection.

For this example, we will use the Palm OS 5.2 Simulator.

Note: The Palm Simulator must be configured for Web access. To do this, install a Web browser onto the Simulator and select **Redirect NetLib calls to Host TCP/IP**. For details, see Appendix C, “Palm Simulator installation” on page 1177.

See also the Palm Simulator documentation for more information or visit the Palm developer Web site at `http://www.palm.com/us/developers/`

To begin, we will follow the exact steps as detailed in 6.5.1, “Setting up a test environment and running a portlet” on page 155 to set up the test environment. Run your portlet just as shown by right-clicking the project name and clicking **Run on Server**.

1. Once the portlet is up and running, start the Palm Simulator.
2. If the test environment and the simulation software are both located on the same machine, load the Web browser on the Palm Simulator and navigate to `http://localhost/wps/myportal`

Important: If you are not testing your portlets on the same machine as they are being developed, substitute the name of the test server for `localhost` in the above address.

Similarly, whenever you encounter `localhost` in any URL in this book, substitute your server’s name as needed.

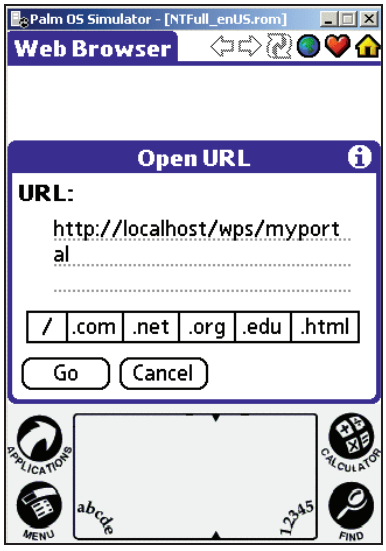


Figure 6-30 Portlet testing URL

Alternatively, you can enter a URL to access a portlet application on a remote test server. It will run just the same as a local portlet application would.



Figure 6-31 A remote Portal application

3. You will need to log in to view the portlet content. The default user name and password are `wpsadmin`.

Important: If your instance of WebSphere Portal Server was configured with a different user name and password, use it instead.



Figure 6-32 Log in icon

4. Once logged in, you will be able to view your portlet.

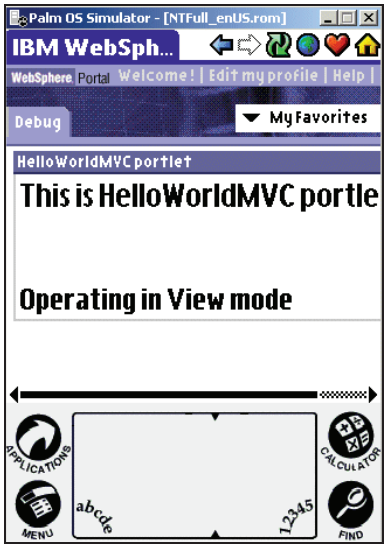


Figure 6-33 Test portlet

Accessing the test portlets from a mobile device

If you want to access your test portlets from the actual device you will be using to view them, you must be able to connect to the machine serving the portlets from the mobile device. For example, to connect a Pocket PC-powered PDA to the test machine, it is probably easiest to use the PDA's cradle and Microsoft ActiveSync software to achieve Internet connectivity from the mobile device.

Once the device has connectivity, navigate to the URL of the test environment as detailed above. You will then have to log in to view your portlets.

See Chapter 4, "Everyplace Client" on page 81 for more information on connecting mobile devices to your computer.



Figure 6-34 Pocket PC

6.5.3 Debugging with the test environment

The process of debugging a portlet is very similar to running a portlet. You can control and trace the execution of the portlet by setting breakpoints in the code.

There is a special view for debugging that allows you to view running threads, variable values, breakpoints, an outline of the project, and the server log/tasks area simultaneously to give you a comprehensive view of how your application is running and if anything is not functioning correctly. You can switch to the Debug

perspective by clicking **Window -> Open Perspective -> Other,** then selecting **Debug**.

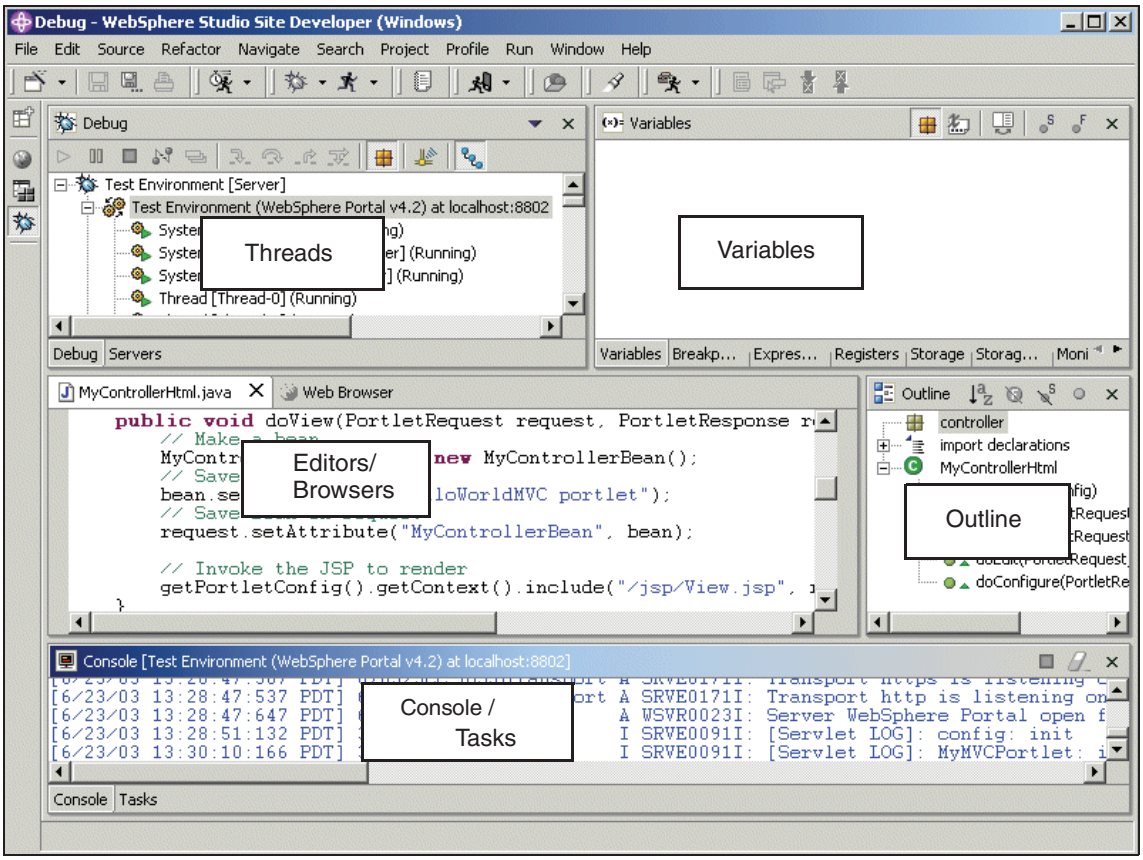


Figure 6-35 Debugging perspective

To debug a project, right-click the project name in the Navigator pane and select **Debug on Server**.

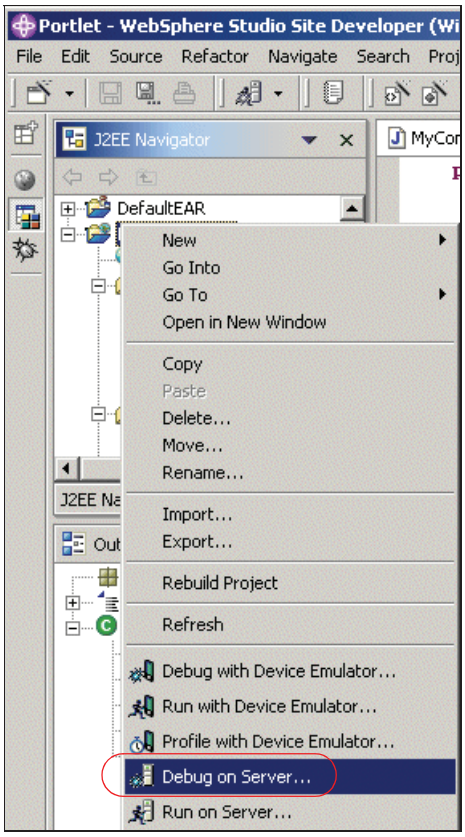


Figure 6-36 Debugging on Server

Debugging

WebSphere Studio makes debugging and finding errors in code very easy with several built-in features.

Breakpoints

To add a breakpoint in a Java file, right-click in the gray column to the left of the code, then click **Add Breakpoint**. A breakpoint will halt execution of your portlet application at that line during execution.

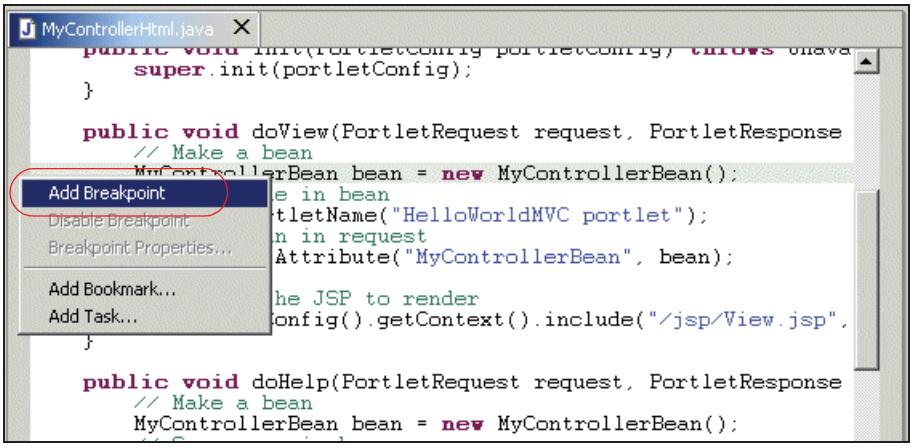


Figure 6-37 Adding a breakpoint

When a breakpoint is set, you will see a small mark to the left of the line where code execution will stop. To remove a breakpoint, right-click it and choose **Remove Breakpoint**.

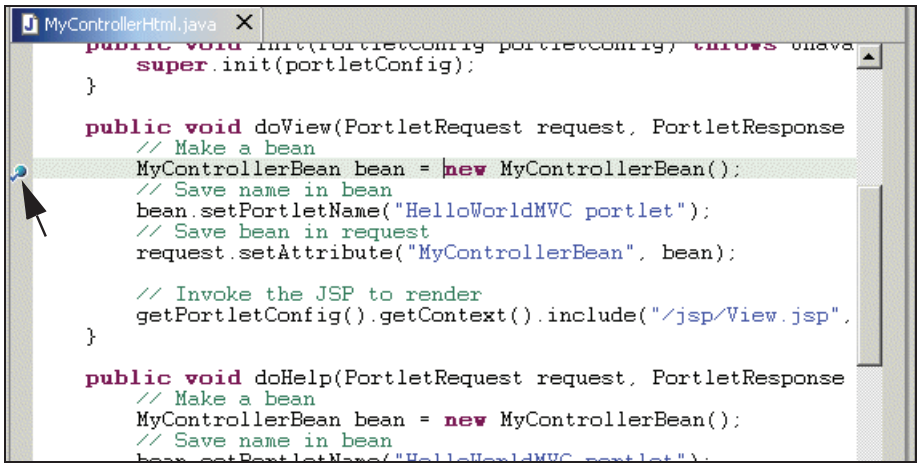


Figure 6-38 Added breakpoint

When a breakpoint is detected as your code is being run, execution halts, and the Debug perspective shows you the current environment status including all running threads, variable values, where execution of the code stopped, and the server log. To change a variable's value, just double-click the variable in the upper-right pane to edit it.

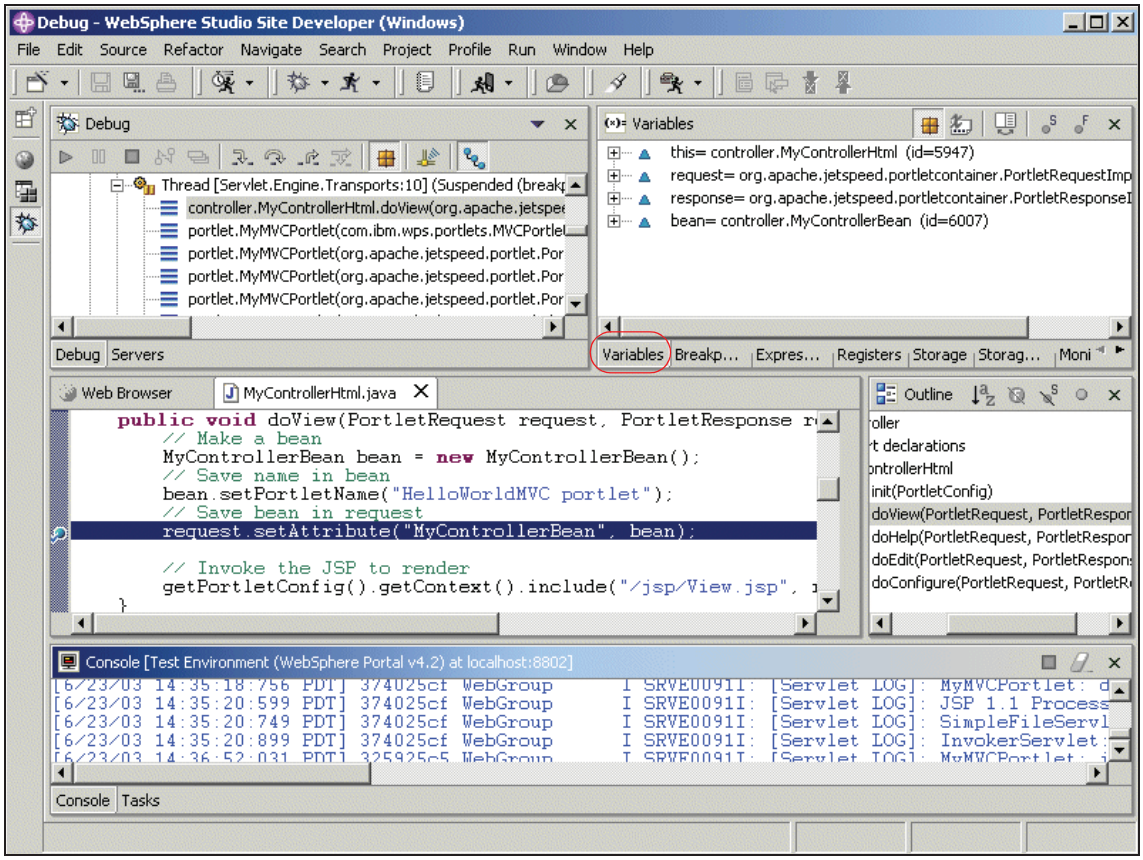


Figure 6-39 Breakpoint detected

Tasks area

The tasks area will list in detail all of the syntactical errors in the code. This feature is mainly used while the portlet application is being coded, rather than as the program is being executed. If there is an error in the code, a red marker to the left of the line of code will indicate this error, as shown in Figure 6-40 on page 176. The tasks area will indicate which line the error is on and give a full description of the error.

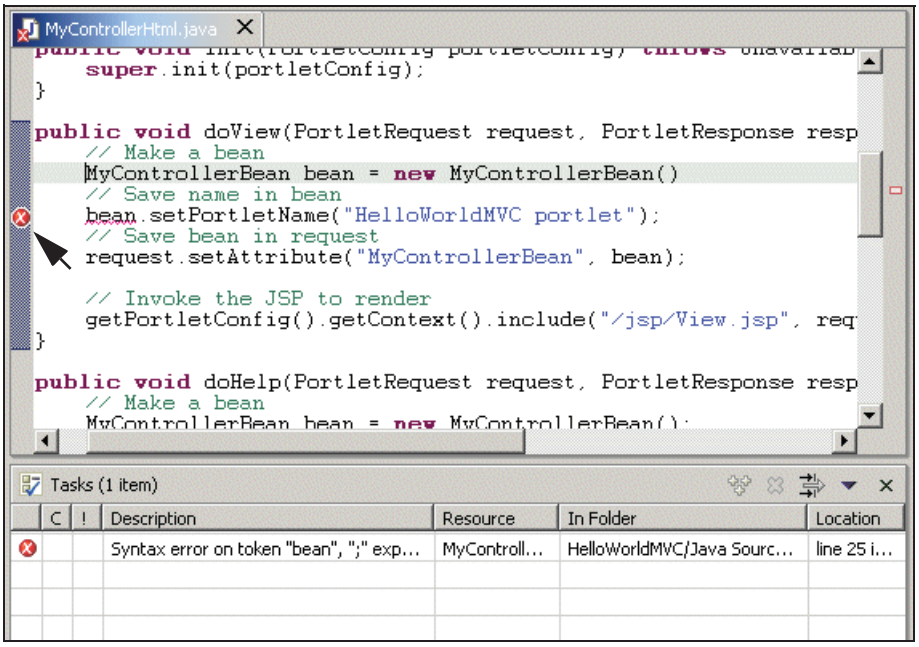


Figure 6-40 Tasks area

The embedded Java development environment also has inline help features that will display as small yellow light bulbs to the left of the lines of code. Double-clicking this marker will bring up a window offering suggestions on how to fix the code.

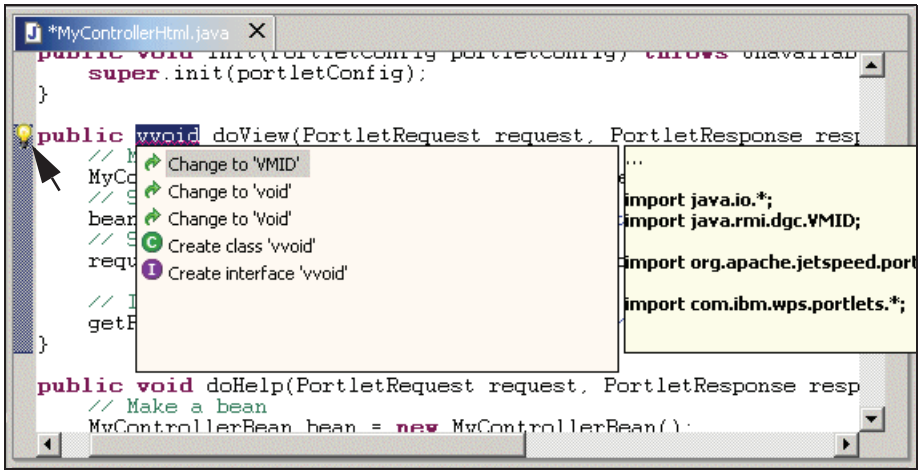


Figure 6-41 Fix code marker

Restrictions on debugging portlets

There are some restrictions on debugging portlets:

- Only one client at a time can use the WebSphere Portal for debugging.
- If you change the portlet properties such as supported markups or modes during the debugging session, you may have to restart the server and republish the portlet project.

7

WebSphere Studio Site Developer and Everyplace Toolkit

This chapter gives an overview of the Everyplace Toolkit and how it integrates with the WebSphere Studio Site Developer environment to develop portlet applications.

It also describes other tools of interest that are used in the development of portlet applications for WebSphere Everyplace Access and how to get help.

This chapter covers:

- WebSphere Studio Site Developer
- The Everyplace Toolkit
- Multiple Device Authoring Technology (MDAT)
- Other studio tools
- The Site Developer workbench interface
- Getting help

© Copyright IBM Corp. 2003. All rights reserved.

7.1 WebSphere Studio Site Developer V5.0

IBM WebSphere Everyplace Access V4.3 includes the WebSphere Studio Site Developer 5.0 development environment. Site Developer can be used to make portlets, which are the building blocks of Portal applications.

7.1.1 WebSphere Studio Site Developer

WebSphere Studio Site Developer is one tool of the WebSphere Studio family of IBM products used to develop Java, Web, and Web services applications. It is included with WebSphere Everyplace Access to develop mobile portlet applications.

It includes built-in Java and JavaScript debuggers and a range of built-in environments to test and debug portlet projects. It includes support for IBM WebSphere V4.0, WebSphere V5.0, WebSphere Express V5.0, and the open source Apache Tomcat.

With Site Developer, a user can develop Web applications that make use of:

- JavaServer Pages (JSPs): A way to create dynamic Web content. Site Developer allows Web developers to create and deploy Web applications that are server and platform independent.
- Servlets: Server applications that execute within a Web application to extend Web server functionality and access existing business systems. Servlets have access to the entire set of Java APIs.
- Web services: Self-contained, reusable, modular applications that are developed to allow business applications to communicate over the Internet.
- Struts: A set of Java classes and JSP tag libraries that provide a conceptual framework for developing Web applications.

WebSphere Studio Site Developer is built on top of an open platform for tool integration known as Eclipse. The whole Eclipse workbench is built on open standards, and the code that Eclipse-based products generate also complies with open standards. As such, Site Developer seamlessly interacts with many third-party software products.

Form more information on Eclipse, visit:

`http://www.eclipse.org/`

Instructions on how to install WebSphere Studio Site Developer and related development software are located in Appendix G, “Portlet development platform installation” on page 1211.

New features in WebSphere Studio Site Developer V5.0

The following new features have been included with WebSphere Studio Site Developer V5.0:

- XHTML Editing for JSPs and static HTML Web pages
- Java code editor with syntax highlighting and inline assistance
- JavaDoc generation
- Device-sensitive editor screens that adjust in size to device types
- Test environments for WebSphere Application Server V4.0 and 5.0, and WebSphere Application Server Express V5.0
- Relational database schemas, SQL wizard, and SQLQuery Builder to help create and manipulate the data design for your project in terms of relational database schemas
- Support for JDK 1.3, JSP 1.2, and Servlet 2.3
- Support for Universal Description Discovery and Integration (UDDI) V2 and discovery via Web Service Inspection Language (WSIL)

A complete listing of WebSphere Studio Site Developer V5.0 features and benefits can be found online at:

`http://www-3.ibm.com/software/awdtools/studiositedev/about/`

7.1.2 Site Developer and Application Developer

As previously stated, WebSphere Site Developer is only one product of the WebSphere Studio family of development tools. Site Developer does not include support for Enterprise JavaBeans (EJBs). If you require this support, you must use WebSphere Studio Application Developer or WebSphere Studio Enterprise Developer.

Site Developer is an easy-to-use tool that minimizes the time and effort required to create, manage, and debug multi-platform Web applications.

Application Developer includes all of the features of Site Developer, but in addition contains plug-in tools for J2EE developers.

If you are new to portlet development or have not used WebSphere Studio development tools before, you may want to refer to the redbook *WebSphere Studio Application Developer Programming Guide,* SG24-6585 for detailed information on the entire tool.

Although the redbook focuses on development using WebSphere Studio Application Developer, there are many similarities between Application Developer and Site Developer that you may want to learn about.

7.1.3 Everyplace Toolkit

The Everyplace Toolkit is a set of plug-ins and extensions for all products of the WebSphere Studio family of development tools that provides an easy way for developers to create and work with:

- Portlet projects, including basic portlets, JSP portlets, Multi-Device/View portlets, Server Invoker portlets, and XSL portlets
- Server configuration options for local testing and debugging within the integrated development environment (IDE) or for debugging on a remote WebSphere Portal Server

  Local portlet debugging support provides the ability to test the application using the same development workstation, thereby eliminating the need for a second machine to handle the debugging process
- Portlet application samples to help you understand how portlets function
- Templates to allow developers to quickly and easily create their own mobile portlets and applications

The Everyplace Toolkit for WebSphere Studio extends the development environment to enable developers to design, develop, and implement portlet applications on mobile devices.

Instructions on how to install WebSphere Studio Site Developer and related development software are located in Appendix G, “Portlet development platform installation” on page 1211.

You can also download the Everyplace Toolkit at no cost from IBM at:

`http://www-3.ibm.com/software/pervasive/products/mobile_apps/everyplace_toolkit.shtml`

New features in Everyplace Toolkit V4.3

- New debugging and test environments
  - Remote Attach: Useful for debugging PDA markup portlets and offline forms on a complex WebSphere Everyplace Access installation
  - Remote Single Server: Useful for debugging a portlet on either a local (outside of the development environment) or remote installation of WebSphere Portal Server, especially when multiple configurations or versions are being used

– Local debug: Useful for debugging simple portlet applications within the WebSphere Studio development environment

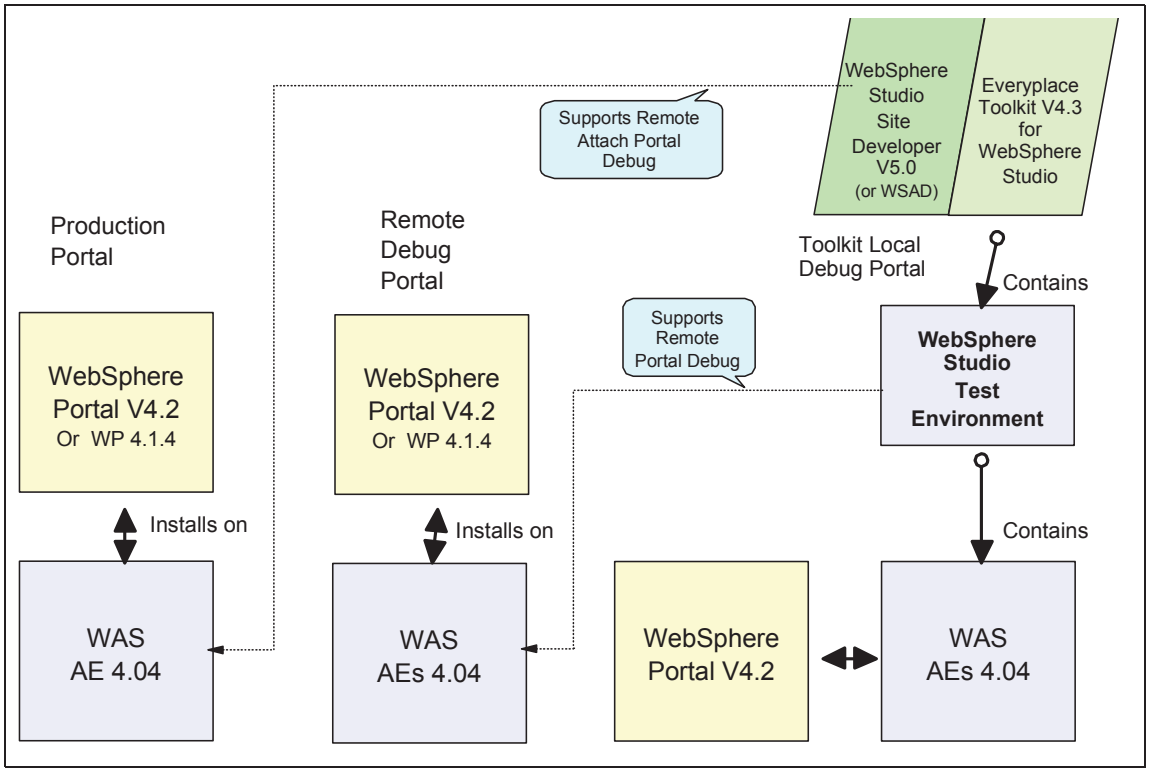


Figure 7-1 Test environments

- Device emulator selection for running or debugging portlet applications
- New offline forms example application
- Support for WML 1.3 and VoiceXML

7.1.4 Everyplace Toolkit and the Portal Toolkit

Portal Toolkit is designed to help you develop portlet applications for WebSphere Portal V4.1 and later. It provides the Portlet Application wizard for creating portlet applications suitable for deployment.

The Everyplace Toolkit is designed to help you develop portlet applications for mobile devices for WebSphere Everyplace Access V4.3 and above. It also has a wizard for creating applications suitable for deployment.

All versions of these toolkits prior to V4.2.5 are identical. Version 4.3 of the Everyplace Toolkit includes examples of portlet applications for mobile devices.

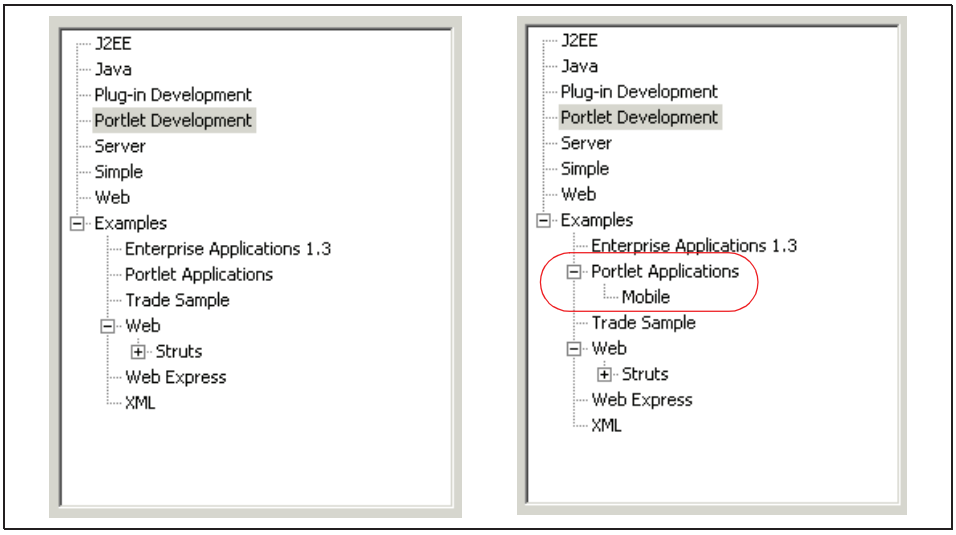


Figure 7-2 New projects window for the Portal Toolkit V4.3 (left) and Everyplace Toolkit V4.3 (right)

7.1.5 Multiple Device Authoring Technology

Multiple Device Authoring Technology (MDAT) is a tool set that allows you to develop portlets for multiple target devices having different screen and Web content handling characteristics. With MDAT, a developer only needs to write a portlet application once in order to generate support for selected mobile devices from a listing of devices that includes all popular PDAs, mobile phones, and other mobile devices. Settings may be custom set or tweaked based on screen resolution and markup language support.

For example, MDAT allows a single portlet to run on a mobile phone (text-based/WML), on a PDA (small screen/HTML), as well as on a desktop computer (full screen/HTML). The content will display correctly on each device used to view it.

The benefits of MDAT include reduced application development costs and the ability to serve portlet applications to a wide variety of mobile devices. It also gives the ability to easily support new devices by simply adding a new device profile to the list.

MDAT can be installed with the Everyplace Toolkit, but in this release it is included only as a technology preview. Future releases of the Everyplace Toolkit (V5.0+) will include MDAT as an integrated technology.

7.1.6 Other tools of interest

WebSphere Studio Site Developer already has several tools built into it to aid with Web and portlet application development.

Java development and debugging tools

WebSphere Studio development products have an integrated Java development environment and built-in Java debugging capabilities that support JDK 1.3.

The Java code editor features full syntax highlighting and inline help features.

Compilation warnings and errors are shown in the tasks area of the WebSphere Studio workbench to allow you to quickly and easily locate and correct them.

Site Developer also includes a new Visual Editor for Java with two-way synchronization that lets you see changes to your graphical user interface (GUI) immediately.

WebSphere Studio also includes support for an integrated roles-based team environment that includes adapters for Concurrent Versioning System (CVS).

Web development tools

The Web development environment in WebSphere Studio includes the tools necessary to develop static Web pages, JSPs, Java servlets, and many other Web resources that support JDK 1.3, JSP 1.2, and Servlet 2.3.

The Web development tools give you the following capabilities:

- JSP and HTML file creation, validation, editing, and debugging
- JavaScript editing and validation
- Custom JSP tags (taglib) support, based on JSP 1.2 specifications
- Image editing and animation
- Cascading Style Sheet (CSS) editing support
- Creation and validation of a Web deployment descriptor (web.xml) file
- HTTP/FTP import
- FTP export to a server
- Web archive (WAR) file import, export, and validation
- Link management
- Automated content wizards
- Integration with the WebSphere test environment

Page Designer

Many of the above capabilities are offered through a tool integrated into the Site Developer environment known as Page Designer. Page Designer includes support for HTML pages, JavaServer Pages, Java servlets, and JavaBeans.

WebArt Designer and AnimatedGif Designer are fully integrated with Page Designer, allowing you to easily place and edit images in your files.

For more information on these tools, refer to the online help.

Web services development tools

Web services are modular, standards-based e-business applications that can be mixed and matched to perform complex transactions with minimal programming.

WebSphere Studio Web services development tools allow you to build Web services that conform to open cross-platform standards, including:

- Universal Description Discovery and Integration (UDDI) for facilitating e-business ventures.
- Simple Object Access Protocol (SOAP) for transporting messages from one business application to another over the Internet.
- Web Services Description Language (WSDL) for describing programs accessible on private networks and over the Internet and the message formats and protocols used to communicate between them.

For more information on these standards, visit:

- For UDDI: `http://www.uddi.org`
- For SOAP: `http://www.w3.org/TR/SOAP/`
- For WSDL: `http://www.w3.org/TR/wsdl`

Information on using UDDI, SOAP, and WSDL in WebSphere Studio projects can be found in the online help.

XML and XSL tools

WebSphere Studio includes a comprehensive XML tool set for building DTDs, XML schemas, and XML and XSL files. It also supports integration of relational data and XML.

XML editor

The XML editor can be used to create, view, and validate XML files. It also provides a wizard to make XML files from scratch or from previously made DTD or XML schema files.

DTD editor
The DTD editor can create, validate, and view DTDs and generate XML schema files, Java beans, and HTML files from them.

XML schema editor
The XML schema editor can create, view, and validate XML schemas and generate DTDs, XML files, relational table definitions, Java beans, HTML documents, and DDLs from an XML schema.

XSL editor
With the XSL editor, you can create, view, and validate XSL files. XSL files are used to describe how to display XML content.

7.2 The WebSphere Studio Site Developer workbench

The WebSphere Studio Site Developer workbench is the graphical user interface used to work with the integrated development environment and all integrated tools within Site Developer.

7.2.1 Starting Site Developer

To start WebSphere Studio Site Developer on Windows, click **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.0.**

The first time you run Site Developer, you will be prompted to specify a workspace, which is the location that Site Developer will use to store the files used in your development projects. By default, your workspace is stored in the \My Documents\IBM\wssd directory.

7.2.2 The workbench user interface

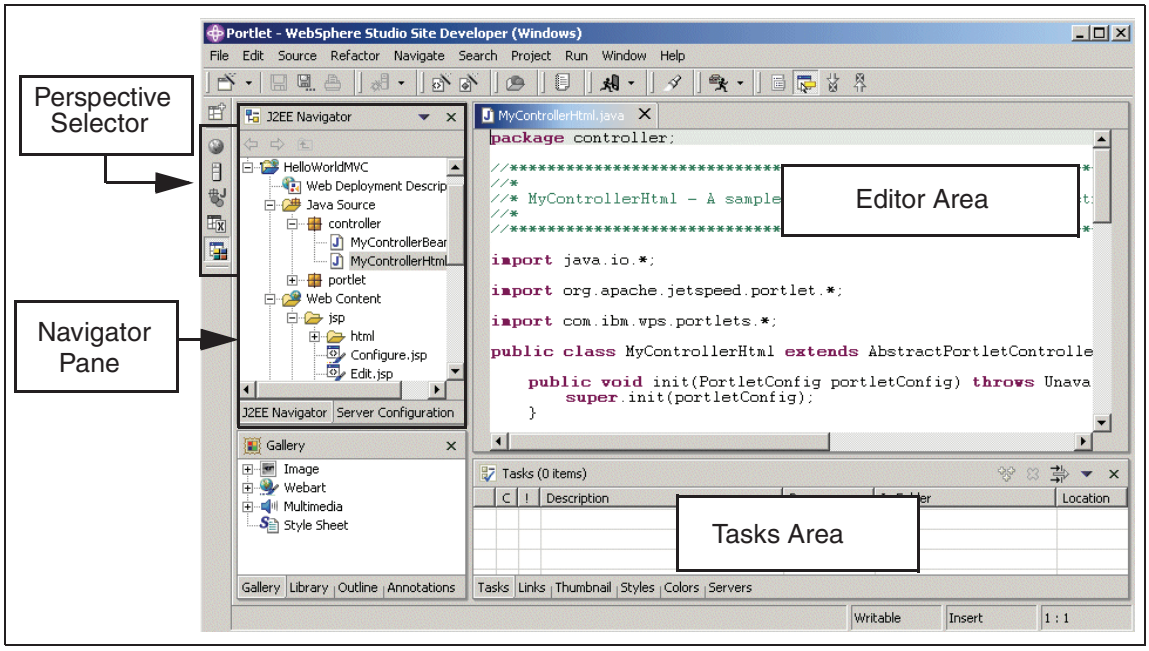


Figure 7-3 The WebSphere Studio Site Developer workbench - Web view

Perspectives, views, and editors

The workbench is the user interface for WebSphere Studio Site Developer. It features an IDE with customizable *perspectives* that provide an easy way for all members of your project team to create, manage, and navigate resources efficiently.

Each perspective is made up of interrelated *views* and *editors*. The views provide different ways to look at the components of your project. The editors allow you to create and modify the components of your Web and portlet applications.

You have the choice of using the predefined perspectives, such as data, debug, J2EE, Java, portlet, server, and Web, or you can customize your own perspective to view your project the way you want to view it. The default perspective shown in Figure 7-3 is the Web perspective.

As you can see, the Web perspective shows the project in such a way that the Navigator pane is shown in the upper-left window. The Navigator pane shows all the files used in the current project as they would be arranged in an actual completed, deployable project.

At the bottom of the Navigator pane, click the **Server Configuration** tab to bring up the current server debugging and testing environments. In Figure 7-4, you can see a local test environment set up to run the HelloWorldMVC portlet application.

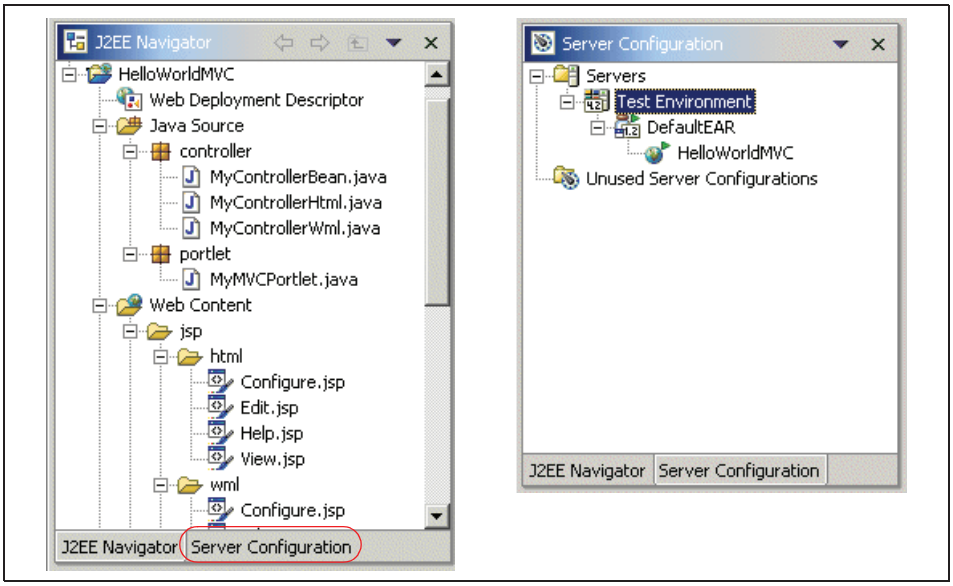


Figure 7-4 The Navigator pane and server configuration pane

The Java source files are located in the \Java Source\ folder. JavaServer Pages (JSPs) are located in the \WebContent\html folder.

To edit a file, you only need to double-click it. WebSphere Studio will automatically open the file in its appropriate editor. Alternatively, you can right-click a file and select which editor to open the file. In a Web perspective, the editor will be in the upper-right pane as shown in Figure 7-3 on page 188.

Directly underneath the editor area, you see the tasks area. The tasks list displays warnings and errors in the Java and JSP code so you can conveniently locate and fix problems with your code.

Notice that many of the individual component panes have many different tabs along their bottoms. For example, the tasks area has several tabs along its bottom.

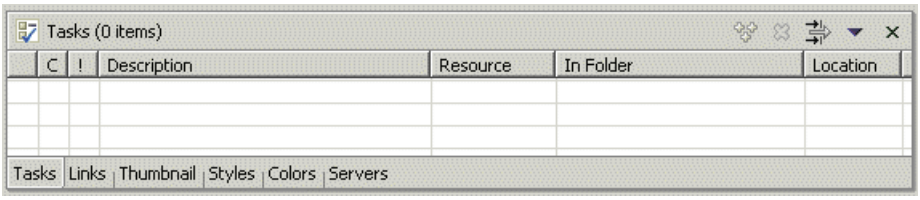


Figure 7-5 The Tasks area

Becoming familiar with these tabs will reduce the time spent looking for the most-used components and increase your productivity.

You can switch perspectives by clicking **Window -> Open Perspective** or choosing from the perspective selector area of the workbench, which lists (in icon form) all of the perspectives you've previously used with your current project.

The perspective selector is located in the upper-left corner of the workbench along the side of the Navigator pane, as seen in Figure 7-3 on page 188.

Tip: If you accidentally modify a perspective and want to reset the layout back to its original state, you can click **Window -> Reset Perspective**.

7.2.3 How to get help

By clicking **Help -> Help Contents**, you can view the online help system. It includes features for browsing and searching help content.

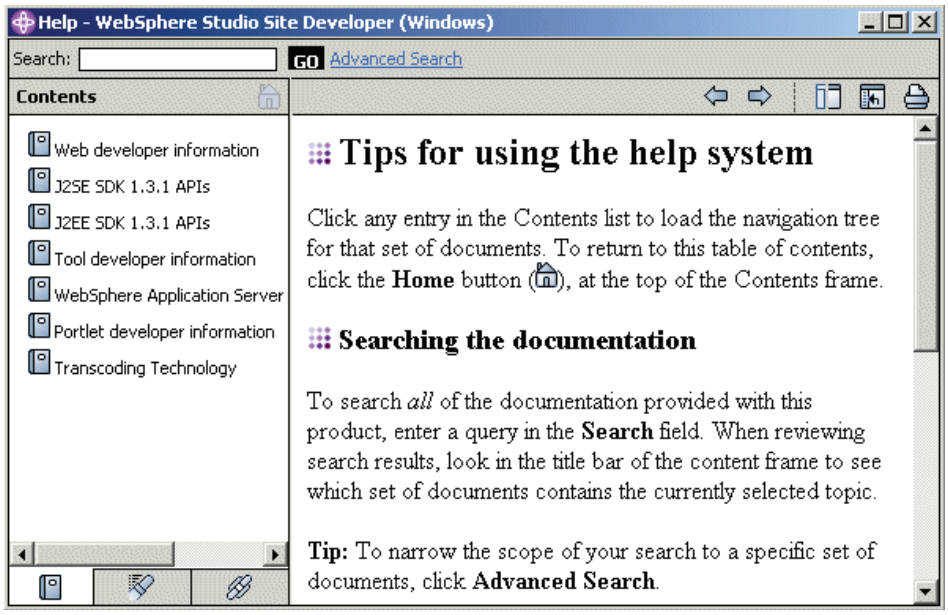


Figure 7-6 Online help

You can also view the WebSphere Everyplace Access InfoCenter, located in the \docs\en\InfoCenter directory of WebSphere Everyplace Access installation CD 1.

You can also visit the WebSphere Everyplace Access Support Web site located at:

`http://www-3.ibm.com/software/pervasive/products/support/ws_everyplace_access.shtml`

This Web site contains the most recent software fixes, hints and tips, product information, white papers, links to other related redbooks, and phone numbers to contact IBM for support.

8

My first portlet applications

This chapter provides a sample scenario for developing two sample portlet projects - one using MVC methodology, the other using a JSP portlet. These activities will allow you to compare the techniques used to develop these two types of portlets.

The development environment used to create and run these portlets is described in Appendix G, “Portlet development platform installation” on page 1211.

In this chapter, we will:

- Create a portlet project with HTML and PDA support using the MVC methodology and run it on both the Palm OS V5.2 Simulator and a Pocket PC.
- Create a portlet project with WML support using a predefined JSP portlet and run it on Openwave SDK V6.1 WAP browser simulator.

© Copyright IBM Corp. 2003. All rights reserved.

8.1 Create a MVC portlet application supporting HTML

We will start by creating a portlet application using the WebSphere Studio Site Developer. This sample portlet is based in the Model-View-Controller (MVC) design pattern. The MVC methodology allows efficient relationships between the user interface and the underlying data model. The main components of this design pattern are:

- Model: The logical structure of data in a software application
- View: All elements in the user interface
- Controller: The elements connecting the Model and View elements

Figure 8-1 illustrates the MVC portlet application. A portlet application can also include a Configure.jsp file that is not shown here.

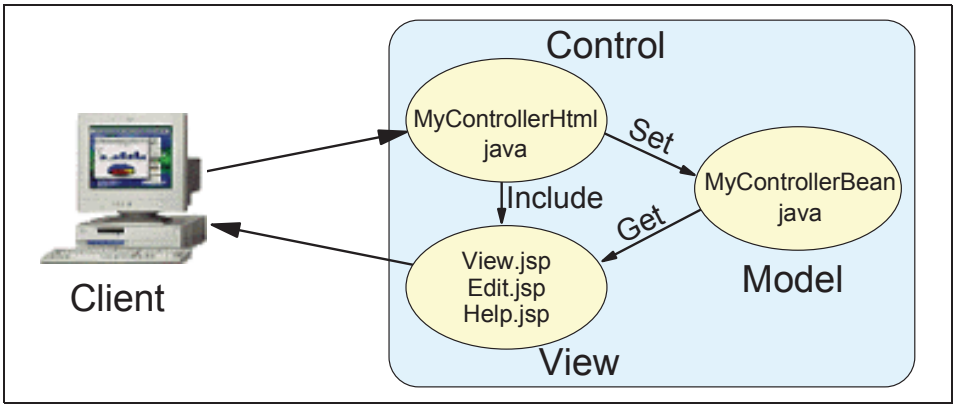


Figure 8-1 Example portlet application using MVC architecture

In this chapter, you will create and test a portlet application. The sample portlet HelloWorldMVC displays a Hello World message and a message indicating the portlet's current mode. Portlet modes allow a portlet to display a different user interface, depending on the task required by the portlet. The Portlet API provides four modes: View, Help, Edit and Configure.

8.1.1 Instructions

Do the following steps to create the portlet application:

1. Open the IBM WebSphere Studio Site Developer by clicking **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.0**.
2. Select **File -> New -> Other**, as shown in Figure 8-2 on page 195.

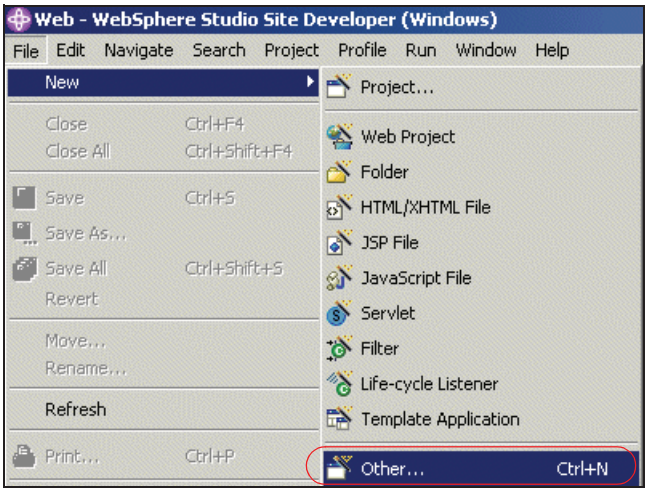


Figure 8-2 Creating a new project

3. In the window shown in Figure 8-3 on page 196, select **Portlet Development** from the left pane, then **Portlet application project** from the right pane. Then click **Next**.

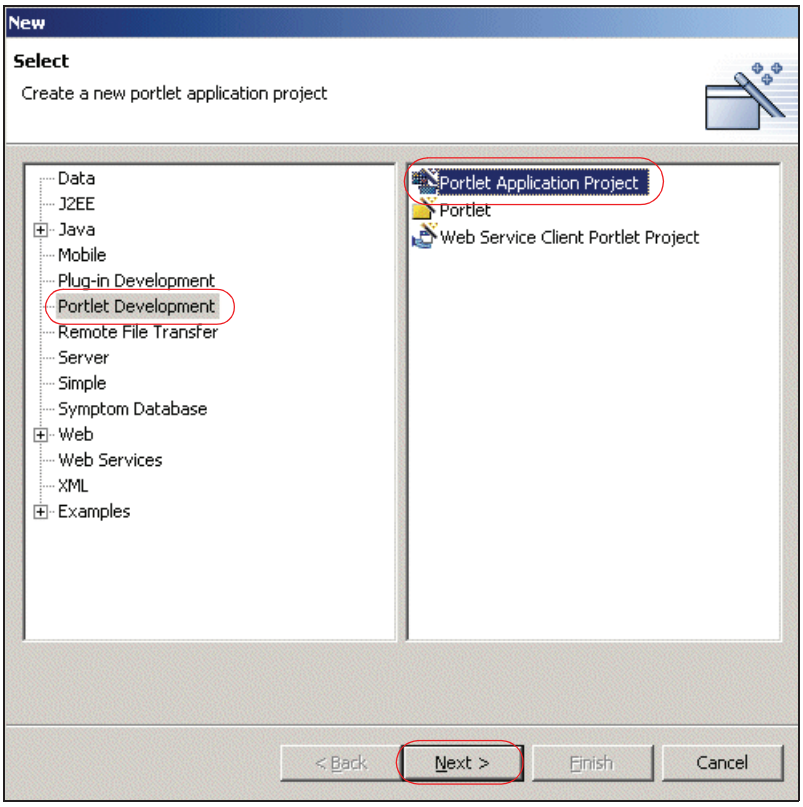


Figure 8-3 Selecting the portlet application project

4. In the Define the Portlet Project window shown in Figure 8-4 on page 197, enter the following information to create your project:
   - Project name: `HelloWorldMVC`
   - Enterprise Application project name: `DefaultEAR` (the default)
   - Context root: `HelloWorldMVC` (the default)

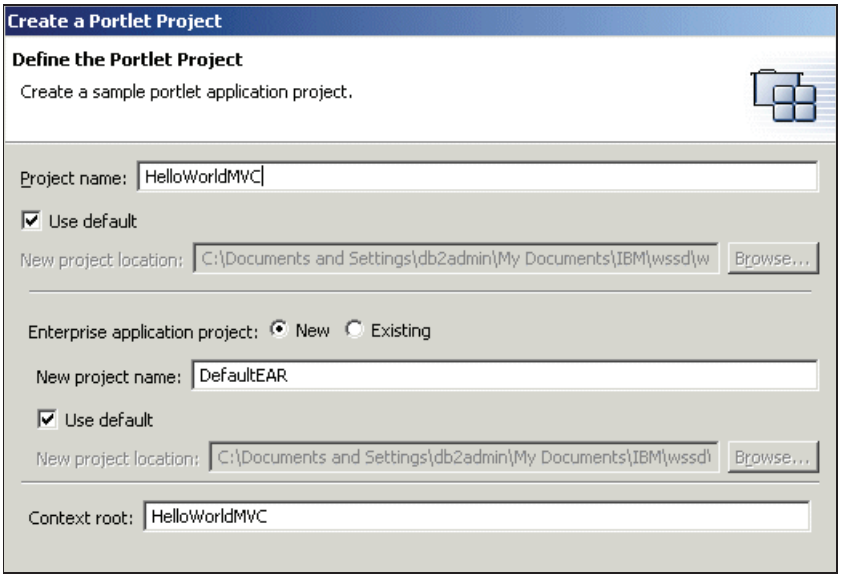


Figure 8-4 Defining the portlet project

5. Click **Next**.
6. In the window in Figure 8-5, select **MVCPortlet**, the type of portlet to be generated. Then click **Next**.

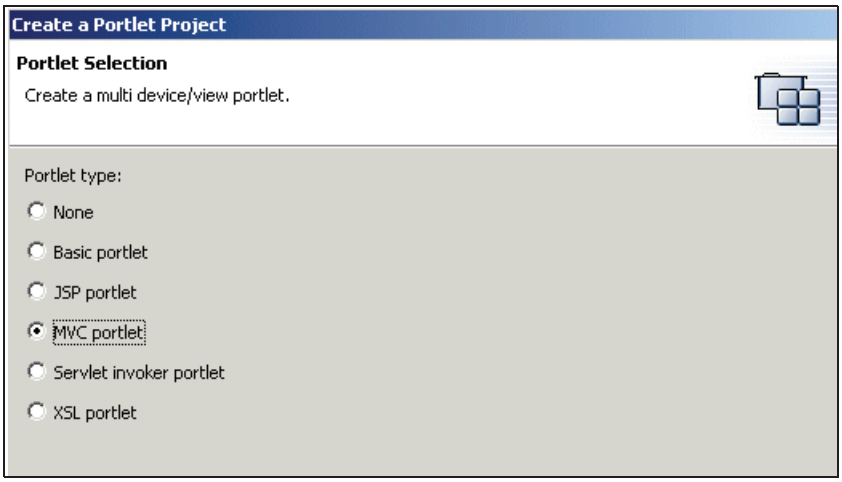


Figure 8-5 Portlet type selection

7. Review and accept the default properties on the MVC Portlet Parameters window (Figure 8-6 on page 198). Click **Finish** to generate your portlet project.

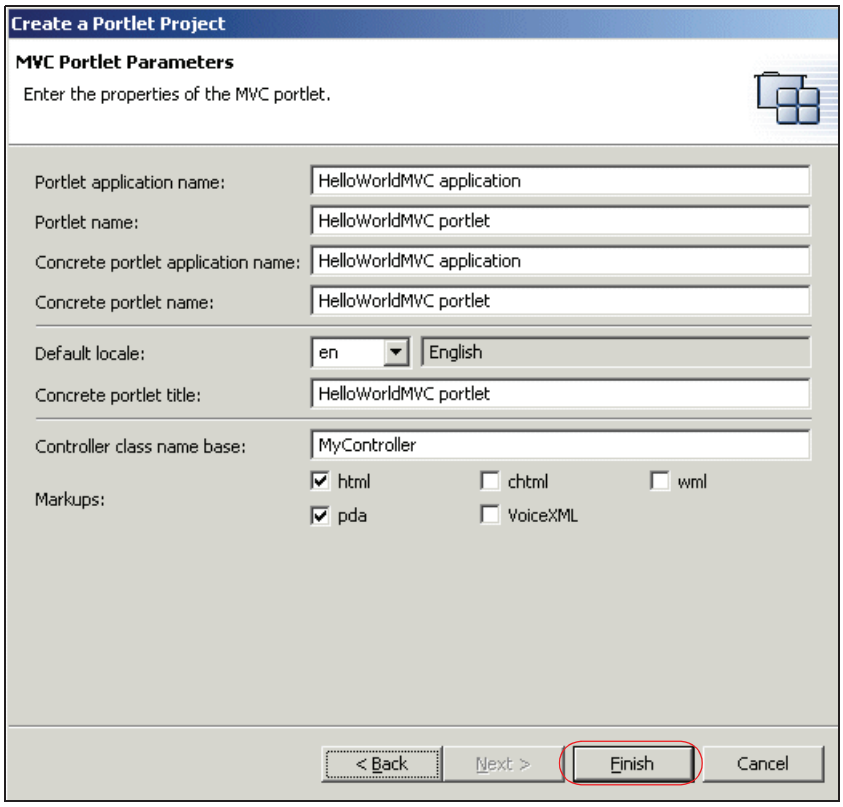


Figure 8-6 MVC portlet parameters

After the project has been configured, you should see two projects listed in the Navigator pane, DefaultEAR and HelloWorldMVC, as shown in Figure 8-7 on page 199.

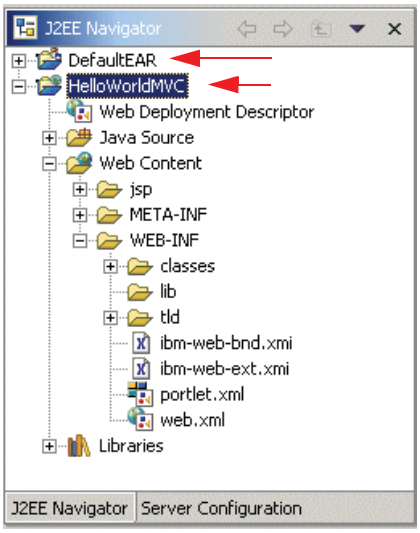


Figure 8-7 Navigator pane

8. If you have already worked on any previous portlet projects, they may still exist in the DefaultEAR project. We suggest that you remove the other portlets from the EAR deployment descriptor before proceeding. You can do this by following these steps:
   a. Open your DefaultEAR / META-INF folder in the navigator pane
   b. Double-click the application.xml file
   c. Select the **Modules** tab at the bottom of the Application deployment descriptor editor
   d. Remove all WAR modules except for your portlet application, HellowWorldMVC
   e. Save any changes by clicking **File -> Save All**
   f. Click **OK** if you receive the Server Configuration window

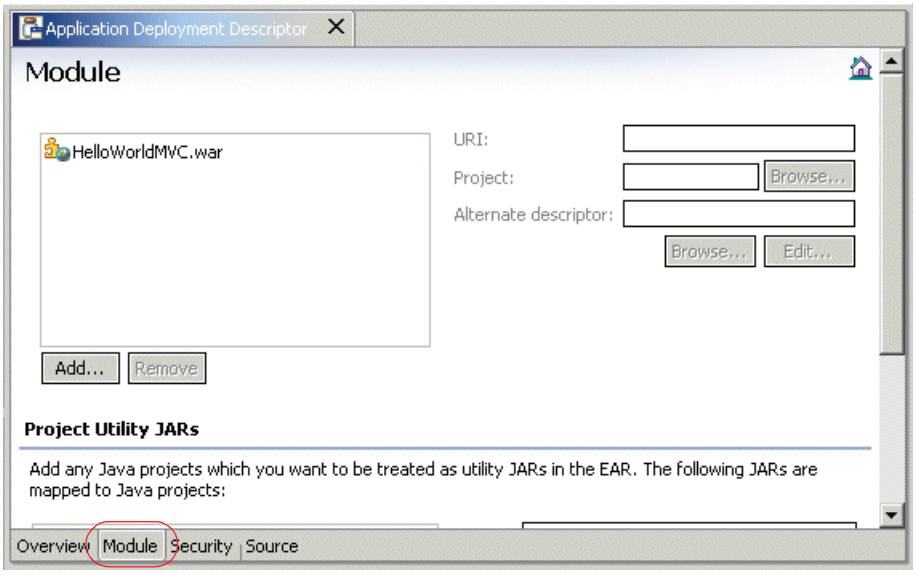


Figure 8-8 Removing WAR modules

9. You will need to add the project to the test environment to execute it. Instructions on configuring a new test environment can be found in Appendix G.8, “Configure WebSphere Studio Site Developer and the Toolkit” on page 1250. Additional information on server configuration can be found in 6.5.1, “Setting up a test environment and running a portlet” on page 155.
10. Add the project to the test environment. On the Navigator pane, select **Server Configuration**.

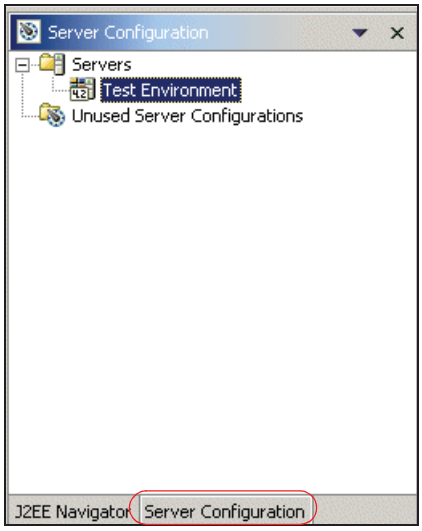


Figure 8-9 Server configuration tab

11. Right-click **Test Environment**, then click **Add -> DefaultEAR**.

Important: You will always need to add a new Enterprise Project (EAR) file to the test environment before it can be run on the server.

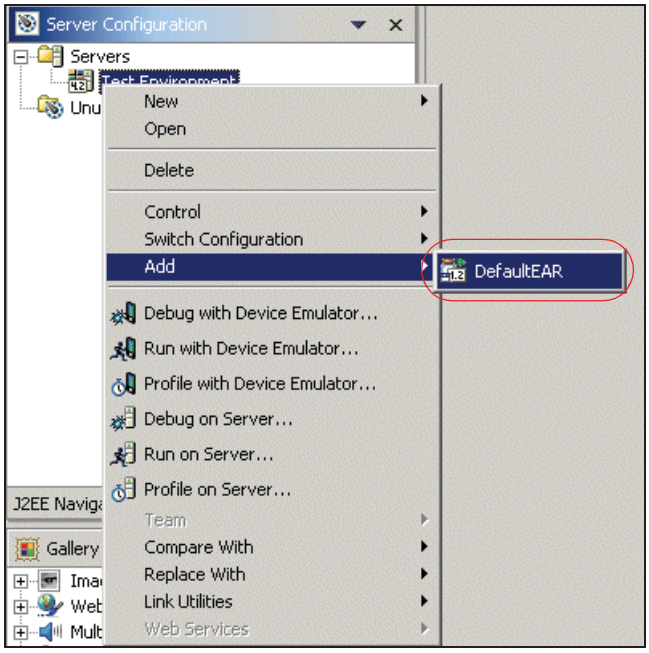


Figure 8-10 Adding a project to the test environment

12. Select the Server status pane by clicking **Servers** in the bottom pane of the window.

Important: If the test environment is running and the server state indicates that the server should be republished or restarted, do so before running your project on the server. If the test environment is *not* running and the server state indicates that the server should be republished or restarted, it is not necessary to do this. The project files will automatically be updated when the project is run.

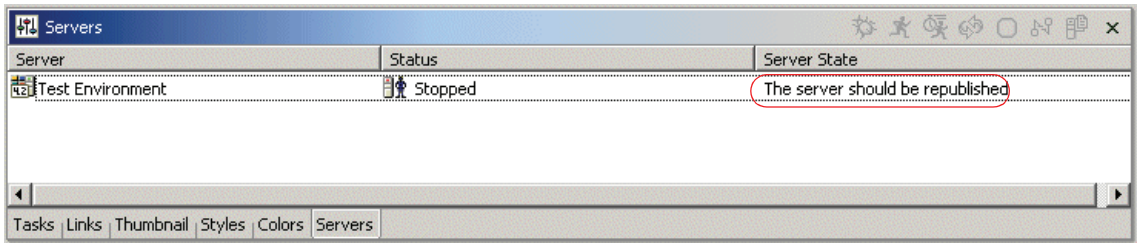


Figure 8-11 Server tab

13. Select and right-click **HelloWorldMVC** in the Navigator pane.
14. Select **Run on server**. The first time it will take a few minutes to start the server.

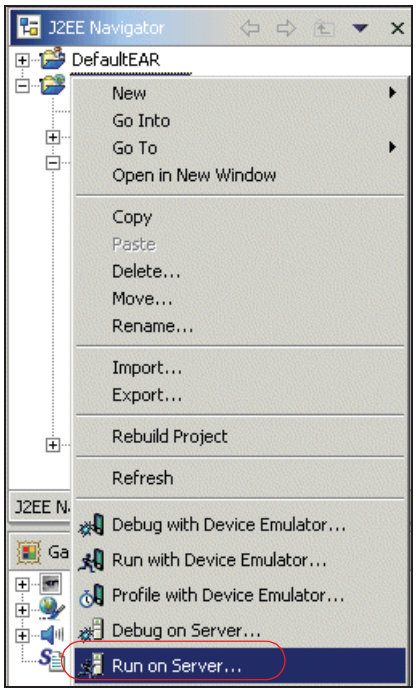


Figure 8-12 Running the Portal project in the test environment

15. Your project will be published, the server will start, and the portlet application will run in View mode. The portlet will display in the integrated Web browser in WebSphere Studio Site Developer. At this time, if you wish to view the portlet with an external device or simulator, follow the instructions in the next section.

Displaying the portlet content in the Palm OS V5.2 Simulator

Note: You must install and configure the Palm Simulator for Internet access before you will be able to view the portlets. See Appendix C, “Palm Simulator installation” on page 1177 for instructions on how to install and configure the Palm Simulator.

1. Open the Palm Simulator and click **Web** to open the Web browser.



Figure 8-13 Accessing the Web browser on the Palm Simulator

2. Click the globe in the upper-right corner of the Simulator to enter the URL to navigate to. If your test environment is located on the same machine as the Palm Simulator, enter `http://localhost/wps/myportal` and click **Go** to continue. If your test environment is located on a remote server, enter `http://your-server-name.your-co.com/wps/myportal` and click **Go** to continue.

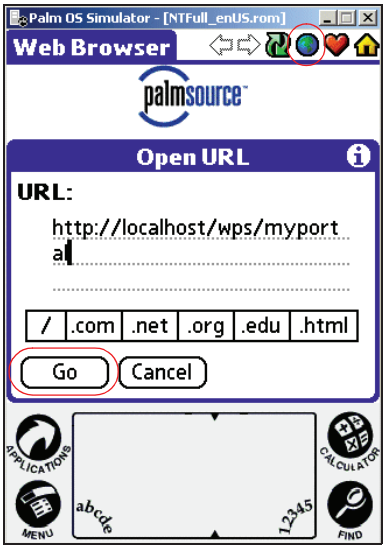


Figure 8-14 Entering a URL

3. Log in using `wpsadmin` as the user name and password.

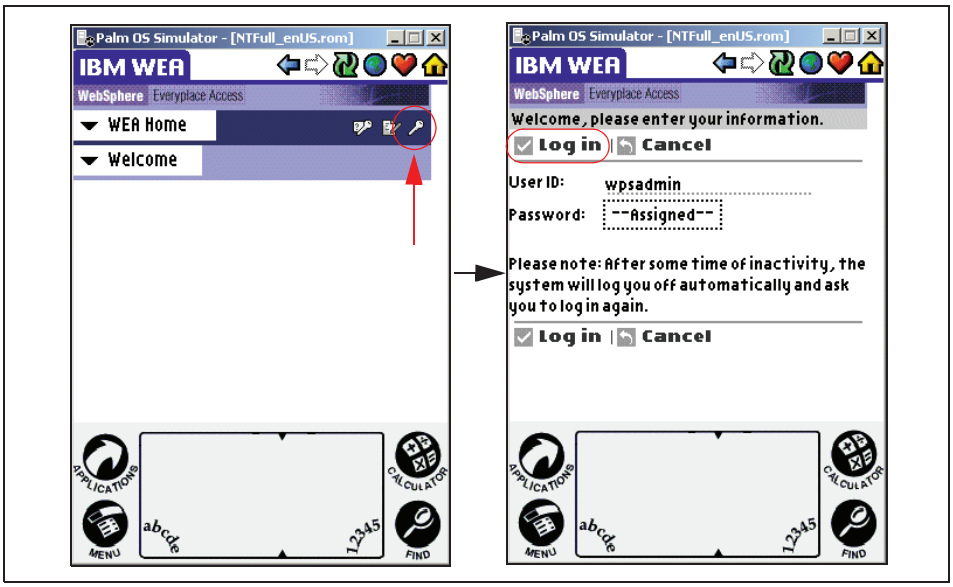


Figure 8-15 Logging in

4. You will now see all the available portlets displayed in icon form. Select the portlet you want to display and click it.

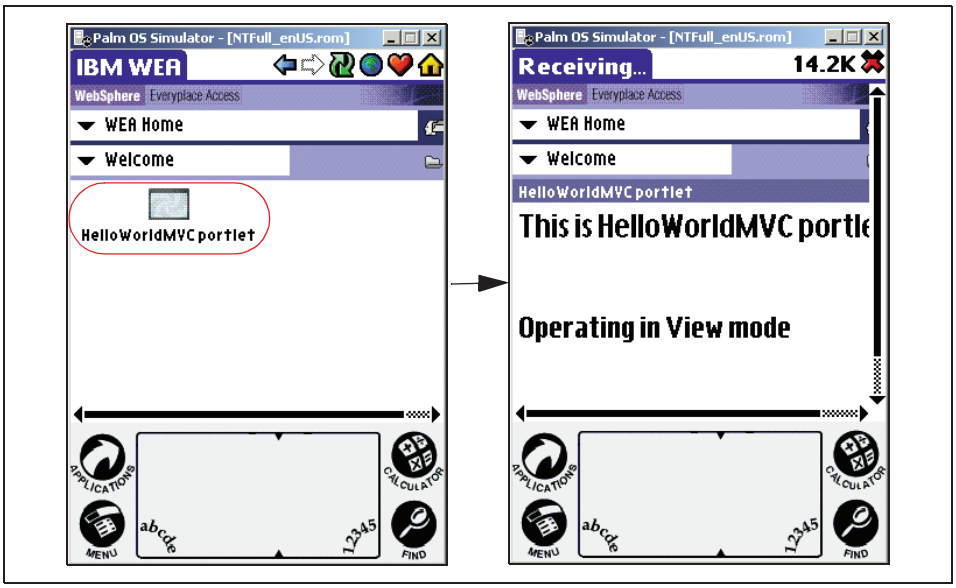


Figure 8-16 Viewing your portlet

Displaying the portlet content on a Pocket PC device.

The easiest way to test your portlets on your Pocket PC device is to connect the PDA to your computer via its cradle using the Microsoft ActiveSync software. See Appendix F, “Pocket PC development tools” on page 1201 for a reference on how to do this.

1. Insert the PDA into its cradle and connect it to the computer via the manufacturer-supplied USB or serial cable.
2. Using the Microsoft ActiveSync software, establish a connection with the host computer.
3. Open a Web browser on the Pocket PC device. Enter `http://your-server-name.your-co.com/wps/myportal` as the destination URL to continue.
4. Just as in the Palm Simulator, you will be prompted to log in. Enter `wpsadmin` as the user name and password.

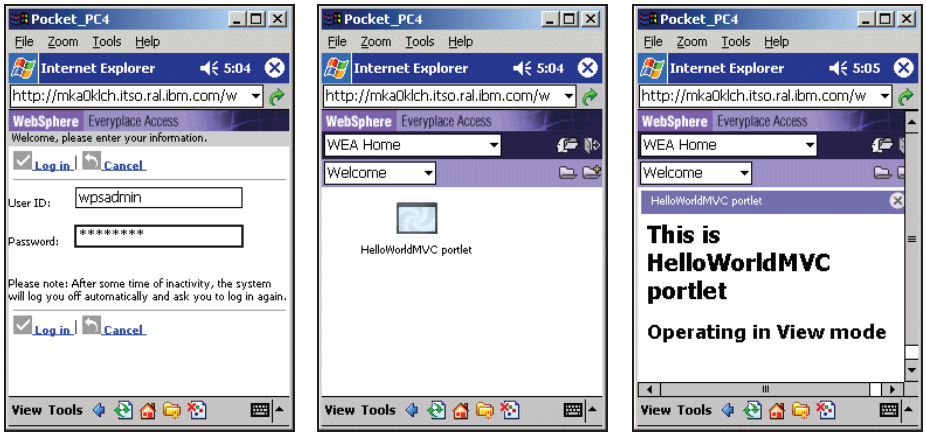


Figure 8-17 Pocket PC login and portlet

5. The portlet will load in the browser on the Pocket PC device.

8.1.2 Examining the HelloWorldMVC project

If you wish to review the generated portlet.xml and web.xml descriptors, follow the steps in this section.

1. In the Navigator pane, double-click the **portlet.xml** file to display the portlet descriptor.

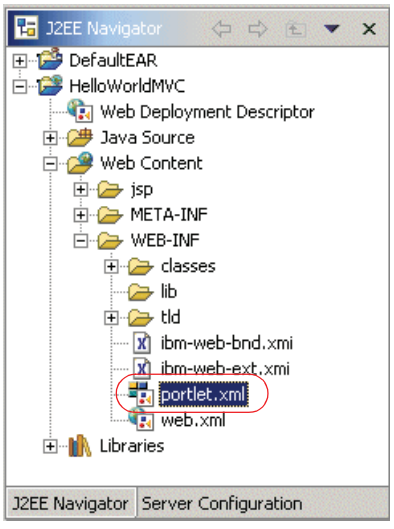


Figure 8-18 Accessing portlet.xml

2. The portlet deployment descriptor editor displays the portlet.xml values.

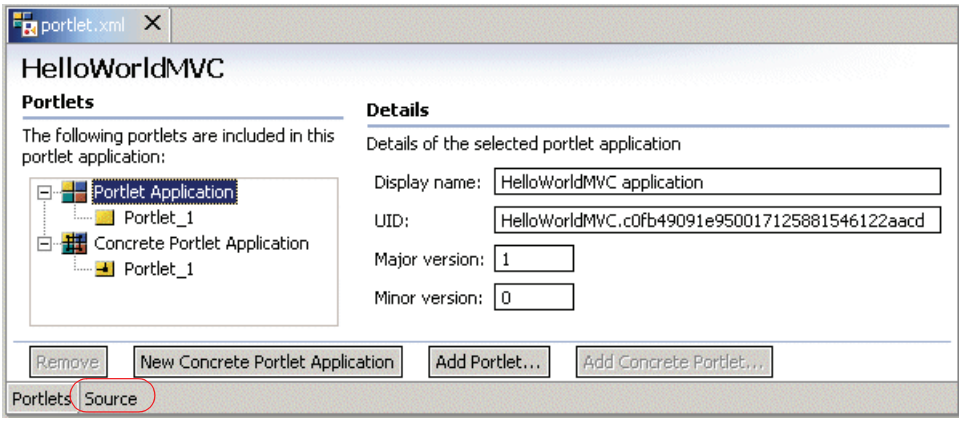


Figure 8-19 Portlet.xml editor

3. Explore the information shown by the portlet deployment descriptor editor to see what values are stored in this file. Also click the **Source** tab along the bottom to display the XML source. It will be similar to what you saw in Example 6-5 on page 151.
4. Double-click the **web.xml** file in the Navigator pane to show its content in the Web deployment descriptor editor.

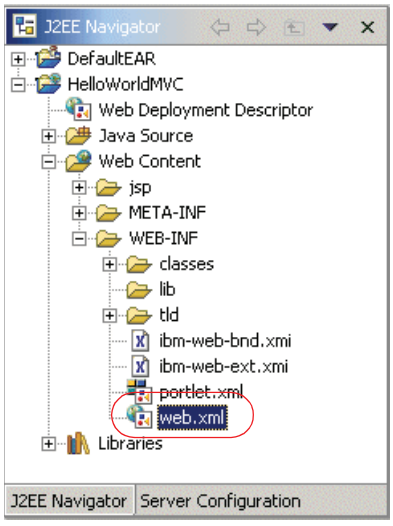


Figure 8-20 Accessing web.xml

5. This editor displays the web.xml values.

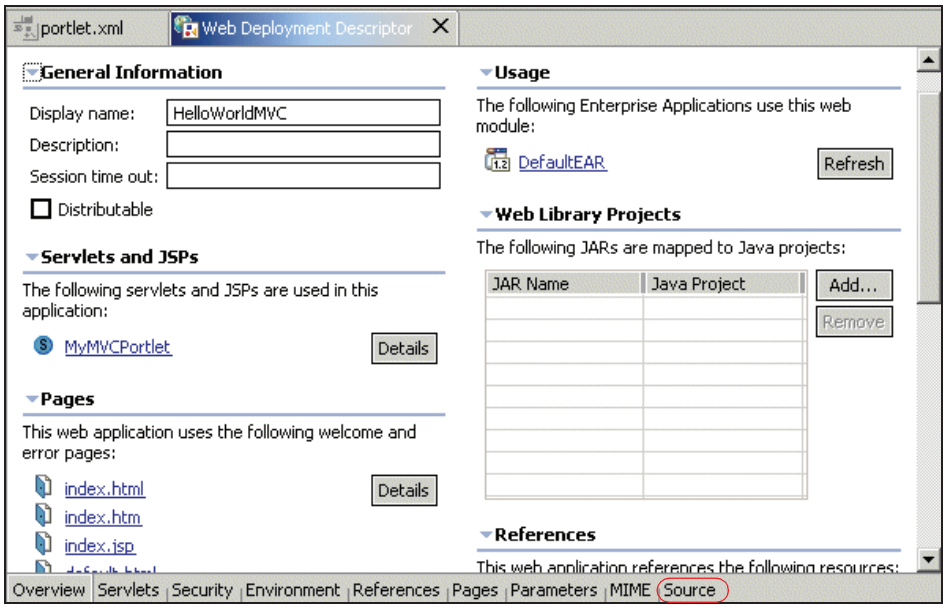


Figure 8-21 Web.xml editor

6. Explore the Web deployment descriptor editor to become familiar with the information stored in this file. Click **Source** to display the XML source file. It will be similar to what you saw in Example 6-6 on page 153.

8.1.3 Changing a portlet application

In this section you will update the portlet application. The following files will be updated:

- MyControllerBean.java
- MyControllerPDA.java
- View.jsp

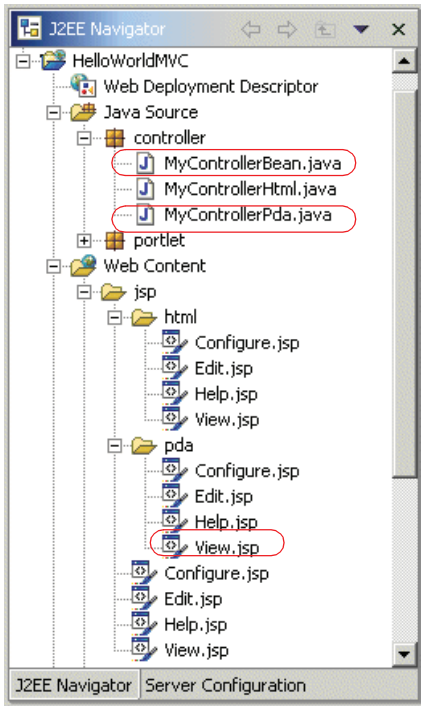


Figure 8-22 Accessing View.jsp

Note: You will be editing the View JSP for the PDA markup (located under the pda directory). The JSPs in this folder allow you to develop content specifically designed for PDA devices.

1. Edit MyControllerBean.java by double-clicking the file name.
2. Update the code shown in Example 8-1with the changes highlighted as follows:
   - Create a new string called myName.
   - Create both a get and a set method to get and set the string myName.

Example 8-1 MyControllerBean.java

```
public class MyControllerBean {

   private String portletName = "";
   private String myName = "";

   public void setPortletName(String s) {
    portletName = s;
```

```
    }

    public void setMyName (String s){
     myName = s;
    }
    public String getMyName (){
     return myName;
    }
    public String getPortletName() {
    return (portletName);
    }
}
```

3. Save your changes by clicking **File -> Save MyControllerBean.java**.
4. Double-click **MyControllerPda.java** to edit the file.
5. Update the doView() method with the highlighted code shown in Example 8-2. The code sets `John Doe` as the value for myName using the set method setMyName.

Example 8-2 doView method

```
public void doView(PortletRequest request, PortletResponse response) throws
PortletException, IOException {
    // Make a bean
    MyControllerBean bean = new MyControllerBean();
    // Save name in bean
       bean.setPortletName("HelloWorldMVC portlet");
    //** Set John Doe as MyName
       bean.setMyName("John Doe");
    // Save bean in request
    request.setAttribute("MyControllerBean", bean);
     // Invoke the JSP to render
        getPortletConfig().getContext().include("/jsp/View.jsp", request,
response);
    }
```

6. To save your changes, use the keyboard shortcut Ctrl+S.
7. Double-click **View.jsp** in the /Web Content/jsp/pda directory to edit the file.

Important: Make sure you select the JSP in the pda directory.

8. Select the **Source** tab and add the highlighted code shown in Example 8-3 on page 212 so the View.jsp will retrieve your name from the bean.

Example 8-3 View.jsp

```
<%@ page contentType="text/html"%>
<jsp:useBean id="MyControllerBean" class="controller.MyControllerBean"
scope="request" />

<h1>This is <%=MyControllerBean.getPortletName()%></h1>
<br>
<h2>Operating in View mode</h2>
<BR>
<H2>Updated by <%=MyControllerBean.getMyName()%></H2>
```

9. Select **File -> Save All** to save your changes to the project.
10. Select **Servers** from the bottom pane of the window and right-click **Test Environment** and click **Publish**. Publishing is necessary to update the content of your portlet application in the test environment.

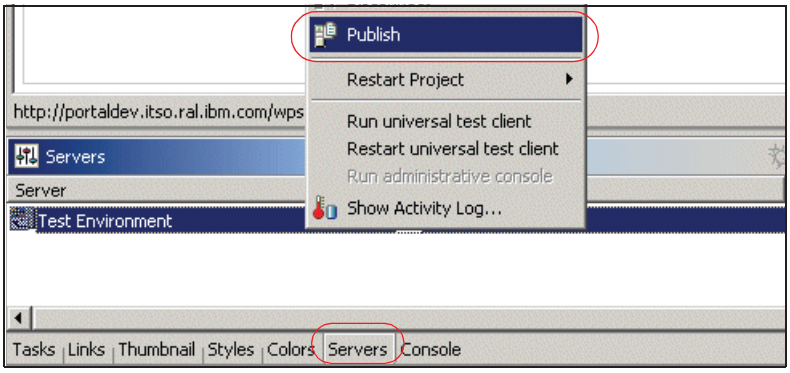


Figure 8-23 Publishing an updated project onto the server

11. When that operation has completed, a status window appears. Click **OK**.

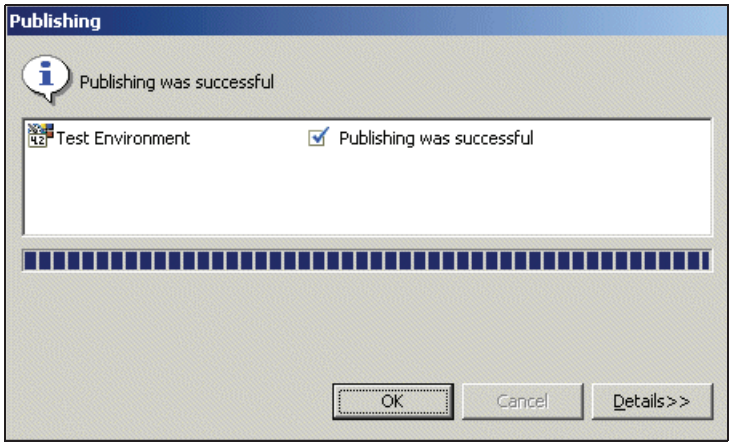


Figure 8-24 Publishing successful window

12. In the Navigator pane, select and right-click **HelloWorldMVC**, then select **Run on Server**.

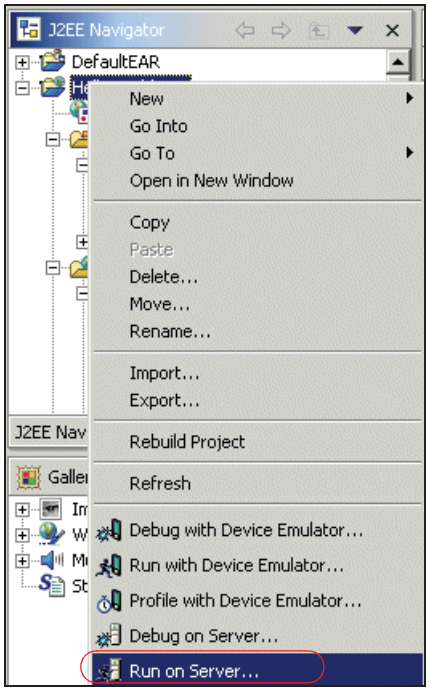


Figure 8-25 Running the project

13. The portlet will show its View mode content in the built-in browser.

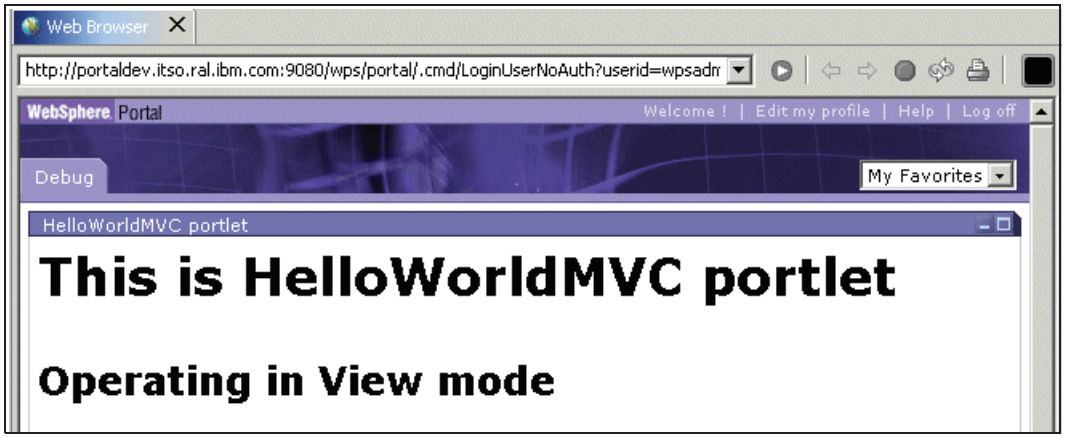


Figure 8-26 The updated portlet in the browser window

14. If you still have the Palm Simulator running from the previous example, just click the **Refresh** button in the upper-right corner of the screen to view the updated portlet on the Palm Simulator. If not, open it and browse to `http://localhost/wps/myportal` and log in to view the updated portlet.

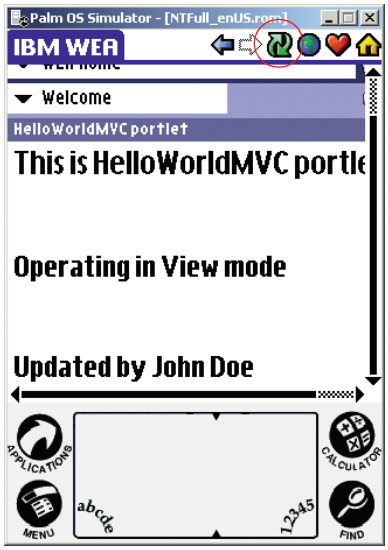


Figure 8-27 Updated portlet on the Palm

8.2 Create a JSP portlet that supports WML

In this section you will create a JSP portlet application using the WebSphere Studio Site Developer. This sample portlet is designed based on a predefined portlet class called JSPPortlet allowing you to display JSP content. The main advantage of using these techniques is that you do not need to write Java code.

You will create HelloWorldJSP, a JSP portlet application that supports only the WML markup language, and test it on the Openwave browser simulator. More information on the Openwave browser simulator and configuring it as a device emulator can be found in 6.5.2, “Testing portlet projects on Pocket PC, Palm, and WAP devices” on page 161.

This exercise assumes that you already have the Openwave simulator configured for the Run with Device Emulator command.

1. If not already running, start the IBM WebSphere Studio Application Developer, click **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.0**.
2. Select **File -> New -> Other**.

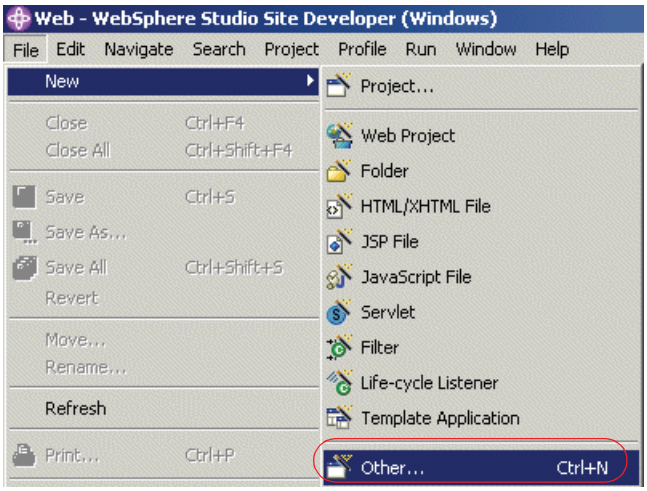


Figure 8-28 Creating a new project

3. Select **Portlet development -> Portlet application project**.

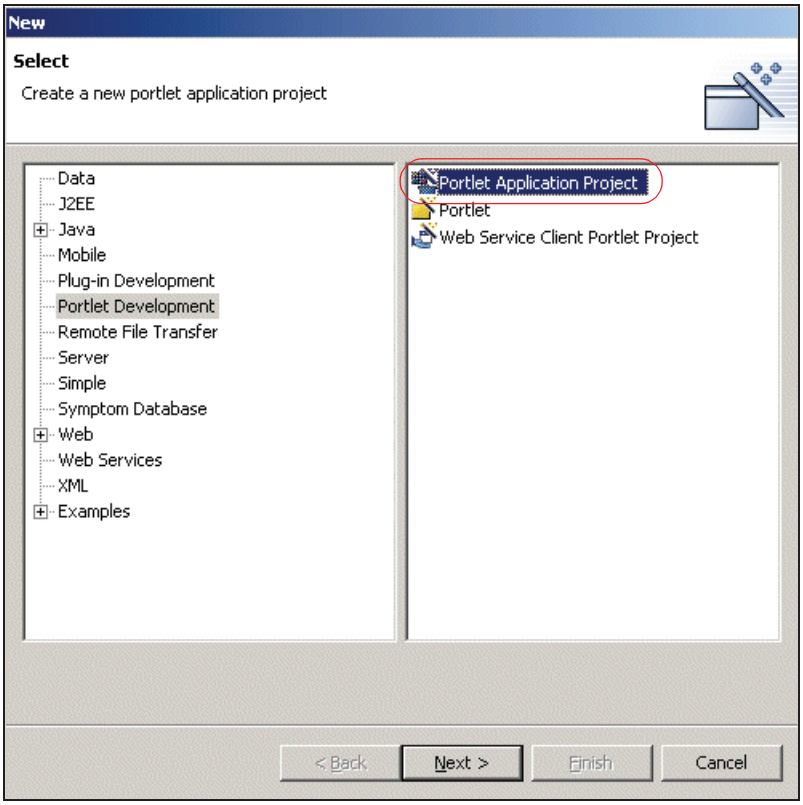


Figure 8-29 Selecting the type of project

4. Click **Next**.
5. In the Define the Portlet project window enter the following information:
   - Project name: `HelloWorldJSP`
   - Enterprise Application project name: `DefaultEAR` (the default)
   - Context root: `HelloWorldJSP` (the default)

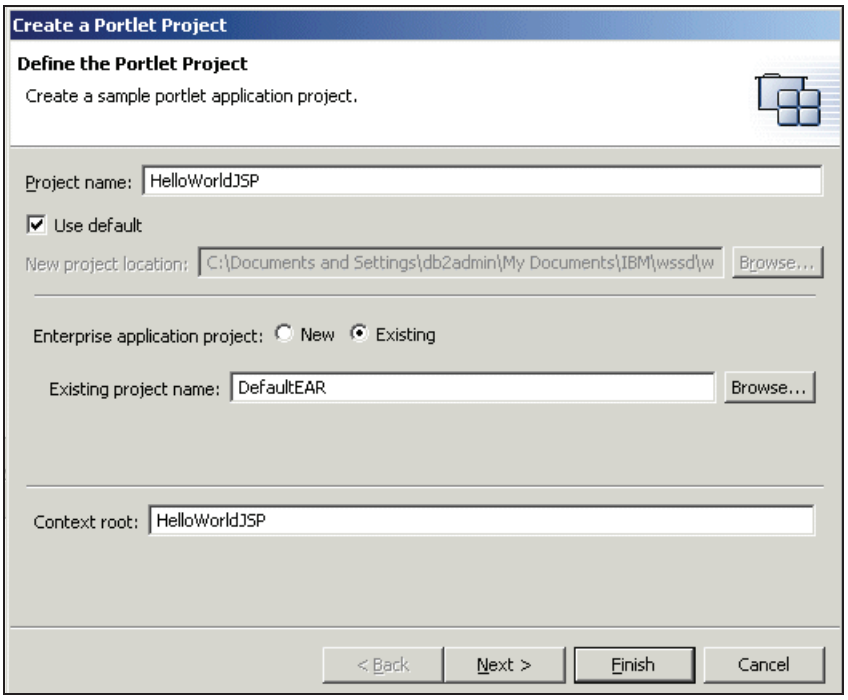


Figure 8-30 Defining the project

6. Click **Next**.
7. Select **JSP Portlet** as the type of portlet to be generated.

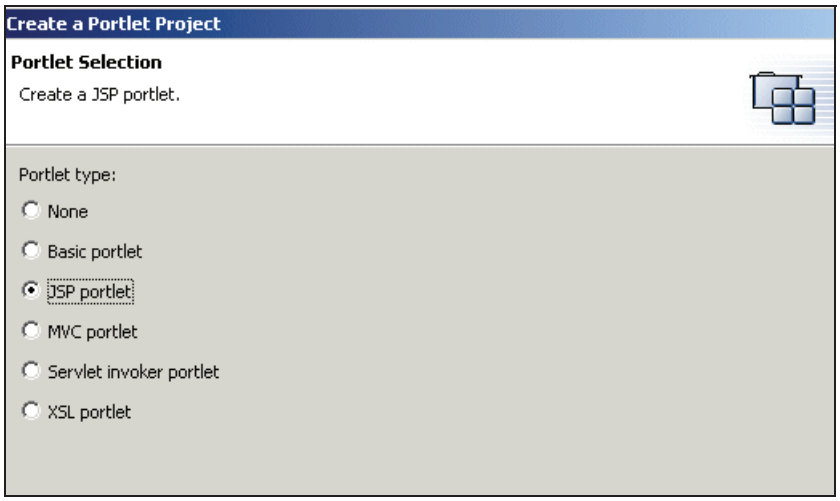


Figure 8-31 Selecting the type of portlet

8. Click **Next**.
9. Uncheck the HTML markup language and check the **WML** markup language selection. This will give the portlet support for only the WML markup language. Click **Finish** to accept the properties.

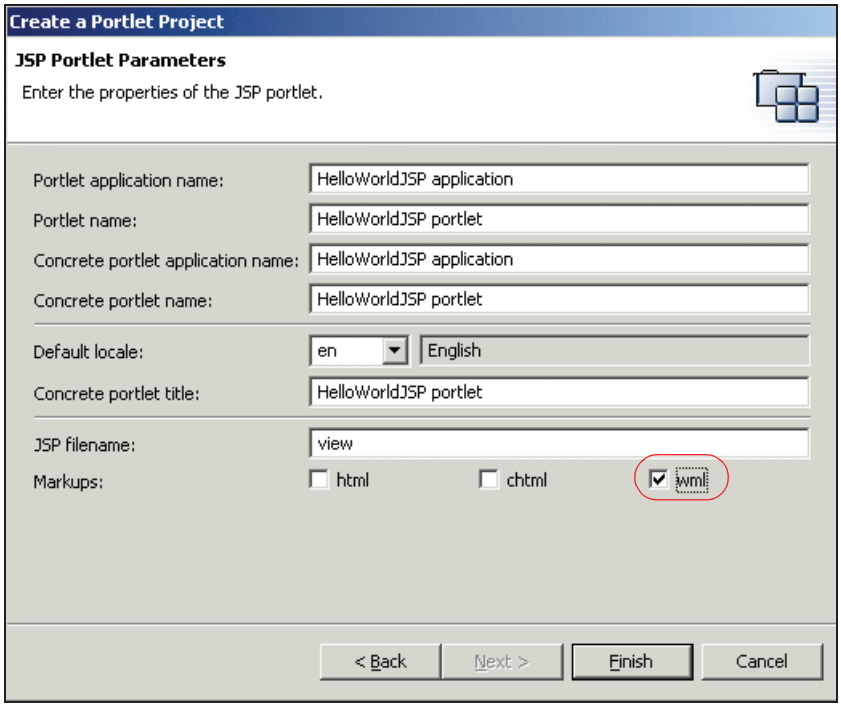


Figure 8-32 Selecting parameters and markup languages

10. If you receive the Server Configuration window, click **OK** to accept the changes.

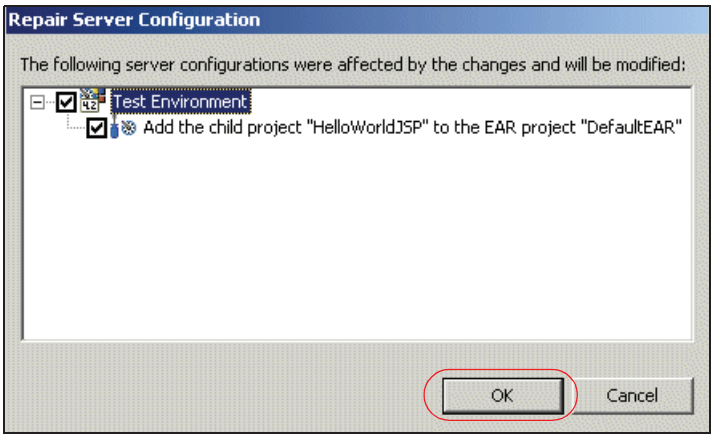


Figure 8-33 Server Configuration window

11. Once the project has been configured, the new project HelloWorldJSP will be included in the Navigator pane.

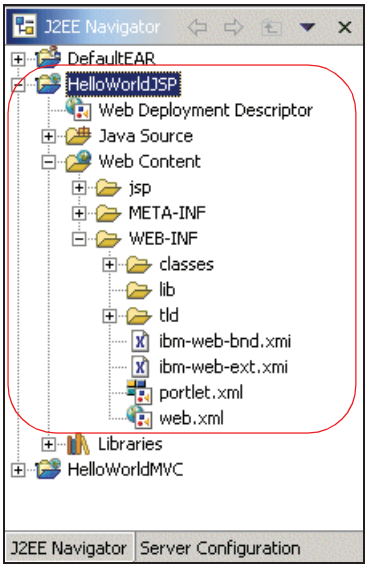


Figure 8-34 HelloWorldJSP project

12. If you already have portlet applications in the DefaultEAR project, remove them as follows.
    a. Open your **DefaultEAR / META-INF** folder
    b. Double-click the **application.xml** file
    c. Select **Modules**
    d. Remove all war modules except HelloWorldJSP
    e. Save any changes. For example, in the workspace click **File -> Save All**.

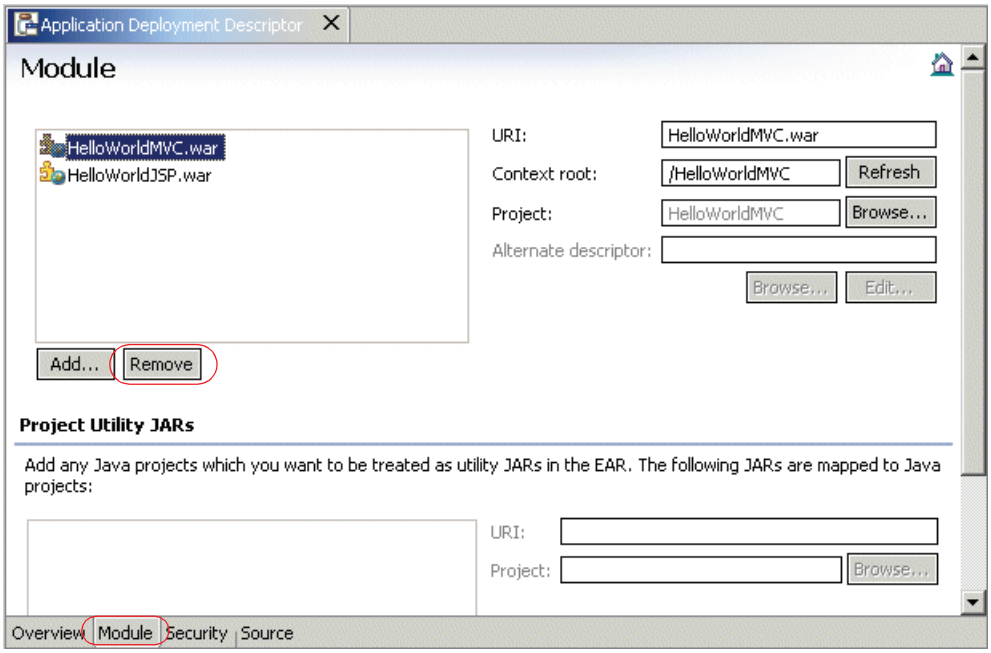


Figure 8-35 Removing unnecessary WAR modules

13. If you receive the Repair Server Configuration window, click **OK**.

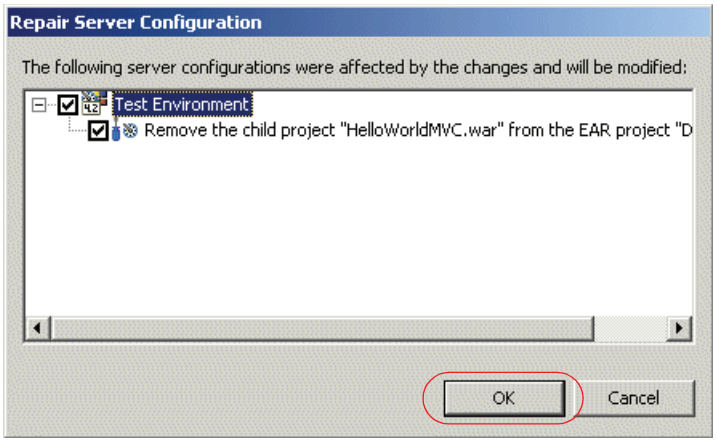


Figure 8-36 Updating Server Configuration window

14. Notice the Server Configuration tab of the Navigator pane now shows your project.

Important: If your project does not show up on the Server Configuration tab, you will need to add it by right-clicking the **Test Environment** and clicking **Add**.

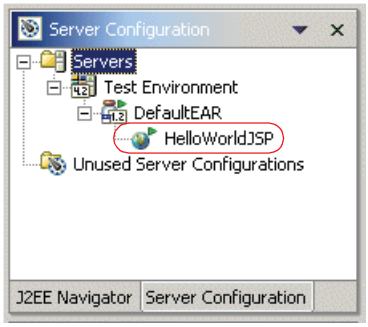


Figure 8-37 The project in the test environment

15. If the server is running while you make changes to the server's configuration, you will need to restart the server before these changes take effect. To do this, click the **Servers** tab along the bottom of the tasks area, right-click **Test Environment**, then click **Restart**. It may take a minute or more until the server has finished restarting.

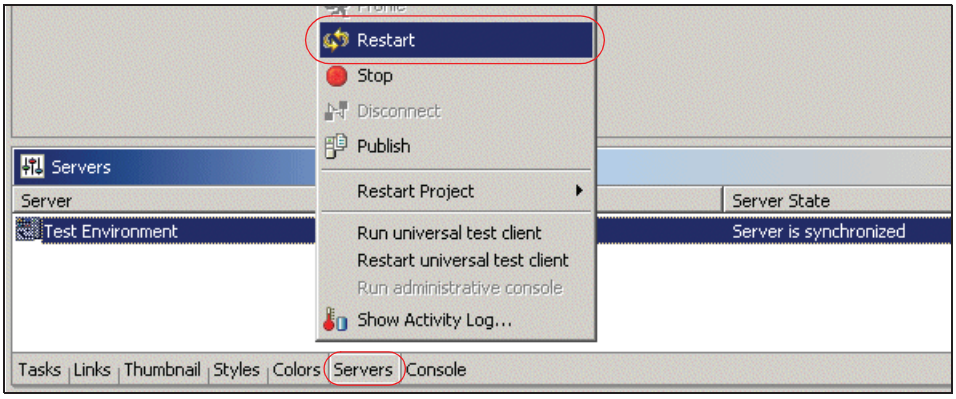


Figure 8-38 Restarting the server after changing its configuration

16. From the Navigator pane, right-click **HelloWorldJSP**, then select **Run with Device Emulator**.

Important: Make sure that the Openwave browser is configured as a device emulator in WebSphere Studio Site Developer before clicking **Run with Device Emulator**.

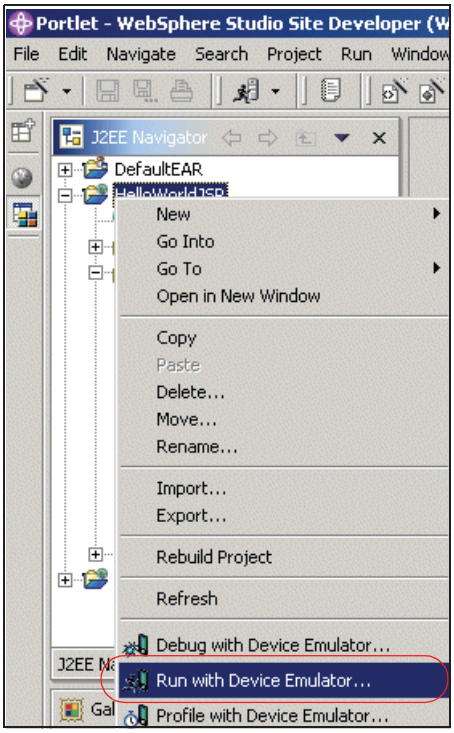


Figure 8-39 Run with Device Emulator

17. If you receive a Server Selection window, click **Finish** to use the existing test environment.

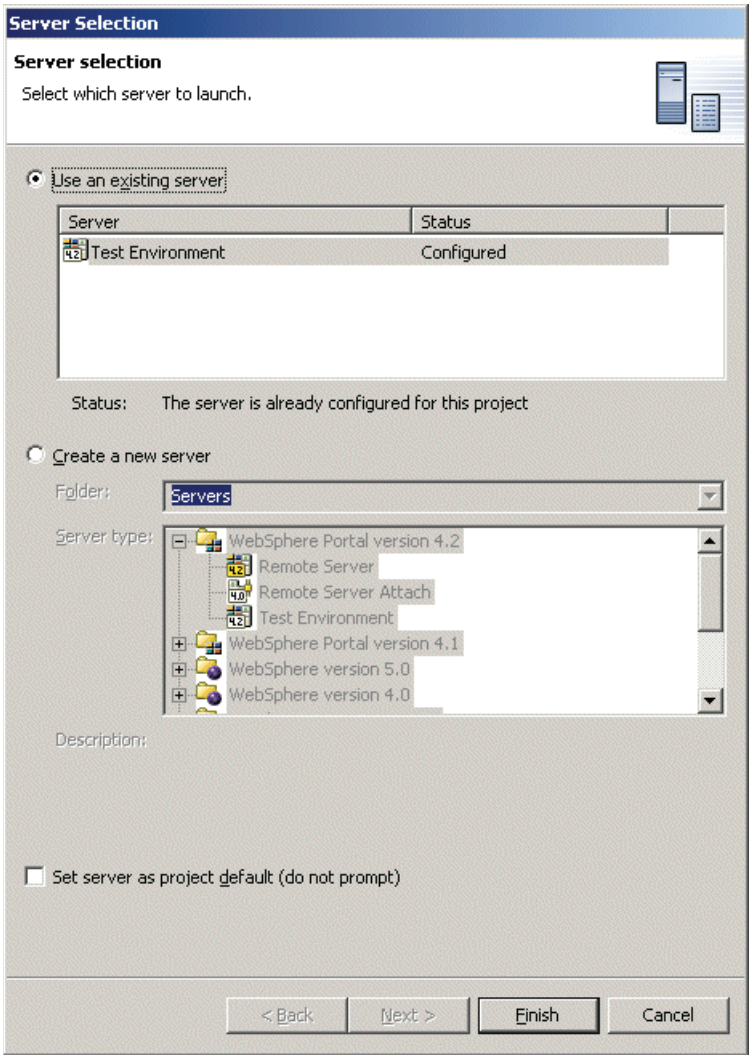


Figure 8-40 Server selection window

18. The portlet will run in the Openwave simulator. To run the HelloWorldJSP portlet, click the number **1** on the mobile phone.

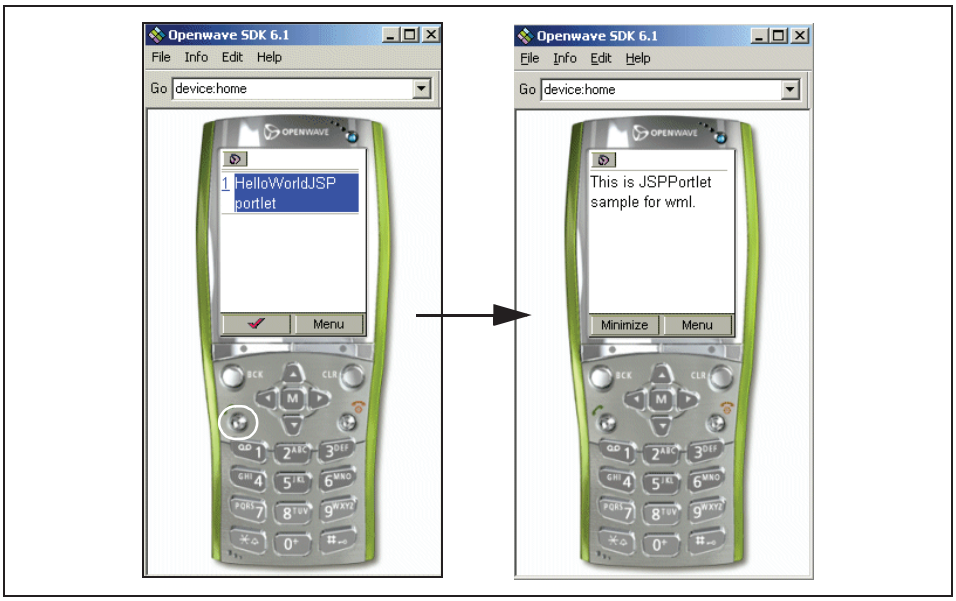


Figure 8-41 Running the portlet on the simulator

8.2.1 Modifying the HelloWorldJSP portlet

We will now make a change to the JSP portlet and see the result.

1. Expand the HelloWorldJSP project in the Navigator pane to see the portlet elements. Notice that it is a subset of the components found in the HelloWorldMVC.
2. Right-click **view.jsp** and choose **Open With -> Source Editor** to edit the code.

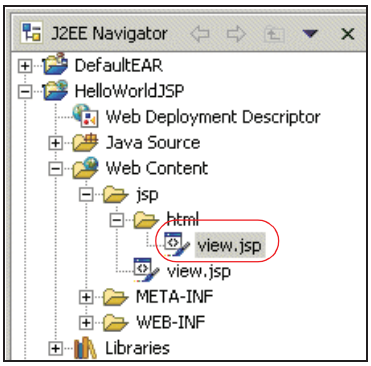


Figure 8-42 Accessing view.jsp

3. Update the code by adding new JSP expressions to display the current date, time and your workstation host name. The updated code is illustrated in Example 8-4.

Example 8-4 view.jsp

```
<h1>This is JSPPortlet sample. </h1>
<br>
<h2>Current time: <%=new java.util.Date() %></h2>
<br>
<h2>Your hostname: <%= request.getRemoteHost() %></h2>
```

4. Click **File -> Save All** to save your changes.
5. Click the **Servers** tab in the tasks area. Then right-click **Test Environment** and click **Publish** to update.

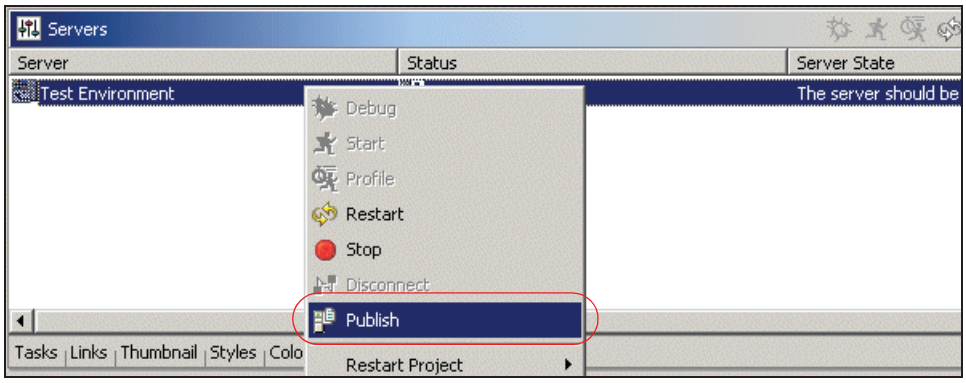


Figure 8-43 Publishing the updated project to the server

6. When the operation has completed a window appears. Click **OK**.
7. In the Navigator pane, select **HelloWorldJSP**, right-click and select **Run with Device Emulator**.
8. If you receive the Server Selection window, select the existing **Test Environment** and click **Finish**.
9. The updated portlet displays its View mode content in the simulator.



Figure 8-44 The updated portlet

10. Exit Site Developer by clicking **File -> Exit**.

9

Portlet action event handling

This chapter introduces you to the action event handling capabilities of portlet applications. Included is a scenario that will allow you to understand the techniques used to develop portlets that process action events.

In this chapter you will find the following topics:

- How action events are processed
- Creating a basic portlet to implement an action event
- Code samples
- Running the portlet from a Palm OS V5.2 Simulator
- Running portlet from the Openwave WAP browser simulator
- You can also download the sample code available as additional materials. See Appendix I, “Additional material” on page 1271.

© Copyright IBM Corp. 2003. All rights reserved.

9.1 Overview

Portlet events contain information to which a portlet might need to respond. To receive these events, an event listener must be implemented in the portlet class. The three most used types of events found in the Portlet API are:

- ActionEvents
- MessageEvents
- WindowEvents

There are also two additional events, PortletSettingsAttributeEvent and PortletApplicationSettingsAttributeEvent, which are used for notifications about changes to the attributes of the portlet settings of a concrete portlet or concrete portlet application.

In this lab, you will create a sample portlet based on the Basic portlet type using the Portlet wizard, you will then add code to support action event handling.

To send an ActionEvent you must associate a PortletAction with the HTTP request. The PortletAction is normally linked with URLs or buttons in an HTML form providing a way for portlet programmers to implement different processing actions for different user selections.

See Figure 9-1 on page 231 for an illustration of action event handling.

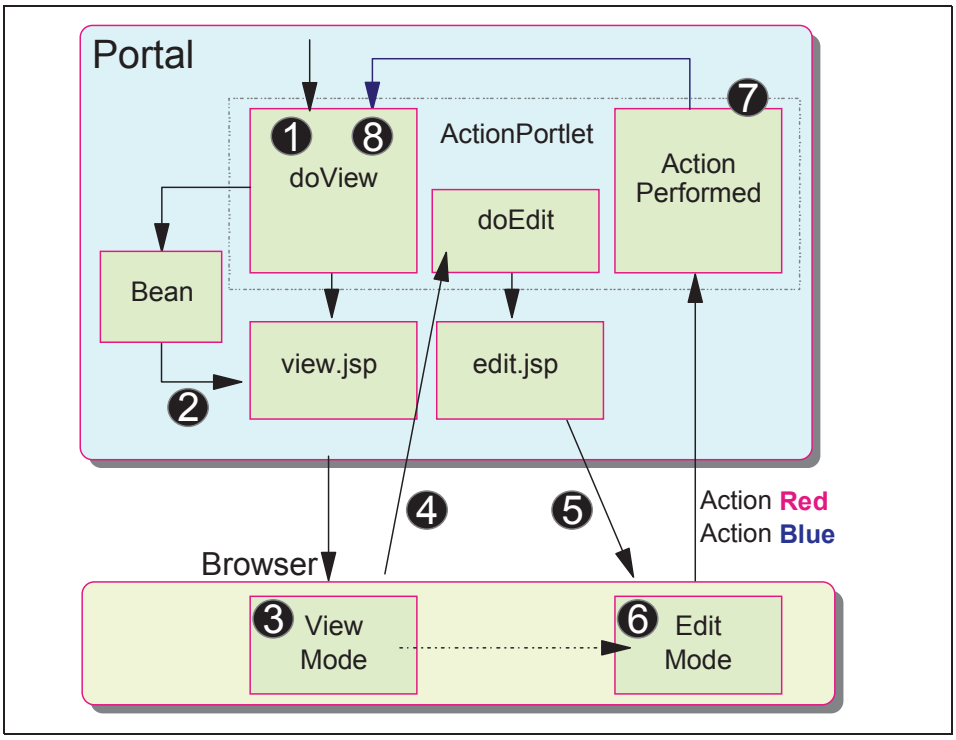


Figure 9-1 The event handling scenario

The sequence flow for this scenario is as follows:

1. The doView method is executed.
2. A JSP is invoked to render an initial screen. A message is obtained from the session object.
3. Portlet View mode window is shown in the browser.
4. User clicks **Edit** to go into Edit mode.
5. The doEdit mode is executed, and the Edit mode window is displayed.
6. User selects desired action button (in this example, red or blue).
7. actionPerformed method is executed to process the action. A resulting message is stored in the session object. Notice that portlet data can also be used to save the message.
8. The doView method is executed to complete the cycle and a message is obtained from the session object.

9.2 Create the ActionEvent portlet

In this section, you will create a Basic type portlet application with the name ActionEventLab. The portlet application will be published and executed in the test environment and viewed on the Palm Simulator.

1. Before you start this lab, make sure that the test environment is stopped and all perspectives (editors, browsers) have been closed. Click the **Server** tab in the tasks area to verify this.
2. Start the IBM WebSphere Studio Application Developer by clicking **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.0**.
3. Select **File -> New -> Other**.

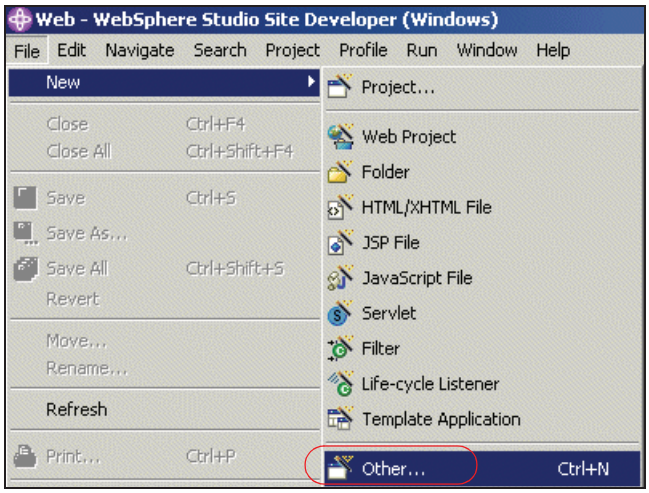


Figure 9-2 Selecting a new project to create

4. Select **Portlet development -> Portlet application project**.

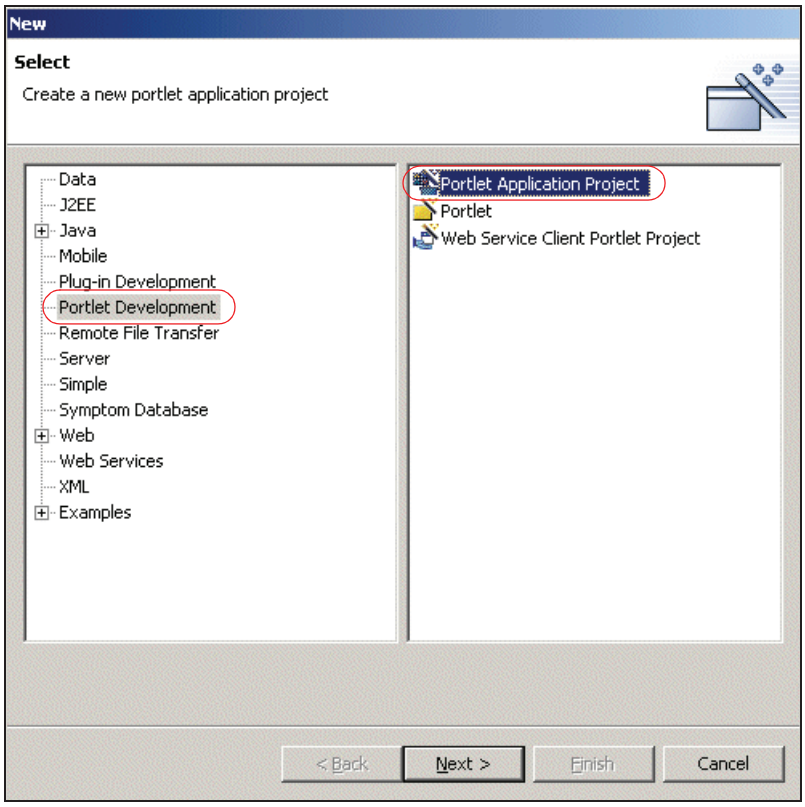


Figure 9-3 Selecting the type of project to create

5. Click **Next**.
6. On the Define the Portlet project window, enter the following information:
   - Project name: `ActionEventLab`
   - Enterprise Application project name: `DefaultEAR`.

     (If no Enterprise Application exists, the default new project is named DefaultEAR. If an Enterprise Application already exists, select **Existing** to use the existing DefaultEAR project).
   - Context root: `ActionEventLab`

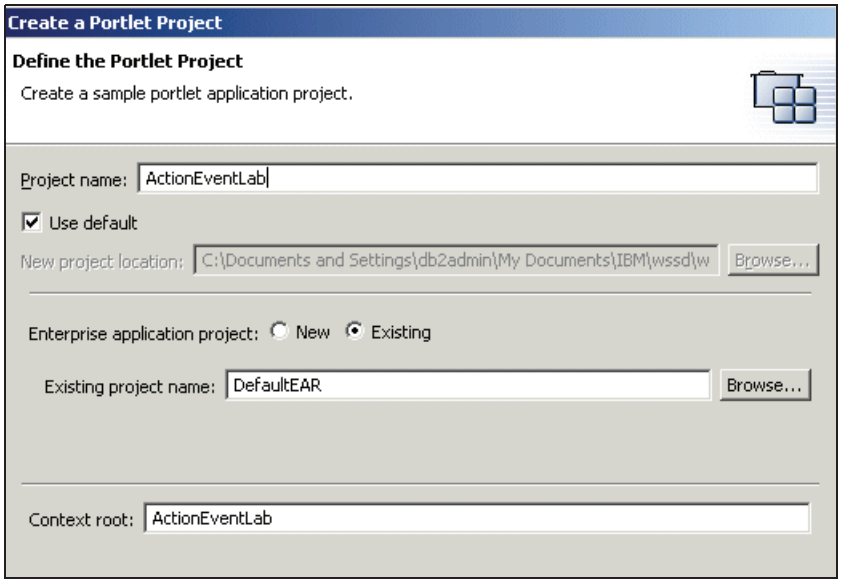


Figure 9-4 Defining the portlet project

7. Click **Next**.
8. Select **Basic Portlet** for the type of portlet to be generated.

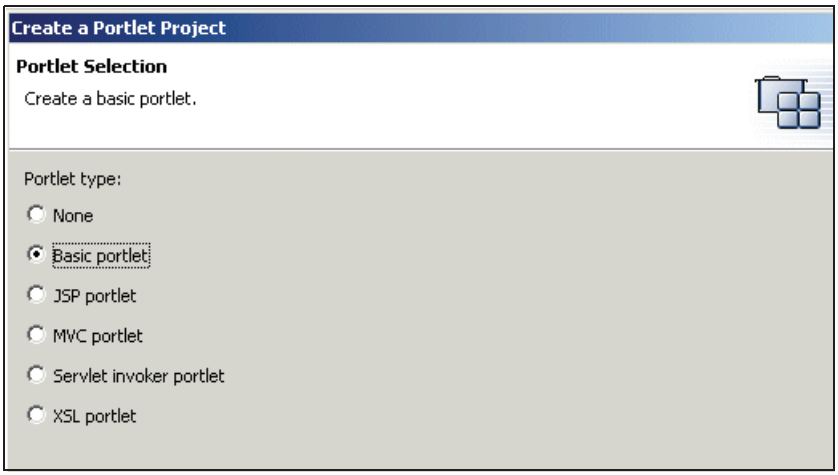


Figure 9-5 Selecting the portlet type

9. Click **Next**.
10. Enter `ActionPortlet` for the Portlet class name, check the WML box to enable WML support for the portlet, and click **Finish**.

Important: Make sure you enter `ActionPortlet` as the portlet class name.

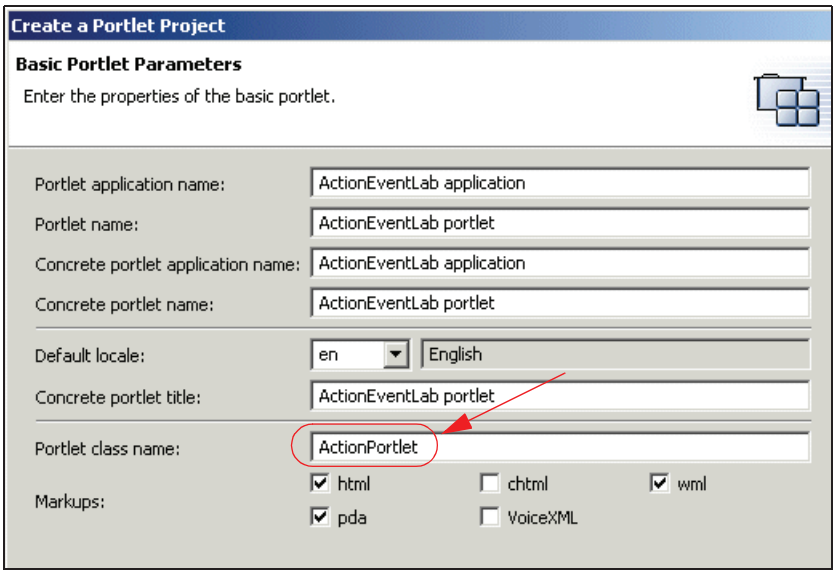


Figure 9-6 Changing the portlet class name

11. If you receive the Repair Server Configuration window, click **OK**.

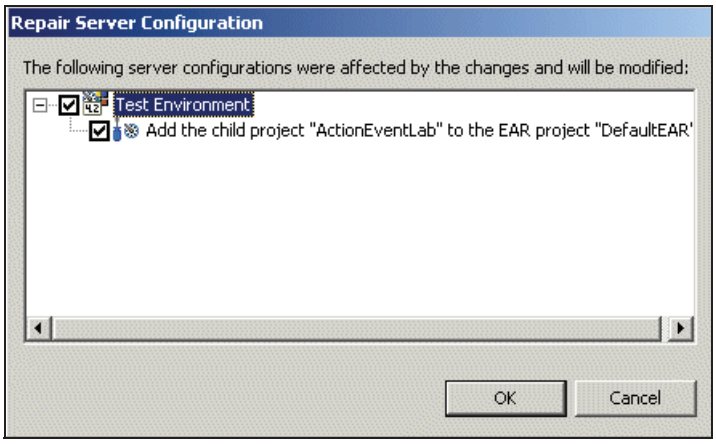


Figure 9-7 Server Configuration window

12. Once the project has been configured you will see the new project ActionEventLab in the Navigator list along with any other portlet projects you may have been working on.

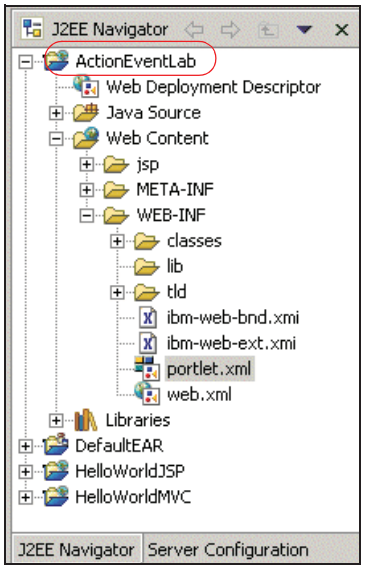


Figure 9-8 Navigator panel showing project

13. If you already have portlets in the DefaultEAR project, remove these as follows.

    a. Open your DefaultEAR / META-INF folder.

    b. Double-click the application.xml file.

    c. Select **Modules**.

    d. Remove all WAR modules except for this project ActionEventLab.

    e. In the workspace, click **File -> Save All**.

    f. If you receive the Repair Server Configuration window, click **OK**.

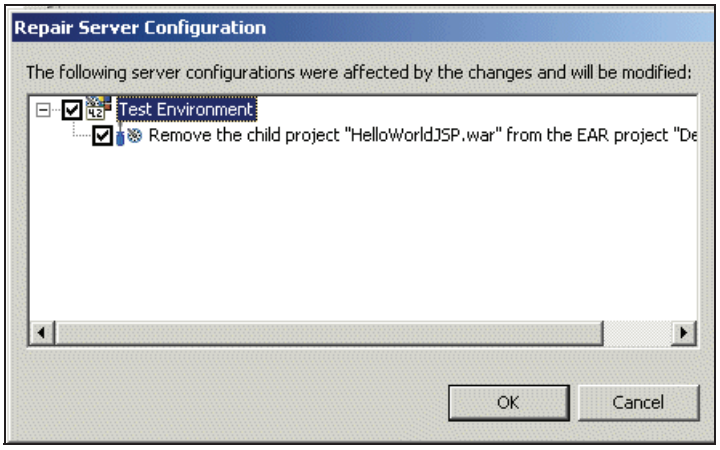


Figure 9-9 Repair Server Configuration window

9.3 Update the portlet code

If you want to import the WAR file, follow these steps:

1. Click **File -> Import**.
2. Select **WAR File** and click **Next**.
3. On the Import Resources window:
   a. Browse to the directory of the ActionEventLab.war file
   b. Select the existing project, **ActionEventLab**
   c. Check **Overwrite Existing Resources**
4. Click **Finish** to import the files.

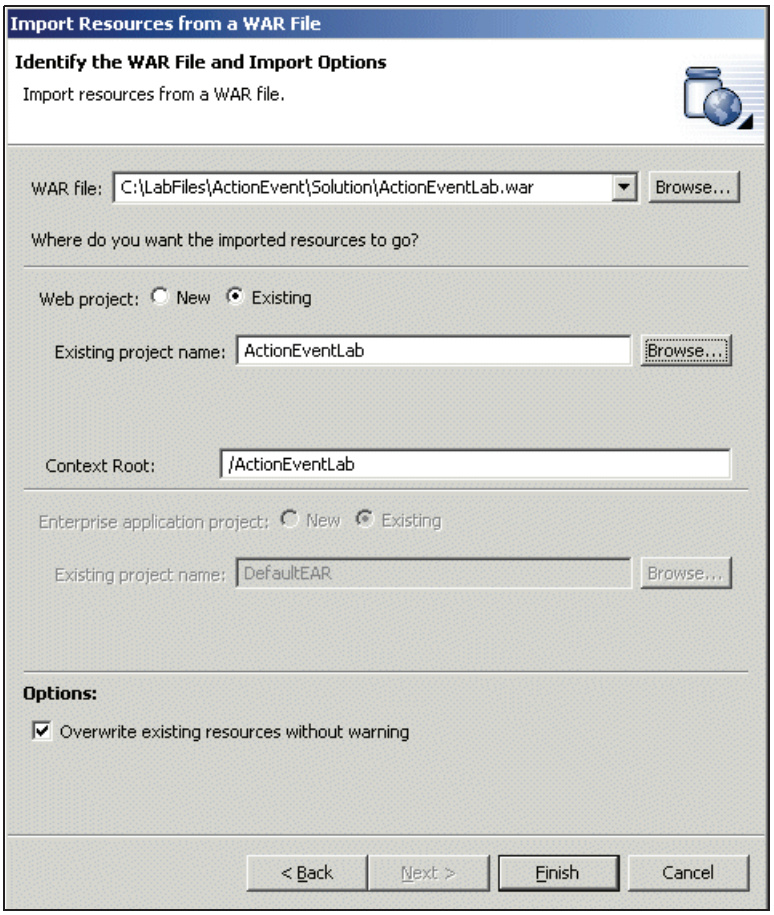


Figure 9-10 Importing files

9.4 Look inside the ActionEventLab project

1. Expand the ActionEventLab project in the Navigator pane.
2. In the Navigator pane, double-click **portlet.xml** to view the portlet settings.

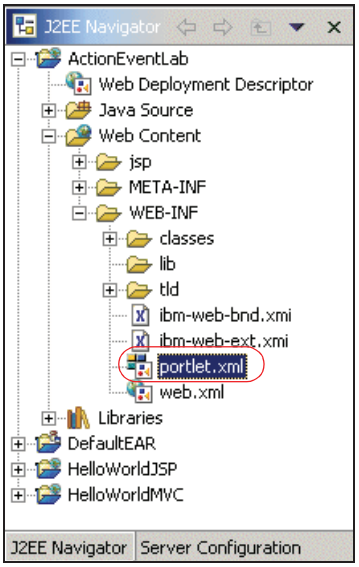


Figure 9-11 Navigator panel

3. The editor displays the portlet deployment descriptor values.

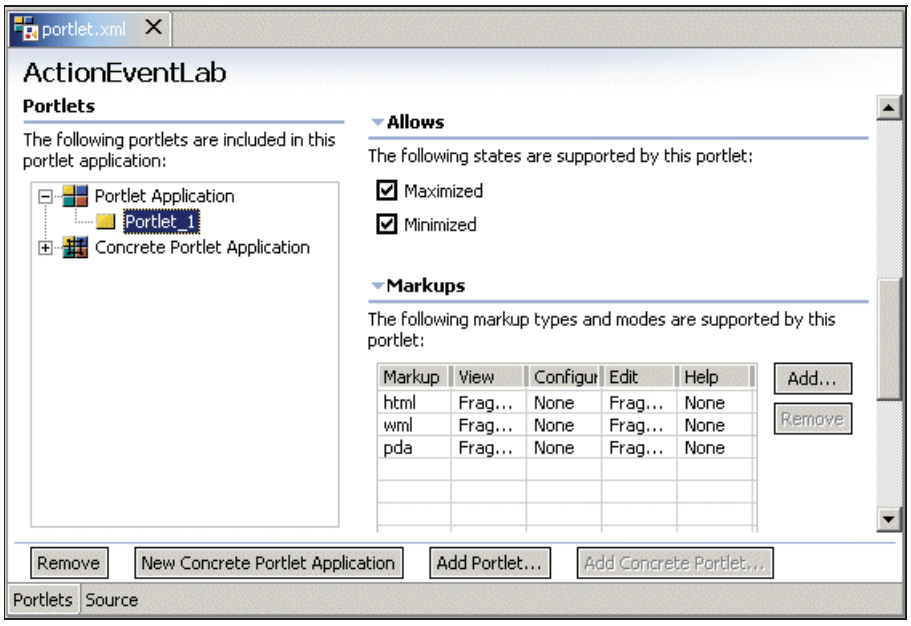


Figure 9-12 Portlet.xml editor

4. Note that the portlet deployment descriptor editor is set to render in Edit mode for HTML, PDA, and WML.
5. Next you will view the portlet code. The files you will view are:
   - ActionPortlet.java
   - Edit.jsp
   - View.jsp

   Note: The sample code can be obtained from the additional materials of this redbook. See Appendix I, “Additional material” on page 1271.

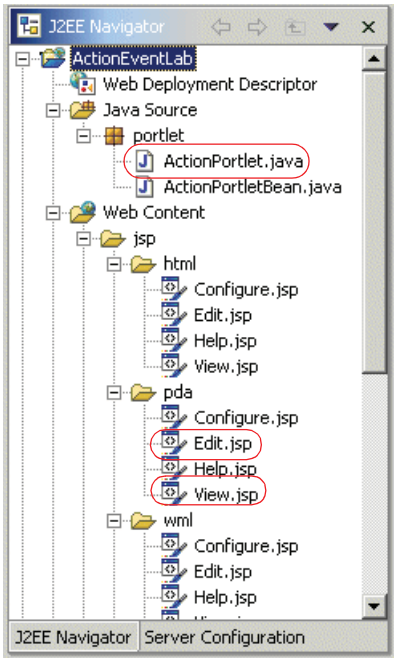


Figure 9-13 The files that you will modify

6. To edit ActionPortlet.java, double-click the file name.
7. The code is shown in Example 9-1, Example 9-2 on page 241, and Example 9-3 on page 241:
   a. In order for this portlet to support action events, the ActionListener interface must be implemented in the ActionPortlet.java class. The highlighted text indicates the necessary code to do this.

Example 9-1 ActionPortlet.java

```
...
import java.io.*;
import org.apache.jetspeed.portlet.*;
```

```
import org.apache.jetspeed.portlet.event.*;

// In order to support action events ActionListener interface must be
implemented
public class ActionPortlet extends PortletAdapter implements ActionListener {
...
```

b. Two strings are created to be used in the actionPerformed method.

Example 9-2 ActionPortlet.java

```
...
   public static final String ACTION_RED  = "ACTION.RED";
   public static final String ACTION_BLUE = "ACTION.BLUE";
...
```

c. This implements code to inherit the method actionPerformed. In this method, the selected action is checked. To do this, a String class is used to retrieve the action, a new string is created, and the information is saved in the PortletSession, and finally, the information saved in PortletSession will be used to render the View.jsp page.

Example 9-3 ActionPortlet.java

```
...
public void actionPerformed (ActionEvent event) throws PortletException
    {
   // Retrieve the action event
   String action = (String) event.getActionString();

   // Check which action was selected
   if (action.equalsIgnoreCase(ACTION_RED))
   {
      // Create a string to be rendered
      String value = "Action <FONT color=\"#ff0000\">RED</FONT>";
      // Create a Portlet request
      PortletRequest request = event.getRequest();
      // Create a session
      PortletSession session = request.getPortletSession();
      // Save value in session
      session.setAttribute("action",value);
   }

   // Check which action was selected
   if (action.equalsIgnoreCase(ACTION_BLUE))
   {
      // Create a string to be rendered
      String value = "Action <FONT color=\"#0000ff\">BLUE</FONT>";
      // Create a Portlet request
```

```
        PortletRequest request = event.getRequest();
        // Create a session
        PortletSession session = request.getPortletSession();
        // Save value in session
        session.setAttribute("action",value);
    }
}
...
```

8. Double-click **Edit.jsp** under the pda directory to edit the file.

   Important: Make sure you select the Edit.jsp file in the pda folder.

9. View the code as follows:

   a. The portlet container provides tags for use in portlet JSPs. You need to include this directive to make these tags available:

      ```
      <%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
      ```

   b. You also need to specify the following package to be imported:

      ```
      <%@ page import="portlet.*" %>
      ```

   c. In this scenario, two HTML forms are used that create a URI that returns the user to the portlet's previous mode (using the createReturnURI tag) in order to pass an action using the URIAction tag corresponding to the action selected.

Example 9-4 Edit.jsp

```
...
<TABLE class="Portlet" border="0" cellspacing="0" cellpadding="0">
  <TR>
    <TD><B>Please select an action:</B>
    <FORM method='POST' action="<portletAPI:createReturnURI>
                                <portletAPI:URIAction
                                      name='<%=ActionPortlet.ACTION_RED%>'/>
                                </portletAPI:createReturnURI>">
      <TABLE class="Portlet" border="0" cellspacing="0" cellpadding="0">
        <TR>
          <TD>  <INPUT type='submit' name='redButton' value='Red
                                                              Action'></TD>
        </TR>
      </TABLE>
      </FORM>

    <FORM method='POST' action="<portletAPI:createReturnURI>
                                <portletAPI:URIAction
                                      name='<%=ActionPortlet.ACTION_BLUE%>'/>
```

```
                                </portletAPI:createReturnURI>">
      <TABLE class="Portlet" border="0" cellspacing="0" cellpadding="0">
        <TR>
          <TD>  <INPUT type='submit' name='blueButton' value='Blue
                                                                  Action'></TD>
        </TR>
      </TABLE>
    </FORM>
    </TD>
  </TR>
</TABLE>
...
```

10. Save your changes.
11. Double-click **View.jsp** under the pda directory to edit the file.
12. The following JSP scriptlets are included to conditionally show information in View mode. Until the user selects an action in Edit mode, the message `No action performed` is shown. Select your action in Edit Mode. When the user selects an action, the JSP Expression `<%=session.getAttribute("action") %>` retrieves the action value and prints it in View mode.

Example 9-5 View.jsp

```
...
<% if (session.getAttribute("action") == null) { %>
   <B>No action performed, select your action in Edit Mode</B>
<% } else { %>
   <B><%= session.getAttribute("action") %> ...was selected!</B>
<% } %>
...
```

13. Select **File -> Save All** to save all your changes to the project.
14. Click **Server**. Right-click **Test Environment** and select **Publish**.
15. In the Navigator pane select and right-click **ActionEventLab**, then select **Run on Server**.

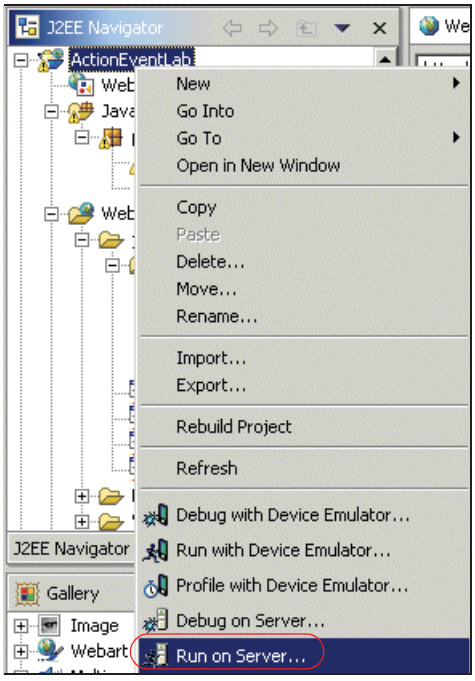


Figure 9-14 Running the portlet project

16. The portlet will be run in the built-in browser shown in the upper right pane.
17. Open the Palm Simulator, and open a Web browser (Refer to “Displaying the portlet content in the Palm OS V5.2 Simulator” on page 203 for information on how to do this). Enter `http://localhost/wps/myportal` (or the location where you are testing your portlets) as the URL to view, and click **Go**.
18. You will be prompted to log in to WebSphere Portal. Enter `wpsadmin` for the user name and password. Click **Log in** to continue.

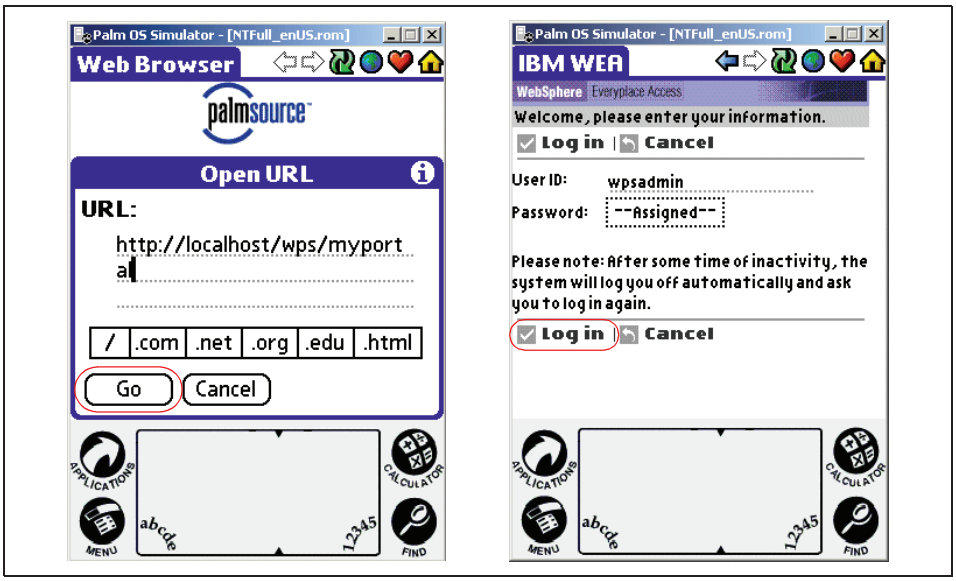


Figure 9-15 Browsing to the portlet project with the Palm Simulator

19. You will then see the ActionEvent portlet icon. Click this icon to view the ActionEvent portlet.

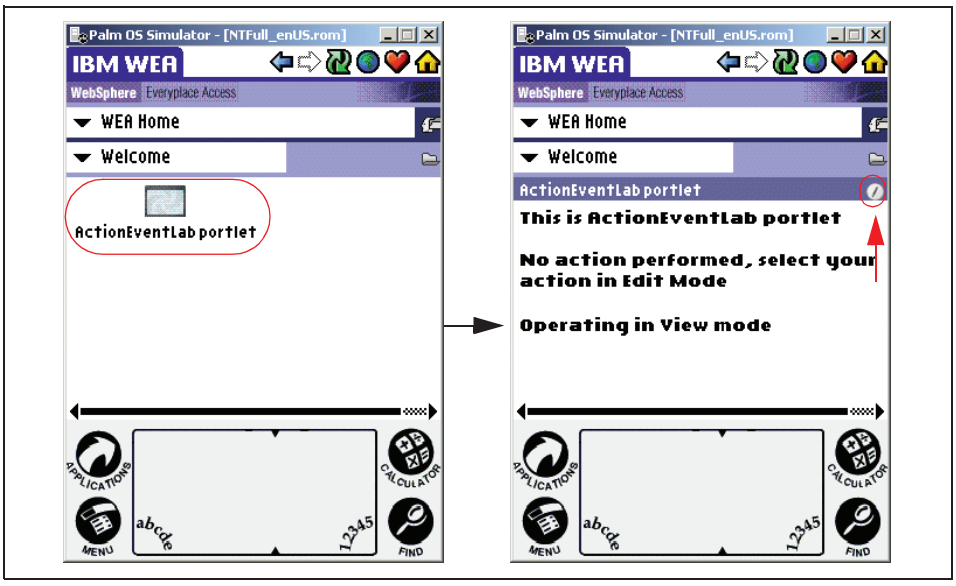


Figure 9-16 Viewing the portlet

20. To access the edit mode of the portlet, click the **Edit** icon in the title bar as highlighted in Figure 9-16 on page 245.

21. You will now see the ActionEvent portlet running in edit mode. Click the **Red Action** button to send an action event. The actionPerformed() method implemented in ActionPortlet.java will receive this action event and store it in PortletSession. You will be sent back to the icon listing of the available portlets. Click the **ActionEvent** portlet to display that the Red Action was selected.

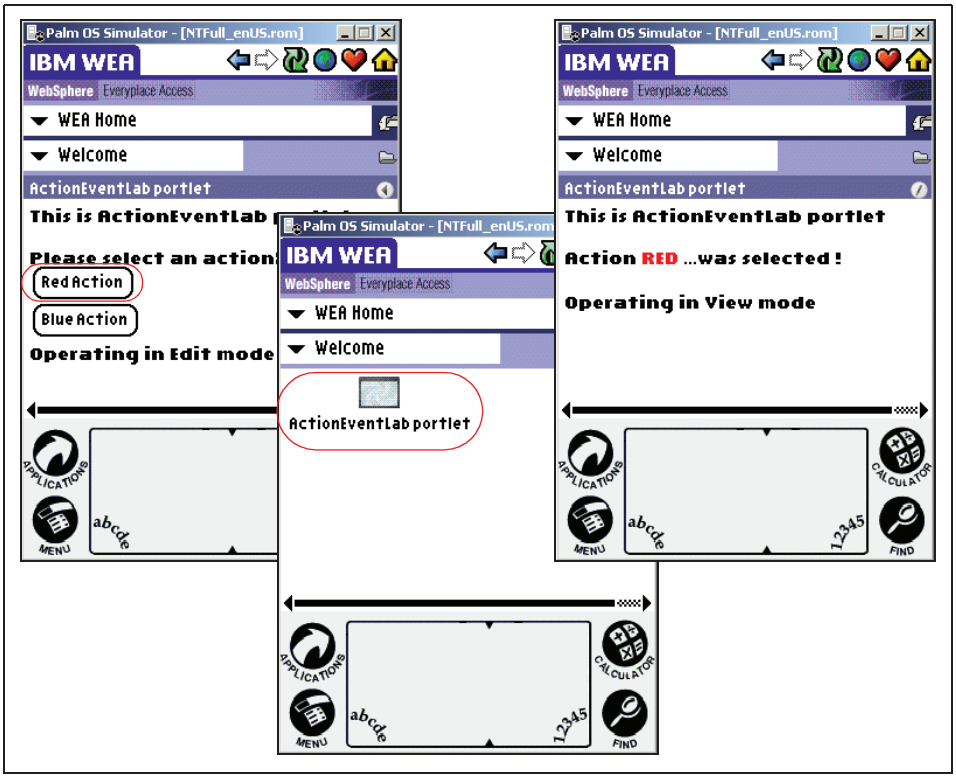


Figure 9-17 Performing an action

22. Click the **Edit** icon to enter edit mode again. This time, click the **Blue Action** button and see the result.

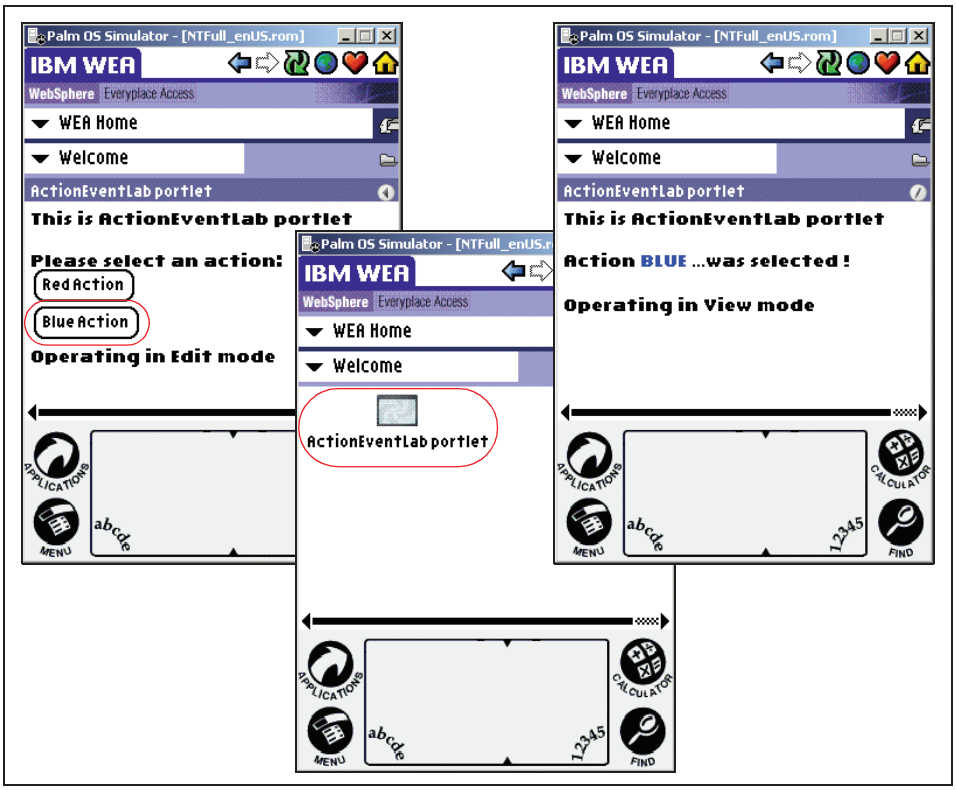


Figure 9-18 Selecting an alternate action

23. The Blue Action button sends a different value to View.jsp, resulting in a different message being displayed.

9.5 WML and the ActionEvent portlet

Next, you will examine the source code that generates the WML content and run your portlet on the Openwave WAP browser simulator. It is assumed that you have set up this simulator as a Device Emulator from within WebSphere Studio Site Developer.

9.5.1 Examine the code that renders the content

As you learned before, the MVC architecture allows you to separate the process from the view. Following this architecture, this portlet can display WML markup by including an additional set of JSPs. The ActionPortlet.java file does not need to be modified in any way.

To examine the source code of the JSPs, follow these steps:

1. In the Navigator pane, open the \ActionEventLab\Web Content\jsp\wml folder. Right-click **Edit JSP** and then select **Open With -> Source Editor**.

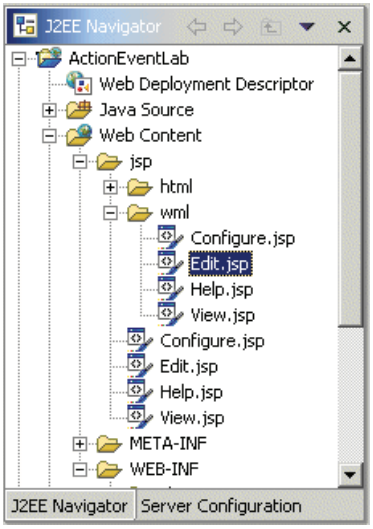


Figure 9-19 WML JSPs

When you select the check box to support WML in the Portlet Application wizard, a new folder named wml is created in the jsp directory under Web Content. These JSPs are used to generate the WML content.

Notice that the Edit JSP uses the WML markup language instead of HTML to render the content of the page. This makes it compatible with a WAP browser. You can see the differences between WML and HTML in Example 9-6, especially in form handling.

Example 9-6 Edit.jsp for WML markup

```
<%@ page contentType="text/vnd.wap.wml" %>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ page import="portlet.*" %>
<jsp:useBean id="ActionPortletBean" class="portlet.ActionPortletBean"
scope="request" />

<b>This is <%=ActionPortletBean.getPortletName()%></b>
<br/><b>Select action:</b>
<br/>

<do name="te2" type="accept" label="Red Action">
   <go href="<portletAPI:createReturnURI>
               <portletAPI:URIAction name='<%=ActionPortlet.ACTION_RED%>'/>
```

```
            </portletAPI:createReturnURI>" method="post"
                                                accept-charset="ISO-8859-1">
       <postfield name="redButton" value="Red Action"></postfield>
   </go>
</do>

<br/>

<do name="te3" type="accept" label="Blue Action">
   <go href="<portletAPI:createReturnURI>
               <portletAPI:URIAction name='<%=ActionPortlet.ACTION_BLUE%>'/>
            </portletAPI:createReturnURI>" method="post"
                                                accept-charset="ISO-8859-1">
       <postfield name="blueButton" value="Blue Action"></postfield>
   </go>
</do>
...
```

2. To run the portlet, in the Navigator pane, right-click the **ActionEventLab** project and select **Run with Device Emulator**.

Important: You must have the Openwave browser simulator configured as a device emulator for the Run with Device Emulator selection to work.

3. You will then see the portlet listing on the WAP browser simulator. Click the number **1** to view the ActionEventLab portlet.



Figure 9-20 Portlet listing

4. You are now in view mode. To navigate to edit mode and select an action, click the **Menu** button, then select the number **4** on the keypad. Alternatively, you can use the arrow keys on the phone to move down to Edit, and click the **Checkmark** button to accomplish the same action.



Figure 9-21 Navigating to edit mode

5. You are now in edit mode. Notice that there are no icons on your screen. To select an action, you must again click the **Menu** button. Click the number **1** on the keypad to select the **Red Action**. Doing this will return you to the portlet listing.

.

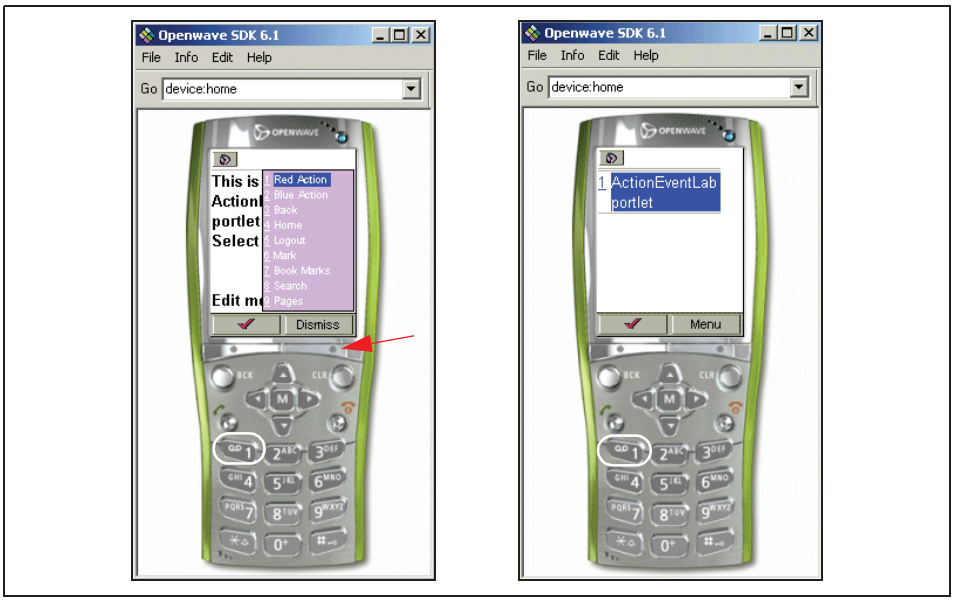


Figure 9-22 Selecting the Red Action, storing it in the session

6. Click number **1** on the keypad to return to the ActionEventLab portlet and view the result of your action.



Figure 9-23 Portlet action received and message displayed

7. You can click the **Menu** button again and follow steps 5 and 6 to select the Blue Action and see the result.



Figure 9-24 Result for Blue Action

10

Portlet messaging

Message events are a way for portlets to communicate messages to each other. This is accomplished through a familiar event listener model. Portlets that need to listen for message events must implement a MessageListener interface, and portlets that need to send message events do so within the handling of their own ActionEvents, as you will see in the following scenario. Message events can be sent to named portlets or broadcast to all portlets on the same Portal page. All events are handled during the page's event-processing phase, which comes before the content generation phase, as detailed in "Portlet events" on page 132.

In this chapter you will find the following topics:

- How message events are processed
- Using a portlet to send a message
- Receiving and displaying a message
- Running the message event handling portlet on the Palm Simulator

© Copyright IBM Corp. 2003. All rights reserved.

10.1 Overview

In this lab you will enhance the ActionEvent portlet application that you created in the previous chapter to send messages to a new message receiver portlet.

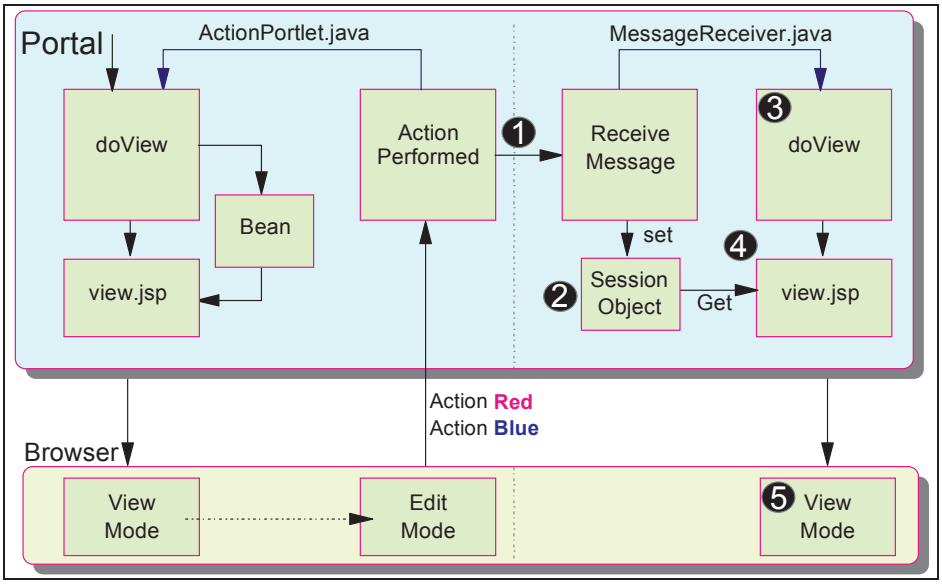


Figure 10-1 Message Event handling scenario

The flow for this scenario as follows:

1. The actionPerformed() method in the ActionPortlet.java portlet will be extended to send a broadcast message event (DefaultPortletMessage) upon arrival of action events.

   The MessageReceiver portlet that will implement the MessageListener interface receives the message in the new messageReceived method.

2. In this scenario, the received message is saved in the PortletSession object but portlet data can also be used.
3. Portal invokes the doView method, which in turn invokes the JSP (select).
4. The JSP retrieves the message from the session object and displays the message.

This scenario will be implemented using a broadcast style of message event rather than point-to-point messaging. In addition, the DefaultPortletMessage object will be used.

Note: While a DefaultPortletMessage object allows you to send messages to portlets in different applications, you can only send a String type message.

10.2 Using a portlet to send a message

This section builds upon the ActionEvent portlet constructed in Chapter 9, “Portlet action event handling” on page 229 to send out a broadcast message from within its actionPerformed method. The message will be broadcast to all portlets implementing the MessageListener interface and using the DefaultPortletMessage object. Follow these steps to complete this modification and view this scenario:

1. Before you start this lab, make sure the test environment server has been stopped and all perspectives, editors, and browsers have been closed.
2. If not already running, start the IBM WebSphere Studio Application Developer by clicking **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.0.**
3. Next you will update the actionPerformed() method to instantiate a DefaultPortletMessage object (the parameter `value` contains the message to be included in the object) and send the message (the parameter `null` indicates that this is a broadcast message). Add the highlighted code to the actionPerformed method into ActionPortlet.java as shown in Example 10-1.

Example 10-1 ActionPortlet.java

```
// Check which action was selected
   if (action.equalsIgnoreCase(ACTION_RED)) {
          // Create a string to be rendered
          String value = "Action <FONT color=\"#ff0000\">RED</FONT>";
          // Create a Portlet request
          PortletRequest request = event.getRequest();
          // Create a session
          PortletSession session = request.getPortletSession();
          // Save value in session
          session.setAttribute("action", value);

          PortletMessage message = new DefaultPortletMessage(value);
          try {

             this.getPortletConfig().getContext().send(null,message);

          } catch (AccessDeniedException e) {
          }
      }
// Check which action was selected
```

```
    if (action.equalsIgnoreCase(ACTION_BLUE)) {
          // Create a string to be rendered
          String value = "Action <FONT color=\"#0000ff\">BLUE</FONT>";
          // Create a Portlet request
          PortletRequest request = event.getRequest();
          // Create a session
          PortletSession session = request.getPortletSession();
          // Save value in session
          session.setAttribute("action", value);

          PortletMessage message = new DefaultPortletMessage(value);
          try {

             this.getPortletConfig().getContext().send(null,message);

          } catch (AccessDeniedException e) {
          }
}
```

4. Next you will slightly update the View.jsp page to notify that you are now sending a message. Double-click **View.jsp** under the html directory.

   Important: Make sure it is View.jsp in the html directory.

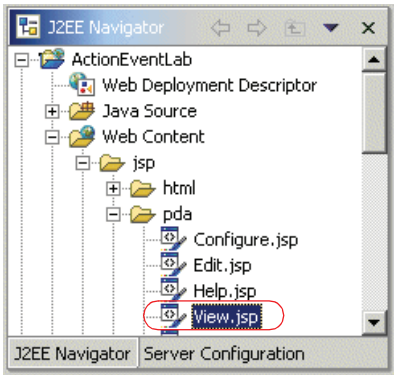


Figure 10-2 View.jsp file

5. Insert the text highlighted in Example 10-2.

Example 10-2 View.jsp

```
<%@ page contentType="text/html"%>

<jsp:useBean id="ActionPortletBean" class="portlet.ActionPortletBean"
scope="request" />
```

```
<B>This is <%=ActionPortletBean.getPortletName()%></B>
<BR>
<% if (session.getAttribute("action") == null) { %>
   <B>No action performed, select your action in Edit Mode</B>
<% } else { %>
   <B><%= session.getAttribute("action") %> ...was selected ! and this
information was broadcasted as a message.</B>
<BR>
<B>Operating in View mode</B>
```

6. Save and close the ActionPortlet.java file.

At this point you have implemented all the required logic in ActionPortlet to be able to send a broadcast message from within its actionPerformed() method.

10.3 Creating the target portlet

In this section you will create a target portlet to receive the message sent by the message sender portlet ActionPortlet.java.

1. You will create a new portlet within the same portlet application. The following files are provided:
   - MessageReceiver.java
   - MessageReceiverBean.java
   - ViewMessage.jsp
2. Select **ActionEventLab -> Java Source -> portlet**, as shown in Figure 10-3.

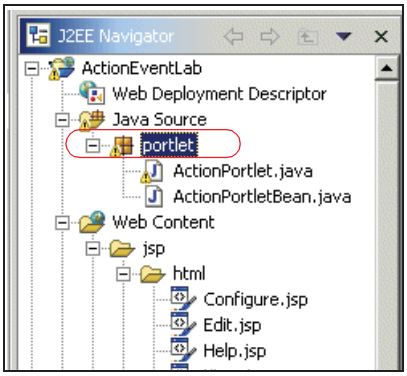


Figure 10-3 The portlet folder

3. Import the new files into the workspace. Select **File -> Import**.

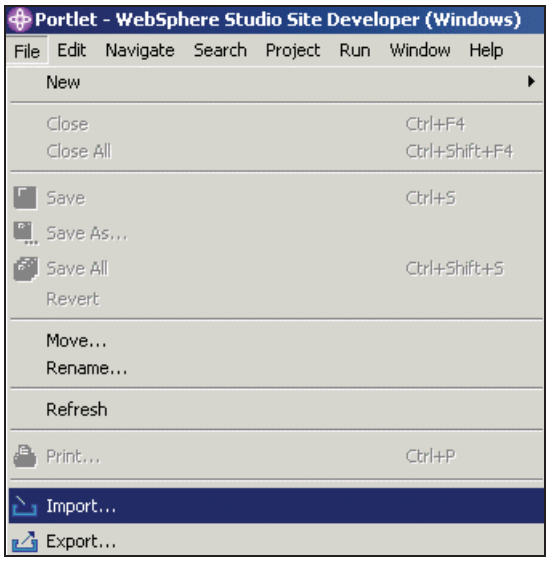


Figure 10-4 Importing a file

4. Select **File system** and click **Next**.

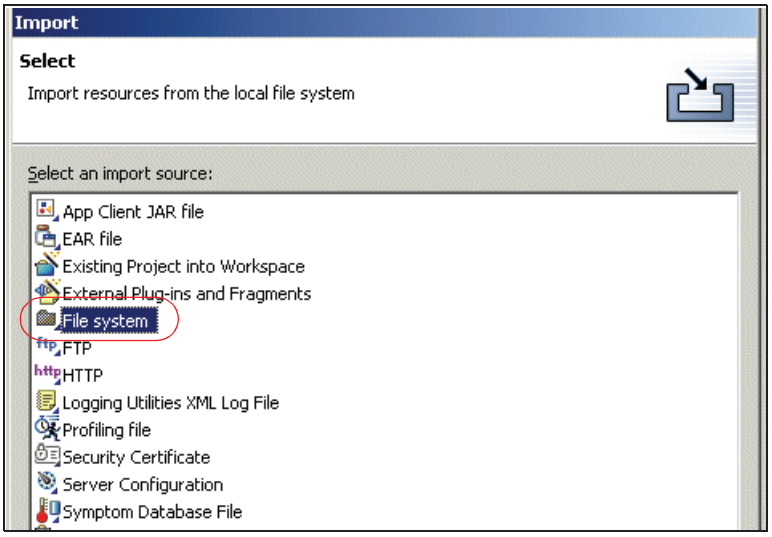


Figure 10-5 Selecting the file system

5. In the Import resources from a local file system window, browse to the directory where you downloaded the Redbook sample code and select

MessageReceiver.java and **MessageReceiverBean.java**, as shown in Figure 10-6.

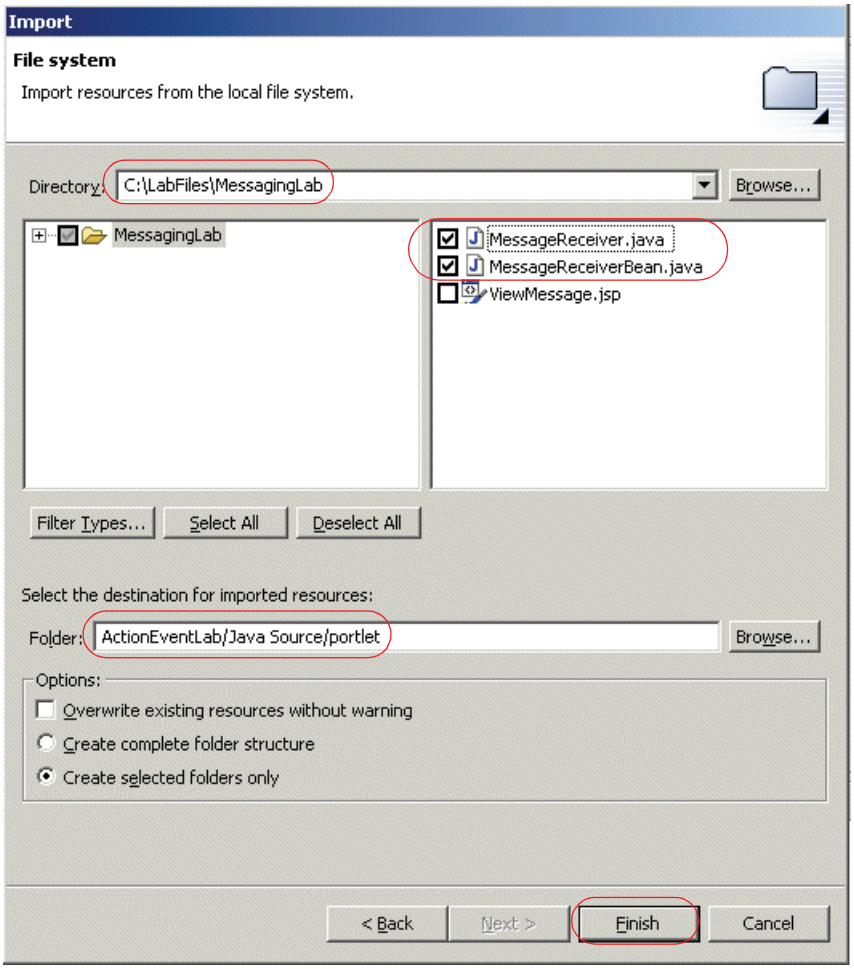


Figure 10-6 Selecting which files to import

6. Click **Finish**.
7. In a similar way, execute the same procedure for the ViewMessage.jsp file. Select the **ActionEventLab/webApplication/jsp/html** folder.
8. Import the JSP file into the workspace. Select **File -> Import**.
9. Select **File system** and click **Next**.

10. In the Import resources from a local file system window, browse to the directory where you downloaded the Redbook sample code and select **ViewMessage.jsp**, as shown in Figure 10-7.

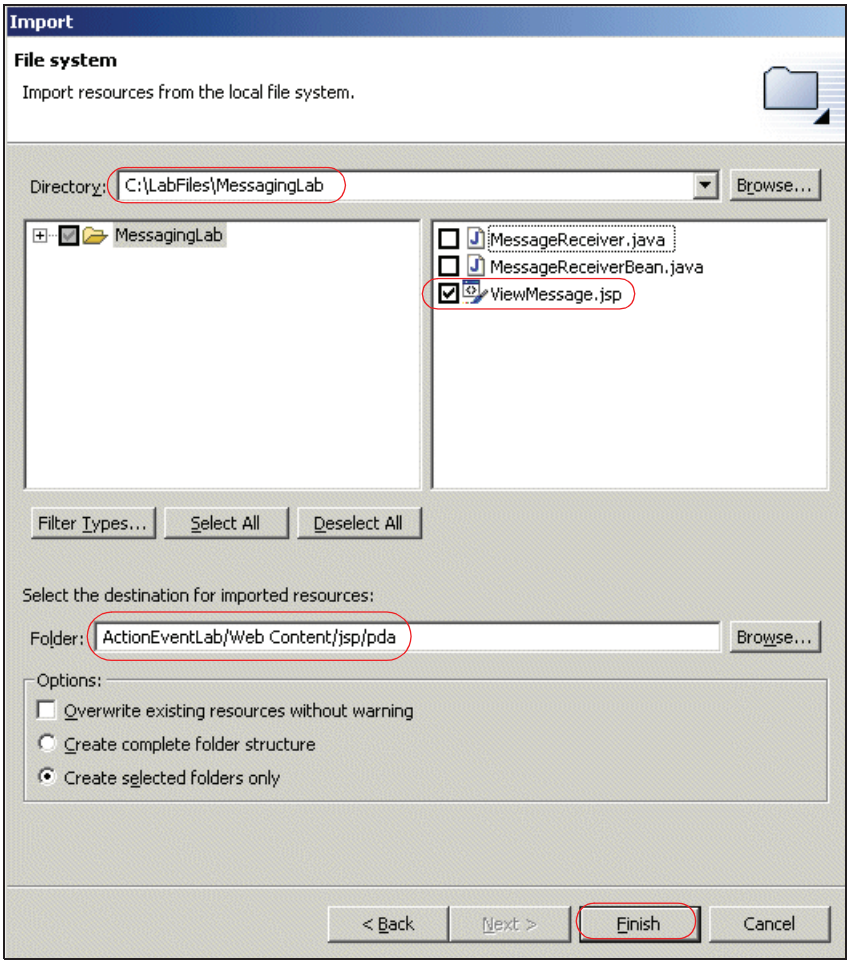


Figure 10-7 Selecting which file to import

11. Click **Finish**.

12. Your directory structure should now look as shown in Figure 10-8 on page 261.

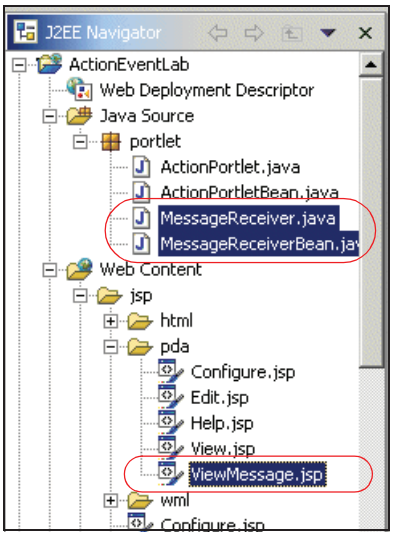


Figure 10-8 The ViewMessage.jsp file

13. Next, you will need to update the web.xml and portlet.xml deployment descriptors to include the new MessageReceiver portlet. Update web.xml as follows:

    a. Open the web.xml file in ActionEventLab/webApplication/WEB-INF.

    b. Click the **Details** button under the Servlets and JSPs heading.

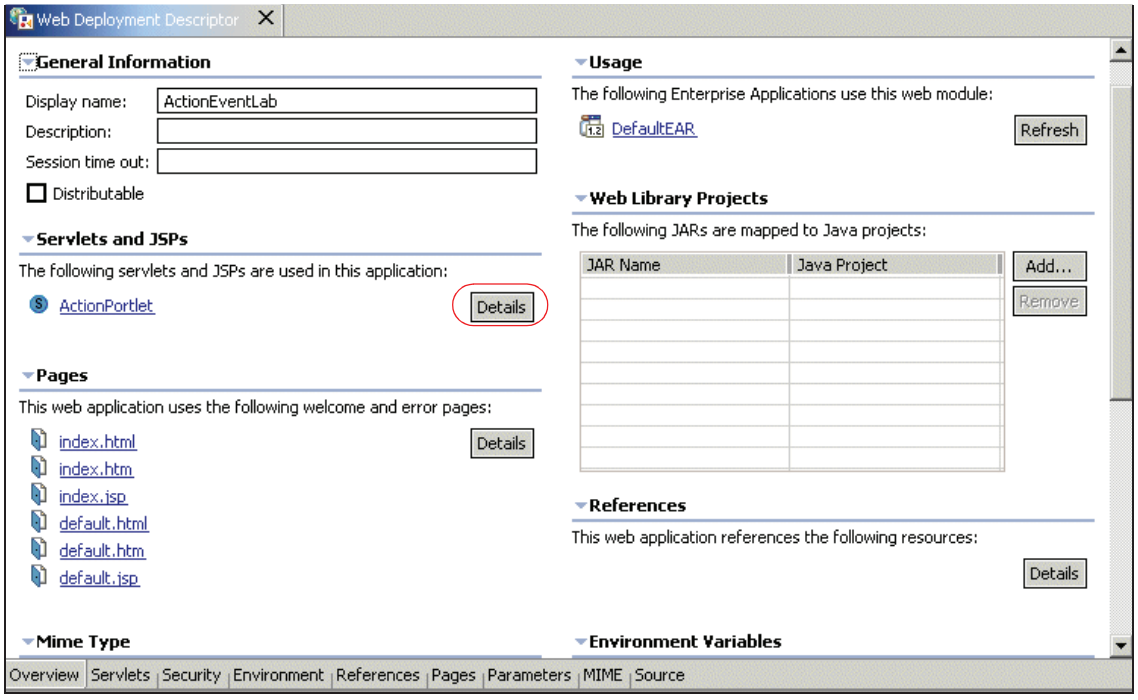


Figure 10-9 Web.xml editor

c. Click the **Add** button.

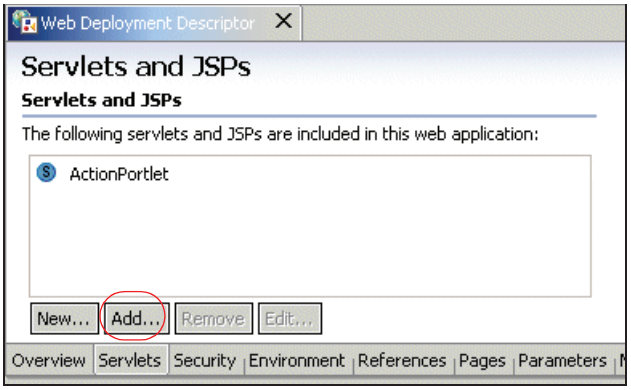


Figure 10-10 Adding a servlet to your Web application

d. Choose **MessageReceiver** and click **OK**.

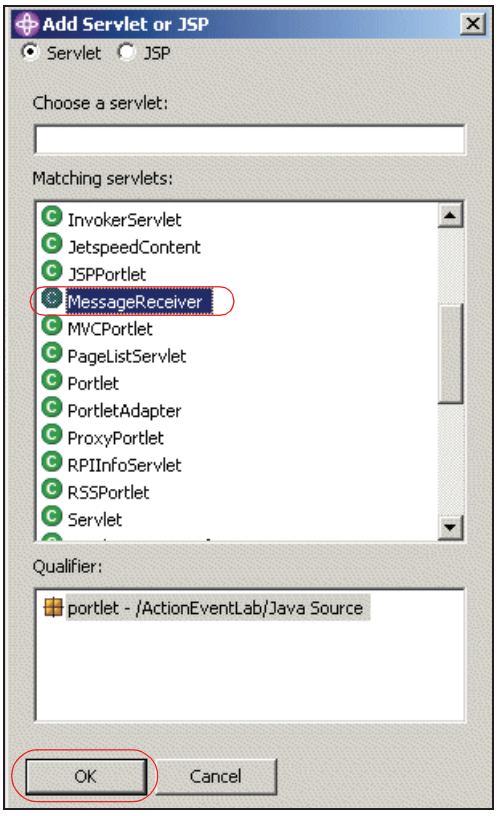


Figure 10-11 Choose MessageReceiver

e. To the right and below the Servlets list, you will see a table of URL mappings that is empty. Click the **Add** button.

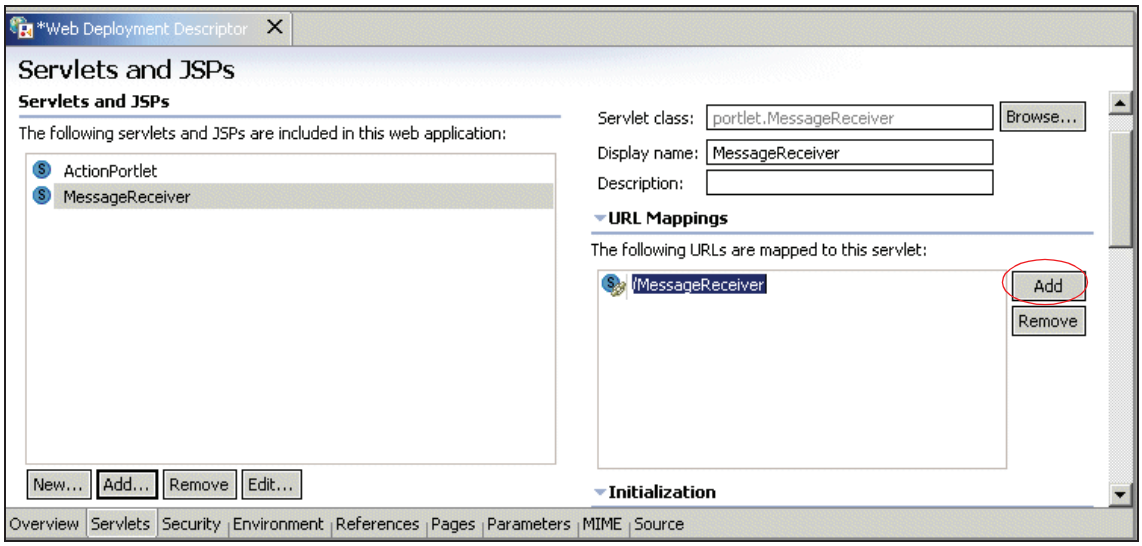


Figure 10-12 Adding a URL mapping

f. Select **MessageReceiver** and click **OK**.

g. Save and exit the file.

14. Update portlet.xml.

a. Open the portlet.xml file from the same location.

b. Select the **Add Portlet** button at the bottom of the window

c. Select **MessageReceiver** and click **OK**.

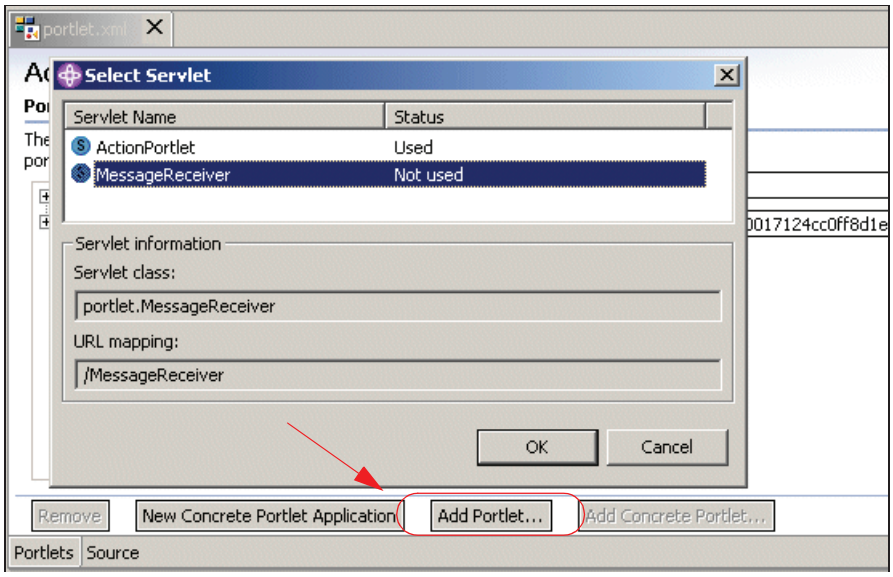


Figure 10-13 Adding the MessageReceiver portlet

d. You will now have a new entry called Portlet_2 under Portlet application. Click the **Portlet_2** entry. Change the display name to MessageReceiver.

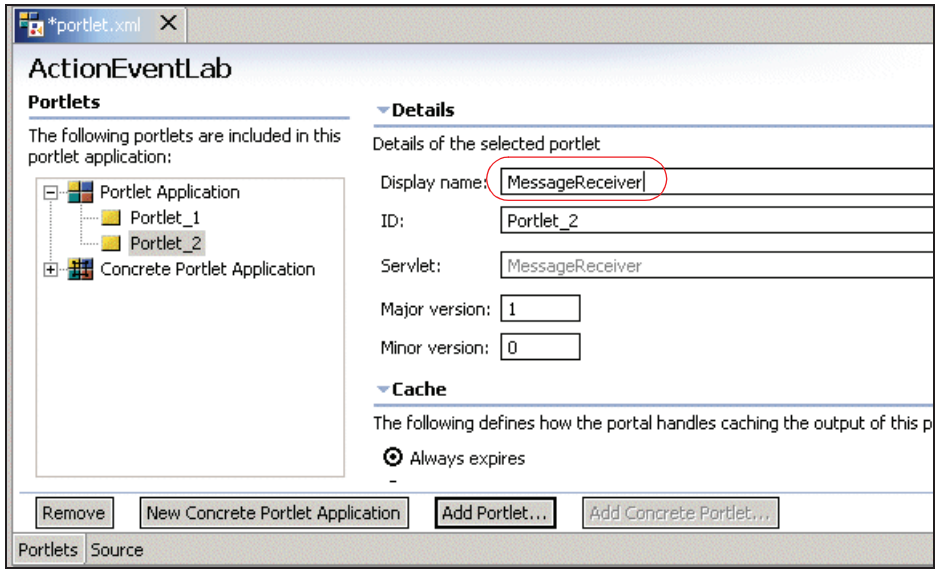


Figure 10-14 Renaming the portlet

e. Scroll down to the Markups heading and add PDA markup support to this portlet.

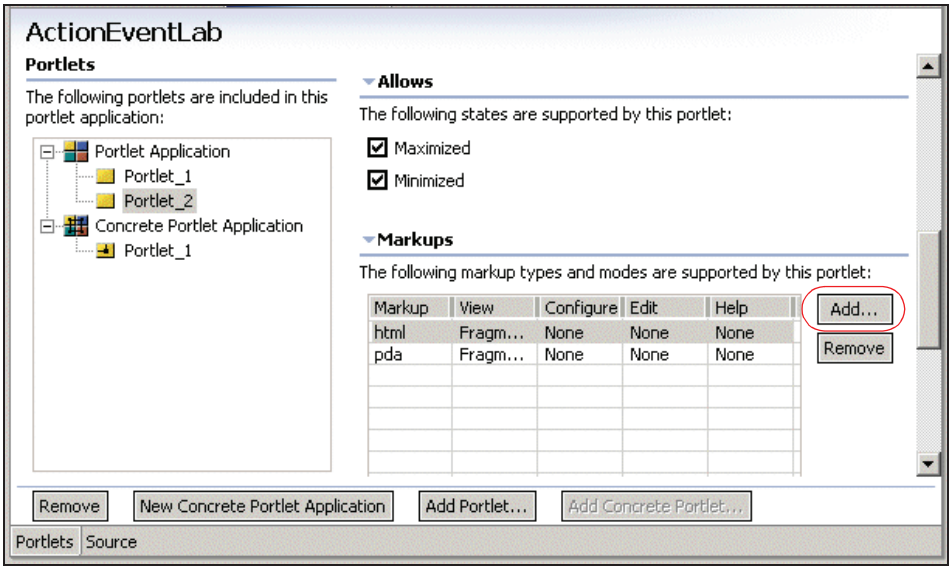


Figure 10-15 Adding PDA markup support

f. Select **Concrete Portlet Application** from the left pane, then click the **Add Concrete Portlet** button at the bottom right, and choose **Portlet_2**.

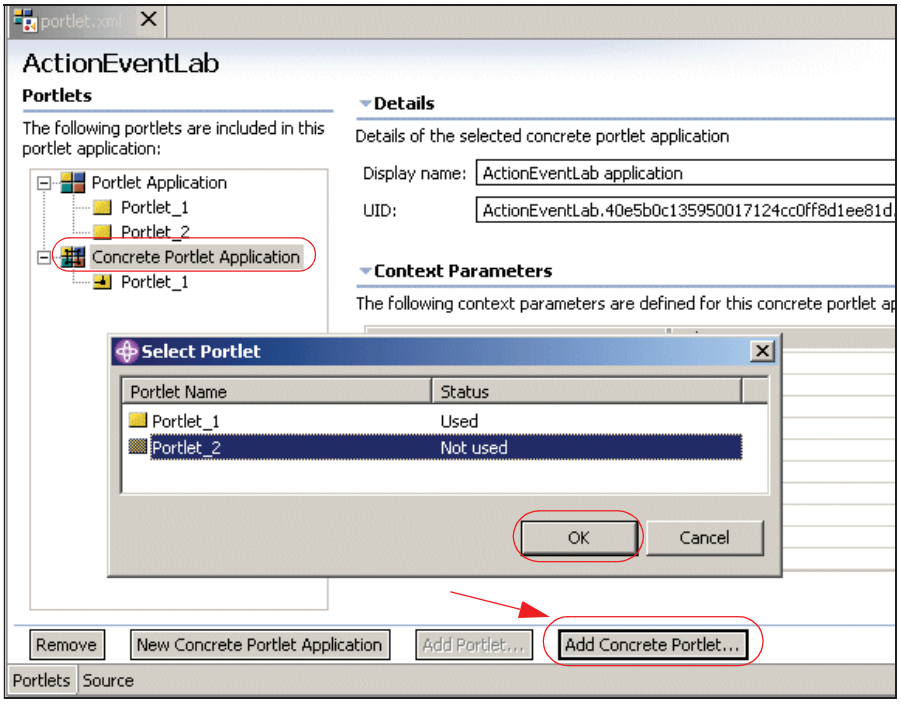


Figure 10-16 Adding a concrete portlet

g. Type in `MessageReceiver portlet` as the display name and title.

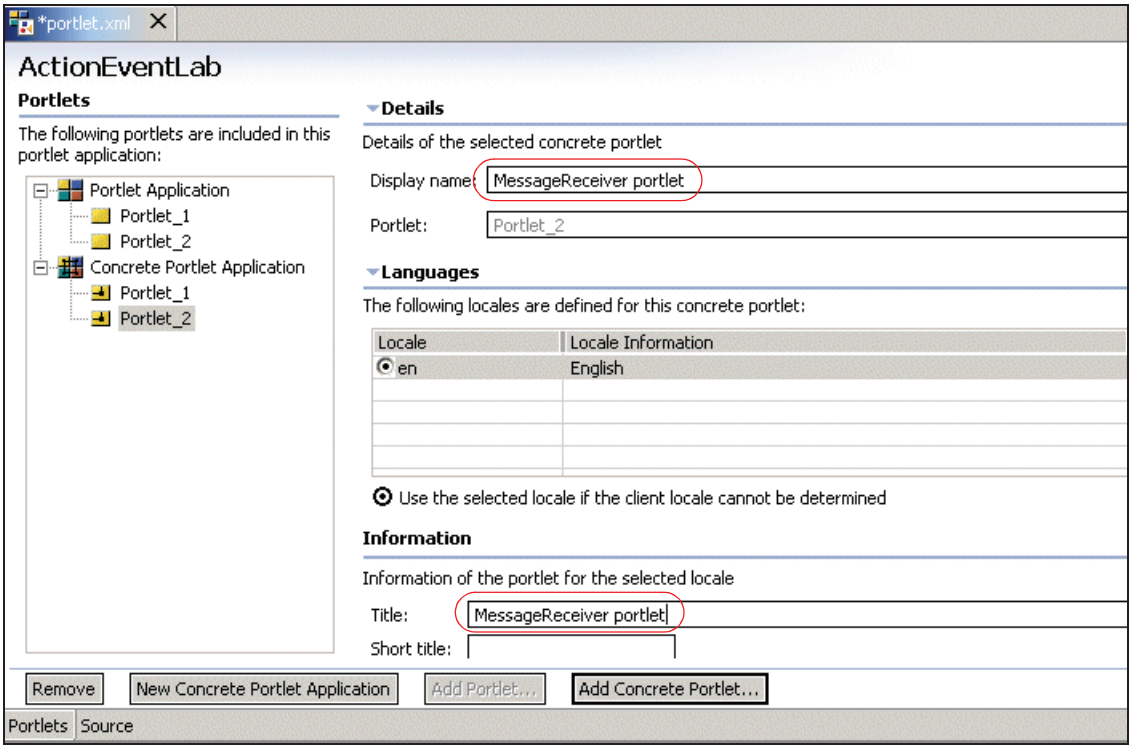


Figure 10-17 Renaming the concrete portlet

h. You have now added the MessageReceiver portlet into the portlet application and configured it with the appropriate display names and titles.

i. Save and exit the portlet.xml file.

10.4 Receiving a message

The next task will be to implement the logic within MessageReceiver portlet to listen for and act on PortletMessages.

In this section you will add a messageReceived method to the MessageReceiver portlet. Once the message is received, the JSP will be invoked to render the message in view mode. The message is passed in the request object.

Open the MessageReceiver.java class. Since you want to extend this portlet so that it listens for PortletMessages, update the code to add an import statement for the portlet event classes. In addition, update the code to implement the MessageListener interface.

Example 10-3 Adding MessageListener interface to MessageReceiver.java

```
package portlet;

//**************************************************************************
//*
//* MessageReceiver - A sample portlet based on PortletAdapter
//*
//**************************************************************************
import java.io.*;
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlet.event.*;

public class MessageReceiver extends PortletAdapter implements MessageListener
{
...
```

15. Add the messageReceived() method. This method implements the logic to receive the PortletMessage. In this example, you only need to check for messages of type DefaultPortletMessage, which is the type of message sent by ActionPortlet. Then the message is extracted via the getMessage() method, and you will set the text of this message into a portlet request as an attribute with the name PortletMessage.

Example 10-4 Adding the messageReceived method to MessageReceiver.java

```
...
public void messageReceived(MessageEvent event) throws PortletException {
   // Check whether message is a DefaultPortletMessage; if so, extract string
if (event.getMessage() instanceof DefaultPortletMessage) {
         DefaultPortletMessage defmsg = (DefaultPortletMessage)
                                                    event.getMessage();
         String message = defmsg.getMessage().substring(6);
         //Create a portlet request
         PortletRequest request = event.getRequest();
         //Create a session
         PortletSession session = request.getPortletSession();
         //Save value in the session
         session.setAttribute("PortletMessage", message);
   }
}
...
```

16. Save and close the MessageReceiver.java file.

10.5 Displaying the message in View mode

The View.jsp file will need to be updated to display the message to the user after it is received.

1. Open the ViewMessage.jsp file in the PDA directory.
2. Add the highlighted code to the JSP after the line ending with **"**`getPortletName()%>`**"**.

Example 10-5 ViewMessage.jsp

```
<%@ page contentType="text/html"%>

<jsp:useBean id="MessageReceiverBean" class="portlet.MessageReceiverBean"
scope="request" />

<B>This is <%=MessageReceiverBean.getPortletName()%></B>
<BR>
<% if (session.getAttribute("PortletMessage") == null) { %>
   <B>Ready to receive message...</B>
<% } else { %>
   <B>Received a message:</B>
   <B><%= session.getAttribute("PortletMessage") %></B>
<% } %>
<BR>
<B>Operating in View mode</B>
```

You can see that this is implemented through a JSP scriptlet. The Java code inside the scriptlet checks the value of the portlet request attribute PortletMessage. If null, no message has been received yet and it displays that it is ready to receive a message. If not null, the message is displayed with HTML/PDA markup.

3. Save and close the ViewMessage.jsp file.

10.6 Running the portlet application

In this section you will run the portlet application you have developed to send a message from the message sender portlet (ActionPortlet.java) to the message receiver portlet (MessageReceiver.java).

Follow these steps:

1. In the Navigator view, select the **ActionEventLab** project, right-click and select **Run on Server**. The server will be published and started. You will see the internal Web browser displays the two portlets.

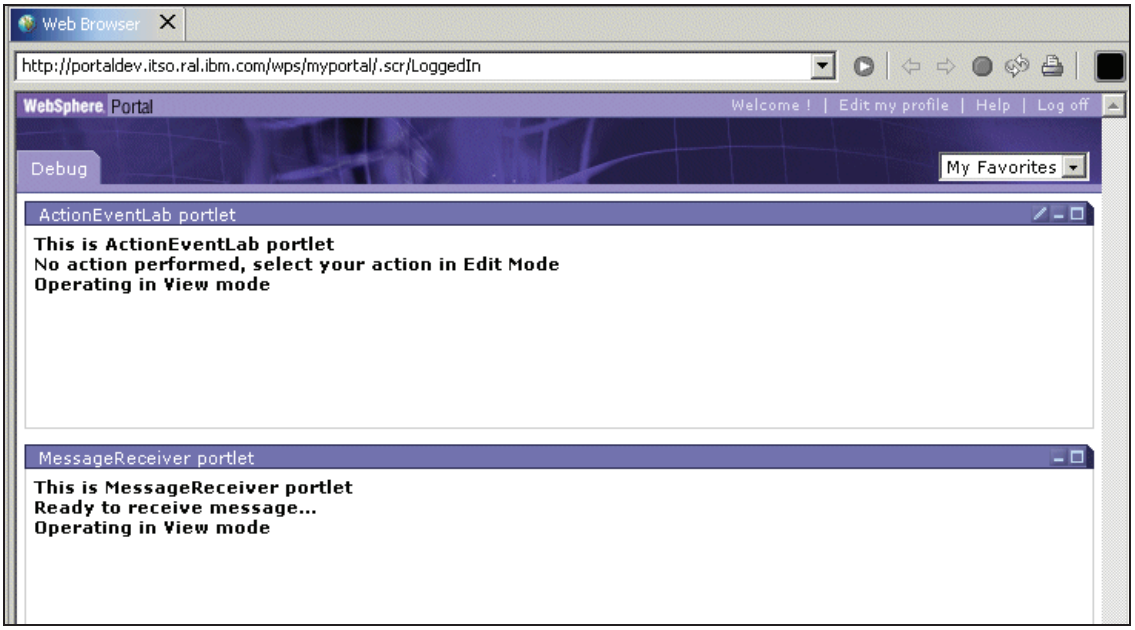


Figure 10-18 The running portlet in the Web browser

2. Open the Palm OS V5.2 Simulator, and use the Web browser to navigate to http://localhost/wps/myportal. Log in with user name and password of `wpsadmin`.

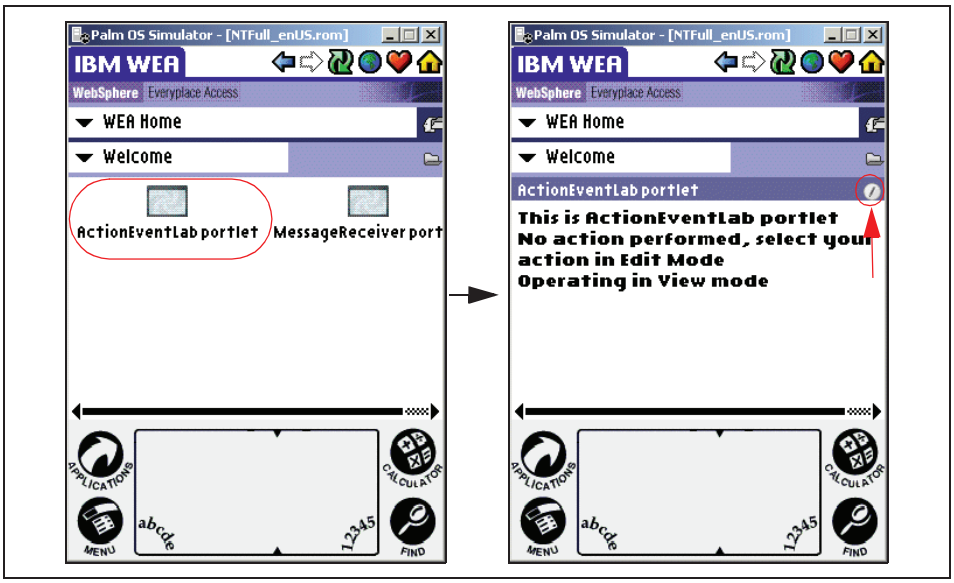


Figure 10-19 The ActionEvent portlet on the Palm Simulator

3. Click the **Edit** icon as highlighted in Figure 10-19 to enter edit mode.
4. You will see two buttons: one for performing the Red Action and sending the Red Action message to the MessageReceiver portlet, the other for performing the Blue Action and sending the Blue Action message to the MessageReceiver portlet. Click the **Red Action** button. This will send the Red Action message.
5. You will then be brought back to the portlet listing in icon form. Click the **Message Receiver** portlet to view the result of sending a portlet message.

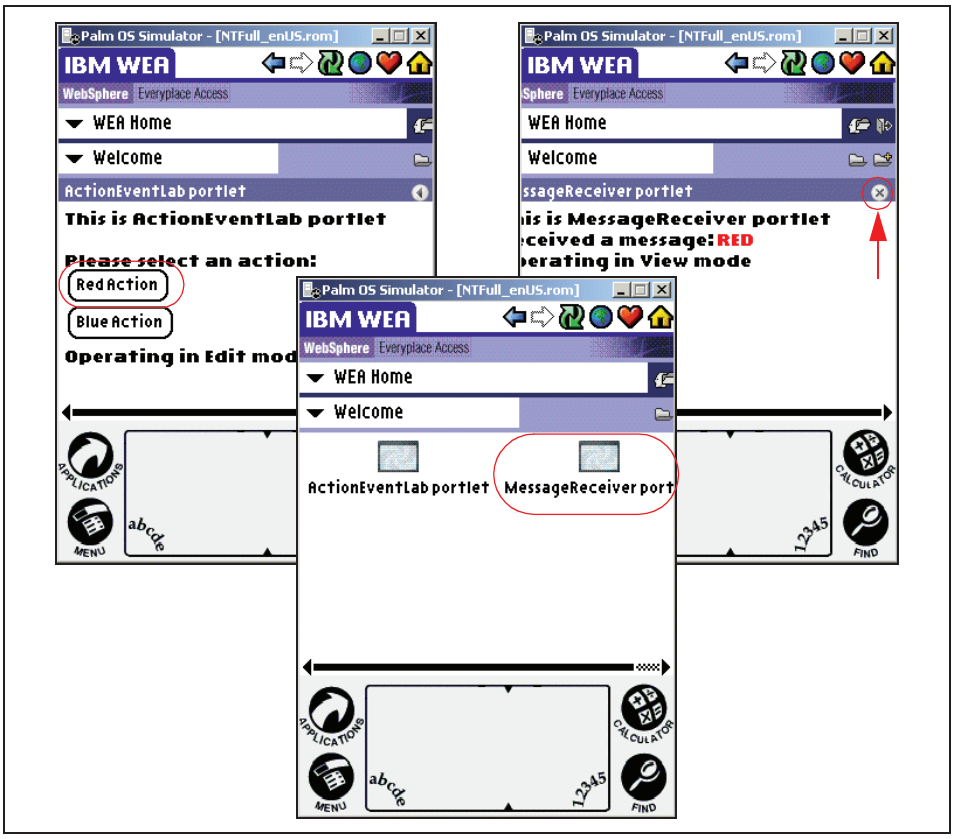


Figure 10-20 Choosing an action and viewing the result on the simulator

6. Exit the Message Receiver portlet by clicking the **X** icon in the title bar as shown in Figure 10-20.
7. Repeat the above steps, but this time select the **Blue Action** button. You will see that the Blue Action message was sent to the MessageReceiver portlet.

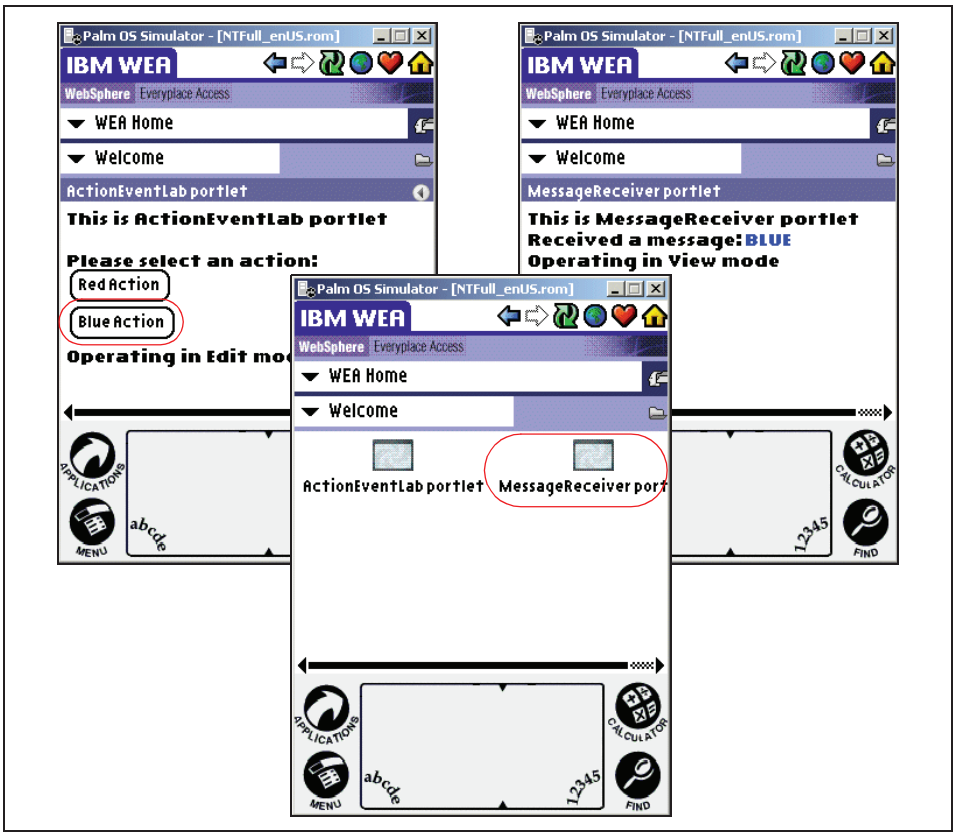


Figure 10-21 Viewing the result of the Blue Action

11

Portlet National Language Support (NLS)

When building portals of any kind, whether for your employees, customers, or business partners, it is important to ensure that your content can reach your intended audience in the form that they need it. This becomes especially important when your audience includes speakers of multiple languages. WebSphere Portal was designed with this requirement in mind. The portlet API has implemented explicit support for presenting translated content in multiple languages to users.

One way to accomplish this is to translate and maintain separate JSP and HTML pages within the folder structure of your portlet application. Another way is to use property bundles that contain key-value pairs for specific translated strings that you can use in your portlet. These property bundles can be accessed from your Java portlet code or via JSP tags from your JSPs. Either way, the Portal Server handles which translated content to present to the user, based on either default locale setting for the portal or by the locale preference stored in the user profile.

In this scenario, you will use the JSP tag library approach to present appropriate content to users based on the language setting in their user profile. You will build upon the portlet from the previous chapter by extending the JSP pages with tags to access NLS-translated content from property bundles and render the content in English and Spanish and show what happens when a language it not supported.

© Copyright IBM Corp. 2003. All rights reserved.

11.1 Overview

In this lab, NLS bundles will be created to support multiple languages. Once you have done this using WebSphere Studio, the JSP that delivers markup content in view mode for portlet MessageReceiver will be enhanced to access the NLS bundles.

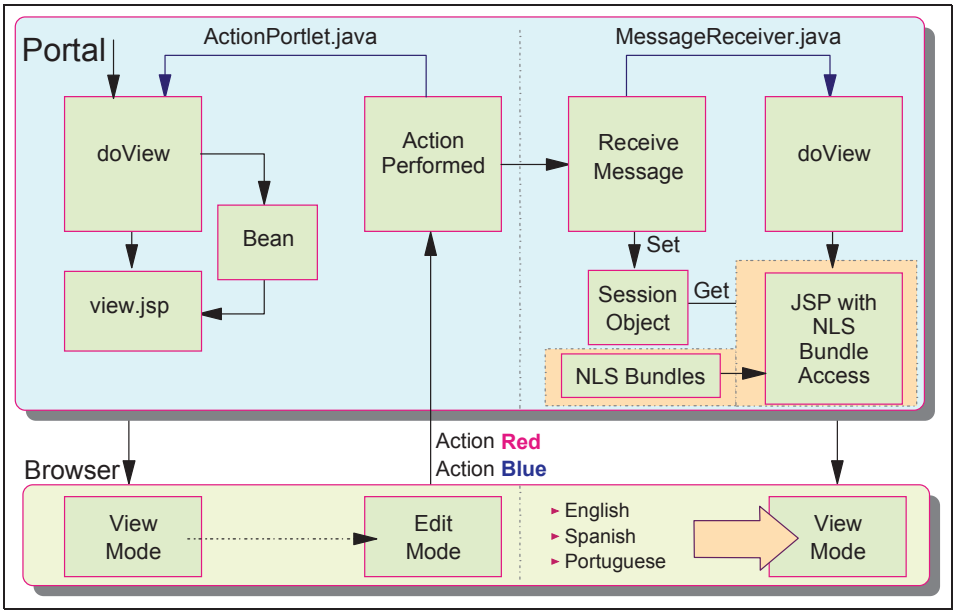


Figure 11-1 National language support (NLS) scenario

The resource bundle is accessed via the PortletContext object's getText method. You will need to provide the following:

- Bundle Base Name: The first parameter indicates the base name of the resource bundle. The name includes the path relative to the classes directory. The name does not specify the locale suffix or the properties file type. If the base file name cannot be found, or the key is not present in the properties file, a PortletException will be thrown.
- Key: This parameter maps to a key value in the properties file. If the key is not found, a PortletException is thrown.

In addition the locale is used by the Portal to select the proper language bundle. However, you cannot set this value when invoking NLS bundles from JSPs.

11.2 Creating the NLS bundles

In this section, you will build upon your previous portlet project. The portlet application will be enhanced to support NLS. Follow these steps:

1. Before you start this lab, make sure the test environment server has been stopped and all perspectives, editors, and Web browsers have been closed.
2. If not already running, start the IBM WebSphere Studio Application Developer by clicking **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.0**.
3. In the ActionEventLab project, create a new folder with the name `nls` to store the resource bundles. The following resource bundles will be imported into this folder:
   - NLSLab.properties (default)
   - NLSLab_en.properties (English)
   - NLSLab_es.properties (Spanish)
   - NLSLab_pt_BR.properties (Brazilian Portuguese)

   Select your **ActionEventLab/webApplication/WEB-INF/classes** folder.

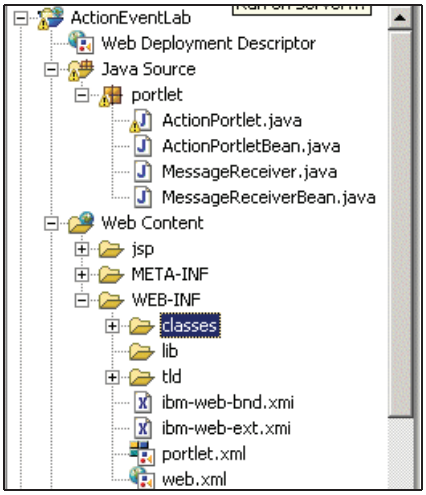


Figure 11-2 Select classes to create nls folder

4. Right-click the **classes** folder and select **New -> Folder**.

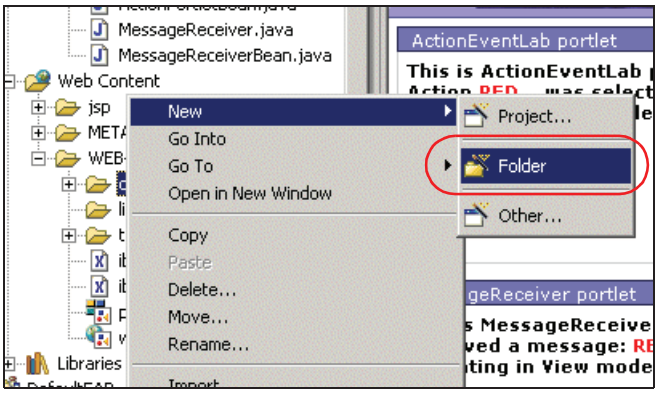


Figure 11-3 Select a new folder

5. Type `nls` in the Folder name field, then click **Finish**.

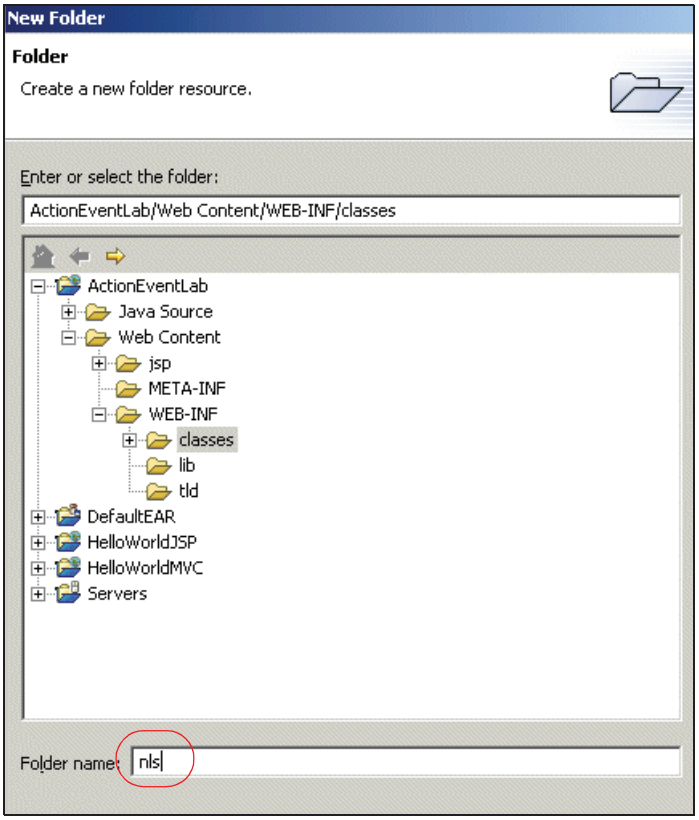


Figure 11-4 Adding a folder to a project

6. Your directory structure should now look like the one shown in Figure 11-5.

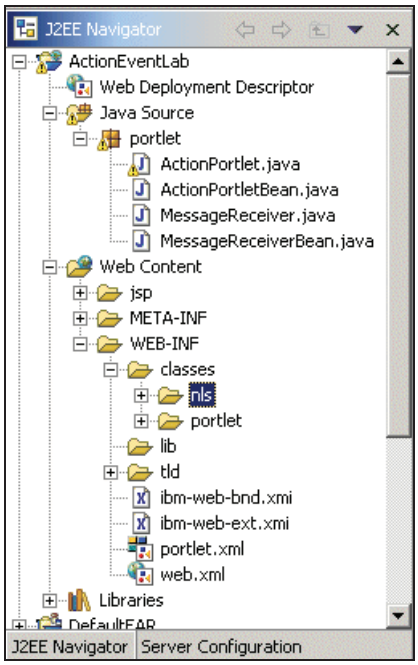


Figure 11-5 Viewing the added folder

7. Select the **ActionEventLab/webApplication/WEB-INF/classes/nls**.
8. Click **File -> Import**.
9. Select **File System** and click **Next**. Click **Finish** to import.

   Note: Sample code can be obtained from additional materials of this redbook.
10. View the files in the nls folder by double-clicking them. Notice how they are structured.

Example 11-1 NLSLab.properties (default)

```
readystatus=Ready to receive message
receivedstatus=Received a message
viewmode=Operating in View mode
```

Example 11-2 NLSLab_en.properties (English)

```
readystatus=Ready to receive message
receivedstatus=Received a message
viewmode=Operating in View mode
```

Example 11-3 NLSLab_es.properties (Spanish)

```
readystatus=Listo para recibir mensaje
receivedstatus=Mensaje recibido
viewmode=Operando en modo de visualizacion
```

11. When done, select **File -> Save All** to save all your changes to the project.

11.3 Accessing NLS bundles from JSPs

In this section you will update the JSP ViewMessage.jsp to display NLS content based on the locale value (English or Spanish). Follow these steps:

1. Open the ViewMessage.jsp file from the pda directory in WebSphere Studio Site Developer.
2. In this JSP you have the following text with static information:
   - Ready to receive message
   - Received a message
   - Operating in view mode

Example 11-4 Static information

```
<%@ page contentType="text/html"%>

<jsp:useBean id="MessageReceiverBean" class="portlet.MessageReceiverBean"
scope="request" />

<B>This is <%=MessageReceiverBean.getPortletName()%></B>
<BR>
<% if (session.getAttribute("PortletMessage") == null) { %>
   <B>Ready to receive message...</B>
<% } else { %>
   <B>Received a message:</B>
   <B><%= session.getAttribute("PortletMessage") %></B>
<% } %>
<BR>
<B>Operating in View mode</B>
```

3. Add logic to display messages in the proper language. Updates to this JSP are highlighted in Example 11-5 but if you want to save time, copy and paste from the additional materials of this redbook. See Appendix I, “Additional material” on page 1271.

Example 11-5 Information-specific languages

```
<%@ page contentType="text/html"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
```

```
<jsp:useBean id="MessageReceiverBean" class="portlet.MessageReceiverBean"
scope="request" />

<B>This is <%=MessageReceiverBean.getPortletName()%></B>
<BR>
<% if (session.getAttribute("PortletMessage") == null) { %>
   <B><portletAPI:text key="readystatus" bundle="nls.NLSLab">Ready to
receive message...</portletAPI:text></B>
<% } else { %>
   <B><B><portletAPI:text key="receivedstatus" bundle="nls.NLSLab">Received
a message:</portletAPI:text></B>
   <B><%= session.getAttribute("PortletMessage") %></B>
<% } %>
<BR>
<B><portletAPI:text key="viewmode" bundle="nls.NLSLab">Operating in view
mode</portletAPI:text></B>
```

4. Select **File -> Save All** to save all your changes to the project.

11.4 Running the portlet

1. Close any open Web browser windows.
2. Right-click **ActionEventsLab** and click **Run on Server**.

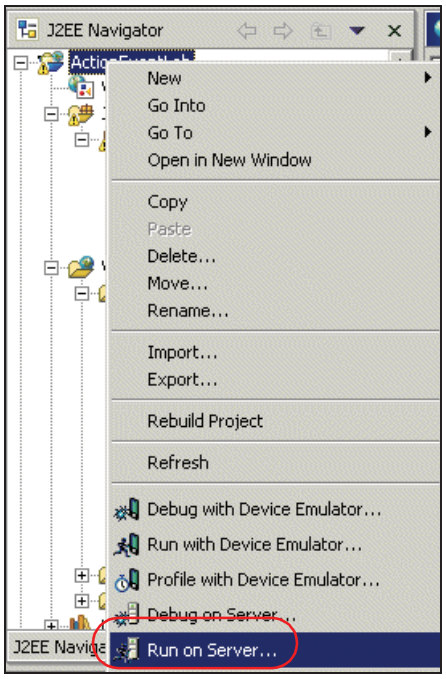


Figure 11-6 Running the portlet application

3. Open the Palm Simulator Web browser and navigate to http://localhost/wps/myportal and authenticate as user ID `wpsadmin` and password `wpsadmin`.
4. The MessageReceiver portlet will now display its markup using NLS. Note that it will still display in English.



Figure 11-7 Viewing the portlet and editing your profile

5. Click the **Edit my profile** link at the top of the screen as shown in Figure 11-7.
6. You will need to enter a first name and last name (in this example, we entered both as `wps` to fill the blanks). Select **Spanish** as the language and click **Continue**. You will need to click **Continue** again on the next page to confirm your changes.

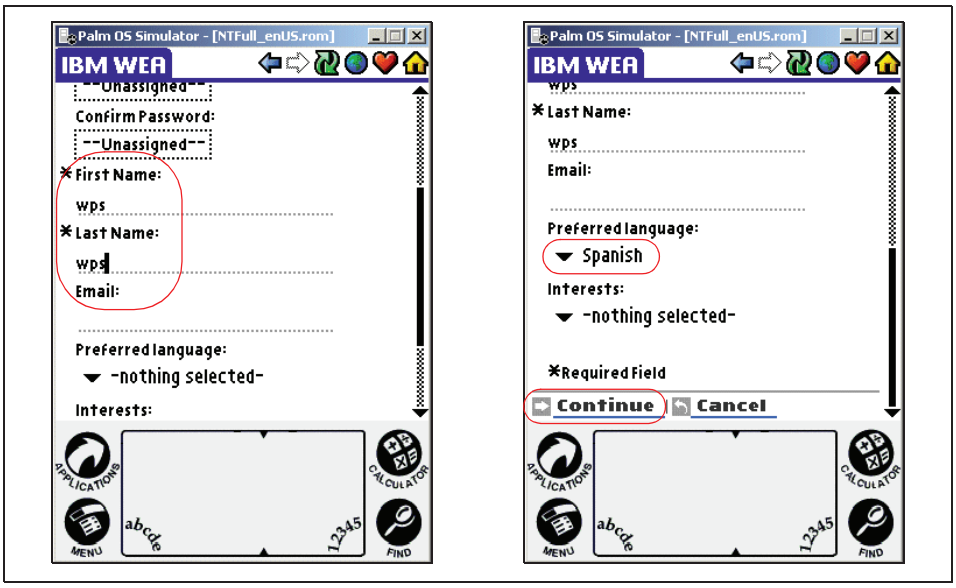


Figure 11-8 Changing your name and default language

7. Now, clicking the **MessageReceiver** portlet will display its content en español.

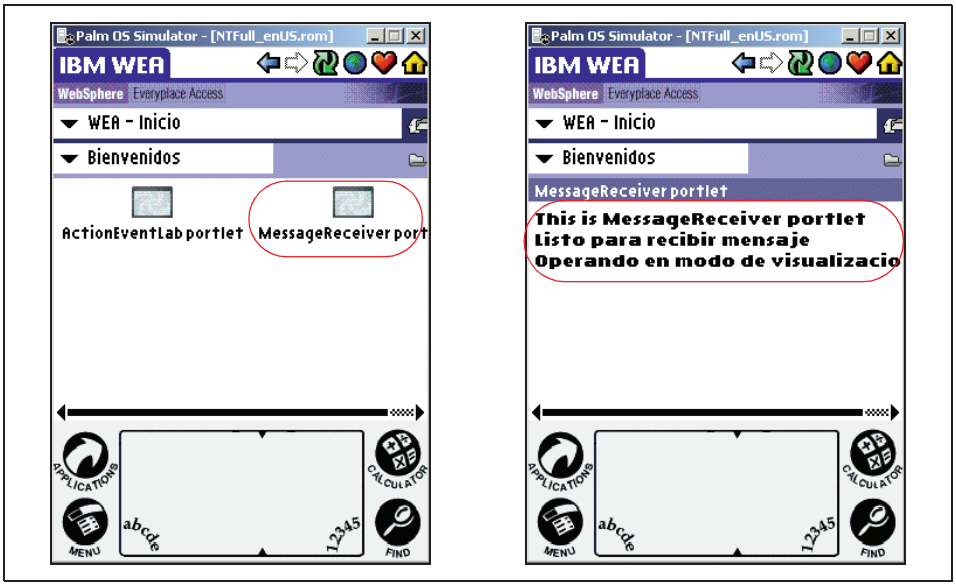


Figure 11-9 Viewing the MessageReceiver portlet in Spanish

8. Edit your profile again. This time, select **French** as your language. Note that there is no National Language Support for French included in your portlets.



Figure 11-10 Changing your language to French

9. Although the WebSphere Portal header has support for the French language, the text in your portlet does not. This is why the header displays certain information in the French language, but your portlets do not. You will want to reset the language back to English before you exit this scenario.



Figure 11-11 Since French is not supported, English (the default) is displayed

12

Portlet Credential Vault

This scenario provides an exercise for creating a sample portlet application that uses the Credential Vault to log in to interact with a protected back-end system. You will create, deploy and run one portlet application that will demonstrate the active Credential Vault method. This exercise will allow you to understand the techniques used to develop portlets using the Credential Vault provided by WebSphere Portal.

In this chapter you will find the following topics:

- The active and the passive Credential Vault authentication methods
- Creating and running a portlet using the active Credential Vault on the Palm OS V5.2 Simulator

© Copyright IBM Corp. 2003. All rights reserved.

12.1 Overview

In this scenario, you will learn how to implement the active Credential Vault authorization method, but it will also discuss the passive Credential Vault method.

Credentials

Portlets running on WebSphere Portal may need to access resources that require some form of authentication by using appropriate credentials. Examples of credentials that may be used are user ID/password combinations, SSL client certificates, and private keys. For resources protected by the Portal, WebSphere uses CORBA credentials and an encrypted LTPA cookie to authenticate users.

However, many back-end systems require their own form of authentication such as the SSL client certificates or private keys mentioned above. The Credential Vault provides several classes that allow the portal to authenticate on the user's behalf and then use the already authenticated connections.

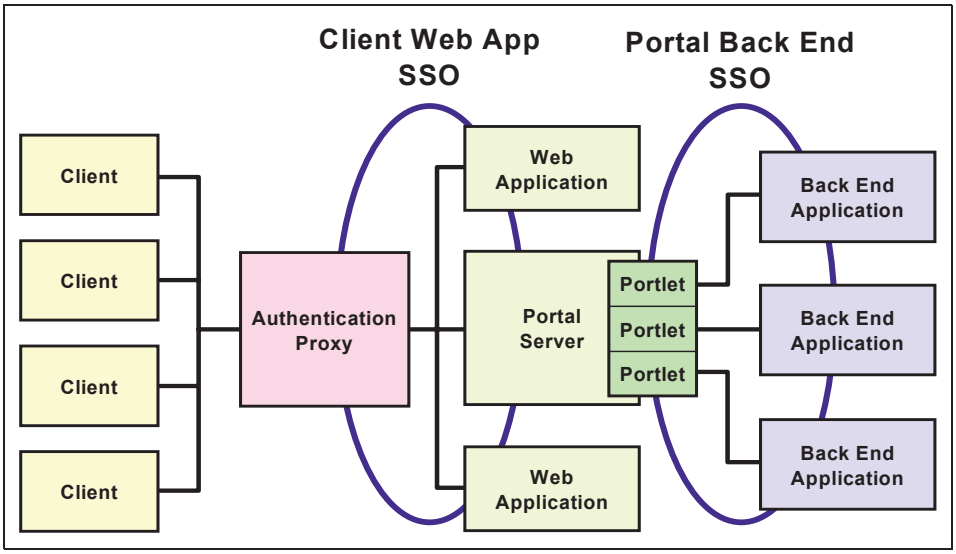


Figure 12-1 Credential Vault in action

The Credential Vault provides this functionality, and portlets can use it through the Credential Vault PortletService.

Components of Credential Vault organization

The following components are available in the Credential Vault:

- **Vault segments**
  - Can be created and configured only by Portal administrators
  - Contain one or more vault slots
- **Vault Slots**
  - Are “drawers” where portlets store and retrieve a user’s credentials
  - Each slot holds one credential
  - Linked to a *resource* in a *vault implementation*
    - An implementation is the place where a user’s credentials are stored, such as in WebSphere Portal’s default database vault
    - A resource within the implementation is the back-end resource that requires its own authentication
  - There are four different types of slots:
    - Administrative slots
    - System slots
    - Shared slots
    - Portlet private slots

Credentials objects

WebSphere Portal differentiates between passive and active credential objects:

- **Passive credential objects** are containers for the credential’s secret. Portlets that use passive credentials need to extract the secret out of the credential and do all the authentication communication with the back end itself. The following classes are included with the current release of WebSphere Portal for authentication purposes (additional credential objects can easily be registered and used as well):
  - SimplePassive
  - UserPasswordPassive
  - JaasSubjectPassive (Java Authentication and Authorization Service)
- **Active credential objects** hide the credential's secret from the portlet. There is no way of extracting it out of the credential. In return, active credential objects offer methods that take care of all the authentication. The following classes are provided for authentication purposes using the active Credential Vault:
  - HttpBasicAuth
  - HttpFormBasedAuth
  - JavaMail

– LtpaToken
– SiteMinderToken
– WebSeal Token

From a security point of view, portlets never get in touch with the credential secrets and thus there is no risk that a portlet could violate any security rules, for example by storing the secret on the portlet session. While there might not always be an appropriate active credential class available, it is the preferred type of credential object to use. The scenario in this chapter shows how to use the active credential method.

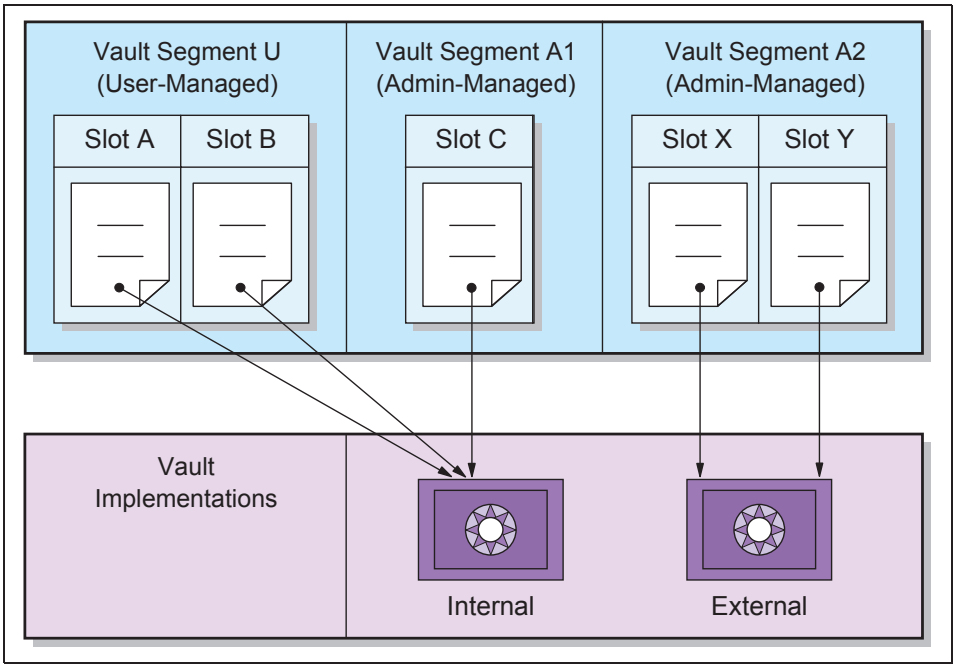


Figure 12-2 The vault segments, slots and vault implementation

Credential Vault scenario

In these scenarios, you will create a sample portlet based on the Basic portlet type using the Portlet wizard. You will then implement a portlet that uses the Credential Vault to interact with back-end resources (in this case, a protected JSP). The scenario shows how the active Credential Vault works. The sample code that you will need to run each of these scenarios can be obtained from additional materials of this redbook. See Appendix I, “Additional material” on page 1271.

In this scenario, the protected back-end resource is a servlet and requires user ID/password credentials to log in to the Web application. The servlet application has been secured with HTTP Basic Authentication.

This scenario illustrates the following:

- How the Credential Vault (with active credentials) is used
- How to create a private vault slot to store credentials
- How to store credentials
- How to retrieve credentials
- How to log in to the Web application
- How to retrieve the Web application content in the portlet's view mode

In this exercise, you will be using active credentials to access a Web application using HTTP Basic Authentication.

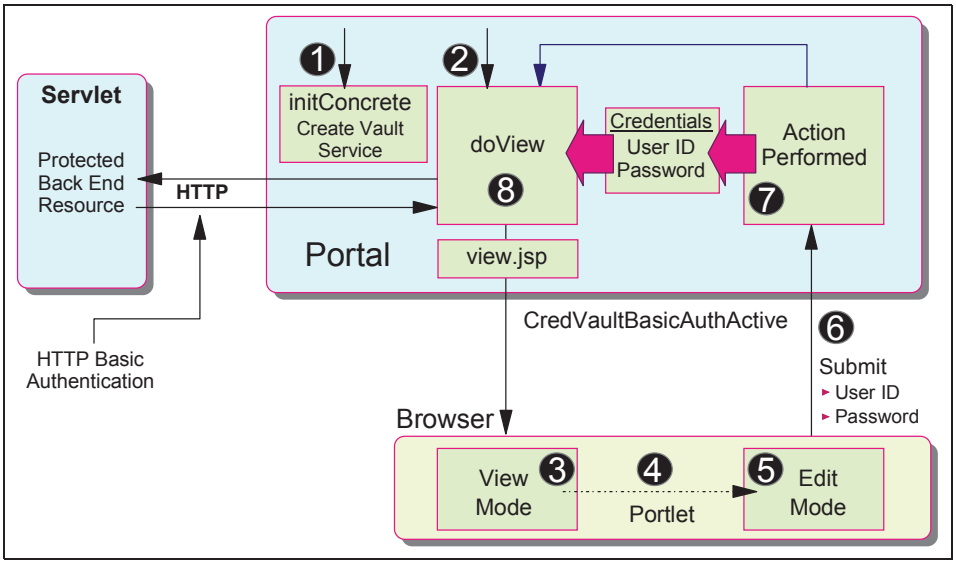


Figure 12-3 Credential Vault lab scenario - active credentials

The flow can be summarized as follows:

1. The initConcrete() method creates the vault service at initialization time.
2. Assuming that there are no actions and messages pending, the doView() is invoked.
3. The first time in view mode there are no credentials. Therefore the user is asked to switch to edit mode and enter a user ID and password.
4. The user switches to edit mode.
5. The user enters and submits credentials, an action event.

6. The actionPerformed() method is invoked.
7. The action is processed in the actionPerformed() by storing the credentials in the proper Credential Vault slot.
8. The doView() method is invoked again and this time it finds the credentials. The credentials are used to establish the HTTP secure connection using basic authentication. The servlet results are displayed on the user's screen in view mode.

12.2 Creating the portlet project

In this section, you will create a portlet project with the name CredVaultBasicAuthACTIVE. The portlet will be created based on the Basic portlet type and will demonstrate the use of the active credential. You will run this portlet in the test environment to see how it works.

Follow these steps:

1. If not already open, Start the IBM WebSphere Studio Application Developer by clicking **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.0**.

Note: If Site Developer is already open, stop the test environment by clicking the **Server** tab in the tasks area, right-clicking **Test Environment**, and clicking **Stop**. This ensures that the next time you run a project, the project files will be published and the server refreshed with the most current portlet project configuration.

2. Select **File -> New -> Other**.

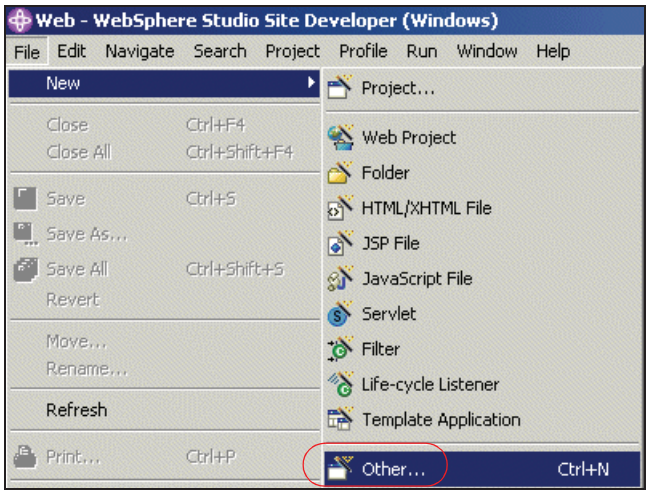


Figure 12-4 Starting a new project

3. Select **Portlet Development -> Portlet Application Project**.

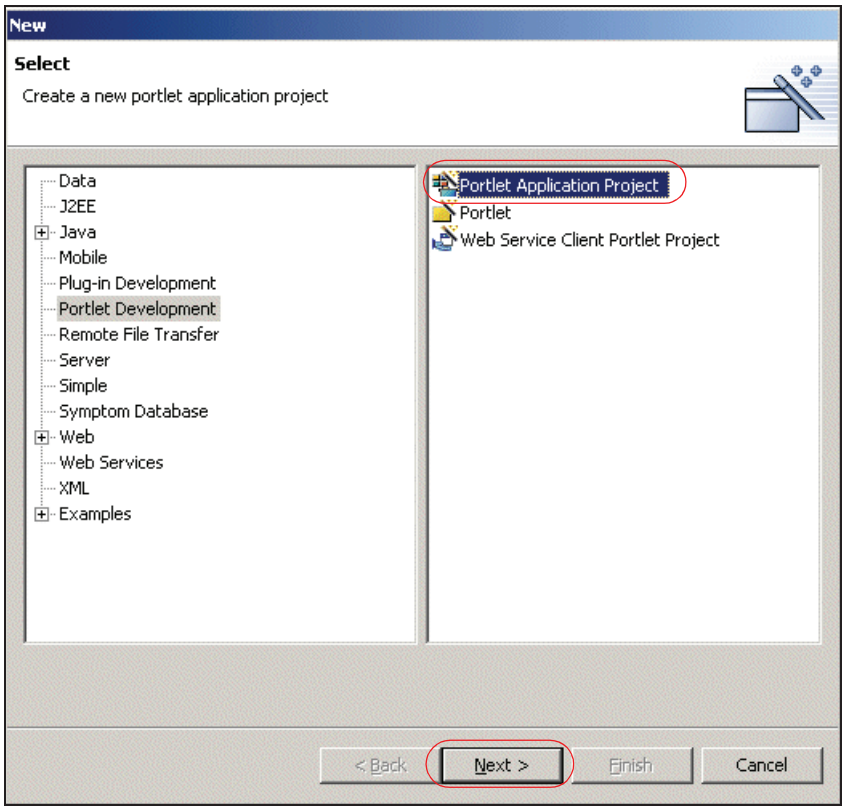


Figure 12-5 Selecting the project type

4. Click **Next**.
5. In the Define the Portlet Project window, enter the following information:
   - Project Name: CredVaultBasicAuthACTIVE

     Note: it is important that ACTIVE is capitalized in this scenario for the example WAR file to import correctly.
   - Enterprise Application project: Existing/DefaultEAR
   - Context root: CredVaultBasicAuthACTIVE

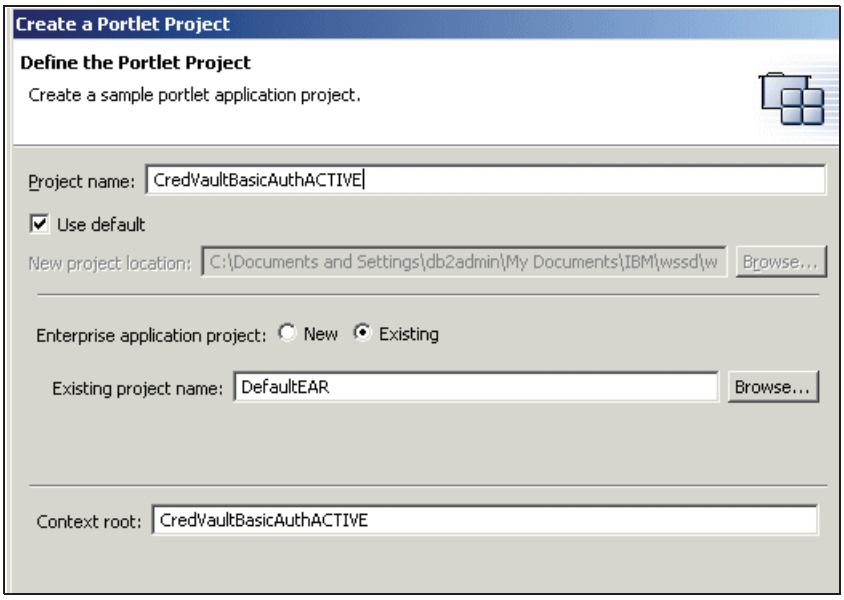


Figure 12-6 Defining the project

6. Click **Next**.
7. Select **Basic Portlet** for the type of portlet to be generated.
8. Click **Next**.
9. Change the Portlet class name to `CredVaultBasicAuthACTIVE`. Also select the **pda** markup so you can run this portlet on a PDA device.

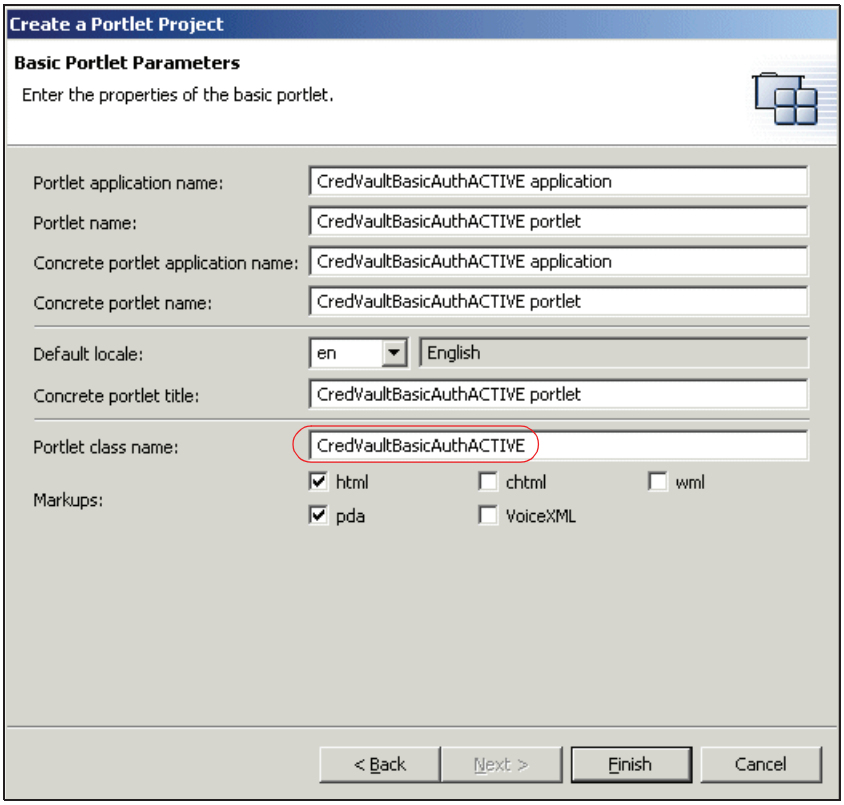


Figure 12-7 Changing the portlet class name

10. Click **Finish**.
11. If you receive the Repair Server Configuration window, click **OK**.

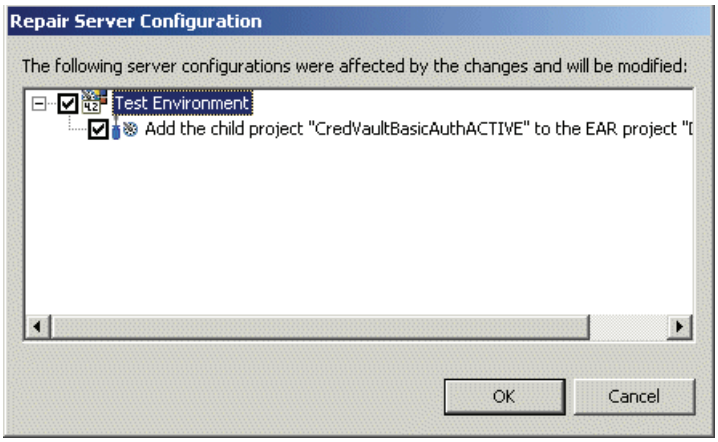


Figure 12-8 Server configuration window

12. The new project CredVaultBasicAuthACTIVE will show in the Navigator list.
13. If you have any previous portlets in the DefaultEAR project, remove them at this time from the project:
    a. Open your **DefaultEAR/META-INF** folder.
    b. Double-click the application.xml file.
    c. Select **Modules**.
    d. Remove all WAR modules except for CredVaultBasicAuthACTIVE.
    e. Click **File -> Save All**.

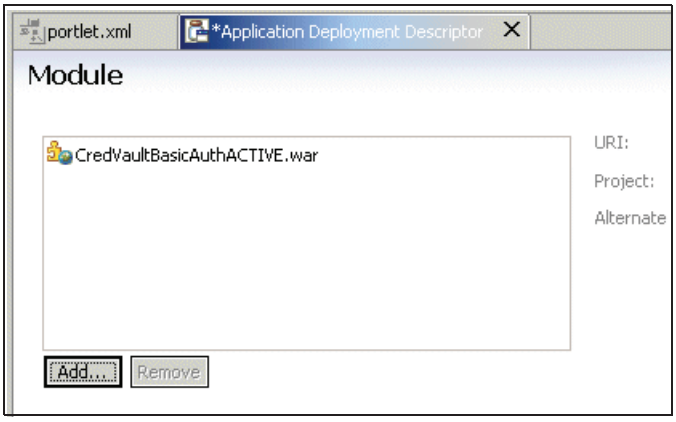


Figure 12-9 Removing WAR modules

    f. If you receive the Repair Server Configuration window, click **OK**.

12.3 Updating the active Credential Vault project

In this lab, as in the first, a WAR file is provided to update the original project created by the wizard. It will update the same files as the passive credential portlet project.

1. Import the WAR file into the workspace. Click **File -> Import**.
2. Select **WAR file** and click **Next**.
3. In the Import Resources from a WAR file window, enter the following information:
   - WAR File: Select the file **CredVaultBasicAuthACTIVE.war**
   - Web project: Select **Existing**, then select **CredVaultBasicAuthACTIVE**
   - Context root: Select **/CredVaultBasicAuthACTIVE**
   - In Options, select **Overwrite existing resources without warning**

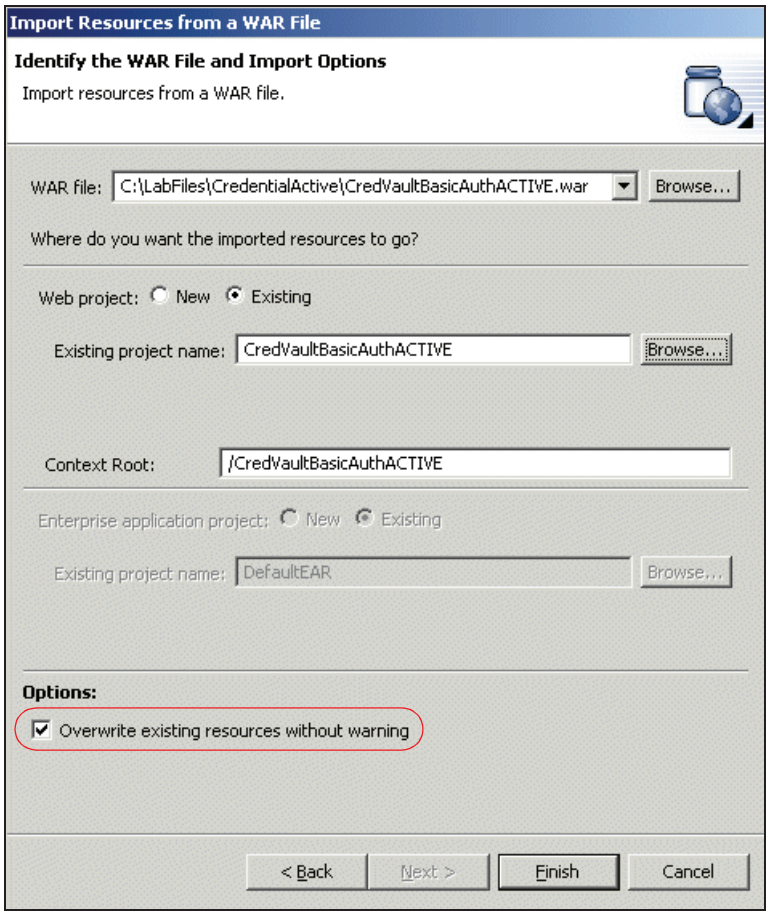


Figure 12-10 Overwriting existing files

4. Click **Finish**.

12.4 Reviewing the portlet code

In this section you will observe updates made to the original portlet code:

1. To edit CredVaultBasicAuthACTIVE.java, double-click the file name.
2. In the initConcrete method, the Credential Vault service is created using the PortletService interface. It is done in the same way for both passive and active credentials.

Example 12-1 CredVaultBasicAuthACTIVE.java - Creating the Credential Vault service

```
...
   private static CredentialVaultService credVaultService = null;
   /*********************************************************************
   * initConcrete - This method initializes data member credVaultService
   *********************************************************************/
   public void initConcrete(PortletSettings settings) throws
UnavailableException {
      super.initConcrete(settings);
      try {
         credVaultService = (CredentialVaultService)

getPortletConfig().getContext().getService(CredentialVaultService.class);
      }
      catch (Exception e){
         return;
      }
   }
...
```

3. As you view the portlet, the doView method attempts to see that a credential slot is present. The first time the portlet application is run, it won't be. This results in the error message "`Credential not found. Please set it in the edit mode!`"

Example 12-2 CredVaultBasicAuthACTIVE.java - credential slot

```
...
String slotId = (String)
request.getData().getAttribute("PrivateSlotSamplePortletSlotID");
      if (slotId == null) {
         writer.println(
            "<h2>Credential not found. Please set it in the edit mode!
</h2>");
         return;
      }
...
```

4. When the user enters the credentials (user ID and password) and clicks submit in edit mode, the action is checked in the actionPerfomed method to create the slot and store the credentials (user ID and password).

   Note: The credentials used for this example are a user-entered ID and password. In a real-world implementation, any type of credential might be used (for example, certificates, digital signatures, keys, etc.). The purpose of active credentials is to put the control of access to the back-end resource into the hands of the administrator rather than the user. Active credentials are also more secure than passive credentials, and as such are more favored.

Example 12-3 CredVaultBasicAuthACTIVE.java - actionPerformed - create slot and store credential

```
...
   try{
          PortletData data = portletRequest.getData();
                String slotId = (String) data.getAttribute("PrivateSlotSamplePortletSlotID");

             // create slot if necessary
               if(slotId==null) {
                String resourceName = "POP3MailApp";
                ObjectID segmentID = service.getDefaultUserVaultSegmentId();

             Map descripMap = new Hashtable();
                  Map keywordMap = new Hashtable();
                  int secretType =
service.SECRET_TYPE_USERID_STRING_PASSWORD_STRING;
                  boolean active = true;
                  boolean portletPrivate = true;

             //create the slot
                  CredentialSlotConfig slot= service.createSlot(resourceName,
segmentID,
                       descripMap, keywordMap, secretType, active,
portletPrivate,
                       portletRequest);

             slotId=slot.getSlotId();
             data.setAttribute("PrivateSlotSamplePortletSlotID",
slot.getSlotId());
                  data.store();
               }

             // store credentials in vault
               service.setCredentialSecretUserPassword(slotId, userID,
                   password.toCharArray(), portletRequest);
         }
         catch(Exception e){
            e.printStackTrace(System.out);
         }
   }
...
```

5. After the credentials have been entered, the doView method is called again. This time, the credential slot is found, and the user is authenticated. If this process completes successfully, the protected back-end resource is displayed to the user.

Example 12-4 CredVaultBasicAuthPassive.java - creating the authorization header

```
...
try {
          String host = request.getServerName();
            String port = String.valueOf(request.getServerPort());
            String path = request.getContextPath();
            URL urlSpec = new URL("http://" + host + ":" + port + path +
"/TreasurePage");
          connection = credential.getAuthenticatedConnection(urlSpec);

          connection.connect();

          String responseMessage = connection.getResponseMessage();
          int responseCode = connection.getResponseCode();

          // Were we successful?
          if (HttpURLConnection.HTTP_OK == responseCode) {
             writer.println("<P>Successfully connected to " + urlSpec +
"!</P>");
          } else {
             writer.println("<P>Unable to successfully connect to " + urlSpec +
", HTTP Response Code = " +
             responseCode + ", HTTP Response Message = \"" + responseMessage +
"\"</P>");
          }
...
```

12.5 Running the portlet project

In this section, you will run the portlet application to verify that it functions correctly:

1. Right-click the **CredVaultBasicAuthACTIVE** project in the Navigator pane and click **Run on Server**.

If the **Server Selection** window is displayed, click **Finish** to use the test environment.

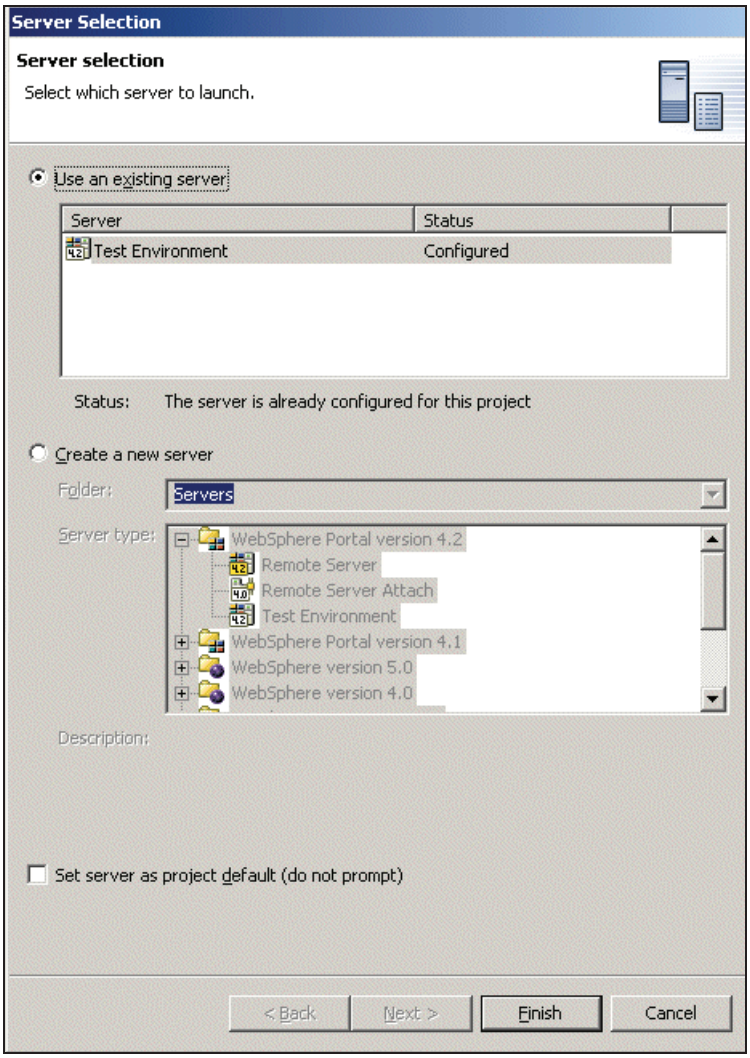


Figure 12-11 Server selection window

2. The portlet project will load in Site Developer's integrated Web browser. To display it on the Palm Simulator, open the Simulator, open a Web browser, and navigate to http://localhost/wps/myportal. Log in with a user name and password of `wpsadmin` to view the portlet.
3. The portlet will execute the initConcrete method to initialize the Credential Vault Service, and the doView method. Since there are no credentials yet, you receive the message "`Credential not found. Please set it in edit mode.`"

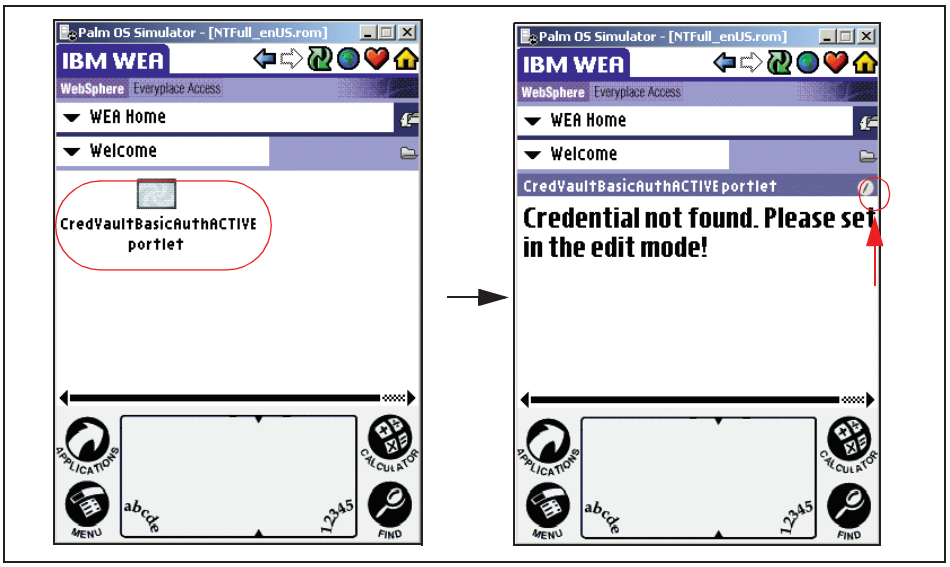


Figure 12-12 Viewing the portlet

4. Click the **Edit mode** icon to enter edit mode.
5. In edit mode, enter the following authentication information:
   - UserID: `user1`
   - Password: `password1`

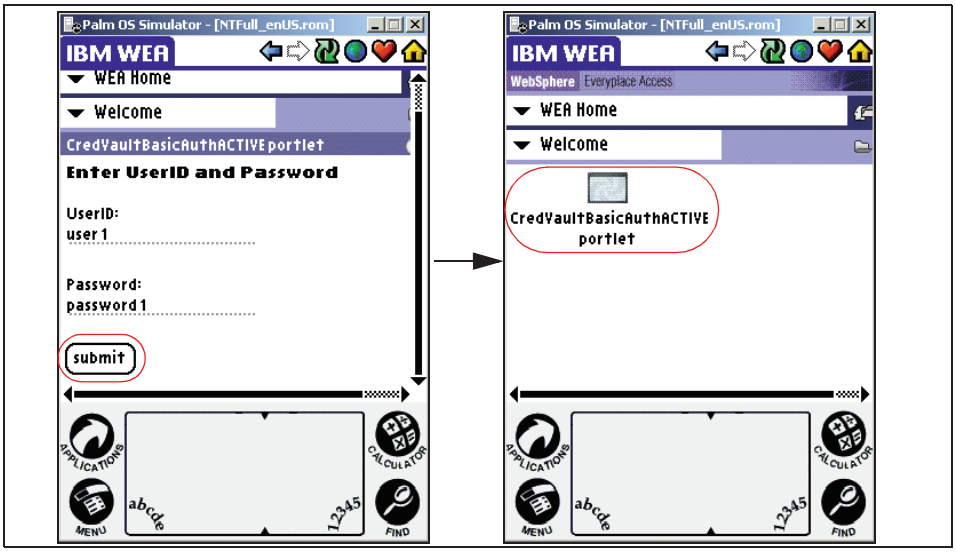


Figure 12-13 Entering credentials

6. Submit the action. This will call the actionPerformed method in the CredVaultBasicAuthACTIVE class and return you to the icon listing of the portlets. Click the **Portlet** icon to return to the view mode, which will call the doView class showing the contents of the Treasure Servlet.



Figure 12-14 Viewing the protected resource

7. Click **File->Exit** to close WebSphere Studio Site Developer.

13

Offline Portal Content

Due to cost, limitations on technology, and sometimes lack of infrastructure, mobile users cannot always be connected to the enterprise. However, they still need access to information that is readily available on their enterprise Portal. This chapter discusses the offline browsing and form submission capabilities provided by WebSphere Everyplace Access.

The topics discussed in this chapter include:

- Overview of Offline Portal Content
- How it works
- Limitations
- Configuration
- Usage
- Development guidelines

Important: To ensure you have the latest code, you should visit the WebSphere Everyplace Access support Web site at `http://www.ibm.com/software/pervasive/ws_everyplace_access/support/` to download the latest fix packs and code fixes. Apply these fixes prior to using the samples within this redbook.

© Copyright IBM Corp. 2003. All rights reserved.

13.1 Overview

In a mobile work environment where being online is not always possible, accessing information from the Enterprise Portal can be difficult. Everyplace Access helps overcome this difficulty by providing users with the ability to view, browse, and interact with their Portal Content while offline. This is achieved through the Offline Portal Content functionality of Everyplace Access. In addition to viewing and browsing pages while offline, Offline Portal Content also provides users with the ability to complete and submit forms while offline. The form data is stored on the client device and then submitted to the server the next time the user connects to the Portal.

13.2 How it works

There is an offline page that comes installed with Everyplace Access. Users select portlets that they wish to see while offline and then modify the access rights on these portlets to make them viewable on this page. All portlets to be viewed offline need to support PDA markup. When the page is synchronized using the offline Portal pages functionality of the Everyplace Access Client, the content of this page is downloaded to the user's handheld device. This is achieved in Everyplace Access by a servlet (called WebCache) that accesses each link on the offline page and rewrites the link and pages referenced by the link. The link is modified such that it references a cached page (the rewritten page) as opposed to a page sitting on the server. This rewrite, of both the links and pages referenced by the link, is done recursively until the defined link-depth is reached. The link-depth can be defined by the Administrator on the Offline Browsing Administration portlet. Additionally, each user can define a custom link-depth using the Offline Browsing Configuration.

Using the Everyplace Access Client, users can view the offline page cached on the client device while disconnected. When a link within the offline page is clicked, the requested page is pulled from the cache on the client device instead of the server. If the content of the offline page is changed online on the server, the user should refresh the cached copy of their offline page by running the synchronization process.

With offline forms, the process of making the forms viewable while offline is the same as described above. However, while offline, users can complete and submit a form. The details of the completed form will be stored on the client device until the user connects and synchronizes with the Portal. Upon synchronization, the form is submitted to the server and a confirmation is then sent to the user as if the submission occurred in real time. If the connection between the client device and the server is lost before the form is submitted during synchronization, the

form will be resubmitted during the next synchronization. Likewise, if the connection is lost before the user receives confirmation, the server will store the confirmation and send it to the user during the next synchronization.

13.3 Limitations

There are several limitations on Offline Portal Content and Forms that users should be aware of:

- The Offline Portal Content functionality was designed for static information. As such, only static content can be added to the offline page. Adding dynamically generated content may not yield the same result. This applies to offline forms as well. Furthermore, the pages a user browses to navigate to the offline form cannot contain dynamically generated links.
- Users can only submit one form per page.
- Only the POST method of form submission is supported.
- Even though Everyplace Access will make every attempt to ensure the form is submitted, there is no guarantee that the form will be submitted only once.

13.4 Configuration

Offline Portal Content must be configured before it can be used. There are 3 components to configuring Offline Portal Content: Server Configuration, Administrator Configuration and User Configuration.

13.4.1 Server configuration

Offline Portal browsing requires a function called *URI Addressability* to be turned on. By default, this function is not turned on during the install of Everyplace Access. To turn it on, do the following:

1. Using a text editor, open the following file for editing:

   <WAS Home>\lib\app\config\services\ConfigService.properties

2. Locate the property “use.requestId” and change its value from “true” to “false”.
3. Stop and restart the Portal Server using the WebSphere Administration Console.

13.4.2 Administrator configuration

Before users can start using Offline Portal Content, the Administrator will have to:

- Define the users that have access to Offline Portal Content. Refer to Chapter 2, “Administration” on page 25 for details on creating and maintaining users.
- Define the base Portal URL. The URL should take the following format:

  `http://<WEA_hostname>/<portal_base_uri>/<customized_page>`

  where:

  - `<WEA_Hostname>` is the fully qualified host name of the server where the Everyplace Access Core Services are installed.
  - `<portal_base_uri>` is the base URI defined for WebSphere Portal.
  - `<customized_page>` is the customized page defined for WebSphere Portal.

  For example:

  `http://mka0kkwd.itso.ral.ibm.com/wps/myportal`

  To define the base Portal URL:

  a. Log in as `wpsadmin` if you have not already done so.

  b. Select **WEA Home -> Administration**

  c. Enter the Portal URL in the Offline Browsing Administration portlet.

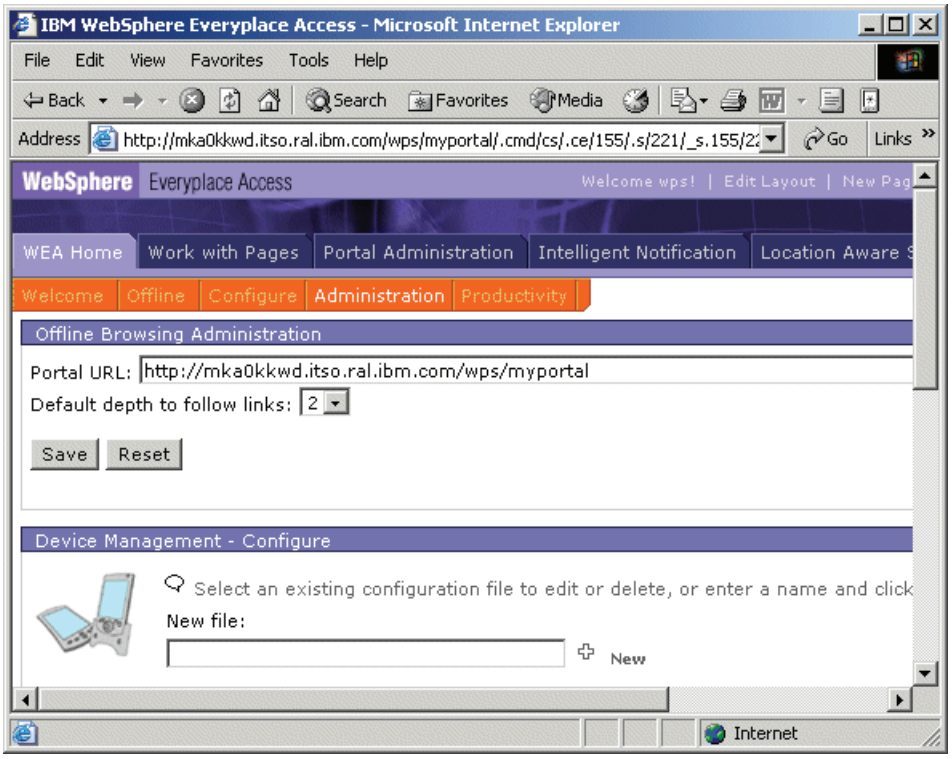


Figure 13-1 Administration configuration

- Define the default link-depth, which is the maximum depth with which to traverse links for portlets to be viewed offline. A link-depth of 0 means that only the offline page will be retrieved and cached during synchronization. A link-depth of 1 means that the offline page and the pages referenced by the links on this page will be retrieved and cached during synchronization. A link-depth of up to 5 can be selected.

  To define the default link-depth:

  a. Log in as `wpsadmin` if you have not already done so.

  b. Select **WEA Home -> Administration**

  c. Enter the default depth to follow links in the Offline Browsing Administration portlet. See Figure 13-1.

13.4.3 User configuration

Some configuration and customization are required by users before Offline Portal Content can be used.

- The maximum depth with which to traverse links for portlets to be viewed offline should be defined by the user. This will override the default link-depth set by the administrator. To define the link-depth:

  a. Log in as the user.

  b. Select **WEA Home -> Configure**.

  c. Enter the depth to follow links in the Offline Browsing Configuration portlet.

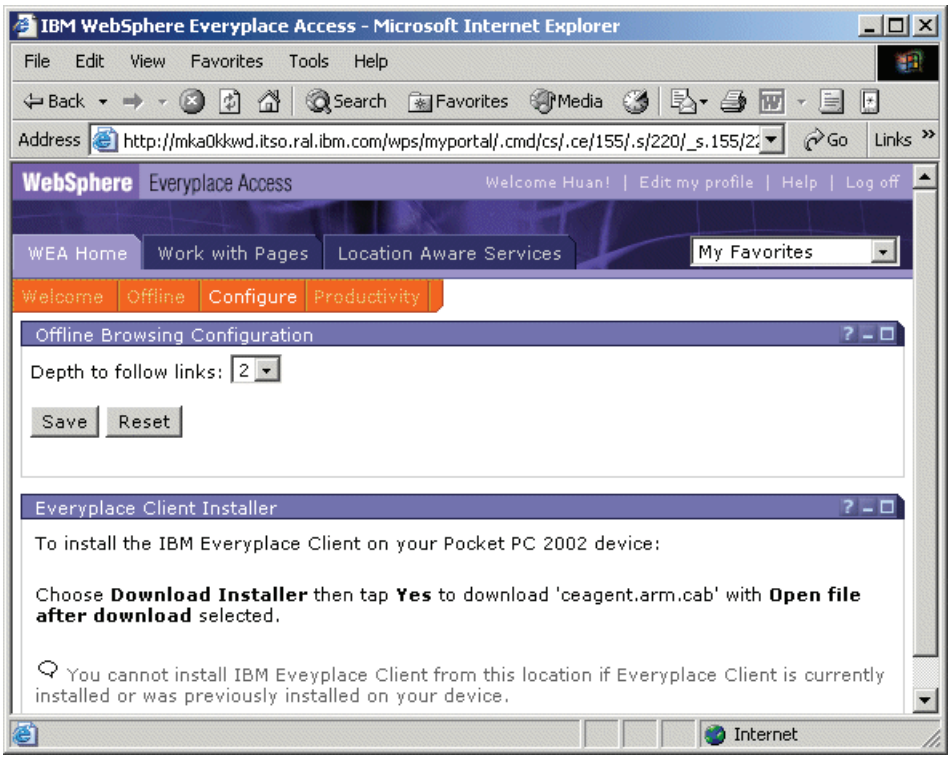


Figure 13-2 User configuration

- The portlets the user wishes to view offline need to be added to the offline Portal page. Assuming the portlets have already been created and the user has been granted permission to view these portlets, perform the following steps to add portlets to the offline Portal page:

  a. Log in as the user.

b. Select **Work with Pages -> Edit My Pages**.

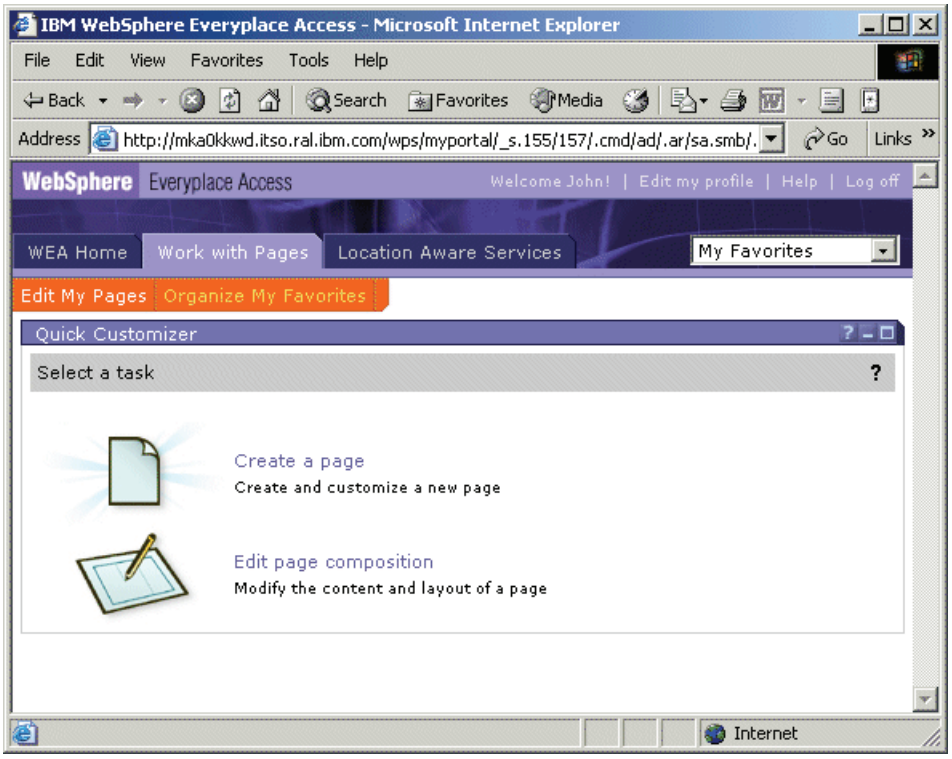


Figure 13-3 Edit My Pages

c. Select **Edit page composition**.

d. Under Available places, select **WEA Home** and under Pages in the place, select **Offline**.

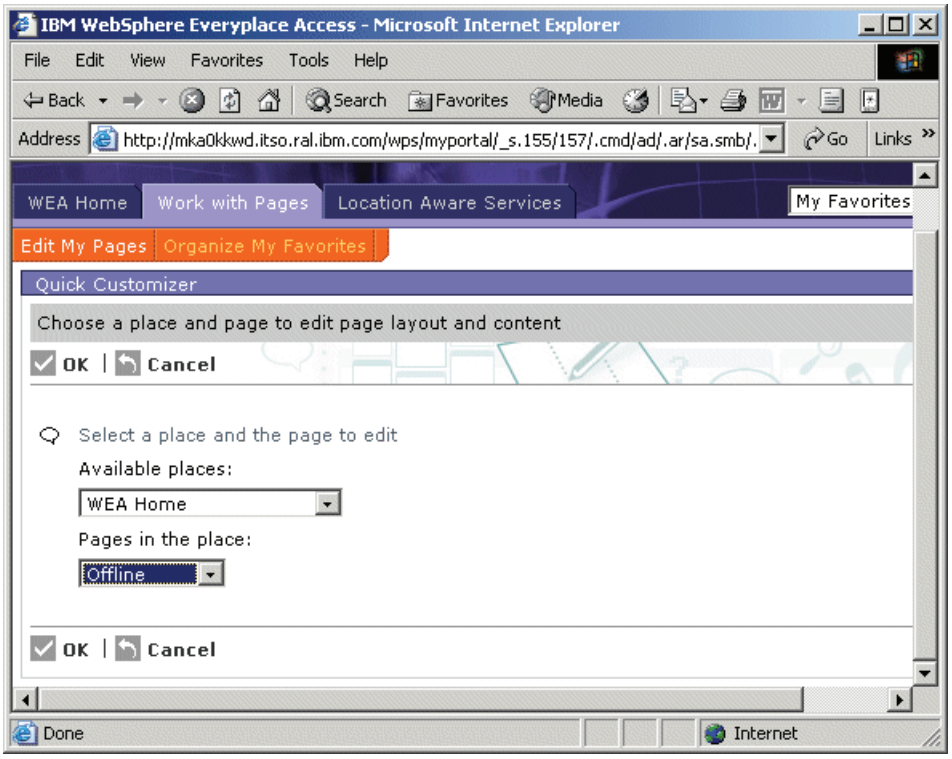


Figure 13-4 Edit page composition

e. Click **OK**.

f. You will be presented with the layout of your offline Portal page.

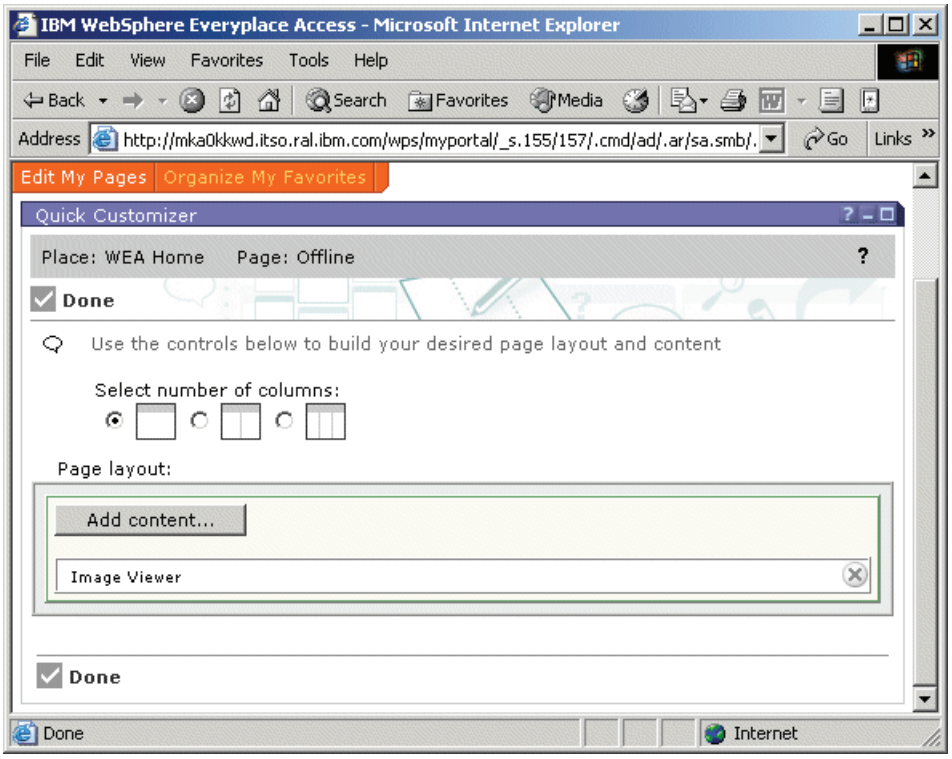


Figure 13-5 Offline page layout

g. Click **Add content**.

h. Select portlets you wish to add to your offline Portal page.

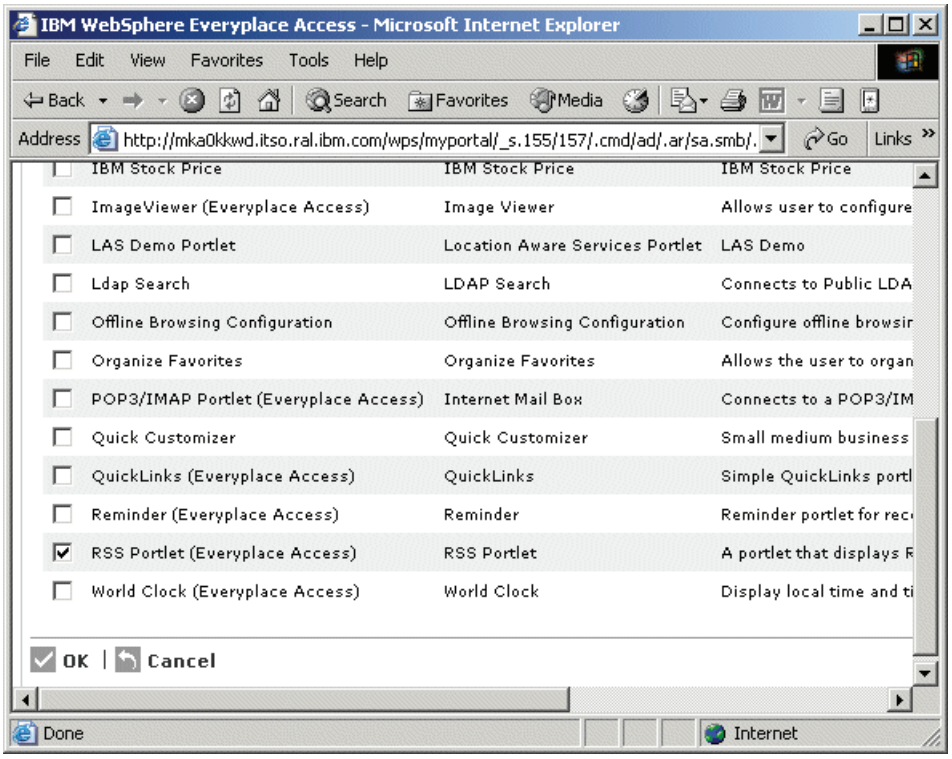


Figure 13-6 Select Portlets to add to offline page

i. Once you've selected the portlets, click **OK**.

j. You will be returned to the layout page of your offline Portal page. Click **Done**.

k. To verify your changes have taken effect, select **WEA Home -> Offline**.

l. The portlets you added should be visible on the offline page.

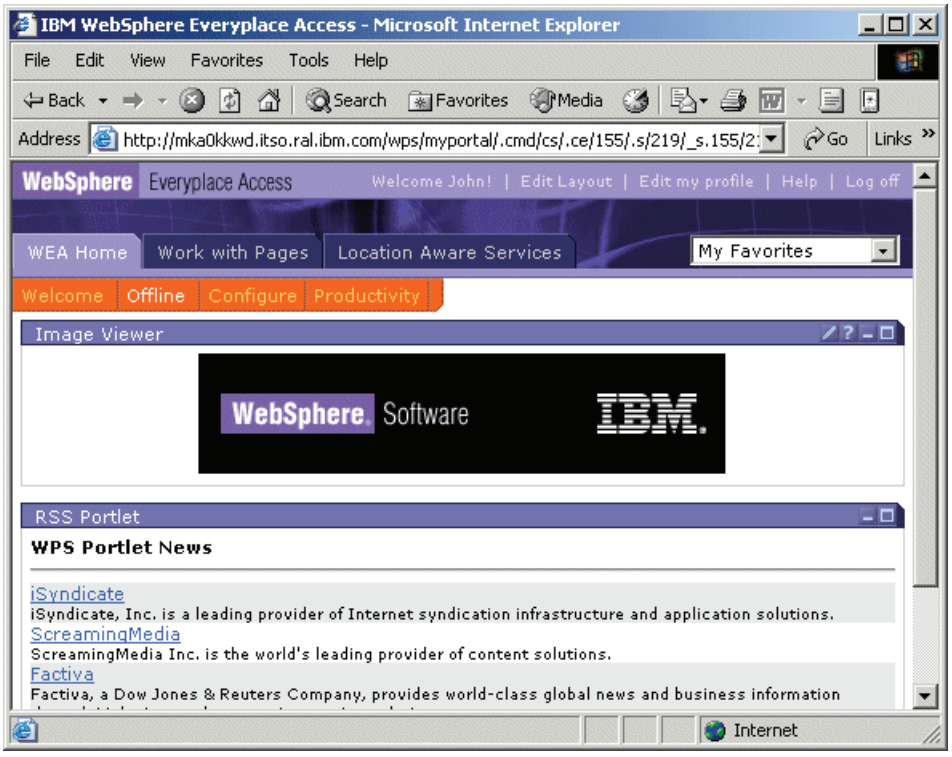


Figure 13-7 Offline Portal page

13.5 Usage

Once the offline Portal page has been configured on the server, it is ready to be downloaded to the client device and viewed offline.

13.5.1 Pocket PC devices

The instructions for using Offline Portal Content on Pocket PC devices are as follows:

1. The client needs to be configured to point to the Offline Portal Server in which offline Portal pages exist:
   a. Launch Everyplace Client on your Pocket PC. Select **Start -> Everyplace Client**.
   b. If prompted, enter your Portal user name and password.

c. The Everyplace Client Home page will appear. Select **Home** to open the drop-down list. Select **My Settings**.

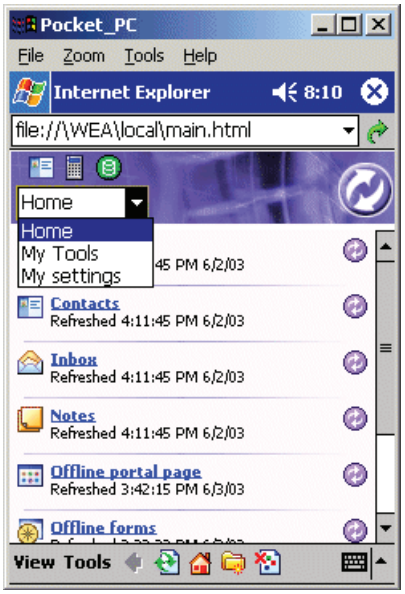


Figure 13-8 Everyplace Client Home - Select My settings

d. On the **My Settings** page, select **Network Profiles**.

e. Under Offline Portal Content server, enter the fully qualified server name of your Offline Portal Content server and then select the checkmark in the top-right corner of the Pocket PC browser window.



Figure 13-9 Enter Offline Portal Content server

2. Go back to the Home page by selecting **Home** from the drop-down list. Refer to Figure 13-8 on page 318.
3. Select the **Synchronize** icon on the right side of the offline Portal page and offline forms.
4. Wait until synchronization is complete. Disconnect from the server and select the **Offline Portal page** link to view the content of your offline Portal page offline.
5. All portlets on this page will appear as icons. Click an icon to display the portlet individually on a page.
6. If you are viewing and completing an offline form, submit the form. When you're done, go back to the Home page. Connect your Pocket PC to the server and select the **Synchronize** icon on the right side of offline forms.
7. After the synchronization completes, select the **Offline forms** link to view the submission status. If you submitted a form to retrieve data, select the **Success** link to view the retrieved data.

13.5.2 Palm OS V5.2 devices

The instructions for using Offline Portal Content on Palm OS V5.2 devices are as follows:

1. The client needs to be configured to point to the Offline Portal Server in which offline Portal pages exist.

   a. Launch Everyplace Client on your Palm OS device. Select the **WCC** icon on the All application page.



Figure 13-10 Select WCC icon

   b. Select the **Menu** button in the bottom-left corner and select **Settings** on the drop-down Options menu at the top.

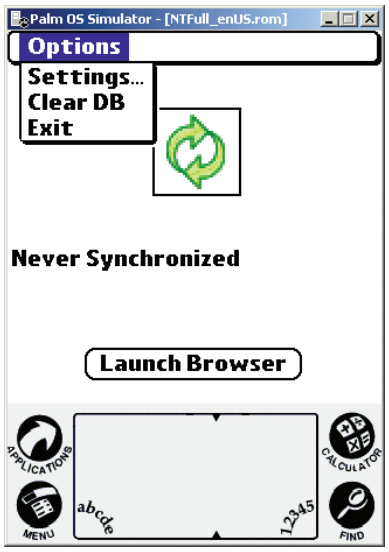


Figure 13-11 Select Settings

c. Enter the following information and select **OK**:

- Host: Your server name - fully qualified
- User name: The Portal user name for the user you want to connect as
- Password: The password for the user you want to connect as

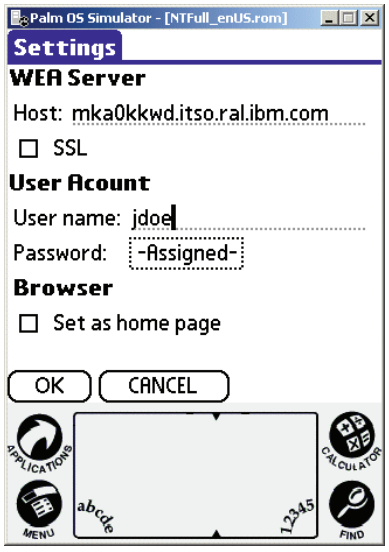


Figure 13-12 WebSphere Everyplace Access server settings

2. Select the **Synchronize** icon.

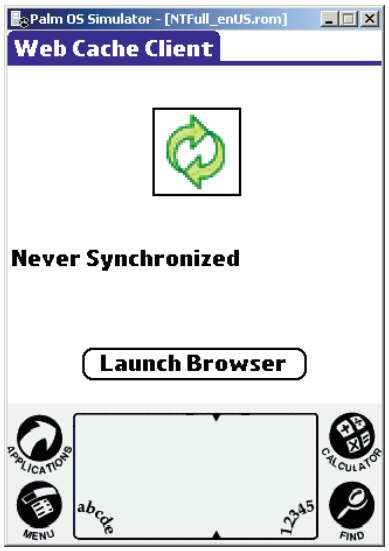


Figure 13-13 Select Synchronize icon

3. When the synchronization is complete, select **Launch Browser**.

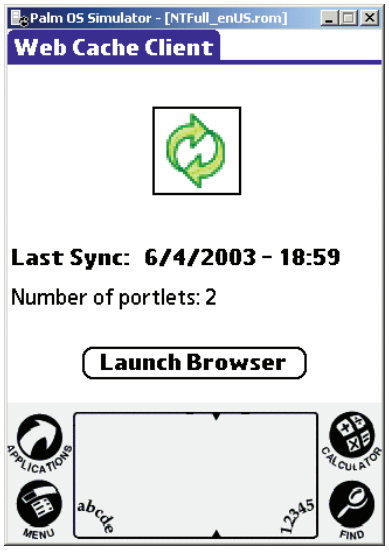


Figure 13-14 Synchronization Complete

4. All portlets on this page will appear as icons.

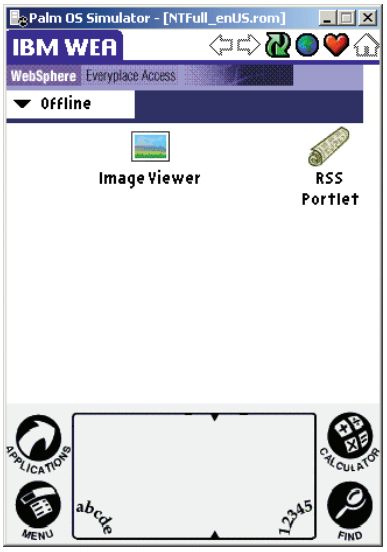


Figure 13-15 Offline page

5. Click an icon to display the portlet individually on a page.



Figure 13-16 Individual Portlet

Note: For WebSphere Everyplace Access Version 4.3, there is only support for offline browsing on Palm OS devices. Offline forms on Palm OS devices is not supported.

13.6 Development guidelines

In this section, we discuss a few guidelines that should be followed when developing portlets for offline browsing.

13.6.1 Enable support for PDA markup

In order for a portlet to be viewed offline it must:

- Exist on the offline Portal page
- Support PDA markup

The latter allows the Portal Server to recognize that the request is coming from a PDA device and map it to the appropriate controller and JSP packaged in the portlet. To designate a portlet as supporting PDA markup, you need to insert a markup name of `pda` between the supports tag in the portlet.xml file of the portlet. See Example 13-1.

Example 13-1 Adding Support for PDA markup in portlet.xml

```
...
<supports>
  <markup name="html">
    <view output="fragment">
  </markup>
  <markup name="pda">
    <view output="fragment">
  </markup>
</supports>
...
```

13.6.2 Adhere to XML “well-formedness”

During synchronization, the Everyplace Access offline server must crawl portlet pages and process them for offline viewing. To ensure this offline processing is done correctly, portlet pages must adhere to the rules of XML “well-formedness”. For example, tags such as <input> or <img> must have a closing bracket as shown in Example 13-2 on page 325.

Example 13-2 Well-formed tags

```
...
<img width="1" height="1" border="0" src='/wps/images/dot.gif' />
...
```

13.6.3 Do not use Form GET method

When developing portlets with forms, do not use the Form GET method. Everyplace Access only supports the Form POST method for submission.

13.6.4 Do not use PortletActions

PortletActions are implemented in WebSphere Portal Server using server-side processing. When pages containing PortletActions are browsed offline, the server-side processing associated with these PortletActions cannot be completed in real time. As such, the resulting links may not be valid. A workaround for this is to replace PortletActions with request parameters and make any business logic flow decision internally based on the value of these parameters.

13.6.5 Plan for dynamic content

The offline forms functionality of Everyplace Access relies on the relative location of links on a page. If these links are dynamically generated, there is no guarantee that the form will be in the same relative location during synchronization. For example, let's consider the case where we have a page containing five dynamically generated links with a form positioned between the second and third links. The form is submitted offline. During synchronization, the page gets refreshed and instead of five links, it now contains seven dynamically generated links with the form positioned between the third and fourth links. Because the relative location of the form has changed, the offline server will get confused when it tries to traverse the portlet to submit the form.

It is recommended that the use of dynamic links (or dynamic content that may affect the relative location of the form) be avoided. Where there is no option but to use dynamically generated links (or content), it is recommended that the form be strategically placed such that its relative location will not be affected by the dynamic nature of the page.

13.6.6 Avoid action buttons

During synchronization, the offline server traverses every link up until the link-depth is reached. For this reason, if a portlet contains a list of items and

beside each item is a corresponding Delete button, all items will be deleted each time synchronization is performed. A workaround for this is to put all items into a form with a check box beside each item. With this approach, there is no link to traverse that would invoke the action to delete the item.

13.6.7 Avoid cascading forms

It is common to have a form that is generated from information from a preceding form. For example, let's consider a case where we have a form containing one drop-down list of countries. On selection of a country in the list, the page is refreshed, resulting in a second drop-down list appearing on the page. This second list contains cities belonging to the country selected in the first drop-down list. These types forms are called cascading forms and should be avoided for offline forms. The offline forms functionality only supports single-level forms and cannot handle cascading forms.

13.7 Scenarios

In this section we will go through three scenarios. The first one will demonstrate the use of offline browsing. The second one will demonstrate the use of offline forms. The third one will demonstrate the process involved in converting an existing portlet, which has not been designed for offline usage, to one that can be used offline. The aim is to give you an example of the end-to-end process involved in setting up your portlets for use offline. For the purposes of these demonstrations, we will use the Pocket PC device.

13.7.1 Scenario 1: Offline browsing

In Scenario 1, we demonstrate the use of the Web Clipping functionality to produce a portlet for offline browsing. The steps involved are as follows:

1. Configure the WebSphere Everyplace Access server for offline browsing. Refer to 13.4.2, "Administrator configuration" on page 310 for details on how to do this.
2. Create a portlet using Web Clipping.
   a. Log in as `wpsadmin` if you have not already done so.
   b. Select **Portal Administration -> Portlets -> Web Clipping**.

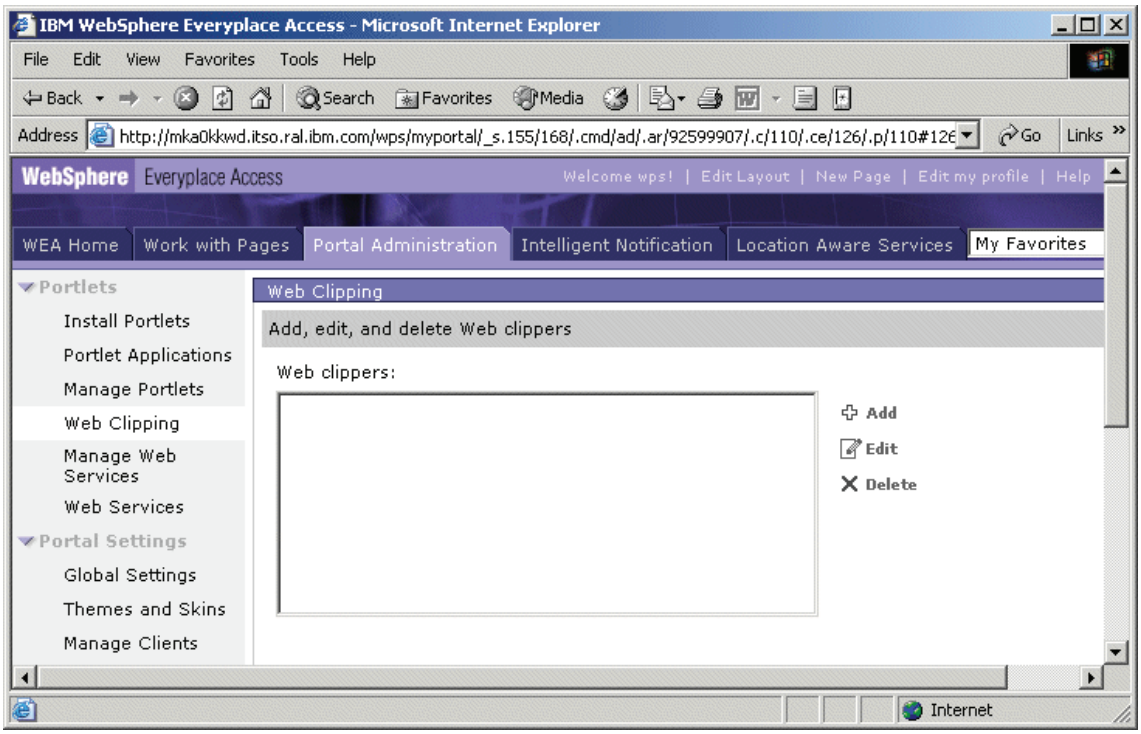


Figure 13-17 Web Clipping portlet

c. Click **Add**.

d. Complete the requested information:

- Name and default locale title: IBM Stock Price
- Description: IBM Stock Price
- URL to clip: `http://www.ibm.com/ibm/stock`

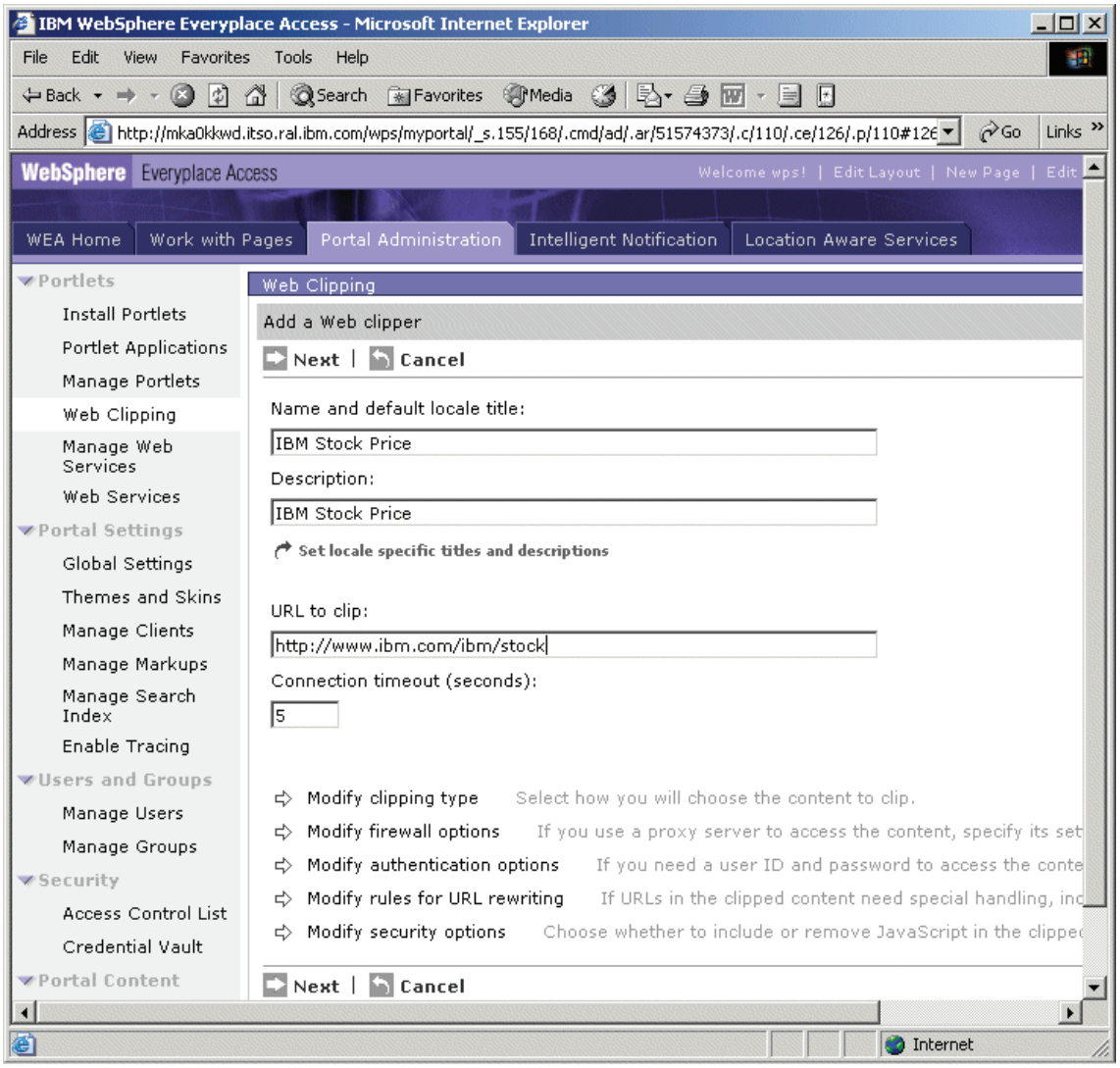


Figure 13-18 Add a Web clipper

e. Click **Next** and you will be presented with the `http://www.ibm.com/ibm/stock` page.

f. We now need to select the areas of the page we want to include the IBM Stock Price portlet. Select the fields you want to include. Each click will add the field to your selection. Each selected field will be highlighted in yellow. You may need to click multiple times to get all the fields you want.

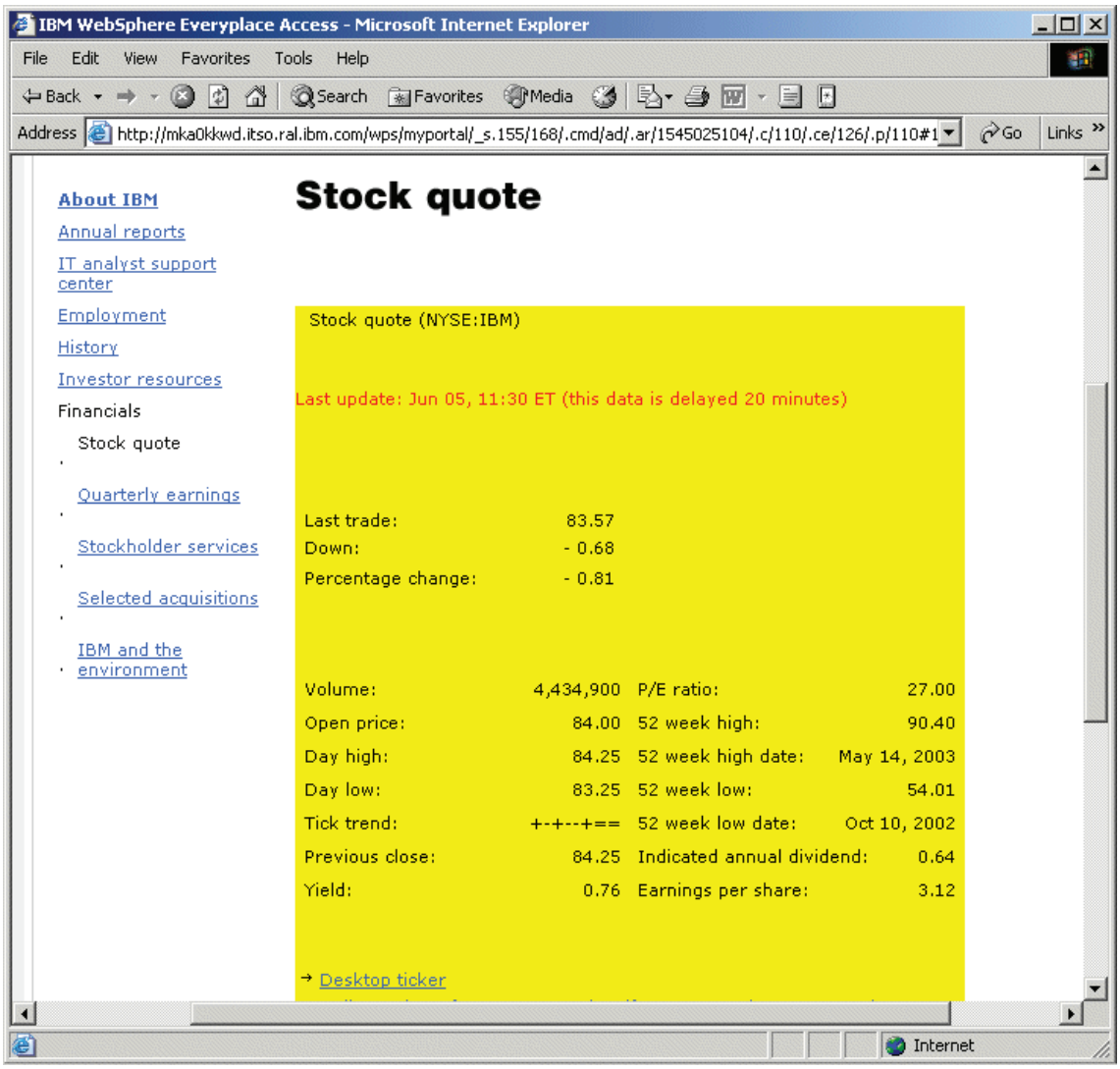


Figure 13-19 Select fields to include in IBM Stock Price portlet

g. Click **Next** when you are done selecting the fields.

h. The next window will be a preview of your selection. If you are happy with it, click **Done**. Otherwise, click **Cancel** and start again.

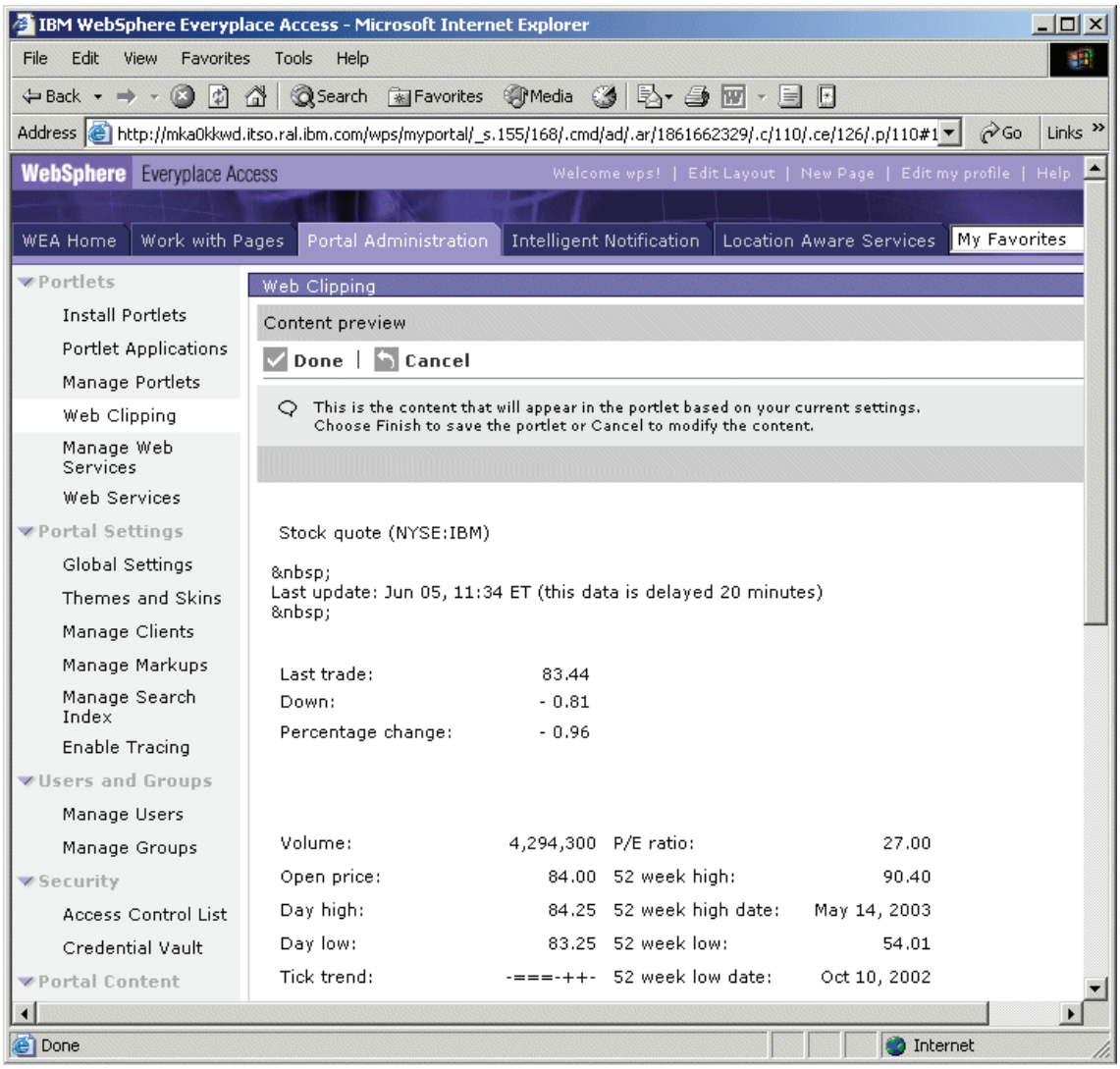


Figure 13-20 Content preview

i. You will be returned to the first page of the Web Clipping porlet. In the Web clippers box, you will see your newly created IBM Stock Price portlet.

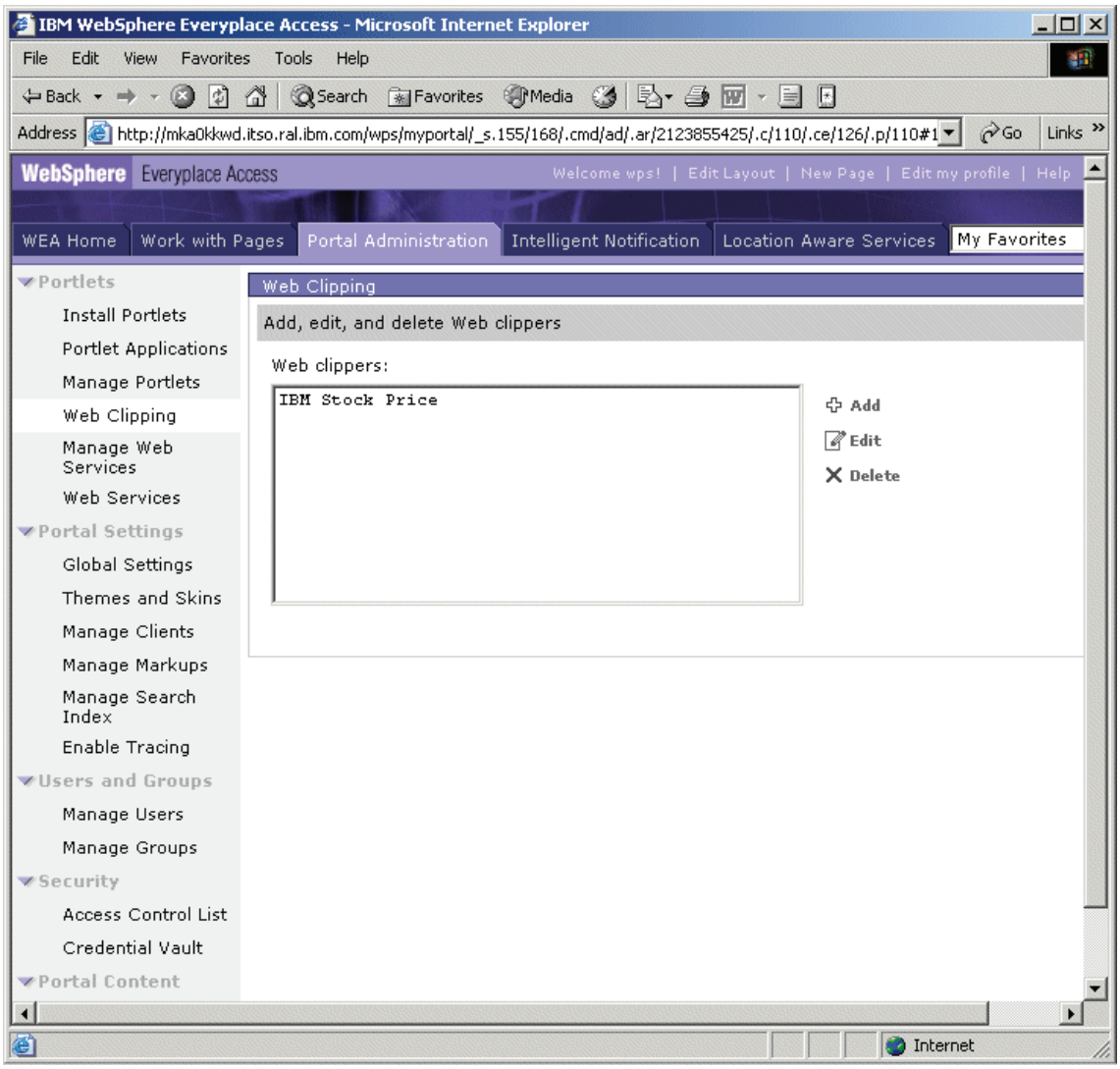


Figure 13-21 Web Clipping portlet

j. We have now created a portlet using Web Clipping. This completes the Web Clipping part of this scenario.

3. Change the access rights on the created portlet.

   a. Log in as `wpsadmin` if you have not already done so.

   b. Select **Security -> Access Control List**.

c. Select the **Selected users and groups** radio button and click **Get groups and users**.

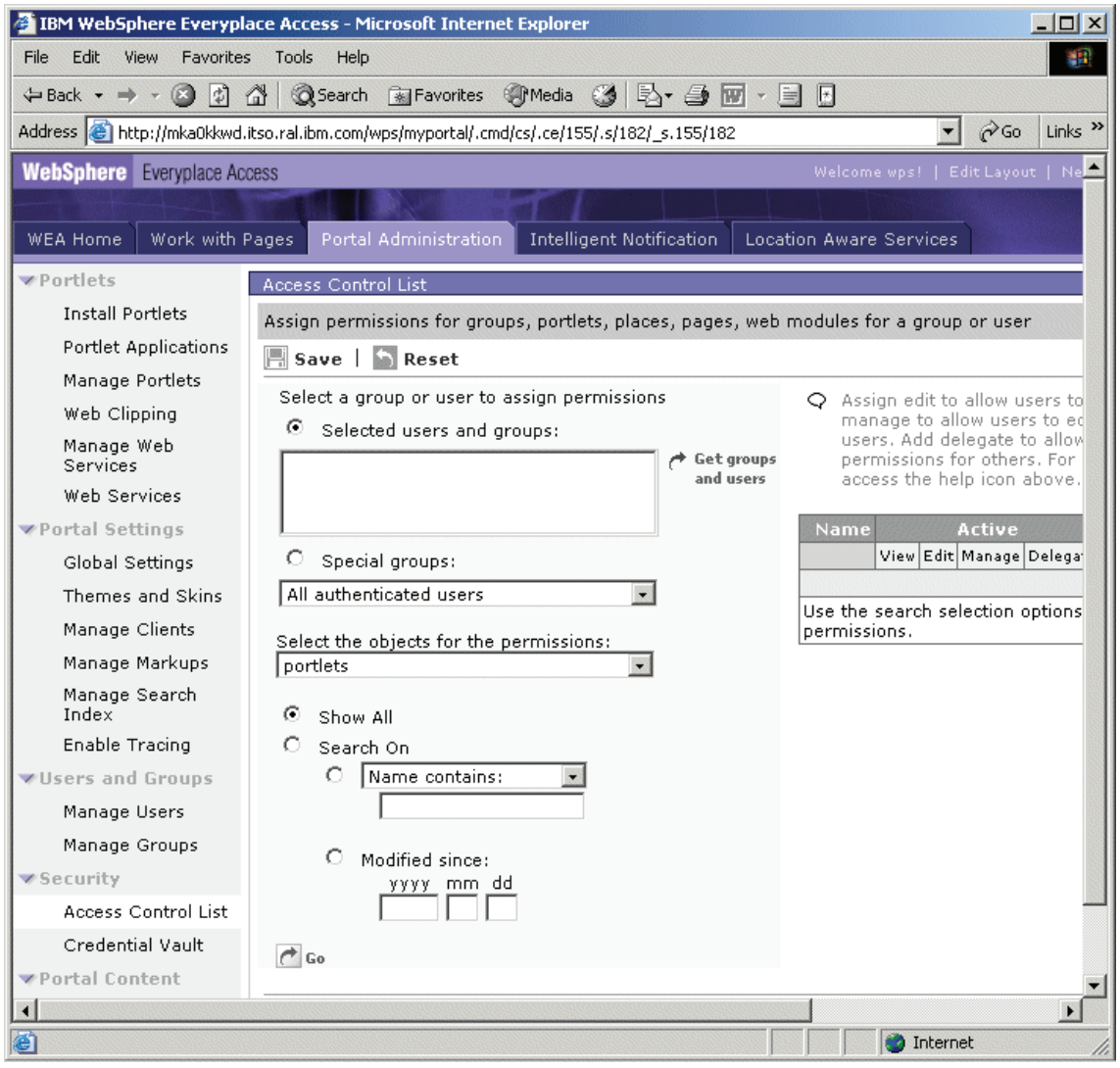


Figure 13-22 Access Control List

d. Select the **Search for users** radio button and enter an asterisk (*) in the Name is field.

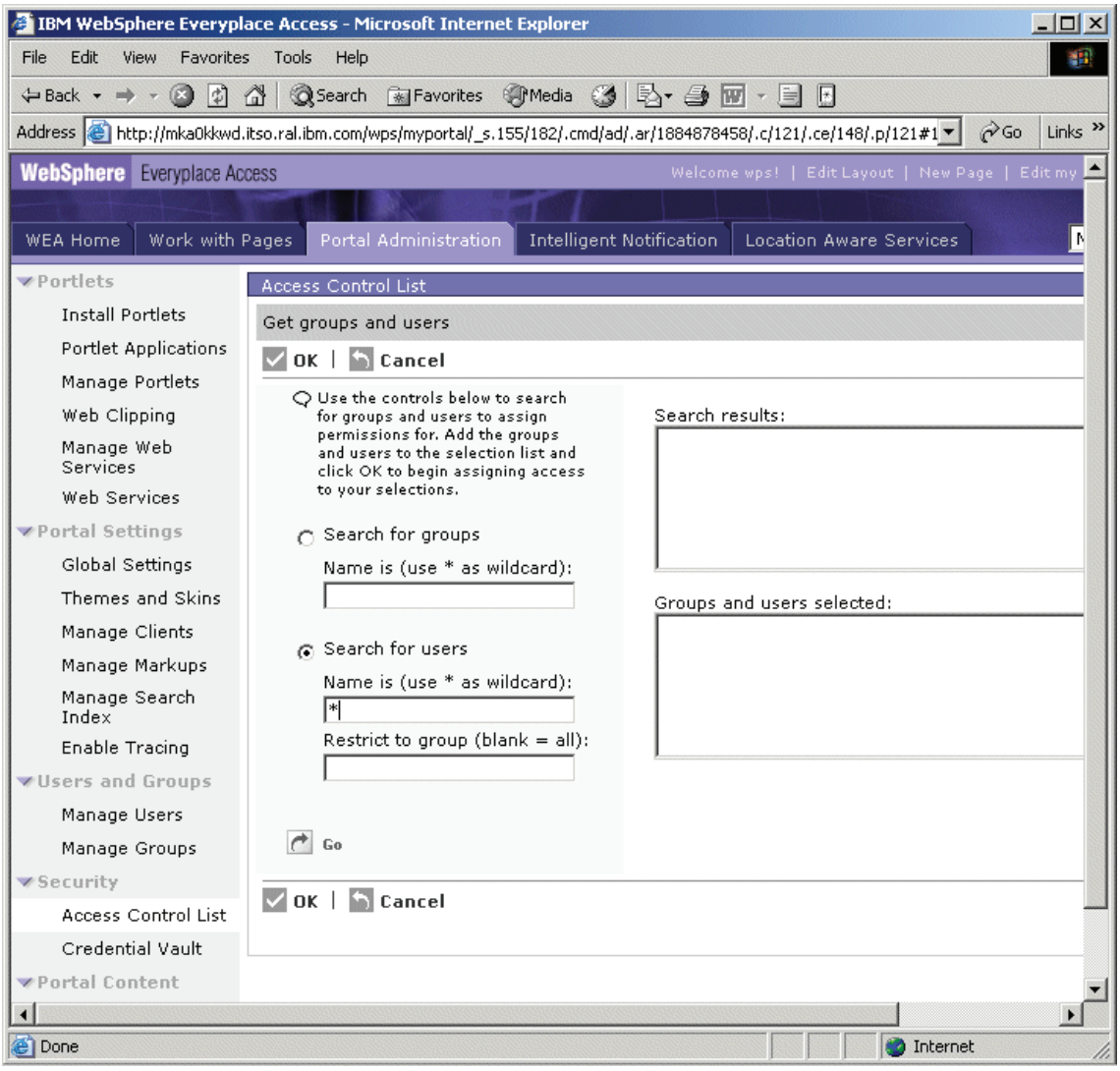


Figure 13-23 Get groups and users

e. Click **Go** and you will be presented with a list of all users in the Search results box.

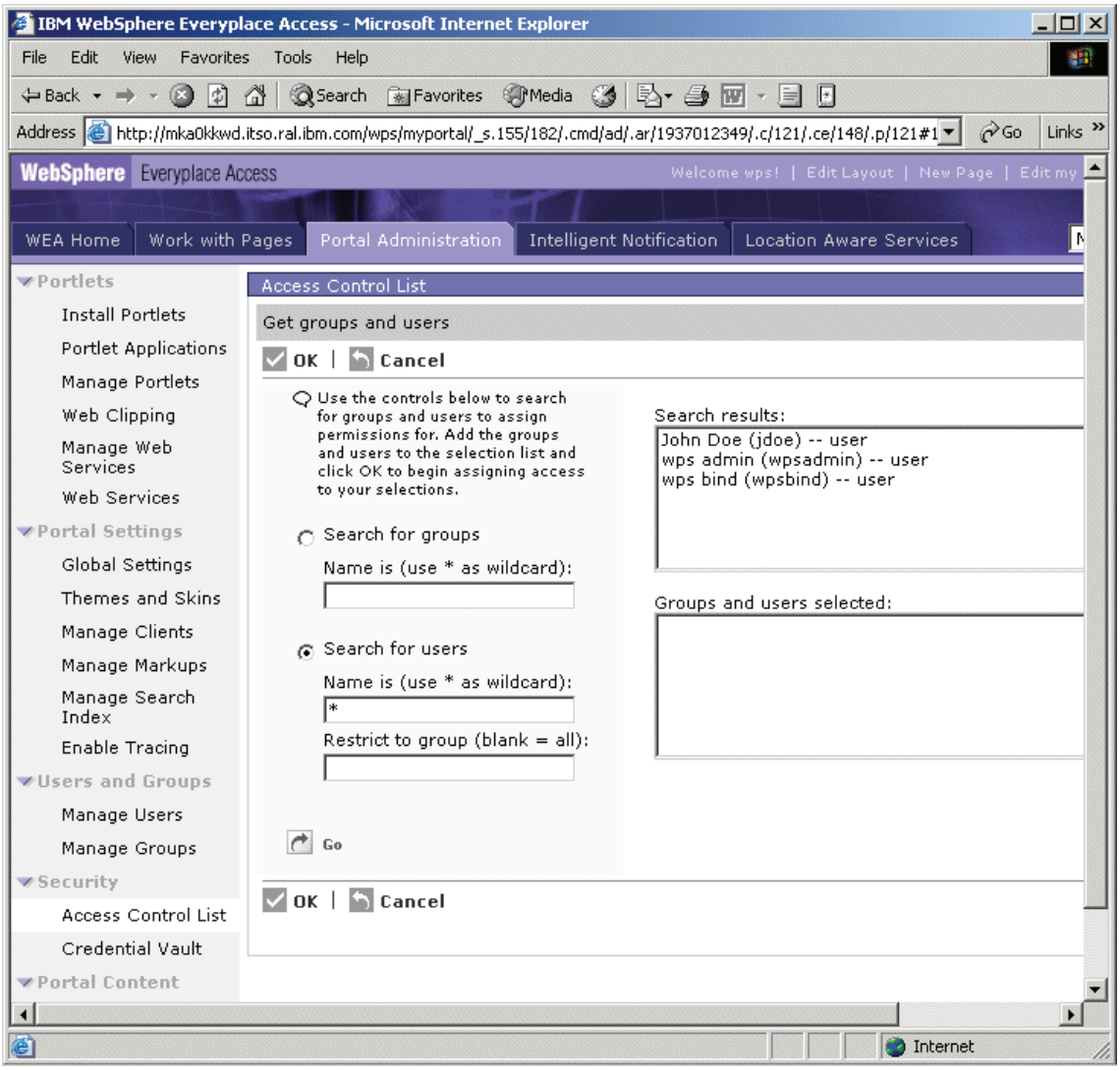


Figure 13-24 Get groups and users - search results

f. From the Search results box, select the user who is going to have access to this new portlet and then click **Add to list**.

g. The selected user will now appear in the Groups and users selected box. Click **OK**.

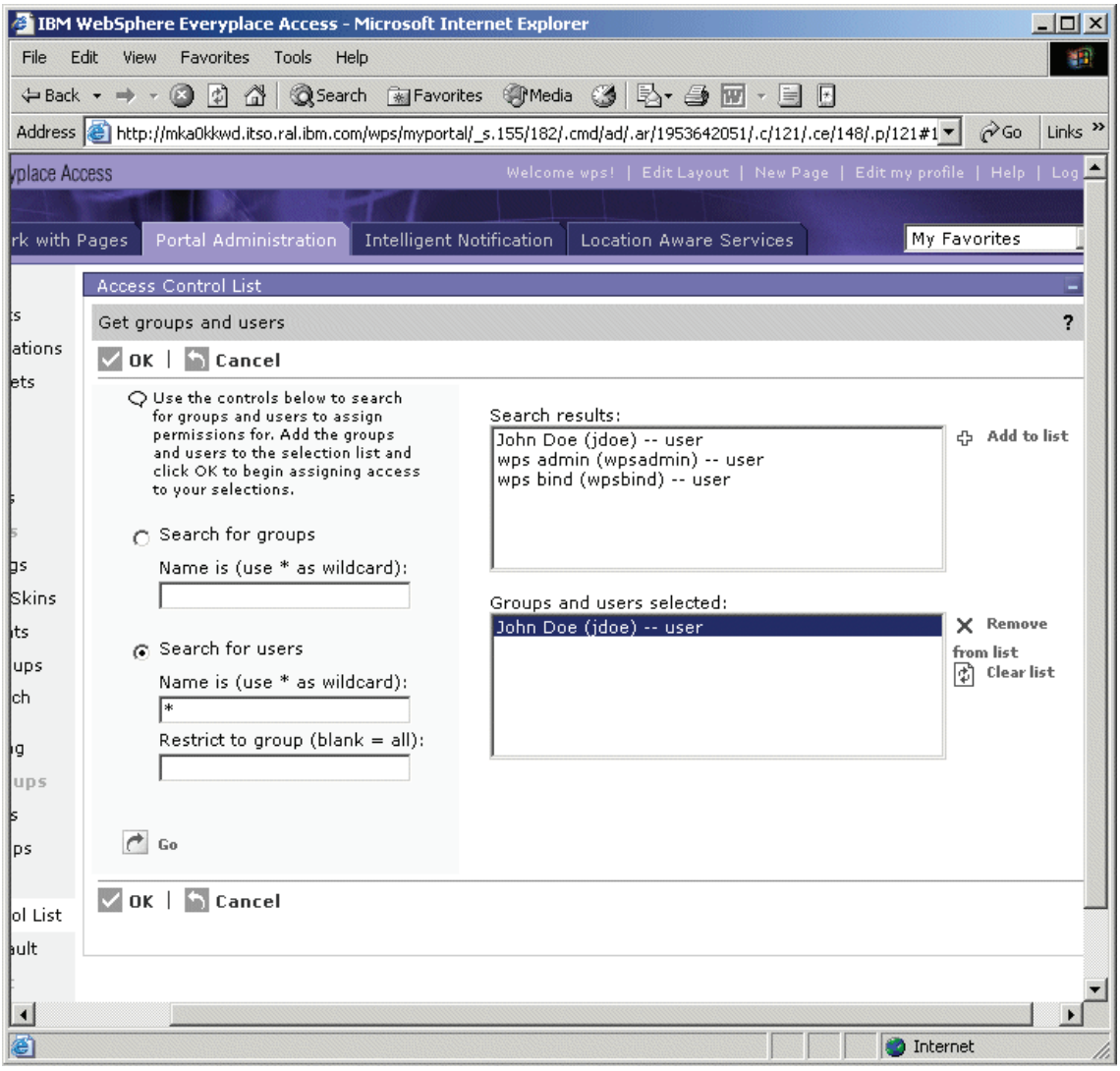


Figure 13-25 Groups and users selected

h. You will be returned to the Access Control List page. In the Select the objects for the permissions drop-down list, make sure **portlets** is selected. Click **Go**.

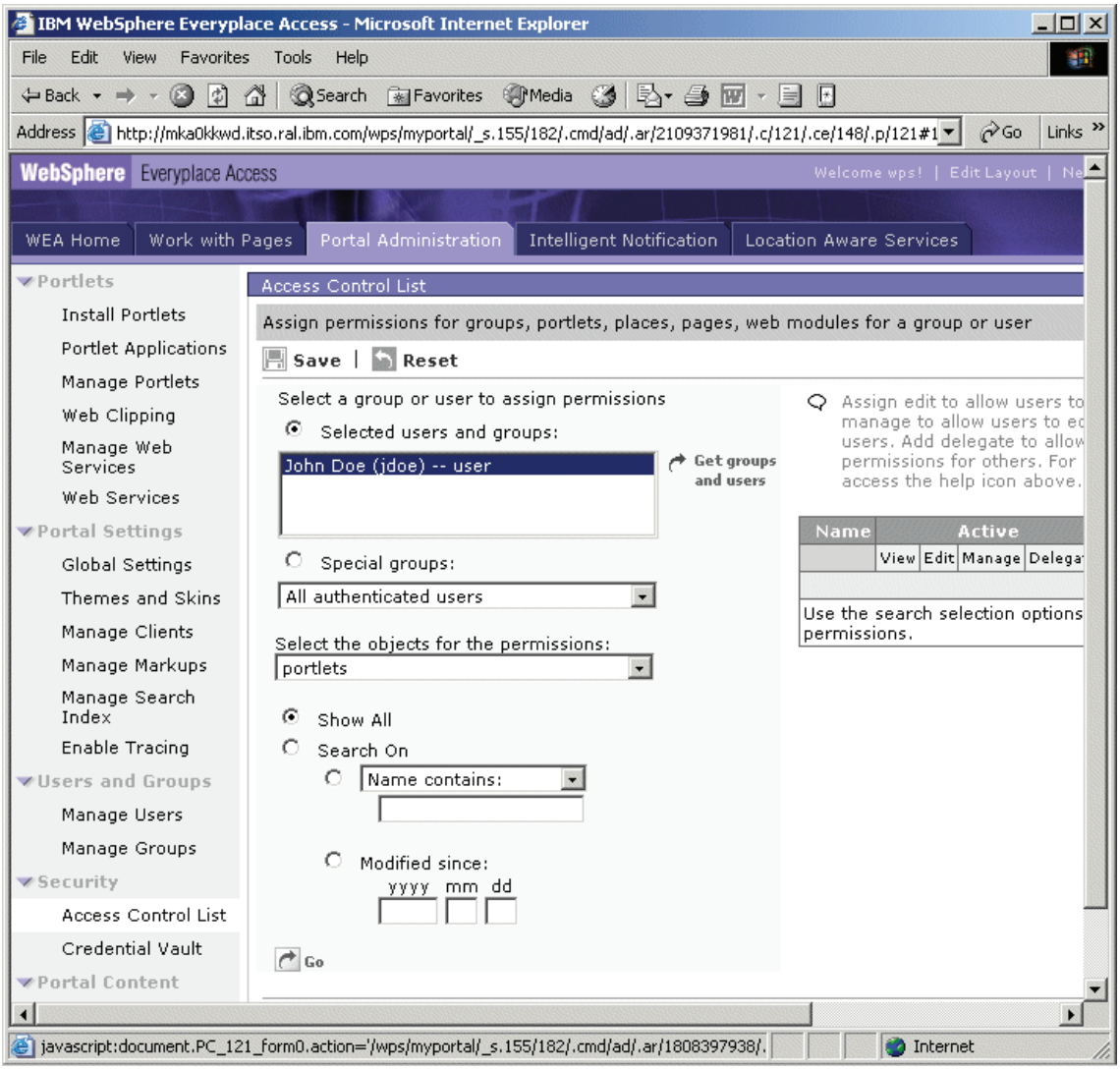


Figure 13-26 Access Control List - Select the objects for the permissions

i. A table will now appear on the right showing all the portlets and the current permissions set for that user.

j. Locate the IBM Stock Price portlet and select the **View** radio button.

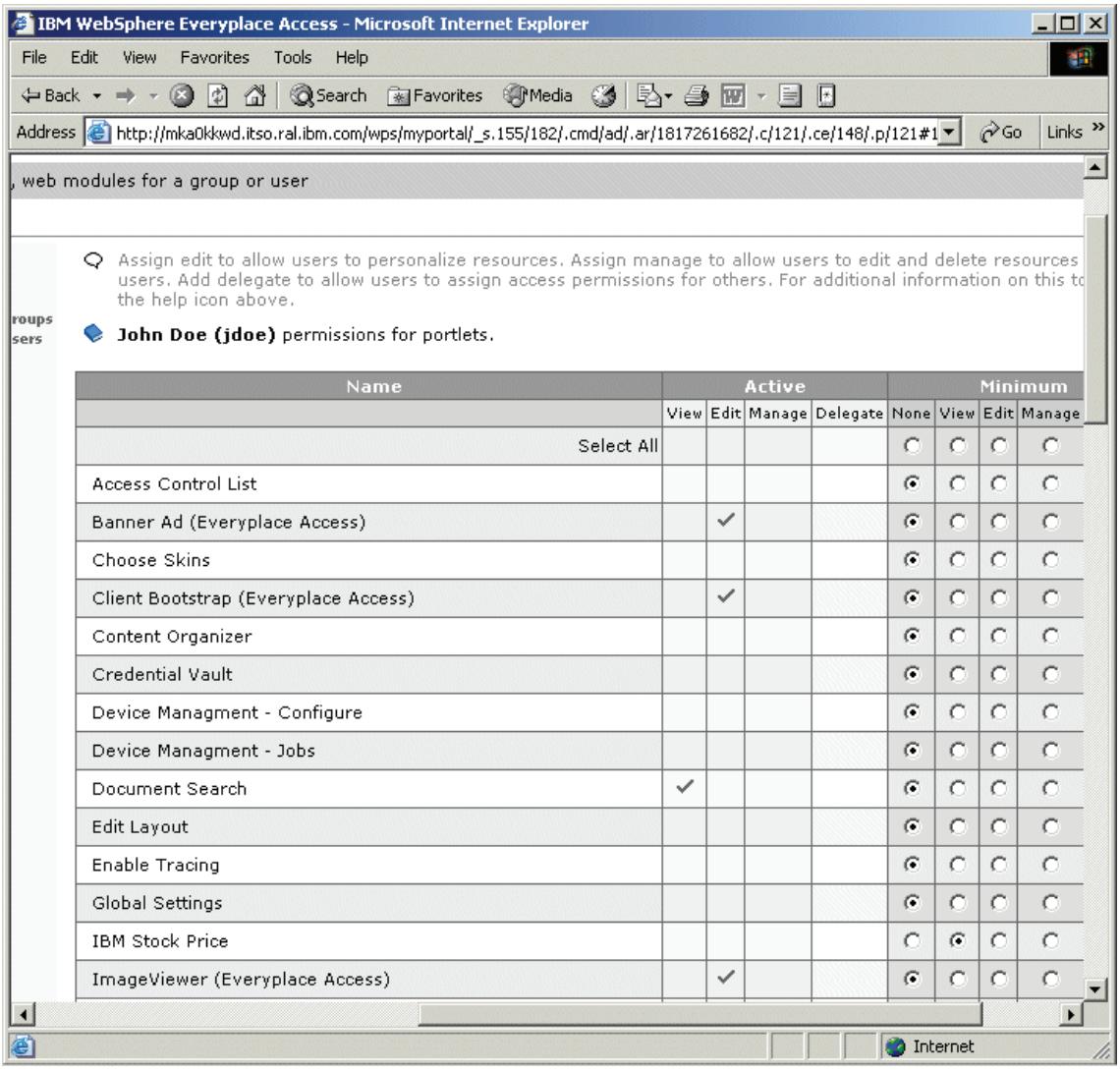


Figure 13-27 Change permissions for portlets

k. Click **Save**.

4. Add the portlet to the offline Portal page.

   a. Log in the as the user who will view this portlet offline.

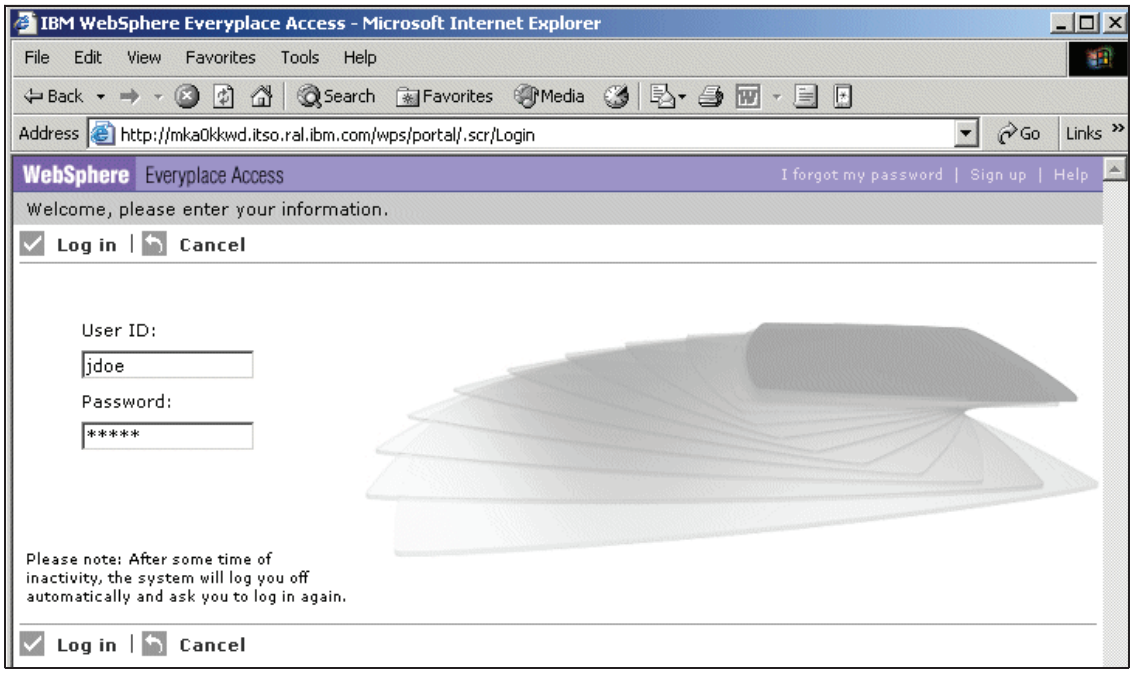


Figure 13-28 Log in as user

b. Select **Work with Pages -> Edit My Pages -> Edit page composition**.

c. Under Pages in the place, select **Offline**.

d. Click **OK**.

e. Click **Add content**.

f. From the list of portlets, select **IBM Stock Price** and click **OK**.

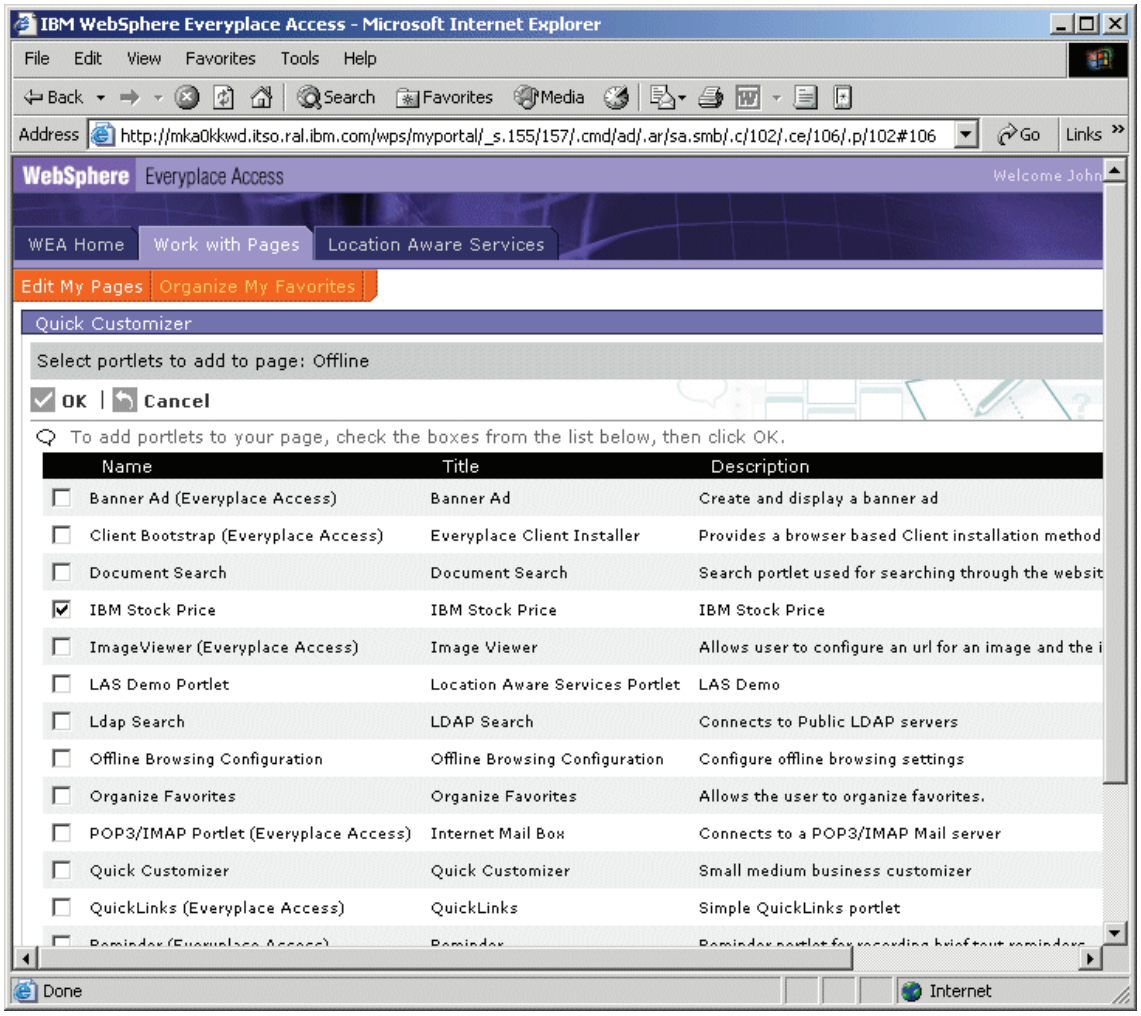


Figure 13-29 Select portlet to add to offline page

g. The IBM Stock Price portlet should now appear in the Page layout for the offline page. Click **Done**.

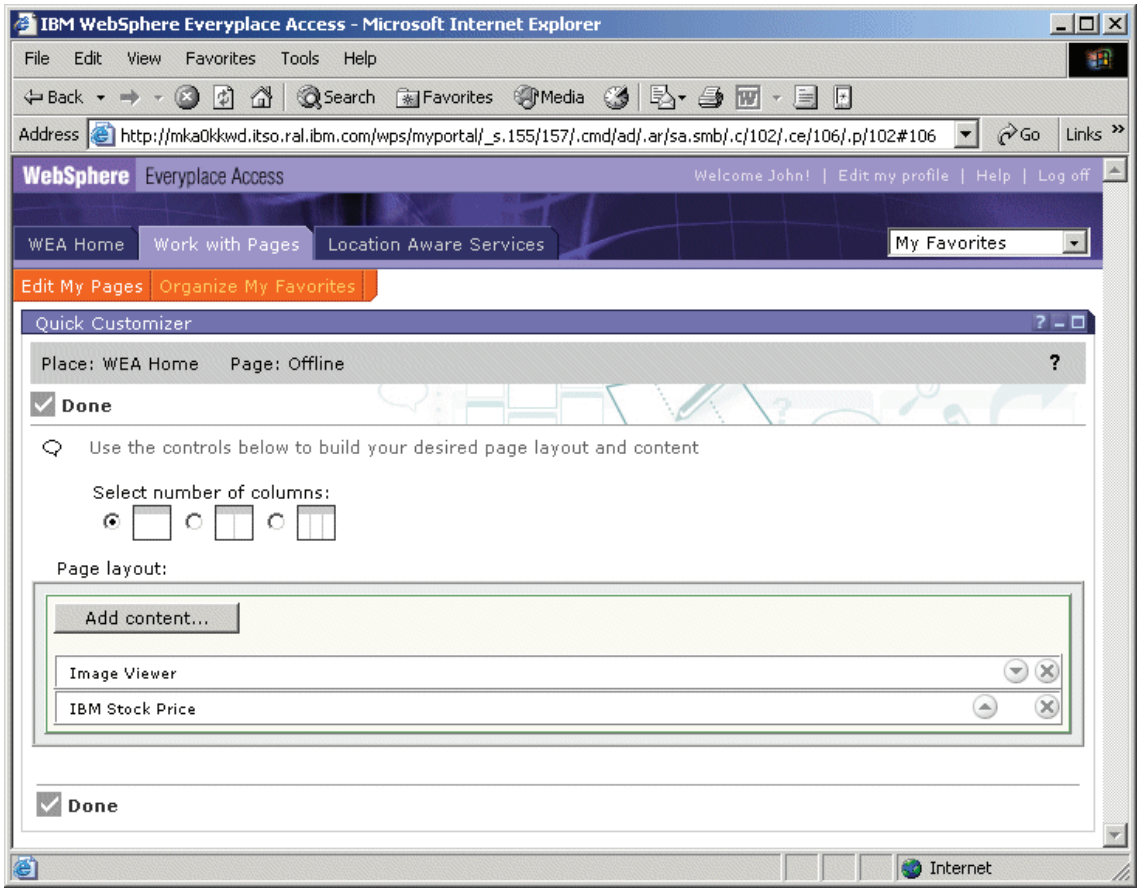


Figure 13-30 Page layout of offline page

h. The new portlet has now been added to the offline Portal page. To verify this, select **WEA Home -> Offline**. The IBM Stock Price portlet should be visible on this page.

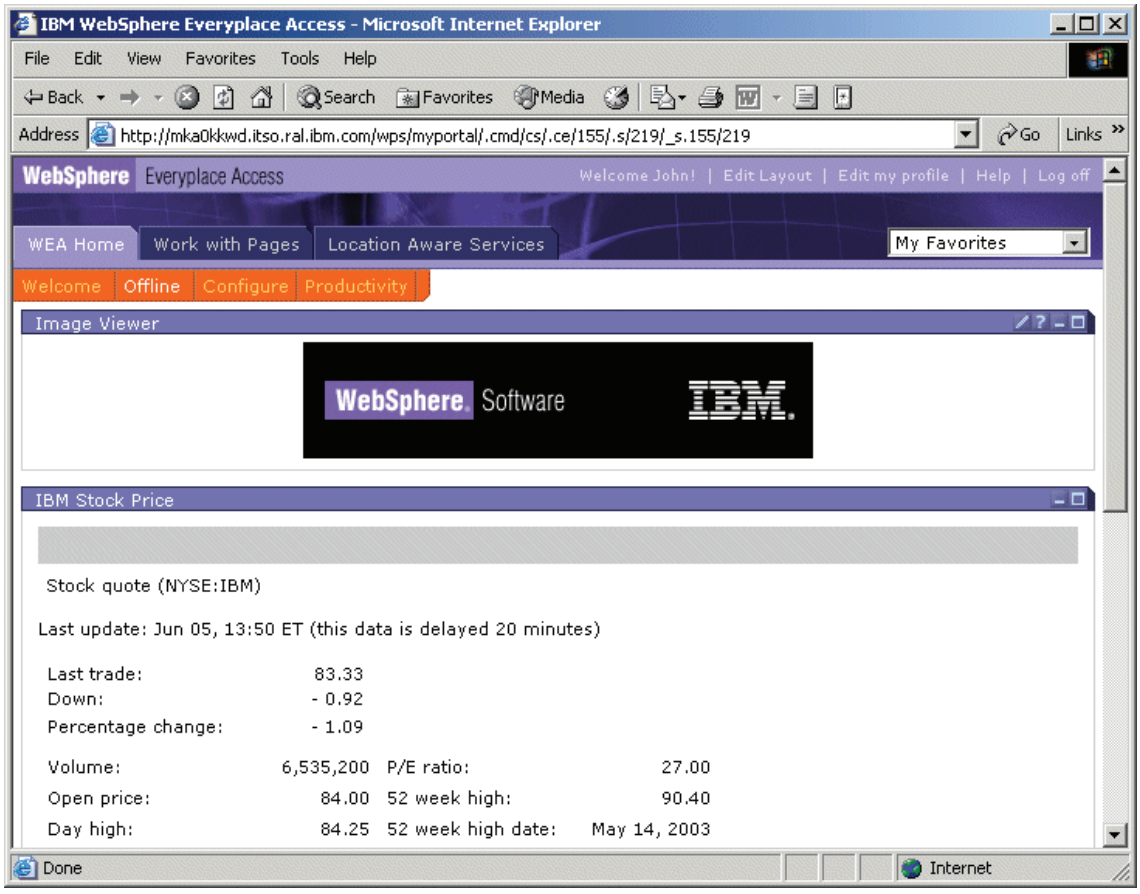


Figure 13-31 Offline page viewed online

5. Configure the client device for offline browsing. Refer to step 1 in 13.5.1, “Pocket PC devices” on page 317.
6. Synchronize the client device with the server:
   a. Start Everyplace Client if you have not already done so. Click **Start -> Everyplace Client**.
   b. If prompted, enter the user name and password.
   c. On the Home page, select the **Synchronize** icon on the right side of the offline Portal page.

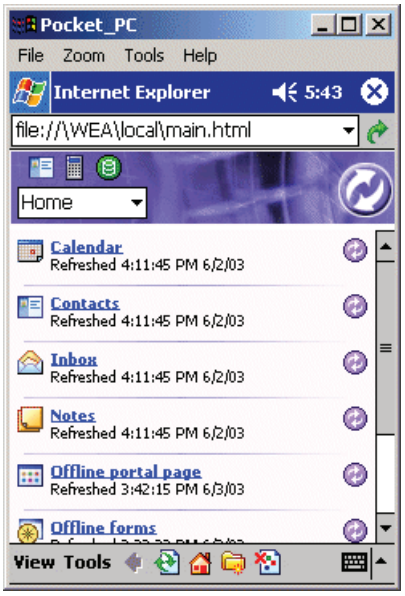


Figure 13-32 Select Synchronize icon for offline Portal page

d. If the synchronization was successful, the Home page would be refreshed and under offline Portal page, there will be some text indicating when the offline page was last refreshed. If the synchronization was not successful, you will get an Error icon (a red circle with a cross in it) next to the Synchronize icon. You can click the **Synchronize** icon to find out why the synchronization failed.

7. View the portlet offline.

   a. Disconnect the Pocket PC from the Server and select the **Offline Portal page**. The portlets on the offline Portal page will appear as icons.

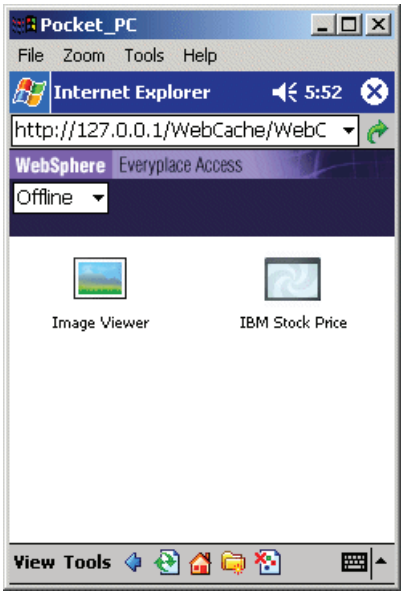


Figure 13-33 Offline Portal page viewed offline.

b. Select the **IBM Stock Price** icon to view the portlet.

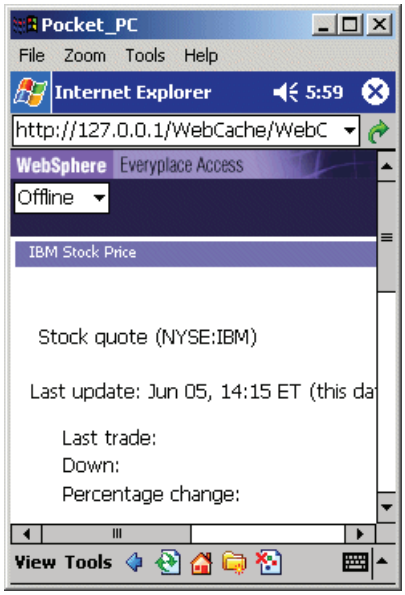


Figure 13-34 IBM Stock Price portlet viewed offline

13.7.2 Scenario 2: Offline forms

In Scenario 2, we will step through an example where an existing form is added to the offline Portal page and then used offline. The steps involved are as follows:

1. Configure the WebSphere Everyplace Access server for offline browsing. Refer to 13.4.2, “Administrator configuration” on page 310 for details on how to do this.
2. Add the portlet containing the form to the offline Portal page.
   a. Log in the as the user who will view this form offline.

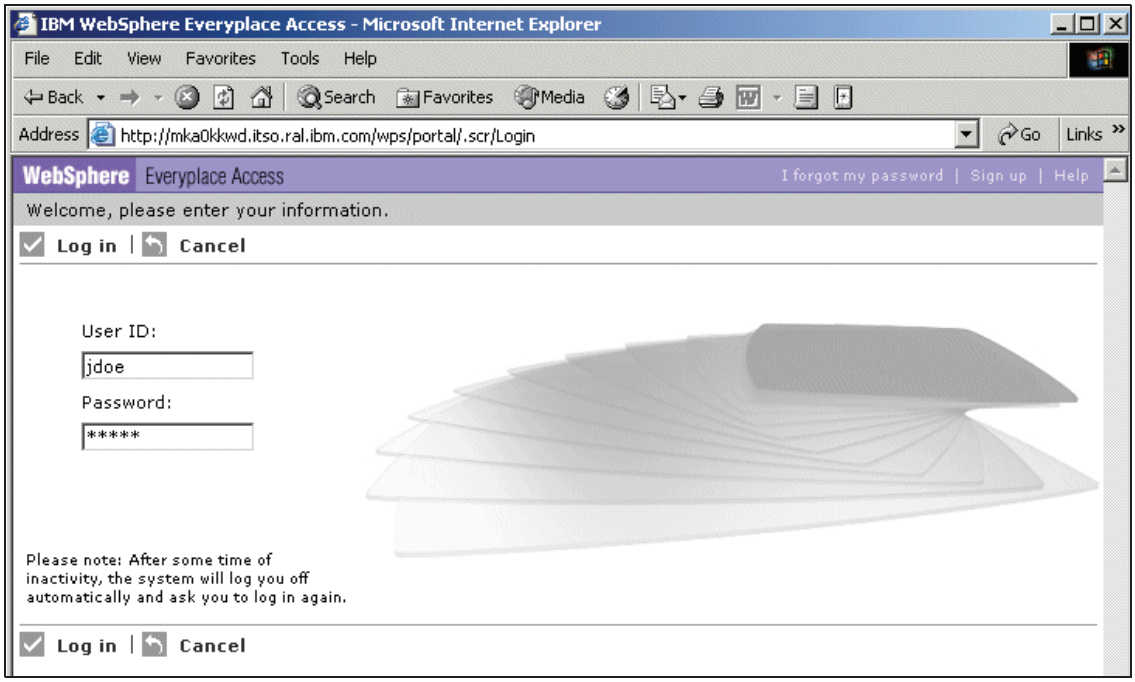


Figure 13-35 Log in as user

   b. Select **Work with Pages -> Edit My Pages -> Edit page composition**.
   c. Under Pages in the place, select **Offline**.
   d. Click **OK**.
   e. Click **Add content**.
   f. From the list of portlets, select the **World Clock** portlet. This is a portlet that comes with Everyplace Access and contains a form. The form takes in a time zone /location as a parameter and retrieves the current time of that time zone /location when submitted. Click **OK**.

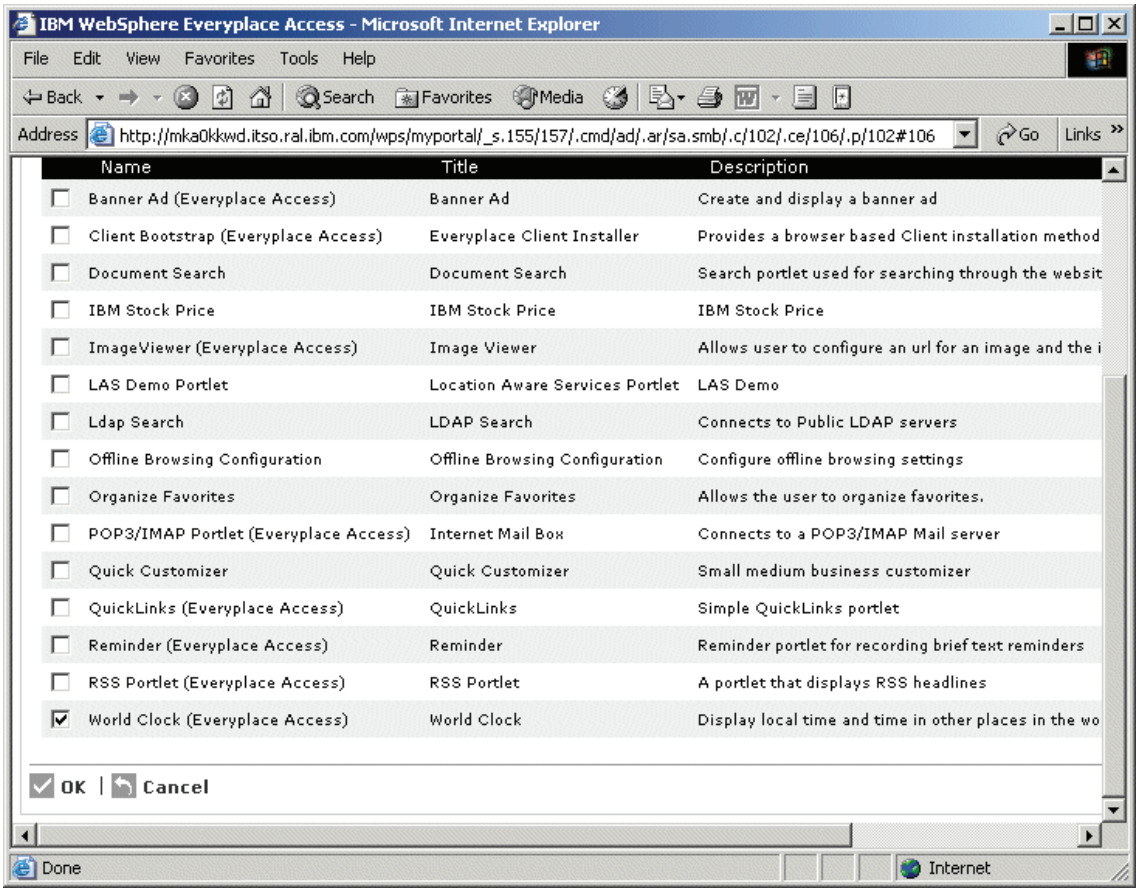


Figure 13-36 Selecting portlets to add

g. The World Clock portlet should now appear in the Page layout for the offline page. Click **Done**.

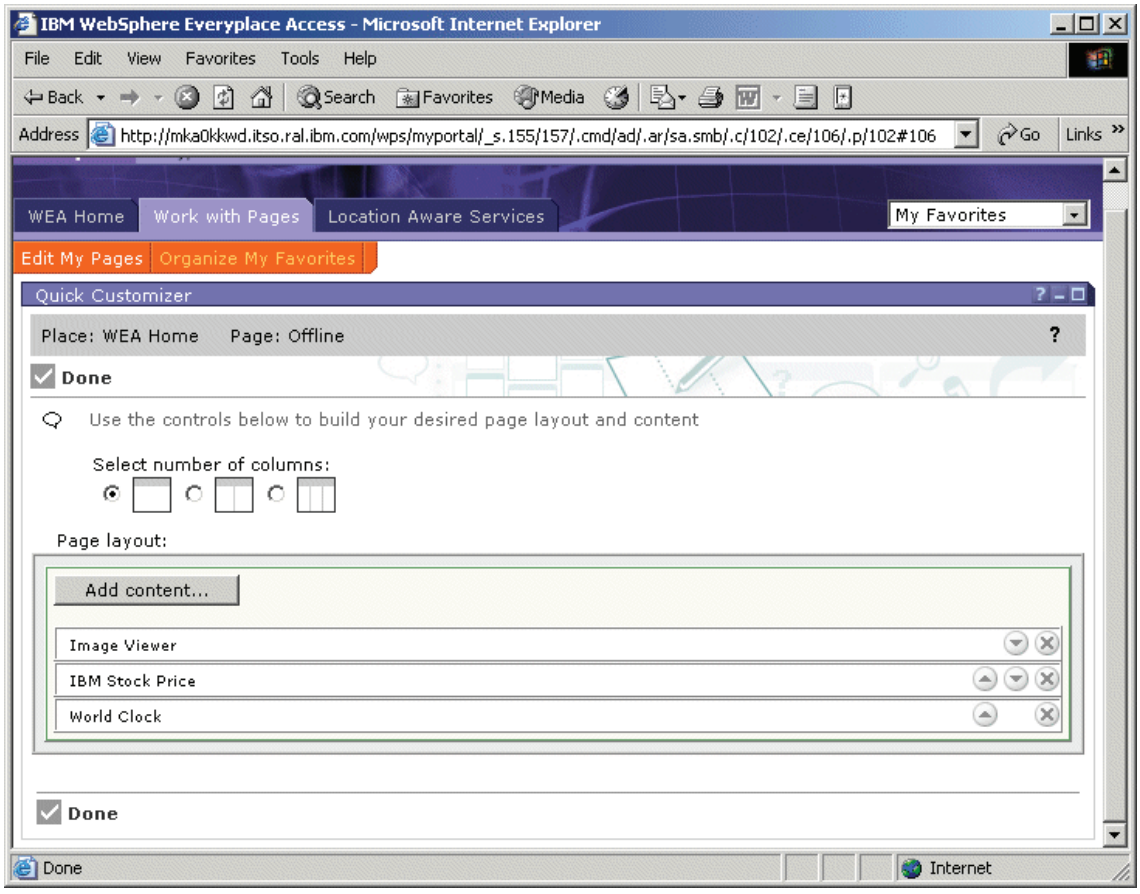


Figure 13-37 Page layout of offline page

h. The new portlet has now been added to the offline Portal page. To verify this, select **WEA Home -> Offline**. The World Clock portlet should be visible on this page.

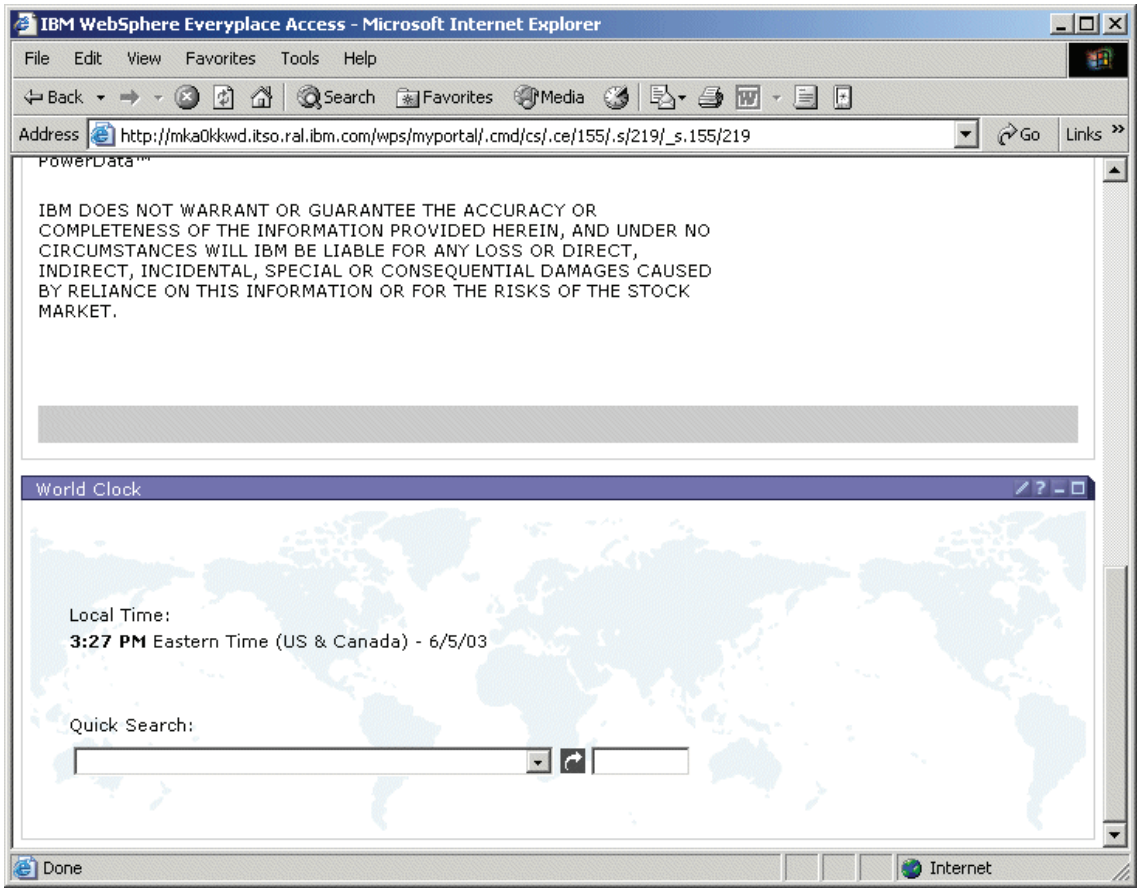


Figure 13-38 Offline page viewed online.

3. Configure the client device for offline browsing. Refer to step 1 of “Pocket PC devices” on page 317.
4. Synchronize the client device with the server.
   a. Start Everyplace Client if you have not already done so. Click **Start -> Everyplace Client**.
   b. If prompted, enter the user name and password.
   c. On the Home page, select the **Synchronize** icon on the right side of offline Portal page.

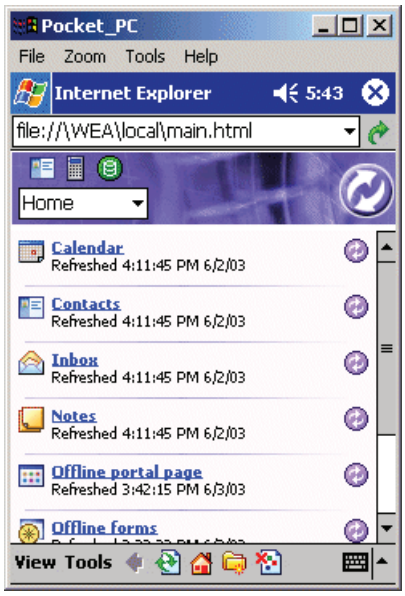


Figure 13-39 Select Synchronize icon for offline Portal page

d. If the synchronization was successful, the Home page would be refreshed and under the offline Portal page, there will be some text indicating when the offline page was last refreshed. If the synchronization was not successful, you will get an Error icon (a red circle with a cross in it) next to the Synchronize icon. You can click the **Synchronize** icon to find out why the synchronization failed.

5. View and use the portlet offline.

   a. Disconnect the Pocket PC from the server and select the **Offline Portal page**. The portlets on the offline Portal page will appear as icons.

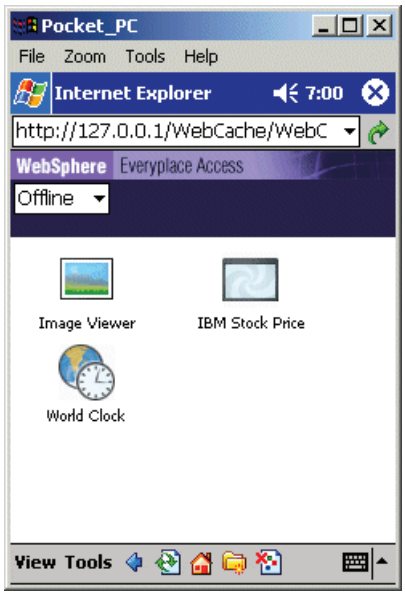


Figure 13-40 Offline Portal page viewed offline

b. Select the **World Clock** icon to view the portlet.

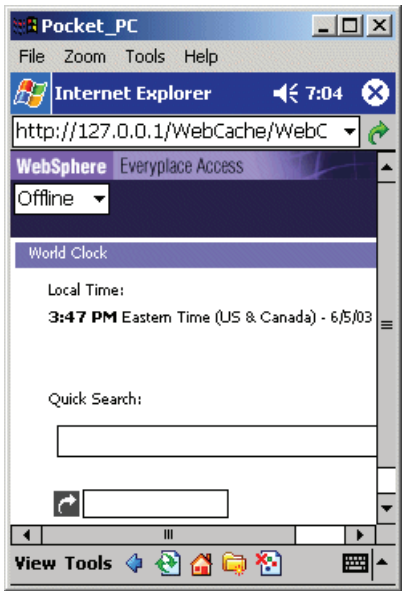


Figure 13-41 World Clock portlet viewed offline

c. Select the **Quick Search** drop-down list and select **Hawaii**.

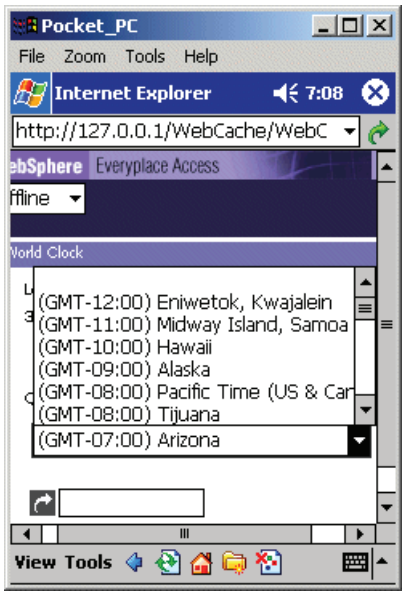


Figure 13-42 Select time zone

d. Submit the form by clicking the button with the curved arrow pointing to the right.

e. You will get a message telling you that your form data has been saved and will be submitted the next time you invoke a synchronization.

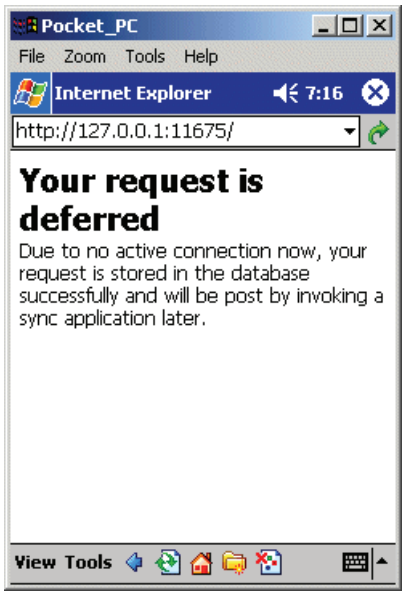


Figure 13-43 Request deferred

f. Close Everyplace Client by clicking the **X** in the top-right corner.

6. Synchronize and view results of form submission:

   a. Connect your Pocket PC to the server and launch Everyplace Client. Click **Start -> Everyplace Client**.

   b. If prompted, enter the user name and password.

   c. On the Home page, if you look at the text under offline forms, you will notice that it says there is one pending form. This means that there has been one form already submitted offline and is waiting to be submitted during the next synchronization. Select the **Synchronize** icons on the right side of offline Portal page and offline forms. This will result in offline pages being refreshed and any pending forms being submitted to the server. The results of the form submission will also be downloaded to the client device during synchronization.

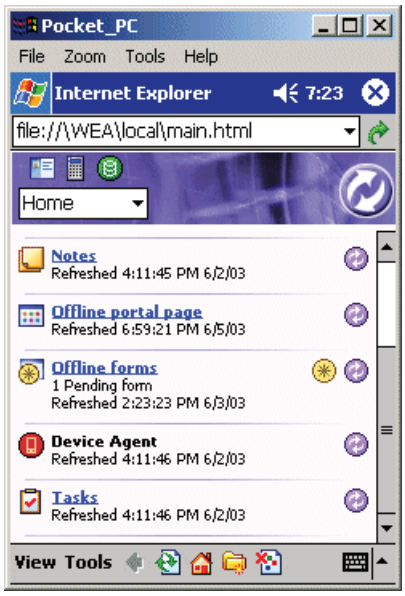


Figure 13-44 Synchronize offline Portal page and offline forms

d. After the synchronization for offline forms is complete, your Everyplace Client Home page will be refreshed and you will notice a green circle with the exclamation mark next to the Synchronize icon for offline forms.

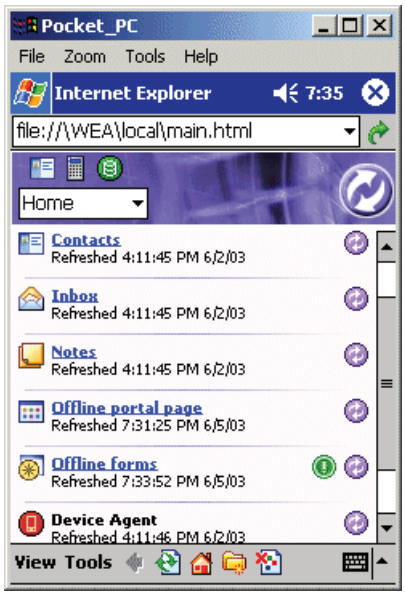


Figure 13-45 Synchronization of offline forms complete

e. Select **Offline forms** and you will be brought to the Manage Submissions page. This page stores information on all your form submissions. You can delete old entries if you don't need them anymore by checking the appropriate check box and clicking **Delete**.

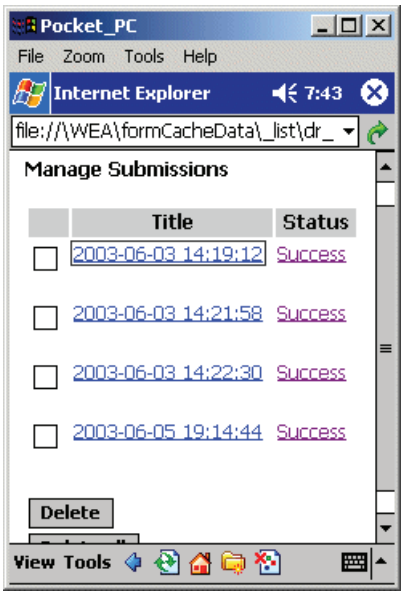


Figure 13-46 Manage Submissions page

f. Select the date corresponding to your form submission and then select **Show response**. In our case, we will select **2003-06-05 19:14:44.** Alternatively you can just select the **Success** link next to the appropriate date and bypass the status page shown in Figure 13-47 on page 355.

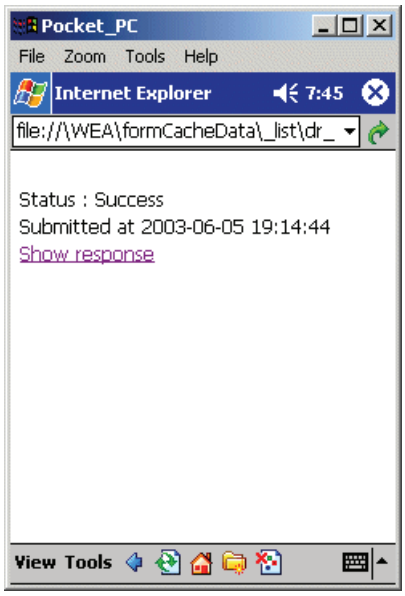


Figure 13-47 Select Show response

g. You will now be presented with the response of the form submission. You will notice the time for the selected time zone (Hawaii) has been retrieved and placed in the box below the Quick Search drop-down listbox.

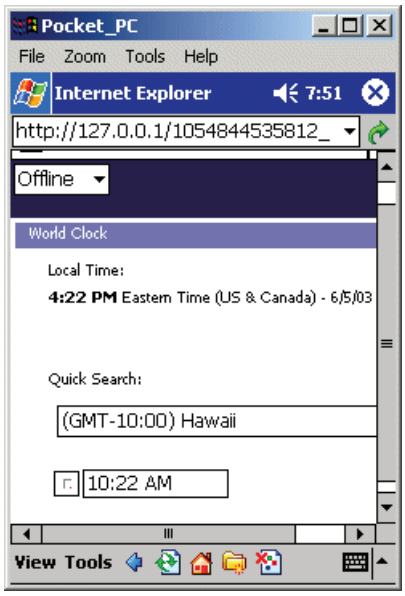


Figure 13-48 Response of form submission

13.7.3 Scenario 3: Converting an online portlet for offline usage

There are often times when we have existing portlets that have not been designed for offline usage. Viewing these portlets offline will often result in unpredictable and incorrect behavior. In this scenario, we will take an existing portlet that has not been designed for offline usage and modify it to be *offline friendly*.

The portlet example

We will start with a simple portlet example to demonstrate the steps involved in converting the portlet for offline usage. The portlet has two main functions:

- Allowing a user to select (by clicking a Submit button) from a RED or BLUE action in the edit mode.
- Displaying the last action selected by the user in the view mode.

When the portlet is first loaded, it will be in the view mode. Because there will be no selected action, it will look like Figure 13-49 on page 357.

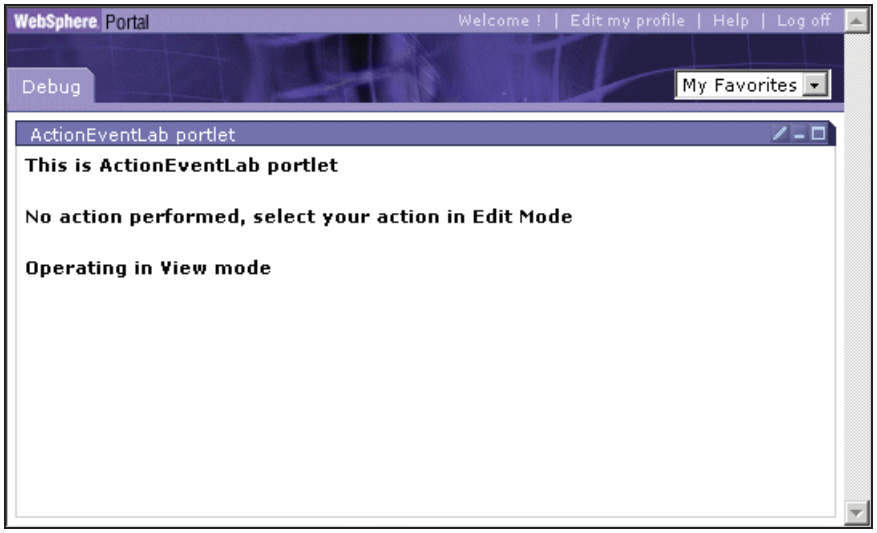


Figure 13-49 Portlet in view mode - First time JSP is loaded in view mode

When the **Edit** button is clicked, the portlet will be in placed in edit mode.

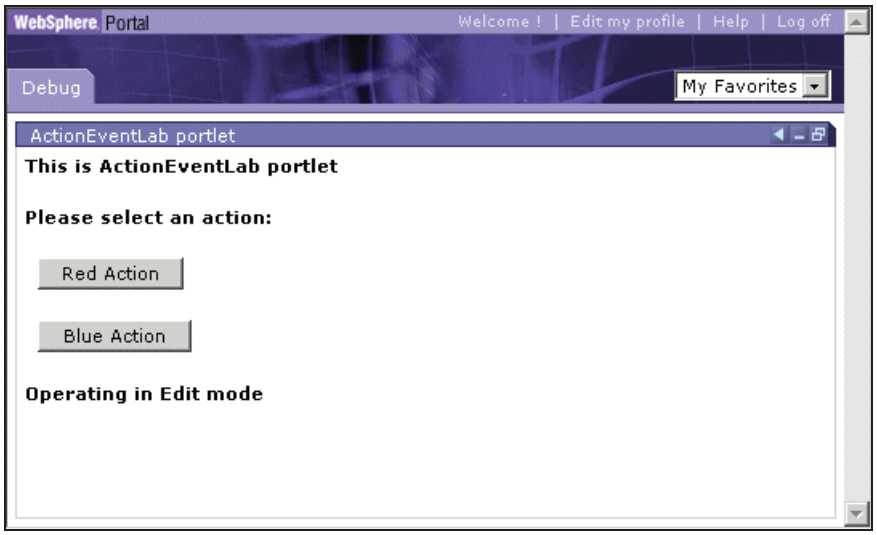


Figure 13-50 Portlet in Edit mode

Once the user selects an action (either Red or Blue), the portlet will be returned to view mode. This time, because an action was selected, the selected action will be written in the View JSP.

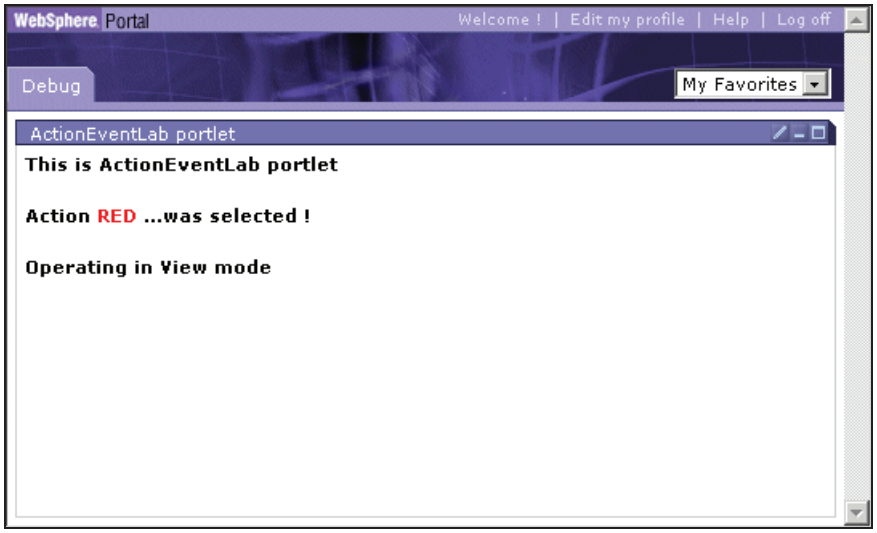


Figure 13-51 Portlet in View mode

To implement the portlet, a Portlet class called ActionPortlet was written. The portlet uses a JavaBean class called ActionPortletBean to store the name of the portlet. It invokes the JSP's view.jsp and edit.jsp to render the display in view and edit mode respectively.

The source listing for ActionPortlet.java is shown in Example 13-3.

Example 13-3 ActionPortlet.java

```
package portlet;

//**************************************************************************/
//*
//* ActionPortlet - A sample portlet based on PortletAdapter that implements
//* ActionListener
//*
//**************************************************************************/

import java.io.*;
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlet.event.*;

// In order to support action events ActionListener interface must be
implemented
public class ActionPortlet extends PortletAdapter implements ActionListener {

   public static final String ACTION_RED = "ACTION.RED";
```

```
   public static final String ACTION_BLUE = "ACTION.BLUE";

   public void actionPerformed(ActionEvent event) throws PortletException {
      // DefaultPortletAction is a portlet action with default parameters
      String action = (String) event.getActionString();

      // Check which action was selected
      if (action.equalsIgnoreCase(ACTION_RED)) {
         // Create a string to be rendered
         String value = "Action <FONT color=\"#ff0000\">RED</FONT>";
         // Create a Portlet request
         PortletRequest request = event.getRequest();
         // Save value in request
         request.setAttribute("action", value);
      }

      // Check which action was selected
      if (action.equalsIgnoreCase(ACTION_BLUE)) {
         // Create a string to be rendered
         String value = "Action <FONT color=\"#0000ff\">BLUE</FONT>";
         // Create a Portlet request
         PortletRequest request = event.getRequest();
         // Save value in request
         request.setAttribute("action", value);
      }
   }
   public void doView(PortletRequest request, PortletResponse response)
      throws PortletException, IOException {
      // Make a bean
      ActionPortletBean bean = new ActionPortletBean();
      // Save name in bean
      bean.setPortletName("ActionEventLab portlet");
      // Save bean in request
      request.setAttribute("ActionPortletBean", bean);

      // Invoke the JSP to render
      getPortletConfig().getContext().include("/jsp/View.jsp", request,
response);
   }

   public void doHelp(PortletRequest request, PortletResponse response)
      throws PortletException, IOException {
      // Make a bean
      ActionPortletBean bean = new ActionPortletBean();
      // Save name in bean
      bean.setPortletName("ActionEventLab portlet");
      // Save bean in request
      request.setAttribute("ActionPortletBean", bean);
```

```
      // Invoke the JSP to render
      getPortletConfig().getContext().include("/jsp/Help.jsp", request,
response);
   }

   public void doEdit(PortletRequest request, PortletResponse response)
      throws PortletException, IOException {
      // Make a bean
      ActionPortletBean bean = new ActionPortletBean();
      // Save name in bean
      bean.setPortletName("ActionEventLab portlet");
      // Save bean in request
      request.setAttribute("ActionPortletBean", bean);

      // Invoke the JSP to render
      getPortletConfig().getContext().include("/jsp/Edit.jsp", request,
response);
   }

   public void doConfigure(PortletRequest request, PortletResponse response)
      throws PortletException, IOException {
      // Make a bean
      ActionPortletBean bean = new ActionPortletBean();
      // Save name in bean
      bean.setPortletName("ActionEventLab portlet");
      // Save bean in request
      request.setAttribute("ActionPortletBean", bean);

      // Invoke the JSP to render
      getPortletConfig().getContext().include("/jsp/Configure.jsp", request,
response);
   }

}
```

The source listing for View.jsp is shown in Example 13-4.

Example 13-4 View.jsp

```
<%@ page contentType="text/html"%>

<jsp:useBean id="ActionPortletBean" class="portlet.ActionPortletBean"
scope="request" />

<B>This is <%=ActionPortletBean.getPortletName()%></B>
<P>
<% if (request.getAttribute("action") == null) { %>
   <B>No action performed, select your action in Edit Mode</B>
<% } else { %>
```

```
   <B><%= request.getAttribute("action") %> ...was selected !</B>
<% } %>
<P>
<B>Operating in View mode</B>
```

The source listing for Edit.jsp is as follows:

Example 13-5 Edit.jsp

```
<%@ page contentType="text/html"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ page import="portlet.*" %>

<jsp:useBean id="ActionPortletBean" class="portlet.ActionPortletBean"
scope="request"/>

<B>This is <%=ActionPortletBean.getPortletName()%></B>
<P>
<TABLE class="Portlet" border="0" cellspacing="0" cellpadding="0">
  <TR>
    <TD><B>Please select an action:</B>
    <FORM method='POST' action="<portletAPI:createReturnURI>
                                <portletAPI:URIAction

name='<%=ActionPortlet.ACTION_RED%>'/>
                                </portletAPI:createReturnURI>">
      <TABLE class="Portlet" border="0" cellspacing="0" cellpadding="0">
        <TR>
          <TD>  <INPUT type='submit' name='redButton' value='Red
Action'></TD>
        </TR>
      </TABLE>
    </FORM>

    <FORM method='POST' action="<portletAPI:createReturnURI>
                                <portletAPI:URIAction

name='<%=ActionPortlet.ACTION_BLUE%>'/>
                                </portletAPI:createReturnURI>">
      <TABLE class="Portlet" border="0" cellspacing="0" cellpadding="0">
        <TR>
          <TD>  <INPUT type='submit' name='blueButton' value='Blue
Action'></TD>
        </TR>
      </TABLE>
    </FORM>
    </TD>
  </TR>
</TABLE>
```

```
<B>Operating in Edit mode</B>
```

The source listing for portlet.xml is as follows:

Example 13-6 portlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portlet-app-def PUBLIC "-//IBM//DTD Portlet Application 1.1//EN"
"portlet_1.1.dtd">
<portlet-app-def>
   <portlet-app uid="ActionEventLab.b0fa2d48a0a600171aa6ab9a0e3409f9"
major-version="1" minor-version="0">
      <portlet-app-name>ActionEventLab application</portlet-app-name>
      <portlet id="Portlet_1" href="WEB-INF/web.xml#Servlet_1"
major-version="1" minor-version="0">
         <portlet-name>ActionEventLab portlet</portlet-name>
         <cache>
            <expires>0</expires>
            <shared>NO</shared>
         </cache>
         <allows>
            <maximized/>
            <minimized/>
         </allows>
         <supports>
            <markup name="html">
               <view />
               <edit />
            </markup>
         </supports>
      </portlet>
   </portlet-app>
   <concrete-portlet-app
uid="ActionEventLab.b0fa2d48a0a600171aa6ab9a0e3409f9.1">
      <portlet-app-name>ActionEventLab application</portlet-app-name>
      <concrete-portlet href="#Portlet_1">
         <portlet-name>ActionEventLab portlet</portlet-name>
         <default-locale>en</default-locale>
         <language locale="en">
            <title>ActionEventLab portlet</title>
            <title-short></title-short>
            <description></description>
            <keywords></keywords>
         </language>
      </concrete-portlet>
   </concrete-portlet-app>
```

```
</portlet-app-def>
```

Modifying the portlet for offline usage

The example portlet, as it stands, cannot be used offline, primarily because:

- The portlet uses PortletActions. PortletActions do not work with offline browsing.
- The portlet uses action buttons. Because of the way the offline server works during synchronization, it will result in both buttons being selected.
- The portlet does not have support for PDA markup. It currently only supports HTML markup.
- The HTML produced by the portlet does not entirely adhere to the rules of XML well-formedness.
- The portlet edit mode is not supported in offline browsing.

Refer to 13.6, “Development guidelines” on page 324 for explanations of these reasons.

Given the above reasons, we will need to modify the portlet as follows:

1. Replace the use of action buttons with a form utilizing radio buttons. The new form will not use PortletActions, since they are not supported for offline usage.
2. Since edit mode does not work with offline browsing, we will need to either combine the two screens (view mode and edit mode) into one screen or add a link to the view mode screen that will take you to the edit mode screen. For this example, we will combine the two screens. This makes the edit.jsp redundant.
3. Add PDA markup support in the portlet.xml file.
4. Modify the HTML in JSPs to ensure they adhere to the rules of XML well-formedness.
5. During deployment of the portlet, the JSPs will have to be moved to the highest possible directory structure, since the JSPs will now be used for both HTML and PDA markup. If you want to have separate JSPs for both HTML and PDA markup, you will need to store the HTML-specific JSPs under an `html` directory and the PDA-specific JSPs under a `pda` directory.

As a result of the above changes, the code now looks like Example 13-7.

Example 13-7 View.jsp

```
<%@ page contentType="text/html"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ page import="portlet.*" %>
```

```
<jsp:useBean id="ActionPortletBean" class="portlet.ActionPortletBean"
scope="request" />

<B>This is <%=ActionPortletBean.getPortletName()%></B>
<P />
<% if (request.getAttribute("action") == null) { %>
   <B>No action performed.</B>
<% } else { %>
   <B><%= request.getAttribute("action") %> ...was selected !</B>
<% } %>
<P />
<TABLE class="Portlet" border="0" cellspacing="0" cellpadding="0">
  <TR>
    <TD><B>Please select an action:</B>
    <FORM method='POST' action="<portletAPI:createReturnURI>
                                </portletAPI:createReturnURI>">
      <TABLE class="Portlet" border="0" cellspacing="0" cellpadding="0">
        <TR>
          <TD>  <INPUT type='radio' name='selectedAction'
value='<%=ActionPortlet.ACTION_RED%>' />Red Action</TD>
        </TR>
        <TR>
          <TD>  <INPUT type='radio' name='selectedAction'
value='<%=ActionPortlet.ACTION_BLUE%>' />Blue Action</TD>
        </TR>
        <TR>
          <TD>  <INPUT type='submit' name='submit' value='Select Action'
/></TD>
        </TR>
      </TABLE>
    </FORM>

    </TD>
  </TR>
</TABLE>
```

The View.jsp now includes the content of Edit.jsp. The action buttons that use to be on Edit.jsp are now implemented as a form with radio buttons. PortletActions have been removed and replaced with the request parameter `selectedAction`. Some of the HTML in the JSP has also been slightly modified to make it more compliant with the rules of XML well-formedness.

Example 13-8 ActionEvent.java

```
package portlet;
```

```
//*************************************************************************
***********
//*
//* ActionPortlet - A sample portlet based on PortletAdapter
//*
//*************************************************************************
***********/

import java.io.*;
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlet.event.*;

public class ActionPortlet extends PortletAdapter {

   public static final String ACTION_RED = "ACTION.RED";
   public static final String ACTION_BLUE = "ACTION.BLUE";

   public void doView(PortletRequest request, PortletResponse response)
      throws PortletException, IOException {

      // Get selected action from PortletRequest
      String action = (String) request.getParameter("selectedAction");

      if (action != null) {
         // Check which action was selected
         if (action.equalsIgnoreCase(ACTION_RED)) {
            // Create a string to be rendered
            String value = "Action <FONT color=\"#ff0000\">RED</FONT>";
            // Save value in request
            request.setAttribute("action", value);
         }

         // Check which action was selected
         if (action.equalsIgnoreCase(ACTION_BLUE)) {
            // Create a string to be rendered
            String value = "Action <FONT color=\"#0000ff\">BLUE</FONT>";
            // Save value in request
            request.setAttribute("action", value);
         }
      }

      // Make a bean
      ActionPortletBean bean = new ActionPortletBean();
      // Save name in bean
      bean.setPortletName("ActionEventLab portlet");
      // Save bean in request
      request.setAttribute("ActionPortletBean", bean);

      // Invoke the JSP to render
```

```
        getPortletConfig().getContext().include(
            "/jsp/View.jsp",
            request,
            response);

    }

    public void doHelp(PortletRequest request, PortletResponse response)
        throws PortletException, IOException {
        // Make a bean
        ActionPortletBean bean = new ActionPortletBean();
        // Save name in bean
        bean.setPortletName("ActionEventLab portlet");
        // Save bean in request
        request.setAttribute("ActionPortletBean", bean);

        // Invoke the JSP to render
        getPortletConfig().getContext().include(
            "/jsp/Help.jsp",
            request,
            response);
    }

    public void doEdit(PortletRequest request, PortletResponse response)
        throws PortletException, IOException {
        // Make a bean
        ActionPortletBean bean = new ActionPortletBean();
        // Save name in bean
        bean.setPortletName("ActionEventLab portlet");
        // Save bean in request
        request.setAttribute("ActionPortletBean", bean);

        // Invoke the JSP to render
        getPortletConfig().getContext().include(
            "/jsp/Edit.jsp",
            request,
            response);
    }

    public void doConfigure(PortletRequest request, PortletResponse response)
        throws PortletException, IOException {
        // Make a bean
        ActionPortletBean bean = new ActionPortletBean();
        // Save name in bean
        bean.setPortletName("ActionEventLab portlet");
        // Save bean in request
        request.setAttribute("ActionPortletBean", bean);

        // Invoke the JSP to render
```

```
      getPortletConfig().getContext().include(
         "/jsp/Configure.jsp",
         request,
         response);
   }

}
```

You will notice that the Portlet class no longer implements ActionListener, since we are not using PortletActions anymore. As a result, the actionPerformed method has been removed. Any logic performed in this method has been moved to the beginning of the doView method. Since we are not using PortletActions, the selected action is now stored as the request parameter `selectedAction`. The parameter is pulled out of the request and inspected at the beginning of the doView method.

Example 13-9 Portlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portlet-app-def PUBLIC "-//IBM//DTD Portlet Application 1.1//EN"
"portlet_1.1.dtd">
<portlet-app-def>
   <portlet-app uid="ActionEventLab.b0fa2d48a0a600171aa6ab9a0e3409f9"
major-version="1" minor-version="0">
      <portlet-app-name>ActionEventLab application</portlet-app-name>
      <portlet id="Portlet_1" href="WEB-INF/web.xml#Servlet_1"
major-version="1" minor-version="0">
         <portlet-name>ActionEventLab portlet</portlet-name>
         <cache>
            <expires>0</expires>
            <shared>NO</shared>
         </cache>
         <allows>
            <maximized/>
            <minimized/>
         </allows>
         <supports>
            <markup name="html">
               <view />

            </markup>
            <markup name="pda">
               <view />

            </markup>
         </supports>
      </portlet>
   </portlet-app>
```

```
   <concrete-portlet-app
uid="ActionEventLab.b0fa2d48a0a600171aa6ab9a0e3409f9.1">
      <portlet-app-name>ActionEventLab application</portlet-app-name>
      <concrete-portlet href="#Portlet_1">
         <portlet-name>ActionEventLab portlet</portlet-name>
         <default-locale>en</default-locale>
         <language locale="en">
            <title>ActionEventLab portlet</title>
            <title-short></title-short>
            <description></description>
            <keywords></keywords>
         </language>
      </concrete-portlet>
   </concrete-portlet-app>
</portlet-app-def>
```

The main changes to portlet.xml are the addition of PDA markup support and the removal of view mode support.

Once we have modified and tested the portlet, we can then add it to the offline page. After synchronization, the page should look like Figure 13-52 on a client device.

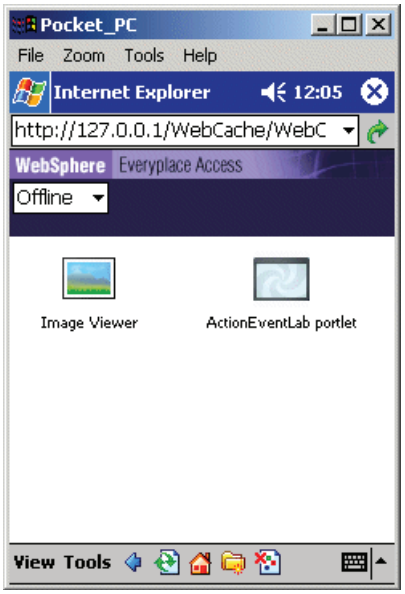


Figure 13-52 The offline page viewed on a Pocket PC

Clicking the **ActionEventLab** portlet icon will take you into the portlet. The screen will show the last selected action. In this case, it will say "`No action`

performed", since it is the first time we have viewed this screen. Select an action and click **Select Action**.

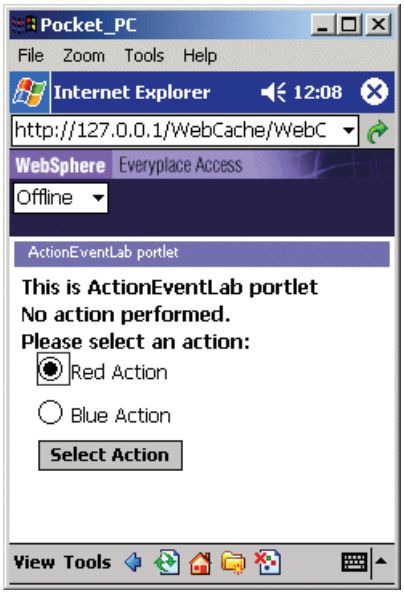


Figure 13-53 ActionEventLab portlet

You will then get a message indicating that your request has been queued and will be submitted the next time you synchronize.

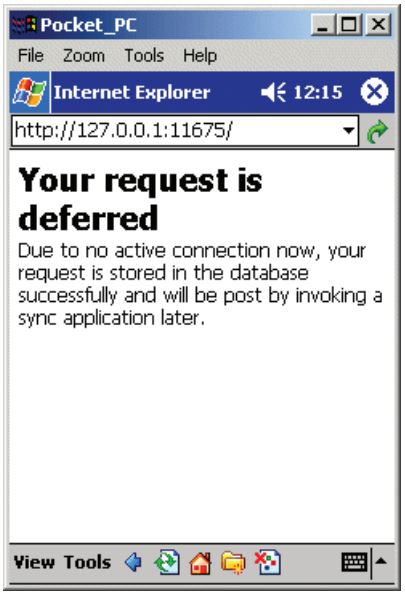


Figure 13-54 Request deferred message

When you return to the Everyplace Client Home page, you will see that there is one pending form under offline Forms.

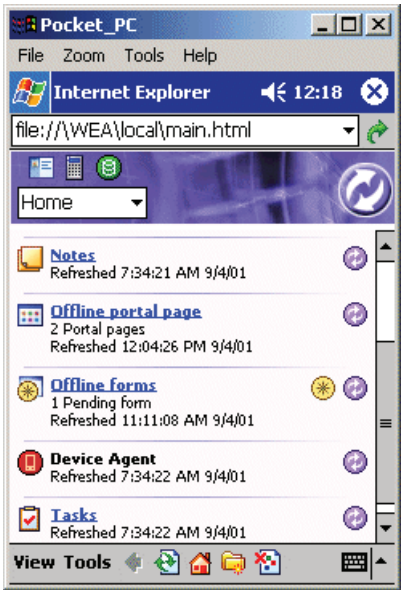


Figure 13-55 Everyplace Client Home page

You need to synchronize offline forms by clicking the **Synchronize** icon on the right. Once synchronization is complete, click **Offline forms** and you should see the status of the submission, as illustrated in Figure 13-56.

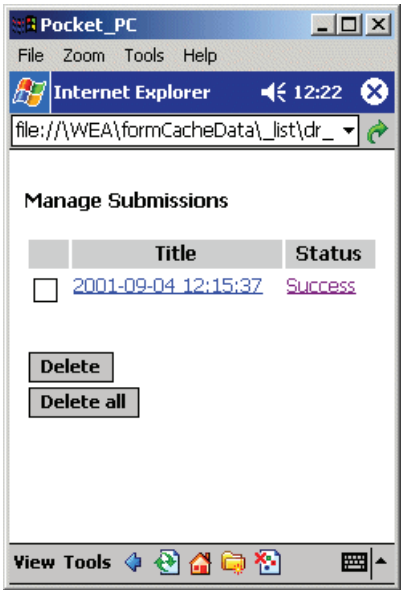


Figure 13-56 Submission status

You can then view the response from the submission by clicking the link.

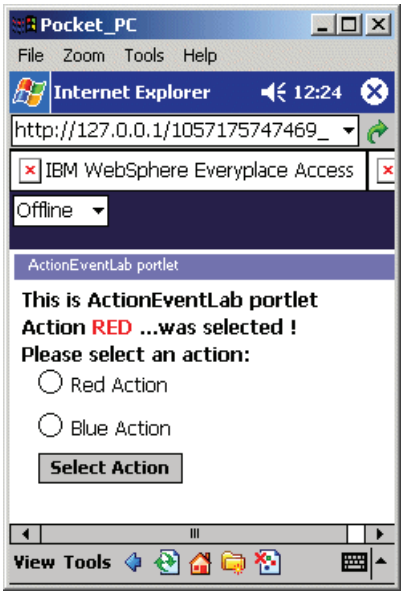


Figure 13-57 Response of form submission

13.8 Hints and tips

In this section, we include some helpful recommendations to delete users and change a PDA icon.

13.8.1 Deleting users

Offline Portal Content information is stored in a separate database table from the Portal user information. When a Portal user is deleted, Everyplace Access does not delete the user information from this Offline Portal Content database table. If you are deleting large numbers of users and you wish to conserve space, you should delete the users from the Portal first and then delete the users from this table. The database table information is as follows:

- Database name: WEASDB
- Table name: OFFLINE_CFG

Search on one of the following to remove users:

- `OBJECTID`
- `USERID`

Note: You should be familiar with DB2 before attempting to delete user information from this table.

13.8.2 Changing PDA icon

The portlets on the offline Portal page appear in the form of icons when viewed offline. When a new portlet is added to this offline page, a generic icon is assigned to the portlet. To change the icon, add a configuration parameter named `pda-icon` and point it to an image under the document root of the Portal Server. See Example 13-10.

Example 13-10 Adding custom PDA icons in portlet.xml

```
...
<concrete-portlet-app uid="WorldClockPortlet.concreteportletapp">
  ...
  <concrete-portlet href="#WorldClockPortlet">
    ...
    <config-param>
      <param-name>pda-icon</param-name>
      <param-value>/images/pda/worldClock.gif</param-value>
    </config-param>
    ...
  </concrete-portlet>
</concrete-portlet-app>
...
```

The directory in which to place your custom icon image is <*wps_home*>\app\wps.ear\wps.war\images\pda, where <*wps_home*> represents your WebSphere Portal directory.

For Portlets created using the Web Clipping functionality, there are no portlet.xml files to modify. You will need to add the configuration parameter directly using the Portal Administration features of Everyplace Access:

1. Log in as `wpsadmin`.
2. Select **Portal Administration -> Manage Portlets**.
3. Locate the created Web clipper and select **Modify Parameters**.

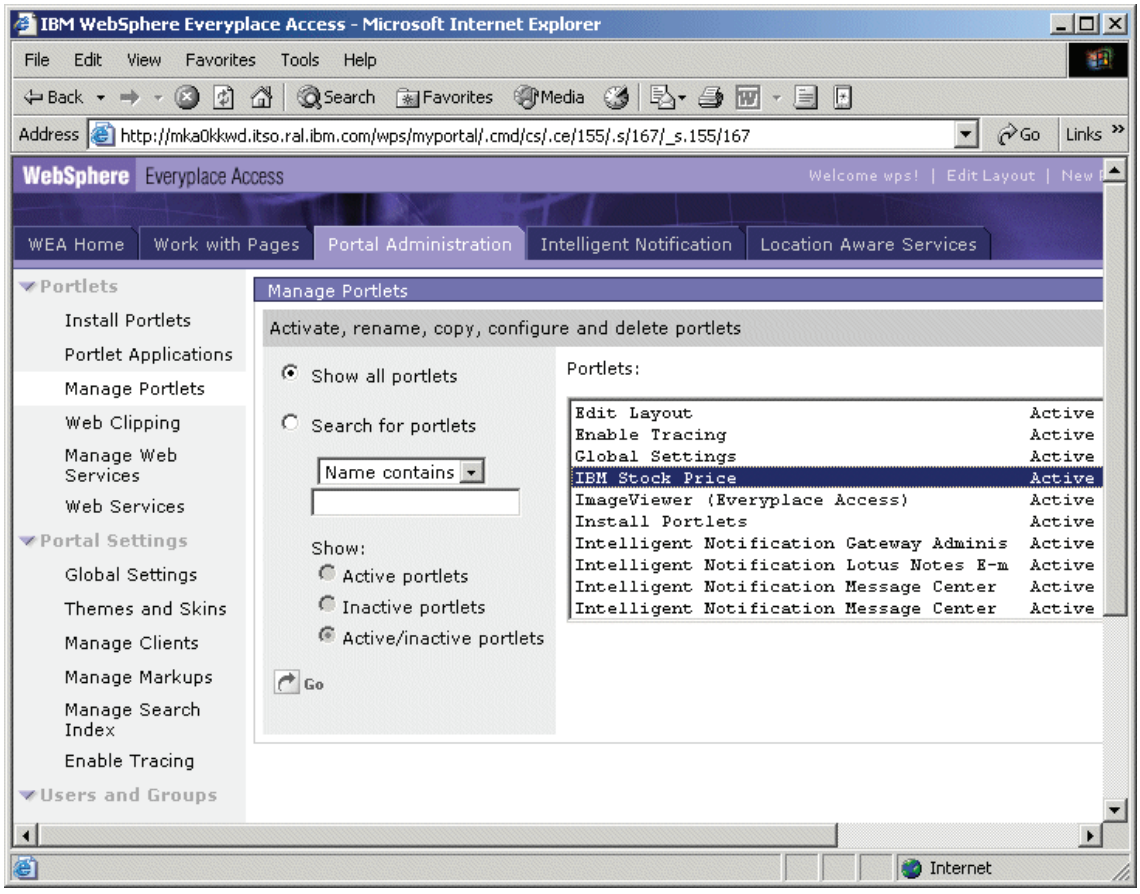


Figure 13-58 Modify parameters for web clipper

4. Enter the parameter name and value and then click **Add**.

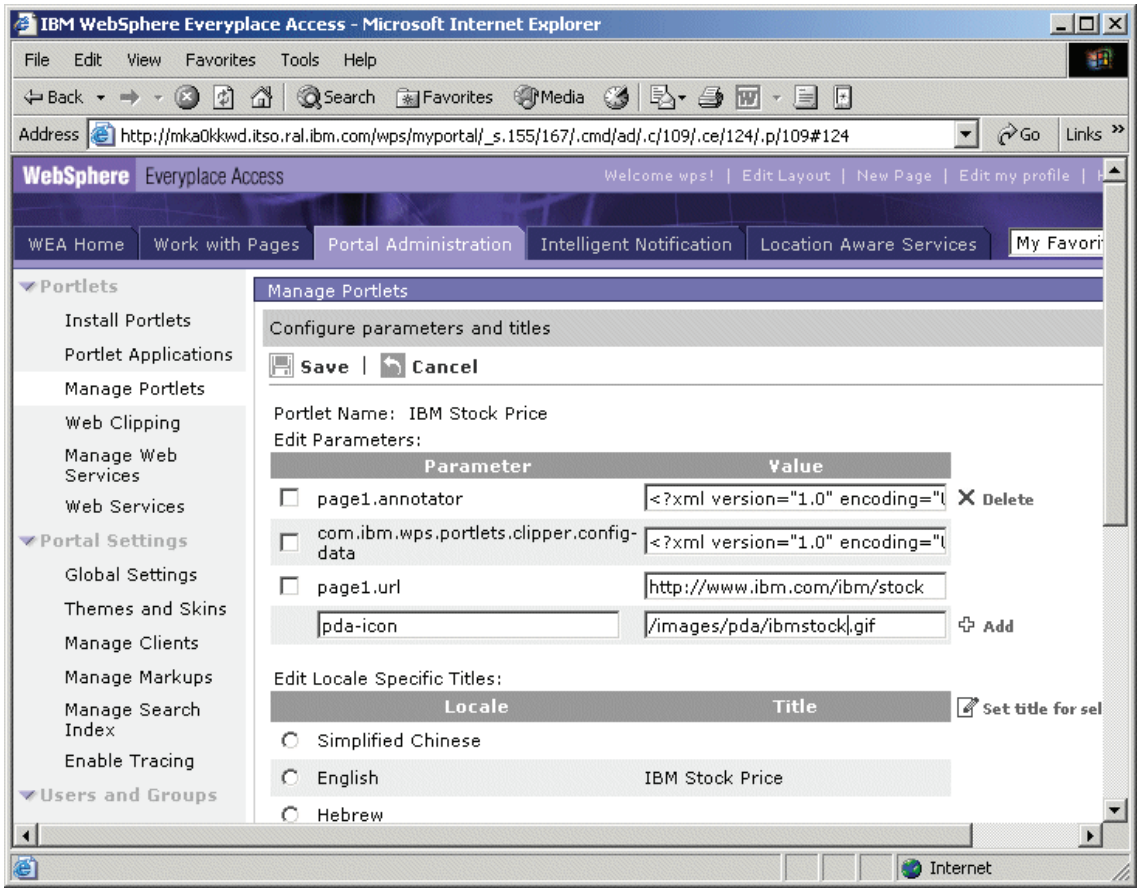


Figure 13-59 Add configuration parameter

5. The new parameter should now appear in the parameter list. Click **Save**.

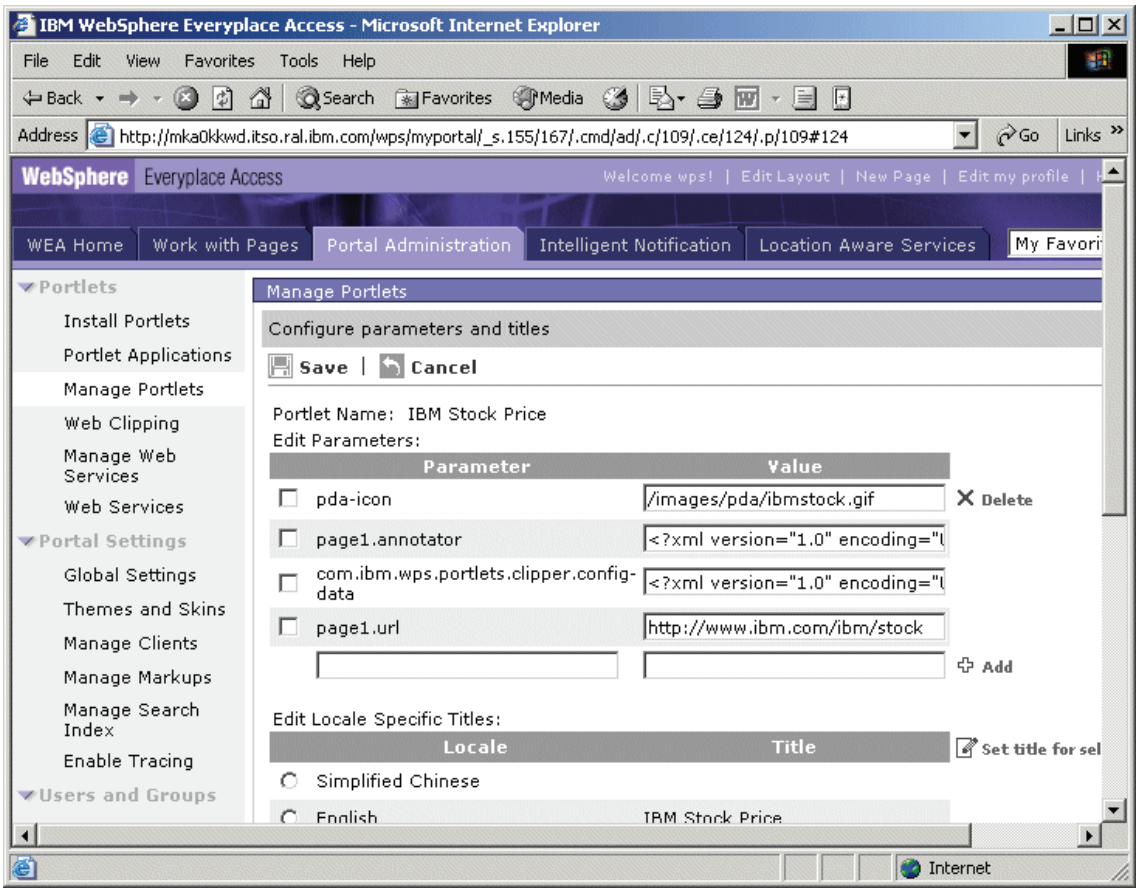


Figure 13-60 Click Save

6. The icon for the clipped portlet should now appear in the offline page viewed on the client device.

13.9 Resources

- For more information on developing offline portlets, refer to Michael Wanderski's article at:

  `http://www.software.ibm.com/wsdd/techjournal/0303_wanderski/wanderski.html`

14

Transcoding Technology

This chapter provides an introduction to Transcoding Technology, which is part of WebSphere Everyplace Access, and presents an overview of the relevant tools and transcoding options available.

Important: To ensure you have the latest code, you should visit the WebSphere Everyplace Access support Web site at http://www.ibm.com/software/pervasive/ws_everyplace_access/support/ to download the latest fix packs and code fixes. Apply these fixes prior to using the samples within this redbook.

© Copyright IBM Corp. 2003. All rights reserved.

14.1 Overview

Transcoding Technology, included in WebSphere Everyplace Access, provides the ability to tailor Web-based information to the needs of mobile users and their mobile devices. For example, Web pages targeted for the desktop browser can be adapted for viewing on mobile devices, such as PDAs.

With Transcoding Technology, users receive information tailored to the capabilities of the devices they are using. For example, users with small-screen devices access a re-scaled version of the information, while users with devices that require a specialized markup language can access the same information in the format suitable for their device. By providing a single dissemination point for multiple renderings of information, Transcoding Technology eliminates the expense of re-authoring or modifying applications to handle multiple networks and devices. In essence, Transcoding Technology extends the reach of existing information to a new class of users with mobile devices.

Transcoding Technology transforms content based on the information associated with the request, such as device constraints and administrative policies. Also, Web content can be transformed differently to meet the characteristics of the specific devices. Transcoding Technology can support all common types of Web data, including HTML pages and Extensible Markup Language (XML) documents. Transcoding Technology also tailors images by adjusting to screen size, file size, and the color pallet available in the target device.

Transcoding Technology offers three ways to transform content:

- XML stylesheets
- Annotators
- Transcoding plug-ins

These transformation details are also referred to as *resources* in Transcoding Technology. There is another type of resource called *preference profiles*, which are used to represent the characteristics of devices and users. Profiles can be used to determine which stylesheet, annotator, or plug-in to use in a given situation.

Note: In the following sections, the root directory where Transcoding Technology is installed is referred to as TT_Root. You should replace this with your actual directory name, for example C:\WebSphere\PortalServer\IBMTrans\.

Transcoding Technology helps you ensure the Web data delivered to the devices is appropriate for the device and meets your administrative policy, which can consist of:

- Size: Images are scaled for presentation on the device and text is compressed, which could in effect reduce the size of the file that is transmitted to the device.
- Detail: Content tailored to the specific amount and kind of information your users need and the device can accommodate.
- Format: Content presented in the format supported by your users' device type(s).
- Language: Data is translated into a user's language.

14.1.1 Architecture

Transcoding Technology in WebSphere Everyplace Access is invoked at two different levels:

- Portlet-level transcoding. This is used for content transformations. For example, markup language conversions, stylesheets, and annotators are applied at this level.
- Portal page-level transcoding. This is used for whole-page modifications, for example content fragmentation.

Figure 14-1 on page 380 illustrates Transcoding Technology as implemented in IBM WebSphere Everyplace Access Version 4.3.

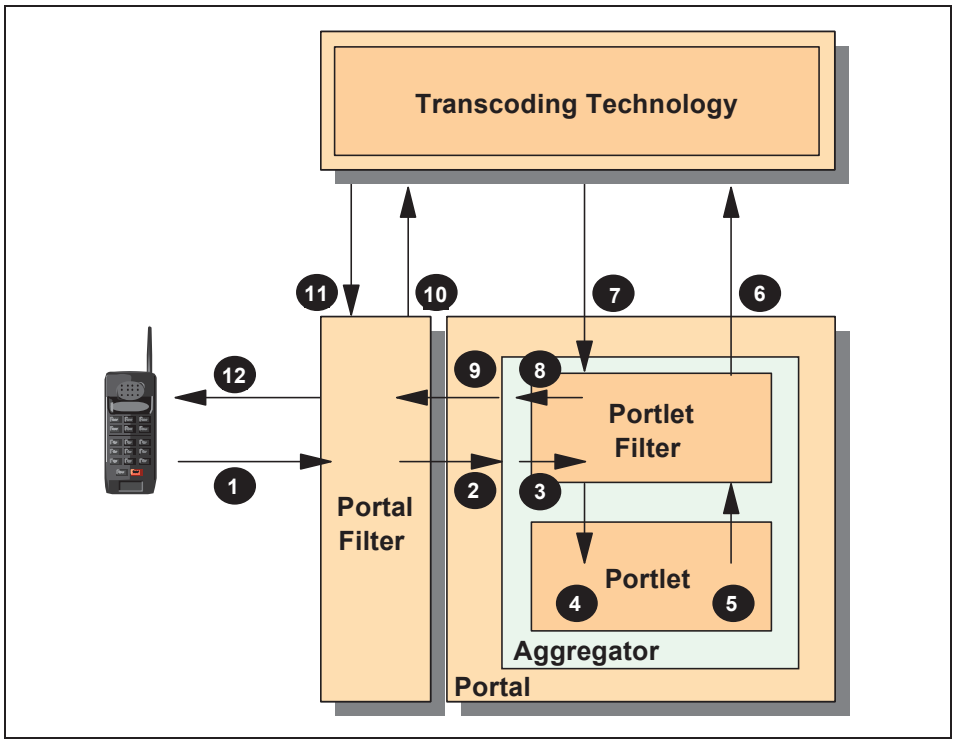


Figure 14-1 Transcoding Technology in WebSphere Everyplace Access

The request flow is explained as follows:

1. The user agent (WAP browser, Pocket PC browser, or desktop browser) makes a request to the portal. In this environment, the request is received by the Portal filter.
2. The Portal filter calls the portal to receive its contents, which calls the aggregator to aggregate the page.
3. The aggregator selects portlets based on a combination of values, such as user credentials and whether the portlet supports the current aggregator. If a portlet is configured for transcoding and it provides a markup that Transcoding Technology can use, it will be selected. For example, if the portlet provides HTML and the client needs WML content, the aggregator will select the portlet because the portlet content must be transformed from HTML to WML.

   Therefore, for a portlet that is configured to use Transcoding Technology, the aggregator calls the portlet filter.
4. The portlet filter calls the portlet to receive its contents.
5. The portlet returns its contents.

6. The portlet filter sends the portlet's contents to Transcoding Technology for processing, associating the portlet's contents with a default URI, unless otherwise specified. This step is known as *portlet-level transcoding*.

   Note that the portlet thinks it is communicating directly to the aggregator, so no special programming style is needed to write a portlet that is filtered.

7. Transcoding Technology returns the processed contents to the portlet filter.
8. The portlet filter returns the processed contents as if it were a proper portlet returning contents normally.
9. The aggregator, having finished aggregating the page, returns the page.
10. The Portal filter sends the Portal's aggregated contents to Transcoding Technology for processing. This step is known as *portal-level transcoding*.
11. Transcoding Technology returns the processed contents.
12. The Portal filter returns the processed contents to the target device.

14.1.2 Preference profiles

Transcoding Technology uses preference profiles to represent the characteristics of a particular device, a group of devices, a user, or user group. Preference profiles are used to decide how to treat documents that will be delivered to different devices and different users. For example, on a device with a small screen, it is desirable to convert tables to lists to reduce horizontal scrolling. A particular user may request that images are eliminated from the content viewed on their device.

When Transcoding Technology processes a document, a device profile and a user profile are selected to apply to that document.

Device preference profiles are represented by .prop files located in <TT_Root>\etc\preferences\device.

User preference profiles (a default profile is included) are represented as .prop files located in <TT_Root>\etc\preferences\user.

If the X-IBM-PVC-Device-Type field is present in the HTTP header, Transcoding Technology uses the device profile whose file name matches the value specified for that field. The value of the X-IBM-PVC-Device-Type field is set by the Portal's preference aggregation.

For example, if the value of X-IBM-PVC-Device-Type is Microsoft!Internet+Explorer!6.0, the <TT_Root>\etc\preferences\device\Microsoft!Internet+Explorer!6.0.prop device preference profile will be used. Figure 14-2 illustrates the properties file for Microsoft Internet Explorer Version 6.

```
#version = 1.0
#Sat Jun 07 08:49:18 EDT 2003
framesSupported=true
deviceRule=(User_Agent%e*MSIE 6.0*)
javaAppletsSupported=true
portalOrdinal=360
portalMarkupVersion=ie
createCHTML=false
portalClient=true
parent=NT.InternetExplorer
desiredContentTypes=[text/html]
javaScriptSupported=true
```

Figure 14-2 Microsoft!Internet+Explorer!6.0.prop

If the X-IBM-PVC-Device-Type field is not present in the HTTP header, Transcoding Technology uses the device profile whose user-agent value matches the value of the user-agent field in the HTTP header. If no matching profile is found, Transcoding Technology uses the default device profile.

User resources are not maintained through Transcoding Technology. However, other programs can pass information to Transcoding Technology specifying the use of a user profile. A user profile is selected for a request in the following way. If a value is specified for userAndSessionExtractor in <TT_Root>/etc/localConfig.prop, Transcoding Technology tries to execute the referenced implementation of the UserAndSessionExtractor interface to obtain user and session names and select the user profile that matches the user name.

You can specify a field in the HTTP header to be used to select a user profile by setting the httpUserIdField value in <TT_Root>/etc/localConfig.prop. For example, if you specify httpUserIdField=X-UserField, then Transcoding

Technology would find the value of the X-UserField keyword in the HTTP header and select the user profile that matches that value.

If the X-IBM-PVC-User field is present in the HTTP header, Transcoding Technology uses the user profile that matches the value specified for that field.

If none of these checks identify a user profile, then Transcoding Technology does not use a specific user profile. If one of these methods is used to specify a user profile and the specified file is not found, Transcoding Technology does not try the other methods; it does not use a specific user profile. For example, if you specify httpUserIdField=myUserField but myUserField is not specified in the HTTP header, Transcoding Technology would not look for the X-IBM-PVC-User field.

If more than one device preference profile matches the incoming request, it is impossible to predict which of the matching profiles would be selected. For example, if you create two device profiles with expressions that could match the same user-agent value, either one could be selected for a request that specifies that user-agent value. Be sure to specify unique values for any new profiles you create.

Each profile contains values for the preferences that are important to the device that the profile represents. If a preference is not important to the device, it can be omitted so that a value can be chosen from a different profile. For most preferences, Transcoding Technology will check profiles for a value in this order:

1. Specific user
2. Specific device
3. Default user
4. Default device

If a value is not specified for a preference in one profile, Transcoding Technology will work down the list until a value is found. The transcoding plug-ins that will be applied to the document are selected based on the combined profiles.

Transcoding Technology provide profiles for several common pervasive devices. There are default profiles to be used if none of the existing profiles matches the device being used.

In addition to determining which transcoding plug-ins will be used, a profile can be used to select a stylesheet or annotator. You can also specify parameters to be used by stylesheets that accept parameters.

14.1.3 XML stylesheets

When Transcoding Technology process documents represented in Extensible Markup Language (XML), it uses XSL stylesheets to convert these documents to another markup language, such as HTML, WML, or other forms of XML.

Stylesheets are associated with the portlet emitting the XML by either adding a config-param parameter to the portlets portlet.xml file or by adding parameters to the portlet by using Portal Administrations Manage Portlet (modify parameters).

Multiple stylesheets can be associated with an XML document, by including the wtp-config processing instruction within the target XML document. One of the stylesheets is selected to process against the XML document based on the values of one or more fields in the HTTP header or the DTD associated with the XML document. A stylesheet can also be selected based on criteria that match a preference profile.

Some stylesheets accept parameters that affect how they operate on documents. A stylesheet that accepts parameters can retrieve values for the parameters from:

- The HTTP header of the request
- Values specified for the Parameters field in the WTPResources file
- Values specified in a preference profile

Stylesheets will be discussed in more detail in Chapter 16, “Using XSL transformations and XML tools” on page 441.

14.1.4 Annotators

For pervasive devices with limited screen size, it is often desirable to show only the key information from a Web page, and get rid of the less critical information such as graphics and extra content. Annotators can be used to specify which portions of a Web page to include or discard when the Web page is transcoded.

There are two types of annotators:

- Internal annotators

  These are annotation instructions that are created using the Page Designer tool available in WebSphere Studio Application Developer and WebSphere Studio Site Developer. Internal annotation instructions consist of special tags embedded in the Web page. When the page passes through Transcoding Technology, these tags are examined and actions are taken according to the particular instruction(s). For details about and samples of internal annotators, see Chapter 15, “Using annotation to clip HTML documents” on page 403.

- External annotators

  These annotation instructions are contained in a separate and independent annotation file that the Transcoding Technology uses to operate on the target Web content. External annotation files can be created with simple text editors, or by using the HTML Annotation Editor supplied with the Everyplace Toolkit (a plug-in for WebSphere Studio).

  External annotation files can be packaged with a portlet .war file. The portlet.xml file can contain the configuration parameters to support annotation. For details about and samples of external annotation, see Chapter 15, “Using annotation to clip HTML documents” on page 403.

14.1.5 Transcoding plug-ins

A transcoding plug-in is a program written with programming languages that modifies the content of a document. Transcoding plug-ins are selected to process a document based on conditions specified by the program when the transcoding plug-in is created.

Several transcoding plug-ins are provided with Transcoding Technology in WebSphere Everyplace Access, and you can obtain or develop others:

- The image transcoding plug-in modifies images to better support the display capability of a device.
- The text transcoding plug-in converts textual data, such as HTML or XML, from one format to another and can perform a number of transformations to simplify the output.
- The fragmentation transcoding plug-in fragments XML documents into pieces small enough to be managed by the target device.
- The HTML DOM generator creates a Document Object Model (DOM) version of incoming HTML documents.
- The annotation transcoding plug-in, also called annotation engine, interprets the contents of files written with Transcoding Technology annotation language to perform document clipping.
- The HTML to WML transcoding plug-in converts HTML documents to WML for devices with WAP browsers.
- The HTML to compact HTML transcoding plug-in converts HTML documents to Compact HTML documents for devices with CHTML browsers.

These transcoding plug-ins are installed with the product and are enabled by default (with the exception of the HTML DOM generator). To verify whether the transcoding plug-ins are enabled or to change their status, use the XML configuration utility to export the configuration for reviewing.

14.2 XML configuration utility

The Transcoding Administrative Console (part of the WebSphere Transcoding product) is not available with Everyplace Access. Transcoding Technology provides the XML configuration utility, which provides a set of commands (entered on the command line) that perform the following tasks:

- Export resources: Used to export the resource definitions to an XML document from transcoding resources
- Import resources: Used to import the resource definitions from an XML document into the transcoding resources
- Disable resources: Disables a resource that is currently active in the transcoding resources
- Enable resources: Enables a resource that is currently part of the transcoding resources
- ImportCocoon: Imports a Cocoon properties file to support stylesheet specification within XML documents

Note: Resources such as device profiles, stylesheet registrations, and external annotator registrations can be configured using this utility.

The configuration file can be modified outside of Transcoding Technology, which allows adjustments to the configuration as needed. Figure 14-3 on page 387 shows the XML configuration management consisting of the following steps:

1. Use the ExportResources command to read the existing Transcoding Technology configuration.
2. A Transcoding Technology XML configuration file (WTPResources.xml) is created in the specified directory.
3. The WTPResources.xml file can be modified as needed.
4. Use the ImportResources command to write the modified WTPResources.xml into Transcoding Technology.

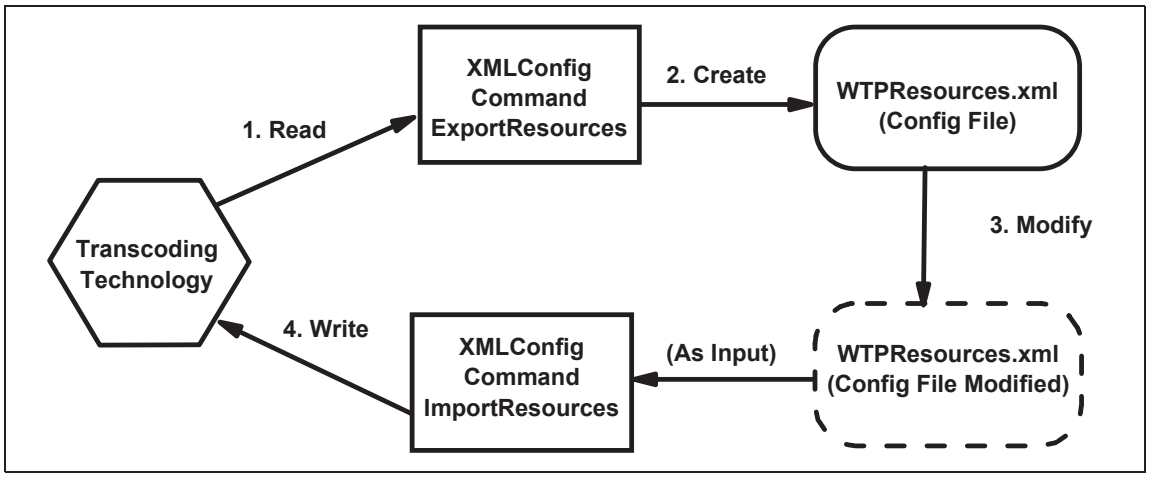


Figure 14-3 XML Configuration management steps

For an example of how to use these commands, see “Preparation steps” on page 390.

14.3 Request Viewer

Request Viewer is a very useful visual tool for monitoring the traffic going through Transcoding Technology. The Request Viewer is included in the Everyplace Toolkit for use in debugging your annotation files and styesheets. You can use Request Viewer to view the configuration and status information of the registered transcoding plug-ins within Transcoding Technology.

Request Viewer is particularly handy as a debugging tool, because it enables you to examine the flow of requests and responses through the server and observe which plug-ins are triggered and when they are triggered. For each transaction, Request Viewer also displays the header and content information as they are manipulated by the transcoding plug-ins.

The primary window of Request Viewer consists of these parts:

1. Menu bar: Provides access to the application options for Files, Actions, Requests and Help. Menu options can be disabled when the action is not available.
2. View options: Two tabs, each of which specifies a particular view selection. One is for the Server Configuration and the other is the Request Processing view. When either of these views is selected, the associated content is displayed in the main window. The Server Configuration view provides the

transcoding server information, including the transcoders registered, preference profiles, and so on. The Request Processing view shows the flow of the request through the server.

3. The Server Configuration view (shown in Figure 14-4) subdivides the window into three parts:

   a. Server Configuration: A list view of the contents of this configuration (including sublayers and transcoding plug-ins). This is found in the top-left pane.

   b. Details: Displays the details of an item within the configuration. Content is displayed in this pane when an item is selected in the Server Configuration tree view. This is found in the top-right pane, across from the Server Configuration.

   c. Output Messages: Displays messages generated at execution time. This is found across the bottom of the window, below the Server Config and Details panes.

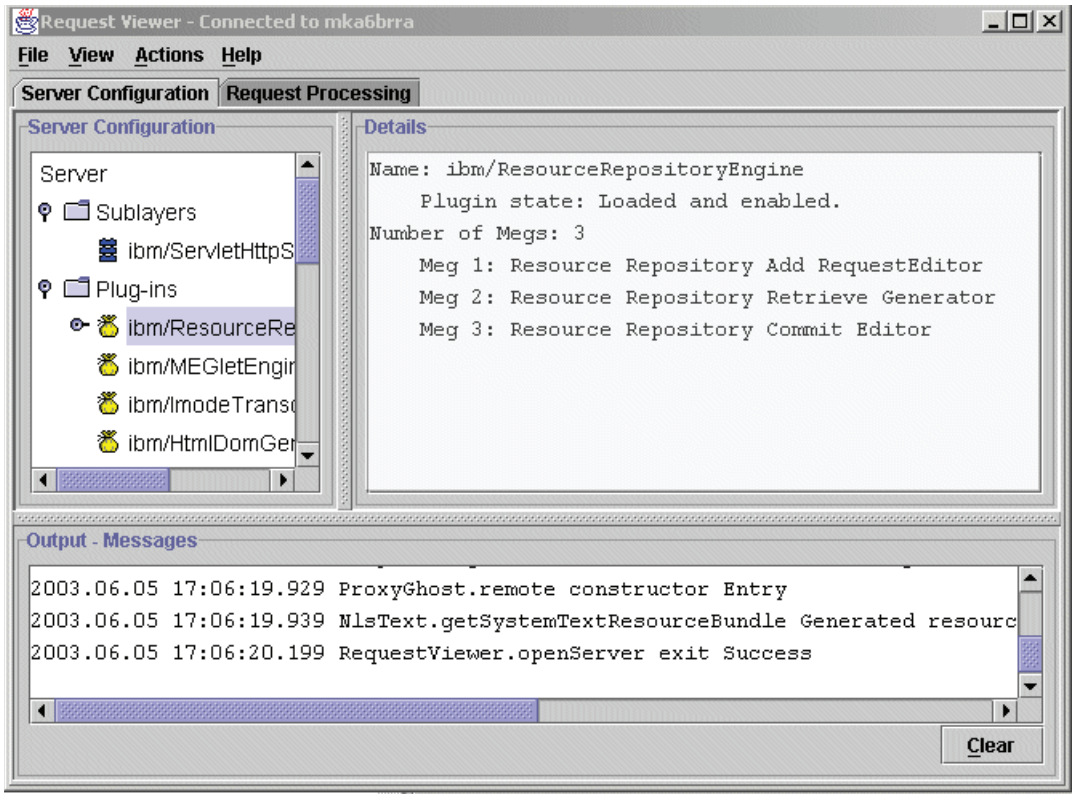


Figure 14-4 Request Viewer server configuration

4. The Request Processing view (shown in Figure 14-5) subdivides the window into three parts:
   a. Request Processing: A tree view of the flow of requests through Transcoding Technology (in the top-left pane).
   b. Transaction Header: Displays the contents of the HTTP header. The content is displayed in this pane when an item is selected in the Request Processing tree view. This is found in the top-right pane, across from Request processing.
   c. Transaction Content: Displays a particular transcoder's input or output (below the Transaction Header).

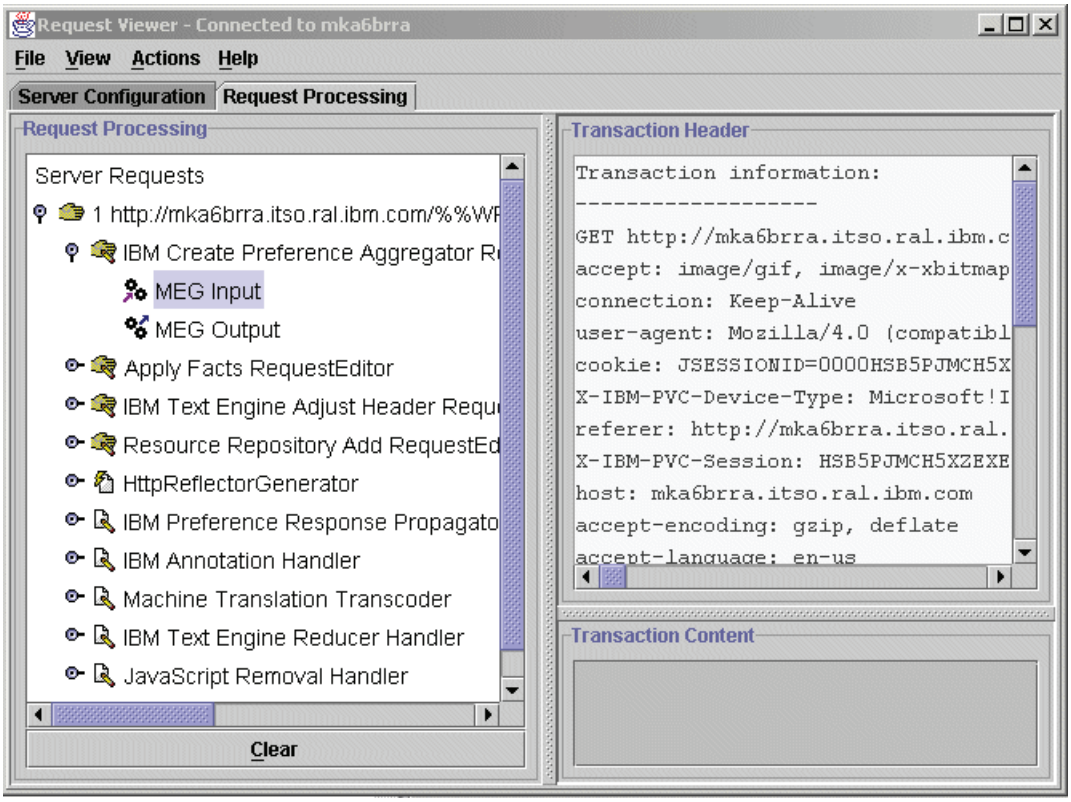


Figure 14-5 Request Viewer request processing

For more details on the Request Viewer, see the *WebSphere Transcoding Publisher Developer Guide V4.0* found at:

`http://www.ibm.com/software/webservers/transcoding/library.html`

14.3.1 How to start Request Viewer

The Request Viewer is now included in the Everyplace Toolkit, which is a plug-in for the WebSphere Studio offering (currently WebSphere Studio Site Developer and WebSphere Studio Application Developer products). We are using WebSphere Studio Site Developer with the Everyplace Toolkit installed.

Preparation steps

Prior to starting Request Viewer in WebSphere Studio, you must perform these steps on the Everyplace Access server:

1. Use the Services to stop the IBM WebSphere Application Server and the IBM Everyplace Sync Server.
2. Start a DOS prompt and change the working directory to the directory where Transcoding Technology is installed. In our case it was installed in the default directory of C:\WebSphere\PortalServer\IBMTrans.
3. You must change the transcoding configuration to enable RMI registry and enable two of the transcoding plug-ins, DatabaseWatcherPlugin and ViewerServerPlugin. To do this perform the remaining steps.
4. Run the ExportResources command to export the resources that must be modified. The specific command is:

   ```
   ExportResources -rt setup -rt
   plugin,DatabaseWatcherPlugin,ViewerServerPlugin
   ```

5. A WTPResources.xml file is placed in the transcoding directory (IBMTrans). Open this file for editing and make the following changes:

   a. For both the DatabaseWatcherPlugin and the ViewerServerPlugin plug-ins (which are defined as <Plugin> elements) add the statement shown in Example 14-1 before the </Plugin> tag, to enable the plug-ins.

Example 14-1 Enable the plug-ins

```
<Plugin>
.............
<Enable>true</Enable>
</Plugin>
```

   b. Enable the RMI Registry, which is the <RMIRegistry> element, as shown in Example 14-2.

Example 14-2 Enable RMI Registry

```
<RMIRegistry>
.....
<Enable>true</Enable>
```

```
</RMIRegistry>
```

c. Enable the Remote Request Viewer and authorize the system where WebSphere Studio is running by adding the statements shown in Example 14-3 to the <RemoteRequestViewer> element.

Example 14-3 Enable and authorize RemoteRequestViewer

```
<RemoteRequestViewer>
.....
<Enable>true</Enable>
<AuthorizedHostName>MachinewithWebSphereStudio.myco.com</AuthorizedHostName>
</RemoteRequestViewer>
```

d. Save your changes and close the file.

6. Copy the ViewerServerPlugin.jar from the plug-ins folder (in the IBMTrans directory) as ibm_ViewerServerPlugin.jar to the addedPluginJars folder in the IBMTrans directory.
7. Import the changed WTPResources.xml file back into Transcoding Technology by using the ImportResources command.
8. Open the wtpscripts.jar and extract the StartRegistry.bat script into the IBMTrans directory.
9. Always start the RMI registry before starting the WebSphere Application Server or the Everyplace Sync Server. This is done by running the StartRegistry command.
10. Now you can restart the IBM WebSphere Application Server and the IBM Everyplace Sync Server.

Starting Request Viewer

With WebSphere Studio started and an existing portlet project active, you can launch the Request Viewer by selecting **File -> Transcoding Tools -> Launch Request Viewer** as shown in Figure 14-6 on page 392.

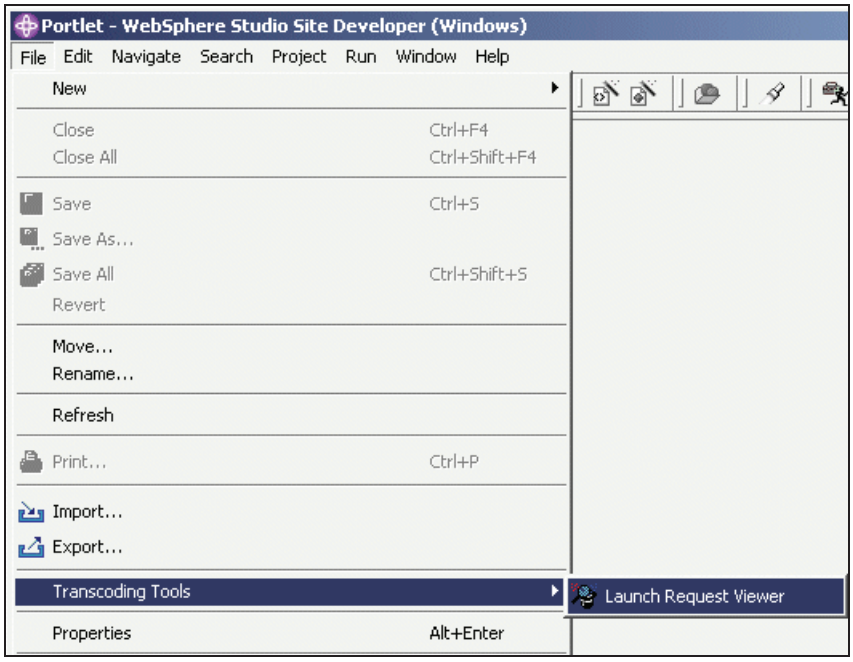


Figure 14-6 Launch Request Viewer

Once the Request Viewer starts, a window pops up asking for information about the machine where Transcoding Technology is running. We are running remote to the Everyplace Access server. Select the **Remote** radio button and in the Host name field enter the remote machine name. Next, select the **Defined in a remote RMI registry** radio button. Both the host name of RMI server and the port number of the RMI server are supplied for you. This window and our entries are shown in Figure 14-7 on page 393. Click **OK** to finish the process.

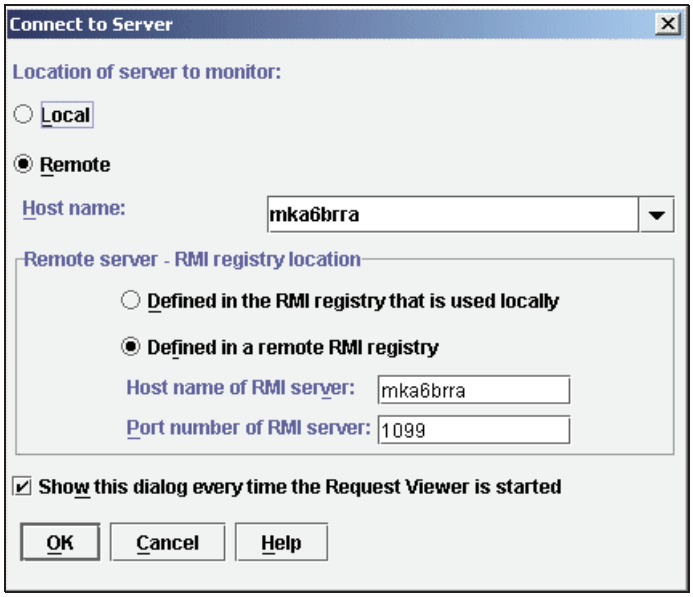


Figure 14-7 Connect to Server

It may take a few seconds for Request Viewer to start. Once it starts you are ready to track requests and view the device-to-server interaction.

Note: Request Viewer is a monitoring tool; therefore, you cannot change the configuration or status of the transcoding plug-ins with it.

14.4 Logging and tracing

To turn on tracing for Transcoding Technology:

1. Log in to the Administration as `wpsadmin` (both user ID and password), then go to the Portal Administration tab. Within the Portal Settings category, select the **Enable Tracing** option. The Enable Tracing tab is shown in Figure 14-8.

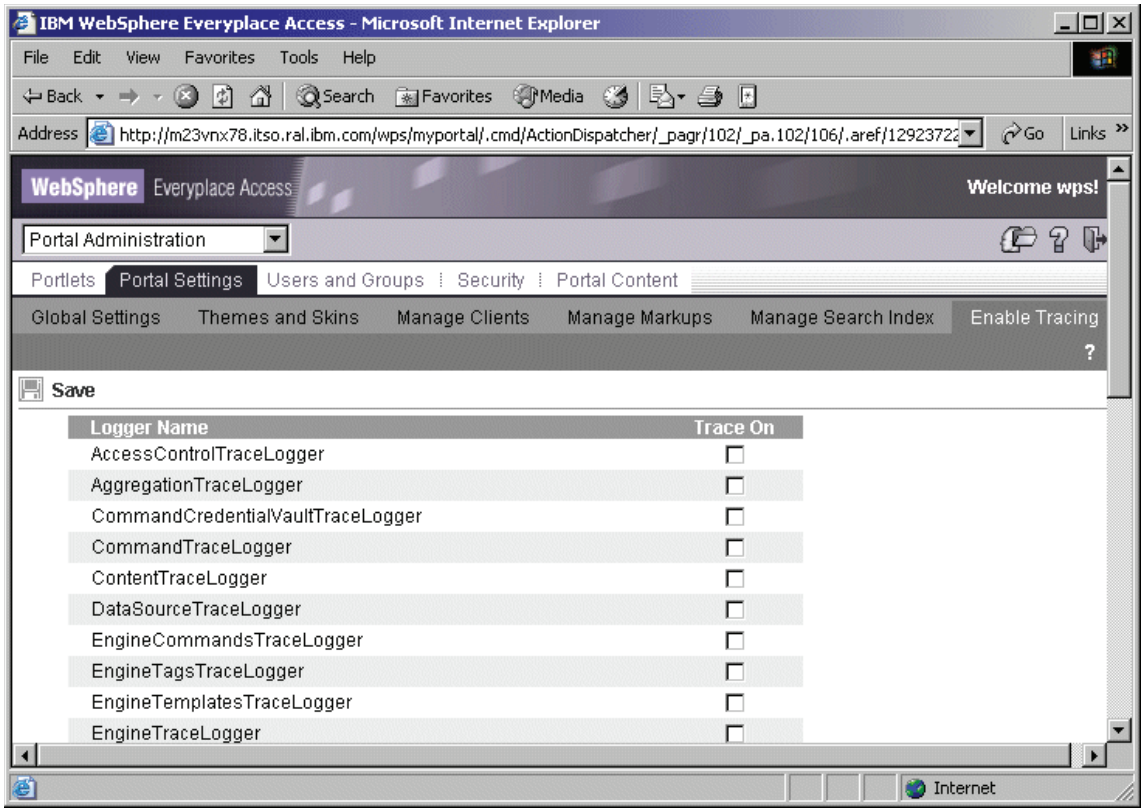


Figure 14-8 Enable tracing portlet

2. Scroll down and check the **TranscodingTraceLogger** box, as shown in Figure 14-9 on page 395.

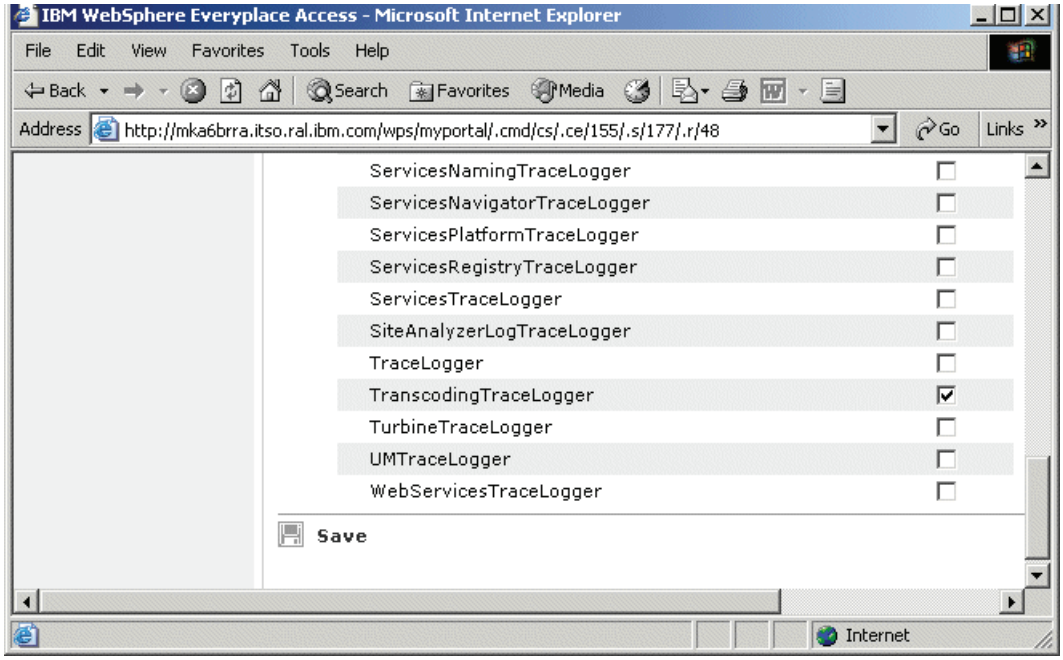


Figure 14-9 Enable tracing for Transcoding Technology

3. Click **Save** after you have selected the desired trace options.

14.4.1 Message files

Message files are created in the Transcoding Technology IBMTrans\log directory. The first message file created is named TranscoderMessages1.log. When the maximum size is reached, this file will be renamed to TranscoderMessages2.log and a new TranscoderMessages1.log file created for new messages. TranscoderMessages1.log is always the newest file. The default message file size is 512 kilobytes.

When the maximum number of message files have been filled, the oldest file will be deleted, the suffix number of each remaining file will be increased by one, and a new TranscoderMessages1.log will be created for new messages. A new file is begun when Transcoding Technology are restarted. The default number of message files is three.

Message file size and maximum number of message files are defined in the <TT_Root>\etc\ras\TranscoderRASMessageFileHandler.properties file.

14.4.2 Trace files

Trace records the specific behavior of Transcoding Technology, which is useful in the diagnosis of problems.

Trace files are created in the Transcoding Technology IBMTrans\log directory. The first trace file created is named TranscoderTrace1.log. When the maximum size is reached, this file will be renamed to TranscoderTrace2.log and a new TranscoderTrace1.log file created for new messages. The default trace file size is 512 kilobytes. TranscoderTrace1.log is always the newest file.

When the maximum number of trace files has been reached, the oldest file will be deleted, the suffix number of each remaining file will be increased by one, and a new TranscoderTrace1.log will be created for new messages. A new file is begun when Transcoding Technology are restarted. The default number of trace files is three.

Tracing file size and maximum number of tracing files are defined in the <TT_Root>\etc\ras\TranscoderRASTraceFileHandler.properties file.

14.4.3 Gather troubleshooting data

When running into problems using Transcoding Technology, there is a certain set of information that will be useful in troubleshooting. There is a batch file named RASCollect.bat in the Transcoding Technology root directory. Executing RASCollect.bat on Windows will package the necessary files and settings to make it easy to gather diagnostic information. When it finishes execution, a zip file is created in the Transcoding Technology IBMTrans\log directory that you can send to the relevant people for diagnosis. It will be named RASCollect.zip on the Windows server.

14.5 A simple portlet using Transcoding Technology

In this section, we include a sample scenario to show how to enable transcoding in WebSphere Everyplace Access.

14.5.1 Enable transcoding

To give an example of how to enable Transcoding Technology for a portlet, a very simple HelloWorld portlet is used, which does nothing but display a `Hello World` message. The portlet descriptor (portlet.xml) for this portlet is shown in Example 14-4 on page 397. Notice that the portlet.xml has two config-param parameters within the concrete-portlet that set this portlet for transcoding. This ensures that transcoding is set.

Example 14-4 portlet.xml for HelloWorld portlet

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portlet-app-def PUBLIC "-//IBM//DTD Portlet Application 1.1//EN"
"portlet_1.1.dtd">
<portlet-app-def>
    <portlet-app uid="HelloWorld.50aee4738a9a00171812d35a82af0a5f"
major-version="1" minor-version="0">
        <portlet-app-name>HelloWord application</portlet-app-name>
        <portlet id="Portlet_1" href="WEB-INF/web.xml#Servlet_1"
major-version="1" minor-version="0">
            <portlet-name>HelloWord portlet</portlet-name>
            <cache>
                <expires>0</expires>
                <shared>NO</shared>
            </cache>
            <allows>
                <maximized/>
                <minimized/>
            </allows>
            <supports>
                <markup name="html">
                    <view/>
                </markup>
            </supports>
        </portlet>
    </portlet-app>
    <concrete-portlet-app uid="HelloWorld.50aee4738a9a00171812d35a82af0a5f.1">
        <portlet-app-name>HelloWord application</portlet-app-name>
        <concrete-portlet href="#Portlet_1">
            <portlet-name>HelloWord portlet</portlet-name>
            <default-locale>en</default-locale>
            <language locale="en">
                <title>HelloWord portlet</title>
                <title-short></title-short>
                <description></description>
                <keywords></keywords>
            </language>
<config-param>
<param-name>DoTranscoding</param-name>
<param-value>true</param-value>
</config-param>
<config-param>
<param-name>FilterChain</param-name>
<param-value>Transcoding</param-value>
</config-param>
        </concrete-portlet>
    </concrete-portlet-app>
```

```
</portlet-app-def>
```

Install this portlet into WebSphere Portal. Make sure that transcoding is enabled by doing the following:

1. Log in to WebSphere Everyplace Access Administration as an administrator, for example `wpsadmin`.
2. Go to the Portal Administration tab and select **Global Settings** and make sure the **Enable transcoding of portlet content** box is checked.

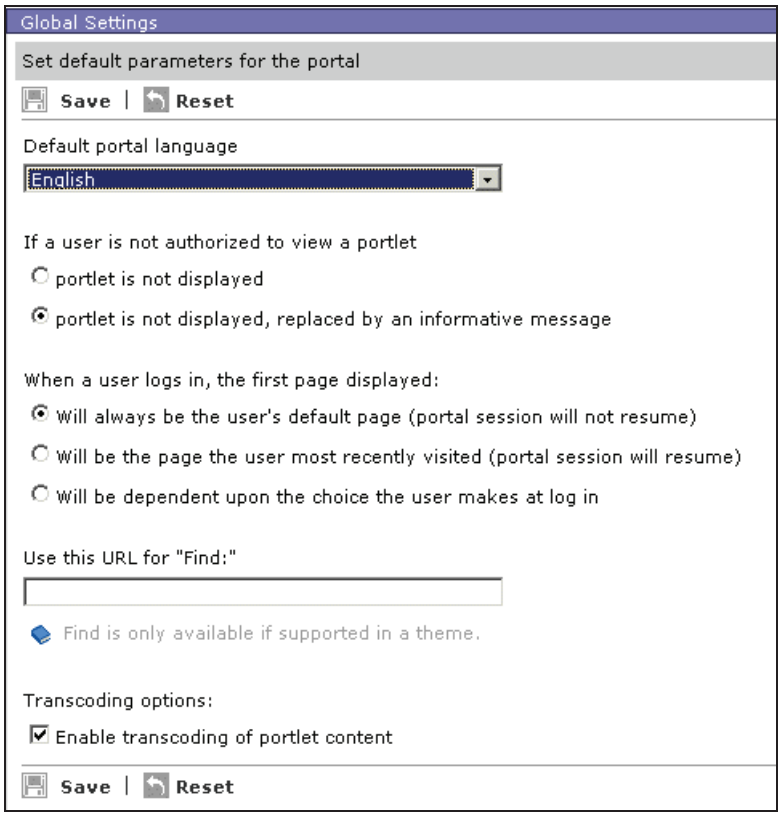


Figure 14-10 Enable global setting for Transcoding Technology

3. To add this portlet to the Welcome page of WebSphere Everyplace Access home page group, perform these steps:
   a. Install the portlet
   b. Set Access Control to allow All anonymous users, retrieve this portlet and select the view for this portlet and save the change.

c. Edit the layout of the Welcome page to add the HelloWorld portlet.

Viewing HelloWorld portlet in a Web browser

Open an Internet Explorer browser and enter the URL. The HelloWorld portlet is shown in Figure 14-11.

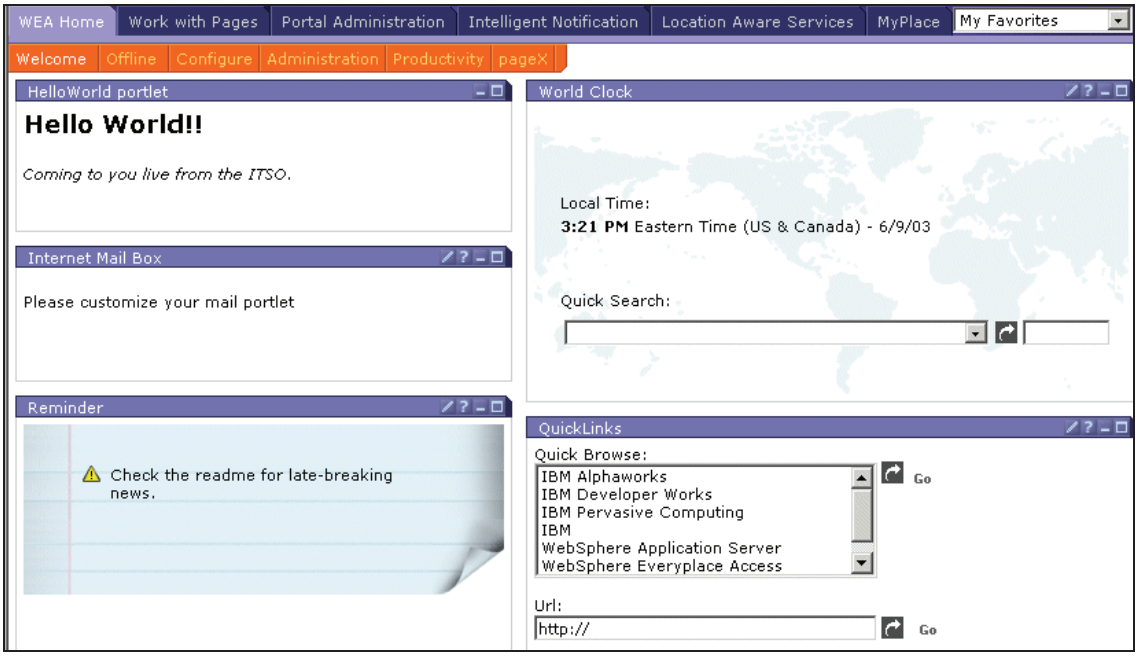


Figure 14-11 HelloWorld portlet displayed in a desktop browser

Viewing HelloWorld portlet in Web browser

Prior to viewing the portlet on the Nokia emulator, we must ensure the device is compatible. To verify this, do the following steps:

1. Use Notepad to open the C:\WebSphere\PortalServer\IBMTrans\etc\plugins\ibm\TextEngine\WMLPrefixer.prop file.
2. Add the statement `!($portletLevel~true)` to the Condition statement. It should look like Example 14-5.

Example 14-5 Condition statement

```
Condition=!($portletLevel~true)&!(x-ibm-wtp~no)&((content-type~text/vnd.wap.wml
)|(content-type~text/x-wap.wml))
```

3. Save the modified file.
4. In the WebSphere Application Server Administration Console, stop and start the WebSphere Portal application server to have the change take effect.

To view the portlet on the Nokia emulator, do the following steps:

1. Use Nokia Mobile Internet Toolkit to view the portlet with a WAP browser. Start Nokia Mobile Internet Toolkit by clicking **Start -> Programs -> Nokia Mobile Internet Toolkit 3.1 -> Mobile Internet Toolkit 3.1**.
2. In the Go field, type the portal URL and press Enter.

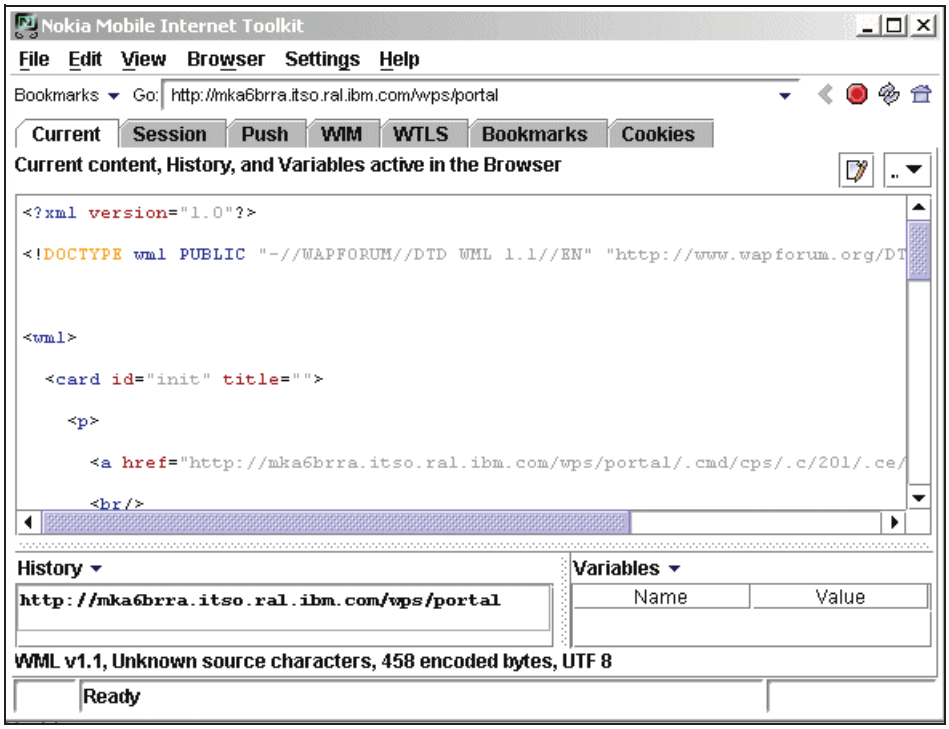


Figure 14-12 Enter URL

3. The Welcome page is shown on the Nokia emulator. Scroll down and select **HelloWorld portlet**. The transcoded portlet is shown in Figure 14-13 on page 401.

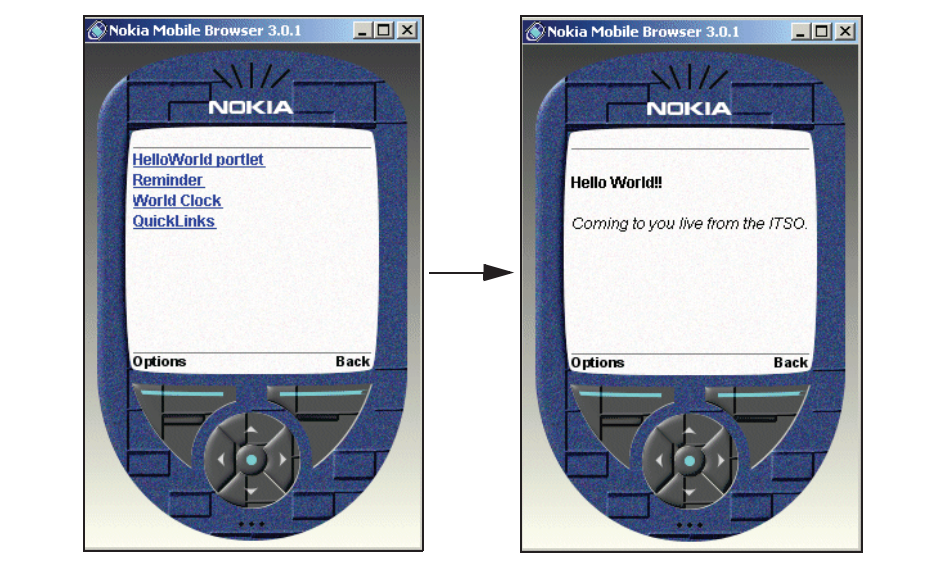


Figure 14-13 HelloWorld portlet displayed in Nokia emulator

14.5.2 Use Request Viewer to monitor the process

Request Viewer can be used to observe and monitor when the portlet and Portal page are transcoded.

Start the Request Viewer (within WebSphere Studio) as detailed in “Starting Request Viewer” on page 391. Use the Nokia emulator to access the HelloWorld portlet. The Request Viewer captures the requests and displays them in its Request Processing window.

As illustrated in Figure 14-14 on page 402, you can expand a request and then expand a transcoder that the request goes through to examine the content changes in MEG Input and MEG Output.

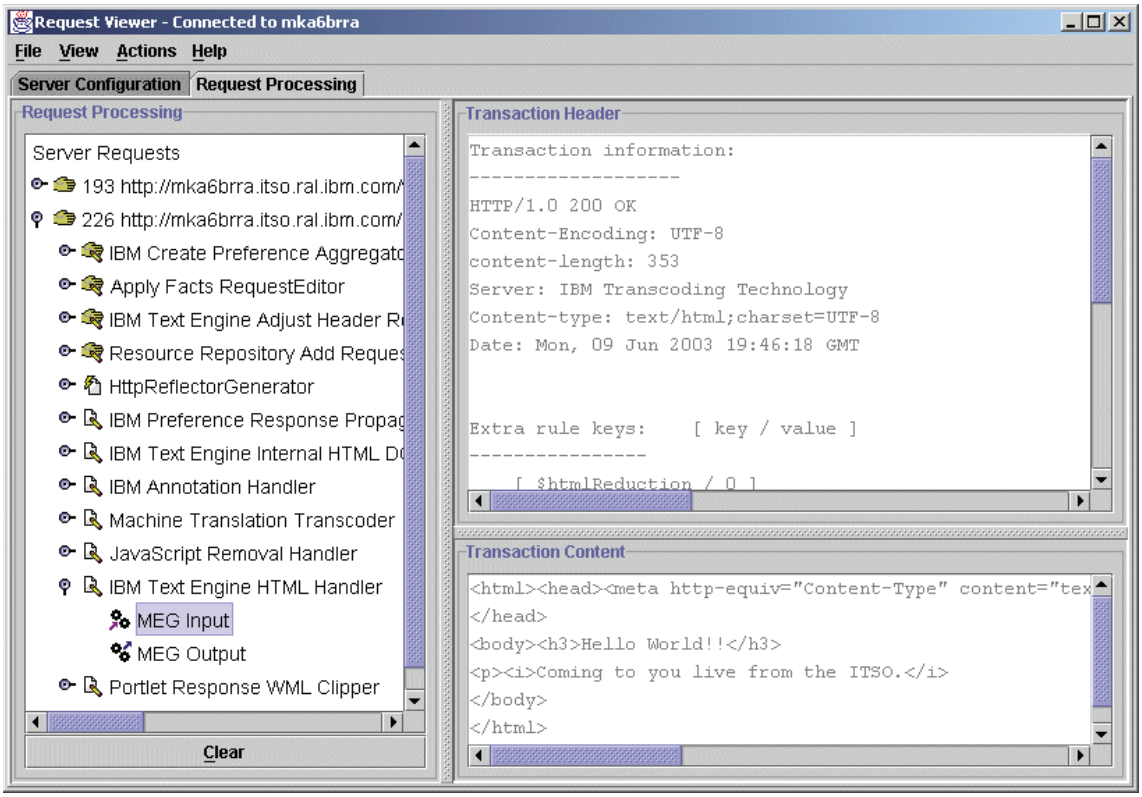


Figure 14-14 Request Viewer

15

Using annotation to clip HTML documents

In this chapter, we introduce you to the Transcoding Technology's annotation capabilities. Annotation is used to perform document clipping, which includes removing and changing the HTML content for the target mobile device dynamically in the runtime environment. Annotation is an approach used to manipulate an HTML document to suit the characteristics of the requesting device by providing annotations either embedded in the HTML document or as a separate annotation file. Using the annotation instructions, a developer can instruct Transcoding Technology how to modify and clip an incoming HTML document to generate a customized output. The selection of the annotations is based on information contained in the HTTP header or the device profile.

This chapter contains the following:

- An overview of annotations for Transcoding Technology
- An example of creating internal annotations using WebSphere Studio
- An example of creating external annotation using the HTML Annotation Editor that is part of the Everyplace Toolkit

© Copyright IBM Corp. 2003. All rights reserved.

15.1 Annotation overview

Document clipping allows an enterprise to deploy existing Web content to mobile devices without creating new versions of the existing content for the target device(s). This ability allows the enterprise to expand the reach of their existing Web applications and allows their mobile workers greater access to information and data.

Annotation, a document clipping technique, allows you to specify the document content to be affected and the action to be performed on the content within an HTML document. The annotation language, an XML dialect, consists of various instructions that are used to direct the handling of the HTML document content by Transcoding Technology. Annotation instructions can be associated with the HTML document in two ways:

1. *External annotations*: The annotation instructions are located in a separate file (.ann file). These instructions consist of two parts:
   - The location: The XPath of the element within the HTML document where the action is to be applied.
   - The action: The particular activity that is to occur against a particular portion of the HTML document (as specified by the XPath). The action may be applied to a particular HTML tag, a group of HTML tags, or to specific content.
2. *Internal annotations*: The annotations reside within the HTML document. They are represented as comments within the HTML file. Internal annotations consist of the actions and are located immediately before the associated HTML tag, group of associated HTML tags, or content.

Both external annotations and internal annotations are processed by the Transcoding Technology within Everyplace Access. Transcoding Technology contains the annotation transcoding plug-in (also called the annotation engine) that processes the annotation instructions and creates a modified version of the document as output. The overall set of annotation instructions available are:

remove	Removes associated HTML tag(s) and content
keep	Keeps associated HTML tag(s) and content
table	Affects overall table (in particular, the heading)
column	Removes a complete table column
row	Removes a complete table row
field	Modifies fields within a form
option	Used to specify a selectable option
insertattribute	Allows insertion of an attribute into an HTML tag

inserthtml	Allows insertion of HTML
replace	Replaces the specified content within the HTML document with the content specified
replacewithhtml	Replaces the associated HTML tag with the new HTML tag specified within the instruction
setpreference	Sets a preference that the Transcoding Technology will use in transcoding the HTML document
splitpoint	Identifies your preferred fragmentation point to the Transcoding Technology

15.1.1 Annotation processing

The annotation processing uses these general steps for internal annotation:

1. Transcoding Technology receives an HTML file (with the internal annotations).
2. A Document Object Model (DOM) is generated from the HTML.
3. The DOM is traversed and each element is manipulated per the annotations as they are encountered by the annotation engine.
4. The DOM is manipulated further if need be by other plug-ins, for example the DOM may be transformed into another markup (if need be) per device profiles.
5. The converted DOM becomes a data stream which is sent to the requesting device.

The primary difference for external annotations is that the annotation instructions are inserted into the DOM tree (think of it as step 2a) prior to any annotations taking place. The annotation instructions are inserted into the DOM according to the XPath with an associated before/after attribute for each annotation instruction.

15.2 Internal annotation

As stated earlier, for internal annotation, the annotation instructions are embedded directly into the HTML document. The new WebSphere Studio products provide the capability to create HTML documents and JSPs. The WebSphere Studio V5 offering contains a new Page Designer (implemented in Java). This new Page Designer tool does not support internal annotation like the old version of the Page Designer. In order to perform internal annotation, you must use WebSphere Studio Application Developer V5 and install the Page Designer Classic. The instructions to install Page Designer Classic are included

in the WebSphere Studio Help's installation instructions (look for Page Designer Classic).

The sample scenarios in this chapter have been developed using WebSphere Studio Application Developer V5, which is a product you must purchase.

We installed the Everyplace Toolkit on WebSphere Studio Application Developer, which provides the tools to create portlets. Next, we performed the steps to install Page Designer Classic, which provides support for internal annotations.

Note: In all subsequent references to Page Designer Classic in this section, the term Page Designer is often used.

15.2.1 Page Designer in WebSphere Studio

The portlet perspective within WebSphere Studio Application Developer is used to create a sample portlet, as well as the JSP and HTML content emitted by the JSP. The WebSphere Studio Application Developer Page Designer is a visual tool that allows you to create JSP statements and HTML content. The Page Designer has three views, which are:

- Design: A WYSIWYG editor that allows you to visually construct the page. This editor is also used to specify your internal annotations.
- Source: An editor that allows you to view and edit the JSP and HTML source.
- Preview: A viewer that allows you to see the HTML as it would appear within a browser.

The WebSphere Studio Application Developer Page Designer is used to create HTML content and JSP code as needed. With Page Designer, you can identify the HTML content that is to be annotated and specify the type of annotation that is to occur. Page Designer provides the following annotation instructions:

- Remove or keep HTML tags or content: Either individual tags or groups of tags can be removed.
- Replace text: Replace tags and their content with text. Note that using replace text removes the HTML tags so any styling provided by the HTML tag(s) is lost.
- Remove table rows or columns: Selected rows or columns can be deleted from the associated table. Multiple rows or columns can be removed in any combination.
- Propagate tables headings: Allows column headings to be propagated as labels with the row content when rendering the tables as lists.

For a good overview on Page Designer in relation to creating internal annotations, refer to Section 5.4 in *IBM WebSphere Everyplace Server Service Provider and Enable Offerings: Enterprise Wireless Applications*, SG24-6519.

15.2.2 Sample application: MyRedbookNews

The MyRedbookNews application is a simple application that consists of a basic portlet named MyRedbookNews.java and a JSP named MyRedbookNews.jsp that emits the HTML. This simple example is intended to show you various annotation capabilities within Page Designer. For example, in this scenario a table and various text areas are included to show various annotation instructions.

As we stated earlier, we must use Page Designer Classic to perform internal annotations. To start Page Designer Classic for the MyRedbookNews.jsp, select and right-click **MyRedbookNews.jsp** and an options menu appears. Select **Open with** and in the second options menu select **Page Designer Classic**, as shown in Figure 15-1.

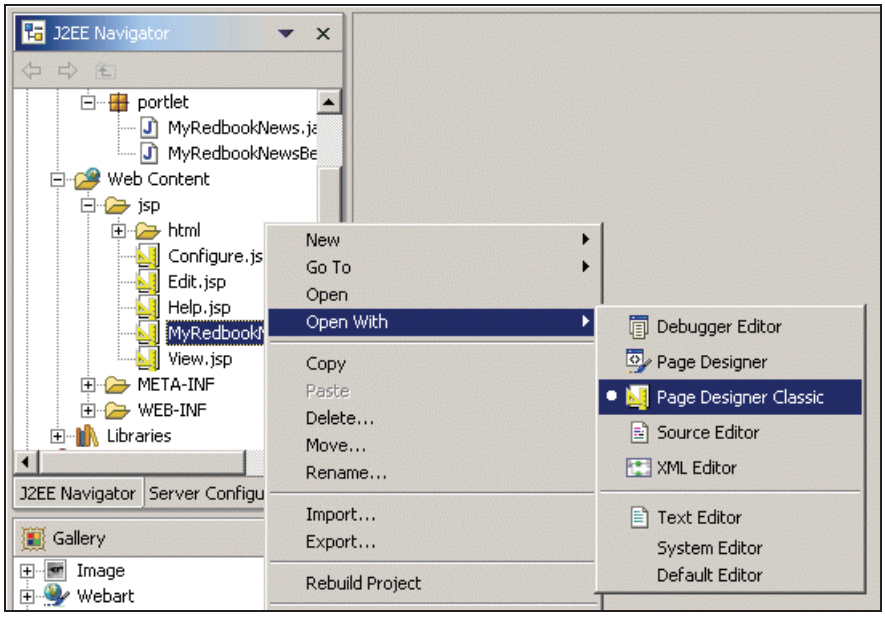


Figure 15-1 Start Page Designer Classic

Figure 15-2 on page 408 shows the MyRedbookNews.jsp Page Designer's Preview view.

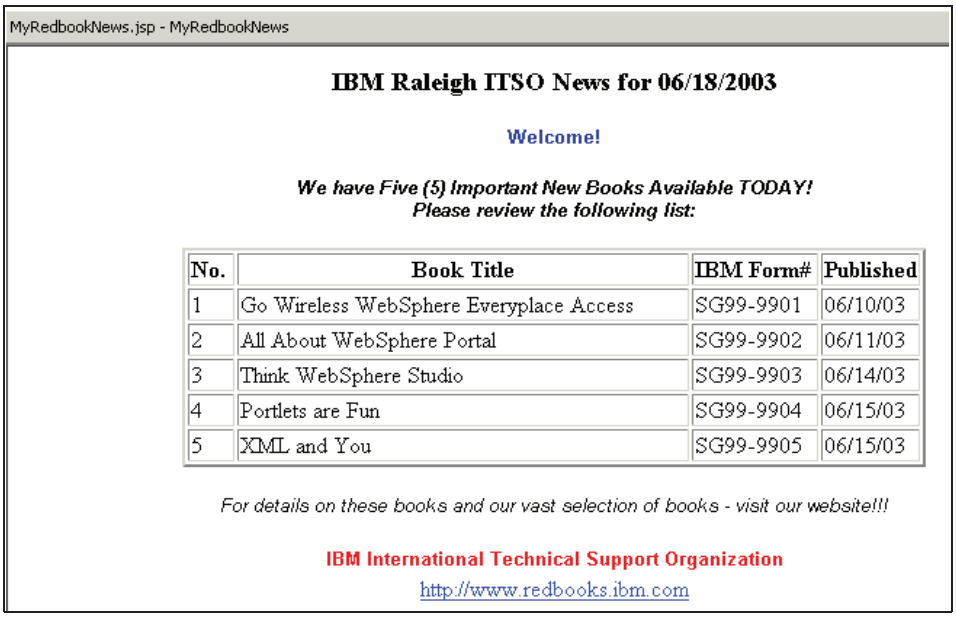


Figure 15-2 MyRedbookNews.jsp in Page Designer - Preview view

Use the Design view within WebSphere Studio Application Developer Page Designer to create the annotation instructions. The annotation default state is set to Keep. However, if you are going to remove more content than you are going to keep, you may want to change the annotation default state, which could minimize the number of steps you would need to perform.

Text replacement

Because many mobile devices have very small screens, it is important to be very selective about the content sent to that target mobile device. The Text Replacement function provides the ability to replace the associated text with your replacement text. This may be a shorter version of the original text. A drawback to this function is that the associated HTML tags are removed and you are not allowed to enter HTML tags in the text. Therefore, any HTML tags used for positioning and improved presentation of the information are lost.

The steps to do text replacement are as follows:

1. In the Design view, click the area that you want to replace; in this case it is **IBM Raleigh ITSO News for 06/18/2003**.
2. Right-click and a drop-down list will appear.
3. Select **Annotation -> Set Text Replacement**, as shown in Figure 15-3 on page 409, and a text box will appear, as shown in Figure 15-4 on page 409.

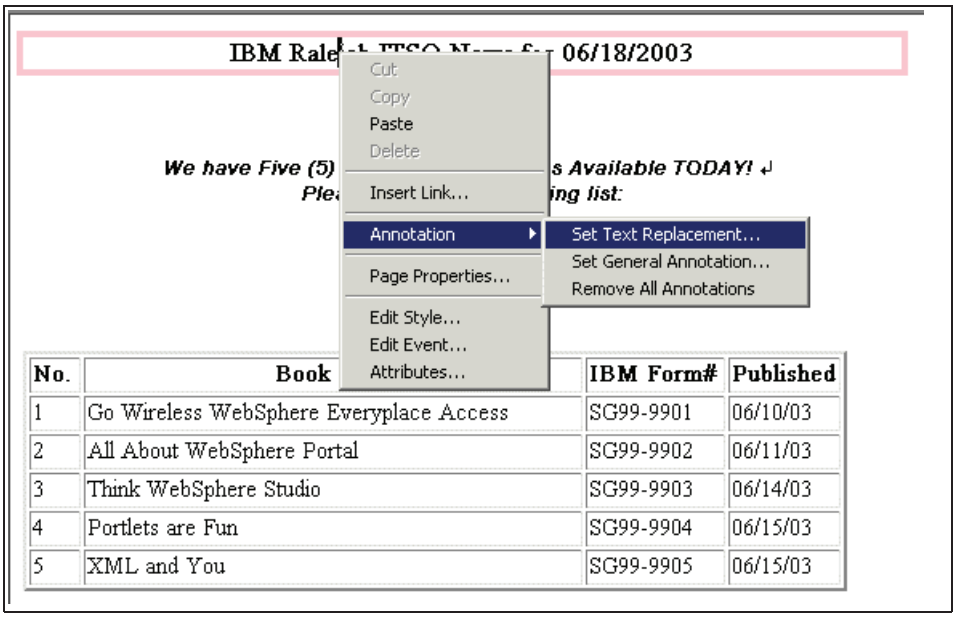


Figure 15-3 Selecting Text Replacement

4. Enter your new (replacement) text into the text entry box, as shown in Figure 15-4, and click **OK**.

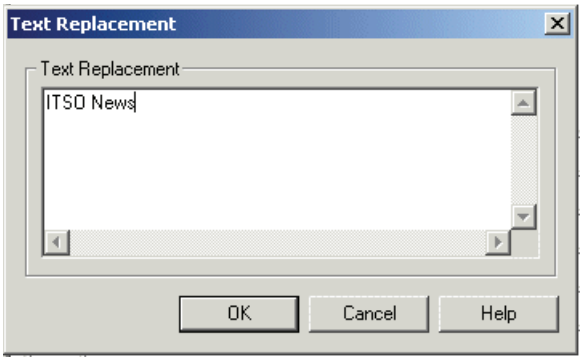


Figure 15-4 Enter replacement text

Tip: If you prefer to work from the menu bar rather than using the right mouse button to get to the annotation option, you can click **Edit** (on the menu bar) and move your mouse to the **Annotate** entry in the drop-down list. While the mouse is on the Annotate entry, the available annotation options will appear in another drop-down list, where you can select the appropriate action. The rest of the steps are the same as when using the right mouse button.

Remove or keep elements

To remove or keep an area (while in the Design view), perform the following steps:

1. Highlight the area you wish to remove or keep. For example, select multiple HTML tags and content to remove. You can select large regions with multiple HTML tags or select each HTML instance individually, whichever approach you prefer.
2. Right-click the area and, as before, the options list will appear.
3. Click **Annotation -> Set Remove Region**. When this operation is complete the specified area will have hash marks through it, as shown in Figure 15-5.

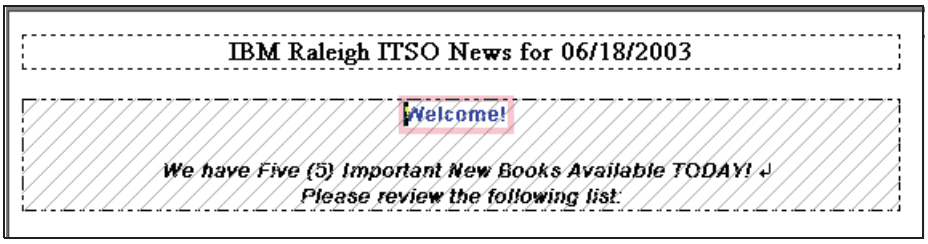


Figure 15-5 Results of remove annotation

Removing columns or rows from a table

Tables are a very nice way to represent information in a Web page; however, some mobile devices do not have the space to display a large table. Page Designer has annotation instructions that allow you to simplify a table. You can remove rows and columns from a table or remove the column headings. In this instance, columns will be removed from the table. You can use the following steps to remove a column or row from a table:

1. Select an entry in the table (column or row) you want to clip.
2. Go to the menu bar and click **Edit -> Attributes**. The Attributes window appears, as shown in Figure 15-6 on page 411. Switch to the Table tab, if not already displayed.
3. Click the **Annotation** tab.

4. Select either **Remove this column** or **Remove this row**, as shown in Figure 15-6. In this case, you will click **Remove this column** and then click **OK**. When the operation is processed, the column appears with hash marks through it (in the Design view).

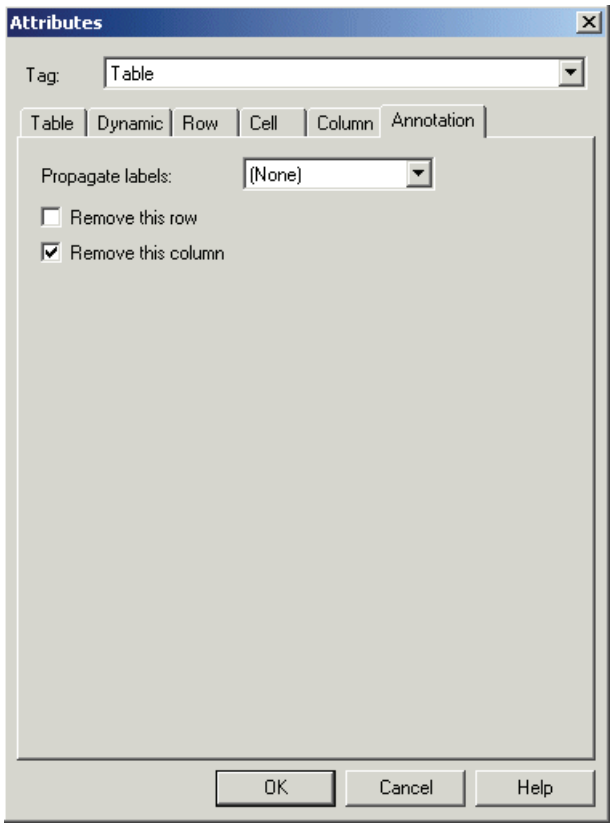


Figure 15-6 Remove column from table

Instead of using the propagate table headings annotation instruction, the headings from the table will be removed by selecting the heading and using the remove rows option.

All the annotations

The complete set of annotations (represented in the Design view) is shown in Figure 15-7 on page 412. These are the unique annotations performed for this sample scenario. To get the desired results, you will use similar actions on other document content. Figure 15-7 on page 412 shows the hash marks in the different areas where content will be removed.

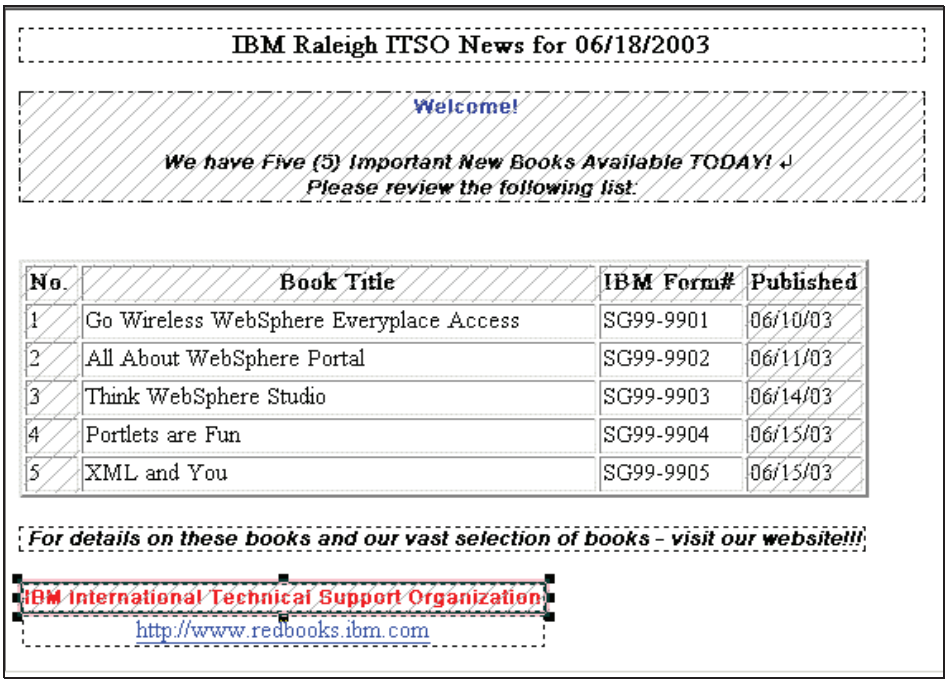


Figure 15-7 Design view with all the annotations

An example of an internal annotation for text replacement is shown in Example 15-1.

Example 15-1 Internal annotations

```
<!--METADATA type="Annotation" startspan
<?xml version="1.0"?><annot version="1.0">
<replace><text>ITSO News</text></replace></annot>-->
<H3>IBM Raleigh ITSO News for 06/18/2003</H3>
<!--METADATA type="Annotation" endspan-->
```

The internal annotation (in Example 15-1) shows replacing the text `IBM Raleigh ITSO News for 06/18/2003` with `ITSO News`. Note that the complete annotation instruction appears before the HTML tag and content that it is to affect. Also note that the internal annotation instruction is represented as a comment.

The results of internal annotation

After the portlet is installed and added to the WebSphere Everyplace Access Home Welcome page, you must make sure the portlet is available to transcoding

by adding the parameter `FilterChain = Transcoding` to the portlet. To set the parameter, execute the following steps within Portal Administration:

1. Click **Manage Portlet** to see the portlets.
2. Highlight the **MyRedbookNews** portlet and click **Modify Parameters**.

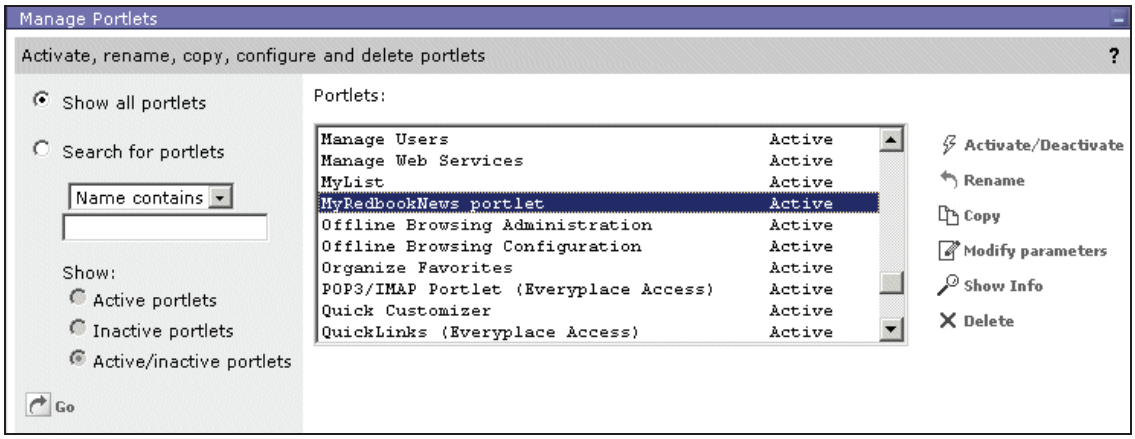


Figure 15-8 Manage Portlet - select MyRedbookNews

3. On the Configure parameters and titles window, specify `FilterChain` as the parameter and `Transcoding` (as shown in Figure 15-9) as the value.
4. Finally, click **Add** and then click **Save**.

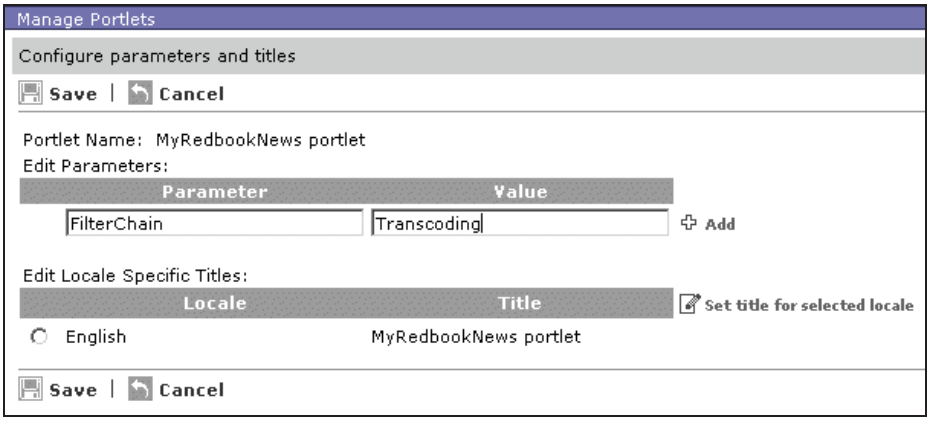


Figure 15-9 Add FilterChain parameter

In Internet Explorer, start a Portal session to view the updated WebSphere Everyplace Access Home Welcome page. You will notice that the new portlet

(MyRedbookNews) contains the annotations intended for the WML browser only. In this situation, you have two options:

1. Disable transcoding for Internet Explorer (IE). This is done by disabling transcoding within the device resource for the Internet Explorer device profile(s). To do this set the <Enable> element value to false in the device profile.
2. Modify the internal annotations to include a condition statement that blocked the annotations from being processed for IE.

In this sample scenario, the internal annotations that affect each annotation instruction within the HTML document are modified. To do this you will edit the annotation instructions within the Page Designer Source view. For example, a modified annotation instruction with the condition (which is highlighted) is shown in Example 15-2.

Example 15-2 Annotation instruction with condition

```
<!--METADATA type="Annotation" startspan
<?xml version="1.0"?>
<annot version="1.0" condition="!(user-agent=*IE*)">
<replace><text>ITSO News</text></replace></annot>-->
<H3>IBM Raleigh ITSO News for 06/18/2003</H3>
<!--METADATA type="Annotation" endspan-->
```

Next, the portlet must be uninstalled and installed again using Portal Administration and added to the WebSphere Everyplace Access Home Welcome page.

Start another Internet Explorer Portal session and the results show the HTML without annotation takes place. The Internet Explorer view of the portlet is shown in Figure 15-10 on page 415.

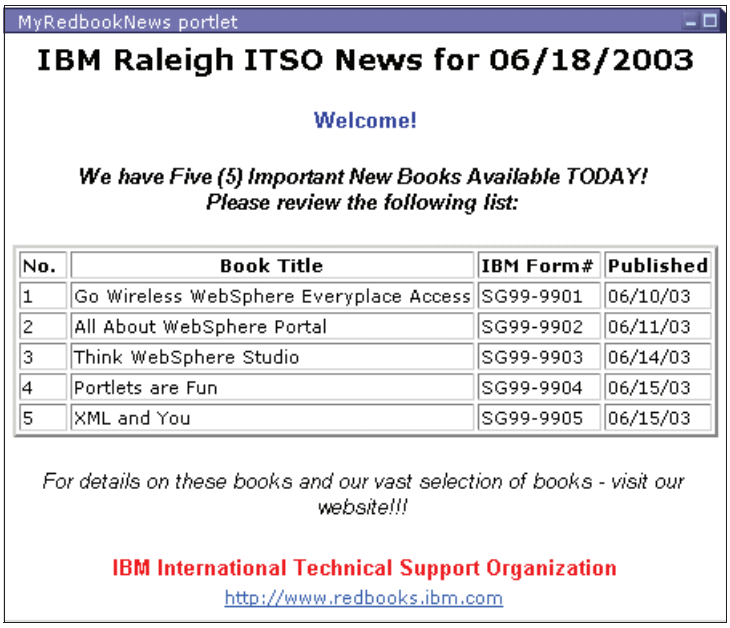


Figure 15-10 Internal annotation results in Web browser

Next the Nokia Mobile Internet toolkit is started and the Portal URL is entered. The MyRedbookNews portlet is selected and the annotated WML document appears. The complete screen is shown in Figure 15-11 on page 416.



Figure 15-11 Internal annotation results in Nokia browser

15.3 External annotation

External annotation allows you to store your annotation instructions in a separate (.ann) file away from the associated HTML source. This approach is appropriate when the developer does not have control over the source HTML or has different annotation files with different sets of annotation instructions for different situations. This approach is also appropriate when the developers do not want internal annotation instructions in the HTML source. The external annotation file consists of both the annotation markup action and the exact location (the XPath) of the associated HTML document tag or content to which the annotation instruction applies. Because the annotations are in a separate file, the XPath expression is needed to indicate to the annotation engine which HTML element or content the annotation instruction is directed against.

Let us take a moment to understand the basics of XPath before looking more closely at external annotations. XPath provides a syntax for defining the specific parts of an HTML or XML document. An HTML or XML document is hierarchical, with a root that contains all other elements; each element can contain other elements as well. There is no limit to the levels of nesting of the elements that can occur within a document. Each element within this hierarchy (tree) has a unique path that defines it. Ultimately, XPath is an expression language for these hierarchical paths. In Figure 15-12, we show an HTML document and the associated XPath expressions (with the syntax as shown in an external annotation file).

Input HTML file	***XPath expression***
<HTML>	/HTML[1]
<BODY>	/HTML[1]/BODY[1]
<H1>Document Clipping </H1>	/H1[1]
<H2>External Annotation</H2>	/H2[1]
<P>Wow we have this new Clipping technique. We can:	/P[1]
<UL>	/UL[1]
<LI>Make the world a better place by</LI>	/UL[1]/LI[1]
<OL>	/UL[1]/LI[1]/OL[1]
<LI>Making web content viewable on handheld devices	/UL[1]/LI[1]/OL[1]/LI[1]
<LI>Opening the world to mobile workers	/UL[1]/LI[1]/OL[1]/LI[2]
</OL>	
<LI>Solve world hunger</LI>	/UL[1]/LI[2]
</UL>	
</P>	
<P>External annotations are fun to create.</P>	/P[2]
</BODY>	
</HTML>	

Note: all XPath expressions within <BODY> have the same starting path of **/HTML[1]/BODY[1]**

Figure 15-12 XPath Example

An external annotation file can be created either with an editor such as Notepad, or by using the HTML Annotation Editor. Our examples of external annotations are created using the HTML Annotation Editor.

There are two key parts to an annotation instruction, which are:

1. Annotation markup action: The particular action to be taken against the HTML element(s)
2. XPath location: The exact location of the element(s) within the HTML source the action is applied against

For internal annotations, the XPath was not necessary because the annotation instruction was located with the HTML tag(s) they were intended to affect. However, the external annotations are in a separate file, so the XPath provides the location of the element(s) to be affected.

15.3.1 The external annotation language

Because the external annotation language is an XML dialect, the first entry within the file is the XML version statement, represented as:

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

Next is the root element of the annotation document:

```
<annot version="2.0" >.
```

Each annotation instruction is represented in a <description> element, which can consist of:

`condition = text` Defines the condition that must be true for the annotation action to be applied. This is an optional attribute.

`take-effect = before or after` Identifies when the annotation is to occur, either before or after the target node.

`target = XPath` Identifies the target node (as defined by the XPath to a given element).

<`action`**>** A child element that states the action to be taken. There are several actions (some of the common actions are included), which are <keep>, <remove>, <replace>, <replacewithhtml>, <setpreferences>, <splitpoint>.

Example 15-3 shows an annotation instruction. This particular annotation shows replacing the text at location `target="/HTML[1]/BODY[1]/CENTER[1]/H3[1]/text()[1]"` with the text `ITSO Redbook News`.

Example 15-3 External annotation description element

```
<description target="/HTML[1]/BODY[1]/CENTER[1]/H3[1]/text()[1]">
```

```
<replace><text>ITSO Redbook News</text> </replace>
    </description>
```

15.3.2 Sample scenario: YourRedbookNews

The YourRedbookNews application used in this scenario shows external annotations. It provides the HTML document used by the HTML Annotation Editor to create the annotation file. The YourRedbookNews application consists of the YourRedbookNews.java portlet, the YourRedbookNews.jsp (which emits the HTML document), and the YourRedbookNews.html file. The HTML consists of headings, text, and tables allowing us to show various annotation instructions. The YourRedbookNews portlet application (running in WebSphere Studio Site Developer), is shown in Figure 15-13.

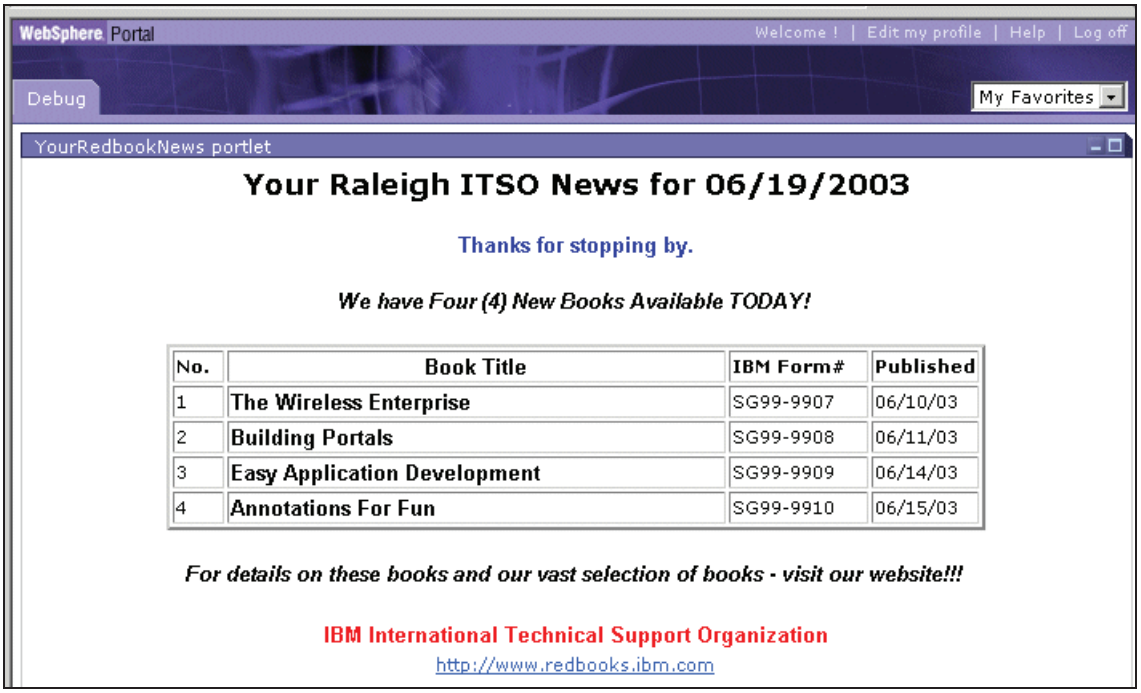


Figure 15-13 YourRedbookNews running in WebSphere Studio

For this application, the target device is a Nokia phone (which we will test using the Nokia Toolkit V3.1 emulator). An external annotation file is created to tune the HTML content for the target device.

15.3.3 Using the HTML Annotation Editor

The HTML Annotation Editor, included in the Everyplace Toolkit, is a tool used to create external annotation files for HTML documents. To start the HTML Annotation Editor, highlight the target HTML file in the portlet project in this case, **YourRedbookNews.html**. Next, right-click to get the pop-up menu and select **Open With -> HTML Annotation Editor** as shown in Figure 15-14.

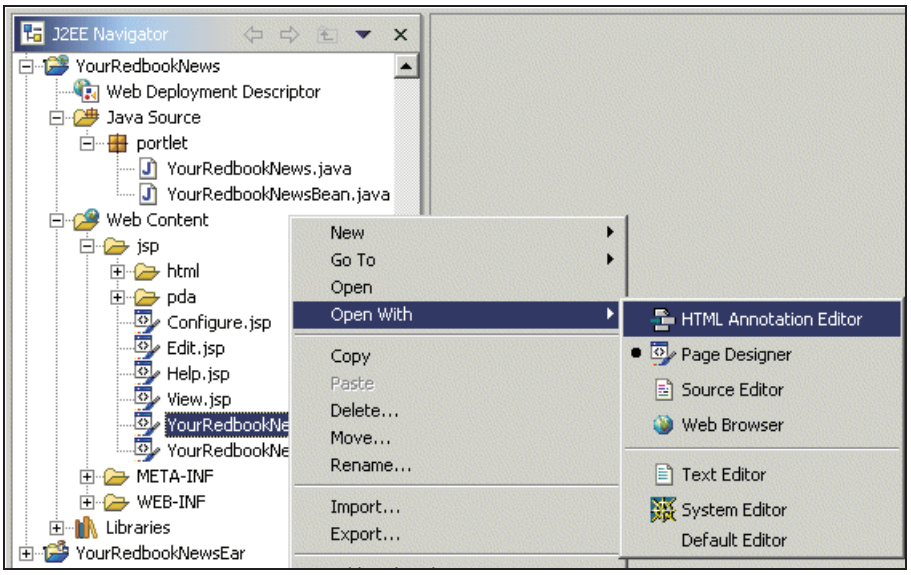


Figure 15-14 Start HTML Annotation Editor

The HTML Annotation Editor displays a condensed (DOM) tree version of your HTML document, as shown in Figure 15-15 on page 421.

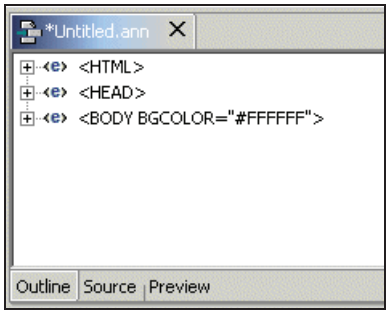


Figure 15-15 Document in HTML Annotation Editor

The HTML Annotation Editor has three tabs (located at the bottom of the window). Each tab provides a different view of the document. The views are:

- Outline view: Displays the DOM tree representation of the HTML document. This view is used to create the annotation instructions.
- Source view: Displays the HTML source, as shown in Figure 15-16.

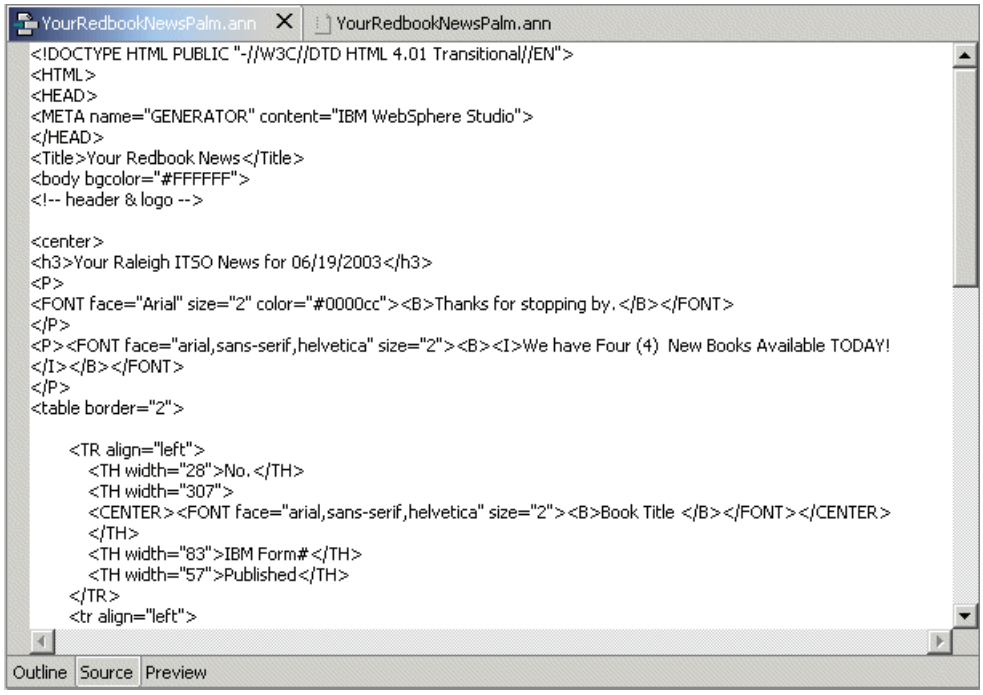


Figure 15-16 Source view

- Preview view: Displays a WYSIWYG (browser) view of the HTML document. Once you have created and saved annotation instructions in the .ann file, this view will invoke the Transcoding Technology to process your annotation file, so that the annotated HTML is displayed.

To expand the Outline view of the document, you can either:

- Click the plus sign (+) beside each of these high-level HTML document elements
- Right-click in the window and a pop-up menu appears. Select either **Expand** (to expand a highlighted element) or select **Expand All**. The expanded HTML document is shown in Figure 15-17.

You can also collapse any of the elements that you have expanded. Simply click the minus sign (-) by the expanded element or use the pop-up menu and click either **Collapse** (to collapse the highlighted element) or **Collapse all** (to collapse the entire document).

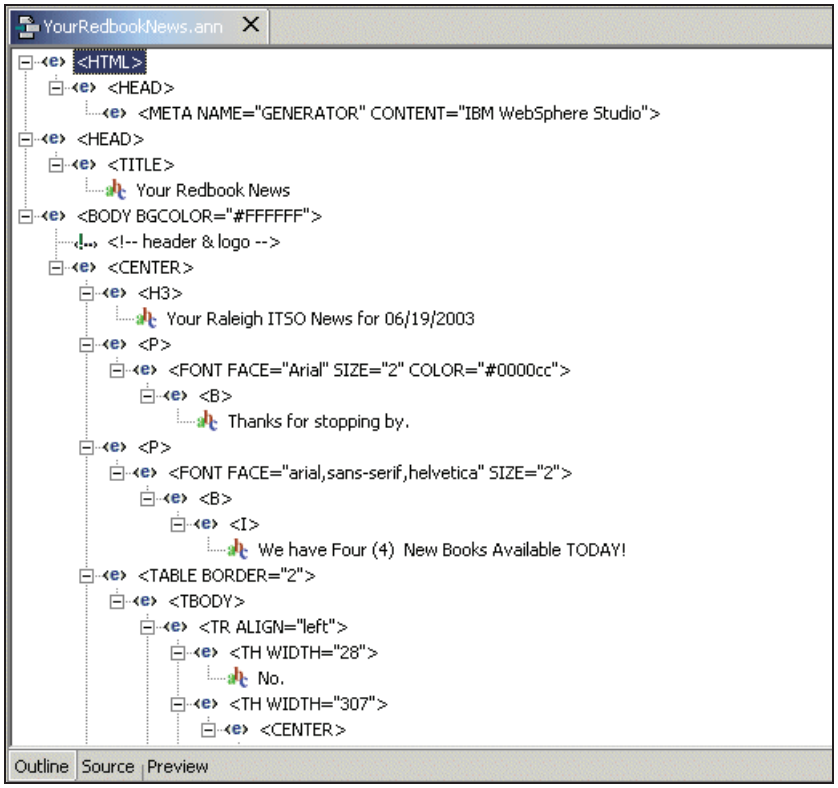


Figure 15-17 Expand HTML document

Save your annotation file

When the HTML Annotation Editor starts for the first time to edit an HTML document, the .ann file is Untitled.ann. You must save this file with its own name. Click **File** and select **Save Untitled.ann as.** Next you will enter or select the folder, in this case `YourRedbookNews/WebContent/WEB-INF`, as shown in Figure 15-18. Enter the file name (YourRedbookNews) and then click **OK**.

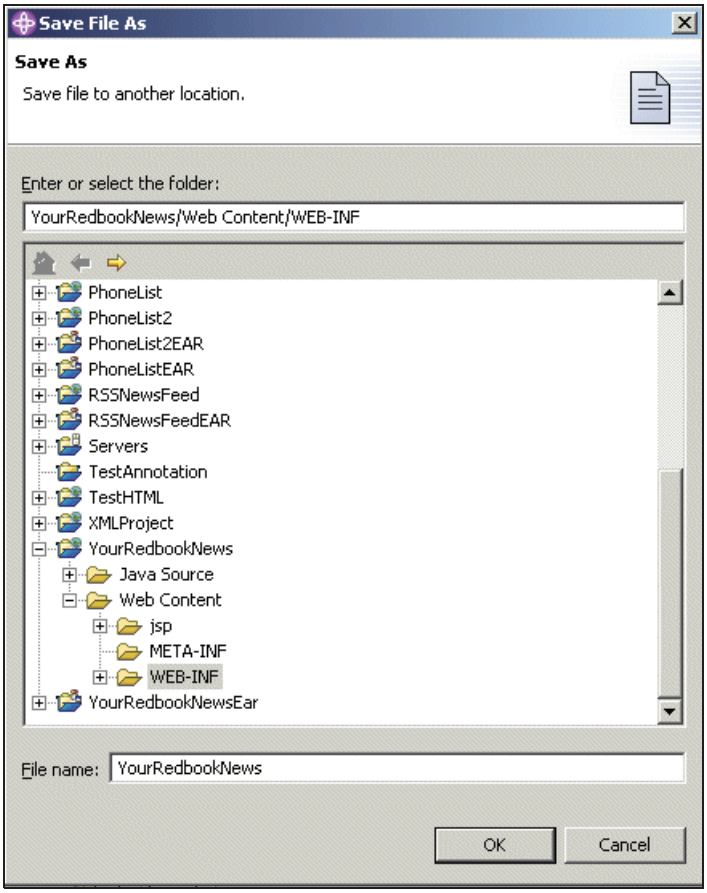


Figure 15-18 Save annotation file as

Steps to create annotations

Document clipping is the process of identifying the specific element(s) within an HTML document that need to be kept, removed, or tailored to meet the needs of the client device. There are two key concepts to understand:

1. Clipping region: The area within the HTML document that has an associated annotation instruction. The area may consist of one or more elements (HTML tags and/or content) of the HTML document.
2. Clipping state: The action to be taken by the annotation engine in that area of the HTML document. The two key clipping states are Keep and Remove.

Annotation instructions are created while in the Outline view of the HTML Annotation Editor, by using the following steps:

1. Select the HTML element(s) that are the target of the annotation instruction:
   - To select an element, right-click the element or content, which brings up the option menu.
   - To select multiple elements, hold down the Control key and click the elements included in the group, or hold down the Shift key and click **Start** and the end element (content) to be included in the selection net.
2. Determine the clipping state (right-click to see the pop-up option menu).
   - Select the desired clipping state option from the option menu, or
   - Click the **Annotation** entry in the WebSphere Studio toolbar and select the desired clipping state option.
3. The associated wizard leads you through the process of filling in the annotation instruction. As an example, let us assume that you selected the **Keep** clip state.
   a. Select a **Begin keeping** radio button, which identifies where the Keep is to start, either *before* the specified HTML element or *after* the specified HTML element.
   b. Select a **Keep Type** radio button to specify either all or selected; let us assume that you selected the **all** option.
   c. Click the **Finish** button.
4. The annotation instruction is created. Similar actions are taken for all clipping activity. You can either return to step 1 to continue the clipping process or move to step 5 to complete the activity.
5. To save your annotation file, from the toolbar select **File ->Save Filename.ann** (your annotation file).

In addition to the two clipping states (Keep and Remove), there are other annotation instructions available. The general instruction types are:

- Replace: Replaces any element (html tag / content) with either plain text or HTML.
- Insert comments: Inserts comments in the output HTML document.
- Insert attribute(s): Inserts additional attributes in an existing HTML element generated in the output stream.
- Insert HTML: Inserts HTML fragments into the output HTML document.
- Define splitpoints (fragmentation directives): Identifies where fragmentation should occur.
- Distribute table headers: Identifies tables with headers that, if converted to lists, should have the table headers distributed over each row.

In order to create our annotation instructions, we used the process defined above. However, the wizards may contain questions or options specific to that instruction.

Creating an annotation for YourRedbookNews

To demonstrate how to create an annotation instruction, let's create the annotation to replace the heading `Your Raleigh ITSO News for 06/19/03` with `ITSO Redbook News`:

1. Highlight the heading contents (Your Raleigh ITSO News for 06/19/03).
2. Right-click to obtain the option menu.
3. Select the **Replace** option (from the option menu).
4. The Replace Annotation Wizard window displays, as shown in Figure 15-19 on page 426.
5. Select the **Replace with text** radio button (to replace the heading text).
6. Click **Next**.
7. The wizard presents the Build Text Content window, shown in Figure 15-20 on page 427.
8. Enter the text `ITSO Redbook News` and click **Finish**.

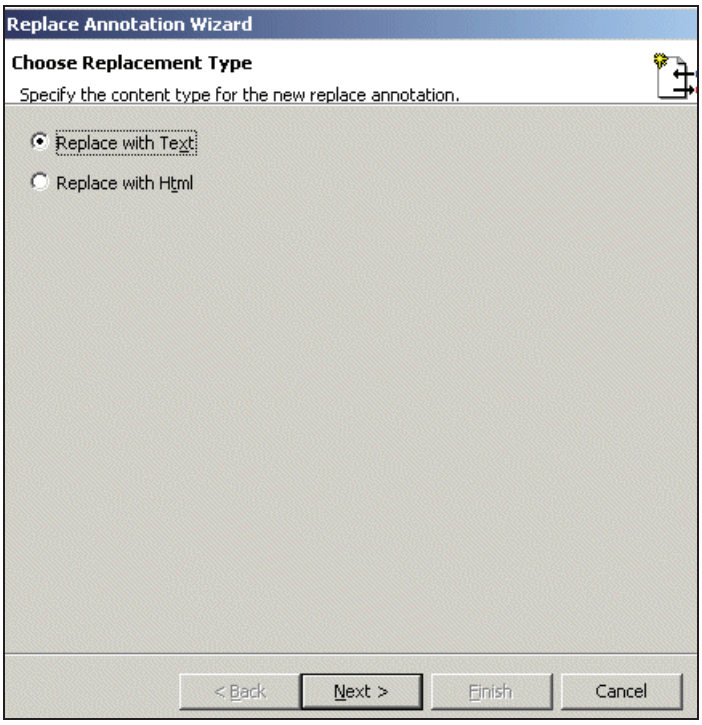


Figure 15-19 Replace Annotation Wizard

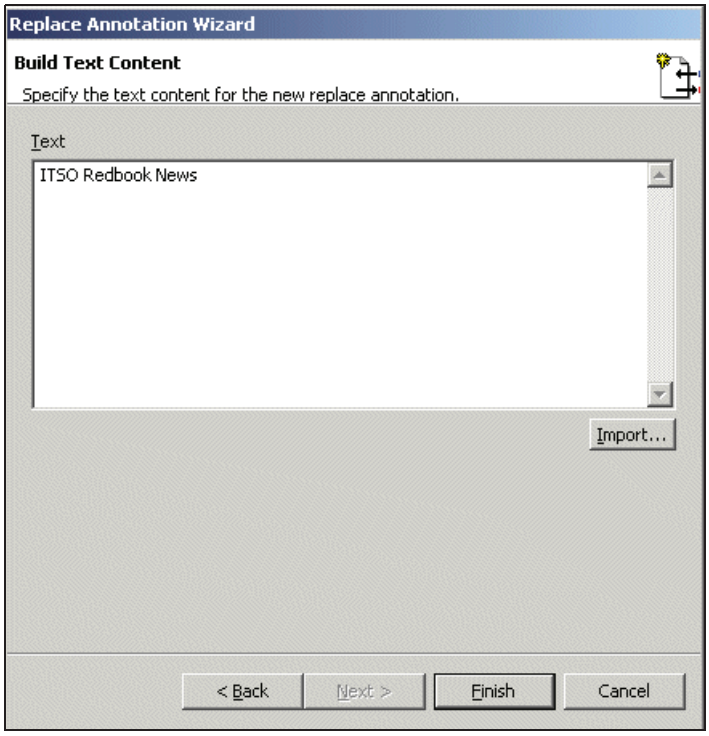


Figure 15-20 Replace Heading Text

We used this same fundamental approach to create the annotation instructions we needed to format the HTML content for a Nokia device.

Viewing Annotation instructions

If you would like to view the annotation instruction, first save the .ann file. Then right-click the annotation file (in the project list) and a pop-up menu appears. Next select **Open With -> Text Editor** from the pop-up menu. You can edit the file using the text editor in WebSphere Studio, which in some cases is faster than creating each individual annotation instruction.

In this case, we used the HTML Annotation Editor to create the following annotations:

- Replace the heading `Your Raleigh ITSO News for 06/19/2003` with the text `ITSO Redbook News`
- Remove the content: `Thanks for stopping by` and `We have Four (4) New Books Available TODAY!`

- Replace the content `For details on these books and our vast selection of books - visit our web site!!` with the text `Details at our Web site`.
- Remove the IBM International Technical Support Organization text.

But when it came to creating annotations for the table, we decided to use the text editor to add the special table processing instruction to the annotation file. This was much easier than creating individual annotation instructions for each column or row in the table we wanted to affect.

Unlike creating internal annotations, you can only create annotation statements for rows and not for columns. This would cause us to create annotation instructions for each row in table 1. Our objective is to remove columns 1 and 4 of the table. A subset of the statements created using the HTML Annotation Editor are shown in Example 15-4. It shows the annotations to:

- Keep the table
- Remove table heading 1
- Keep table heading 2
- Keep table heading 3
- Remove table heading 4

As you can see, it would be a tedious process to remove the entries in the rows we wanted to remove. We basically wanted to remove columns 1 and 4.

Example 15-4 HTML Annotator instruction for table

```
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]">
        <keep/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[1]/TH[1]">
        <remove/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[1]/TH[2]">
        <keep/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[1]/TH[3]">
        <keep/>
    </description>
    <description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[1]/TH[4]">
        <remove/>
    </description>
```

For each subsequent row we would have created similar instructions. However, we used the special table processing instruction, which is shown in Example 15-5 on page 429. This special table processing instruction supports both columns and rows, depending on the orientation that works best for you.

Example 15-5 Special Table instructions

```
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]">
        <keep/>
        <table>
          <column index="1" clipping="remove"/>
          <column index="2" clipping="keep"/>
          <column index="3" clipping="keep"/>
          <column index="4" clipping="remove"/>
        </table>
    </description>
```

The edited version of the annotation file in the text editor is shown in Figure 15-21 on page 430.

YourRedbookNews.ann

```
<?xml version="1.0" encoding="UTF-8"?>
<annot version="2.0">
    <description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/H3[1]/text
        <replace>
            <text>ITSO Redbook News</text>
        </replace>
    </description>
    <description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/P[1] | /HT
        <remove/>
    </description>
    <description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/P[1]">
        <remove/>
    </description>
    <description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]">
        <keep/>
        <table>
            <column clipping="remove" index="1"/>
            <column clipping="keep" index="2"/>
            <column clipping="keep" index="3"/>
            <column clipping="remove" index="4"/>
        </table>
    </description>
    <description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/P[2]">
        <keep/>
    </description>
    <description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/P[3]/FONT[
        <replace>
            <text>Details at our website:</text>
        </replace>
    </description>
    <description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/T
        <remove/>
    </description>
    <description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/T
        <keep/>
    </description>
</annot>
```

Figure 15-21 Annotation file in Text Editor

Note: If you decide to edit the .ann file using the text editor, be sure to save your changes. End the HTML Annotation Editor. Perform your editing using the Text Editor and then restart the HTML Annotation Editor for this target HTML file.

To view the results of creating your annotation, first save your annotation file and then select the **Preview** tab in the HTML Annotation Editor. Our results are shown in Figure 15-22 on page 431.



Figure 15-22 External annotation results

15.3.4 Running our portlet with external annotations

In this section, we show you how to set the portlet parameters, and install and run the sample portlet.

Setting portlet parameters

There are two ways to configure the portlet to use the external annotation file and set other related parameters, which are:

- Use Portal Administration and select **Manage Portlets** and use the Modify parameters action to define the parameters. We used this approach in “The results of internal annotation” on page 412.
- Modify the portlet.xml file (the deployment description) associated with YourRedbookNews portlet to include config-param elements within the concrete-portlet element. Do this by performing the following steps:
  - Select the **portlet.xml file**, right-click and select **Open with -> XML Editor**.
  - In the XML Editor, expand these elements: portlet-app-def, concrete-portlet-app and concrete-portlet.
  - Right-click **concrete-portlet** and in the pop-up menu select **Add Child -> config-param**.
  - A config-param is added to the existing entries. Expand the new entry to show its two children: param-name and param-value.
  - Select the param-name (in the right column) and enter your parameter name (as listed in Table 15-1 on page 432).

– Select the param-value (in the right column) and enter your parameter value (as listed in Table 15-1).
– Continue creating config-param elements and assigning names and values for each entry in Table 15-1.

Table 15-1 Config param

Param-name	Param-value	Comment
FilterChain	Transcoding	Configure portlet to use Transcoding
AnnotationFile	http://mka6brra.itso.ral.ibm.com/YourRedbookNews.ann	Identify annotation file to use and its location

The results of editing the portlet.xml file are shown in Figure 15-23.

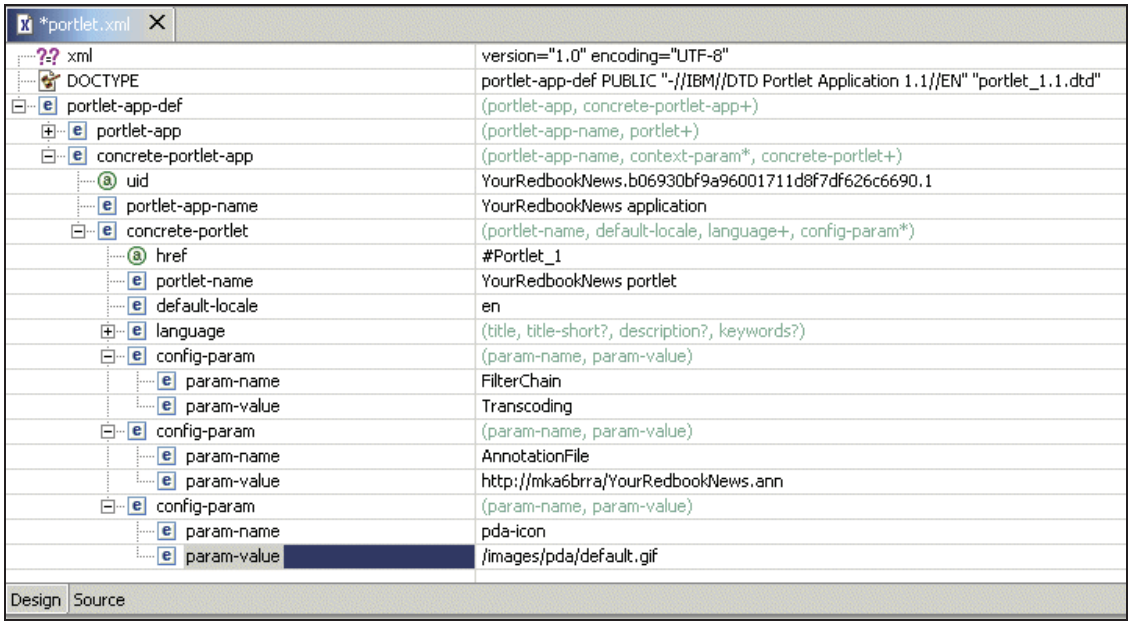


Figure 15-23 Portlet.xml

Installing the portlet

After the portlet and the .ann file have been exported from WebSphere Studio, we must load the YourRedbookNew.war and YourRedbookNew.ann files on the server. We placed the YourRedbookNew.war in a directory on the C drive and placed the YourRedbookNew.ann file in the C:\Program Files\IBM HTTP

Server\htdocs directory, because this is where the HTTP server will look for the file.

We decided to keep the .ann file separate from the WAR file, which allowed us to continue to edit the .ann file as needed, without having to redeploy and reinstall the portlet. This is reflected in the portlet.xml file as a config-param element (the AnnotationFile parameter), as we discussed in the “Setting portlet parameters” on page 431.

To install the portlet and add it to the WebSphere Everyplace Access Home Welcome page, execute the following steps within Portal Administration:

1. Select the **Install Portlets** option in the Portlets category.
2. Click **Browse** and navigate to the portlet, in this case C:\A_ITSO\YourRedbookNews.war, and click **OK**.
3. The YourRedbookNews portlet is displayed. Click **Install**. A few seconds later, you should see the message that the portlet installed successfully.
4. Select **Access Control List**, which is part of the Security category.
5. Select the **All Anonymous users** radio button and select **portlets** under the Select the objects for permissions. Click **Go**.
6. Find **YourRedbookNews** in the list and select the **View** radio button under the Minimum heading and click **Save**. You will see a check in the View column under the Active heading for YourRedbookNews.
7. Select the **Work with Pages** tab. Next select **Edit Layout**, and to expand the **WEA Home** click the + icon.
8. Click **Welcome** and within the Edit layout window (for Welcome page), click the **Add Portlet** icon.
9. To find the YourRedbookNews portlet, click the **Name Contains** radio button and enter `Your` in the entry field and click **Go**.
10. Select the **YourRedbookNews** portlet in the list returned and click **OK**.
11. Click **Activate** to activate the page.

Important: If you change the portlet project in WebSphere Studio and want to redeploy it, you must uninstall the existing portlet and install the modified portlet to have the changes take effect.

15.3.5 Viewing the results of external annotation

We build our external annotation file primarily for the Nokia device. However, we wanted to test it against the Internet Explorer.

Internet Explorer - external annotation

We did not intend the YourRedbookNews.ann file to affect the Internet Explorer view of the HTML file. However as you can see in Figure 15-24, the annotations are in effect.



Figure 15-24 Results - Internet Explorer with annotations

We edited the annotation file, adding a condition parameter to our annotations to stop the annotation of the Internet Explorer. We added the `condition="!(user-agent=*MSIE*)"` attribute to each <description> element. After saving our changes, we refreshed our Internet Explorer session. The results are shown in Figure 15-25 on page 435.



Figure 15-25 Results - Internet Explorer without annotations

WML browser - external annotation

With the ability to edit the annotation file without redeploying the portlet, we decided to change the special table instruction to keep columns 1 and 2 and remove columns 3 and 4.

We also changed the condition statement for the special table processing instruction to reflect ”(user-agent=*Nokia*)” identifying this instruction for the Nokia device. This way you can see the use of variety of condition statements in an annotation file. The results of the annotation process are displayed on the Nokia Internet Toolkit browser, as shown in Figure 15-26 on page 436.

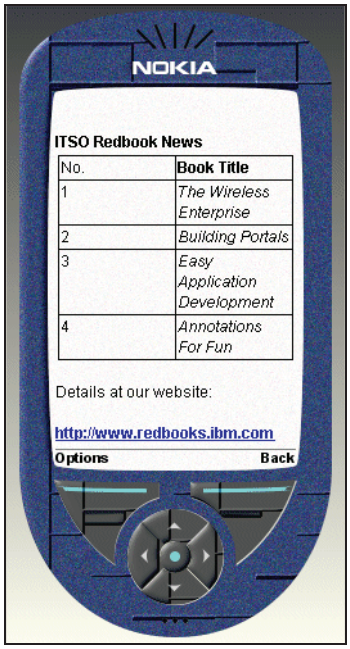


Figure 15-26 Results - Nokia browser external annotation

The complete annotation file is shown in Example 15-6.

Example 15-6 External Annotation for WML browser

```
<?xml version="1.0" encoding="UTF-8"?>
<annot version="2.0">
    <description condition="!(user-agent=*MSIE*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/H3[1]/text()[1]">
        <replace>
            <text>ITSO Redbook News</text>
        </replace>
    </description>
    <description condition="!(user-agent=*MSIE*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/P[1]">
        <remove/>
    </description>
    <description condition="!(user-agent=*MSIE*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/P[2]">
        <remove/>
    </description>
    <description condition="!(user-agent=*MSIE*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]">
        <keep/>
```

```
    </description>
     <description condition="(user-agent=*Nokia*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]">
        <keep/>
        <table>
          <column index="1" clipping="keep"/>
          <column index="2" clipping="keep"/>
          <column index="3" clipping="remove"/>
          <column index="4" clipping="remove"/>
        </table>
    </description>

    <description condition="!(user-agent=*MSIE*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/P[3]/FONT[1]">
        <replace>
            <text>Details at our website:</text>
        </replace>
    </description>
    <description condition="!(user-agent=*MSIE*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]">
        <keep/>
    </description>
    <description condition="!(user-agent=*MSIE*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[1]">
        <remove/>
    </description>
    <description condition="!(user-agent=*MSIE*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[2]">
        <keep/>
    </description>
</annot>
```

Pocket PC - external annotation

To display our portlet on the Pocket PC we modified the .ann file by:

- Changing the condition in the annotations to replace the check for user-agent=*MSIE* to check for user-agent= *Nokia* or user-agent=*Windows CE*
- Copied the special table processing instruction and modified it for user-agent=*Windows CE* and changed the details to remove column 1 and 4 and keep columns 2 and 3.

The results of the external annotations on the Pocket PC are shown in Figure 15-27 on page 438. The modified external annotation file is in Example 15-7 on page 438.



Figure 15-27 Results - Pocket PC external annotation

Example 15-7 External annotations for Nokia and Windows CE

```
<?xml version="1.0" encoding="UTF-8"?>
<annot version="2.0">
    <description condition="((user-agent=*Nokia*) |(user-agent=*Windows CE*))"
take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/H3[1]/text()[1]">
        <replace>
            <text>ITSO Redbook News</text>
        </replace>
    </description>
    <description condition="((user-agent=*Nokia*) |(user-agent=*Windows CE*))"
take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/P[1]">
        <remove/>
    </description>
    <description condition="((user-agent=*Nokia*) |(user-agent=*Windows CE*))"
take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/P[2]">
        <remove/>
    </description>
    <description condition="((user-agent=*Nokia*) |(user-agent=*Windows CE*))"
take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]">
        <keep/>
    </description>
     <description condition="(user-agent=*Nokia*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]">
        <keep/>
```

```
        <table>
          <column index="1" clipping="keep"/>
          <column index="2" clipping="keep"/>
          <column index="3" clipping="remove"/>
          <column index="4" clipping="remove"/>
        </table>
    </description>
    <description condition="(user-agent=*Windows CE*)" take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]">
        <keep/>
        <table>
          <column index="1" clipping="remove"/>
          <column index="2" clipping="keep"/>
          <column index="3" clipping="keep"/>
          <column index="4" clipping="remove"/>
        </table>
    </description>
   <description condition="((user-agent=*Nokia*) |(user-agent=*Windows CE*))"
take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/P[3]/FONT[1]">
        <replace>
            <text>Details at our website:</text>
        </replace>
    </description>
    <description condition="((user-agent=*Nokia*) |(user-agent=*Windows CE*))"
take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]">
        <keep/>
    </description>
    <description condition="((user-agent=*Nokia*) |(user-agent=*Windows CE*))"
take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[1]">
        <remove/>
    </description>
    <description condition="((user-agent=*Nokia*) |(user-agent=*Windows CE*))"
take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[2]">
        <keep/>
    </description>
</annot>
```

16

Using XSL transformations and XML tools

In this chapter, the Transcoding Technology within WebSphere Everyplace Access (Everyplace Access) is used to transform Extensible Markup Language (XML) files to the markup language(s) required by mobile device(s) or Web browsers. It is natural to extend existing applications that generate XML files to the Web and to mobile devices. This approach provides the enterprise with the opportunity to share XML files with both their mobile users and the on-site users. The value of this approach is that the enterprise only needs to maintain or create one representation of the data (as an XML file).

The combination of Extensible Stylesheet Language Transformation (XSLT) and Transcoding Technology allows the transformation of XML files for display on various mobile devices and Web browsers. XSLT is frequently used to convert XML files of a particular document type to other XML dialects, HTML, and device markup languages (such as WML). XSLT files consist of rules that specify the XML content (elements, attributes, and element content) to be processed and how that content is to be processed in order to generate the expected markup language.

Note: In this chapter we use the terms XSLT, XSL and stylesheet interchangeably.

© Copyright IBM Corp. 2003. All rights reserved.

The following topics are discussed in this chapter:

- How to configure the Portal for stylesheet processing
- How to configure a portlet for stylesheet processing
- How to use stylesheets with sample applications
- How to use multiple stylesheets against a particular XML document type
- Explore the various tools within WebSphere Studio Site Developer used to create XML files and stylesheets

16.1 Overview

XML has become the standard for representing information when that data is outside a conventional data store. By its very nature, XML provides the means (through the use of elements and attributes) to express data in a descriptive and meaningful way. XML files consist of both the data description (via elements and attributes) and the content (data) housed together within the same file.

Representing data in XML files makes processing and understanding that data very easy and natural for humans and appropriately equipped applications. However, browsers understand their own markup languages. In order to display XML files in a pleasing and readable fashion on the various browsers, the XML file must be converted to the target browser's markup language.

Stylesheets are used to convert XML to the desired browser markup language. Transcoding Technology (TT) within Everyplace Access provides the mechanism to accept both the stylesheet and the XML file as input and generate the desired markup as is specified by the stylesheet rules.

Transcoding Technology provides you with a means to use stylesheets with your XML files and dynamically convert the XML file to the right markup needed by a specific device.

16.1.1 Configuring stylesheets

The transcoding server must know about the stylesheet(s) in order to perform the XML file conversion. Transcoding Technology provides you with these methods of registering your stylesheets with the transcoding server. These are:

- Configure the portlet to use a stylesheet. There are two ways to configure the portlet. They are:
  - If the stylesheet is part of the portlet WAR file, a <config-param> element with the stylesheet specifics is added to the <concrete-portlet> element within the portlet's portlet.xml file.

- If the stylesheet is not within the portlet WAR file, you can select **Portal Administration -> Manage Portlets** (Modify parameters) to specify the StylesheetFile parameter for the installed portlet.

- Specify the stylesheets within the XML file using the wtp-condition. This approach supports multiple stylesheets for a file. This approach supports situations where the portlet emits more than one XML document type, each of which has its own stylesheet(s).

16.2 Configure Portal for stylesheet processing

Prior to configuring the specific application portlet(s), you must configure the Portal for processing XML files. This is accomplished by updating the PortletFilterService.properties file and adding XML in the Portal markup languages supported by the Portal. To do this, the following steps should be performed:

1. Open the **PortletFilterService.properties** file with a text editor. The file is located within the application server directory at <ApplicationServer>/lib/app/config/services/.
2. After the existing entries within the filtername = Transcoding, add the following entries:

```
Transcoding.transcodeMarkup.4 = xml->hmtl
Transcoding.transcodeMarkup.5 = xml->wml
Transcoding.transcodeMarkup.6 = xml->pda
```

3. Save and close the file.

The results of editing the PortletFilterServices.properties file are shown in Example 16-1. These changes tell Transcoding Technology to convert XML to the associated markups.

Example 16-1 Modified PortletFilterService.properties

```
filtername1 = Transcoding
Transcoding.classname =
com.ibm.transform.wps.portletfilter.TranscodingPortletFilter
Transcoding.transcodeMarkup.1 = html->wml
Transcoding.transcodeMarkup.2 = html->chtml
Transcoding.transcodeMarkup.3 = html->vxml
Transcoding.transcodeMarkup.4 = xml->html
Transcoding.transcodeMarkup.5 = xml->wml
Transcoding.transcodeMarkup.6 = xml->pda
```

To include XML in the markup languages supported by Everyplace Access, access the Portal Administration and execute the following steps:

1. Click the **Portal Administration** tab and select **Manage Markups** (under the Portal Settings category). This gets you to the Manage Markups page, as shown in Figure 16-1.

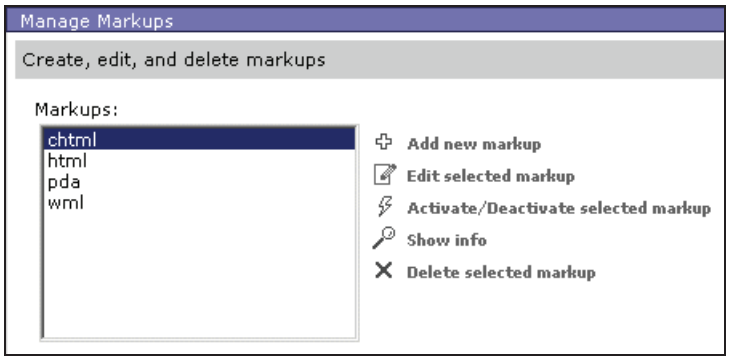


Figure 16-1 Manage Markups

2. .Click **Add new markup**.
3. Enter these three values (as shown in Figure 16-2):
   - Markup name: `xml`
   - MIME type: `text/xml`
   - Default character set: `UTF-8`

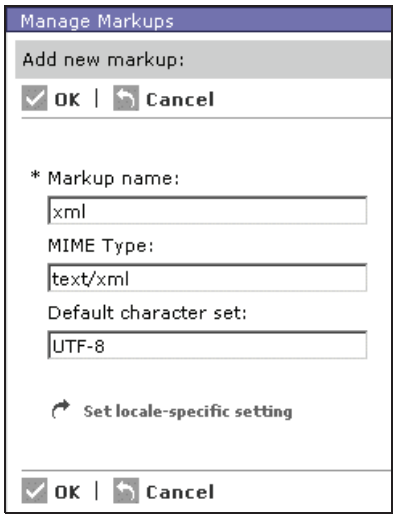


Figure 16-2 Manage Markup - enter values

4. Click **OK**.
5. The Manage Markup window returns a message that the markup was added successfully, but the markup is inactive.

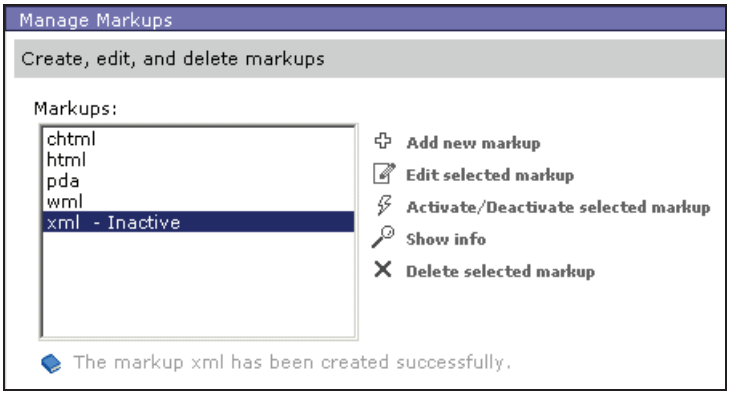


Figure 16-3 Manage Markup - xml added

6. To activate the xml entry select **Activate/Deactivate selected markup**. You will receive a message that the markup was activated, as shown in Figure 16-4.

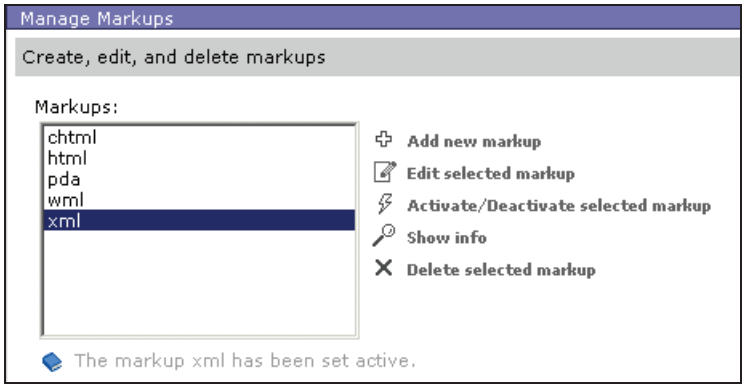


Figure 16-4 Manage markup - markup activated

7. To verify the markup status, select **Show Info** and the results are shown in Figure 16-5 on page 446. The new entry for XML is the last entry in the list.

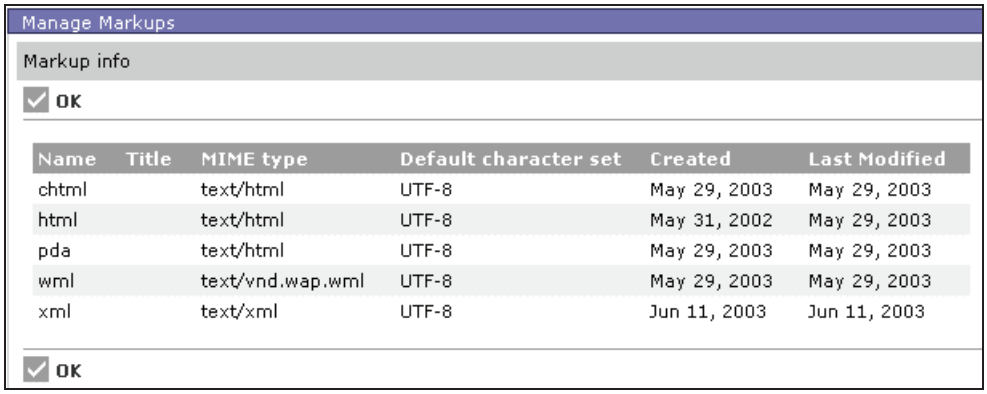


Figure 16-5 Manage markup - Show Info

16.3 Sample scenario 1: RSSNewsFeed

The RSSNewsFeed application accesses a syndicated news feed, in this case, the Wired News Web site at `http://www.wired.com`. The Wired News Web site is a technology and business-oriented news service. The data (generated by Wired News) is in RSS format, which is an XML dialect for syndicated news feeds, Web logs, and other dynamic Web information.

The RSSNewsFeed application consists of a portlet (RSSNewsFeed), which invokes a JSP (RSSNewsFeed.jsp), which links to the Web site, gets the data stream (which is XML) by accessing `http://www.wired.com/news_drop/netcenter/netcenter.rdf`, and sends that stream for display.

Obviously, the Portal does not display XML, so it is necessary to convert this XML data stream for display. To accomplish this, create a stylesheet to convert the RSS data stream to HTML. The sample stylesheet is shown in Example 16-2 on page 447.

Example 16-2 RSS stylesheet

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:param name="textLinksPreferredToImages" select="'true'"/>
 <xsl:output method="html"/>
 <xsl:template match="/">
<html>
 <body>
  <xsl:apply-templates select="/rss/channel/image"/>
  <ul>
   <xsl:for-each select="/rss/channel/item">
    <li><a>
      <xsl:attribute name="href">
       <xsl:value-of select="link"/>
      </xsl:attribute>
      <xsl:value-of select="title"/>
    </a></li>
   </xsl:for-each>
  </ul>
  <p><em>
   <xsl:value-of select="/rss/channel/copyright"/>
  </em></p>
 </body>
</html>
 </xsl:template>

 <xsl:template match="/rss/channel/image">
  <xsl:choose>
   <xsl:when test="$textLinksPreferredToImages='1'">
    <a>
     <xsl:attribute name="href">
      <xsl:value-of select="link"/>
     </xsl:attribute>
     <xsl:value-of select="title"/>
    </a>
   </xsl:when>
   <xsl:otherwise>
    <a>
     <xsl:attribute name="href">
      <xsl:value-of select="link"/>
     </xsl:attribute>
     <img border="0">
      <xsl:attribute name="src">
       <xsl:value-of select="url"/>
      </xsl:attribute>
     </img>
    </a>
   </xsl:otherwise>
```

```
  </xsl:choose>
 </xsl:template>
</xsl:stylesheet>
```

16.3.1 Configuring the RSSNewsFeed portlet

There are two ways to configure the portlet to use a stylesheet, which are:

1. Add the parameters to the portlet.xml
2. Add the parameters to the installed portlet (using the Manage Portlets - modify parameters option)

We included the parameters in the portlet.xml file, which is shown in Example 16-3.

Example 16-3 config-RSS

```
<config-param>
      <param-name>FilterChain</param-name>
      <param-value>Transcoding</param-value>
</config-param>

<config-param>
      <param-name>StylesheetFile</param-name>
      <param-value>WEB-INF/RSSNews.xsl</param-value>
</config-param>
```

The stylesheet can be associated with the portlet two ways:

1. A stylesheet file is added to the WAR file, as shown in Example 16-3. Notice it is relative to the WAR file root (in this case WEB-INF).
2. A stylesheet file is located on the server and is defined using the location on the server, as shown in Example 16-4. In this case the file would be located in the c:/Program Files/IBM HTTP Server/htdocs (theServer).

Example 16-4 Stylesheet parm - on server

```
<config-param>
      <param-name>StylesheetFile</param-name>
      <param-value>http://theServer/RSSNews.xsl</param-value>
</config-param>
```

Tip: Make sure the Portlet supports the markup of XML by adding a markup element with a name=”xml” attribute to the support element in the portlet.xml file.

With the stylesheet created and the portlet installed and added to the Welcome page, we are ready to view the results, shown in Figure 16-6. The StylesheetFile parameter, defined as a config-param, works fine in this instance because only one stylesheet is used in this portlet.

The RSSNewsFeed portlet results are shown in Figure 16-6.

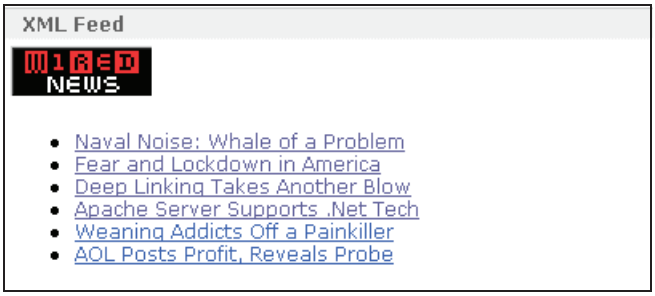


Figure 16-6 RSSNewsFeed portlet results using StylesheetFile parameter

16.4 Sample scenario 2: ITSONewsBrief

The ITSONewsBrief portlet is similar to the RSSNewsFeed portlet in that the JSP accesses an XML file and sends the XML file to the output stream. The ITSONewsBrief application consists of a portlet (ITSONewsBrief) that invokes a JSP (ITSONewsBrief.jsp), which reads the latest NewsBrief.xml file.

To display XML file content on various devices, you will need to create stylesheets to generate the markup language required for each of the target devices. Within the XML file, wtp-condition statements are included that specify both the condition(s) and the location and name of the stylesheet to be used. The NewsBrief.xml file is shown in Example 16-5 on page 450. The wtp-condition statements are located at the beginning of the XML file after the XML version statement.

Example 16-5 NewsBrief.xml file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?wtp-condition stylesheet="http://mka6brra.itso.ral.ibm.com/ITSONews2HTML.xsl"
condition="(user-agent=*MSIE 6.0*)"?>
<?wtp-condition
stylesheet="http://mka6brra.itso.ral.ibm.com/ITSONews2PcktPC.xsl"
condition="(user-agent=*Windows CE*)" ?>
<?wtp-condition stylesheet="http://mka6brra.itso.ral.ibm.com/ITSONews2Wap.xsl"
condition="(user-agent=*Nokia*)" ?>
<newsitem>
<title>"Java gets pervasive computing percolating"</title>
<story>
  <location>NEW YORK </location>
  <releaseDate>June 10, 2003</releaseDate>
  <shortDetail>IBM dealt a blow to both Microsoft and Sun today, announcing
that leading device and telematics players Palm, Nokia and QNX
will integrate IBM middleware and tools into current and future devices and
solutions.
  </shortDetail>
  <fullDetail>
  <paragraph>IBM dealt a blow to both Microsoft and Sun today, announcing that
leading device and
telematics players Palm, Nokia and QNX will integrate IBM middleware and tools
into current and
 future devices and solutions.</paragraph>
<paragraph>The announcement hampers Microsoft's effort to move its embedded
pervasive device
software to mobile devices. Microsoft needs to extend operating system
dominance from desktops -- a
mature market -- into the enterprise, where it can link up with Windows
servers. Microsoft hopes
to extend its proprietary approach in this way, offering built-in connectivity
and easy-to-use
development tools to facilitate its strategy.</paragraph>
<paragraph>Mobile devices represent a significant opportunity. The worldwide
market is expected to
grow from $73 billion this year to $195 billion by 2007.</paragraph>
<paragraph>By building their devices and solutions around IBM's Java-based
software, companies
like Palm, Nokia, and QNX are endorsing two of the key tenets of the on demand
era: open standards
and integration. Our pervasive computing strategy enables any device to access
any service on any server,
linked by open and industry standards. Tested middleware connects devices to
enterprise business
processes, and intelligent devices avoid server access bottlenecks. Devices can
also be maintained
and updated, over the air or over the wire.</paragraph>
```

```
</fullDetail>
</story>
</newsitem>
```

As you can see from the example, the wtp-conditions each apply to a different device and each have an associated stylesheet. The wtp-conditions and associated stylesheets (located on the server) are shown in Table 16-1.

Table 16-1 Sample conditions (wtp-conditions)

wtp-condition	Associated stylesheet
user-agent=*MSIE 6.0*	ITSONews2HTML.xsl
user-agent=*Windows CE*	ITSONews2PcktPC.xsl
user-agent=*Nokia*	ITSONews2Wap.xsl

Transcoding Technology can use the information from the HTTP header and the device profile to determine which stylesheet to apply. Using the wtp-condition makes it very easy to manage a set of stylesheets that need to be applied to an XML file.

Tip: For simplicity reasons, we located the stylesheets in the <IBM HTTP Server>/htdocs directory. However, you should use a proper folder for these files.

16.4.1 Sample stylesheets

The goal is to have Transcoding Technology select the stylesheet appropriate to the device making the request. The wtp-condition gives you the ability to define the conditions and the XSL files to use in the XML file.

Internet Explorer stylesheet

First, let us look at the stylesheet for an Internet Explorer browser, which is shown in Example 16-6 on page 452. If you remember, the XML file (shown in Example 16-5 on page 450) contains two types of story content: a <shortDetail> element and a <fullDetail> element. The <shortDetail> was a very brief key statement from the story; this content was intended for mobile devices. The <fullDetail> contains the complete story and is intended for Web browsers. This stylesheet displays the title, location, date, and full story.

Note: Since there is space within the Portal page, the full story will be displayed.

The stylesheet is shown in Example 16-6.

Example 16-6 Stylesheet for IE browser

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.w3.org/1999/xhtml"
  version="1.0">

<xsl:template match="newsitem">
  <html>
  <body bgcolor="#FFFFFF">
  <center> <h2>ITSO Breaking News</h2> </center>
  <xsl:apply-templates/>
  <!-- footer -->
  <center>
  <table border="0">
    <tr align="center"><td><b>
IBM International Technical Support Organization</b></td></tr>
    <tr align="center"><td><a
href="www.redbooks.ibm.com">http://www.redbooks.ibm.com</a></td></tr>
  </table>
  </center>
  </body>
  </html>
</xsl:template>

<xsl:template match="title">
        <h3>
       <xsl:value-of select="."/>
      </h3>
</xsl:template>

<xsl:template match="story/location">
       <i>
       <xsl:value-of select="."/>
      </i>
</xsl:template>

<xsl:template match="story/releaseDate">
       <b>
       <xsl:value-of select="."/>
      </b>
</xsl:template>
<xsl:template match="story/shortDetail">
       <br/>
</xsl:template>
<xsl:template match="story/fullDetail/paragraph">
       <p>
```

```
       <xsl:value-of select="."/>
      </p>
</xsl:template>
</xsl:stylesheet>
```

The results are shown in Figure 16-7.



Figure 16-7 NewsBrief.xml - Results in IE browser

Pocket PC stylesheet

Next you will look at the stylesheet for the Pocket PC and the results of applying the stylesheet to the XML file. The sample stylesheet is shown in Example 16-7.

Example 16-7 Pocket PC stylesheet

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<!-- template for each newsitem -->
```

```
<xsl:template match="newsitem">
      <html>
  <body bgcolor="#FFFFFF">
  <center> <h3>ITSO Breaking News</h3> </center>
  <br/>
      <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>

<xsl:template match="title">
        <h4> <xsl:value-of select="."/> </h4>
      <br/>
</xsl:template>

<xsl:template match="story">
     <p> <xsl:value-of select="shortDetail"/> </p>
</xsl:template>
</xsl:stylesheet>
```

As you can see from the stylesheet, the rule with `match="story"` contains `value-of select = shortDetail`, which causes the shortDetail element content to be put in the output stream. The results displayed on the Pocket PC are shown in Figure 16-8.

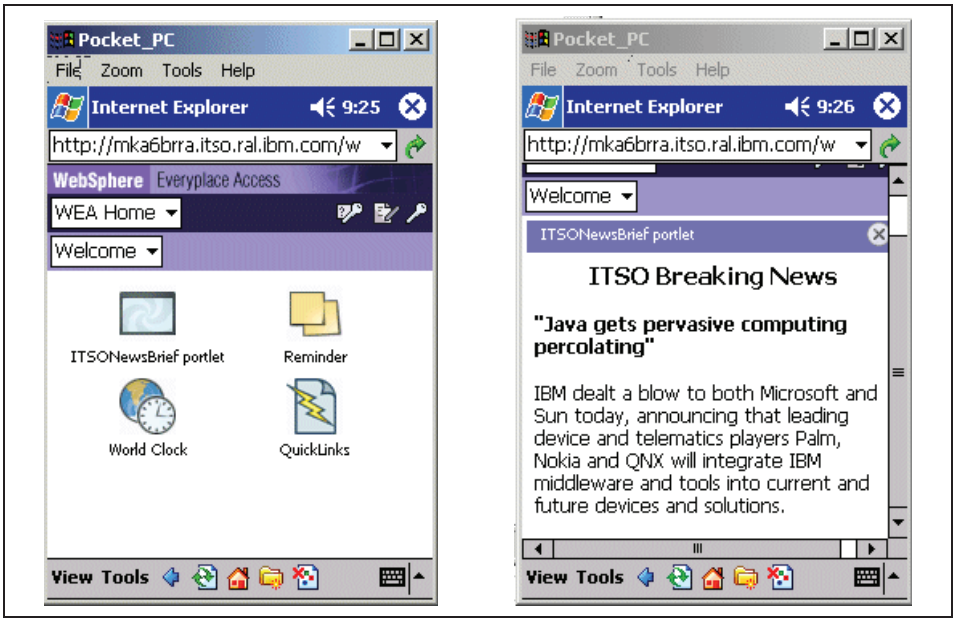


Figure 16-8 NewsBrief.xml - Results in Pocket PC

Tip: Make sure in the portlet.xml file to set the <supports> element's child markup to XML (<markup name="xml">) and have XML as the *only* markup. Otherwise transcoding will not work. Also, be sure to set a config-param parameter with the param-name as pda-icon and the param-value as the location of the icon image, which is necessary to display the portlet on the PDA.

Nokia Toolkit

The final stylesheet is for a WML device. The stylesheet is shown in Example 16-8.

Example 16-8 WML stylesheet

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<!-- template for each newsitem -->
<xsl:template match="newsitem">
<xsl:text disable-output-escaping="yes">
  &lt;!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http:///www.wapforum.org/DTD/wml_1.1.xml"&gt;
</xsl:text>
        <wml>
          <card>
             <p>
                <xsl:apply-templates/>
              </p>
          </card>
        </wml>
</xsl:template>
<xsl:template match="title">
      <strong>
        <b>
         <xsl:value-of select="."/>
        </b>
      </strong>
      <br/>
</xsl:template>
<xsl:template match="story">
         <xsl:value-of select="shortDetail"/>
</xsl:template>
</xsl:stylesheet>
```

As in the Pocket PC example, the shortDetail element is selected as the story body. The results are shown in Figure 16-9 on page 456.

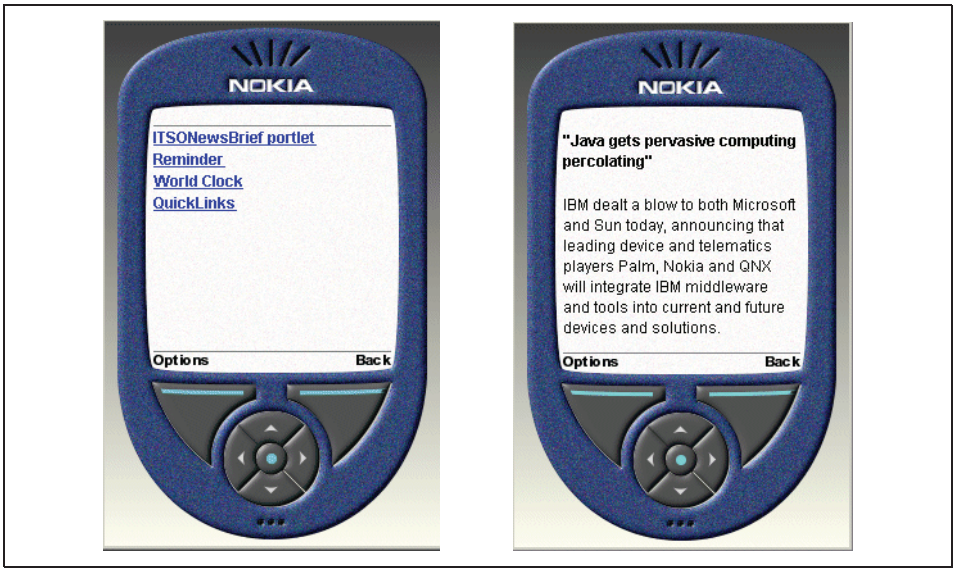


Figure 16-9 NewsBrief.xml in Nokia emulator

16.5 XML tools

The WebSphere Studio tools for XML and XML-related technology provide the means to create and maintain XML files. The full list of XML tools can be found in the WebSphere Studio documentation. In this section, we are interested in the tools that help us create XML, DTD and XSL files. The tools we will be using are:

- XML editor: To create, view and maintain XML files
- DTD editor: To create and view Document Type Definitions (DTD)
- XSL editor: Create new XSL files or edit existing XSL files
- XPath Expression wizard: Create XPath expressions
- XSL Debug and Transformation Tool: To apply XSL files to XML files transforming the XML file(s) to new XML, HTML, or text files
- XML to XML mapping editor: Maps one or more source XML files to a single target XML file

We created a simple project called AXMLProject to contain our DTD, XML, and XSL files.

16.5.1 DTD editor

The DTD editor can be used to create a new DTD and to view an existing DTD. Once a DTD is created, that DTD can be used to generate XML schema files. The DTD editor can also be used to generate a default HTML form based on the DTD.

A DTD or XML schema file should be the basis for creating related XML files. Either of these file types define the structure and possible content for the XML file. Either the DTD or XML schema can be used to validate an XML file, and more importantly used to ensure that the XML file is valid. In essence, the DTD or XML schema provide the specifications for the XML file.

Creating a DTD

Within our AXMLProject, we select **File -> New -> Other** to get to the New window. In the New window, we select **XML** (in the left pane) and select **DTD** (in the right pane) as shown in Figure 16-10. Click **Next** to further define the DTD.

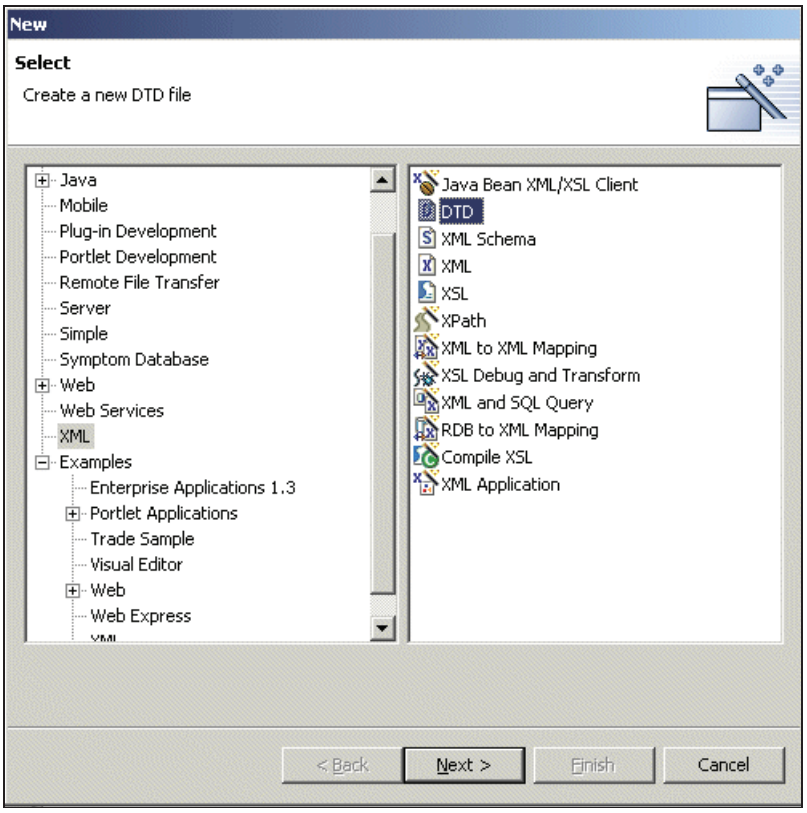


Figure 16-10 Select DTD

The Create DTD wizard provides two ways to generate a DTD:

- Create DTD from scratch
- Create DTD file from an XML file

In this case we are creating the DTD from scratch. Select the **Create DTD from scratch** radio button and click **Next** to process.

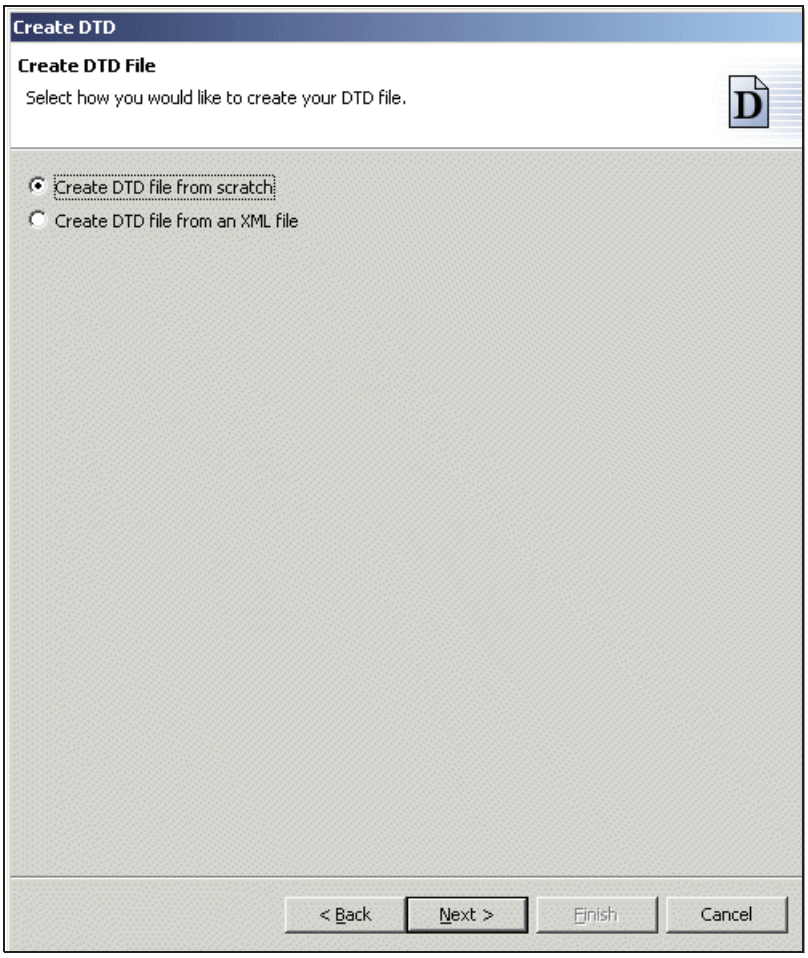


Figure 16-11 Create DTD from scratch

The final step in creating the DTD is to select the folder **AXMLProject** and name the file `AddressBook.dtd`, then click **Finish**.

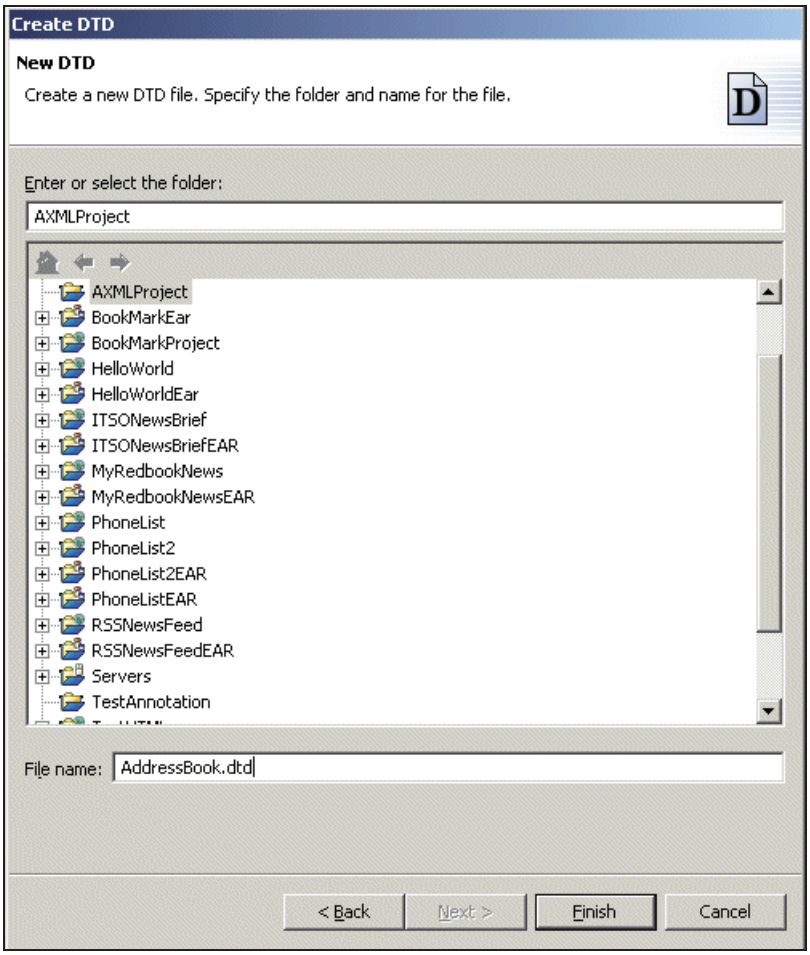


Figure 16-12 Name DTD

Using the DTD editor

The DTD editor opens with the new AddressBook.dtd displayed. The only entry is the `<?xml version="1.0" encoding="UTF-8">` element. The DTD editor has two tabs at the bottom of the editor: one is to work with the DTD in the Design view and the other tab provides the Source view. Figure 16-13 on page 460 shows the DTD editor in the Source view.

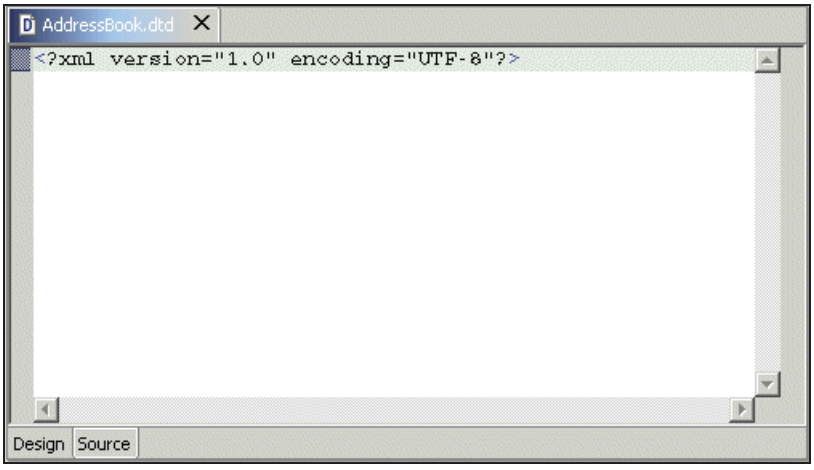


Figure 16-13 DTD editor in Source View

Now we are ready to input the contents of the DTD. The DTD consists of the following elements and attributes:

- AddressBook (root element)
- Address (one per address entry, which contains Name & Address elements)
- Name (complete name of the person or company)
- AddressDetail (contains address details)
  - MailAddress, which contains:
    - Street
    - City
    - State
    - Zip code
  - eMail
  - Phone, has an attribute TYPE with a value of home, mobile or work

To create an element, click the **Element** icon on the toolbar.

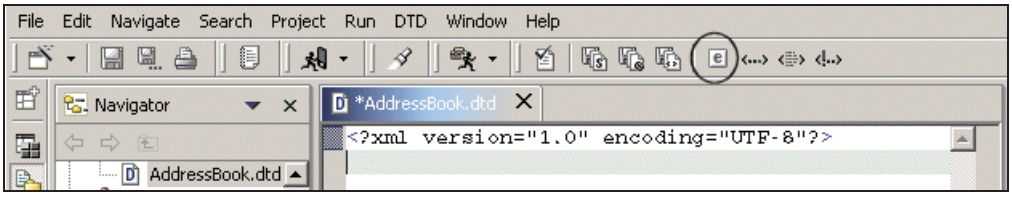


Figure 16-14 Create an element

Switch to Design mode and enter the element name as shown in Figure 16-15 on page 461.

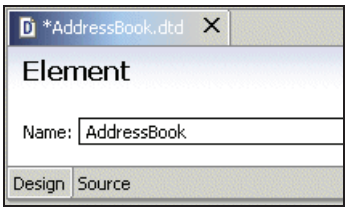


Figure 16-15 Enter Element Name AddressBook

As illustrated in Figure 16-16, you can switch to the Source view to see the actual content.

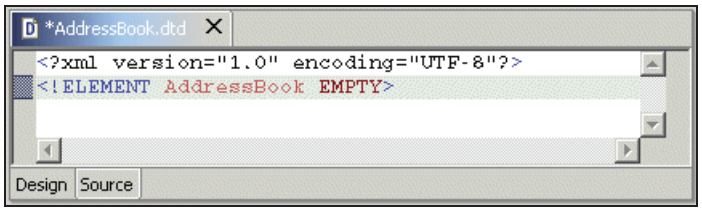


Figure 16-16 Element AddressBook in Source view

You can create all the elements by using the same steps:

- Click the **Element** icon to create a skeleton element.
- In Design view, replace the Name: Element with your element name.
- Continue this process until all elements have been created.

The Phone element has an attribute named TYPE, which can have one of these values (home, work, mobile). To add an attribute to the Phone element, right-click the **Phone** element in the Outline view (as shown in Figure 16-17 on page 462). Select **Add Attribute** to add an attribute to the Phone element.

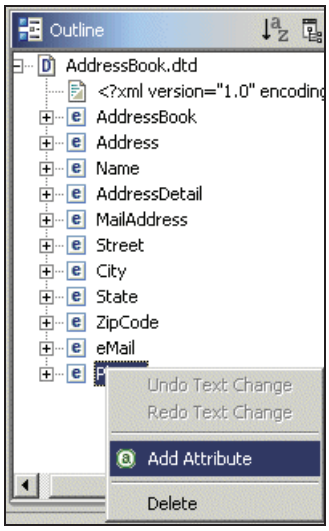


Figure 16-17 Add attribute

An attribute is added to the Phone element in both the editor and the Outline windows. Highlight the attribute in the editor and switch to the Design view. In the Design view, enter the Name of `TYPE` and the Type of `Enumerated Name Tokens` and in the Enumerated defaults add `home`, `work`, and `mobile`. Finally highlight **home** and check **Default**. These entries are reflected in Figure 16-18 on page 463.

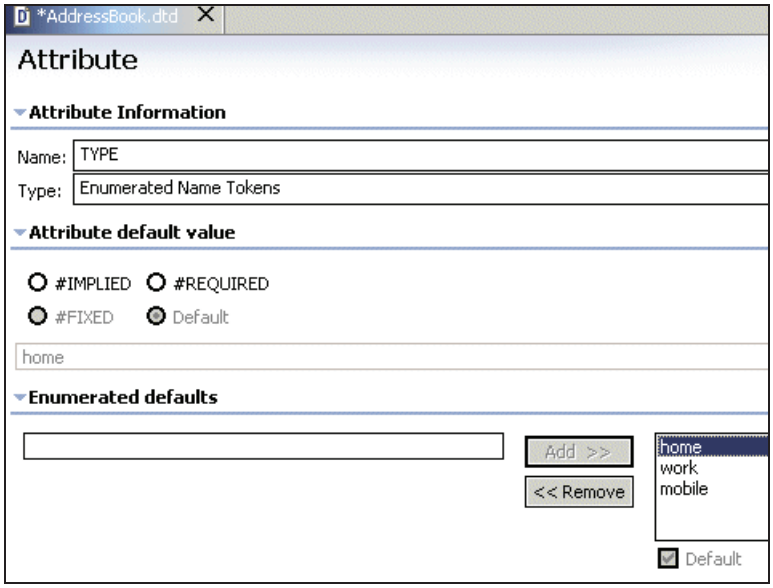


Figure 16-18 Edit Attribute details

As each element was created it was added to the Outline view of the DTD, which is shown in Figure 16-19.

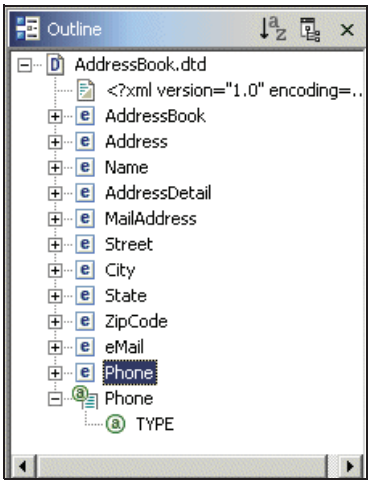


Figure 16-19 DTD elements in Outline view

Now we need to fill in the details of our DTD elements. Let's start by assigning the children elements to their parents. For example, the MailAddress has the

children Street, City, State and ZipCode. There are three ways to start the assignment of a child to its parent:

- Select an element's **EMPTY** value in the Source view and switch to the Design view to fill in the details.
- Expand an element in the Outline view and select **EMPTY**, then switch to the Design view in the editor.

The Design view with the EMPTY content mode type is shown in Figure 16-20.

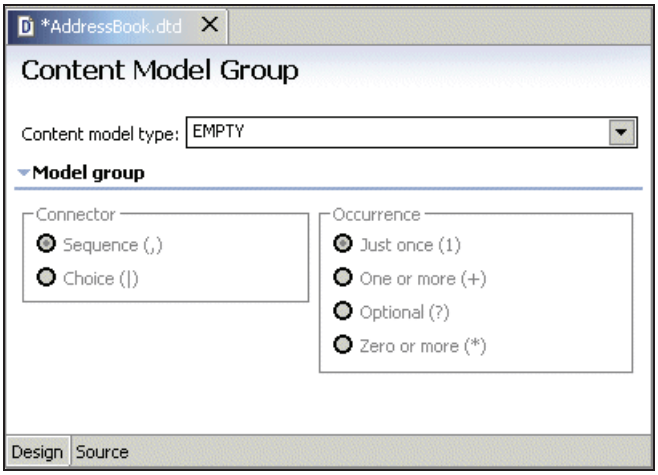


Figure 16-20 Empty Element in Design View

We selected the element MailAddress's EMPTY content mode type. MailAddress should have the children elements Street, City, State and ZipCode. To add its children, switch to Design mode. In the Content mode type, open the drop-down list (shown in Figure 16-21 on page 465) and select **Street**.

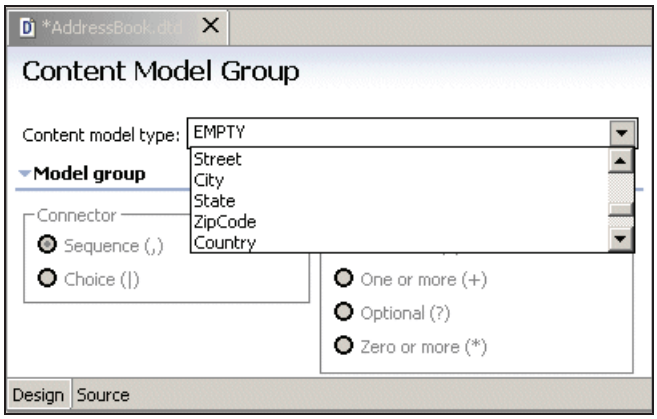


Figure 16-21 Select child

The Content Model window appears for the Street element (as shown in Figure 16-22). In this case, the **Just once (1)** radio button should be selected for the occurrence.

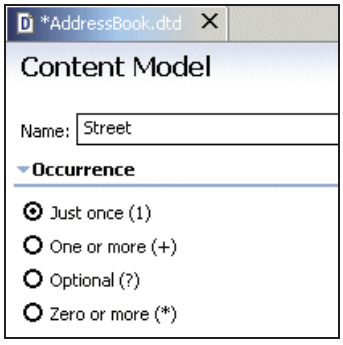


Figure 16-22 Street occurrence

To add the rest of the children for MailAddress, we entered each of them in the Source view with a comma delimiter. Once we had entered each child, we highlighted it and modified the occurrence (in the Design view) as needed. We continued this process until we had assigned all the children to their parents and had set each child's occurrence value. Table 16-2 on page 466 shows the elements, their occurrence value, their data type, and their parent. Those elements without data types are structural elements only.

Table 16-2 Element details

Element	Occurrence	Data type	Parent
AddressBook	Just once		(the root)
Address	One or more		AddressBook
Name	Just once	#PCDATA	Address
AddressDetail	Just once		Address
MailAddress	One or more		AddressDetail
Street	Just once	#PCDATA	MailAddress
City	Just once	#PCDATA	MailAddress
State	Just once	#PCDATA	MailAddress
ZipCode	Just once	#PCDATA	MailAddress
eMail	Zero or more	#PCDATA	AddressDetail
Phone	One or more	#PCDATA	AddressDetail

To assign the data types to the children elements, do the following:

1. Highlight **EMPTY** for a child element.
2. Switch to the Design view.
3. In the Content mode type drop-down list, select **(#PCDATA)** as shown in Figure 16-23.
4. Select the appropriate occurrence radio button.

Perform these steps for each child element.

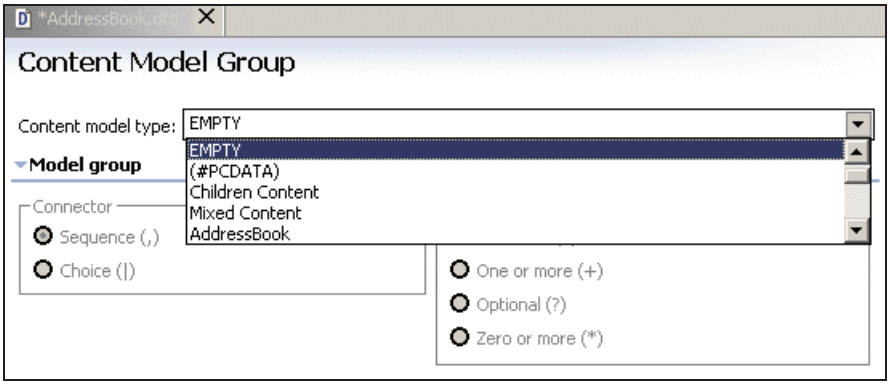


Figure 16-23 Select PCDATA

The final DTD in the DTD editor in the Source view is shown in Figure 16-24. The final DTD in the Outline view with those elements with children expanded is shown in Figure 16-25 on page 468.

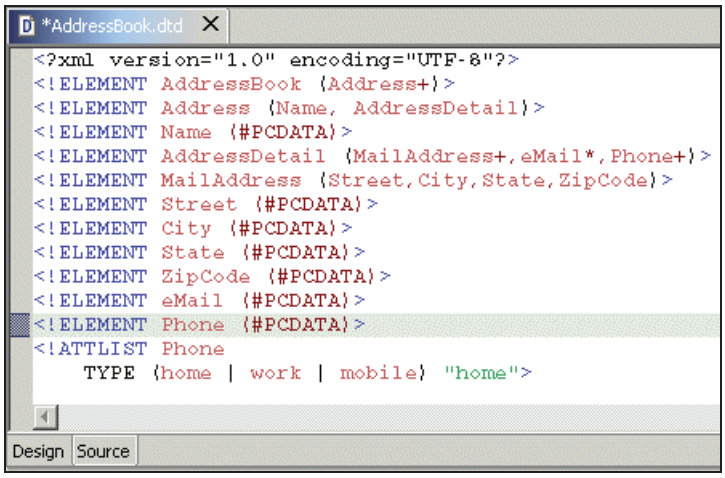


Figure 16-24 Final DTD in Source view

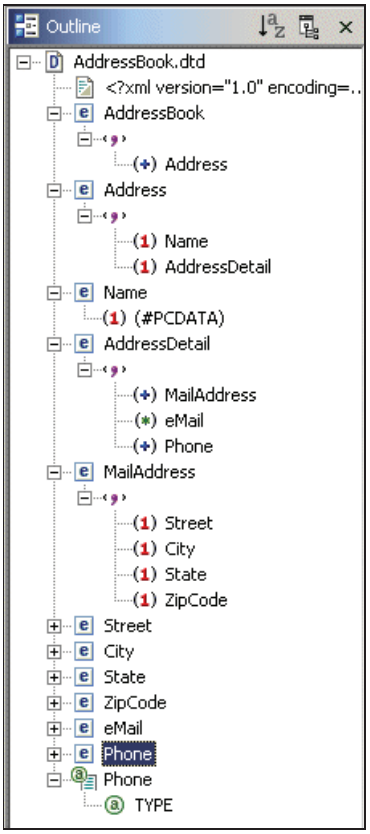


Figure 16-25 Final DTD in Outline view

The character in parentheses before the children listed under their parents (in the Outline view) identifies the occurrences. The occurrences identify how many of that element may exist for its parent.

- 1 indicates Just One
- + indicates One or more
- * indicates Zero or more
- ? indicates Optional

Make sure to save your new DTD.

Validating the DTD

It is important to make sure that the DTD is valid, especially if you import the DTD from an outside source or create it from an existing XML file. To validate a DTD, you can either:

- Right-click the **DTD** (in the project) and select **Validate DTD** from the pop-up menu.
- Highlight the **DTD** (in the project) and select the **Validate DTD** icon from the toolbar.

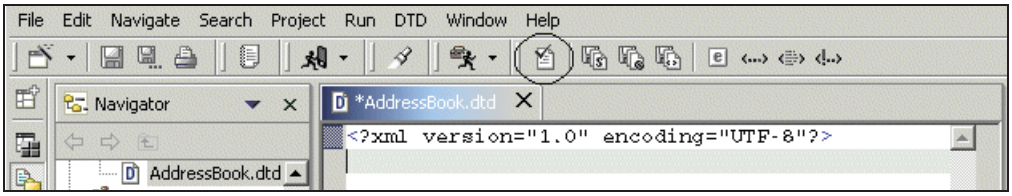


Figure 16-26 Validate DTD via toolbar entry

- Highlight the **DTD** and select **DTD** from the main menu bar.

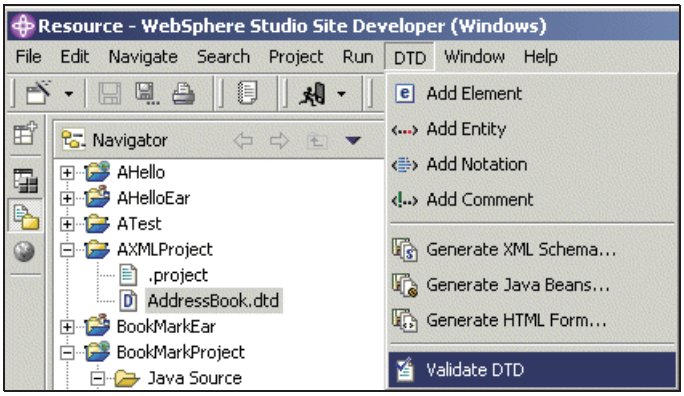


Figure 16-27 Validate DTD

Either way you decide to start validation, once the validation process is successful you will get a message indicating the DTD is valid. If the file is invalid, any errors will appear in the Tasks view.



Figure 16-28 Validate DTD message

16.5.2 XML editor

The XML editor is a tool used to create and view XML files. You can use the XML editor to perform these activities:

- Create a new XML file:
  - From scratch
  - From an existing DTD
  - From an existing XML schema
- Edit and validate XML files
- Import existing XML files for structured viewing
- Associate XML files with DTDs or XML schemas

Create an XML file

To create an XML file, select **File -> Other**. In the New window, select **XML** (in the left pane) and **XML** (in the right pane), and then click **Next**.

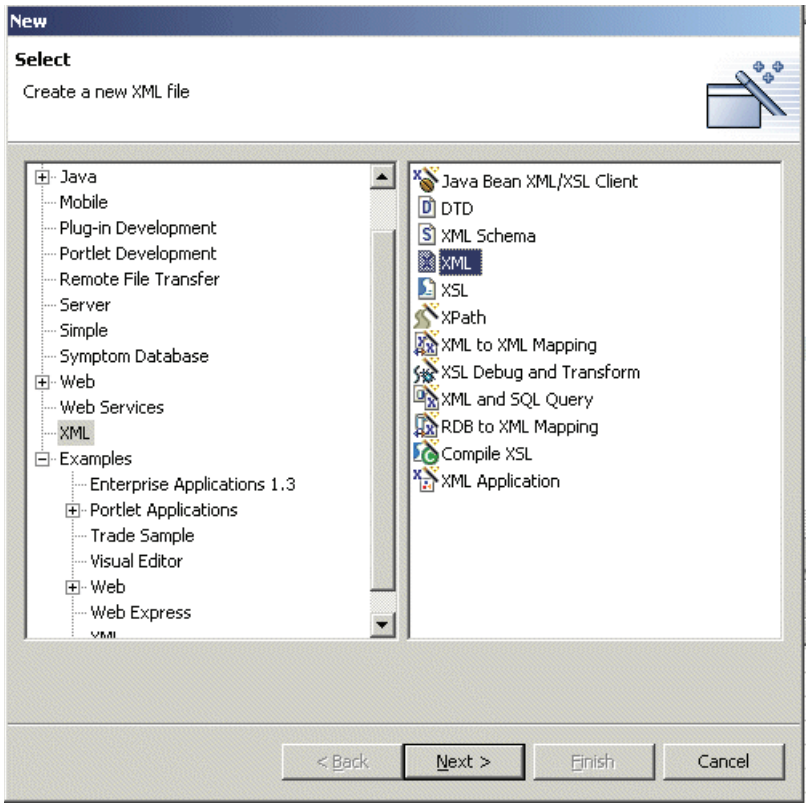


Figure 16-29 Select XML

We are creating an XML file for the AddressBook.dtd that we created in “Creating a DTD” on page 457. On the Create XML file window, select the **Create XML file from DTD file** radio button and click **Next**. Enter the name of the XML file. In this case we called the file MyAddressBook.xml. The folder should be AXMLProject. Click **Finish** to complete the process.

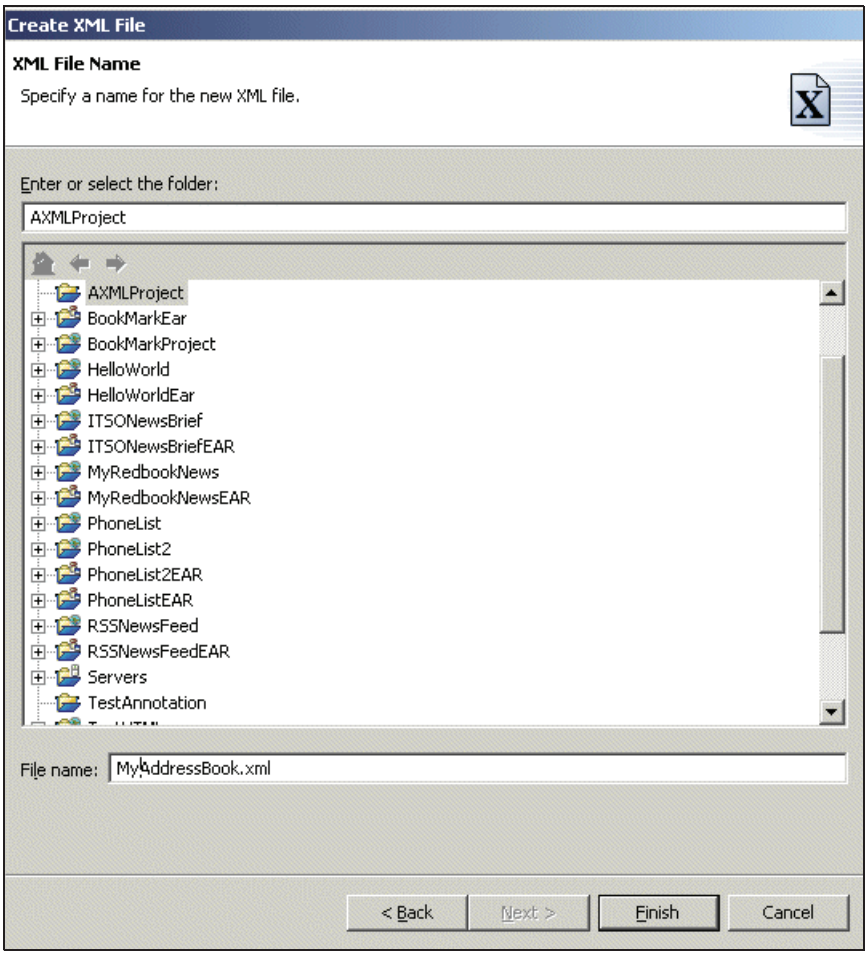


Figure 16-30 Name XML file

Next you must select the DTD file, select the **Select file from workbench** radio button and select **AddressBook.dtd** in the project list. Click **Next**.

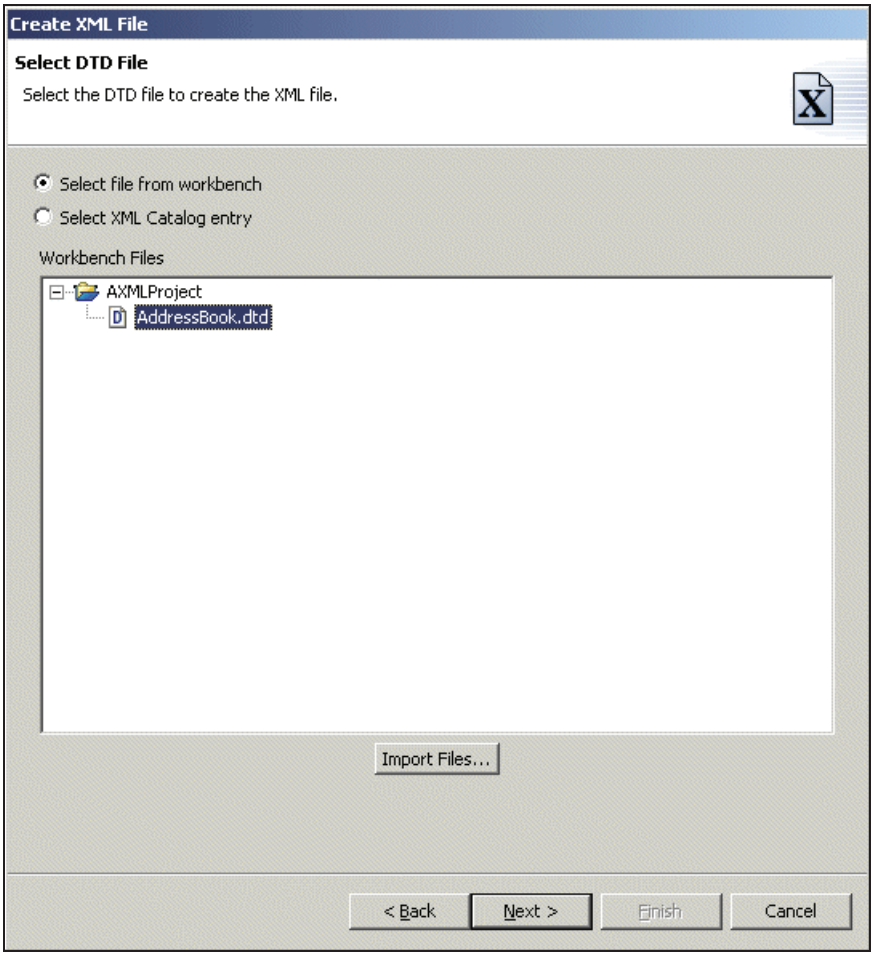


Figure 16-31 Select DTD

In the Select the Root Element window, the root element is AddressBook. Select the **Create required content only** radio button and leave the Public ID blank and the System ID as AddressBook.dtd. Finally, click **Finish**.

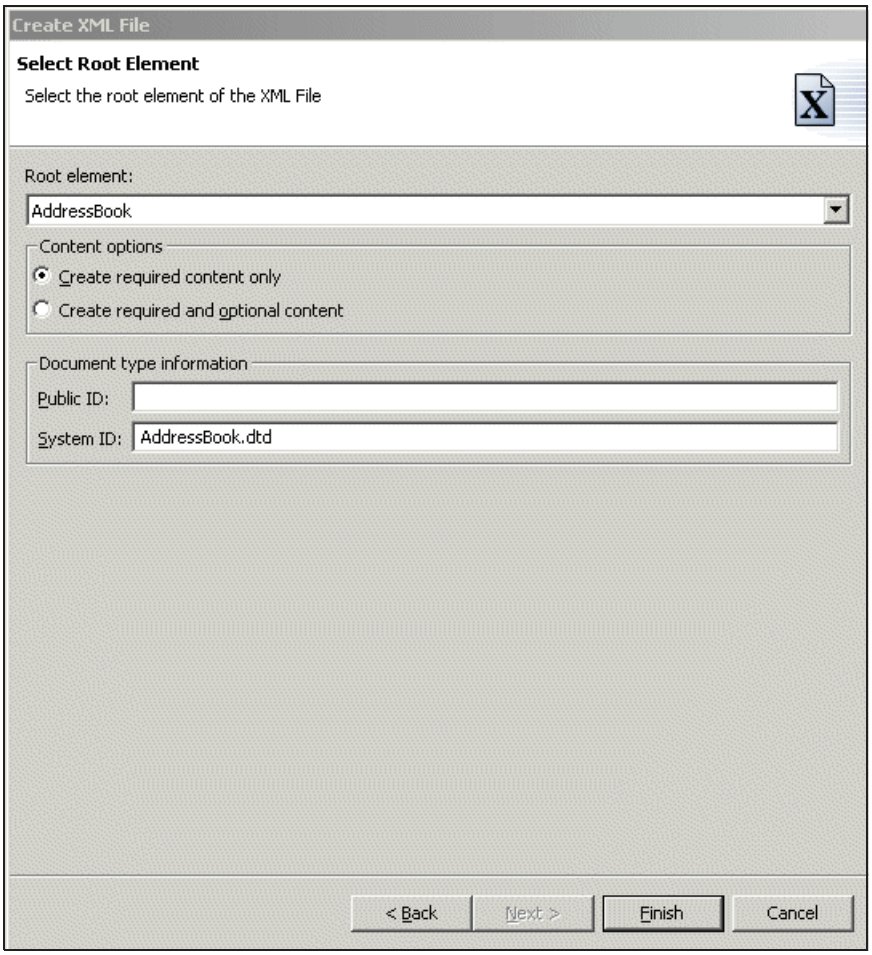


Figure 16-32 Select Root Element

The XML editor appears with the new skeleton version of the XML file displayed. The XML editor provides two views of the file:

- Source view: Allows viewing and working directly with the XML file's source XML. The Source view is shown in Figure 16-33 on page 475.
- Design view: Represents the XML simultaneously as a table and a tree. Content and attributes are edited directly in the table cells. The expanded Design view is shown in Figure 16-34 on page 475.

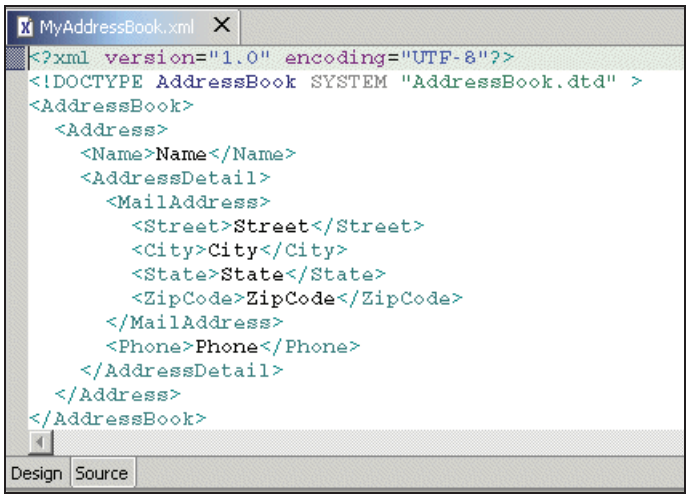


Figure 16-33 XML in Source view

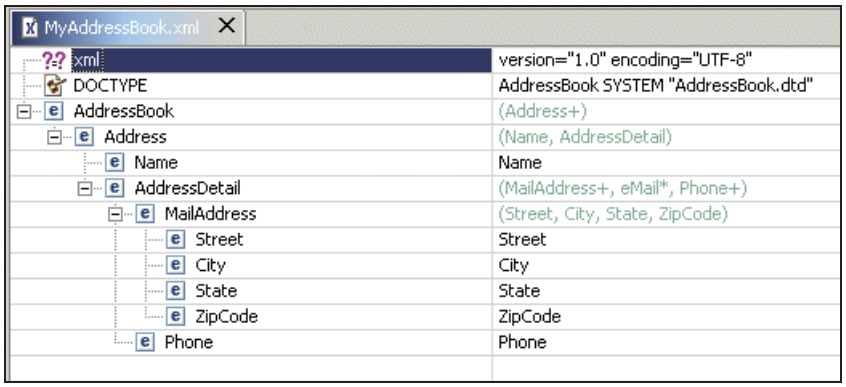


Figure 16-34 XML in Design view

Edit XML file

The easiest way to add content to the XML file is to edit the Design view. Just type the appropriate values in the right column of the table. The results of adding “Brian Pollak” are shown in Figure 16-35 on page 476.

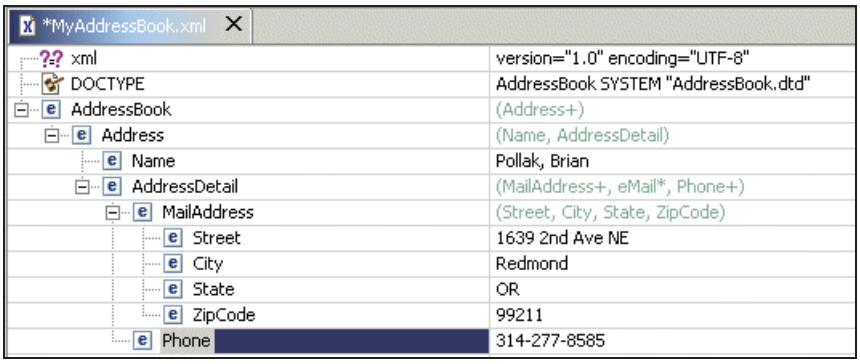


Figure 16-35 Add Brian Pollak

To add the attribute to the Phone element, right-click the **Phone** element and select **Add Attribute -> TYPE** in the pop-up menu (as shown in Figure 16-36). The attribute is added to the Phone element with the default value of home. To change the default value from home to a value of work, just type the new value work over the value of home, which is the default. This is shown in Figure 16-37.

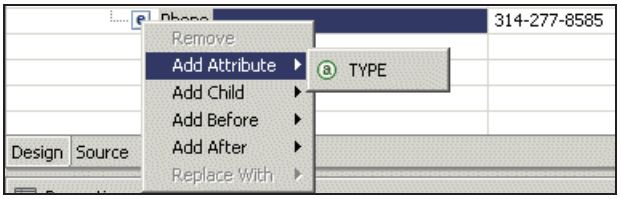


Figure 16-36 Get Phone Attribute

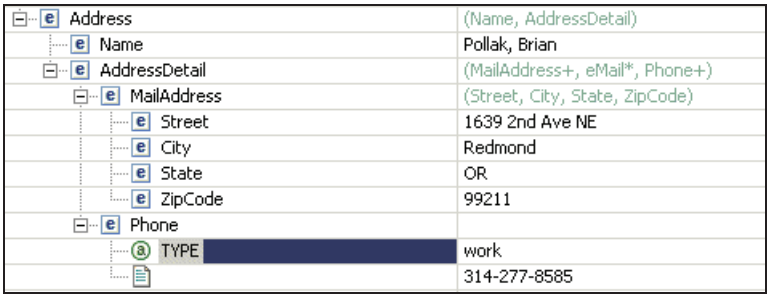


Figure 16-37 Add Phone attribute

To add another phone for Brian, right-click the **AddressDetail** element. In the pop-up menu, select **Add Child -> Phone**, as shown in Figure 16-38 on page 477.

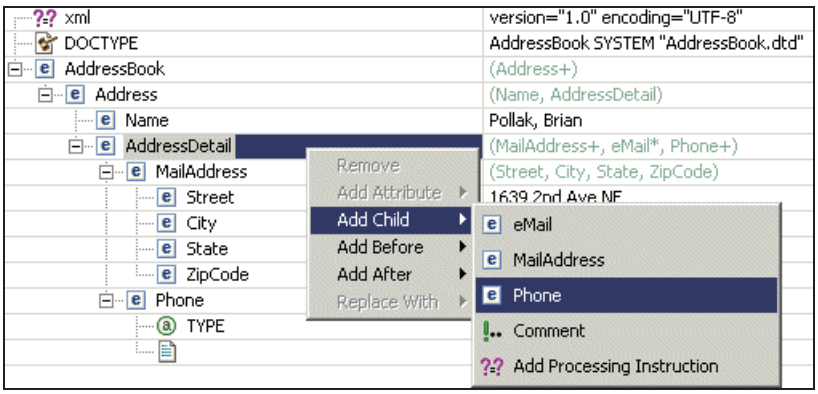


Figure 16-38 Add second phone

By selecting the child, Phone, a new Phone element is added to Brian's address information. Add the attribute as we did before. The new phone information is shown in Figure 16-39.

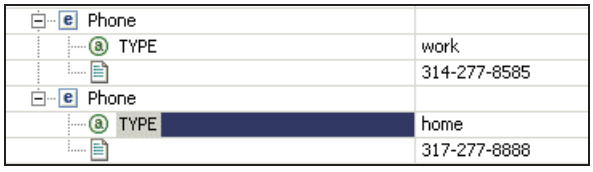


Figure 16-39 Second phone

The children of AddressDetail are eMail, MailAddress, and Phone. You might also notice that the eMail element was not part of the initial skeleton XML file (shown in Figure 16-33 on page 475) that is because eMail is an optional element. When we defined the XML file, we selected the **Create required elements only** radio button, so no optional elements were included (eMail is an optional element).

To create a new address or add a child to any parent, the same process is used:

1. Right-click the parent element.
2. Select the appropriate child from the pop-up list.
3. The editor creates a new instance of that child and its children (if there are any).

Figure 16-40 on page 478 shows the children that can be added to the AddressBook parent. This is how you would add a new address element or any children for a particular parent element.

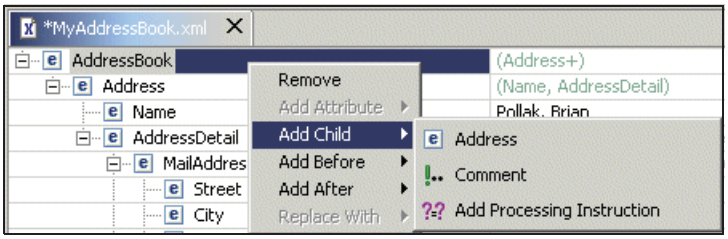


Figure 16-40 Add new address

We created two new address entries, as shown in Table 16-3.

Table 16-3 Additional address entries

Name	MailAddress	Phone	eMail
Bob White	28 Tree Lane Chicago, IL 55999	work = 202-333-2221 mobile = 222-315-8832	bwhite@itso.com
Teddie Bear	10 Pine Dr. Lakeville, MN 66001	work = 444-329-9080	teddie@itso.com

We now have three address entries in our XML file.

Validating the XML file

We can validate the XML file by either selecting **XML** from the main menu list and then selecting **Validate the current state of the XML file**, or by selecting the **XML icon** from the toolbar (as shown in Figure 16-41).



Figure 16-41 Select validate XML from toolbar

Once the validation is complete, you will receive a validation message as shown in Figure 16-42 on page 479. Click **OK** to close the Message window. If the file is invalid, any errors will appear in the Tasks view.



Figure 16-42 Validation message

16.5.3 XSL editor

The eXtensible Stylesheet Language Transformation (XSLT) is used to transform an XML file into another XML markup or HTML and to format the output. The XSL file defines the layout of the target markup and the location of the data in the source file. The selected content from the source becomes part of the target file and any constant data located inside a template rule.

The editor will work with existing XSL files or allow you to create a new XSL file for a target XML file. The editor provides assistance to help you edit the XSL file.

Creating an XSL file

To create an XML file, perform the following steps:

1. Select **File -> New -> Other**.
2. In the New wizard, select **XML** (in the left pane) and **XSL** (in the right pane) and click **Next**.
3. Select the folder (in this case **AXMLProject**) and enter the file name of `Address2HTML.xsl`.
4. Select an XML file to associate with the XSL file. We selected **MyAddressBook.xml**.
5. The XSL editor opens with the preliminary XSL file, as shown in Figure 16-43.



Figure 16-43 Skeleton XSL in editor

Editing an XSL file

When the XSL editor is active, new icons are added to the toolbar.



Figure 16-44 XSL editor icons

The XSL editor related icons are (from left to right):

- Validate current XSL file
- Create an XPath expression using XPath wizard
- Create a default template for an HTML header
- Create a table in the XSL
- Create a form in the XSL
- Add a template rule
- Add a call-template
- Add an apply-template
- Add conditional logic to the file
- Add the output method
- Debug the XSL Stylesheet
- Run transformation on the XSL
- Spell check
- Hide Hover Help

Some of these XSL wizards are also available from the main menu. Select **XSL** to see the list as shown in Figure 16-45.

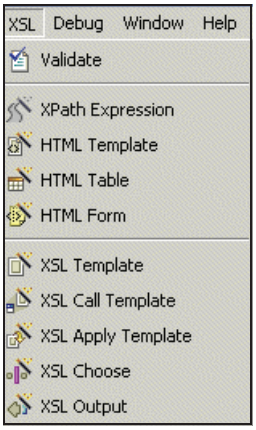


Figure 16-45 XSL editor options

The XSL file we are creating is going to convert the XML file to HTML for display in a Web browser. To do this we must add various template rules to the XSL file. We want the MyAddressBook.xml file converted to HTML with these HTML tags:

- Title with text My Address Book
- H3 with text My Address Book
- A table with this structure
  - Table Headings: Name, Mail Address, Phones
  - Table Rows: information (per column)
    - 1 - Name
    - 2 - All MailAddress elements
    - 3 - All Phones with type in parenthesis

First, let's create the default HTML document structure XSL template rule by selecting the **Create a default template for HTML** icon. This adds a new template rule that matches the root, which is reflected by the `match="/"` statement. The new template rule is shown in Figure 16-46.

*AddressBook2HTML.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0"
    xmlns:xalan="http://xml.apache.org/xslt">
    <xsl:output method="html" encoding="UTF-8"/>

<xsl:template match="/">
  <html>
    <head>
      <title>Untitled</title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Figure 16-46 Create default HTML template

We made changes in this template rule by replacing the term Untitled with `My Address Book` and adding the statement `<H3>My Address Book</H3>` after the <body> tag. When you start to type, an HTML tag content assist will pop up the HTML tag templates to help you build the HTML tag. The content assist helps with both the start tag and the end tag. Figure 16-47 on page 482 shows content assist on the Heading end tag.

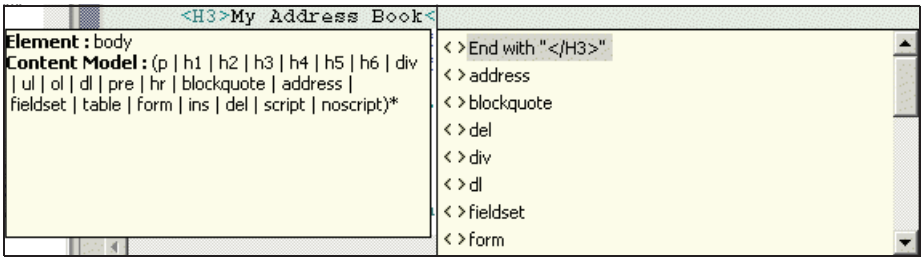


Figure 16-47 Content Assist to build H3 tag

Create Table Template Rule

The rest of the template rules support the creation of the table and loading it with the XML content. To create a table in the XSL file, select the **Create a Table in XSL** option. The XSL Table wizard appears with a skeleton HTML table defined.

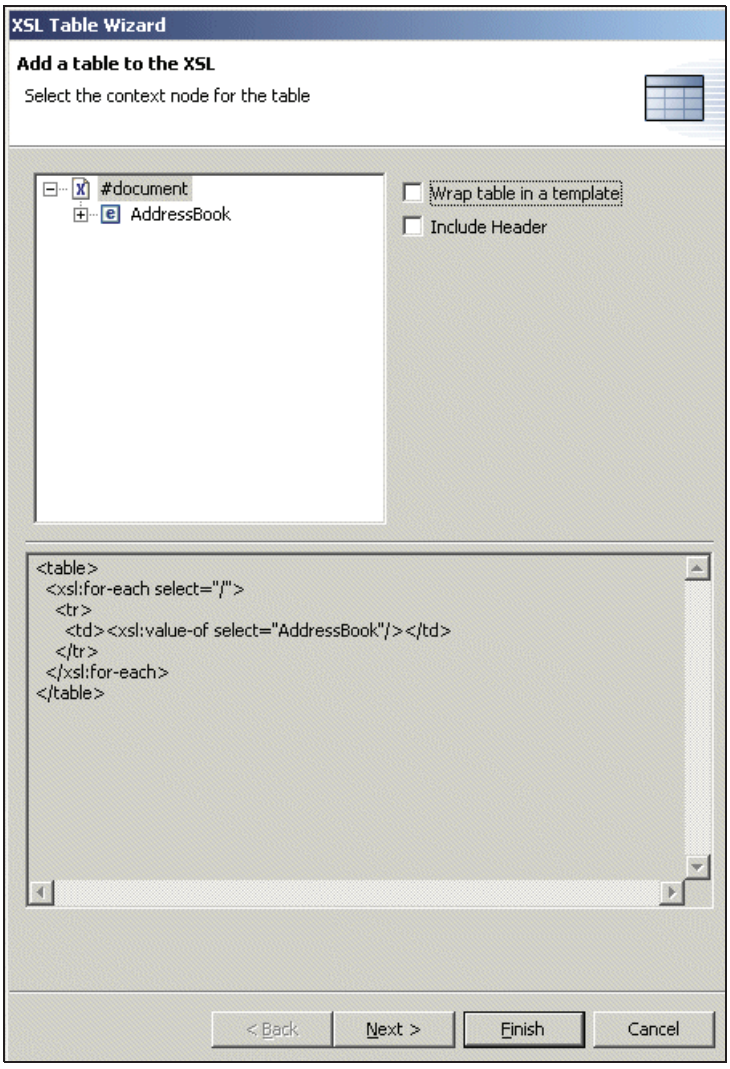


Figure 16-48 Table template

We customized the skeleton XSL template rule by doing these steps:

1. Expanded the AddressBook element to show the Address elements
2. Highlighted the first Address element. This modified the <:xsl:> for each select value from the root (/) to the value “/AddressBook/Address”.
3. Checked **Wrap table in a template**, which adds the template rule statements around the table. Note the match value is the term “AddressBook”.

4. Checked **Include Header**, which added a skeleton table heading before the table body.

The XSL Table Wizard window with these changes is shown in Figure 16-49.

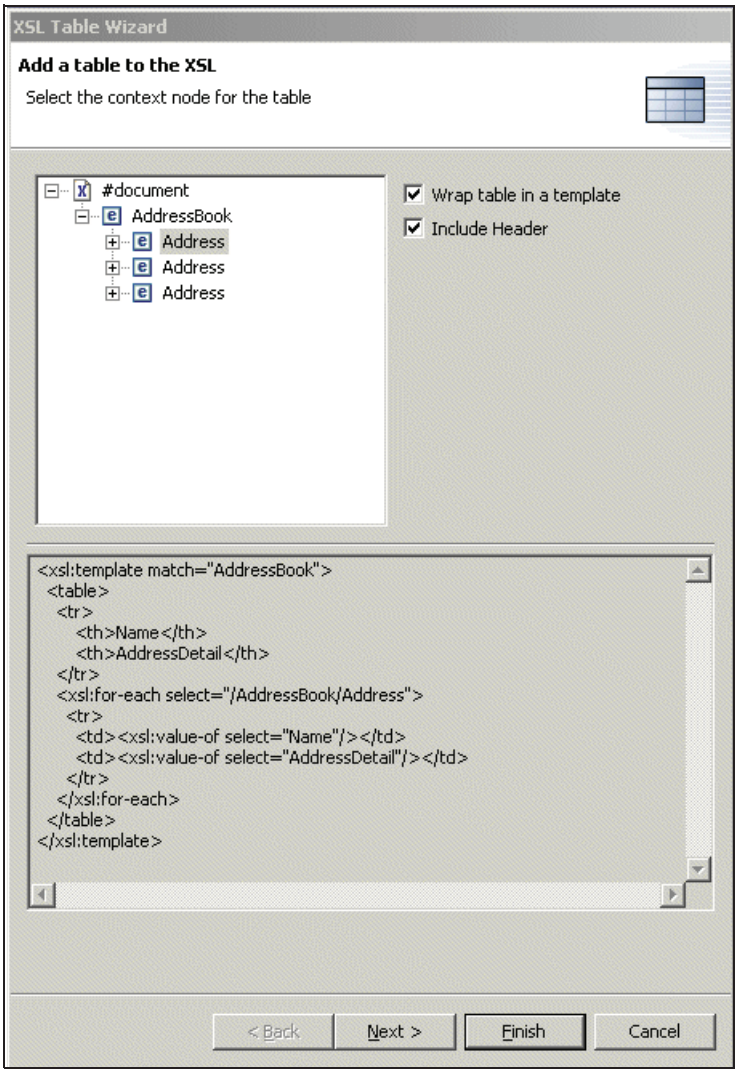


Figure 16-49 Change Table in XSL Table Wizard

Click **Next** to further customize the table. The XSL Table wizard displays table properties and row properties to make your table more attractive, as shown in Figure 16-50 on page 485.

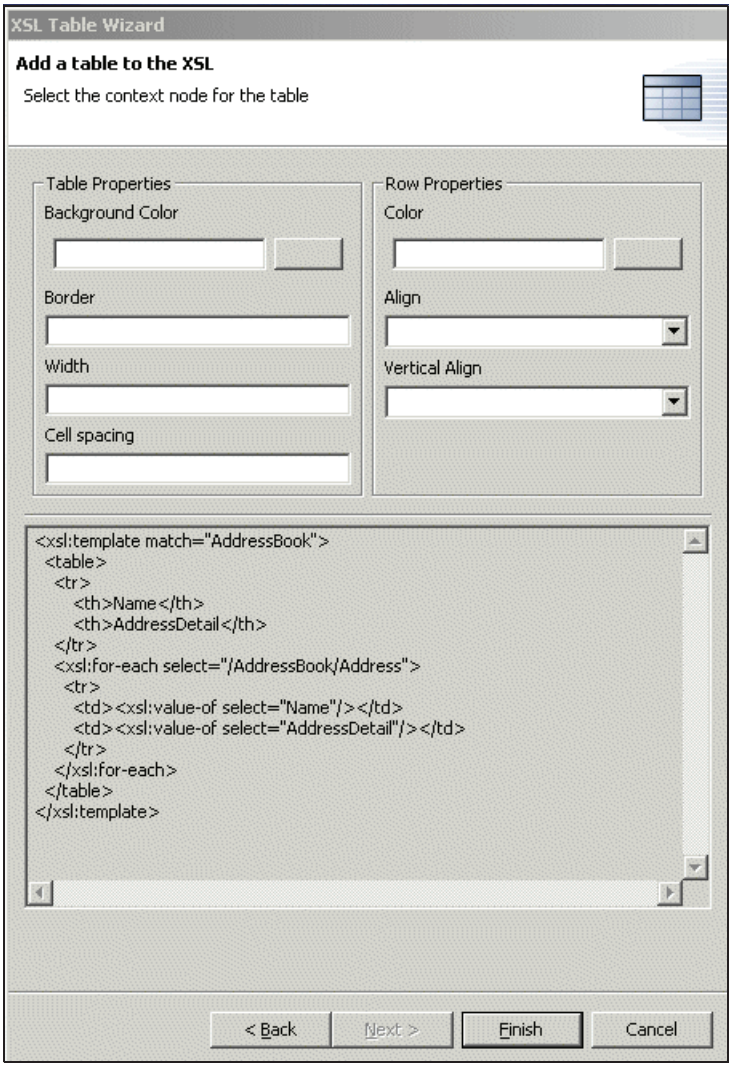


Figure 16-50 Add table details

If you click either the **Table Properties Background Color** or the **Row Properties Color** button, a color palette appears allowing you to select a standard color or your own custom color(s). This is shown in Figure 16-51 on page 486.



Figure 16-51 Table colors

We selected a background color for the table and a color for the rows. For the table you can also define a border, a width, and cell spacing. For the row properties you can also define the row alignment (center, char, justify, left) and the vertical alignment (baseline, bottom, middle, top) by selecting an entry from the drop-down list (shown in Figure 16-52). We changed the align (row alignment) to left and the Vertical Align to top.

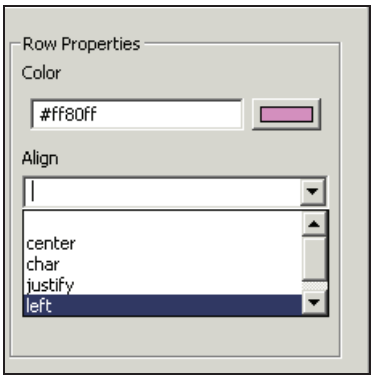


Figure 16-52 Row Property Align

The resulting template rule for the table is shown in Figure 16-53 on page 487. To complete the process of creating the template rule, click **Finish**.

```
<xsl:template match="AddressBook">
  <table>
    <tr>
        <th>Name</th>
        <th>AddressDetail</th>
    </tr>
    <xsl:for-each select="/AddressBook/Address">
      <tr>
        <td><xsl:value-of select="Name"/></td>
        <td><xsl:value-of select="AddressDetail"/></td>
      </tr>
    </xsl:for-each>
  </table>
</xsl:template>
```

Figure 16-53 Results of XSL Table Wizard

The XSL Table wizard created the HTML table tags we need, but some of the tags need to be moved to other template rules and additional tags must be added in order to create the HTML table the way we want. We accomplished this by editing the template rules in the XSL editor Source view. We made the following changes by moving tags from the template rule for the table to the root template rule:

- Move <table> and <th> tags before the <xsl:apply-templates/>
- Move the </table> tag after the <xsl:apply-templates/>
- Change the <th> tag from the value of AddressDetail to Mail Address
- Add a new table heading Phones

These changes are reflected in Figure 16-54.

```
<xsl:template match="/">
  <html>
    <head>
      <title>My Address Book</title>
    </head>
    <body>
      <H3>My Address Book</H3>
      <table bgcolor="#80ffff">
       <tr bgcolor="#ff80ff" align="left" valign="top">
        <th>Name</th>
        <th>Mail Address</th>
        <th>Phones</th>
        </tr>
         <xsl:apply-templates/>
       </table>
    </body>
  </html>
</xsl:template>
```

Figure 16-54 Modified Root template rule

Create template rule

Now we will add the rest of the template rules needed to fill the table rows. To do this select the **Add a template** rule option. The XSL Template wizard appears, as shown in Figure 16-55.

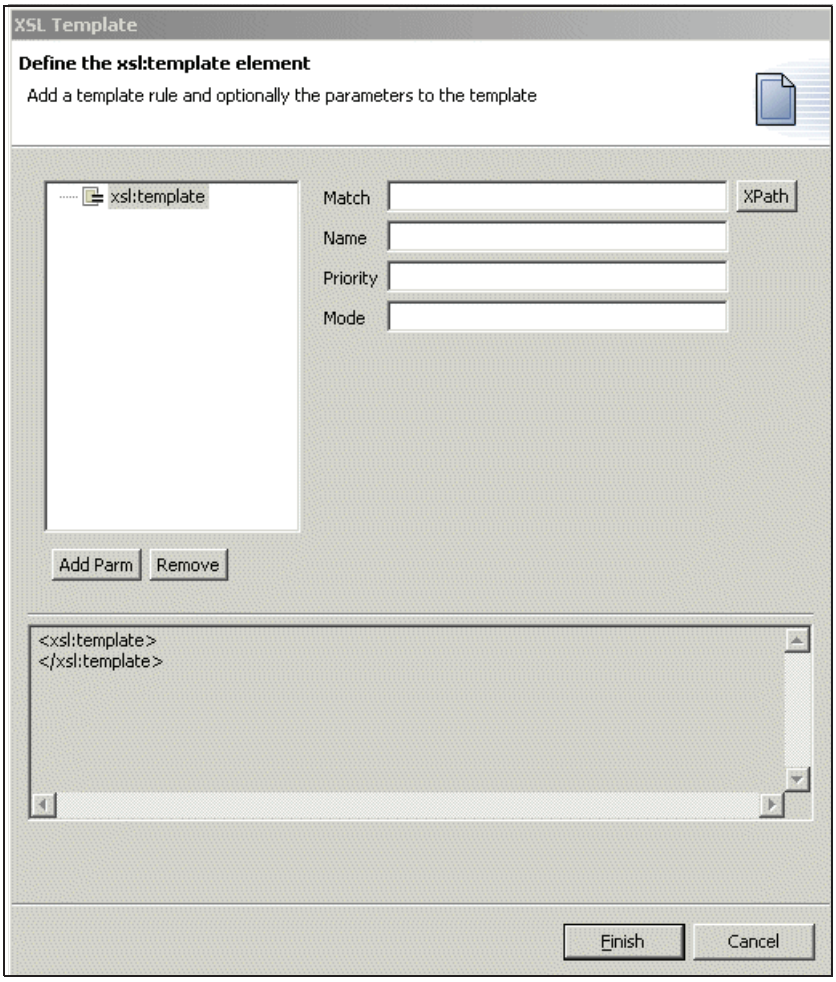


Figure 16-55 XSL Template wizard

The XSL Template wizard creates basic template rules. To identify the match path for the template, click **XPath**. The XPath wizard appears (shown in Figure 16-56 on page 489). We select the root element **AddressBook** and click **Next**.

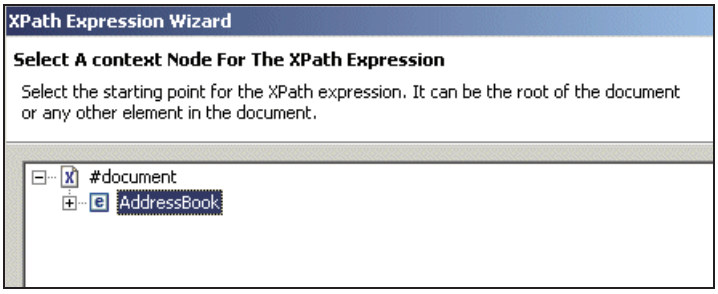


Figure 16-56 Select XPath

In the left pane (shown in Figure 16-57) containing the XML tree, select **AddressDetail** and the AddressDetail appears in the XPath field. Next expand the AddressDetail and select **MailAddress**, the MailAddress is appended to the AddressDetail/ in the XPath field, also shown in Figure 16-57.

To complete using the XPath wizard, click **Finish**. This returns to the XSL Template wizard with the match value modified to the selected XPath. Click **Finish** to complete this process.

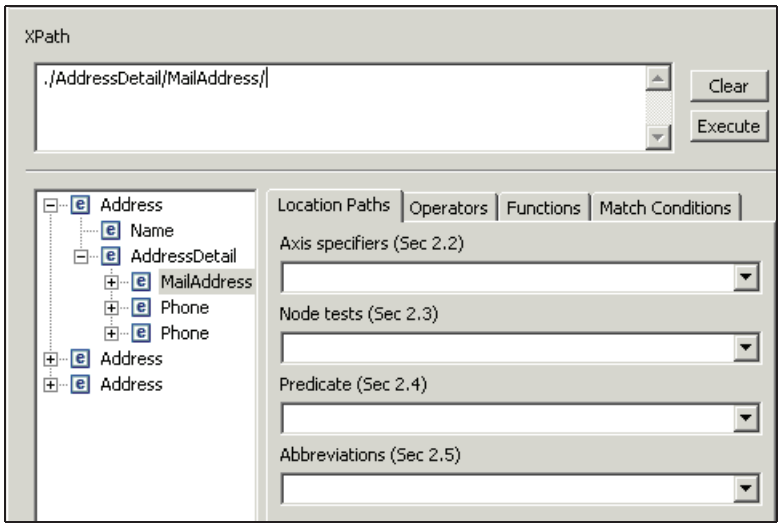


Figure 16-57 Select MailAddress

A new template rule is added to the XSL displayed in the editor. We filled in the body of the template rule with statements that get the contents of these elements (Street, City, State and ZipCode) and place them in the output stream. The modified template rule is shown in Figure 16-58 on page 490.

Example 16-9 Template body

```
<xsl:value-of select="Street"/> <br/>
<xsl:value-of select="City"/>,
<xsl:value-of select="State"/> <br/>
<xsl:value-of select="ZipCode"/> <br/>
```

```
<xsl:template match="./AddressDetail/MailAddress">
    <xsl:value-of select="Street"/> <br/>
    <xsl:value-of select="City"/>  ,
    <xsl:value-of select="State"/> <br/>
    <xsl:value-of select="ZipCode"/>
</xsl:template>
```

Figure 16-58 Edited MailAddress template rule

We created another template rule for the Phone element using the same technique just described for the AddressDetail/MailAddress template rule. This rule is shown in Example 16-10. At runtime when a Phone element is found, the Phone element's contents will be put in the output stream. Also the Phone element's attribute (TYPE) value enclosed in parentheses and followed by a break HTML tag.

Example 16-10 Template rule to match on Phone

```
<xsl:template match="./AddressDetail/Phone">
<xsl:value-of select="."/> (
<xsl:value-of select="@TYPE"/>) <br/>
</xsl:template>
```

Updates to the table template

Now that we have created all the rules needed to access the XML file, we need to change the table template rule discussed in "Create Table Template Rule" on page 482. We made the following changes to the template rule, as follows:

- Modify the match from AddressBook to AddressBook/Address.
- Remove the <xsl:for each start and end tags. This is now controlled by the match.
- Modify the <xsl:value-of statement to an <xsl:apply-template statement, which invokes the MailAddress rule.
- Add a new <xsl:apply-template statement to invoke the Phone rule.

The results of the modifications are shown in Example 16-11.

Example 16-11 Template rule for building table body

```
<xsl:template match="AddressDetail/Address">
```

```
<tr bgcolor=”#ff80ff” align=”left” valign=”top”>
<td> <xsl:value-of select="Name"/> </td>
<td> <xsl:apply-template select="AddressDetail/MailAddress"/> </td>
<td> <xsl:apply-template select="AddressDetail/Phone"/> </td>
</xsl:template>
```

The template rule shown in Example 16-11 on page 490 invokes two other template rules by name. In order to find these rules, the select value of the <apply-template statement must be the same as the match value of the target rule. However, when this template rule was created by the XSL Template wizard, the wizard created these match values: `match=”./AddressBook/MailAddress”` and `match=”./AddressBook/Phone”`. If these rules are left as they are, the apply-template statements are not executed properly. To fix this, we removed the **./** from the match statements, so now the match statements of these template rules are `match=”AddressBook/MailAddress”` and `match=”AddressBook/Phone”` respectively.

The XSL file is now complete with all the rules necessary to convert the XML file to the desired HTML file. The complete XSL file is shown in Example 16-12.

Example 16-12 Address2HTML.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0"
    xmlns:xalan="http://xml.apache.org/xslt">
    <xsl:output method="html" encoding="UTF-8"/>

<xsl:template match="/">
  <html>
    <head>
      <title>My Address Book</title>
    </head>
    <body>
      <H3>My Address Book</H3>
      <table bgcolor="#80ffff">
       <tr bgcolor="#ff80ff" align="left" valign="top">
        <th>Name</th>
        <th>Mail Address</th>
        <th>Phones</th>
        </tr>
         <xsl:apply-templates/>
       </table>
    </body>
  </html>
</xsl:template>

<xsl:template match="AddressBook/Address">
```

```
      <tr bgcolor="#ff80ff" align="left" valign="top">
        <td><xsl:value-of select="Name"/></td>
        <td><xsl:apply-templates select="AddressDetail/MailAddress"/> </td>
      <td> <xsl:apply-templates select="AddressDetail/Phone"/> </td>
      </tr>
</xsl:template>
<xsl:template match="AddressDetail/MailAddress">
    <xsl:value-of select="Street"/> <br/>
    <xsl:value-of select="City"/>  ,
    <xsl:value-of select="State"/> <br/>
    <xsl:value-of select="ZipCode"/>
</xsl:template>

<xsl:template match="AddressDetail/Phone">
   <xsl:value-of select="."/> (
    <xsl:value-of select="@TYPE"/>)  <br/>
</xsl:template>

</xsl:stylesheet>
```

Validating the XSL file

You can validate the XSL file as follows:

- Click **XSL** from the main menu list and select **Validate**.
- Or, select the **Validate the current state of the XSL file** icon from the toolbar.

Once the validation is completed successfully, you will receive a validation message. Click **OK** to close the message window. If the file is invalid, any errors will appear in the Tasks view.

Transform or debug the XSL file

The XSL editor can be used to quickly transform or debug an XSL file using the Debug and Transform toolbar buttons. To debug an XSL file, perform the following steps:

1. Open the XSL file you want to work with in the XSL editor.
2. Either select **Debug** from the menu bar and select **Debug** or click the **Debug icon** on the toolbar.
3. When we created the XSL file, we associated it with the XML file (MyAddressBook.xml). If we had not done this, we would have been prompted for the XML file. Once this relationship is established, this prompt does not reappear.

4. The XSL file and the XML file open in the XSL Debug perspective, as shown in Figure 16-59 on page 494. The Window opens with four panes:
   - The Session (upper left) pane provides these options in the form of icons across the top of the pane (left to right):
     - Step forward through the result document
     - Step backward through the result document
     - Restart from the beginning
     - Run to breakpoint
     - Open the browser on the transform results
   - The Activity (upper right) pane provides tabs at the bottom of the window that provide these activities:
     - Current XSL Element - shows the XSL element being processed
     - XSL Breakpoints - shows the breakpoints
     - Template Call Stack - shows the templates being processed
     - Task - shows outstanding tasks
   - The Outline (lower left) pane shows the outline of the file active in the file view pane.
   - The File (lower right) pane shows the XSL file and the XML file side by side. In this case we have expanded the XSL file.

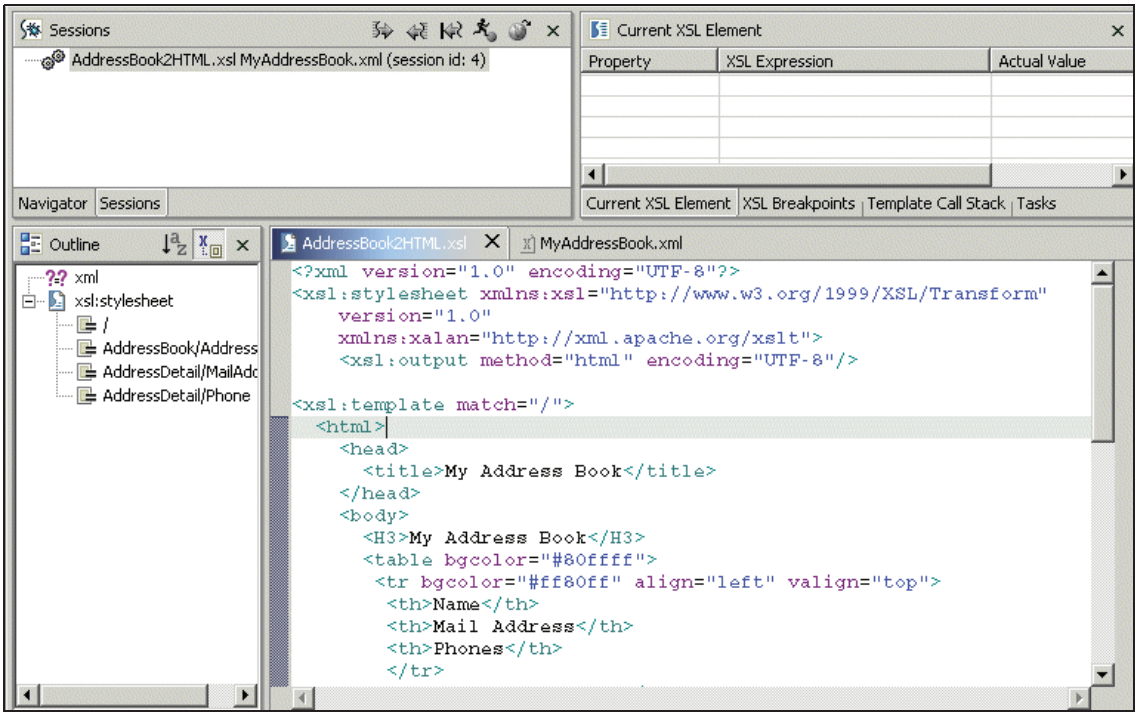


Figure 16-59 XSL Debug perspective

5. Use the Session view to step through the XSL and XML files. Click the **Step forward through the result document** icon. The Activity view shows the current template rule ready for processing, as shown in Figure 16-60.

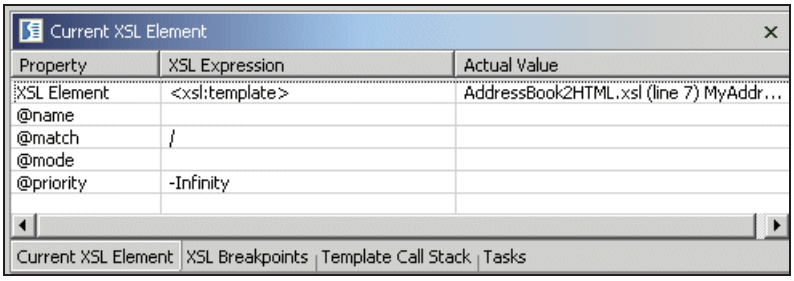


Figure 16-60 Activity view

6. You can continue to step through the files or select the **Run to breakpoint** icon in the Session view, which will complete the processing because we do not have breakpoints set.

7. To view the transformation results, select **Open the browser on the transform results** in the Session view.

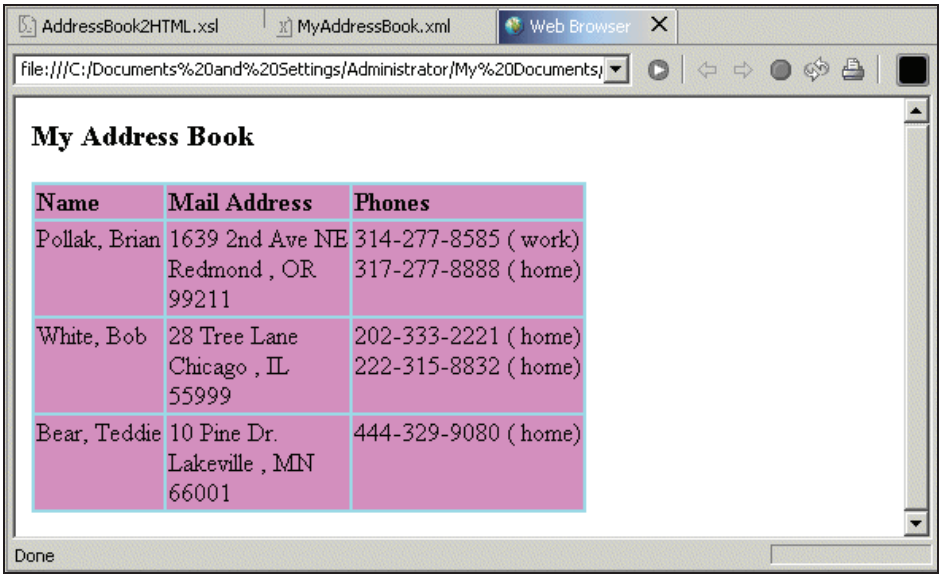


Figure 16-61 Transform results in browser

You can also use the XSL Debugger to transform the XSL file, as follows:

1. Open the XSL file in the XSL editor
2. Either select **Debug -> Transform** from the menu bar or select the **Transform** icon from the toolbar.
3. The transformation process takes place and the results are displayed in the browser the same as shown in Figure 16-61.

You can also start the debug tool by performing the following steps:

1. Right-click the **XML** file (MyAddressBook.xml).
2. From the menu select **Apply XSL -> As HTML**.
3. The XSL Selection wizard appears (as shown in Figure 16-62 on page 496). Select the **Workbench Project** radio button.

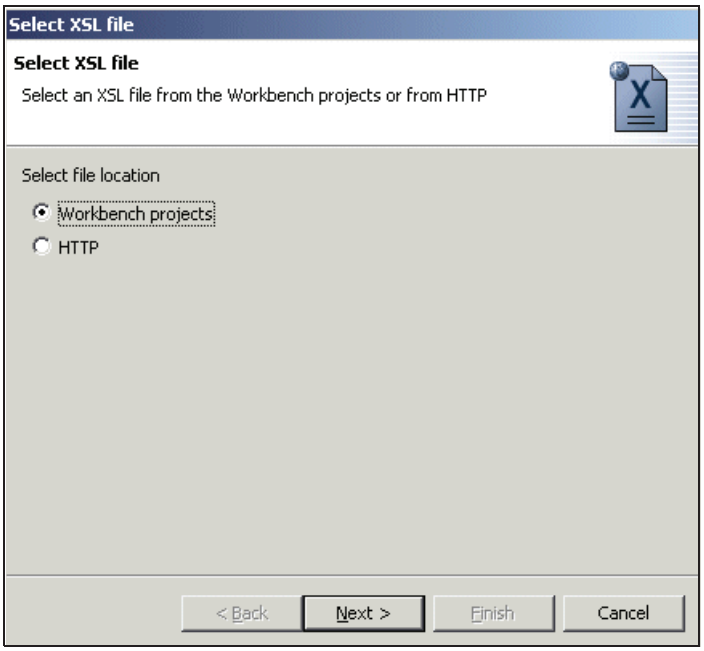


Figure 16-62 Validate XSL file

4. In the Select XSL file window, select the XSL file in the AXMLProject folder named **AddressBook2HTML.xsl**, as shown in Figure 16-63 on page 497. Click **Next**.

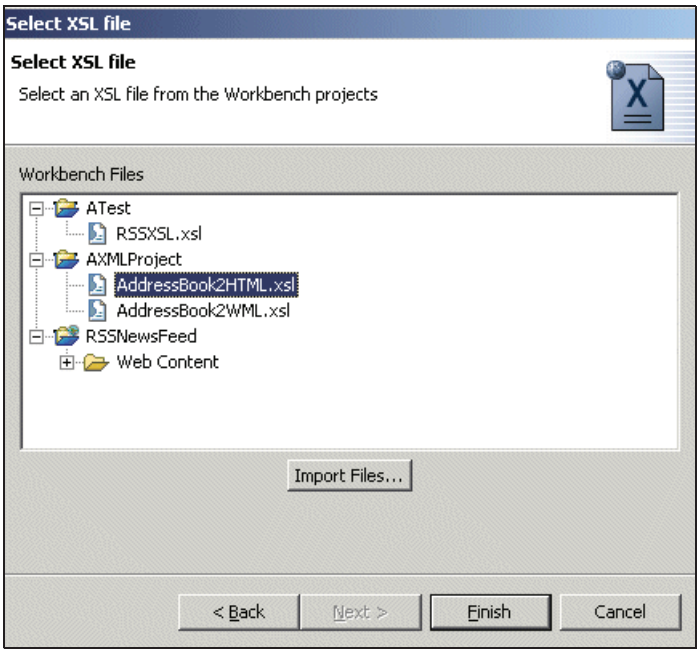


Figure 16-63 Validate XML - Select XSL directory

5. Click **Finish**.

Now you are in the XSL Debug tool. Use the XSL Debug tool as discussed previously.

16.5.4 XML to XML mapping editor

The XML to XML mapping editor is used to map one or more source XML files to a single target XML file. You can include XPath expressions, groupings, Java methods or conversion functions to your mapping. Mappings can also be edited, deleted or persisted for later use. After defining the mappings, you can generate XSLT script, which can be used to combine and transform any XML files that conform to the source DTD.

We created a simple XML file that contains a simple WML document consisting of a card with a table. We named the XML file SimpleWMLTable.xml. Creating this simple WML file allows us to show you how to use XML-to-XML mapping. The structure of the file is shown in Example 16-13.

Example 16-13 WML file

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

```
<wml>
   <card id="card1" title="Addresses">
     <p>
   <table columns="2" align="LCC">
     <tr>
      <td>Name</td>
      <td>Address</td>
     </tr>
   </table>
     </p>
   </card>
</wml>
```

Create XML mapping

To start the XML to XML mapping editor, perform the following steps:

1. Start select **File -> Other**.
2. In the New Wizard window, select **XML** in the left pane and **XML to XML Mapping** in the right pane.
3. On the New XML to XML Mapping window, select the folder **AXMLProject** and enter the file name `AddressBook.xmx`. Click **Next**.
4. In the Source XML, DTD or XSD files window, expand the AXMLProject, select **AddressBook.dtd**, click the **>** (move) button. The AddressBook.dtd is added to the Selected Files list, as shown in Figure 16-64 on page 499. Click **Next**.

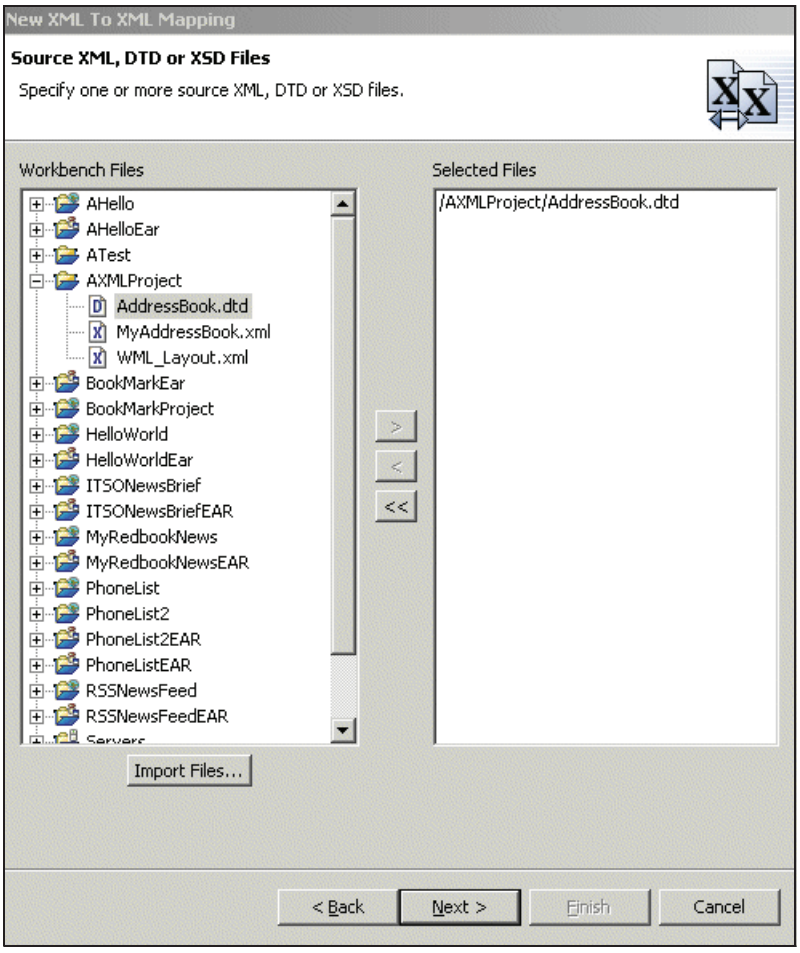


Figure 16-64 Select DTD (source)

5. To select the target file, select **SimpleWMLTable.xml**.
6. Next, select a Root element for the target and source document. The target root is WML and the source root is AddressBook. This is shown in Figure 16-65 on page 500. Click **Finish** to complete the process.

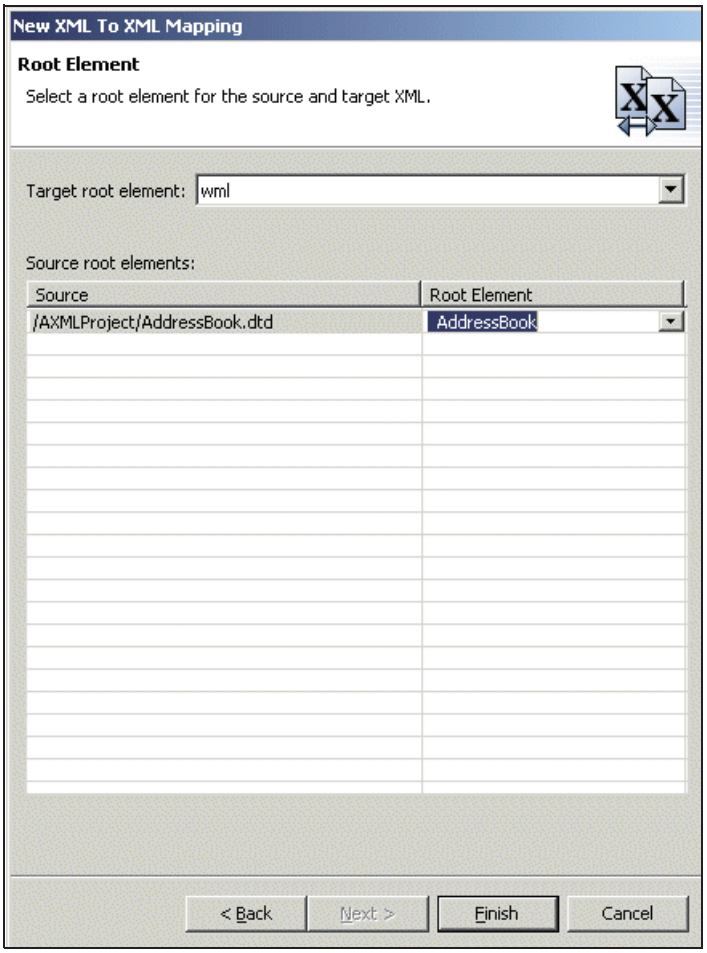


Figure 16-65 Select roots

Using the XML to XML mapping editor

The mapping editor provides two new panes within the WebSphere Studio display, which show the Source and Target DTDs (in this particular case). The mapping editor is shown in Figure 16-66 on page 501.

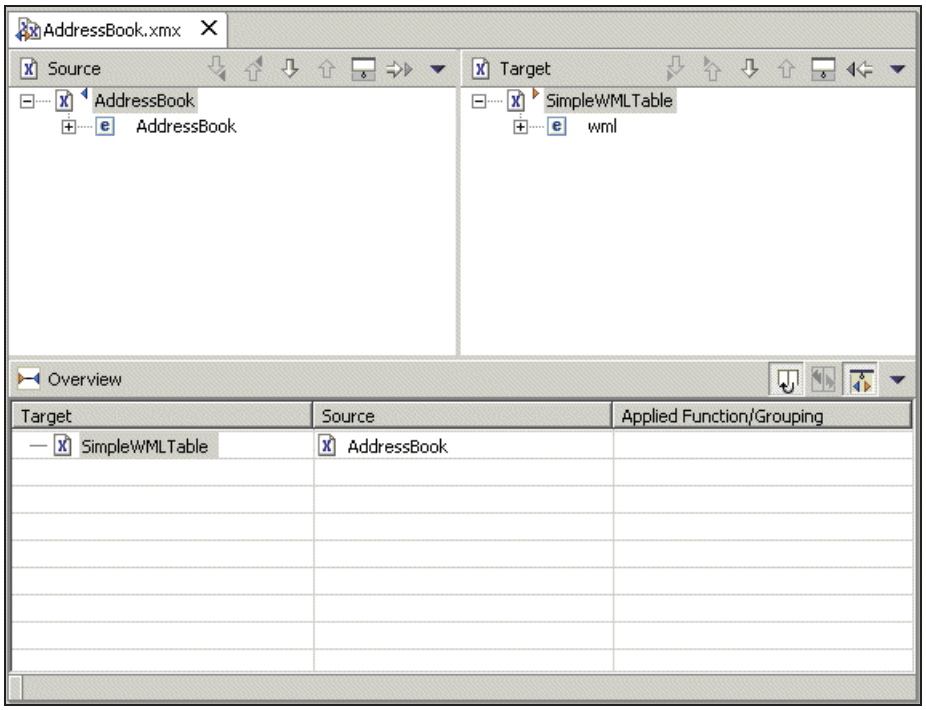


Figure 16-66 XML to XML mapping editor

To work with the elements within the target and source, we expanded the complete target and source structures. To do this click the + (plus sign) by each element. The expanded source and target are shown in Figure 16-67.

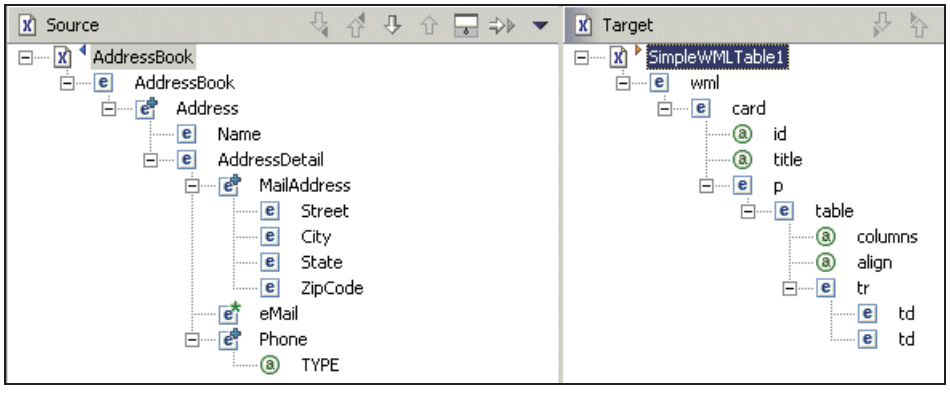


Figure 16-67 Expanded files

To create the mappings, select an element from the Source (Name) and an element from the target (the first td). Right-click and in the pop-up menu select **Create Mapping**. This is shown in Figure 16-68.

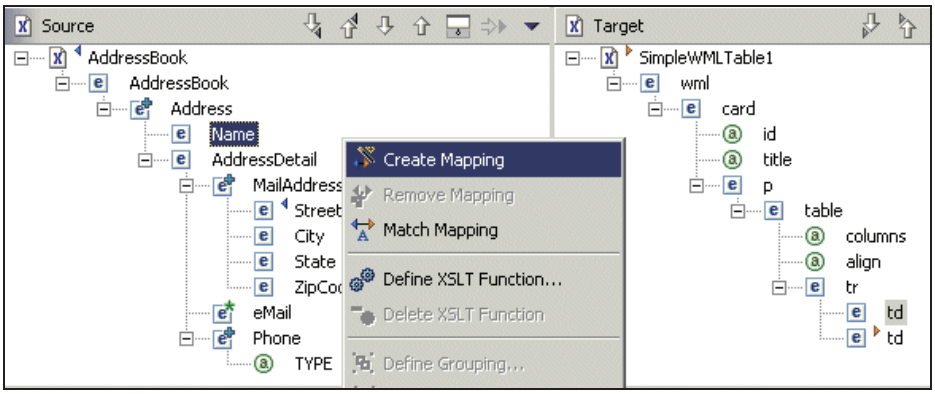


Figure 16-68 Create a mapping

An entry is created in the lower pane of the mapping editor, as shown in Figure 16-69.

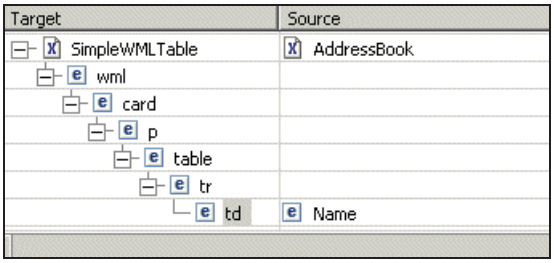


Figure 16-69 Mapping an entry

The complete set of mappings we created are shown in Figure 16-70 on page 503. Even though we wanted to map all MailAddress children to the second table detail tag, the tool would not let us. We will have to add that to the XSL file later.

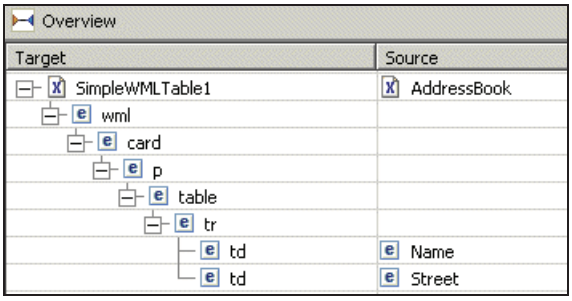


Figure 16-70 All mappings

Next we saved our modified AddressBook.xmx file. The first mapping in the AddressBook.xmx file is shown in Example 16-14.

Example 16-14 AddressBook.xmx

```
<?xml version="1.0" encoding="UTF-8"?>
<Mapping:MappingRoot xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:Mapping="Mapping.xmi" xmlns:XML="XML.xmi" xmi:id="MappingRoot_1"
outputReadOnly="true" topToBottom="true">
<!-- *** the Mapping Rule -->
  <nested xmi:id="Mapping_1">
    <inputs xmi:type="XML:XMLElement"
href="platform:/resource/AddressBook.dtd.AddressBook.dtdxml#1:AddressBook.1:Add
ress:1/1.1:Name:1/1"/>
    <outputs xmi:type="XML:XMLElement"
href="platform:/resource/SimpleWMLTable.xml.wml.dtdxml#1:wml.1:card.1:p.1:table
.1:tr.1:td"/>
  </nested>
```

Generate an XSLT

Once the mapping process is complete and the additions and modifications have been saved in the AddressBook.xmx file, we are ready to generate an XSL file. To generate the XSL file, perform the following steps:

1. In the project select the **AddressBook.xmx** file, right-click and select **Open With -> XML to XML Mapping Editor**.
2. Select **Mapping** from the menu bar and select **Generate XSLT script**.



Figure 16-71 Generate XSLT

3. On the Generate XSLT Script window, select **AXMLProject** for the folder and enter `AddressBookXML.xsl` as the file name. Click **Finish**.
4. The XSL file is now displayed in the XSL editor. The XSL file is shown in Example 16-15.

Example 16-15 XSL file

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
    xmlns:xalan="http://xml.apache.org/xslt">
<xsl:output method="xml" encoding="UTF-8" indent="yes" alan:indent-amount="2"/>
<xsl:strip-space elements="*"/>
  <!--============================================================================-->
  <!-- We REMOVED THE GENERATED COMMENT -->
  <!--============================================================================-->
  <xsl:template match="/">
    <wml>
      <card>
        <xsl:attribute name="id">
          <xsl:value-of select="'card1'"/>
        </xsl:attribute>
        <xsl:attribute name="title">
          <xsl:value-of select="'Addresses'"/>
        </xsl:attribute>
<p>
```

```
          <table>
            <xsl:attribute name="columns">
              <xsl:value-of select="'2'"/>
            </xsl:attribute>
            <xsl:attribute name="align">
              <xsl:value-of select="'LCC'"/>
            </xsl:attribute>
            <tr>
              <td>
                <xsl:value-of select="/AddressBook/Address/Name"/>
              </td>
              <td>
                <xsl:value-of
select="/AddressBook/Address/AddressDetail/MailAddress/Street"/>
              </td>
</tr>
          </table>
        </p>
      </card>
    </wml>
  </xsl:template>
</xsl:stylesheet>
```

The new XSL file gives us a very good base to start with, but we need to further customize the XSL file to fit our needs. Part of the work we must do is because we used an XML file as the source. But if we had used the DTD instead, we could not have mapped the table detail entries to the source. In the XSL editor, we made the following changes:

- Change the <card> for the output stream to <card id=”card1” title=”Addresses”> and remove the <xsl:attribute statements relating to the card.
- Remove the <xsl:attribute statements relating to the table and change the table tag to <table columns=”2” >

Once these changes are made, save the XSL file. Now, we are ready to test the new XSL file. With the XSL file in the editor, click the **Transform** icon. The results of the transformation are displayed in the browser and shown in Figure 16-72 on page 506.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wml>
  - <card id="card1" title="Addresses">
    - <p>
      - <table columns="2">
        - <tr>
            <td>Pollak, Brian</td>
            <td>1639 2nd Ave NE</td>
          </tr>
        </table>
      </p>
    </card>
  </wml>
```

Figure 16-72 XML-to-WML transform results

There are three problems with the generated results:

1. The DOCTYPE for the WML.dtd is missing
2. All the Address entries within the XML file must be processed, not just the first one, and the other MailAddress children must be added to the second table d
3. A heading is needed for the table

To fix these problems, we added the code in EXAMPLE to put the DOCTYPE statement into the output stream. This code is added before the <WML> tag.

Example 16-16 Adding DOCTYPE

```
<xsl:text disable-output-escaping="yes">
  &lt;!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http:///www.wapforum.org/DTD/wml_1.1.xml"&gt;
</xsl:text>
```

4. To process all the addresses in the XML file, we needed a new template rule, which would include the table row instructions from the existing template rule. We also needed to modify the <xsl:value-of select statements. With the table row instructions removed, we must add a <xsl:apply-templates/> statement to invoke other possible rules (which includes the new rule we just created). The new template rule is shown in Example 16-17.

Example 16-17 Process all addresses

```
<xsl:template match="/AddressBook/Address">
   <tr>
    <td> <xsl:value-of select="Name"/> </td>
    <td> <xsl:value-of select="AddressDetail/MailAddress/Street"/> <br/>
<xsl:value-of select="AddressDetail/MailAddress/City"/> ,
```

```
<xsl:value-of select="AddressDetail/MailAddress/State"/> <br/>
<xsl:value-of select="AddressDetail/MailAddress/ZipCode"/> </td>
   </tr>
</xsl:template>
```

5. Add a new <tr> statement with the heading information shown in Example 16-18. This code goes after the existing <table> tag.

Example 16-18 Heading

```
<tr>
<td>Name</td>
<td>Mailing Address</td>
</tr>
```

With these changes in place and the file saved. We executed the transform again. The transform failed with this error

```
Unable to connect to the target server. Error processing source
‘http://www.wapforum.org/DTD/wml_1.1.xml’.
```

In essence, the browser could not process the DOCTYPE statement the stylesheet put into the output stream. So we commented that statement out and ran the transform again. This time the transform ran fine and generated the WML output, which is shown in Example 16-19.

Example 16-19 Resulting WML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http:///www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="card1" title="Addresses">
    <p>
      <table columns="2">
          <tr>
            <td>Name</td>
            <td>Mailing Address</td>
          </tr>
          <tr>
            <td>Pollak, Brian </td>
            <td>1639 2nd Ave NE <br/>Redmond,OR<br/> 99211</td>
          </tr>
          <tr>
            <td>White, Bob</td>
            <td>28 Tree Lane<br/> Chicago,IL<br/> 55999</td>
          </tr>
          <tr>
            <td>Bear, Teddie</td>
```

```
            <td>10 Pine Dr<br/> Lakeville ,MN<br/> 66001</td>
          </tr>
        </table>
      </p>
    </card>
</wml>
```

We decided it would be a good test to view the WML in the Nokia emulator. To do this, we saved the output from the browser session and exported it to a file. We changed the file type to .wml and added the DOCTYPE statement. We moved the file into the /Nokia/MobileInternet/samples/WML directory. We selected the file from the Nokia Mobile Internet Toolkit by selecting **Browser -> Load File** and selecting the file from the directory. The results are shown in Figure 16-73.



Figure 16-73 Results on Nokia emulator

16.5.5 Summary

In this chapter, we started out by showing how to use a portlet application (through the JSP) to read an XML file and apply one or more stylesheets to the XML file. This allows us to transform the XML to the desired markup for the target device. Then we looked at the various tools available in WebSphere Studio to create XML and related file types such as DTD and XSL files. The XML-related tools help create the XML files and the XSL files to transform the XML to the target markup. The Everyplace Toolkit provides the portlet development tools

necessary to create the associated Java code to process the XML document and associate the XSL file(s) used to transform the XML to the target market. The combination of the WebSphere Studio tools and the Everyplace Toolkit provide all the tools necessary to build Portal applications that work with XML documents.

17

Portal-level transcoding

The main function of Portal-level transcoding implemented in IBM WebSphere Everyplace Access is to provide services to transform portlet content into a series of dynamically linked information (called *decks*), suitable for handling by client devices using markup languages such as WML for mobile devices and compact HTML (cHTML) for i-mode devices.

In this chapter, we describe the fragmentation function provided by Transcoding Technology and how this support is integrated in a WebSphere Everyplace Access environment for proper portlet access.

Sample scenarios using WML client devices are included to show how to configure a WAP Gateway and its associated WAP proxy. In these scenarios, the Edge Server or Web Traffic Express (WTE) proxy is used as a WAP proxy when deploying portlet applications using IBM WebSphere Everyplace Access to generate HTML content that is transcoded to the markup for the target device.

© Copyright IBM Corp. 2003. All rights reserved.

17.1 Overview

Many phones have limited storage capacity (for example, 2880 bytes). However, many Web pages exceed these limits. Therefore, converting an HTML page to i-mode (cHTML) or WML is very likely to result in a deck and/or page that exceeds the maximum storage capacity of a phone. The same problem may be encountered with native wireless content if the content generator is unaware of the specific limits of the phone being used. Exceeding the storage capacity of the cell phone means that the page cannot be viewed.

The fragmentation function provided by Transcoding Technology makes it possible to view these oversized pages on the limited storage cell phones. Fragmentation solves this problem by splitting a single oversized deck and/or page into multiple smaller decks/pages, each one smaller than the maximum size limitation.

Note: The main function provided by Portal-level transcoding is deck fragmentation.

Figure 17-1 illustrates Portal-level transcoding as implemented in WebSphere Everyplace Access.

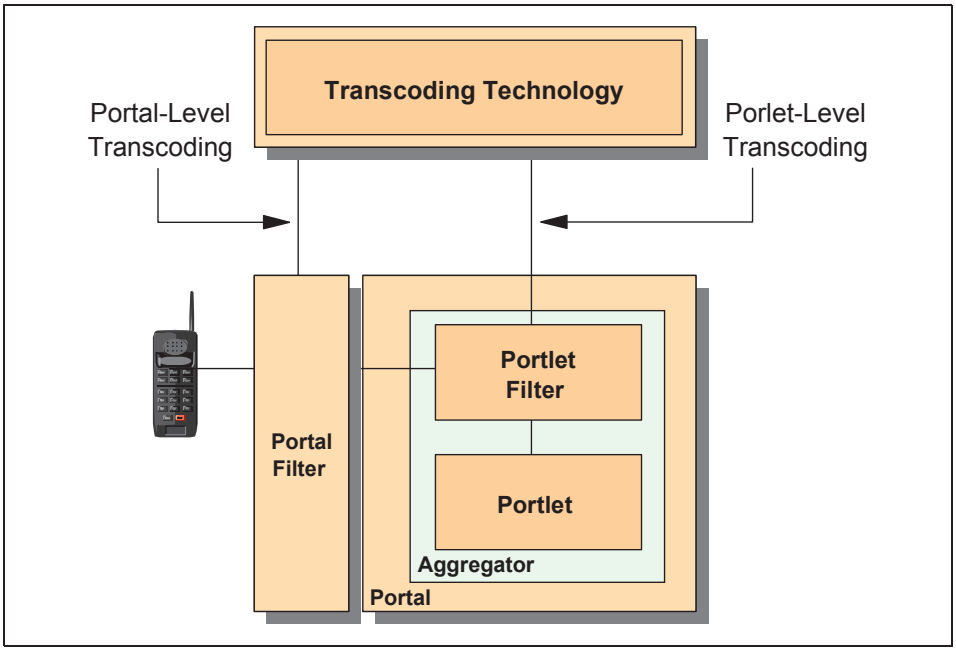


Figure 17-1 Portal-level transcoding in WebSphere Everyplace Access

Fragmentation is performed by the fragmentation transcoder. This transcoder is registered and enabled by default.

Note: Transcoding Technology in IBM WebSphere Everyplace Access supports fragmentation for WML and i-mode (cHTML).

17.1.1 How it works

The fragmentation transcoder traverses the Document Object Model (DOM) tree representing the oversized card/page. The DOM is generated internally and automatically when HTML is transcoded to WML or i-mode (cHTML). Therefore, you do *not* need to enable the HTML Document Object Model Generator transcoder.

If elements will fit on the new page, they are added to it and removed from the original. If elements do not fit, there are two ways to handle them:

- They can be fragmented: Recursively traverse the element's subtree.
- They cannot be fragmented: Finish fragmenting the new page; the original may still be too large.

The maximum size for a fragment is a parameter in a particular mobile device property file. The property file we are using for the WML device is located at WebSphere\PortalServer\IBMTrans\etc\preferences\device\WML-Device.prop.

The fragmentor determines the size value from the information, as follows:

- For WML there is a parameter that specifies the maximum size value in the device preference profile. The key parameter (WMLMaximumDeckSize) is a configurable property and is shown in Example 17-1 on page 514.

Example 17-1 WML device preference profile

```
# IBM WebSphere Transcoding Publisher WML Device Characteristics and
Preferences
deviceType=WML Device
desiredContentTypes=[text/vnd.wap.wml]
textLinksPreferredToImages=true
propagateFirstTableRowData=false
disposeImages=false
transcodeImages=true
convertTablesToUnorderedLists=false
supportedImages=[wbmp]
fixedImageScale=true
imageScaleFactor=0.5
colorSupported=false
screenCapability=low
WMLMaximumDeckSize=2880
deviceRule=(User_Agent=*WAP*) | (User_Agent=*Wap*) | (User_Agent=*MOT-CB*) |
(((User_Agent=*UP.Browser/3.1*) | (User_Agent=*UP.Browser/4.*) | (User_Agent=*
UP/4.*)) & (Accept=*wml*)) | (User_Agent=*Nokia*) & (!User_Agent=*Rover*)
ConfigurableProperties=transcodeImages{bool} disposeImages{bool}
fixedImageScale{bool} imageScaleFactor{itext} textLinksPreferredToImages{bool}
desiredContentTypes{text} propagateFirstTableRowData{bool}
convertTablesToUnorderedLists{bool} WMLMaximumDeckSize{itext}
NonConfigurableProperties= screenCapability{text} supportedImages{text}
colorSupported{bool}
```

- i-mode (cHTML): The maximum size value is 2048 bytes by default. There is no parameter as with WML. i-mode phones can also have different cache sizes. The size is specified in the User-Agent field in the HTTP header. For example, User-Agent DoCoMo/1.0/N502/c8 specifies an 8 KB cache. In this case, the fragmentor adjusts fragmentation size based on this cache size. If the cache size is c8 (8 KB), the fragmentor sets the maximum size to 3000 bytes. If the cache size is c10 (10 KB), the fragmentor sets the maximum size to 4000 bytes.

In addition to splitting up the deck/page into smaller chunks, the fragmentor adds links to each of the generated pieces to allow navigation from one piece to the next and the previous one. The Continue link allows you to move to the next fragment and the Return link moves you to the previous one. The first fragment has no Return link and the last no Continue link.

Figure 17-2 on page 515 shows an example of WML fragmentation. A single oversized WML deck is fragmented into two smaller pieces. The Continue and Return links are inserted into the fragments to allow for navigation between the fragments. Also, any intra-deck links in the original deck are fixed to point to the target in whatever deck/card in which they are placed.

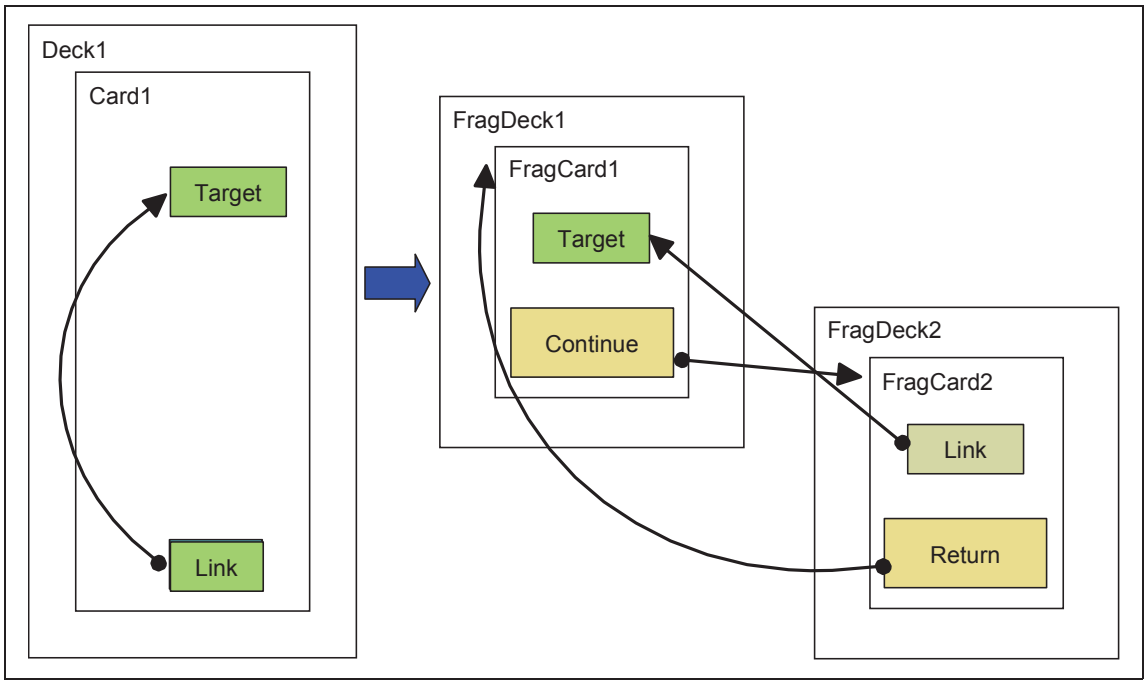


Figure 17-2 Fragmented WML deck

After fragmentation is performed, the first fragment is sent to the client as an HTTP response. The fragmentation engine stores non-first fragments in a general-purpose resource repository. Making the resource repository general purpose will allow for reuse by other components needing a similar service in the future. But for now, only the fragmentation engine uses the resource repository. Because the fragmentor needs the resource repository to save fragments for later retrieval, the resource repository should *not* be disabled if fragmentation is being used. If the fragmentor is disabled, the resource repository should also be disabled; this will improve performance.

All fragments are named so that a request for any fragment will be routed back to the transcoder. This means including the Web server host name and required fields so that the transcoder is invoked to handle the fragment request.

For example:

```
http://mka6brra.itso.ral.ibm.com/wps/TranscodingUtilities/ifrag-4739I34/mka6brra...
```

Where `mka6brra.itso.ral.ibm.com` is the Portal host name.

Note: Fragments are kept until the original document expires. If a request for a discarded fragment is received, a `Fragment expired` message is sent.

17.1.2 Fragmentable elements

Fragmentable elements are:

- Nodes (tags) with children
- Nodes (tags) that can be safely cloned with child nodes distributed among the clones (*safely* means that the resulting markup is valid and the content's meaning and/or presentation is essentially unchanged)

Note that elements with no children (for example, break elements) are not listed below as fragmentable, but a card/page may be split at one of these elements.

- WML fragmentable elements are:

  <wml>, <card>, <p>, , , <i>, <b>, <u>, <big>, <small>, <table>

- i-mode fragmentable elements are:

  <html>, <body>, <p>, <blockquote>, <blink>, , <dir>, , <font>, <plaintext>, <pre>,

17.1.3 Common problems

Invalid input (invalid content) will cause a request to be rejected (FragmentRejectedException). Sometimes, content cannot be fragmented into small enough pieces. This is likely to happen with the following elements:

- Fragmentable elements provided as a reference
- Long paragraphs with no breaks
- Large forms

17.1.4 Example

Figure 17-3 on page 517 is a simple example of WML deck fragmentation. The fragmentor traverses the tree depth-first. At each node (tag), the fragmentor calculates the size of the page represented by the nodes visited so far, plus any descendants of the current node. If this amount exceeds the maximum size, then either the tree is fragmented before the current node, or the subtrees of the current node are recursively considered for fragmentation.

The double line in Figure 17-3 on page 517 indicates where the fragmentor determines that this tree must be fragmented.

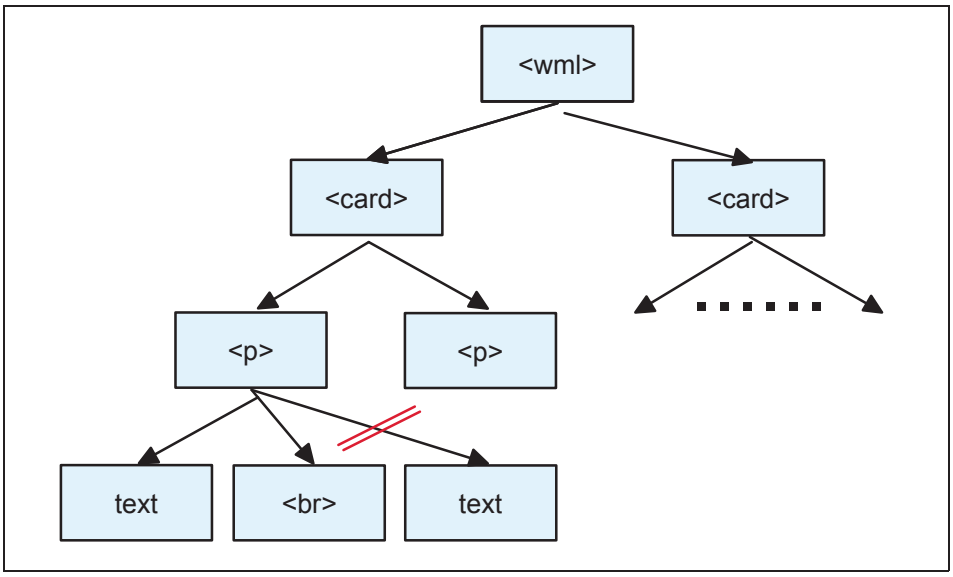


Figure 17-3 Fragmentation example

Figure 17-4 illustrates the first fragment resulting from fragmenting the WML deck, as indicated in Figure 17-3.

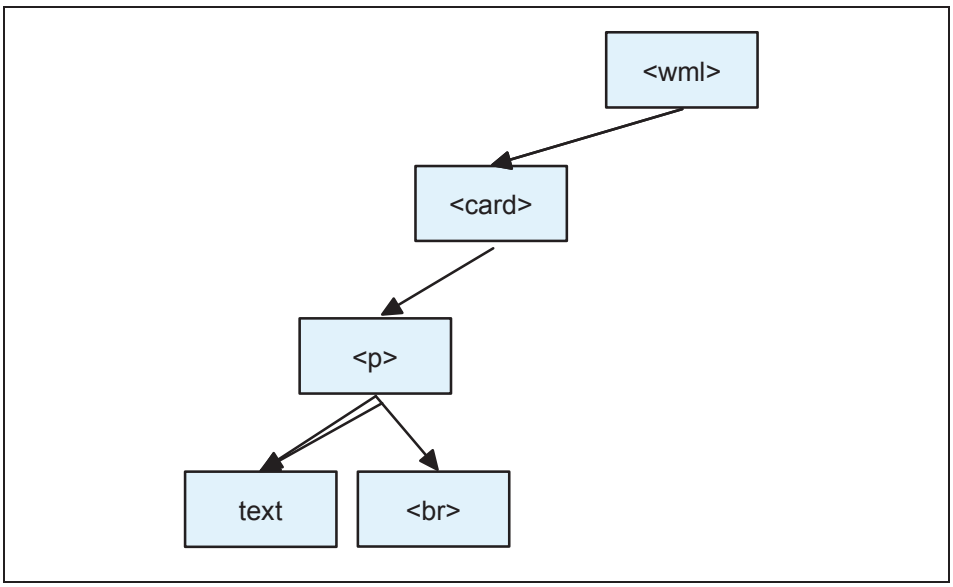


Figure 17-4 First fragment

Figure 17-5 is the second fragment. Notice that the <wml>, <card>, and <p> elements from the original card were duplicated, with children distributed or duplicated as necessary between the new and old elements. This remaining fragment may still be too large for the target device, so the fragmentor begins again at the top of this DOM to see if it needs to be fragmented.

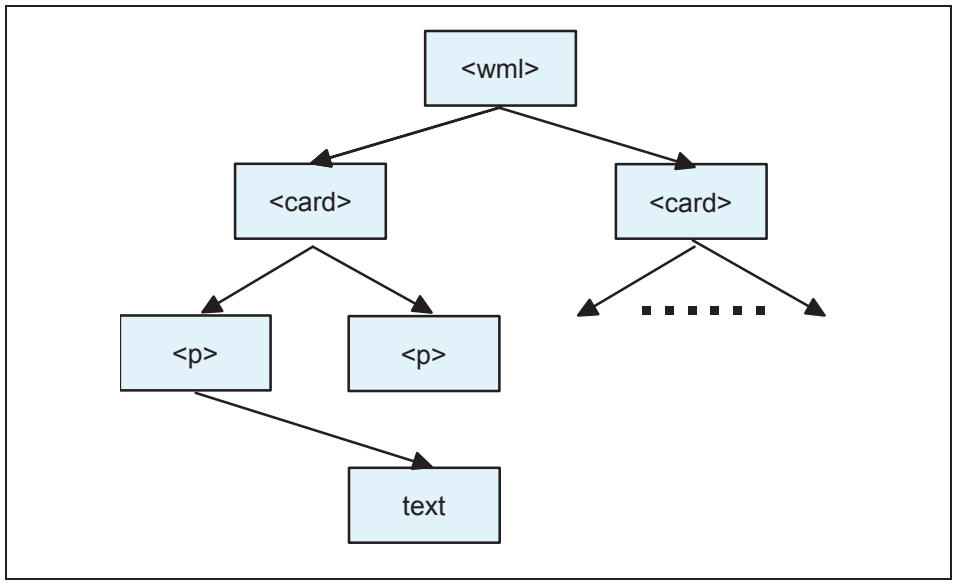


Figure 17-5 Next fragment

17.2 WML fragmentation

WML fragmentation algorithms in Transcoding Technology include URLs and, therefore, some special considerations are required when running in a WebSphere Everyplace Access environment.

In this section we include three sample scenarios to illustrate how you will configure a WAP Gateway and a WAP proxy using WML fragmentation in a WebSphere Everyplace Access environment.

Note: A reverse proxy must be properly configured when running portlet applications using WML fragmentation.

17.2.1 Scenario 1: Using a WAP reverse proxy

In this section we describe a sample scenario for WML fragmentation in a WebSphere Everyplace Access environment where a WAP proxy is configured as a reverse proxy. The WAP client device can be connected to this domain in the following ways:

- Using HTTP to the WAP proxy (reverse proxy) using a WAP simulator. This is commonly done for application development.
- Using a WAP/IP connection to WebSphere Everyplace Connection Manager.
- Using other WAP connections, such as using PPP protocol or LAN.

In this scenario we show you how to configure the WAP Gateway and the WAP reverse proxy. The sample scenario is illustrated in Figure 17-6.

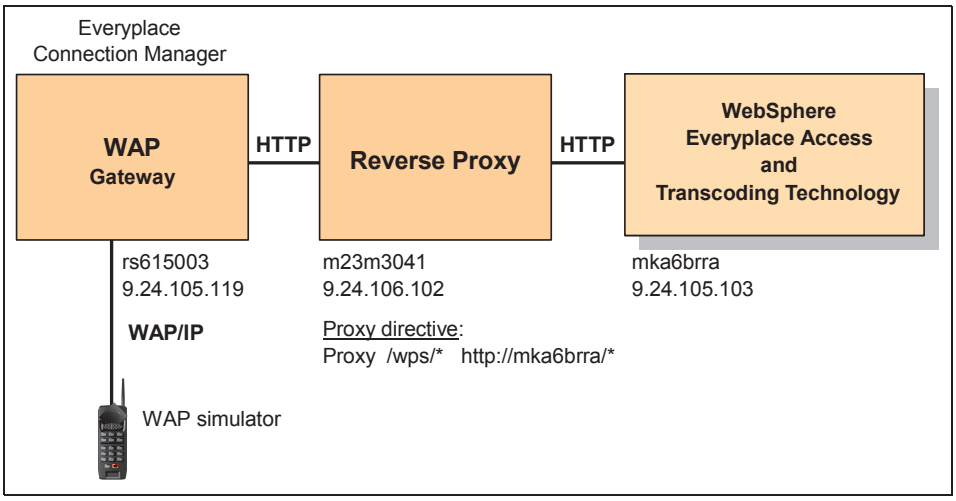


Figure 17-6 Sample scenario using a reverse proxy

For WAP connections, the Everyplace Connection Manager must be configured to connect to a reverse proxy. The WAP Gateway configuration is shown in Figure 17-7 on page 520. This option is not available during installation and you will need to use the Connection Manager Administrative Console (Gatekeeper).

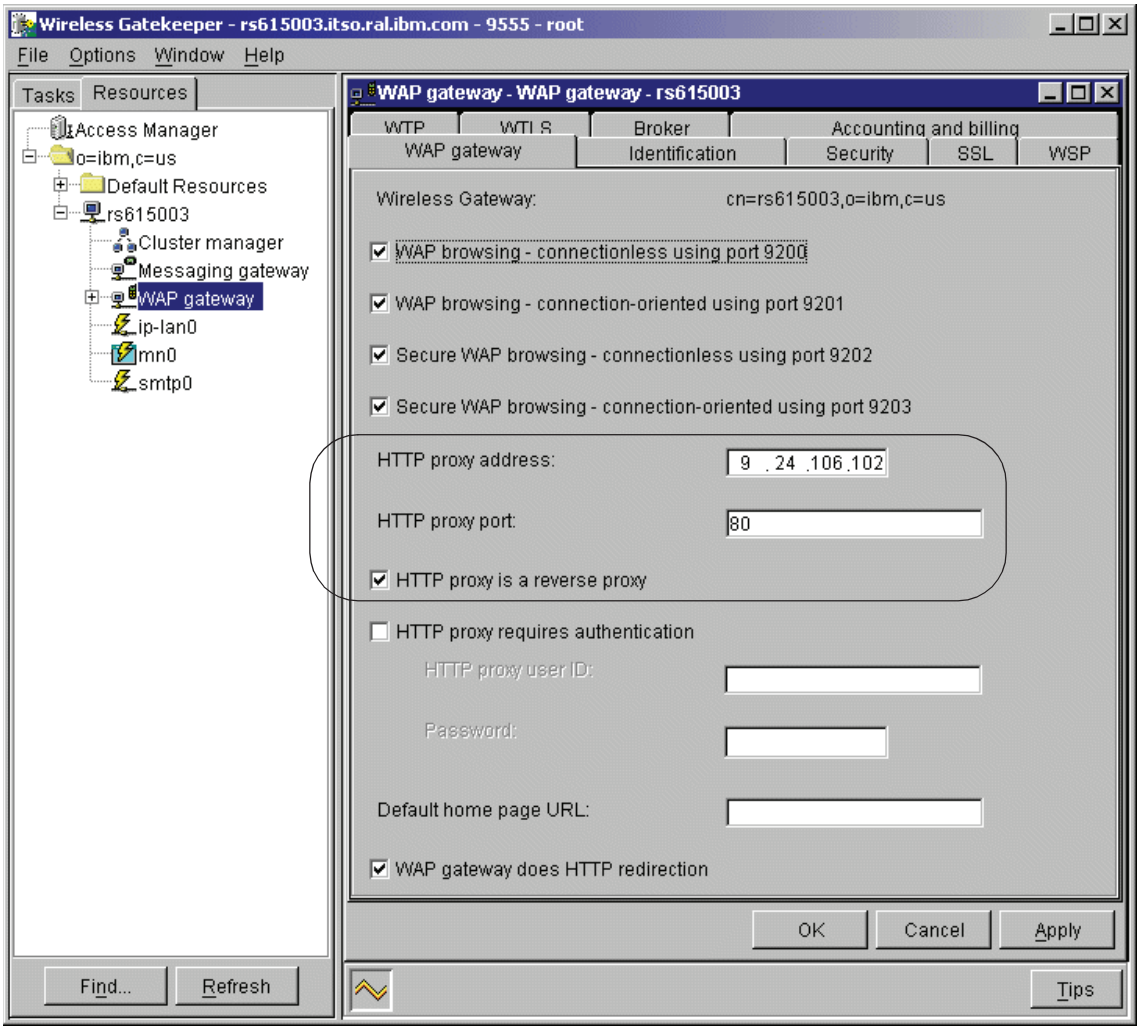


Figure 17-7 WAP Gateway configuration to connect to a WAP reverse proxy

When using the WAP Gateway connected to a reverse proxy, the values you configured in the WAP Gateway are used. This means that any computer name and port values can be used in the application since they will be replaced. For example:

```
<a href="http://xxxx:yy/wps/portal">WEA Portal</a>
```

Where xxxx is any host name or IP address and yy is the port number. These values will be replaced with the configured values (9.24.106.102:80 in this scenario).

Of course, this is not true if you are connected directly to the WAP reverse proxy using the HTTP protocol. In this case, the application values will be used.

The WML device in this scenario is not directly connected to WebSphere Everyplace Access. Therefore, the WAP reverse proxy must be configured to provide the connectivity to the WebSphere Everyplace Access machine.

For this scenario the WAP reverse proxy is assumed to be configured to listen on port 80, which is the default port. If using a different port, you will need to configure the listening port in the port directive in the ibmproxy.conf file. For example:

```
Port 80
```

In addition, you will need to specify the protocols that this proxy server will forward. For example, to forward all HTTP requests with the string wps in the link, you configure the proxy directive as follows:

```
Proxy   /wps/*         http://mka6brra/*
```

Where mka6brra is the host name (you can also use the IP address) of the WebSphere Everyplace Access machine and listening port 80.

In this sample scenario, the resulting link to access the Portal will be:

```
http://mka6brra.itso.ral.ibm.com/wps/portal
```

Therefore, it will reach the target WebSphere Everyplace Access machine.

The same is true for WML fragment links. For example, Figure 17-8 on page 522 shows a WML deck where you can see that, as expected, all references in the links point to the WAP reverse proxy port 80.

You can also see that links for WML fragments, when using Portal-level transcoding, have the following format:

```
http://<computer-name>/wps/TranscodingUtilities/ifrag-......
```

Because of the configured proxy directive in this sample scenario, the effective link to access the fragment will result in something similar to the following:

```
http://mka6brra.itso.ral.ibm.com/wps/TranscodingUtilities/ifrag-......
```

Therefore, Transcoding Technology will effectively retrieve the proper fragment.

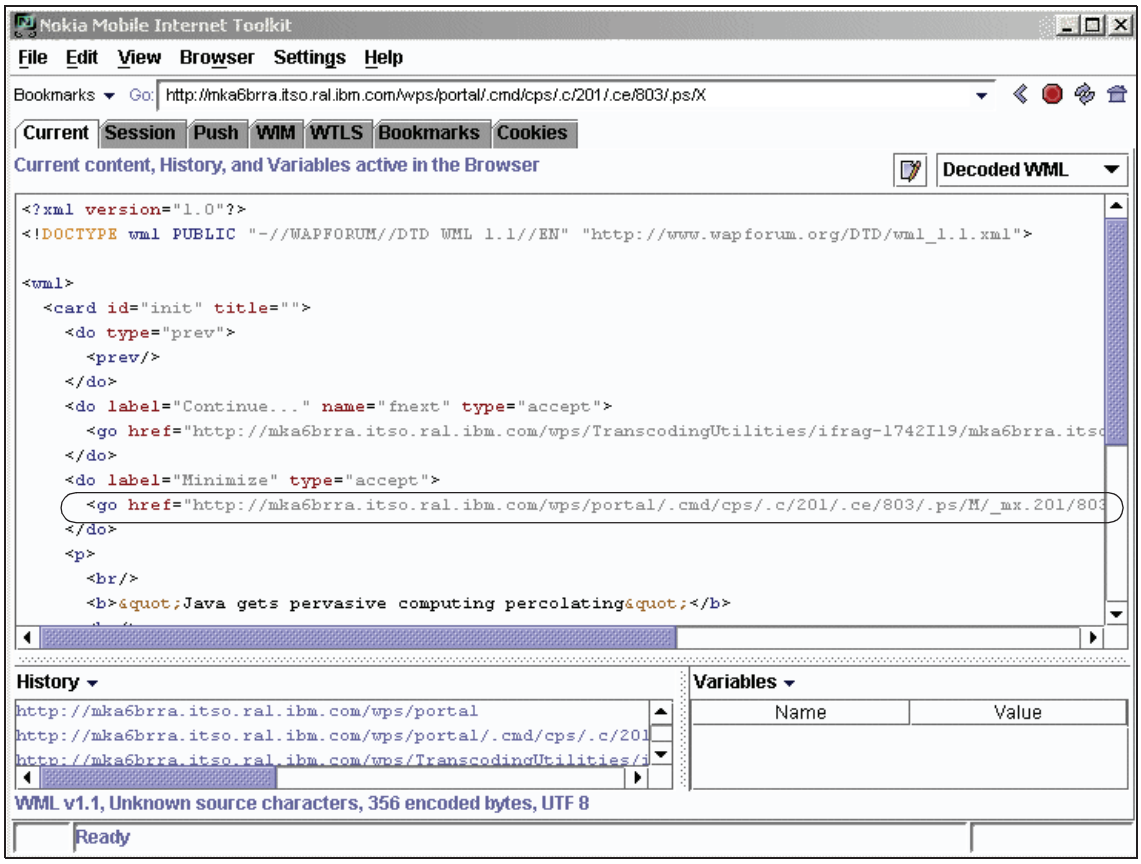


Figure 17-8 WML deck in Nokia Toolkit showing a link for a fragment

Figure 17-9 on page 523 illustrates the option to request a fragment.

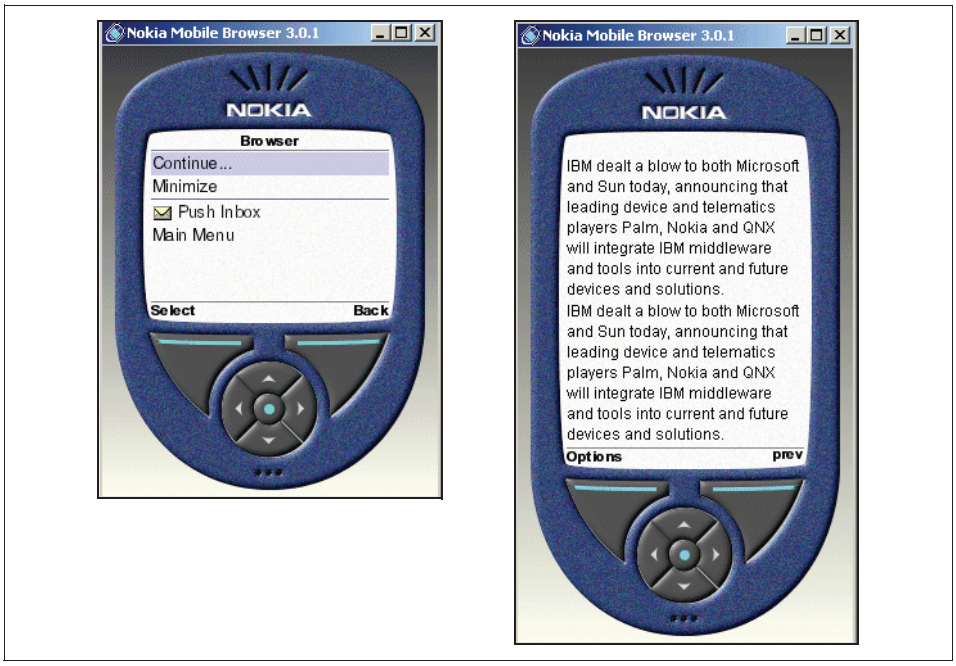


Figure 17-9 Requesting a fragment

The WML fragment link can also be monitored when using the Request Viewer tool. See Figure 17-10 on page 524.

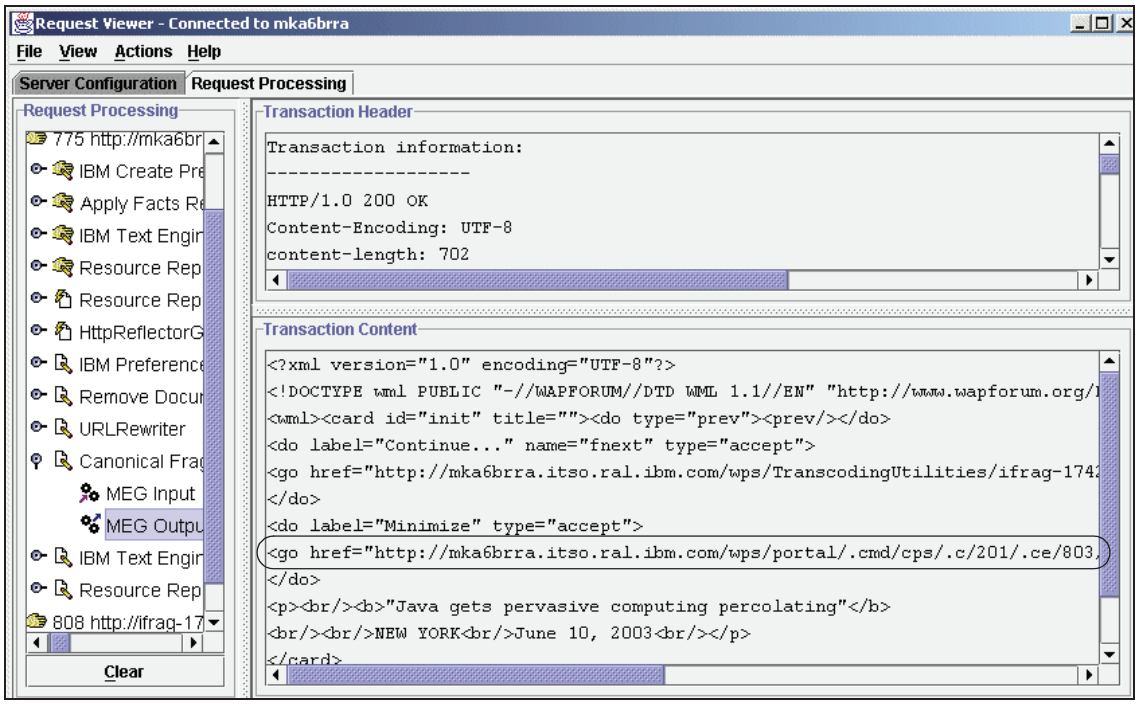


Figure 17-10 Monitoring fragmentation with Request Viewer tool

17.2.2 Scenario 2: Using a forward proxy

In this section we show a scenario using a forward proxy (see Figure 17-11).

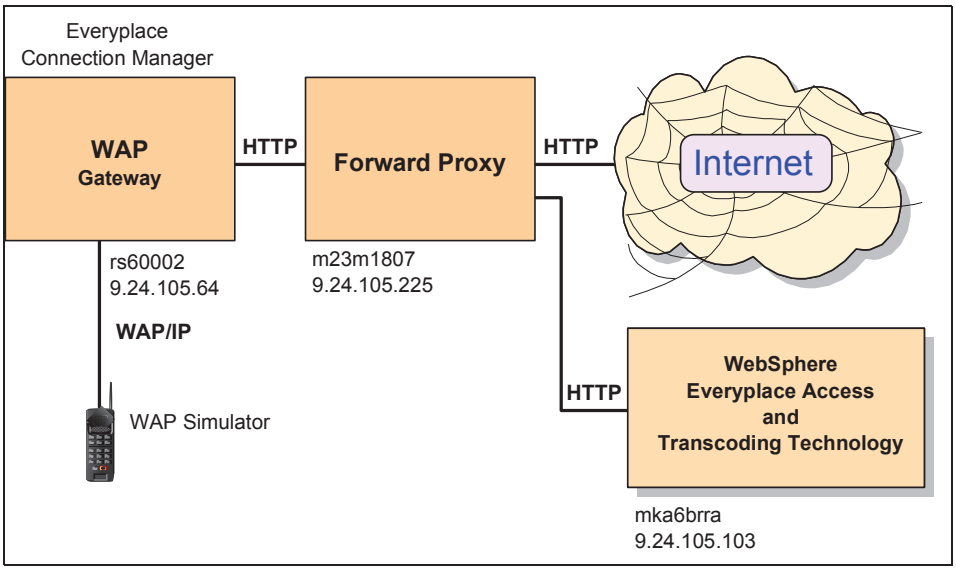


Figure 17-11 Intranet and Internet access using a forward proxy

The forward proxy in this scenario will give you access to Web sites on the Internet. However, in this scenario Portal-level WML fragmentation takes place for portlet content in WebSphere Everyplace Access only.

This is a very simple scenario, since no extra configuration is required other than the WAP Gateway, which needs to be configured to use the WAP forward proxy. Figure 17-12 on page 526 illustrates the WAP Gateway configuration for this scenario.

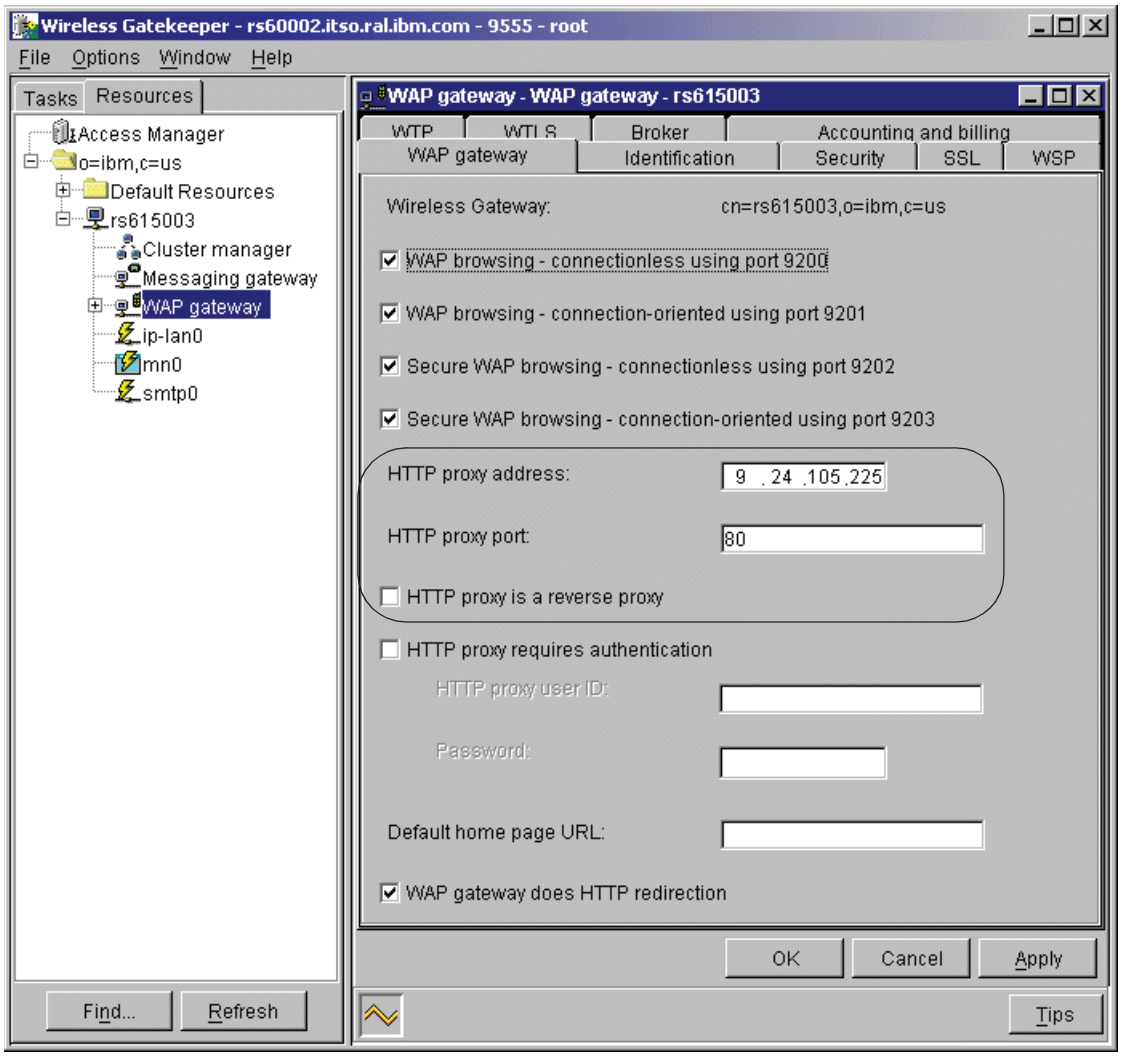


Figure 17-12 WAP Gateway configuration using a forward proxy

Note: In this scenario, URLs and links must include the WebSphere Everyplace Access computer name and listening port.

17.2.3 Scenario 3: Using a forward proxy and reverse proxy

In this section, we describe a sample scenario for Portal-level WML fragmentation in a WebSphere Everyplace Access environment where a forward proxy is configured in the WAP Gateway to provide general access to Internet sites. In addition, a reverse proxy is used to access WebSphere Everyplace Access portlet content. The scenario is illustrated in Figure 17-13.

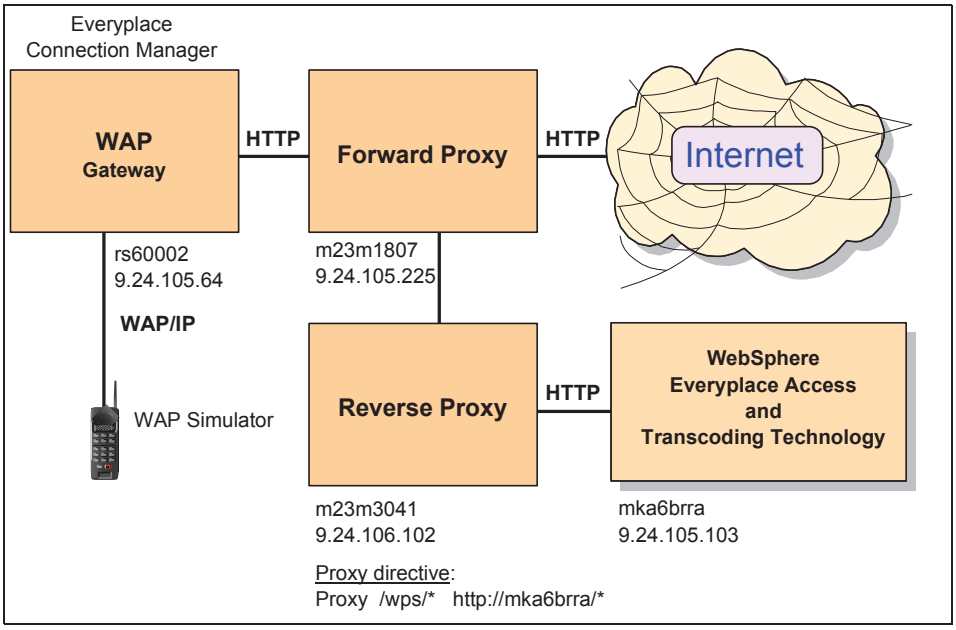


Figure 17-13 Sample scenario using a forward proxy for Internet access

In this scenario, consider the following:

1. The WAP Gateway is configured to use a WAP forward proxy. See Figure 17-12 on page 526.
2. Computer names in URLs and links must point to the reverse proxy.
3. The reverse proxy must be configured to forward requests to the WebSphere Everyplace Access machine. For example, to forward all HTTP requests with the string `wps` in the link, you configure the proxy directive as follows:

   ```
   Proxy    /wps/*          http://mka6brra.itso.ral.ibm.com/*
   ```

   Where `mka6brra` is the host name (you can also use the IP address) of the WebSphere Everyplace Access machine and listening port 80.
4. All fragment requests will be forwarded to Portal-level transcoding for retrieval:

   ```
   http://mka6brra.itso.ral.ibm.com/wps/TranscodingUtilities/ifrag-......
   ```

18

DB2 Everyplace applications with WebSphere Studio Device Developer

This chapter describes the use of WebSphere Studio Device Developer to create Java 2 Micro Edition (J2ME) based applications using DB2 Everyplace database and data synchronization capabilities provided by WebSphere Everyplace Access.

© Copyright IBM Corp. 2003. All rights reserved.

18.1 Overview

WebSphere Studio Device Developer is part of the WebSphere Studio family of products that have been developed based on the Eclipse workbench technology. In this section we provide an overview of this product.

18.1.1 Introduction

The Eclipse workbench is an open source platform, designed by IBM and released to the open source community. It is an open, portable, universal tooling platform that provides frameworks, services and tools for building development tools.

In essence the workbench provides the tool infrastructure. With this infrastructure in place, the tool builders are able to focus on the actual building of their tools. The workbench has been designed for maximum flexibility to support the development of tools for new technologies that may emerge in the future.

Development tools written for the workbench should support a role-based development model, in which the outcomes of the developers' work will be consistent. The developers should not have to be concerned with how different individual tools may be treating their files.

The WebSphere Studio family of products is an integrated platform (IDE) for development, testing, debugging and deploying. It provides support for each phase of the application development life cycle.

For more information about the WebSphere Studio family of products, see for example `http://www.ibm.com/software/awdtools/.`

18.1.2 Java 2 Micro Edition

Java 2 Micro Edition (J2ME) is the Java platform for a broad range of devices, including PDAs, cellular phones and embedded devices. The J2ME platform includes APIs for the development of Java applications in such devices.

The J2ME architecture

J2ME define configurations and profiles for building complete Java runtime environments. Each configuration and profile is targeted to a related category of devices.

Configurations

J2ME is composed of the virtual machine and core class libraries. Each configuration provides the base functionality for developing pervasive applications. Two configurations currently exist:

- Connected Limited Device Configuration (CLDC)

  For devices with sporadic network connections, slow processors, and low memory. Typically includes cell phones and low-end PDAs with 16-bit or 32-bit processors and 512 KB or less of memory for the Java runtime and applications.

- Connected Device Configuration (CDC)

  This configuration is for devices with powerful processors, lots of memory and maybe a large bandwidth, such as high-end PDAs (ARM based). This configuration implements an almost complete subset of the J2ME.

WebSphere Studio Device Developer comes with the J9 Java Virtual Machine and with implementations for the two configurations mentioned above.

Note: J9 is not currently available on Palm 5.x. However, it is expected to be available in a future release of WebSphere Studio Device Developer.

Profiles

Profiles are high-level APIs that define the application life cycle model, the user interface, and device-specific properties. The more important profiles are:

- Mobile Information Device Profile (MIDP)

  The MIDP profile is used on cell phones and low-end PDAs. It offers a basic user interface, basic network connectivity, local storage, and application management. The MIDP profile, together with CLDC configuration, provides a complete Java runtime environment for applications that runs on resources-constrained devices.

- Foundation Profile (FP)

  This is the lowest level profile of the CDC configuration. It provides a broad subset of the J2SE platform including network, I/O, security and text support. It doesn't include any user interface toolkit. However, it can be combined with others profiles that provide GUI support.

- Personal Profile (PPro)

  This profile provides a complete implementation of the Abstract Window Toolkit (AWT) for the development of GUI-centric applications. Together, the Foundation and Personal Profiles provide the most complete Java runtime for the development of business-oriented applications.

To complete the scenario, WebSphere Studio Device Developer provides an implementation of the JDBC APIs for database access on the device. However, it is not part of any J2ME configuration or profile, but is part of the WebSphere Custom Environment, an IBM-specific environment that fills in the gaps in the J2ME platform.

To learn more about J2ME, visit http://java.sun.com/j2me.

18.1.3 WebSphere Micro Environment

WebSphere Studio Device Developer comes by default with WebSphere Micro Environment (WME) that supports the development of J2ME-based applications. It provides an implementation for the CDC and CLDC configurations, and an implementation for the Mobile Information Device, Foundation and Personal profiles.

The WebSphere Custom Environment (WCE) provides custom characteristics for device and embedded development.

18.2 Installing WebSphere Studio Device Developer

WebSphere Studio Device Developer can be installed stand-alone or as a component of WebSphere Studio Application Developer Version 5. Because WebSphere Everyplace Access comes with WebSphere Studio Site Developer instead of WebSphere Studio Application Developer, we describe here the stand-alone installation option.

To install WebSphere Studio Device Developer in stand-alone mode:

1. Run the setup.exe file located in \preview\wsdd\win on the WebSphere Studio Device Developer V5.5 CD.
2. Select a language for the install, as shown in Figure 18-1. Click **OK**.

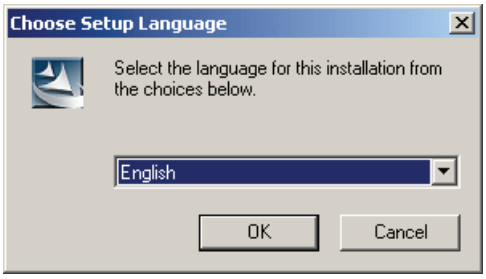


Figure 18-1 Selecting the language for the install

3. The Welcome window appears, as shown in Figure 18-2. Click **Next**.

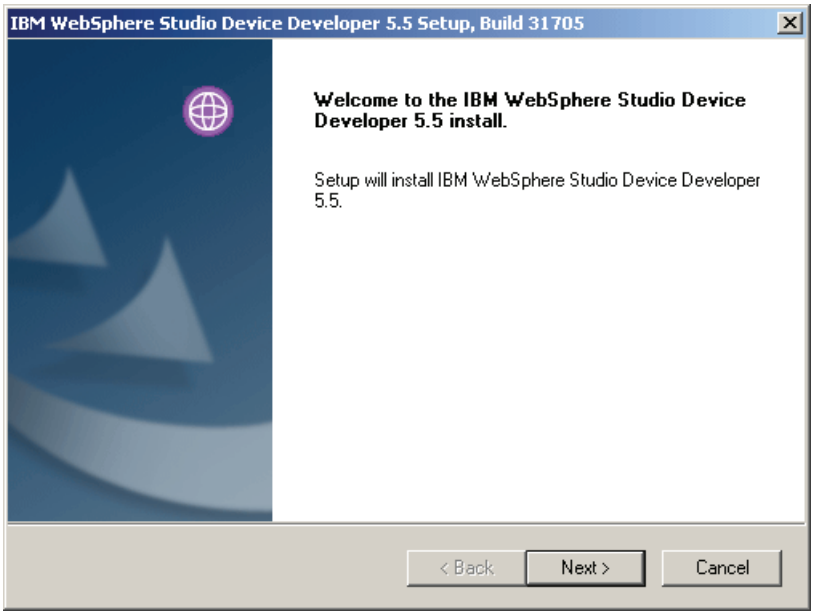


Figure 18-2 WebSphere Studio Device Developer Welcome install window

4. Read the license agreement and click **Yes**.
5. Select the destination folder for the product as shown in Figure 18-3 on page 534. Click **Next**.

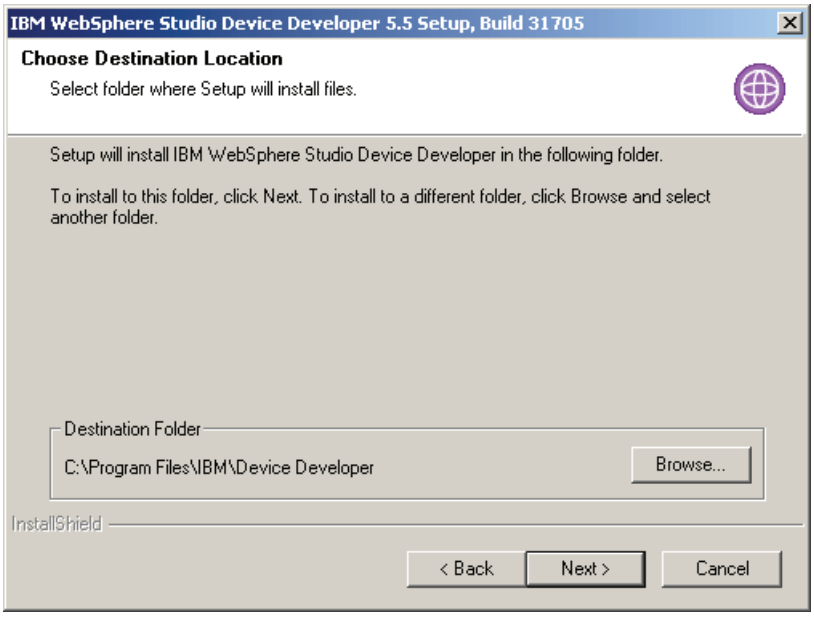


Figure 18-3 Choosing destination location

6. Select the program folder destination, as shown in Figure 18-4. Click **Next**.

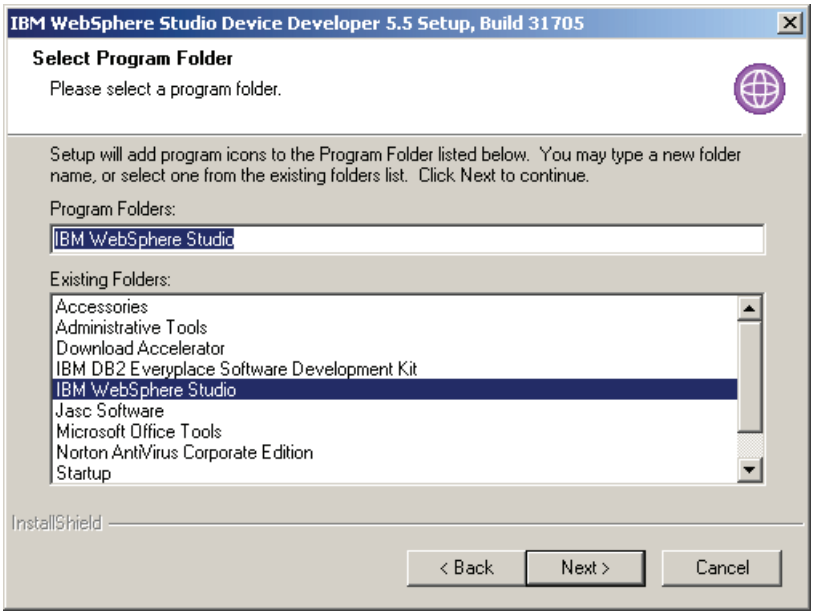


Figure 18-4 Selecting the program folder

7. Review the installation settings, as shown in Figure 18-5. Click **Next**.

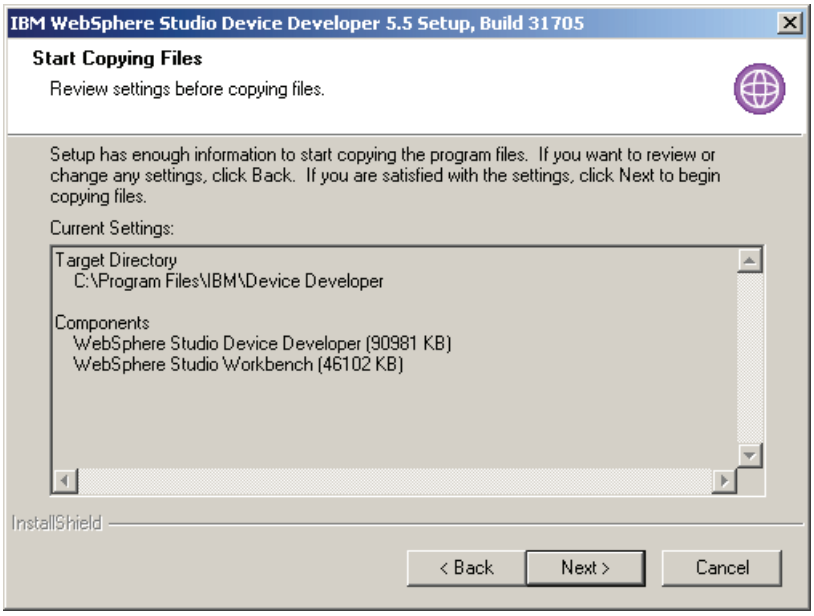


Figure 18-5 Reviewing the installation settings

8. Wait while the installation program copies the required files to your PC.
9. The installation has ended successfully as shown in Figure 18-6 on page 536. Click **Finish** to exit the wizard.

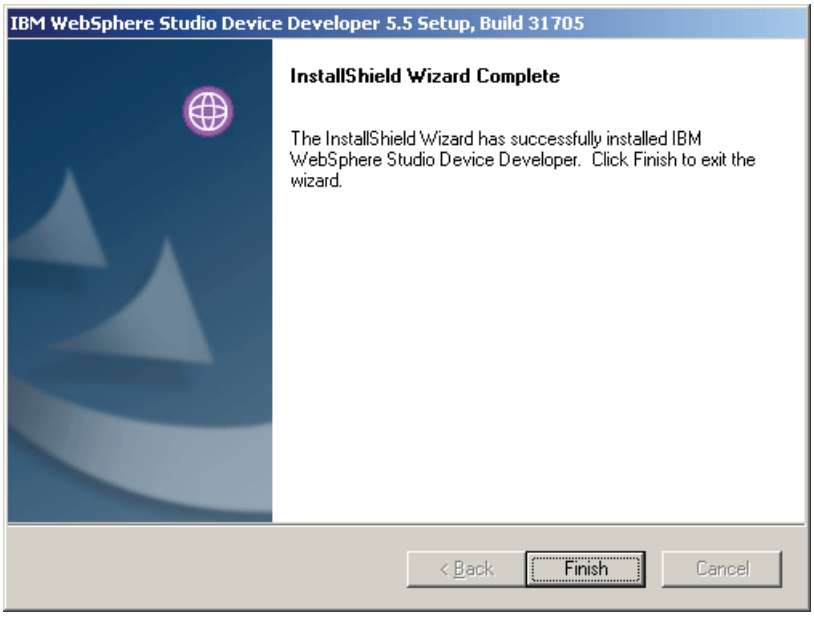


Figure 18-6 WebSphere Studio Device Developer installed successfully

18.3 Working with WebSphere Studio Device Developer

In this section, we show you how to use the Device Developer workbench.

18.3.1 Using the workbench

Start WebSphere Studio Device Developer using the shortcut on your Start menu. Select **Programs -> IBM WebSphere Studio -> Device Developer**. When you start the program you see the main window, shown in Figure 18-7 on page 537.

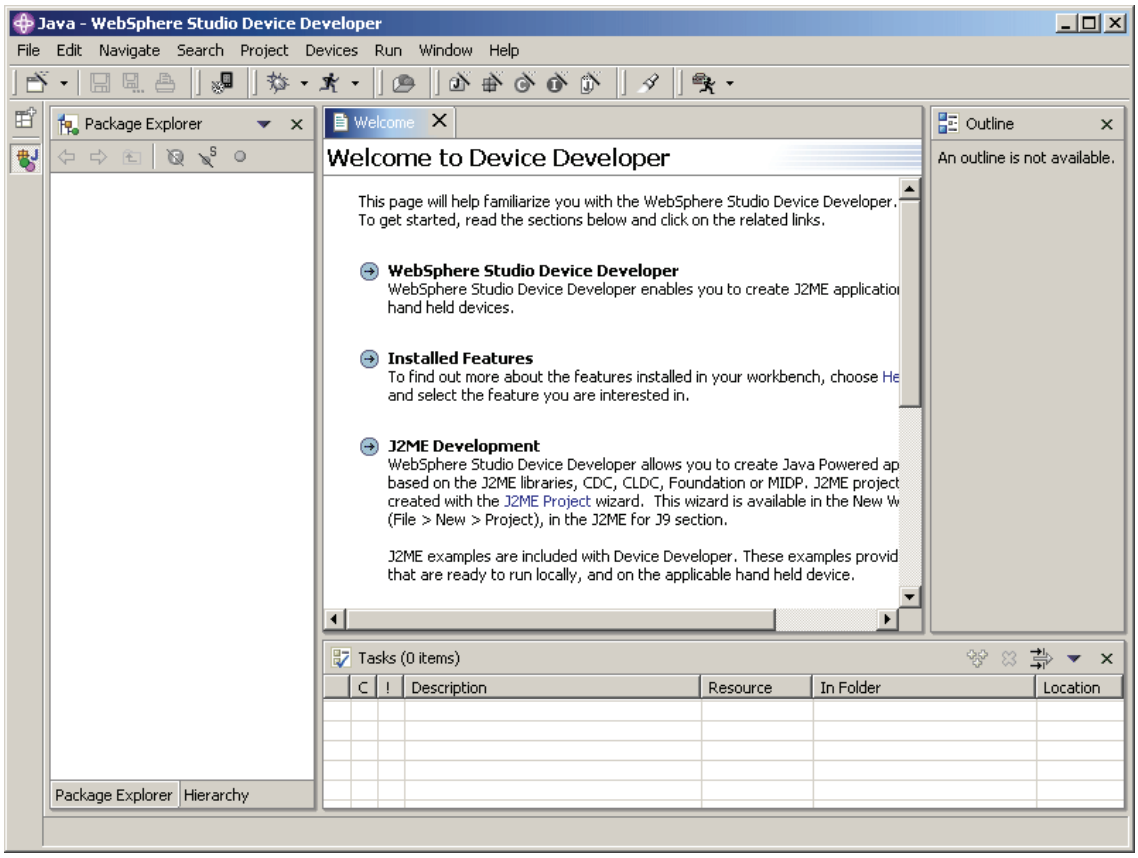


Figure 18-7 The WebSphere Studio Device Developer workbench

For device Java applications, you will work almost always in the Java perspective.

18.3.2 Using the Update Manager

The Update Manager tool allows you to install features that don't come with the packaged version of the product. Among these are:

- WebSphere Custom Environment
  - Database Enabler
  - Serial Library
  - JCLMax Library
  - Xtreme Palm Library
- Updates for the product

To install these features, you should use the Update Manager perspective. For more information about the workbench, views and perspectives, review 18.3.1, “Using the workbench” on page 536.

For the development of DB2 Everyplace (DB2E) applications on Windows CE targets, you need to install the Database Enabler library. Follow these steps to install it in the WebSphere Studio Device Developer environment:

1. Go to the Update Manager perspective by selecting **Help -> Software Update -> Update Manager**, as shown in Figure 18-8.

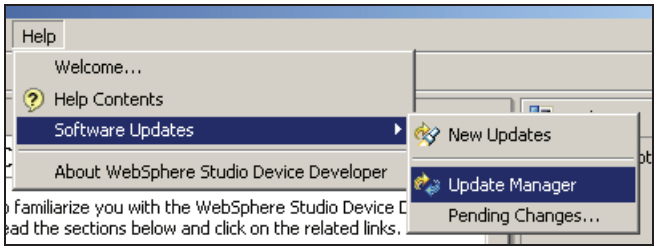


Figure 18-8 Going to the Update Manager perspective

2. In the Features Updates view, click the **Sites to Visit** option to expand the tree, as shown in Figure 18-9.

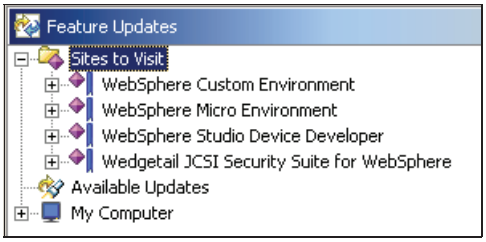


Figure 18-9 The Features Updates view

Note: if you can't see the Features Updates view, reset the perspective by selecting **Window -> Reset Perspective**.

3. Select **WebSphere Custom Environment -> WebSphere Custom Environment**, as shown in Figure 18-10 on page 539.

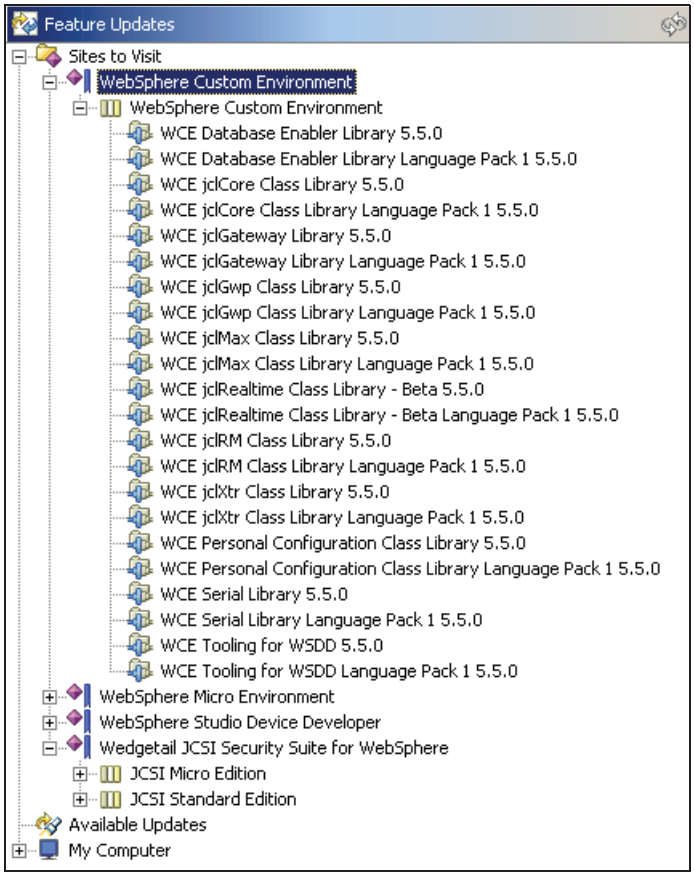


Figure 18-10 WebSphere custom environment features

4. Select the **WCE Database Enabler Library 5.5.0** option. In the Preview view you see the selected feature details, as shown in Figure 18-11 on page 540. Click **Install**.

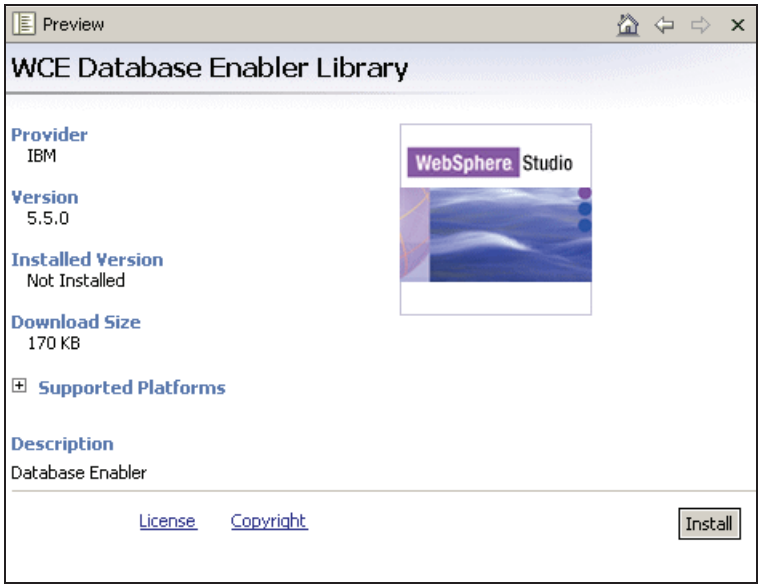


Figure 18-11 Feature details

5. You see the Install Features wizard, as shown in Figure 18-12 on page 541. Click **Next** to continue

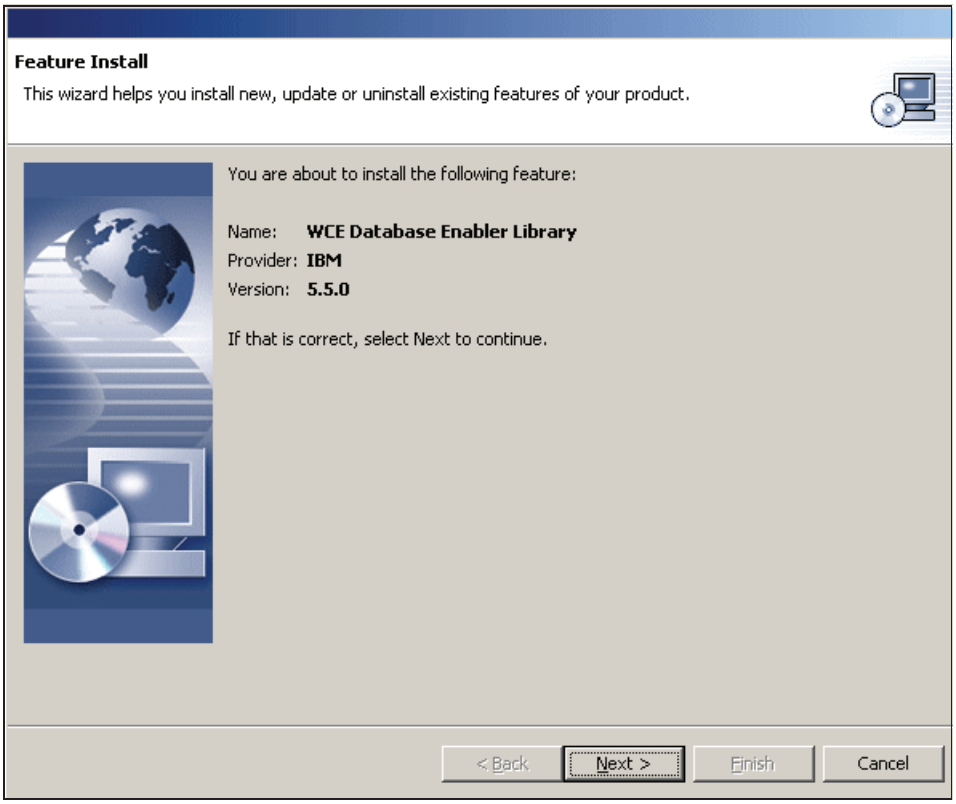


Figure 18-12 Feature Install wizard

6. Accept the license agreement by selecting **I accept the terms in the license agreement**, and click **Next**.
7. The wizard shows you where the feature will be installed, as shown in Figure 18-13 on page 542. Click **Finish**.

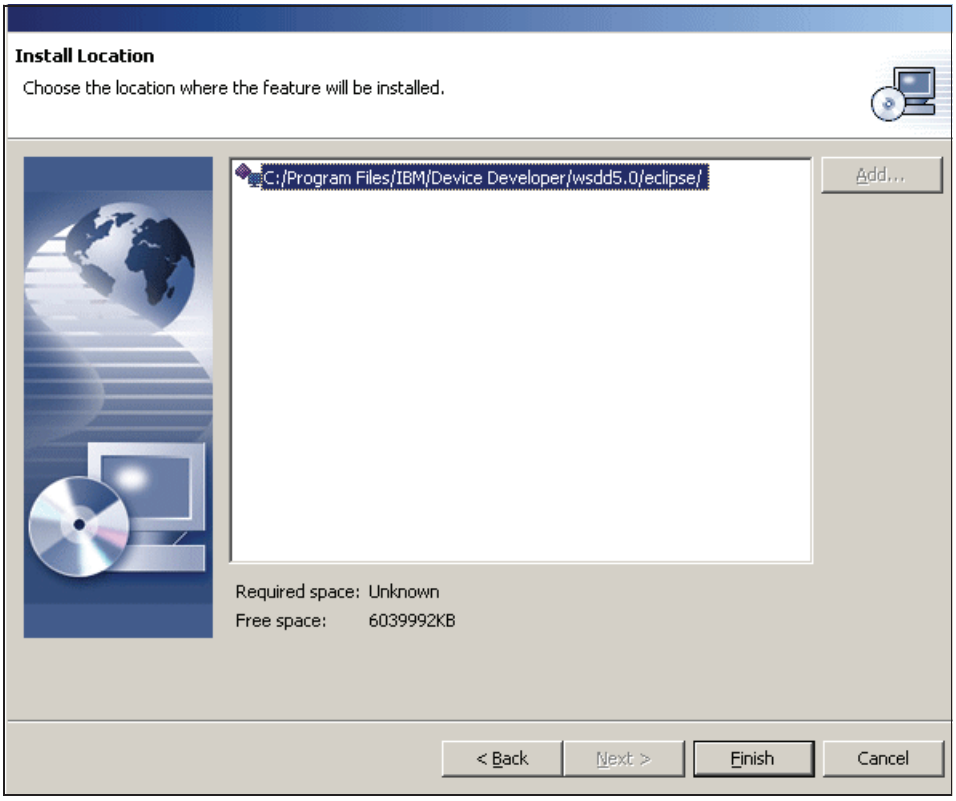


Figure 18-13 Selecting the feature install location

8. The wizard warns you about an unsigned feature. This is safe to ignore. Click **Install**.

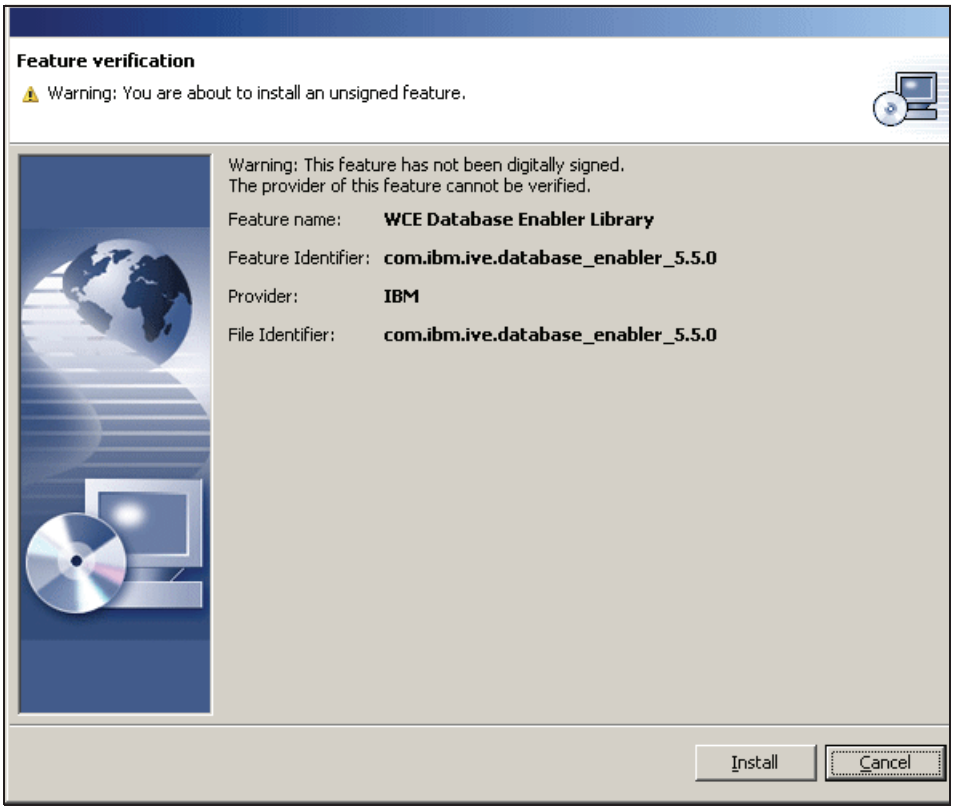


Figure 18-14 Feature verification window

9. Wait while the feature is installed, as shown in Figure 18-15 on page 544.

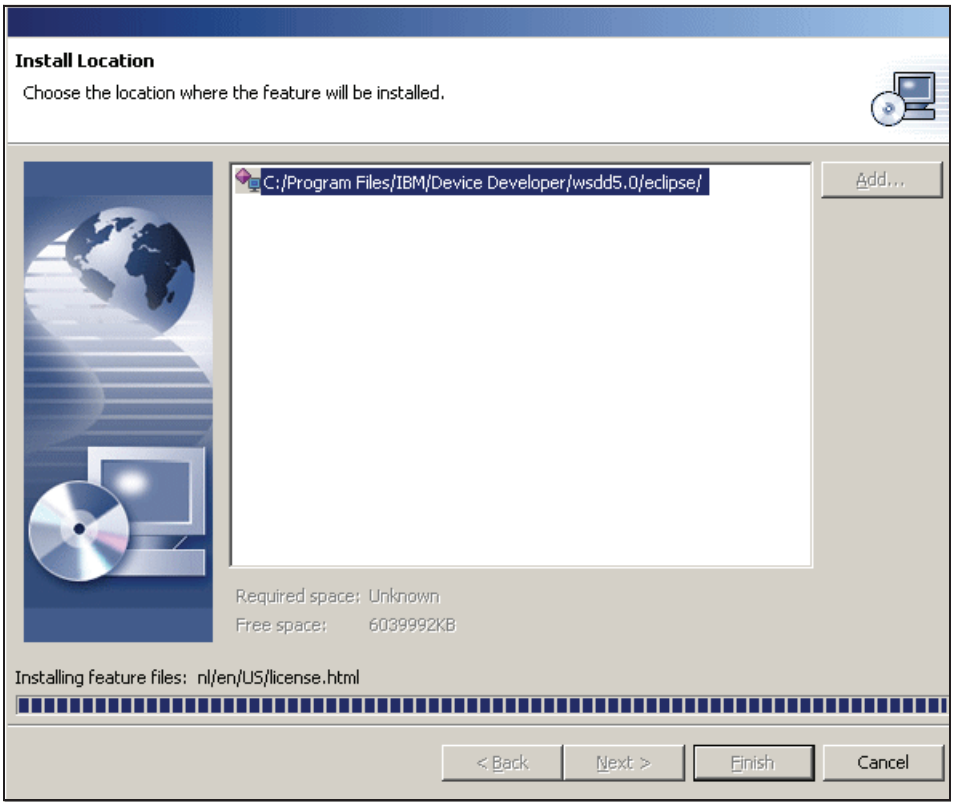


Figure 18-15 Installing the selected feature

10. Click **Yes** to restart the program, as shown in Figure 18-16.

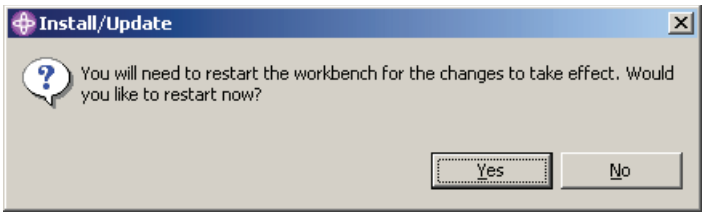


Figure 18-16 Restarting the workbench

Tip: If you are installing several features at the same time, respond **No** when the wizard ask you to restart the workbench, except for the last, when you should respond **Yes**.

18.4 Sample scenario

The sample scenario described here illustrates a sales force application using the Personal Profile configuration. We will develop a sales force application for Pocket PC targets using the J2ME Personal Profile (PPro) configuration. This comes with an implementation of the Abstract Window Toolkit (AWT) and is supported by the J9 Foundation Profile.

Note: The PPro configuration is not actually supported on Palm targets. If you wish to develop GUI-based applications for Palm devices, you should use the J2ME MIDP profile, or for more advanced applications, the WCE Xtreme configuration.

To develop an application using the PPro profile, follow these steps:

1. Define the database tables for the application.
2. Configure the Pocket PC device for development.
3. Create the project on WebSphere Studio Device Developer.
4. Set up the build and test environment.
5. Write the Java classes that compose the application.
6. Test the application on the device.

18.4.1 Define the database tables for the application

The database tables for the application are the same as in Chapter 19, “DB2 Everyplace applications with Mobile Application Builder (MAB)” on page 603. Refer to 19.6.1, “Table definitions for the application” on page 610 for more information.

18.4.2 Configure the Pocket PC device for development

You have to install and configure several components prior to testing and debugging applications on the Pocket PC device using the WebSphere Studio Device Developer workbench. The following components need to be configured:

- Microsoft Active Sync
- J9 runtime for Personal Profile configuration
- DB2e runtime
- Database support, JDBC driver and synchronization driver

Microsoft Active Sync

The Microsoft Active Sync software is necessary to provide connectivity between the development machine and the Pocket PC device. You can connect the Pocket

PC device using a USB port, a serial port, and an Ethernet card. Table 18-1 offers a resume of the connection methods.

Table 18-1 Connection methods for Pocket PC

Connection Method	Typical Connection Speed
Ethernet/wireless (TCP/IP LAN)	Fastest (10 Mbps for wireless, or more for wired)
USB cradle connection	Faster (up to 12 Mbps
Serial cradle connection	Slower (56 kbps to 115 kbps)

The serial connection is not recommended because of its slow transfer ratio. With the USB connection, you can connect the Pocket PC to the network using the ActiveSync pass-through feature; however it is not a true TCP/IP connection. For example the USB connection cannot be used to debug the application.

Note: You need a TCP/IP connection to use the debug capability of WebSphere Studio Device Developer. You can obtain one using an Ethernet card for your Pocket PC device. The WebSphere Studio Device Developer debugger will not work with an ActiveSync USB connection.

J9 runtime for Personal Profile configuration

You need to copy the following files from the WebSphere Studio Device Developer installation directory to the Pocket PC device to support the J9 runtime, where <WSDD> is the directory where you installed the WebSphere Studio Device Developer software:

- Copy from the <WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin to the \j9\bin directory on the device:
  - iverel20.dll
  - j9.exe
  - j9dyn20.dll
  - j9foun20.dll
  - j9int20.dll
  - j9max20.dll
  - j9prt20.dll
  - j9thr20.dll
  - j9vm20.dll
  - j9w.exe
  - j9zlib20.dll
  - swt-win32-2130.dll

- Copy from the <WSDD>\wsdd5.0\ive\runtimes\pocketpc\common\ive\lib\jclPPro directory on the desktop to the \j9\lib\jclPPro directory on the device:
  - ppro-ui-win.zip
- Copy from the <WSDD>\wsdd5.0\ive\runtimes\common\ive\lib\jclFoundation directory on the desktop to the \j9\lib\jclFoundation directory on the device:
  - classes.zip
  - locale.zip
- Copy from the <WSDD>\wsdd5.0\ive\runtimes\common\ive\lib directory on the desktop to the \j9\lib\ directory on the device:
  - charconv.zip

DB2e runtime

The DB2E runtime is installed as part of the Everyplace Client installation on the device. For more information about the the Everyplace Client installation, refer to Chapter 4, “Everyplace Client” on page 81.

Database support, JDBC driver and synchronization driver

Although the Everyplace Client installs the required JAR files on the device, these are installed in the \Windows directory. This means that the WebSphere Studio Device Developer would have to put these files in the classpath when creating the shortcut on the device to run the application. However, there is a limitation on the command size of a Pocket PC shortcut. It cannot be greater than 255 characters.

Therefore, to avoid potential problems and error messages from the WebSphere Studio Device Developer launch process, you have to unify the following DB2e library and database support JAR files into one file:

- db2ejdbc.jar
- isync4j.jar
- database_enabler.jar

The first two are installed on the Pocket PC device in the \Windows directory; you can copy them from there. Also, they come with the DB2 Everyplace Mobile Application Builder software. The last one is installed as a separate feature of WebSphere Studio Device Developer as explained in 18.3.2, “Using the Update Manager” on page 537. By default this file is installed in the <WSDD>\wsdd5.0\ive\runtimes\common\ive\lib\jclMax directory, where <WSDD> is the WebSphere Studio Device Developer installation directory.

To unify the JAR files, follow these steps:

1. Extract the files on any directory on the desktop. Remember that a JAR file is merely a ZIP file, so you can use any ZIP tool to extract the files. Ignore any warn about replacing files.
2. Using any ZIP tool, compress the extracted files in a file and name it db2e.jar.
3. Finally, copy the db2e.jar file to the \j9\lib directory on the Pocket PC device.

18.4.3 Create the project on WebSphere Studio Device Developer

Now that you has configured the Pocket PC device for development, you need to create a J2ME project for the sales force application. Follow these steps:

1. Start WebSphere Studio Device Developer by selecting **Programs -> IBM WebSphere Studio -> Device Developer**.
2. On the workbench, select **File -> New -> Project**.
3. You will see the New Project window. Select **J2ME for J9** on the left pane and select **Create J2ME project** in the right pane, as shown in Figure 18-17 on page 549. Click **Next**.

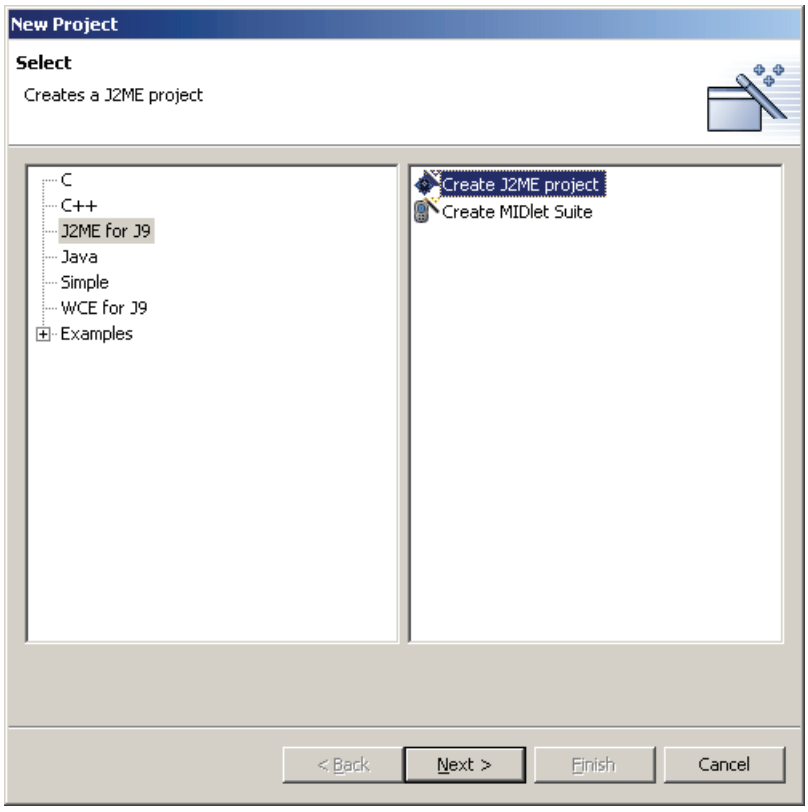


Figure 18-17 Creating a J2ME project

4. Enter `MobileOrderSystem` as the project name, as shown in Figure 18-18 on page 550. In this window you also can choose the directory for the project. In this case, leave as is. Click **Next**.

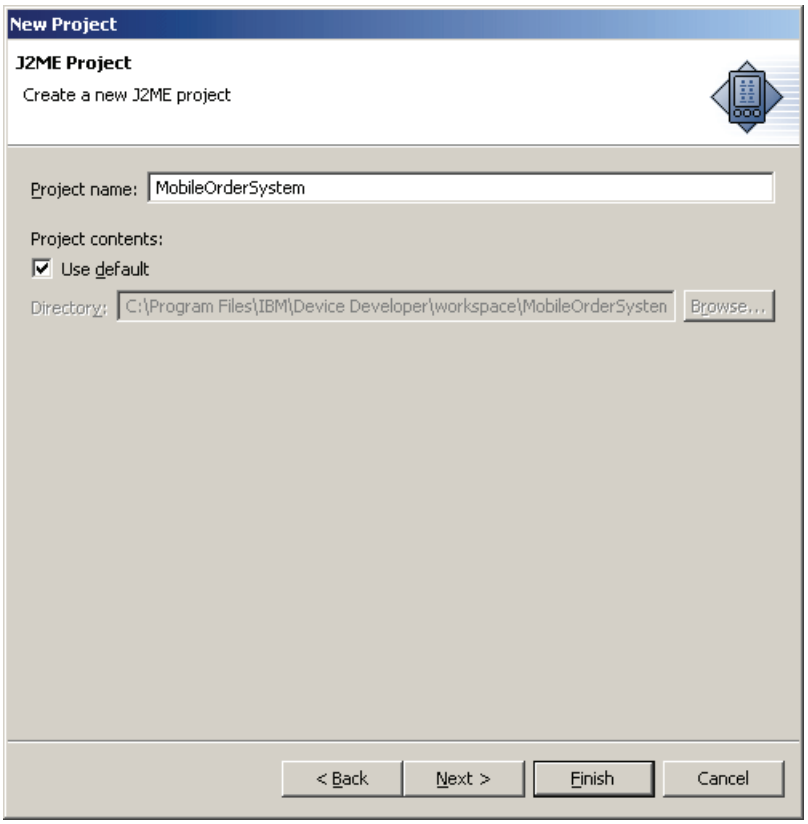


Figure 18-18 Setting the project name

5. In this window you can select the J2ME profile. Select the **WME jclPPro** library configuration, as shown in Figure 18-19 on page 551. Click **Finish**.

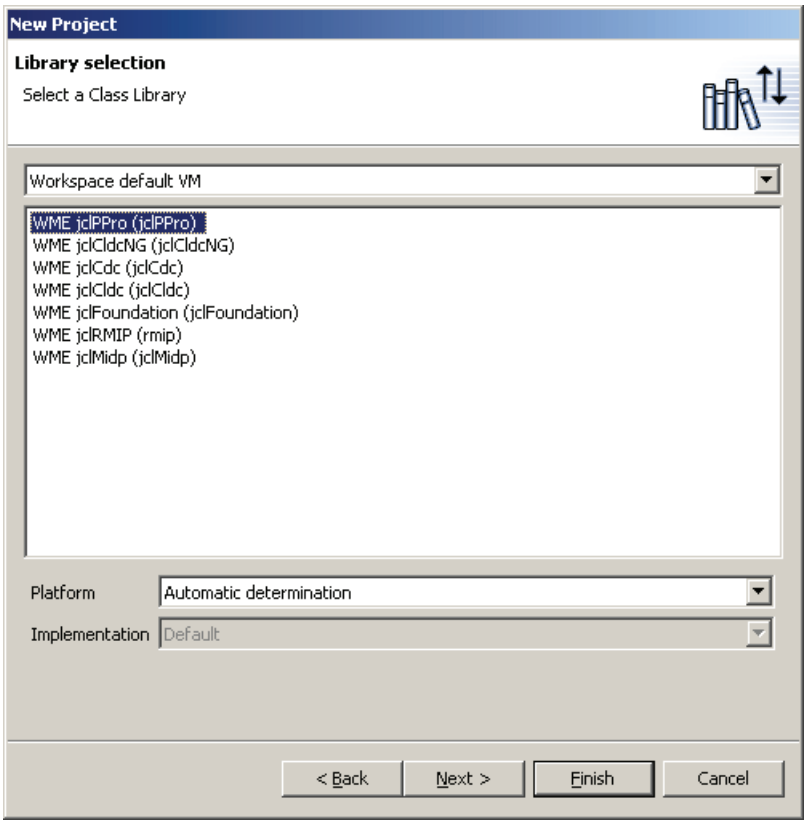


Figure 18-19 Selecting the class library

You can see the new project in the Package Explorer view. If you click the plus icon near to the project name to expand the tree, you will see the libraries that contain the configuration (PPro in this case), the root source directory (src) and the WebSphere Studio Device Developer Build configuration file, as shown in Figure 18-20 on page 552. In the next section you will set up the build and test environment.

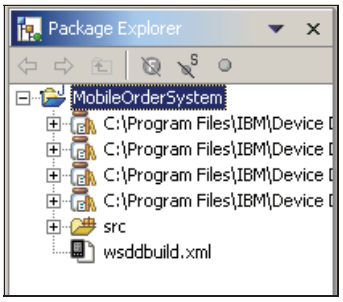


Figure 18-20 The J2ME project created

18.4.4 Set up the build and test environment

To set up the build and test environment, you have to perform the following tasks:

1. Create the device configuration
2. Create the main class of the application
3. Create the build target
4. Import the required libraries
5. Configure the jxeLinkOptions file
6. Create the launch configuration

Create a device configuration

To configure the device target for the project, follow these steps:

1. Select **Devices -> Configure** from the menu bar to open the Devices Configurations window.
2. On the Devices tree pane, select the Pocket PC Handheld device and click **New** to create a new device profile.
3. For the new device profile you need to specify the J9 runtime location, the Application install location, and the Shortcut install location. For the sample application, use the parameters from Table 18-2, as shown in Figure 18-21 on page 553. Click **OK**.

Table 18-2 Device configuration parameters

Parameter	Value
Device Name	MyPocketPC
J9 Runtime location	\j9
Application install location	\MobileOrderSystem
Shortcut install location	\Windows\Start Menu

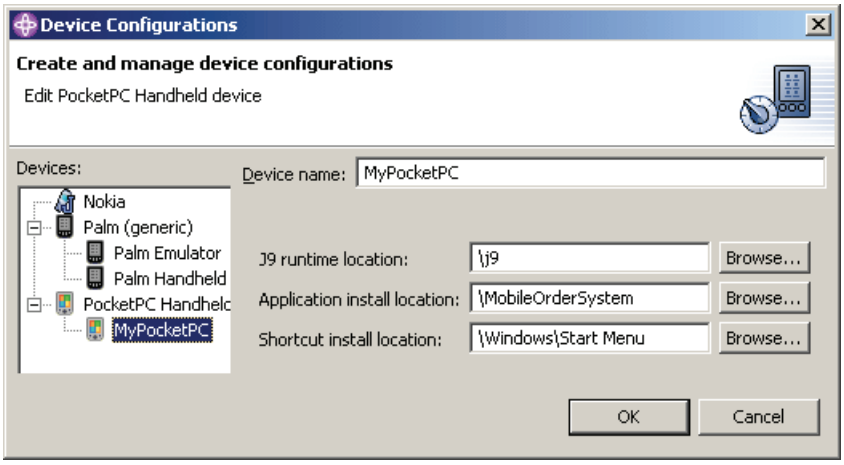


Figure 18-21 Creating a device configuration

Create the main class of the application

You have to create the Main class for the application. This class will be the startup class when the application runs. It must have a Java main method as defined in the Java language. Follow these steps to create it:

1. Select **File -> New -> Class** from the menu bar. The New class window will appear.
2. On the New class window, define the properties for the new class. Use `com.ibm.ral.itso.sv8030r.mos` for the class package, `MobileOrderSystem` for the class name and check **public static void main(String[] args)** , as shown in Figure 18-22 on page 554. Click **Finish**.

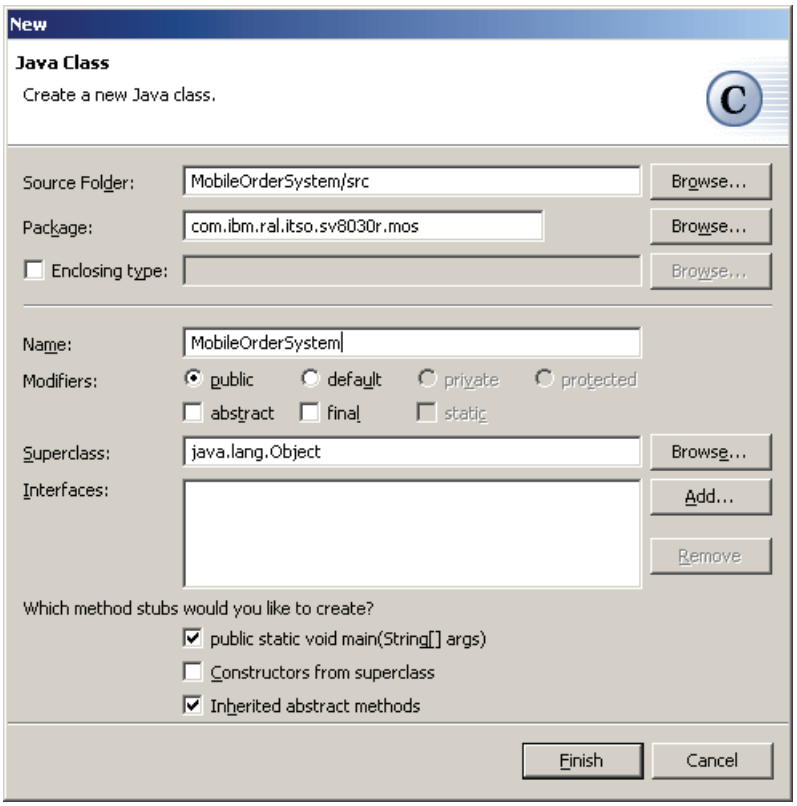


Figure 18-22 Defining the main class for the application

3. In the src folder, the new class is defined. The workbench opens it in the Code view for editing. Later, you will define the other classes in the application.

Create a build target

You have to define a build target in the wsddbuild.xml file. Follow these steps:

1. Double-click the **wsddbuild.xml** file to open the Build view, as shown in Figure 18-23 on page 555. Click **Add Build**.

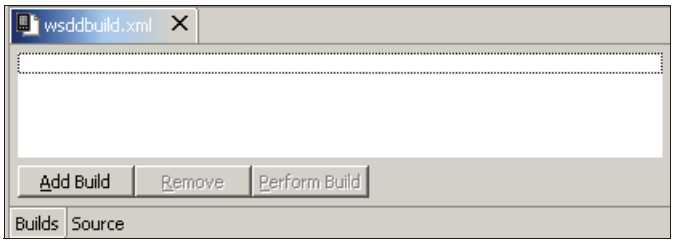


Figure 18-23 The Build view

2. The setup build window opens. Click **Browse** to select the Main class of the application. Select the **com.ibm.ral.itso.sv8030r.mos.MobileOrderSystem** class as the main class for the application, as shown in Figure 18-24. Click **Finish**.

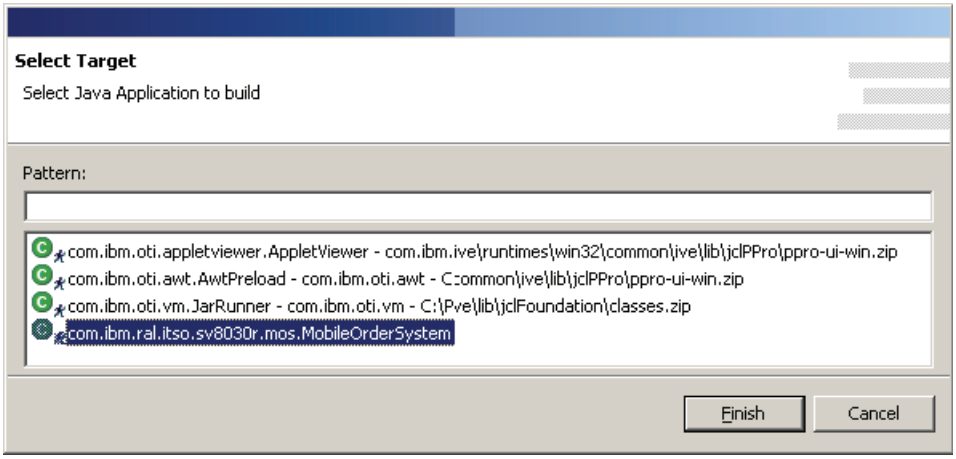


Figure 18-24 Selecting the main class for the build configuration

3. Now the setup build window is correctly configured, as shown in Figure 18-25 on page 556. Note that, by default, the build name is the same as the main class name. Make sure that **J9 for PocketPC ARM** is selected as the target platform. Click **Next**.

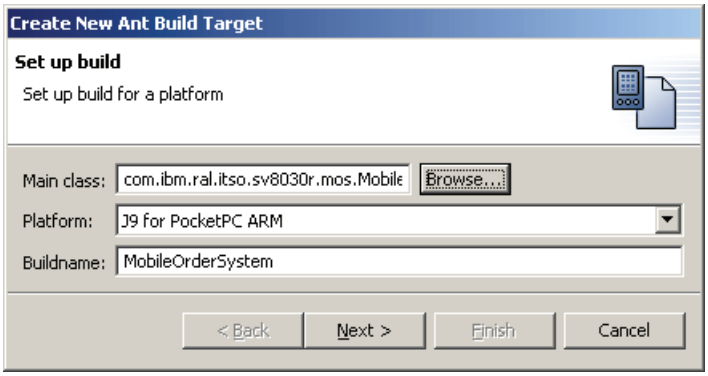


Figure 18-25 Build configured

4. The Build wizard ask you about additional configuration. Leave these settings as is. Click **Finish**.
5. You see the new build configured in the view. Behind the scenes, the wizard has created an Ant task that runs the SmartLinker tool to build the application as a JXE file that the J9 runtime in the device will execute.

Import the required libraries

Now you have to import the DB2e runtime and database support libraries (previously created in “Database support, JDBC driver and synchronization driver” on page 547). Follow these steps to import the db2ert.zip in the workbench:

1. Select **File -> New -> Folder** from the menu bar to create a new folder that will contain the imported library. On the New Folder window, select **MobileOrderSystem** as the root folder and enter `lib` for the folder name, as shown in Figure 18-26 on page 557. Click **Finish**.

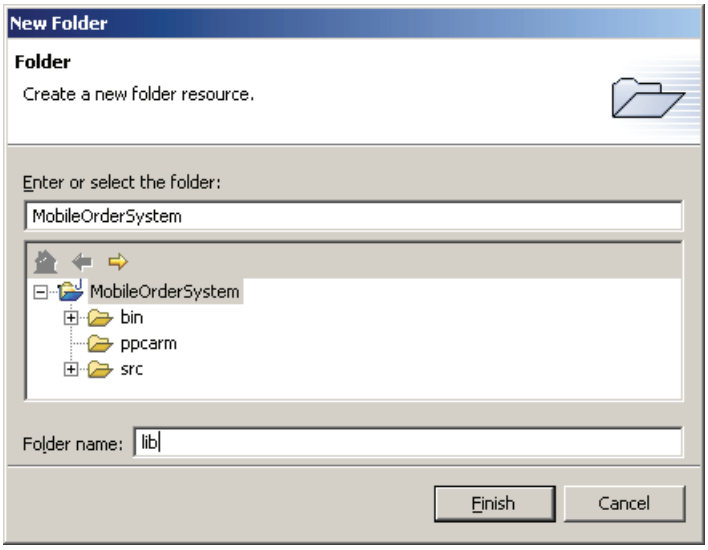


Figure 18-26 Creating the lib folder

2. Select the **lib** folder on the project tree view. Select **File -> Import** from the menu bar. On the Import wizard window, select **File system** as the import source**,** as shown in Figure 18-27. Click **Next**.

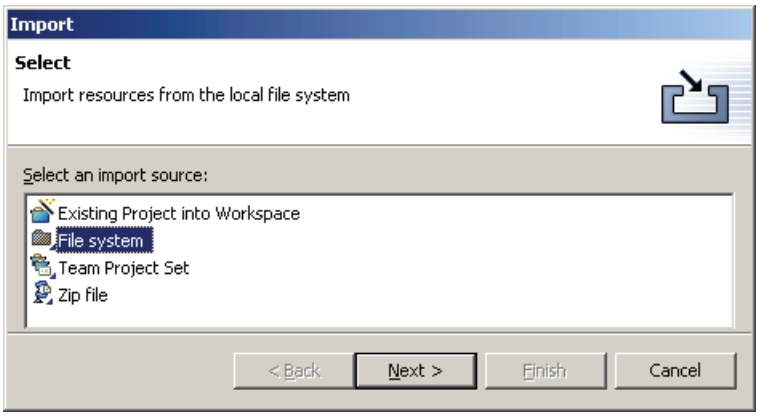


Figure 18-27 Importing from the file system

3. Click **Browse** to search the directory that contains the db2ert.zip file. When you find it, click **OK** on the Browse folder window to open the folder. In the left pane, you see the directory previously selected, and in the right pane, the db2ert.zip file. Select **db2ert.zip**, as shown in Figure 18-28 on page 558. Note that the destination folder for the file is the lib directory. Click **Finish**.

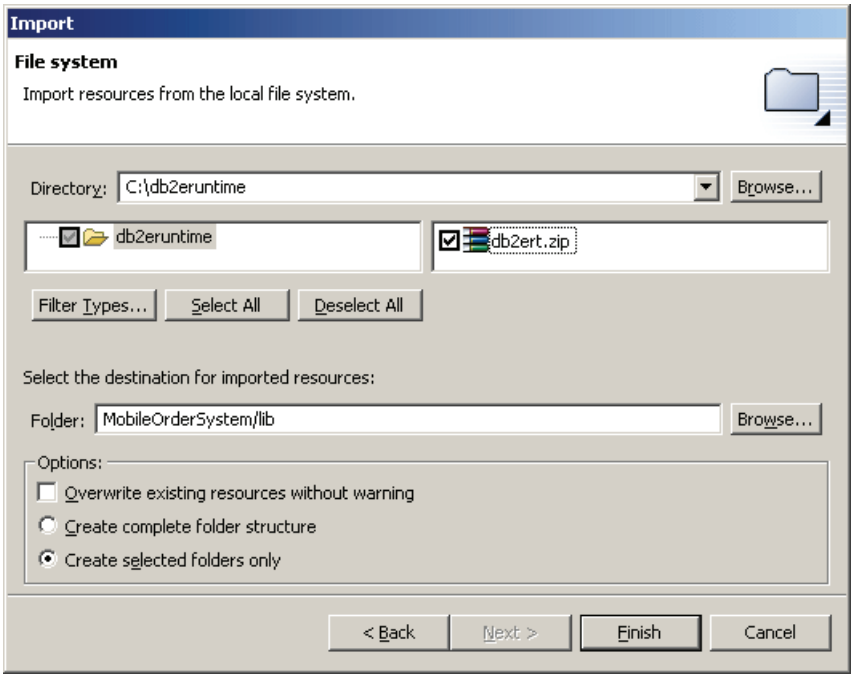


Figure 18-28 Importing the db2ert.zip file to the project

Now you need to add the previously imported library to the build path. Follow these steps:

1. Right-click the project name (**MobileOrderSystem**) in the project tree window and select **Properties**.
2. In the Project Properties window, select **Java Build Path** from the tree. Here you can configure several paths for the project. Click the **Libraries** tab, and you will see the current loaded libraries for the project, as shown in Figure 18-29 on page 559. Click **Add JARs**.

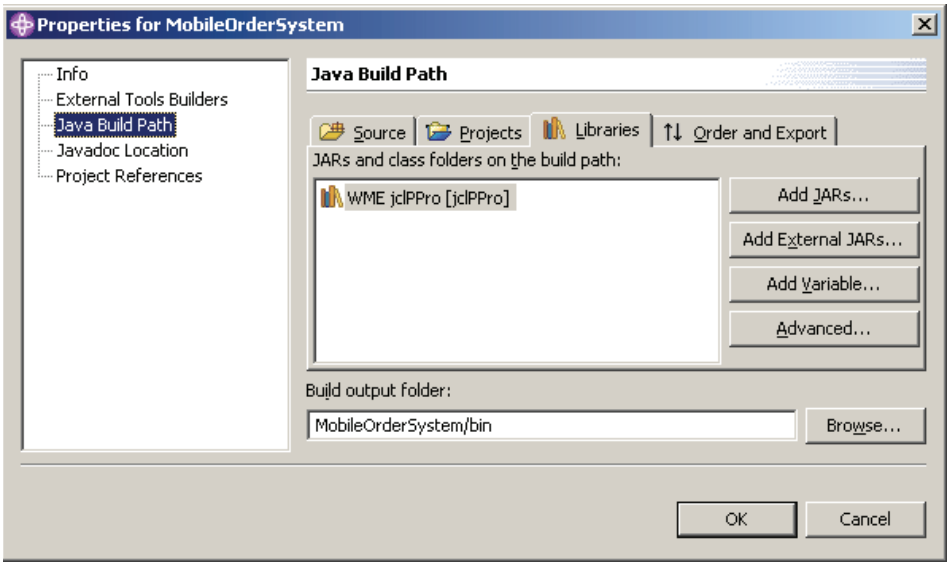


Figure 18-29 Libraries for the project

3. In the JAR Selection window, expand the tree to show the contents of the lib directory. Select **db2ert.zip** and click **OK**.
4. Now you have the library in your build path, as shown in Figure 18-30 on page 560. Click **OK** to finish.

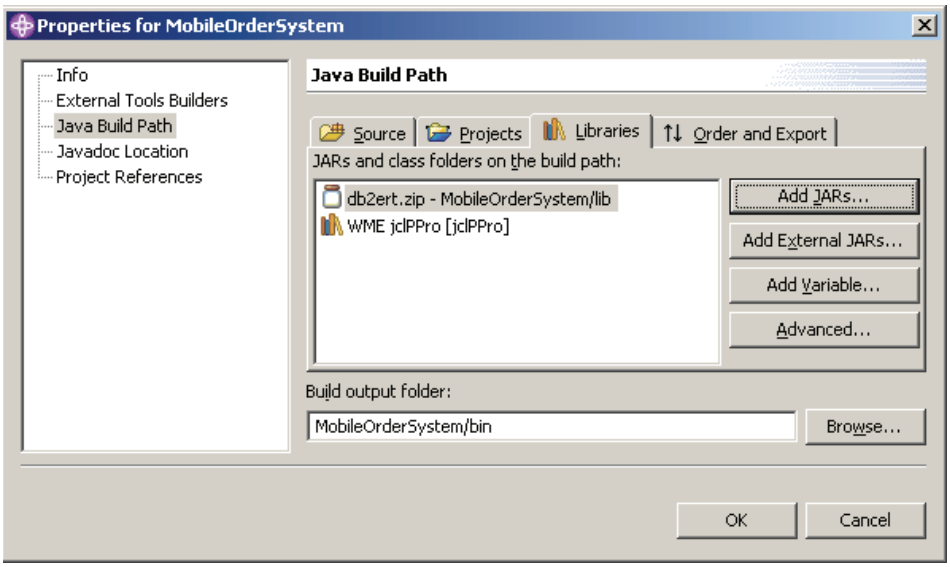


Figure 18-30 The db2ert.zip library added to the project

Configure the jxeLinkOptions file

You need to configure certain options in the MobileOrderSystem.jxeLinkOptions file to build the application successfully. Follow these steps:

1. Double-click **ppcarm\MobileOrderSystem.jxeLinkOptions**, located in the project tree view, to open the file in the jxeLinkOptions editor.
2. Select the **Input** tab. You see the classpath used for the SmartLink linker to build the jxe file, as shown in Figure 18-31. You need to add the db2ert.zip file to this list. Click **New..** near the Class search path list.

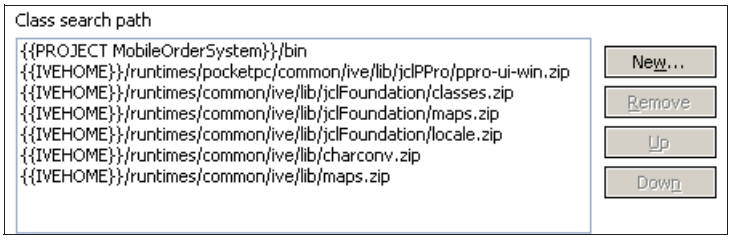


Figure 18-31 Class search path for SmartLink

3. On the Add classpath element window, select **PROJECT MobileOrderSystem** in the Macros field and enter `lib\db2ert.zip` in the Path field, as shown in Figure 18-32 on page 561. Click **OK**.

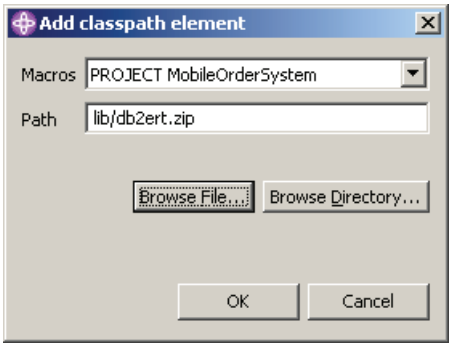


Figure 18-32 Adding db2ert.zip to the class search path

4. Now you have to indicate to SmartLinker that you don't want to eliminate the drivers classes from the JXE file. Select the **In/exclusion** tab. Make sure that **Include whole classes** is selected, as shown in Figure 18-33. Click **New**.

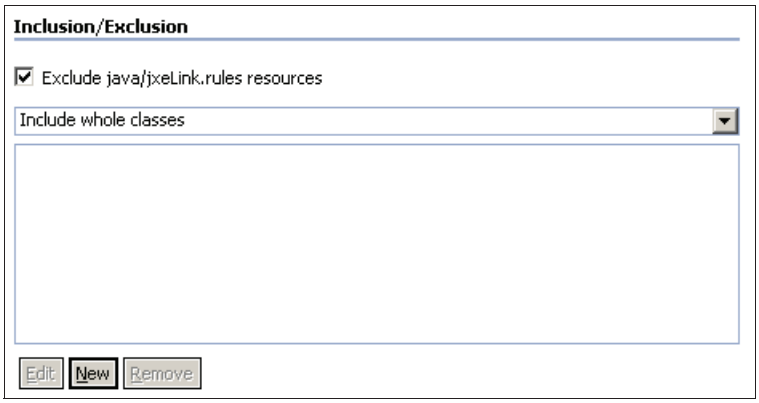


Figure 18-33 Inclusion/Exclusion view

a. Enter the rule pattern to include all the DB2e JDBC driver classes: `com.ibm.db2e.jdbc.*`, as shown in Figure 18-34 on page 562. Click **OK**.

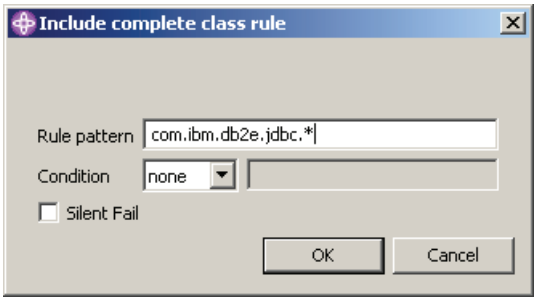


Figure 18-34 Including all the DB2e JDBC classes on the jxe file.

b. Click **New** again. Enter the rule pattern for the DB2e synchronization provider class `com.ibm.mobileservices.isync.db2e.jni.DB2eISyncProvider`, as shown in Figure 18-35. Click **OK**.

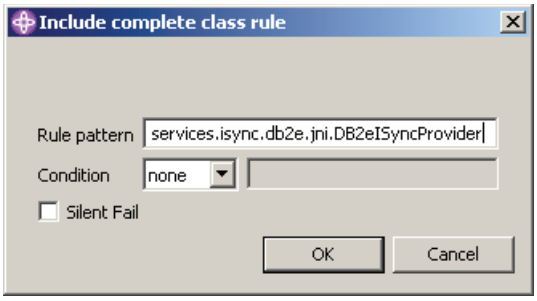


Figure 18-35 Including the DB2e Synchronization provider class

c. Add an entry for the following patterns also:

- `com.ibm.mobileservices.isync.*`
- `com.ibm.mobileservices.isync.event.*`

5. Select **File -> Save MobileOrderSystem.jxeLinkOptions** from the menu bar to save the changes. Close the jxeLinkOptions editor.

Create the launch configuration

To launch the application on the Pocket PC, you have to create a launch configuration. Follow these steps:

1. Select **Run -> Run** from the menu bar. The Launch Configurations window opens, as shown in Figure 18-36 on page 563. Select **Device Java Application** to create a new launch configuration for the Pocket PC. Click **New**.

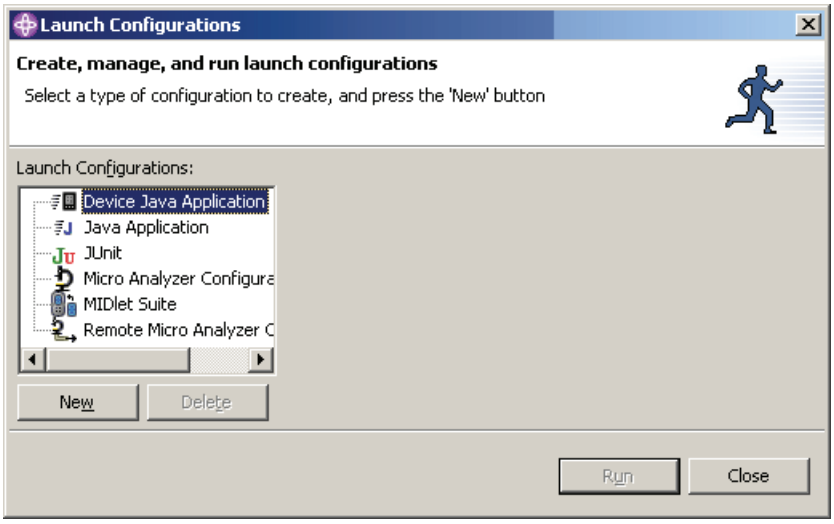


Figure 18-36 The Launch configuration window

2. A new launch configuration is created. Enter `MyConfiguration` as the name and select **MyPocketPC** as the device. Note that the Project field has `MobileOrderSystem` as the value. Click **Search**, near the Java Application field, to select the application to run, as shown in Figure 18-37 on page 564.

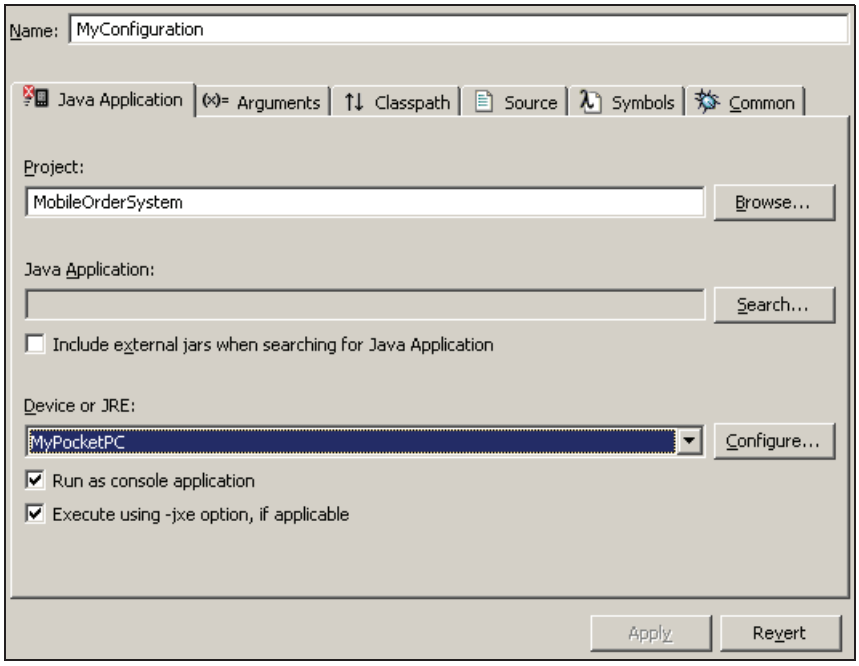


Figure 18-37 Configuring a new Java Application launch

3. On the Select Target window, select **MobileOrderSystem.jxe** as the Java application, as shown in Figure 18-38. Click **Finish**.

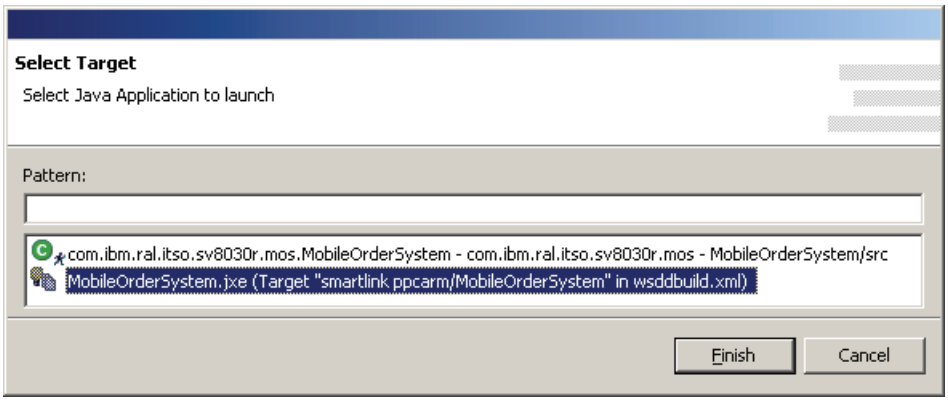


Figure 18-38 Selecting the Java target for the application

4. Click the **Arguments** tab. You need to add the db2ert.zip library to the classpath. In the VM arguments window, enter `-cp "\j9\lib\db2ert.zip"`, as shown in Figure 18-39 on page 565. Remember that in "Database

support, JDBC driver and synchronization driver" on page 547**,** you copied the db2ert.zip file to the \j9\lib library in the Pocket PC.

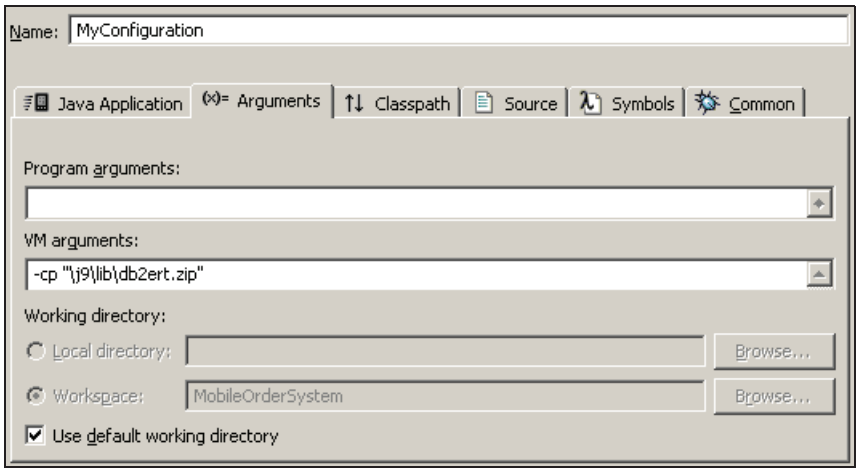


Figure 18-39 Configuring the classpath for the application

5. Click the **Classpath** tab, then click the **Bootstrap classes** tab. Here, you have to add the J2ME configuration libraries to the bootstrap classpath of the application. To do this, uncheck **Use default classpath,** as shown in Figure 18-40 on page 566, and click **Advanced**.

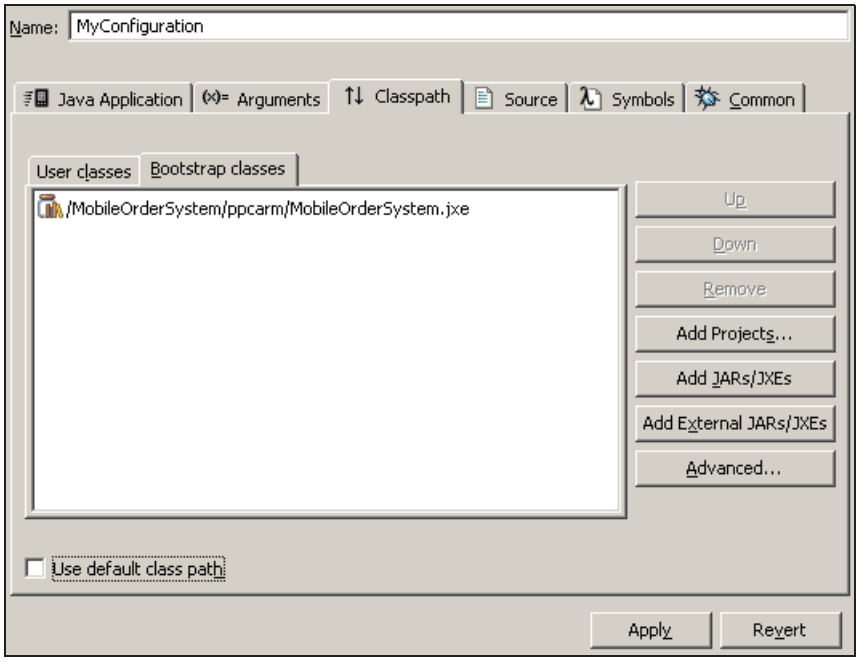


Figure 18-40 Configuring the bootstrap classes

6. On the Advanced Option window, select **Add Container** and select **J2ME Library** as the container, as shown in Figure 18-41. Click **OK**.

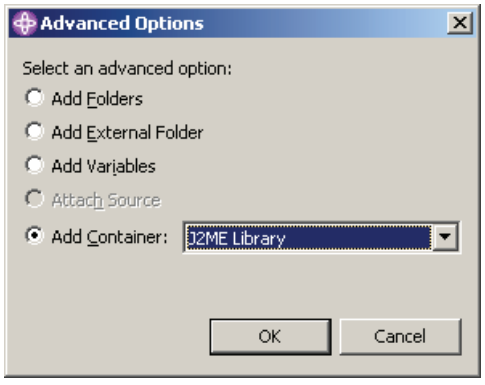


Figure 18-41 Adding the J2ME container to the bootstrap classpath

7. On the J2ME Library, select the **WME jclPPro** library, as shown in Figure 18-42 on page 567. Click **Finish**.

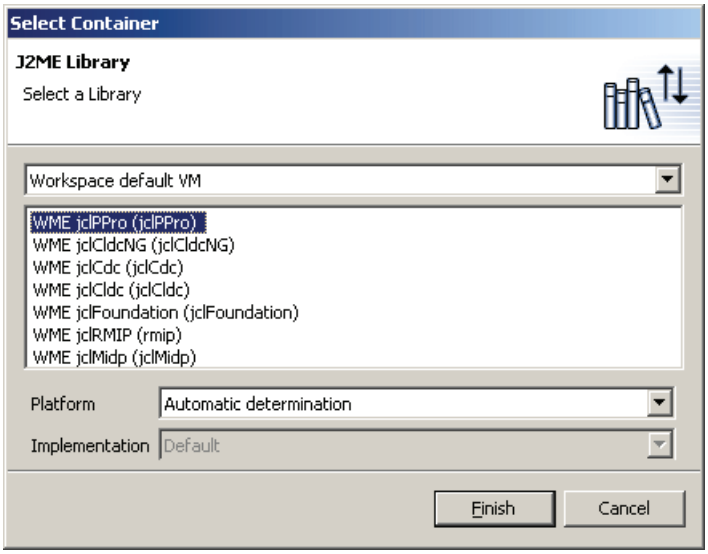


Figure 18-42 Choosing the J2ME PPro library

8. You see the WME PPro configuration added to the bootstrap classpath, as shown in Figure 18-43 on page 568. Click **Apply** to save the changes. Click **Close** to finish.

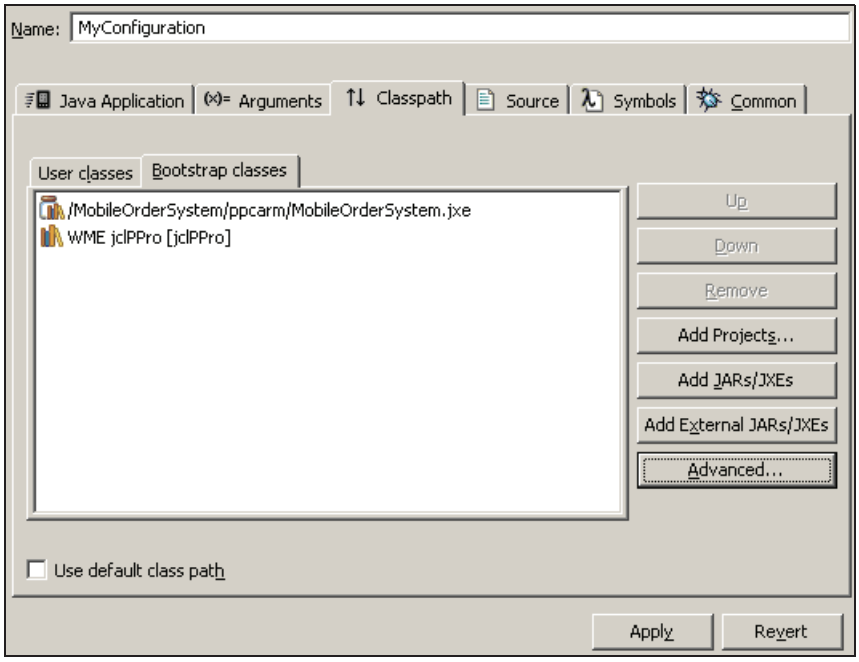


Figure 18-43 The bootstrap classes already configured

Note: There are several ways to configure the classpath to launch the application. Behind the scenes, the launch process deploys the application file on the device and creates a shortcut for it. A shortcut in the Pocket PC 2002 platform has a limit of 255 characters in length. For the sample application, the generated shortcut is like this:

```
227#"\j9\bin\j9.exe"
"-Xbootclasspath:\j9\lib\jclPPro\ppro-ui-win.zip;\j9\lib\jclFoundation\class
es.zip;\j9\lib\jclFoundation\locale.zip;\j9\lib\charconv.zip" "-cp"
"\j9\lib\db2ert.zip" "-jxe:\MobileOrderSystem\MobileOrderSystem.jxe"
```

If you have worked with the Foundation Configuration before, you probably knows that you can select the configuration base class library, by using the `-jcl:xxx` option. In this case, because the J2ME Personal Profile is based on the Foundation configuration, you may think that you can use the `-jcl:foun` option as the VM parameter. But you really can't do that, because the Launch process generates a -Xbootclasspath clause in the shortcut, as you had seen in the generated shortcut for the application. This overrides the -jcl clause, by setting the boot classpath in an absolute manner. For more information about this topic, refer to the WebSphere Studio Device Developer news group at `http://www.ibm.com/developerworks`.

18.4.5 Write the Java classes of the application

Now you have to write the classes of the application. You need to create each class using the New Class wizard, and then write the corresponding code.

MobileOrderSystem

This class has the main method used for start the application. The main method opens the database connection, using the path to the database, and shows the first AWT form, NewOrdersFrame in this case. Also, it defines the database path as a static constant.

The code for the class is shown in Example 18-1.

Example 18-1 MobileOrderSystem.java

```
package com.ibm.ral.itso.sv8030r.mos;

import java.sql.SQLException;

/**
 * Main class for Mobile Order System
 *
 */
public class MobileOrderSystem  {

   public final static String DB2E_DATABASE_PATH = "\\ordsysdb\\";

   public static void main(String[] args) throws SQLException {

      // Opens the connection to the database
      Database.getInstance().open(MobileOrderSystem.DB2E_DATABASE_PATH);

      // Shows the first frame of the application
      new NewOrdersFrame().show();
   }
}
```

Database

This class encapsulates the creation of the connection to the database. It uses the singleton pattern to generate an instance of the Database object for use where it is needed. For the sake of simplicity, we only use a connection for all database accesses.

The code for the class is shown in Example 18-2 on page 570.

Example 18-2 Database.java

```
package com.ibm.ral.itso.sv8030r.mos;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import com.ibm.db2e.jdbc.DB2eDriver;

/**
 * Encapsulate the access to the DB2e database
 *
 */
public class Database {

   // The database url prefix
   private static final String DB2E_DATABASE_URL_BASE = "jdbc:db2e:";
   // The driver class name
   private static final String DB2E_DRIVER_CLASS
      = "com.ibm.db2e.jdbc.DB2eDriver";

   private static Database instance;
   private Connection connection;

   /**
    * Constructor for Database
    */
   private Database() {
      try {
         // Loads the DB2e driver class
         Class.forName(DB2E_DRIVER_CLASS);
      }
      catch(Exception e) {
         e.printStackTrace();
      }
   }

   /**
    * Returns an instance of the Database object
    * .
    * @return Database
    */
   public static Database getInstance() {
      if (instance == null) {
         instance = new Database();
      }

      return instance;
```

```
    }

    /**
     * Returns the database connection
     *
     * @return Connection
     */
    public Connection getConnection() {
       return connection;
    }

    /**
     * Opens the database connection
     *
     * @param databasePath The DB2e database path
     * @throws SQLException
     */
    public void open(String databasePath) throws SQLException {
       this.connection
          = DriverManager.getConnection(DB2E_DATABASE_URL_BASE+databasePath);
    }

    /**
     * Close the database connection
     *
     * @throws SQLException
     */
    public void close() throws SQLException {
       if (connection != null) connection.close();
    }
}
```

NewOrdersFrame

This class shows a list with the orders for a sales representative. It has three possible actions: Edit an order, Delete an order, and Synchronize the database.

The list uses a PreparedStatement.executeQuery() to perform the SQL SELECT required to obtain the data. The Delete button use a PreparedStatement.executeUpdate() to perform the SQL UPDATE sentence for the order selected in the list.

The code for the class is shown in Example 18-3.

Example 18-3 NewOrdersFrame.java

```
package com.ibm.ral.itso.sv8030r.mos;

import java.awt.Button;
```

```
import java.awt.Dialog;
import java.awt.Frame;
import java.awt.List;
import java.awt.MenuBar;
import java.awt.Panel;
import java.awt.ScrollPane;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import java.util.Iterator;
import java.util.Vector;

/**
 * AWT Frame for New Orders
 *
 * Shows a list with the order for a salesman. Also it permits edit and delete
 * an order, and synchronize the database
 */
public class NewOrdersFrame extends Frame {

   private static int NO_ORDER_ID = -1;

   // Selects all the orders
   private static final String SQL_ALLORDERS
      = " SELECT ORDER_ID, ORDER_DATE, CUSTOMERS.NAME"
      +  "FROM ORDERS, CUSTOMERS "
      + " WHERE ORDERS.CUSTOMER_ID = CUSTOMERS.CUSTOMER_ID";

   // Delete an specific order
   private static final String SQL_DELETE_ORDER_BY_ID
      = "DELETE FROM ORDERS WHERE ORDER_ID = ?";


   // Visual control for the frame
   private Panel panel = null;
    private ScrollPane scrollPane = null;
    private Button editButton = null;
    private List list = null;
    private Button syncButton = null;
    private Button deleteButton = null;
    private MenuBar mainMenuBar = null;

    // Vector of orders keys
    private Vector orders = new Vector();
```

```
/**
 * Constructor for NewOrdersFrame
 *
 */
public NewOrdersFrame() {
   super();
   initialize();
}

/**
 * Initialize the AWT form
 *
 */
private void initialize() {
     this.add(getPanel(), java.awt.BorderLayout.SOUTH);
     this.add(getScrollPane(), java.awt.BorderLayout.CENTER);
     this.setMenuBar(getMainMenuBar());
     this.setTitle("New orders");
     initNewOrdersList();
}


/**
 * Initialize the menubar
 *
 * @return MenuBar
 */
private MenuBar getMainMenuBar() {
   if (mainMenuBar == null) {
      mainMenuBar = new MenuBar();
   }
   return mainMenuBar;
}

/**
 * Initializes the panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getPanel() {
   if(panel == null) {
      panel = new java.awt.Panel();
      panel.add(getSyncButton());
      panel.add(getEditButton());
      panel.add(getDeleteButton());
   }
   return panel;
}
/**
```

```
 * Initializes the scrollPane
 *
 * @return java.awt.ScrollPane
 */
private java.awt.ScrollPane getScrollPane() {
   if(scrollPane == null) {
      scrollPane = new java.awt.ScrollPane();
      scrollPane.add(getList(), null);
   }
   return scrollPane;
}

/**
 * Initializes the Edit button
 *
 * @return java.awt.Button
 */
private java.awt.Button getEditButton() {
   if(editButton == null) {
      editButton = new java.awt.Button();
      editButton.setLabel("Edit");
      editButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            editCommand();
         }
      });
   }
   return editButton;
}

/**
 * Initializes the orders list
 *
 * @return java.awt.List
 */
private java.awt.List getList() {
   if(list == null) {
      list = new java.awt.List();
   }
   return list;
}

/**
 * Initializes the Synchronize button
 *
 * @return java.awt.Button
 */
private java.awt.Button getSyncButton() {
   if(syncButton == null) {
```

```
            syncButton = new java.awt.Button();
            syncButton.setLabel("Synchronize");
            syncButton.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent event) {
                  syncCommand();
               }
            });
         }
         return syncButton;
      }

      /**
       * Initializes the Delete button
       *
       * @return java.awt.Button
       */
      private java.awt.Button getDeleteButton() {
         if(deleteButton == null) {
            deleteButton = new java.awt.Button();
            deleteButton.setLabel("Delete");
            deleteButton.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent event) {
                  deleteCommand();
               }
            });
         }
         return deleteButton;
      }

      /**
       * Obtains the data for the list
       */
      private void initNewOrdersList() {

         Connection conn = null;
         try {
            conn = Database.getInstance().getConnection();

            PreparedStatement pstm = conn.prepareStatement(SQL_ALLORDERS);
            ResultSet rs = pstm.executeQuery();

            while (rs.next()) {
               orders.add(new Integer(rs.getInt(1)));
               getList().add(rs.getDate(2)+" "+rs.getString(3));
            }

            rs.close();
            pstm.close();
         }
```

```
   catch(SQLException e) {
      e.printStackTrace();
   }
}

/**
 * Closes this frame and shows the Sync frame
 */
private void syncCommand() {
   // Closes this form
   this.setVisible(false);
   this.dispose();

   // Shows the Sync form
   new SyncFrame().show();
}

/**
 * Close this frame and shows the Edit frame with the order currently
 * selected on the list. If there's no order selected, it does nothing.
 *
 */
private void editCommand() {
   int orderId = getSelectedOrderId();
   if (orderId == NO_ORDER_ID) return;

   this.setVisible(false);
   this.dispose();
   new EditOrderFrame(orderId).show();
}

/**
 * Deletes the order currently selected on the list. If there's no order
 * selected, it does nothing.
 */
private void deleteCommand() {

   int order_id = getSelectedOrderId();

   if (order_id == NO_ORDER_ID) return;

   try {
      PreparedStatement pstm
         = Database.getInstance().getConnection()
                        .prepareStatement(SQL_DELETE_ORDER_BY_ID);
      pstm.setInt(1, order_id);
      int rowsDeleted = pstm.executeUpdate();
      if (rowsDeleted == 1) {
         orders.remove(getList().getSelectedIndex());
```

```
            getList().remove(getList().getSelectedIndex());
         }
      }
      catch(SQLException e) {
         e.printStackTrace();
      }
   }

   /**
    * Looks for the order_id of the currently selected order on the list
    *
    * @return int The order_id
    */
   private int getSelectedOrderId() {
      int code = NO_ORDER_ID;
      int selectedIndex = getList().getSelectedIndex();

      if (selectedIndex >= 0) {
         code = ((Integer)orders.get(selectedIndex)).intValue();

      }

      return code;
   }
}
```

EditOrderFrame

This class shows a frame for edit an order, previously selected in the New Orders frame. You can edit the Rebate and Contact fields and use the Update order button to reflect the changes in the database. The frame also shows a list with the items for the current order.

The code for the class is shown in Example 18-4.

Example 18-4 EditOrderFrame.java

```
package com.ibm.ral.itso.sv8030r.mos;

import java.awt.Frame;
import java.awt.MenuBar;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.math.BigDecimal;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.DecimalFormat;
import java.text.NumberFormat;
```

```
/**
 * AWT Frame for Edit and order
 *
 */
public class EditOrderFrame extends Frame {

   private static final String SQL_ORDER_BYID
      = " SELECT ORDER_DATE, CUSTOMERS.NAME, REBATE, CONTACT"
      + " FROM ORDERS, CUSTOMERS "
      + " WHERE ORDERS.CUSTOMER_ID = CUSTOMERS.CUSTOMER_ID "
      + " AND ORDER_ID = ?";

   private static final String SQL_ORDERITEMS_BYID
      = " SELECT PRODUCTS.DESCRIPTION, QUANTITY, ORDER_ITEMS.PRICE, "
      + " QUANTITY * ORDER_ITEMS.PRICE "
      + " FROM ORDER_ITEMS, PRODUCTS "
      + " WHERE ORDER_ITEMS.PRODUCT_ID = PRODUCTS.PRODUCT_ID "
      + " AND ORDER_ITEMS.ORDER_ID = ?";

   private static final String SQL_UPDATE_ORDER
      = " UPDATE ORDERS SET REBATE = ?, CONTACT = ? WHERE ORDER_ID =?";

   // Order ID for the editing order
   private int orderId;

   // Visual controls
   private java.awt.Panel buttonPanel = null;
   private java.awt.Panel mainPanel = null;
   private java.awt.Panel fieldsPanel = null;
   private java.awt.ScrollPane orderItemScrollpane = null;
   private java.awt.List itemList = null;
   private java.awt.TextField customerNameField = null;
   private java.awt.TextField orderDateField = null;
   private java.awt.TextField orderIdField = null;
   private java.awt.Panel rebatePanel = null;
   private java.awt.Panel contactPanel = null;
   private java.awt.Label label = null;
   private java.awt.Label label1 = null;
   private java.awt.TextField rebateField = null;
   private java.awt.TextField contactField = null;
   private java.awt.Panel itemsPanel = null;
   private java.awt.Label label2 = null;
   private java.awt.Button updateButton = null;
   private java.awt.Button backButton = null;
   private MenuBar mainMenuBar = null;

   /**
    * Constructor for EditOrderFrame
```

```
 *
 */
public EditOrderFrame(int orderId) {
   super();
   this.orderId = orderId;
   initialize();
}

/**
 * Initializes the frame
 *
 */
private void initialize() {
     this.add(getButtonPanel(), java.awt.BorderLayout.SOUTH);
     this.add(getMainPanel(), java.awt.BorderLayout.CENTER);
     this.setMenuBar(getMainMenuBar());
     this.setTitle("Editing order");

}

/**
 * Initializes the main menubar
 *
 * @return MenuBar
 */
private MenuBar getMainMenuBar() {
   if (mainMenuBar == null) {
      mainMenuBar = new MenuBar();
   }
   return mainMenuBar;
}

/**
 * Initializes the button panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getButtonPanel() {
   if(buttonPanel == null) {
      buttonPanel = new java.awt.Panel();
      java.awt.FlowLayout layFlowLayout7 = new java.awt.FlowLayout();
      layFlowLayout7.setAlignment(java.awt.FlowLayout.LEFT);
      buttonPanel.setLayout(layFlowLayout7);
      buttonPanel.add(getUpdateButton(), null);
      buttonPanel.add(getBackButton(), null);
      buttonPanel.setName("buttonPanel");
   }
   return buttonPanel;
}
```

```
/**
 * Initializes the main panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getMainPanel() {
   if(mainPanel == null) {
      mainPanel = new java.awt.Panel();
      java.awt.GridLayout layGridLayout2 = new java.awt.GridLayout();
      layGridLayout2.setRows(2);
      layGridLayout2.setColumns(1);
      mainPanel.setLayout(layGridLayout2);
      mainPanel.add(getFieldsPanel(), null);
      mainPanel.add(getItemsPanel(), null);
      mainPanel.setName("mainPanel");
   }
   return mainPanel;
}

/**
 * Initializes the order fields panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getFieldsPanel() {
   if(fieldsPanel == null) {
      fieldsPanel = new java.awt.Panel();
      java.awt.GridLayout layGridLayout3 = new java.awt.GridLayout();
      layGridLayout3.setRows(5);
      layGridLayout3.setColumns(1);
      fieldsPanel.setLayout(layGridLayout3);
      fieldsPanel.add(getOrderIdField(), null);
      fieldsPanel.add(getOrderDateField(), null);
      fieldsPanel.add(getCustomerNameField(), null);
      fieldsPanel.add(getRebatePanel(), null);
      fieldsPanel.add(getContactPanel(), null);
      initFields();
   }
   return fieldsPanel;
}

/**
 * Initializes the list scrollPane
 *
 * @return java.awt.ScrollPane
 */
private java.awt.ScrollPane getOrderItemScrollpane() {
   if(orderItemScrollpane == null) {
```

```
        orderItemScrollpane = new java.awt.ScrollPane();
        orderItemScrollpane.add(getItemList(), null);
    }
    return orderItemScrollpane;
}

/**
 * Initializes the list
 *
 * @return java.awt.List
 */
private java.awt.List getItemList() {
    if(itemList == null) {
        itemList = new java.awt.List();
        initItemList();
    }
    return itemList;
}

/**
 * Initializes the customer name textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getCustomerNameField() {
    if(customerNameField == null) {
        customerNameField = new java.awt.TextField();
        customerNameField.setFont(new java.awt.Font("sansserif", 1, 12));
        customerNameField.setEditable(false);
    }
    return customerNameField;
}

/**
 * Initializes the order date textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getOrderDateField() {
    if(orderDateField == null) {
        orderDateField = new java.awt.TextField();
        orderDateField.setFont(new java.awt.Font("sansserif", 1, 12));
        orderDateField.setEditable(false);
    }
    return orderDateField;
}

/**
 * Initializes the order id textfield
```

```
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getOrderIdField() {
   if(orderIdField == null) {
      orderIdField = new java.awt.TextField();
      orderIdField.setFont(new java.awt.Font("sansserif", 1, 12));
      orderIdField.setEditable(false);
   }
   return orderIdField;
}

/**
 * Initializes the rebate panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getRebatePanel() {
   if(rebatePanel == null) {
      rebatePanel = new java.awt.Panel();
      java.awt.FlowLayout layFlowLayout19 = new java.awt.FlowLayout();
      layFlowLayout19.setHgap(0);
      layFlowLayout19.setVgap(0);
      layFlowLayout19.setAlignment(java.awt.FlowLayout.LEFT);
      rebatePanel.setLayout(layFlowLayout19);
      rebatePanel.add(getLabel(), null);
      rebatePanel.add(getRebateField(), null);
   }
   return rebatePanel;
}

/**
 * Initializes the contact panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getContactPanel() {
   if(contactPanel == null) {
      contactPanel = new java.awt.Panel();
      java.awt.FlowLayout layFlowLayout18 = new java.awt.FlowLayout();
      layFlowLayout18.setHgap(0);
      layFlowLayout18.setVgap(0);
      layFlowLayout18.setAlignment(java.awt.FlowLayout.LEFT);
      contactPanel.setLayout(layFlowLayout18);
      contactPanel.add(getLabel1(), null);
      contactPanel.add(getContactField(), null);
   }
   return contactPanel;
}
```

```
/**
 * Initializes a label
 *
 * @return java.awt.Label
 */
private java.awt.Label getLabel() {
   if(label == null) {
      label = new java.awt.Label();
      label.setText("Rebate: ");
   }
   return label;
}

/**
 * Initializes a label
 *
 * @return java.awt.Label
 */
private java.awt.Label getLabel1() {
   if(label1 == null) {
      label1 = new java.awt.Label();
      label1.setText("Contact: ");
   }
   return label1;
}

/**
 * Initializes the rebate textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getRebateField() {
   if(rebateField == null) {
      rebateField = new java.awt.TextField();
      rebateField.setColumns(10);
   }
   return rebateField;
}

/**
 * Initializes the contact textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getContactField() {
   if(contactField == null) {
      contactField = new java.awt.TextField();
      contactField.setColumns(18);
```

```
   }
   return contactField;
}

/**
 * Initializes the items panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getItemsPanel() {
   if(itemsPanel == null) {
      itemsPanel = new java.awt.Panel();
      java.awt.BorderLayout layBorderLayout6
         = new java.awt.BorderLayout();
      itemsPanel.setLayout(layBorderLayout6);
      itemsPanel.add(getLabel2(), java.awt.BorderLayout.NORTH);
      itemsPanel.add(getOrderItemScrollpane(),
         java.awt.BorderLayout.CENTER);
   }
   return itemsPanel;
}

/**
 * Initializes a label2
 *
 * @return java.awt.Label
 */
private java.awt.Label getLabel2() {
   if(label2 == null) {
      label2 = new java.awt.Label();
      label2.setText("Items");
      label2.setName("Items");
      label2.setFont(new java.awt.Font("sansserif", 1, 12));
   }
   return label2;
}

/**
 * Initializes the update button
 *
 * @return java.awt.Button
 */
private java.awt.Button getUpdateButton() {
   if(updateButton == null) {
      updateButton = new java.awt.Button();
      updateButton.setLabel("Update order");
      updateButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            updateCommand();
```

```
            }
        });
    }
    return updateButton;
}

/**
 * Initializes the Back button
 *
 * @return java.awt.Button
 */
private java.awt.Button getBackButton() {
    if(backButton == null) {
        backButton = new java.awt.Button();
        backButton.setLabel("Back");
        backButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                backCommand();
            }
        });
    }
    return backButton;
}


/**
 * Initializes the order fields
 *
 * This method execute a SQL SELECT to bring the data for the order id
 * given in the constructor.
 */
private void initFields() {

    getOrderIdField().setText(String.valueOf(this.orderId));

    try {
        PreparedStatement pstm
            = Database.getInstance().getConnection()
                              .prepareStatement(SQL_ORDER_BYID);
        pstm.setInt(1, this.orderId);
        ResultSet rs = pstm.executeQuery();
        if (rs.next()) {
            getOrderDateField().setText(rs.getDate(1).toString());
            getCustomerNameField().setText(rs.getString(2));


            BigDecimal rebate = rs.getBigDecimal(3);
            if (!rs.wasNull()) {
                getRebateField().setText(
```

```
                String.valueOf(rebate.doubleValue()));
            }

            String contact = rs.getString(4);
            if (!rs.wasNull()) {
               getContactField().setText(contact);
            }
         }
         rs.close();
         pstm.close();
      }
      catch (SQLException e) {
         e.printStackTrace();
         System.err.println(e.getSQLState());
      }
   }

   /**
    * Initializes the item list
    *
    * This method execute a SQL SELECT to bring the data for the order id
    * given in the constructor. Please note that we have to use the BigDecimal
    * class instead the double primitive, because the ResultSet.getDouble()
    * method is not currently supported on the DB2e JDBC implementation.
    *
    */
   private void initItemList() {

      NumberFormat numFormat = DecimalFormat.getInstance();
      numFormat.setMaximumFractionDigits(2);
      numFormat.setMinimumFractionDigits(2);

      try {
         PreparedStatement pstm
            = Database.getInstance().getConnection()
                              .prepareStatement(SQL_ORDERITEMS_BYID);
         pstm.setInt(1, this.orderId);
         ResultSet rs = pstm.executeQuery();
         while(rs.next()) {
            StringBuffer sb = new StringBuffer();
            sb.append(rs.getString(1)).append(" ")
               .append(rs.getInt(2)).append(" ")
               .append(numFormat.format(rs.getBigDecimal(3).doubleValue()))
               .append(" ")
               .append(numFormat.format(rs.getBigDecimal(4).doubleValue()));

            getItemList().add(sb.toString());
         }
         rs.close();
```

```
            pstm.close();
        }
        catch (SQLException e) {
            e.printStackTrace();
            System.err.println(e.getSQLState());
        }
    }

    /**
     * Handles the action for the Back button
     *
     * Closes this frame and shows the NewOrders Frame
     */
    private void backCommand() {
        this.setVisible(false);
        this.dispose();
        new NewOrdersFrame().show();
    }

    /**
     * Handles the action for the Update order button.
     *
     * This method takes the values from the rebate and contact fields and
     * execute an SQL UPDATE to reflect the changes in the database
     */
    private void updateCommand() {

        try {
            PreparedStatement pstm
                = Database.getInstance().getConnection()
                                .prepareStatement(SQL_UPDATE_ORDER);
            pstm.setInt(3, this.orderId);
            pstm.setBigDecimal(1, new BigDecimal(getRebateField().getText()));
            pstm.setString(2, getContactField().getText());

            pstm.executeUpdate();

            pstm.close();
        }
        catch(SQLException e) {
            e.printStackTrace();
            System.err.println(e.getSQLState());
        }
    }
}
```

SyncFrame

The SyncFrame class shows you how to use the Sync API for Java to develop a customized synchronization client.

The frame has five text fields to enter the synchronization information. When you click the **Sync** button, the sync() method uses this information to connect to the SyncServer and perform the synchronization. The code for the class is shown in Example 18-5.

Note that you have to close the database connection prior to the database synchronization. Refer to the *DB2e Application Development Guide Version 8 Release 1*, SC18-7185, for more information about the Java Sync API.

An HTTP-based Synchronization API for Pocket PC 2002 devices doesn't need to specify the synchronization information, but it uses the information configured in the Everyplace Client. See Chapter 29, "Everyplace Client API for Pocket PC 2002" on page 1057 for more information.

Example 18-5 SyncFrame.java

```
package com.ibm.ral.itso.sv8030r.mos;

import java.awt.Button;
import java.awt.Frame;
import java.awt.Label;
import java.awt.MenuBar;
import java.awt.Panel;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;

import com.ibm.mobileservices.isync.ISync;
import com.ibm.mobileservices.isync.ISyncConfigStore;
import com.ibm.mobileservices.isync.ISyncDriver;
import com.ibm.mobileservices.isync.ISyncException;
import com.ibm.mobileservices.isync.ISyncProvider;
import com.ibm.mobileservices.isync.ISyncService;
import com.ibm.mobileservices.isync.ISyncSubscriptionSet;
import com.ibm.mobileservices.isync.db2e.jni.DB2eISyncProvider;
import com.ibm.mobileservices.isync.event.ISyncEvent;
import com.ibm.mobileservices.isync.event.ISyncListener;

/**
 * AWT Frame for synchronization
 *
 * Implements an interface for synchronize the database. It uses the Java API
 * for Synchronization.
```

```
 *
 */
public class SyncFrame extends Frame implements ISyncListener {

     // Frame visual controls
     private Panel buttonPanel = null;
     private Button syncButton = null;
     private Panel mainPanel = null;
     private Label label = null;
     private TextField hostNameField = null;
     private TextField hostPortField = null;
     private Label label1 = null;
     private TextField userNameField = null;
     private Label label2 = null;
     private TextField userPasswordField = null;
     private TextField messagesField = null;
     private TextField progressField = null;
     private Button backButton = null;
     private MenuBar mainMenuBar = null;

   /**
    * Constructor for SyncFrame
    *
    */
   public SyncFrame() {
      super();
      initialize();
   }
   /**
    * Initializes the frame
    *
    */
   private void initialize() {
        this.add(getButtonPanel(), java.awt.BorderLayout.SOUTH);
        this.add(getMainPanel(), java.awt.BorderLayout.CENTER);
        this.setMenuBar(getMainMenuBar());
        this.setTitle("Sync Orders");

   }

   private MenuBar getMainMenuBar() {
      if (mainMenuBar == null) {
         mainMenuBar = new MenuBar();
      }
      return mainMenuBar;
   }

   /**
    * Initializes the button panel
```

```
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getButtonPanel() {
   if(buttonPanel == null) {
      buttonPanel = new java.awt.Panel();
      buttonPanel.add(getSyncButton(), null);
      buttonPanel.add(getBackButton(), null);
   }
   return buttonPanel;
}

/**
 * Initializes the Sync button
 *
 * @return java.awt.Button
 */
private java.awt.Button getSyncButton() {
   if(syncButton == null) {
      syncButton = new java.awt.Button();
      syncButton.setLabel("Synchronize");
      syncButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            syncCommand();
         }
      });
   }
   return syncButton;
}

/**
 * Initializes the main panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getMainPanel() {
   if(mainPanel == null) {
      mainPanel = new java.awt.Panel();
      java.awt.FlowLayout layFlowLayout1 = new java.awt.FlowLayout();
      layFlowLayout1.setAlignment(java.awt.FlowLayout.LEFT);
      mainPanel.setLayout(layFlowLayout1);
      mainPanel.add(getLabel(), null);
      mainPanel.add(getHostNameField(), null);
      mainPanel.add(getHostPortField(), null);
      mainPanel.add(getLabel1(), null);
      mainPanel.add(getUserNameField(), null);
      mainPanel.add(getLabel2(), null);
      mainPanel.add(getUserPasswordField(), null);
      mainPanel.add(getMessagesField(), null);
```

```
            mainPanel.add(getProgressField(), null);
        }
        return mainPanel;
    }

    /**
     * Initializes a label
     *
     * @return java.awt.Label
     */
    private java.awt.Label getLabel() {
        if(label == null) {
            label = new java.awt.Label();
            label.setText("Server: ");
            label.setFont(new java.awt.Font("sansserif", 1, 12));
        }
        return label;
    }

    /**
     * Initializes the hostname textfield
     *
     * @return java.awt.TextField
     */
    private java.awt.TextField getHostNameField() {
        if(hostNameField == null) {
            hostNameField = new java.awt.TextField();
            hostNameField.setColumns(14);
        }
        return hostNameField;
    }

    /**
     * Initializes the host port textfield
     *
     * @return java.awt.TextField
     */
    private java.awt.TextField getHostPortField() {
        if(hostPortField == null) {
            hostPortField = new java.awt.TextField();
            hostPortField.setColumns(3);
        }
        return hostPortField;
    }

    /**
     * Initializes a label
     *
     * @return java.awt.Label
```

```
 */
private java.awt.Label getLabel1() {
   if(label1 == null) {
      label1 = new java.awt.Label();
      label1.setText("User: ");
      label1.setFont(new java.awt.Font("sansserif", 1, 12));
   }
   return label1;
}

/**
 * Initializes the username textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getUserNameField() {
   if(userNameField == null) {
      userNameField = new java.awt.TextField();
      userNameField.setColumns(15);
   }
   return userNameField;
}

/**
 * Initializes a label
 *
 * @return java.awt.Label
 */
private java.awt.Label getLabel2() {
   if(label2 == null) {
      label2 = new java.awt.Label();
      label2.setText("Password: ");
      label2.setName("");
      label2.setFont(new java.awt.Font("sansserif", 1, 12));
   }
   return label2;
}

/**
 * Initializes the user's password textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getUserPasswordField() {
   if(userPasswordField == null) {
      userPasswordField = new java.awt.TextField();
      userPasswordField.setEchoChar('*');
      userPasswordField.setColumns(15);
   }
```

```
   return userPasswordField;
}

/**
 * Initializes the messages textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getMessagesField() {
   if(messagesField == null) {
      messagesField = new java.awt.TextField();
      messagesField.setColumns(28);
      messagesField.setEnabled(false);
   }
   return messagesField;
}

/**
 * Initializes the progress textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getProgressField() {
   if(progressField == null) {
      progressField = new java.awt.TextField();
      progressField.setEnabled(false);
      progressField.setColumns(15);
   }
   return progressField;
}

/**
 * Initializes the Back button
 *
 * @return java.awt.Button
 */
private java.awt.Button getBackButton() {
   if(backButton == null) {
      backButton = new java.awt.Button();
      backButton.setLabel("Back");
      backButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            backCommand();
         }
      });
   }
   return backButton;
}
```

```
/**
 * Starts the synchronization.
 *
 * This method has to close the database connection first. Also it disable
 * all the controls while the synchronization is active. It enables the
 * controls and opens the database when the synchronization end.
 */
private void syncCommand() {
   getHostNameField().setEnabled(false);
   getHostPortField().setEnabled(false);
   getUserNameField().setEnabled(false);
   getUserPasswordField().setEnabled(false);
   getSyncButton().setEnabled(false);
   getBackButton().setEnabled(false);

   try {
      Database.getInstance().close();

      sync(getHostNameField().getText(), getHostPortField().getText(),
         getUserNameField().getText(), getUserPasswordField().getText());

      Database.getInstance().open(MobileOrderSystem.DB2E_DATABASE_PATH);
   }
   catch(SQLException e) {
      e.printStackTrace();
      System.err.println(e.getSQLState());
   }
   getHostNameField().setEnabled(true);
   getHostPortField().setEnabled(true);
   getUserNameField().setEnabled(true);
   getUserPasswordField().setEnabled(true);
   getSyncButton().setEnabled(true);
   getBackButton().setEnabled(true);
}

/**
 * Closes the frame and opens the NewOrders frame.
 */
private void backCommand() {
   this.setVisible(false);
   this.dispose();
   new NewOrdersFrame().show();
}

/**
 * Event handler for synchronization
 *
 * This is a callback method that the sync engine use to communicate what
 * is happening with the synchronization process. You can use the message
```

```
 * codes to produce more readables messages.
 *
 * @see com.ibm.mobileservices.isync.event.ISyncListener#eventIssued(
 * ISyncEvent)
 */
public int eventIssued(ISyncEvent evt ) {
   int evtType = evt.getEventType();
   switch(evtType) {
      // display event status
      case ISync.EVTTYPE_INFO:
      case ISync.EVTTYPE_ERROR:

         getMessagesField().setText(
            "EventType:"+evtType+". EventCode:"+evt.getEventCode());
         System.out.println ("Event Type: " + evtType );
         System.out.println ("Event Code: " + evt.getEventCode() );
         getProgressField().setText(
            "Progress: " + evt.getSyncProgress());
         System.out.println ("Progress: " + evt.getSyncProgress());

         return ISync.RTNCB_DONE;
      case ISync.EVTTYPE_RETRY:
         return ISync.RTNCB_REPLY_YES;
      case ISync.EVTTYPE_CONFLICT:
         return ISync.RTNCB_DONE;
      // ignore other event types
      default:
      break;
   }
   // let sync engine take default action
   return ISync.RTNCB_DEFAULT ;
}

/**
 * The worker synchronization method
 * .
 * @param host The DB2e SyncServer hostname or IP
 * @param port The port where the sync engine is listening
 * @param user
 * @param password
 */
private void sync(String host, String port, String user, String password) {

   ISyncProvider provider = null;
   ISyncService service = null;
   ISyncConfigStore config = null;
   ISyncDriver syncer = null;
   String path = MobileOrderSystem.DB2E_DATABASE_PATH;
   ISyncSubscriptionSet ssArr[] = null;
```

```
int rc = 0;
try {
   // Get an instance DB2eISyncProvider
   provider = DB2eISyncProvider.getInstance();

   // Get an instance of synchronization service from the provider
   service = provider.createSyncService(host, port, user, password);

   // Get an instance of the configuration store
   config = service.getConfigStore(path);

   // Get an instance of the sync driver to perform
   // synchronization
   syncer = config.getSyncDriver();

   // Set the listener object for event notification from the
   // syncer object during synchronization
   syncer.setSyncListener(this);

   // Perform synchronization on all enabled subscription sets
   rc = syncer.sync();
   switch (rc) {
      case ISync.RTN_SUCCEEDED:
         getMessagesField().setText("Synchronization succeeded");
         break;
      case ISync.RTN_CANCELED:
         getMessagesField().setText("Synchronization canceled");
         break;
      default:
         getMessagesField().setText("Synchronization failed");
      break;
   }

   ssArr = config.getSubscriptionSets();
   for (int i=0 ; i < ssArr.length; i++ ) {
      System.out.print ("Subscription Set: " +ssArr[i].getName()
         + " Status: ");
      switch(ssArr[i].getStatus()) {
         case ISync.STATUS_READY:
            System.out.println("READY");
            break;
         case ISync.STATUS_COMPLETED:
            System.out.println ("COMPLETED");
            break;
         case ISync.STATUS_CANCELED:
            System.out.println ("CANCELED");
            break;
         default:
            System.out.println ("FAILED");
```

```
                    break;
                }
            }
        }
        catch (ISyncException ie) {
            getMessagesField().setText(
                   "An exception has ocurred. Code:"+ie.getCode());
            ie.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        finally {
            // Close and free all allocated resources
            //
            try {
                if (syncer != null) {
                    syncer.close();
                    syncer = null;
                }
                if (config != null) {
                    config.close();
                    config = null;
                }
                if (service != null) {
                    service.close();
                    service = null;
                }
            }
            catch(ISyncException ie2) {
                getMessagesField().setText(
                    "An exception has ocurred. Code:"+ie2.getCode());
                ie2.printStackTrace();
            }
        }
    } // end

}
```

18.4.6 Test the application on the device

Once you have all the classes coded without compilation errors, you can launch the application to your Pocket PC device.

Important: You must perform an initial synchronization between the device and the server using the DB2e ISync client to initialize the Db2e database. The directory for the database's files accord to the code must be \ordsysdb. Refer to Chapter 20, "Synchronizing with DB2 databases" on page 645 for more information about the Db2e Sync process. In a real application you should put some initialization code in the program or a call to the synchronization module if the database has not been initialized.

To test the application follow these steps:

1. Make sure that the ActiveSync communication link between your desktop and your Pocket PC is enabled.
2. On WebSphere Studio Device Developer, select **Run -> Run** from the menu bar. The Launch configuration window opens. Select **Device Java Application -> MyConfiguration** from the tree view. Click **Run** to launch the application, as shown in Figure 18-44.

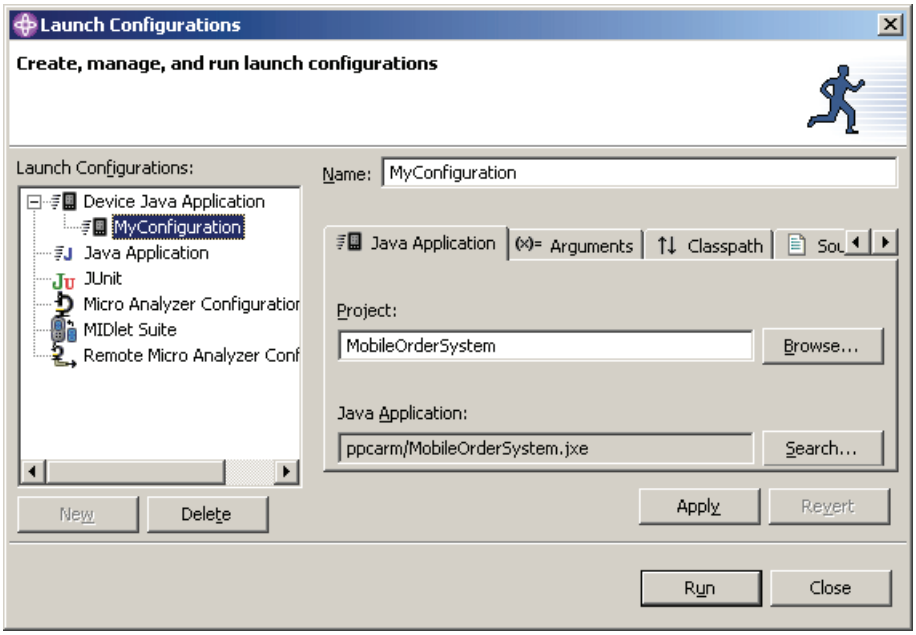


Figure 18-44 Launching the application

3. Wait while WebSphere Studio Device Developer links and launches your application. A window may warn you that certain methods were not found, as shown in Figure 18-45 on page 599. If you see this window, you can safely ignore it. Just click **OK**.

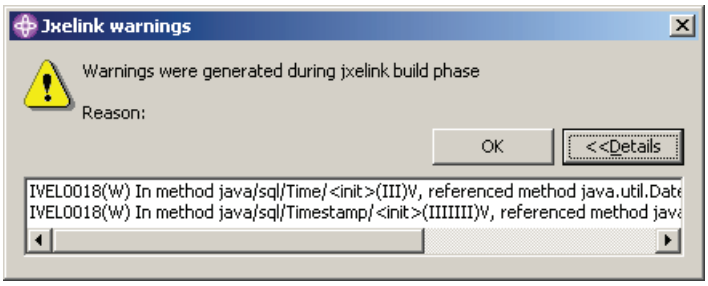


Figure 18-45 SmartLink warning message

Tip: If you experience some problem connecting to the Pocket PC via ActiveSync when the launch process for Pocket PC copies files to the device, try restarting the workbench.

Now you can see on your Pocket PC device the New Orders frame, as shown in Figure 18-46.

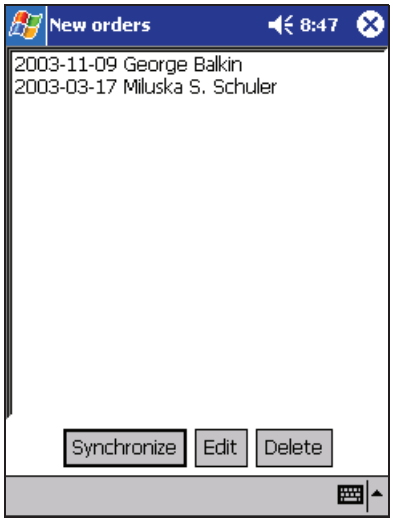


Figure 18-46 New orders

If you select an order and click **Delete,** the order is deleted from the database and, of course, from the list. If you select an order and click **Edit**, you can see the Edit Order frame as shown in Figure 18-47 on page 600.

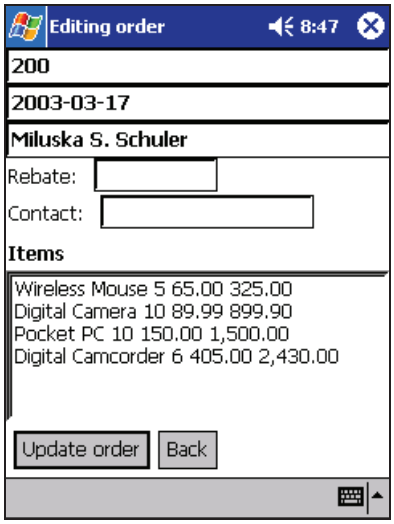


Figure 18-47 Edit order

Enter some values for the Contact and Rebate fields. Consider that the sample application doesn't have any validation code, so enter meaningful values. Click **Update order**, and the order is updated on the database. Click **Back** to return to the New orders frame.

On the New Orders frame, click **Synchronize** to show the Sync Frame. Enter your SyncServer host name, port, user name and password, as shown in Figure 18-48 on page 601.

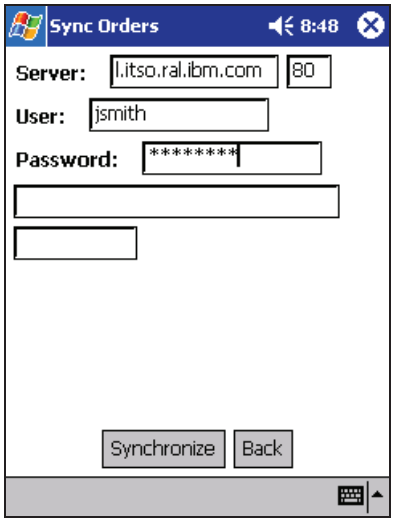


Figure 18-48 Sync orders

Click **Sync** to start the synchronization. You see the synchronization messages and the progress in their respective text fields. When the synchronization ends, you see a message that indicate the result of the synchronization, as shown in Figure 18-49.

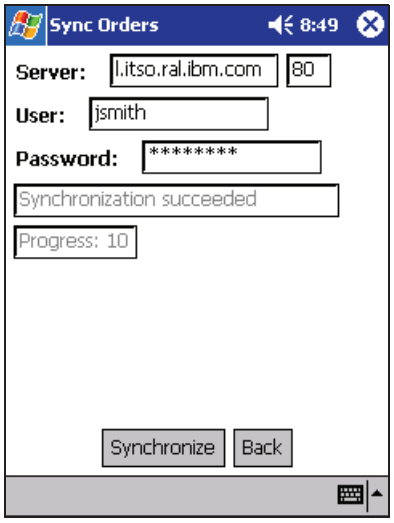


Figure 18-49 Sync ended

19

DB2 Everyplace applications with Mobile Application Builder (MAB)

This chapter describes the use of Mobile Application Builder (MAB) to develop data-oriented applications in a Rapid Application Development (RAD) environment with the synchronization capabilities that WebSphere Everyplace Access DB2 Everyplace provides.

In this chapter, we show you how to install and configure a development environment for MAB and how to develop and test applications for Palm and Pocket PC targets.

© Copyright IBM Corp. 2003. All rights reserved.

19.1 Overview

DB2 Everyplace (DB2E) comes with a Rapid Application Development tool named Mobile Application Builder (MAB). This tool provides an easy-to-use platform for developing data-oriented applications for PDA devices.

Mobile Application Builder supports:

- The Palm OS device platform. GNU-compatible C code is generated for the device.
- Mobile Device platforms that support a Java Virtual Machine, including support for the Sun PersonalJava API (known now as the Connected Device Configuration, Personal Profile - see http://java.sun.com/j2me/). Java code is generated for the supported platforms, which include devices based on Windows CE/Pocket PC, Symbian OS V6, Symbian OS V7, Sharp Zaurus, and PalmOS 5. A StrongARM or XScale processor-based device required.
- Visual construction of forms for different devices
- DB2 Everyplace database on the device
- Scripting capabilities for user-defined logic
- Double-byte character sets
- Integration with other tools for application testing and debugging

19.2 Installation

Install the Mobile Application Builder on the machine where you want to develop applications following these steps:

1. Run the DB2E SDK Installation program, \DB2e\DB2Everyplace.exe, located on the DB2e Mobile Application Builder V8.1.2 CD.
2. Click **Next** to start the installation.

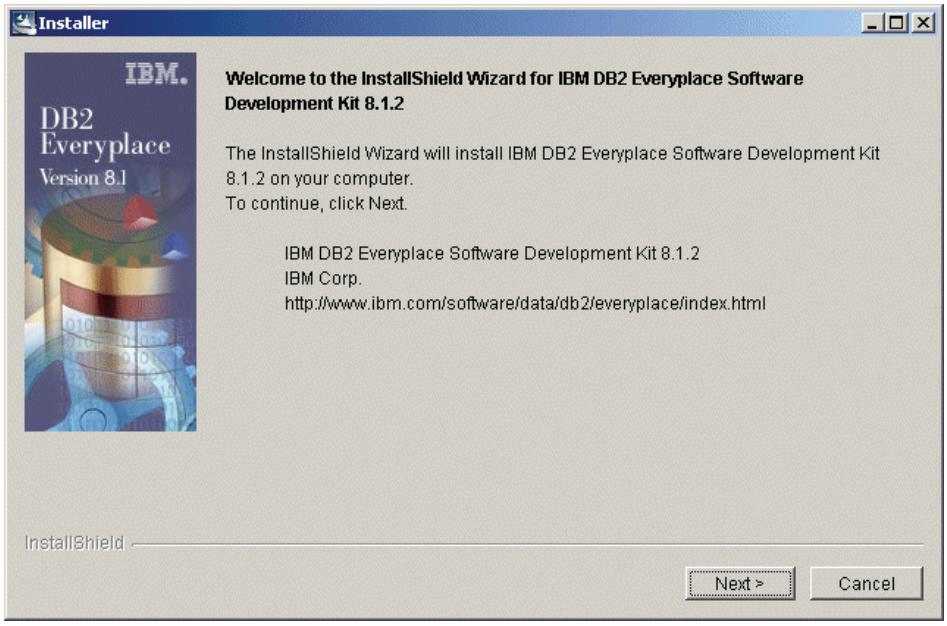


Figure 19-1 DB2E SDK installation program

3. Follow the instructions provided on the installation program. At the end you will see a window as shown in Figure 19-2 on page 606. Click **Finish**.

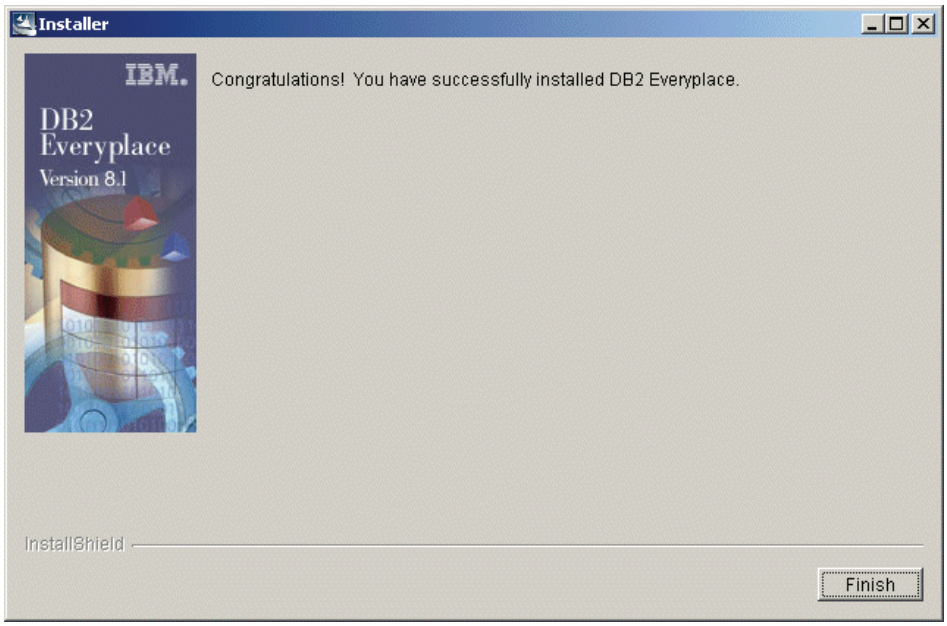


Figure 19-2 DB2E SDK installation program

In addition to the MAB, DB2E SDK installs the DB2E database component to deploy on the supported devices and also several components necessary for the application development on mobile devices.

19.3 Required components for Palm development

For Palm development, you will need to install the following additional components:

- Cygwin Version B-20
- Prc-Tools Version 2.0
- Palm SDK Version 4.0
- Pilrc Version 2.8
- Palm Desktop (for install applications on the device)

Note: The Palm SDK V5.0 is not supported to work with MAB. However, since MAB generates C code for Palm, the application file generated should be compatible with Palm OS V5.

To install these components follow these steps:

1. Install Cygwin by running the cygwin-b20.1-full.exe located in <DB2eSDK-installation-path>\SDK\ApplicationBuilder\Toolkits\Palm Development\MAB_prereqs.
2. Install the GNU PRC-Tools (prc-tools2.0.exe) located in <DB2eSDK-installation-path>\SDK\ApplicationBuilder\Toolkits\Palm Development\MAB_prereqs. This is the GCC Tool Chain for the Palm OS and it includes the cross compiler and other utilities. MAB only works with Version 2.0 of the GNU PRC-Tools. If you have a previous version of the GNU PRC-Tools already installed, you need to remove it in order to avoid conflicts among similar applications.
3. Install the Palm SDK 4.0. Download the SDK from http://www.palmos.com/dev/tools/sdk.You need to install the files to a directory on your machine. Preferably this would be the same directory that you chose when you installed the documentation and sample programs for the GNU PRC-Tools (c:\palmdev). Be sure to choose **PRC-Tools** when you install the SDK, as shown in Figure 19-3.

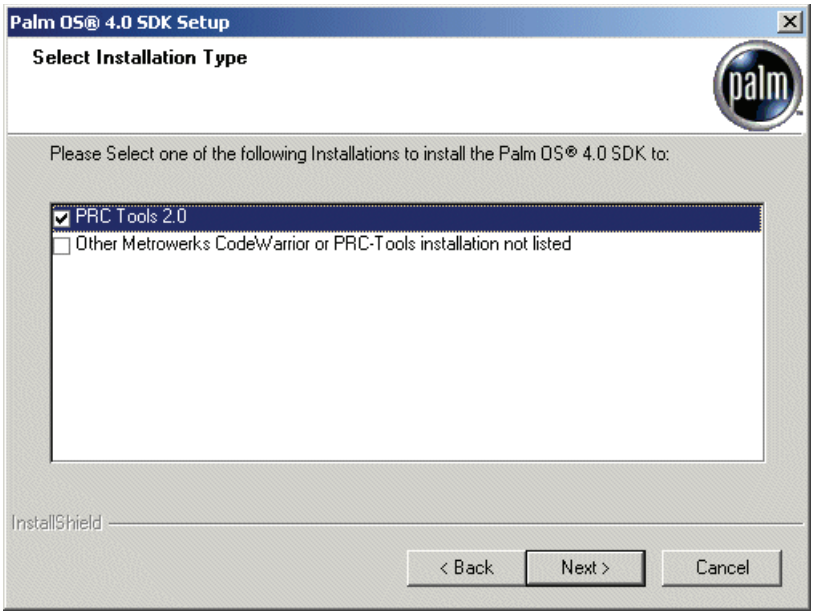


Figure 19-3 Select the PRC-Tools option

4. Install the PilRC resource compiler by extracting the content of the pilrc.zip file located in <DB2eSDK-installation-path>\SDK\ApplicationBuilder\Toolkits\Palm

Development\MAB_prereqs to a directory. By default, MAB expects to find it on c:\pilrc tools.

5. Install the Palm desktop as detailed in Appendix E, “Palm Desktop installation” on page 1193.

You can test the applications that you are writing either on your mobile device, running the Palm OS, or you can set up a testing environment using an emulator such as the Palm OS Emulator (POSE) for Palm OS 3.x/4.x or Palm Simulator for Palm OS 5.

Note: Palm Simulator does not integrate with the MAB user interface, but we can use the Install Database option on the Simulator to deploy the application file that MAB generates.

For this environment, the Palm OS Emulator V3.5 is used. The emulator.exe file must be located in the Application Builder installation path in order for the MAB user interface to find it. When you install the files to this directory, it ask you whether you would like to replace the existing HostControl.h file with the one contained in the POSE ZIP file. We will replace the file, because the one in the ZIP file was more than twice the size of the existing one. For more information about the installation and configuration of the emulator software, refer to Appendix D, “Palm Emulator installation” on page 1185.

19.4 Windows CE development environment

Mobile Application Builder generates applications for Windows CE devices with a JVM installed. MAB-built applications have been tested with the following vendor JVMs:

- IBM J9 (as included in WebSphere Studio Device Developer
- Insignia Corporation’s Jeode PDA Edition Version 1.9
- Sun Microsystem’s Personal Java Runtime Environment 1.0
- CrEme Plus JVM, Version 3.21 (required for barcode-scanning applications)

The Windows CE JVM preferences should be set appropriately for the JVM you are using before the application is built.

The following section covers obtaining and installing IBM J9 JVM on Windows CE devices. For more information about how to configure JVMs for other than J9, refer to the Mobile Application Builder Version 8.1.2 documentation.

To obtain the IBM J9 JVM and install it on a device:

1. Install WebSphere Studio Device Developer as indicated in 18.2, “Installing WebSphere Studio Device Developer” on page 532.
2. WebSphere Studio Device Developer supports many configurations, both standard (J2ME configurations) and custom (IBM configurations). MAB uses a custom configuration called jclMax, which provides advanced functionality. This custom configuration doesn’t come by default with the WebSphere Studio Device Developer product, but you can download it and install it using the WebSphere Studio Device Developer Update Manager as explained in 18.3.2, “Using the Update Manager” on page 537. You need to install the WCE Tooling for WebSphere Studio Device Developer Version 5.5.0, the WCE jclMax Class Library Version 5.5.0, and the WCE Personal Configuration Class Library Version 5.5.0.
3. Copy the following files or directories to the device or emulator from the WebSphere Studio Device Developer installation. The directory structure described below must be maintained for the J9 executable to work properly. Note that <WSDD> is the folder where you installed WebSphere Studio Device Developer software.

   Install the following files obtained from <WSDD>\wsdd5.0\ive\lib\jclMax, in the \j9\lib\jclMax directory on the device:

   – classes.zip
   – database_enabler.jar

   Obtained from <WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\lib\jclMax, in the \j9\lib\jclMax directory on the device:

   – prsnlwin.jar

   Obtained from the directories shown following, in the \j9\lib directory on the device:

   – <WSDD>\wsdd5.0\ive\lib\ charconv.zip
   – <WSDD>\wsdd5.0\ive\lib\jclMax\locale.zip

   Obtained from <WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin, in the \j9\bin directory on the device:

   – iverel20.dll
   – j9.exe
   – j9dyn20.dll
   – j9int20.dll
   – j9max20.dll
   – j9midp20.dll
   – j9prt20.dll
   – j9thr20.dll
   – j9vm20.dll

- j9w.exe
- j9zlib20.dll
- swt-win32-2130.dll

19.5 DB2 Everyplace runtime environment

To run applications that connect to a DB2E database and synchronize to DB2E SyncServer, you need to install the DB2E runtime support that is part of the Everyplace Client. For instructions on installing the Everyplace Client on the supported platforms, refer to Chapter 4, “Everyplace Client” on page 81.

19.6 Scenario: A sales force automation application

The sample scenario in this section is an example of a sales force automation application. It allows a sales person of a company to visit customers and review their orders using mobiles devices running Palm OS and Windows CE devices.

Although MAB is a RAD tool, we recommend you do some planning before starting with the application coding. Use the following recommendations as a guide to develop MAB-built applications:

1. Define the database structure that you will use on your mobile devices. This structure generally is a subset of an Enterprise Relational Database (ERDB) structure.
2. Think about the flow of your application. Make a diagram that shows the sequence of forms that you are going to use.
3. Create a project on the MAB that targets your device platform.
4. Import the database structure to the MAB project, using a DDL file that contains all the SQL CREATE TABLE statements for your tables.
5. Create the forms following the outline that you have designed in step 2.
6. Test each form as soon as you have finished.

19.6.1 Table definitions for the application

Usually when you are planning to create applications for mobile devices, the idea is to bring data from a central database server to this device and load data entered on the mobile device back into the central database.

This means that the layout of the tables is more or less given and you only have to select whether you want all columns of a table to be included in the database table of the mobile device.

To create the database tables necessary for test the application you have two ways:

- Use the DB2eCLP application that comes with Everyplace Client. You have to run the DDL file to create the database tables.
- Use the DB2 Everyplace ISync Client, which is part of the Everyplace Client, to synchronize the database tables. To use it, you have to configure the DB2 Everyplace SyncServer for the Order System sample database tables. For more information, refer to Chapter 20, “Synchronizing with DB2 databases” on page 645. We use this approach in this chapter.

When you have configured the subscription, you should use the DB2 Everyplace ISync Client to bring the database tables and data to the mobile device or to the testing environment. Refer to 20.5, “DB2 Everyplace Client configuration” on page 665 for more information.

A total of four database tables are needed for the application:

- CUSTOMERS
- ORDERS
- PRODUCTS
- ORDER_ITEMS

Important: Be careful when you choose the database tables names. You cannot use a SQL reserved word as a table name. If you do that, your application will fail.

The ORDERS table holds the new orders that are entered during the day, whereas the ORDER_ITEMS table holds the items for every order in the ORDERS table. The CUSTOMERS table contains information about all customers of a sales representative. PRODUCTS shows all available products. Refer to Figure 19-4 on page 612 for the definition of the tables.

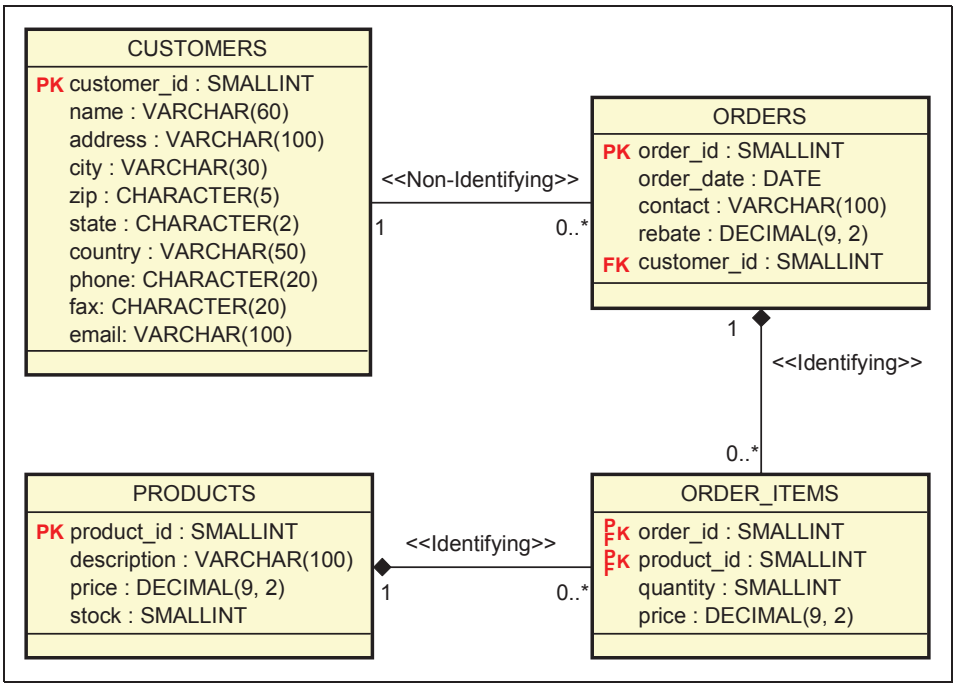


Figure 19-4 Data model diagram

Depending of the complexity of your application, you might need a database modeler tool. For the creation of the sample database model, the IBM Rational® Rose Professional Data Modeler Edition is used as the database modeler. For more information about the complete range of IBM Rational solutions for developers, visit the IBM Rational site at `http://www.rational.com`.

You should create a file containing the necessary SQL table definition statements for the sample application. Make sure that the file extension is .ddl so that the definitions can be imported into the PAB. An example of a DDL file is shown in Figure 19-5 on page 613.

```
CREATE TABLE PRODUCTS (
       product_id           SMALLINT NOT NULL,
       description          VARCHAR(100) NOT NULL,
       price                DECIMAL(9,2),
       stock                SMALLINT,
       PRIMARY KEY (product_id)
);

CREATE TABLE CUSTOMERS (
       customer_id          SMALLINT NOT NULL,
       name                 VARCHAR(60) NOT NULL,
       address              VARCHAR(100) NOT NULL,
       city                 VARCHAR(30) NOT NULL,
       zip                  CHAR(5),
       state                CHAR(2),
       country              VARCHAR(50) NOT NULL,
       phone                CHAR(20) NOT NULL,
       fax                  CHAR(20),
       email                VARCHAR(100),
       PRIMARY KEY (customer_id)
);

CREATE TABLE ORDERS (
       order_id             SMALLINT NOT NULL,
       order_date           DATE NOT NULL,
       contact              VARCHAR(100),
       rebate               DECIMAL(9,2),
       customer_id          SMALLINT NOT NULL,
       PRIMARY KEY (order_id),
       FOREIGN KEY (customer_id)
                             REFERENCES CUSTOMERS
);

CREATE TABLE ORDER_ITEMS (
       order_id             SMALLINT NOT NULL,
       product_id           SMALLINT NOT NULL,
       quantity             SMALLINT NOT NULL,
       price                DECIMAL(9,2) NOT NULL,
       PRIMARY KEY (order_id, product_id),
       FOREIGN KEY (product_id)
                             REFERENCES PRODUCTS,
       FOREIGN KEY (order_id)
                             REFERENCES ORDERS
);
```

Figure 19-5 DDL file for the sample application

Note: You cannot update a primary key value. You need to use a combination of delete and insert statements to do an update of a primary key column. If you try to update a primary key, you get an error of type SQLSTATE 42997 (unsupported feature).

Having created a file containing the table specifications, you are now ready to start working on the application.

19.6.2 Planning the flow of the application

Before you actually start creating forms in the Mobile Application Builder, we suggest you create some sort of flow diagram for the application.

This will identify which form a user sees first, what actions he can take from there, and which other forms these actions call. You can also specify which database tables are accessed from the forms.

19.6.3 Create a project in the Mobile Application Builder

A project in the context of the MAB contains all the resources and controls used in an application. For every application, you create a new project. Start the Mobile Application Builder and select **Start a new project** from the Welcome window of the MAB, as shown in Figure 19-6 on page 615.

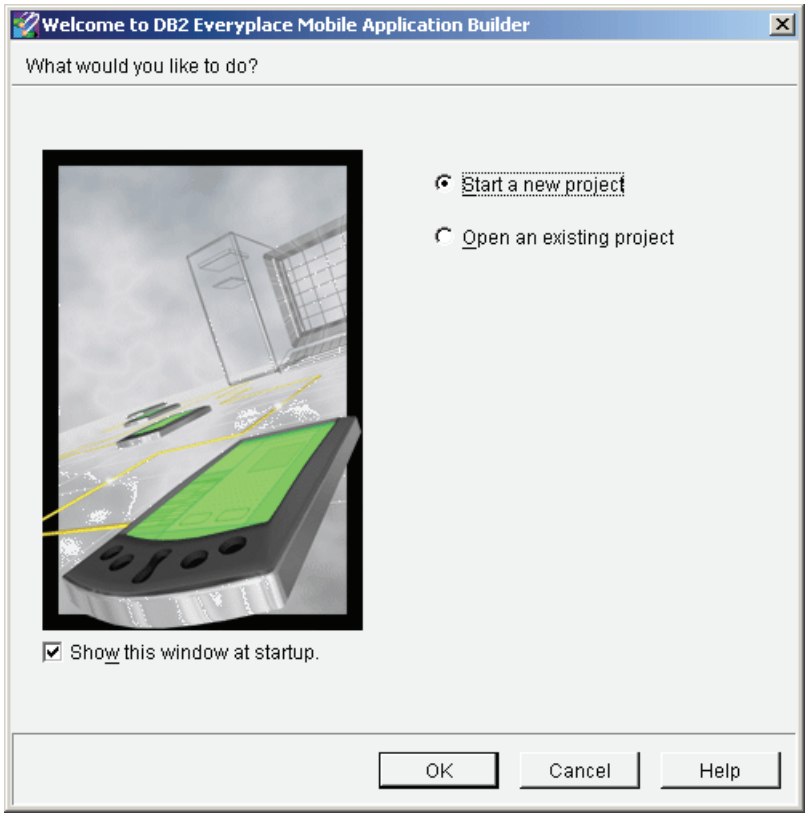


Figure 19-6 MAB Welcome window

MAB supports the following target devices:

- Palm
- Windows CE
- Symbian (not covered in this book)
- Sharp Zaurus

For Palm applications

You need to assign a name to the project and select a directory where all the necessary information is stored, as shown in Figure 19-7 on page 616. Click **Finish**.

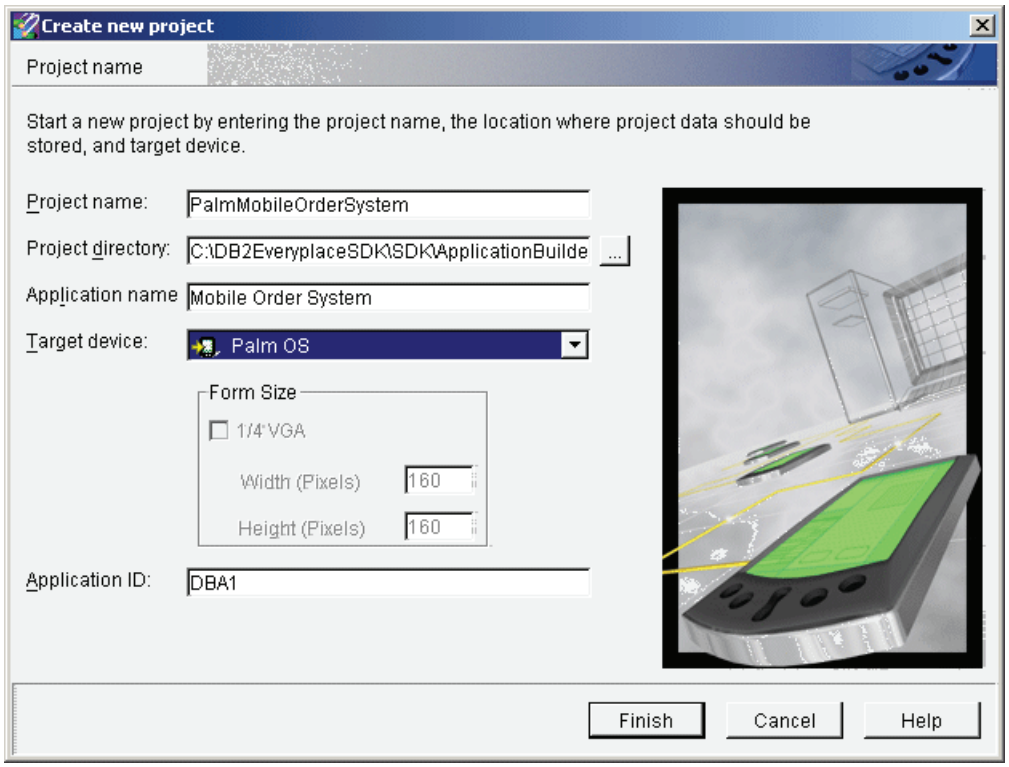


Figure 19-7 Create new project for Palm

Every Palm application is identified by a unique four-digit application ID. Palm Computing maintains a database with all existing Palm application IDs. Before you deploy a Palm application to the field, you need to register the application ID you are using in this database. You can check for existing application IDs and register your application ID at `http://dev.palmos.com/creatorid/`.

When you create a new project, you can use the application ID that is provided by the MAB (DBA1). This ID is registered in the application ID database on the device and can be used for testing purposes. The application ID can always be changed later on when you are ready to deploy your application.

Note: DBA1 is also used by the sample applications provided by the MAB. You need to make sure that you remove the sample applications on the device or emulator before installing your application using the same application ID.

For Windows CE applications

You need to assign a name to the project and select a directory where all the necessary information will be stored, as shown in Figure 19-8. Click **Finish**.

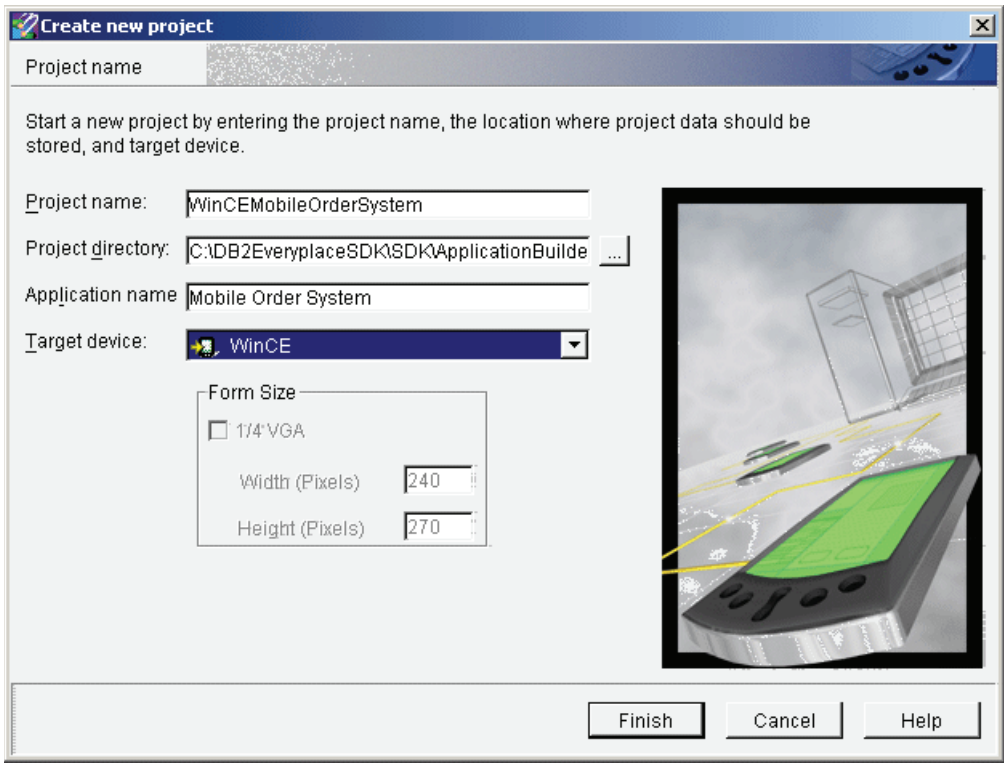


Figure 19-8 Create new project for Windows CE

19.6.4 Configure the preferences

Before starting, you will need to perform certain configurations, related basically to setting up the building and testing environment for each supported platform.

For Palm applications

1. Click **File -> Preferences** from the menu bar. The Preferences window opens.
2. Click **Palm Tools** to see the Palm tools preferences. For each tool, specify the path to the root directory where the tool was installed.

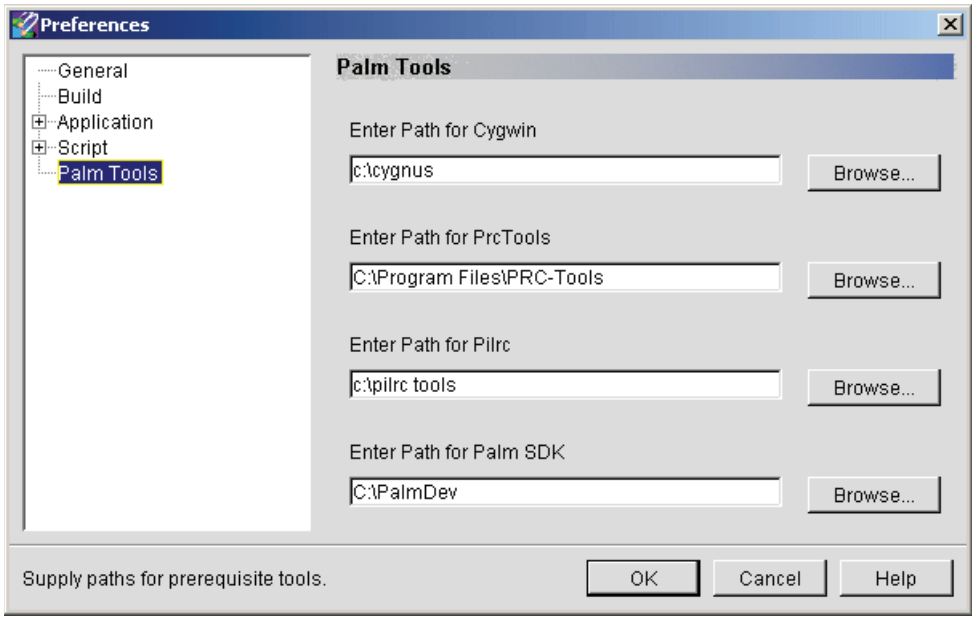


Figure 19-9 Configuring the preferences for Palm

3. Click **OK**. The Preferences window closes.

For Windows CE applications

For Windows CE applications, you will need to configure the emulation and JVM parameters. Follow these steps:

1. Click **File -> Preferences**. The Preferences window opens.
2. Click **Win32 Emulation Path Requirements** to see the Windows CE emulation preferences. Specify the path to the root directory where the DB2E SDK was installed (“C:\DB2EveryplaceSDK\” in this case), and specify the path to the Win32 database tables on the local PC, as shown in the Figure 19-10 on page 619.

Note: You can use the Win32 Sync Client, located on <DB2ESDK installation directory>\ Clients\Win32\sync\en_US, to synchronize the database tables to a local directory for testing purposes. You need to configure the server name, user name, password and target local directory.

For more information about the use of the Sync client for the Win32 platform, refer to the DB2 Everyplace product documentation.

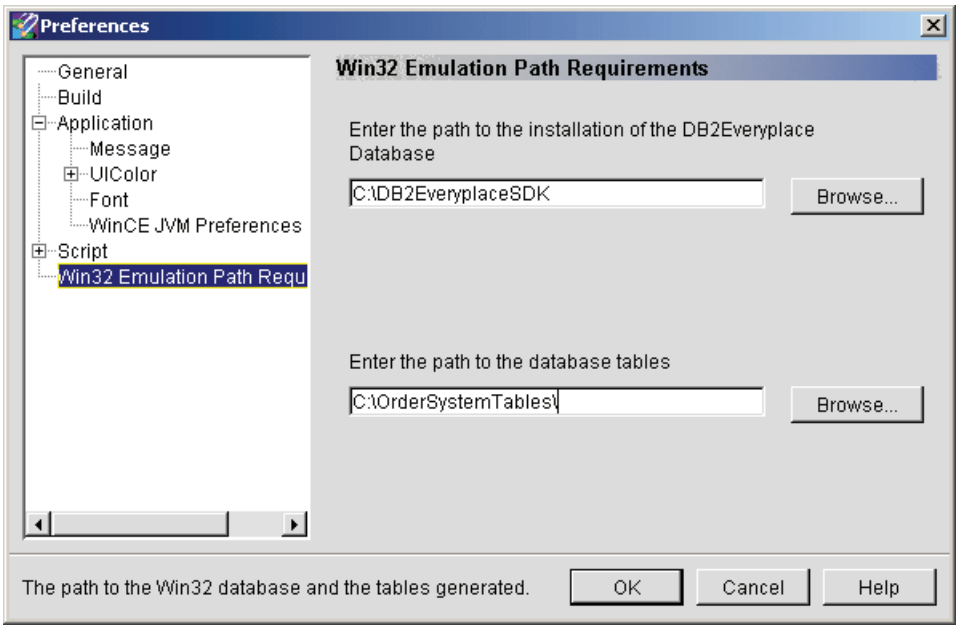


Figure 19-10 Win32 Emulation preferences

3. Click **Application -> WinCE JVM Preferences** to see the Windows CE JVM preferences, as shown in Figure 19-11 on page 620. Here you can select the JVM that you use and perform specific configuration for it.

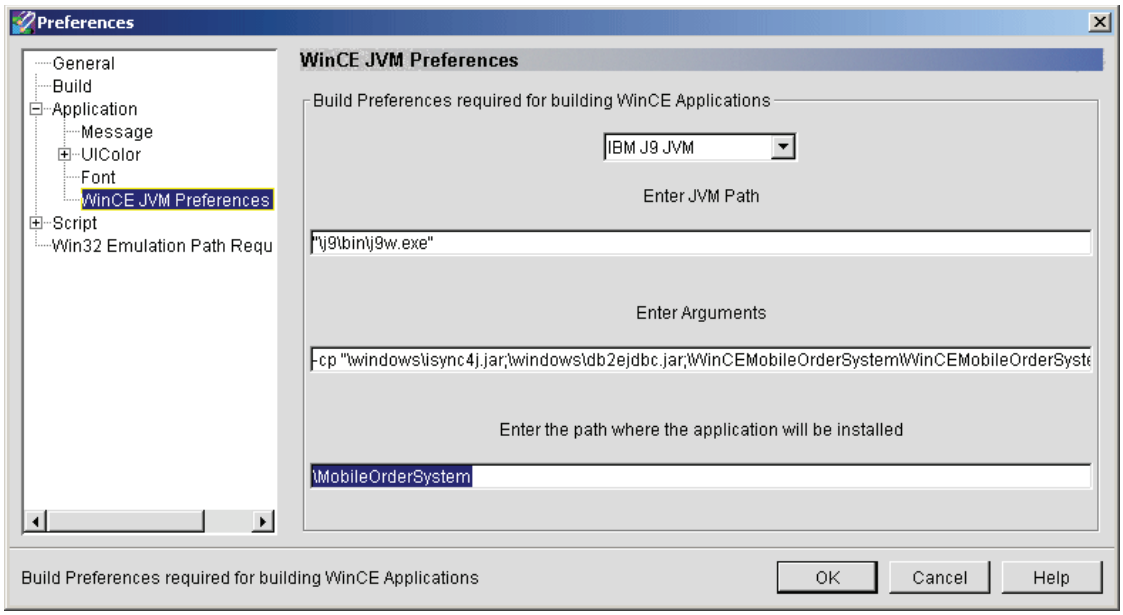


Figure 19-11 JVM preferences

For the sample application, we use the configuration parameters as shown in Table 19-1.

Table 19-1 JVM configuration parameters for the Windows CE application

Parameter	Value
JVM Path on the device	/j9/bin/j9.exe
JVM Arguments	-cp "\j9\lib\jclMax\database_enabler.jar;\j9\lib\jclMax\prsnlwin.jar;\windows\isync4j.jar;\windows\db2ejdbc.jar;\MobileOrderSystem\WinCEMobileOrderSystem.jar;" WinCEMobileOrderSystem.MABAppFrame
Application installation path	\MobileOrderSystem

4. Click **OK**. The Preferences window closes.

19.6.5 Load the table definitions

Once the project is defined, you can start loading the table definitions for the tables that will be used in the application.Mobile Application Builder includes an option to import table definitions into the project file. These definitions can then

be used further when creating forms or referencing columns of other tables on a form.

The DDL files you created in 19.6.1, “Table definitions for the application” on page 610 can now be imported. To perform this, follow these steps:

1. The MAB must be started and the project you are working on must be open.
2. Right-click **Tables** and select **Import table** as shown in Figure 19-12 or select **Tables** and start the import process by selecting **Selected -> Import** table from the menu bar.

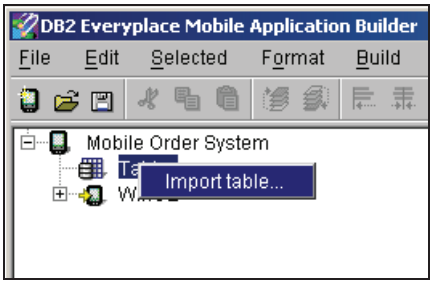


Figure 19-12 Importing table definitions

3. The Import table definition window opens. Select the DDL file that contains the definition for your tables and click **Open**. One DDL file can contain definitions for multiple tables. When the tables are loaded, they will appear in the Tables section of the project tree window (Figure 19-13).

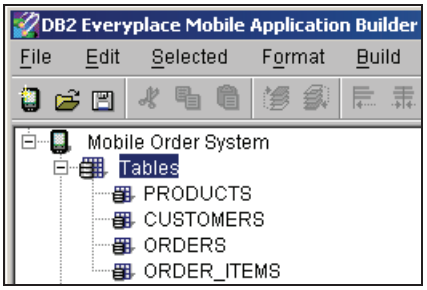


Figure 19-13 Table definitions loaded into the project

If you change the definition of a table, you need to import it again into the project. To do this, you need to delete the existing definition for a table (right-click the table, then select **Delete**) and then run the import process for this table again.

Tip: We suggest that you plan the layout for your tables very carefully to avoid problems when you have to re-import definitions into the MAB.

19.6.6 Start creating the forms

With the table definitions loaded into the project, you are now ready to start creating forms for your application. There are basically two ways to create a form within MAB:

- Create the form manually
- Use the Form Creation wizard to automatically generate a form for a table

We will also show you how to:

- Use host variables to link forms data
- Create a menu for the application
- Start a DB2e synchronization session
- Use the integrated scripting functionality

Creating a form manually

Because the first form of our application contains a list of all new orders, we will use the manual approach to create this form.

Note: By default, there is a blank form, Form1, available. You can delete the default form or you can use it. If you create a new form without deleting the default available form, you will not be able to see the form you have created because the form Form1 is the start form by default. Also, you can right-click a form and select **Set as initial form** to set it as the application initial form.

Follow these steps:

1. To create a new form manually, you can right-click the **Forms** entry in the Project window and select **Add new form** (Figure 19-14 on page 623) or click **Forms** in the Project window and select **Selected -> Add new form** from the menu bar.

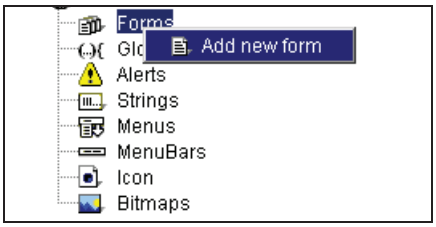


Figure 19-14 Adding a new form manually

2. A new empty form will be added to the Project window. Use the Properties/Events tab to change the caption of the form to something suitable for your application. Figure 19-15 shows you the settings for the caption of the first form of the example application.

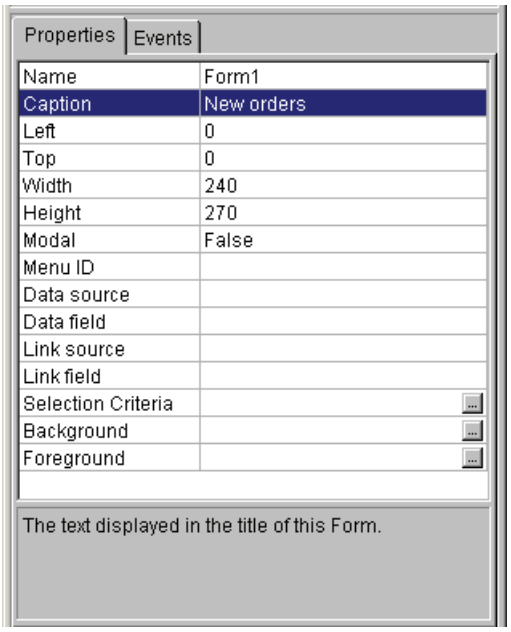


Figure 19-15 Changing the form properties

> **Note:** In this version of MAB the Name property of any object cannot be changed.

3. Use the available controls in the Control Panel to add fields, buttons, lists and more to the form. Table 19-2 on page 624 is an overview of available controls.

Table 19-2 MAB Control Panel

Icon	Palm Control	Windows CE Control
	Label	Label
	Field	Field
	Checkbox	Checkbox
	Button	Button
	List	List
	Push Button	n/a
	Repeating Button	n/a
	Scrollbar	n/a
	Selector Trigger	n/a
	Ink	n/a
	Graffitti Shift	n/a
	Pop-up Trigger	Choice List
	Form Bitmap	Form Bitmap

4. To create a list for all new orders, select the **List** control from the palette and draw a new list in the empty form.

5. Now add three buttons to the form and assign them the captions Sync, Edit and Delete respectively.
6. The columns that will be displayed in the list now need to be defined. Click the button in the Data sources field in the Properties tab of the List1Form1 list. A window opens where you can select a data source for every column in the list as shown in Figure 19-16.

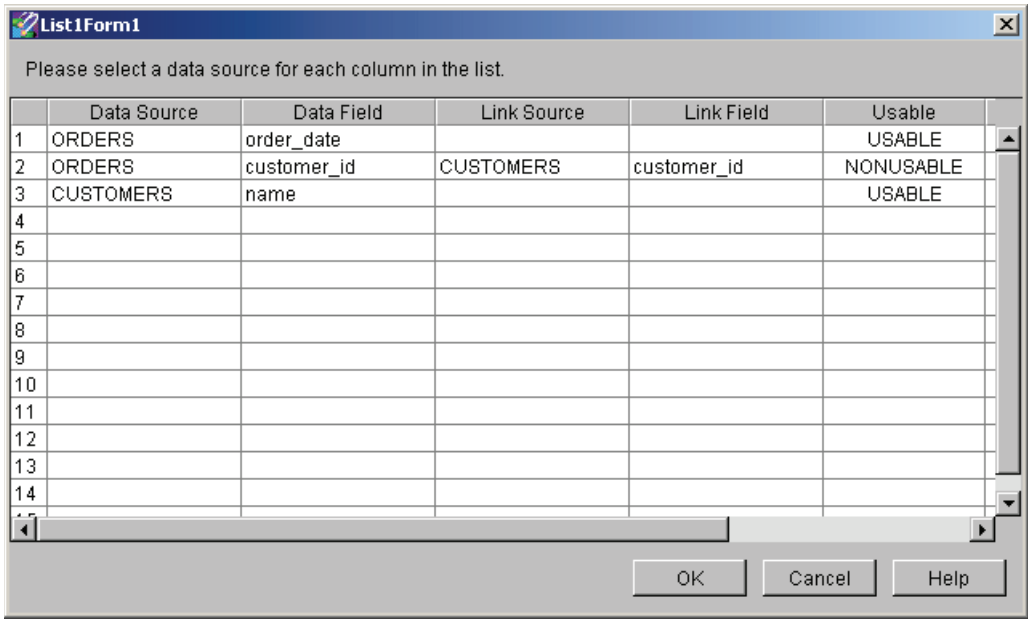


Figure 19-16 Assigning database fields to columns in a list

Note that in the list the order date and the name of the customer will be shown. This is achieved by linking the CUSTOMER_ID field of the ORDERS table to the CUSTOMER_ID field of the CUSTOMERS table as shown above. Also note that the usable option has been set to NONUSABLE in order to prevent this column being displayed.

7. Now let's assign an action to the Delete button. Click the button on the form to select it. Then click the **Events** tab in the Properties/Events window. The window will look like the one shown in Figure 19-17 on page 626.

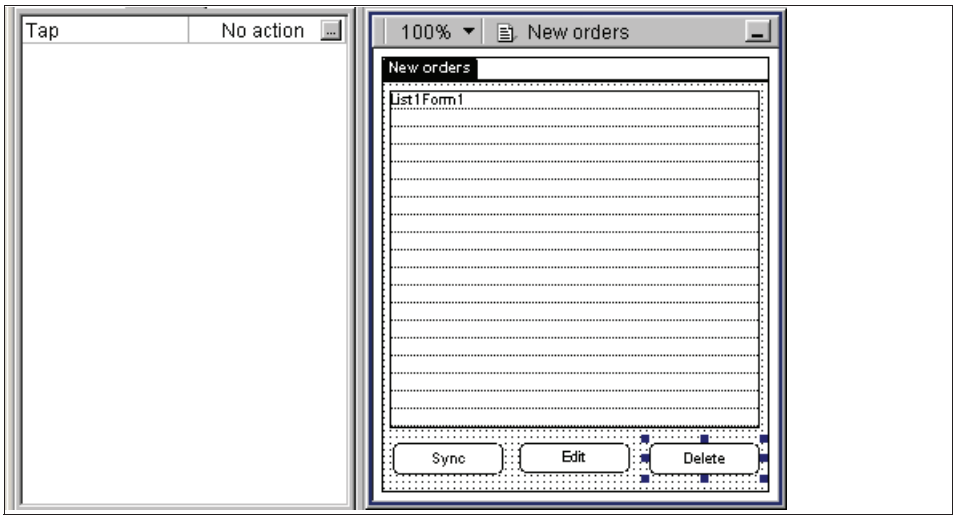


Figure 19-17 Associating an action to a button

8. Click the button near the **No action** label. This will open the Event Action and Target Selection window. Select **Delete record** from the actions list and select **ORDERS** as the target for this action. Figure 19-18 on page 627 highlights the steps to do this.

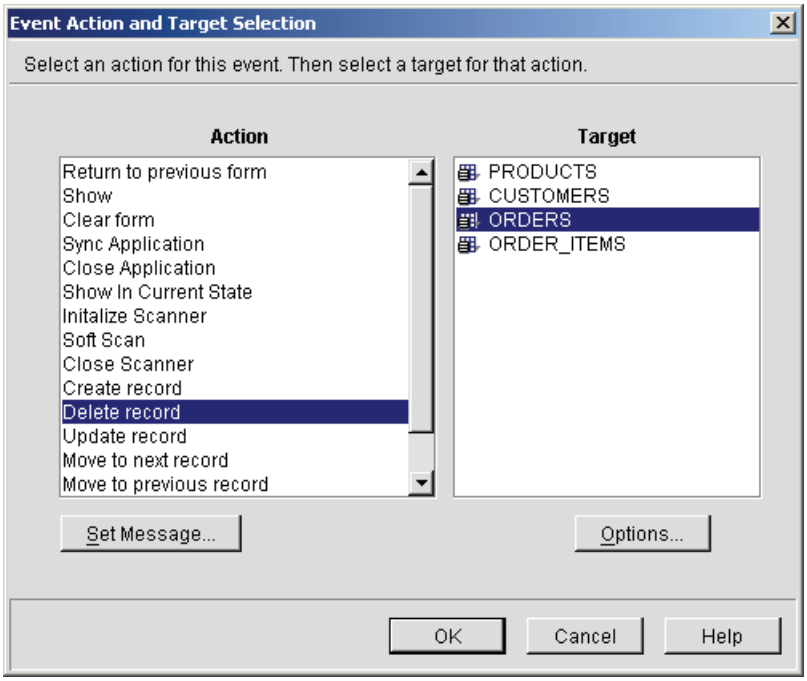


Figure 19-18 Associating an action and target to an event

Click **OK** to close the window. This associates the action to delete the selected record in the **ORDERS** table to the **Delete** button on your form.

9. Now you can build the application by selecting **Build -> Build** from the menu bar as displayed in Figure 19-19.

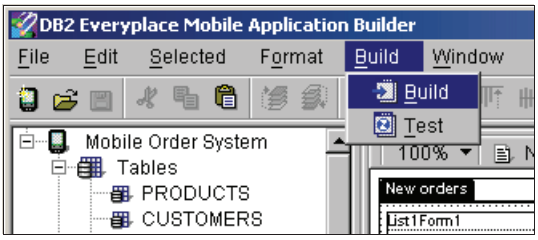


Figure 19-19 Starting the build process for the application

For Palm applications

If you test the application on your Palm OS Emulator or your Palm device, it looks like the one displayed in Figure 19-20 on page 628.



Figure 19-20 Startup form of the sample application running on a Palm device

Note: Selecting the **Build -> Test** menu option only starts the emulator program. It doesn't install the PRC file generated on the emulator. You must do it by selecting the **Install Database Application** menu option from the emulator program. You can locate the PRC file generated in your project directory.

Also, it is your responsibility to make sure that you have installed the Everyplace Client and that you have synchronized the database structure and data to the emulator session before starting the application test. For information about the installation of the Everyplace Client, refer to Chapter 4, "Everyplace Client" on page 81

For Windows CE applications

On Windows CE projects, MAB offers two ways to test the application:

- The MAB Windows CE Emulation environment
- Test directly on the device

The MAB emulation environment is a Java environment running on the desktop and needs to work properly, so that the paths on the **Preferences -> WinCE Emulation Path Requirements** are correctly established as indicated in 19.6.4, "Configure the preferences" on page 617. Although the emulation environment can help to give us an idea of the appearance of the application, it is always better to test on the real device.

Note: To test the real device, you have to install the J9 JVM, as discussed in 19.4, "Windows CE development environment" on page 608.

For real devices, MAB generates a Pocket PC CAB file that you can deploy on the real device. MAB uses the information that you entered in 19.6.4, "Configure the preferences" on page 617 to generate this file.

If you build and test the application on your Windows CE device, it now looks like the one displayed in Figure 19-21.

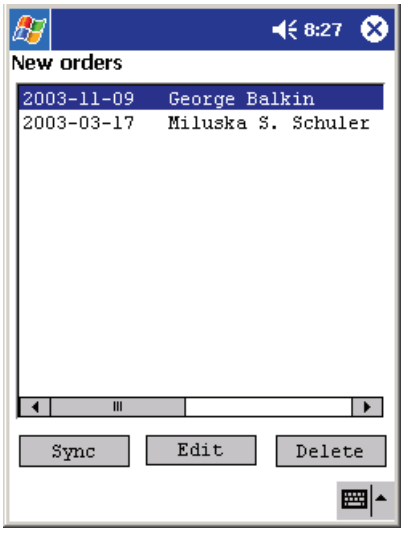


Figure 19-21 Startup form of the sample application running on a Pocket PC device

Before you can assign events to the other buttons, New and Edit, you need to create a New Order and an Edit Order form.

Creating a form using the Form Creation wizard

The Form Creation wizard is a two-step process to create a form for an existing table definition:

- In the first step, you select the columns of the table that should be included on the form.
- In the second step, you select the actions that should be available on the form.

Here is how to use the Form Creation wizard to create the Edit order form for the sample application:

1. Right-click the name of the table in the Tables section of the project tree window as shown in Figure 19-22 on page 630 and select **Launch form creation wizard** from the pop-up window. This starts the Form Creation wizard.

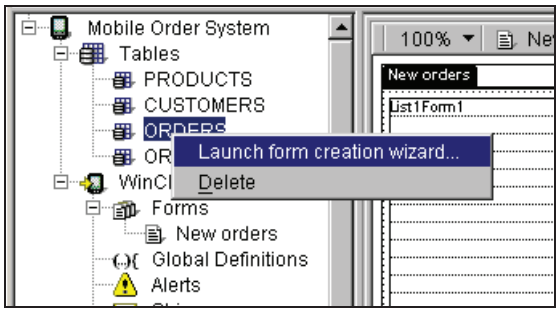


Figure 19-22 Starting the Form Creation wizard

Click **Next** to continue.

2. On the Database columns selection window, select all columns that should be included on the form, as illustrated in Figure 19-23.

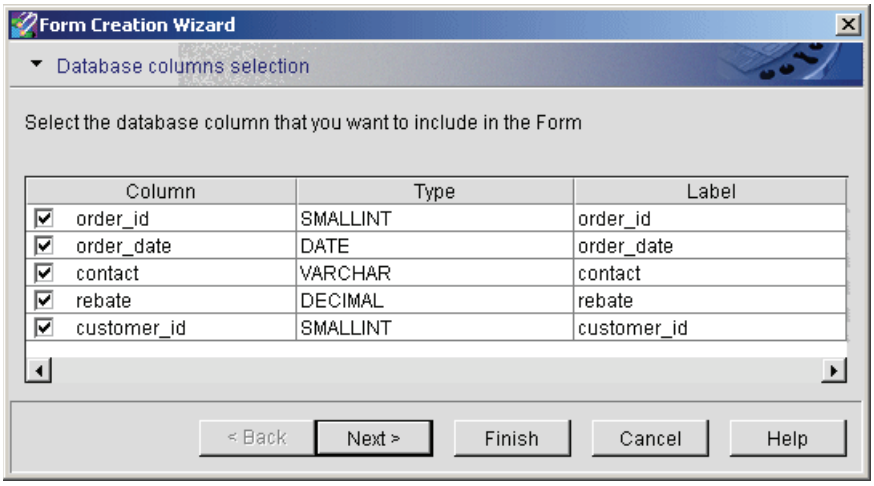


Figure 19-23 Database columns selection page in the Form Creation wizard

3. On the Database operations window, select all actions that should be available on the form (see Figure 19-24 on page 631).

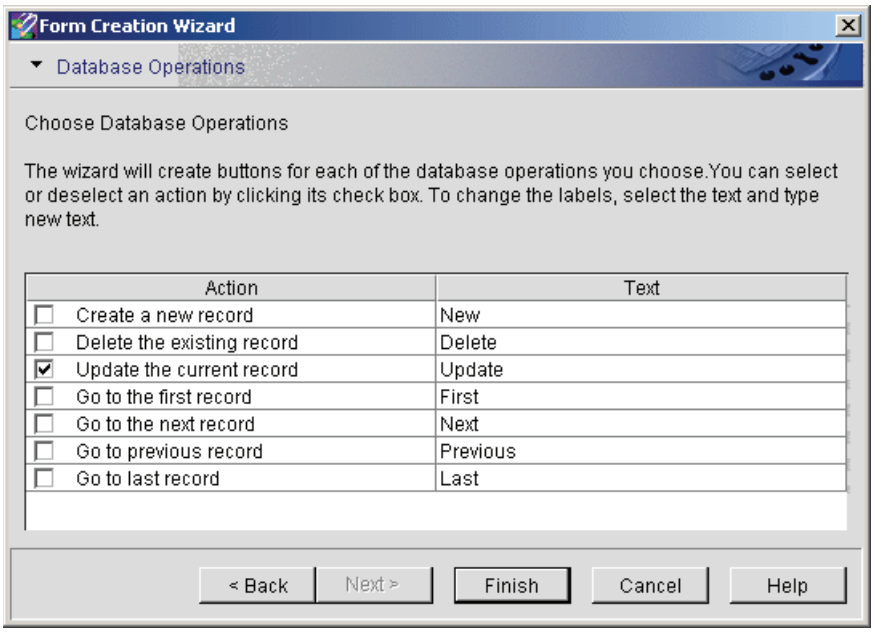


Figure 19-24 Database operations page in the Form Creation wizard

Click **Finish** to close the Form Creation wizard.

4. A new form will be created like the one shown in Figure 19-25.

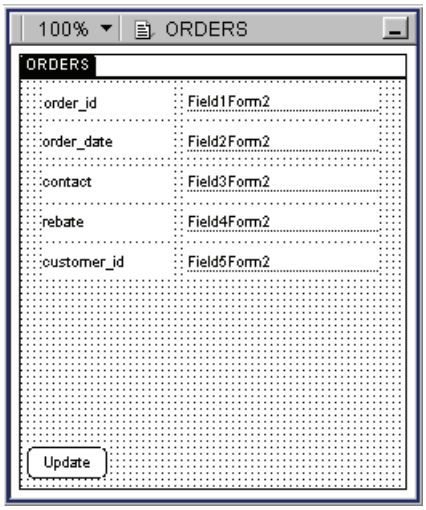


Figure 19-25 The new form created by the Form Creation wizard

Click the form and change the caption to Editing order.

5. Now click the field that was created for the order date. Because we do not want the user to change the date for the order, change the Editable property of this field to NONEDITABLE (see Figure 19-26). Do the same for the order ID field.

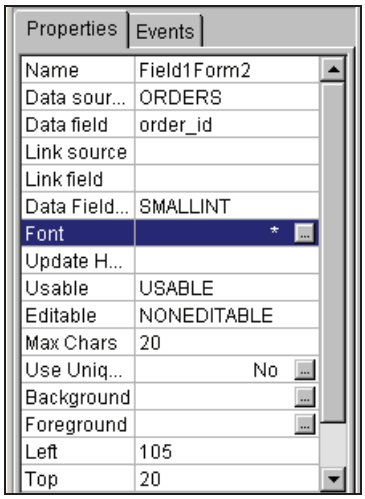


Figure 19-26 Changing the editable property of a form field

6. Change the Font property to FONT 1 for Palm applications. Or change the Font Style property to BOLD for Windows CE applications.

> **Note:** When you change some font property in a control, you should also change the font size value to a value greater than the default (1). On the Windows CE version of the sample application, we will use a 12-point font size.

7. Now click the label that was created for the order date field and delete it by selecting **Edit -> Delete** from the menu bar. Do the same for the order ID and customer ID generated labels.
8. For the customer ID field, change the properties in the Properties/Events window:
   - Change the following settings:
     - Change the font to Font 1 on Palm applications
     - Change the font style setting to BOLD on Windows CE applications
   - Change the Editable property to NONEDITABLE.

9. Rearrange the Form2Field1, Form2Field2 and Form2Field5 fields so that they appear on the top of the form, and change the caption of the Update button to Update order. You might also add an additional button, Back, to the form. This would allow the user to go back to the previous form from this form. Associate the Tap event of the button to the action Show with target New orders in the Events window.
10. Insert a new list on the form. Set its Data sources property in the Properties/Events window as shown in Figure 19-27.

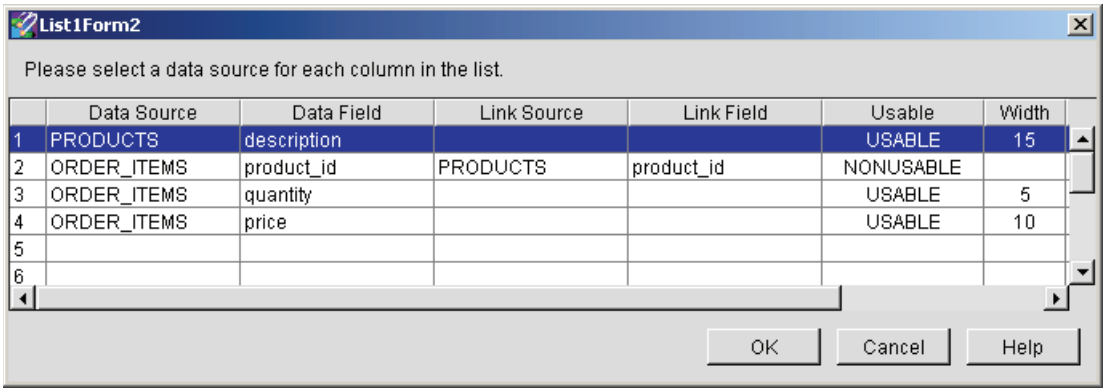


Figure 19-27 Setting the order items list data source

11. Finally change the field names and the caption of the other two fields on the form so that the form looks similar to the one shown in Figure 19-28.

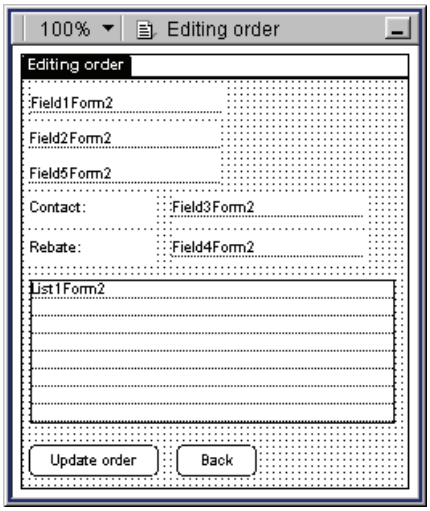


Figure 19-28 Creating the Editing order form

Because you used the Form Creation wizard to create the form, the fields, and the buttons on the form, you do not need to associate events to the Update order button on the form. The actions and the respective targets have been added during the creation process.

What you need to do is to associate the form to an event on the first form you created. Follow these steps:

1. Select the **New orders** form in the project tree of the MAB. We associate the action to display the Editing order form to the Edit button on the New orders form. Refer to the previous section for more details about how to associate an action and a target to an event.
2. To associate the event to the Edit button:
   - Select the button on the form
   - Click the **Events** tab in the Properties/Events window
   - Make sure that the event is Tap
   - Click the button in the action field and select **Show** from the actions list and **Editing orders** as the target
   - Click **OK**

Use host variables to link forms data

Although you have linked the two forms with the Show action, the data between both is still un-synchronized. To have the data from the forms synchronized, you use *host variables.* You will create a host variable named ORDERID. The first form (New orders) sets the value of this variable, and the second form (Editing orders) uses the value of this variable to set selection criteria for the data on the form.

To create a link between the data on the form, follow these steps:

1. To create the host variable, select **WinCE -> Global Definitions** for Windows CE applications, or **Palm OS -> Global Definitions** for Palm applications, in the Project tree view. On the Properties/Events window, click the **Host variables** property button. The Host variables window opens.

   Click the **Add** button, enter the variable name `ORDERID`, and click **OK**. You see the new host variable on the variable list, as shown in Figure 19-29 on page 635. Click **OK** to close the window.

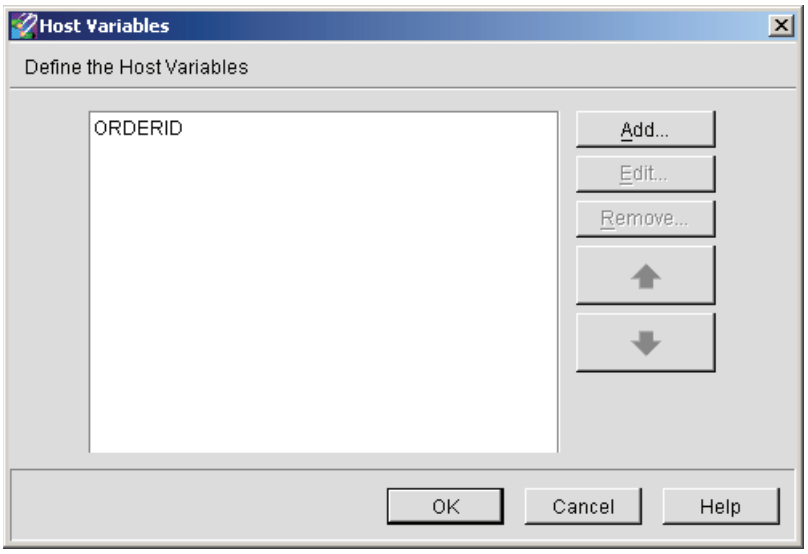


Figure 19-29 Creating a host variable

2. Now you set the value for the ORDERID host variable. Select the **List1Form1** list from the New orders form. On the Properties/Events window, click the **Update host variable** property button. The Update host variable window opens. In the first row, select the table **ORDERS** for the Data Source column, the field **order_id** for the Data Field column, and the **ORDERID** host variable for the Host Variable Name column, as shown in Figure 19-30. Click **OK**.

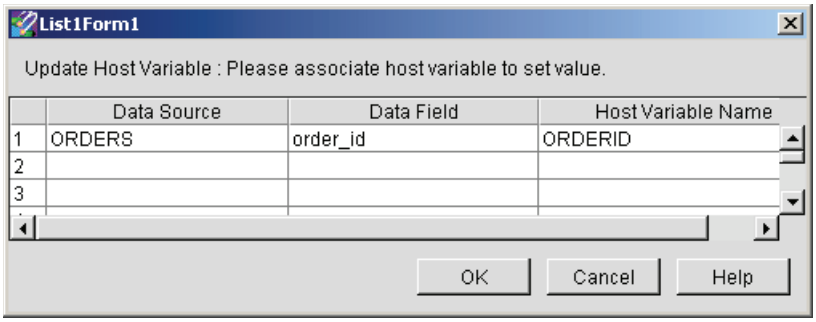


Figure 19-30 Updating a host variable

3. Now, you use the host variable to define selection criteria for the form. Select the **Editing orders** form from the Project tree view and click the **Selection Criteria** property button. The Selection criteria window opens. In the first row, select the table **ORDERS** for the Data Source column, the field **order_id** for

the Data Field column, and the **ORDERID** host variable for the Host Variable Name column, as shown in Figure 19-31. Click **OK**.

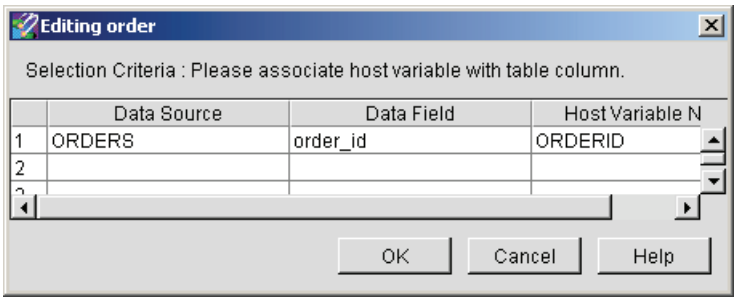


Figure 19-31 Creating a selection criteria

4. Finally, you have to define the selection criteria for the list. Select the list and click the button for the **Selection Criteria** property. The Selection criteria window opens. In the first row, select the table **ORDER_ITEMS** for the Data Source column, the field **order_id** for the Data Field column, and the **ORDERID** host variable for the Host Variable Name column. Click **OK** to finish.

If you run the application in your device, you see the new Editing order form synchronized with the New Orders form, as shown in Figure 19-32. You can update the Rebate and Contact form and use the Update order button to reflect the changes in the database.

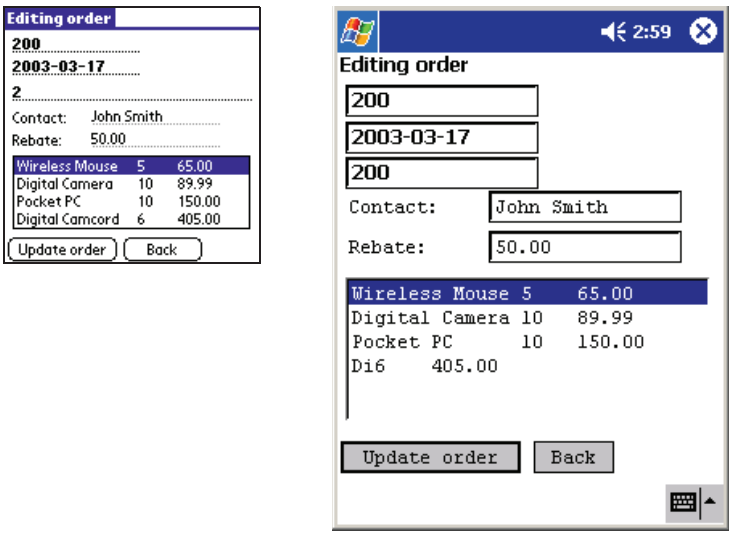


Figure 19-32 Editing order form

Create a menu system for the application

On MAB applications, the menu system is composed of a menu bar, pull-downs, and menu options. The menu bar is the main menu of the application, located on the upper part of the form in Palm applications and in the lower part of the form in Pocket PC applications. When you make a selection from the menu bar for the form, one or more pull-downs are shown. Each pull-down has menu elements that trigger actions. Additionally, each menu option has an accelerator key.

For the application, the only form that has a menu bar is the New order form. It has a pull-down named Options, with two menu elements:

- Close: Closes the application
- Sync Preferences: To configure the DB2e synchronization preferences for the application

To create a menu system for the application, follow these steps:

1. Create a menu element for the Close menu option. Right-click **Mobile Order System -> PalmOS -> Menus** for Palm applications, or **Mobile Order System -> WinCE -> Menus** for Windows CE applications and click **Add new menu** as shown in Figure 19-33. A new menu element is created under the Menus branch.

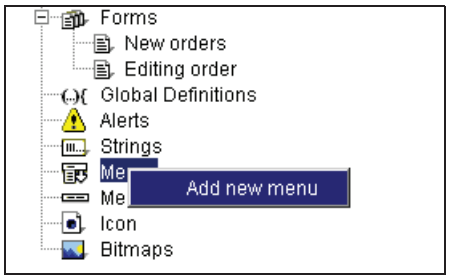


Figure 19-33 Adding a new menu option element

2. Select the recently created menu element and set its properties in the Properties tab as shown in Figure 19-34 on page 638.

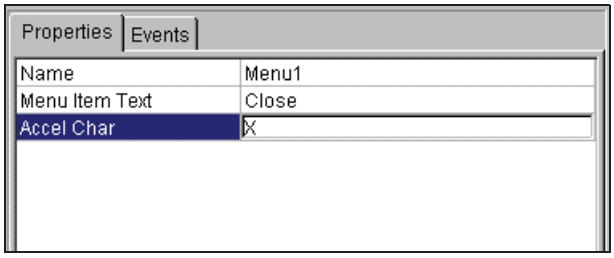


Figure 19-34 Editing the properties for the menu option element

3. To set the action for the menu element, click one of the Actions button in the Events tab. In the Event Action and Target Selections window, select the **Close application** action, as shown in Figure 19-35. Click **OK**.

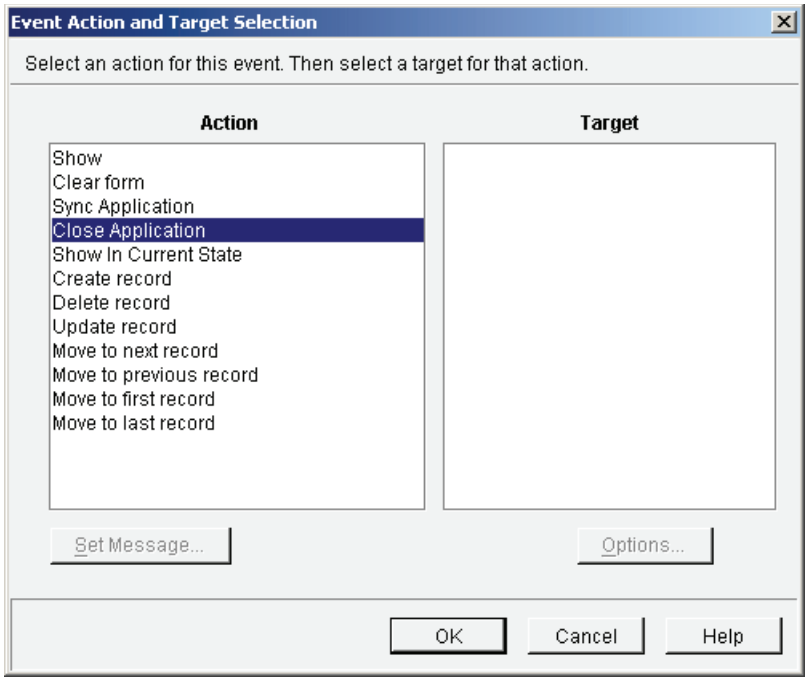


Figure 19-35 Associating an action to a menu element

4. Add an additional menu element following the steps mentioned above. In the Properties window, for the Menu Item Text property enter `Sync Preferences...` and for the Accel Char property enter `P`. In the Event tab, select **Show -> Sync Preferences dialog** as the action. Now you have two menu elements configured as shown in Figure 19-36 on page 639.

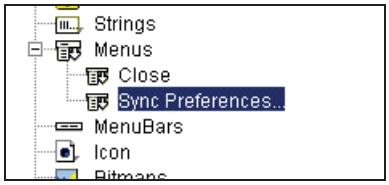


Figure 19-36 Menu options for the application

5. Create the menu bar for the New orders form. Right-click **Mobile Order System -> PalmOS -> MenuBars** for Palm applications or **Mobile Order System -> WinCE -> MenuBars** for Windows CE applications and click **Add new menu bar** as shown in Figure 19-37. A new menu bar is created under the Menu Bars branch.



Figure 19-37 Adding a menu bar to the application

6. Select the recently created menu bar and set its properties in the Properties tab as shown in Figure 19-38. Note in the figure that there are two menu elements for the menu bar. To add additional menu elements, right-click **Menu Items ID** and select **Add Menu Item.** A new menu item field appears.

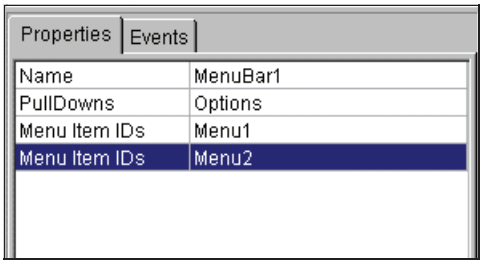


Figure 19-38 Setting the properties for the menu bar

7. Finally assign the new menu bar to the New orders form. Select the form from the tree view, and set its Menu ID property to MenuBar1, as shown in Figure 19-39 on page 640.

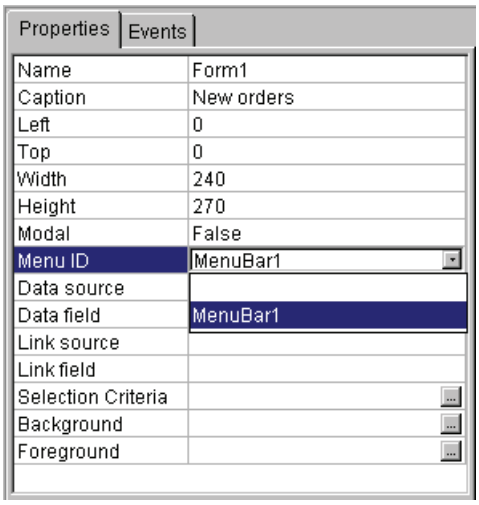


Figure 19-39 Associating the menu bar to the New orders form

Start a DB2e Synchronization session

To add synchronization capabilities to the application is very easy with MAB. In the previous section, you added Sync Preferences... as a menu option. Now, you only have to set the action for the Sync button to **Sync Application**, as shown in Figure 19-40.

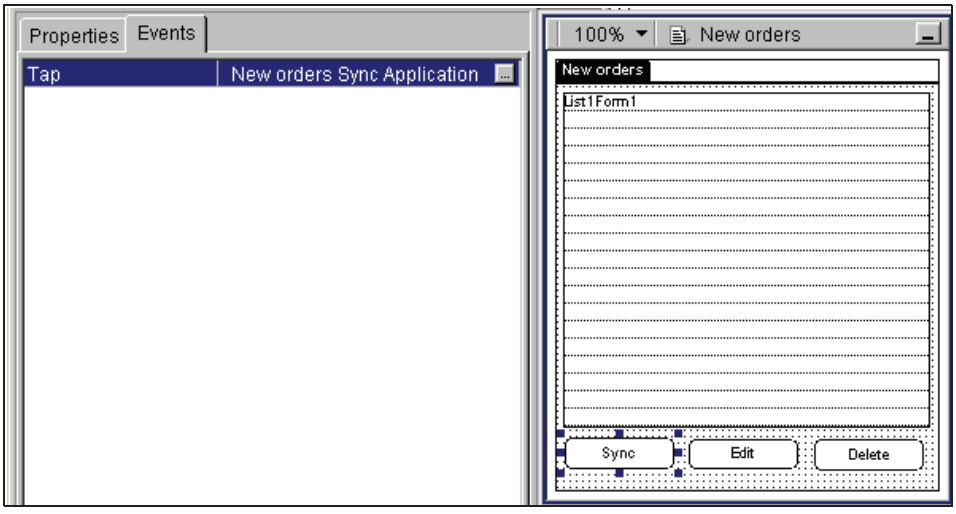


Figure 19-40 Sync Application action for the Sync button

Now build and deploy the application on the device. To test the application:

1. Select **Options -> Sync Preferences** from the menu bar, as shown in Figure 19-41.

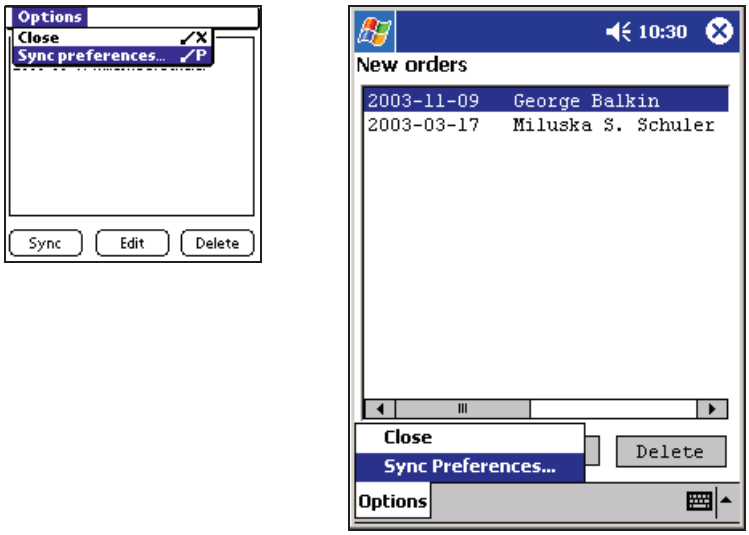
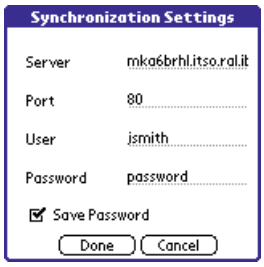


Figure 19-41 The menu system in action

2. In the Sync Preferences window, enter the information required by the ISync client, as shown in Figure 19-42 on page 642. Select **Done** in Palm applications or click **OK** in Windows CE applications.



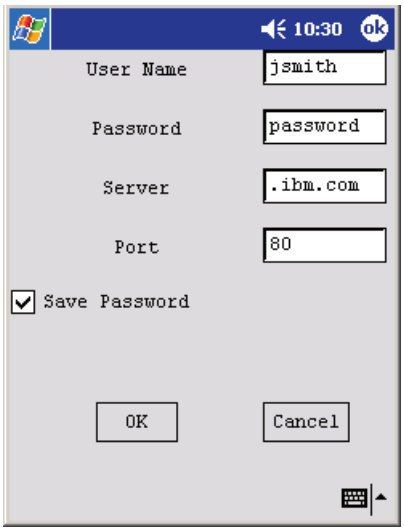


Figure 19-42 Synchronization settings window

3. Now click **Sync** to start the synchronization session. A window, shown in Figure 19-43, shows you the progress of the task.



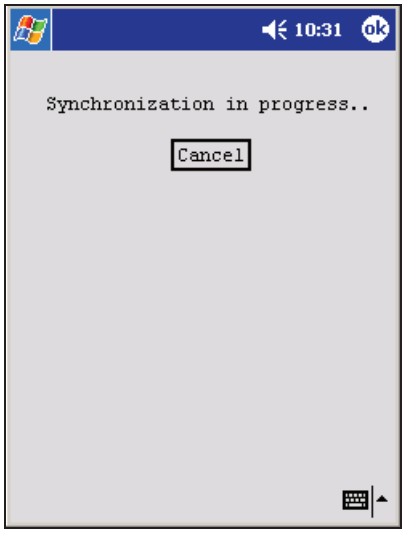


Figure 19-43 Synchronization in progress window

4. At the end, you see a window that shows the synchronization result. Click **OK**.

Use the integrated scripting functionality

As you probably have noted, the Delete button on the New orders form removes the record from the database but doesn't update the order list. To accomplish this task, you have to write some custom code by using the integrated scripting functionality provided by MAB.

Note: The programming language for scripting depends of the target device. For Palm applications the language is C and you can use the PalmOS native API. For the rest of the devices, the language is Java and you can use the API of your particular JVM. For J9 on Windows CE devices, you can review the WCE jclMax documentation for the subset of the J2SE API available.

To create a custom script for updating the order list, follow these steps:

1. Right-click the **Delete** button and select **Create a new script**, as shown in Figure 19-44.

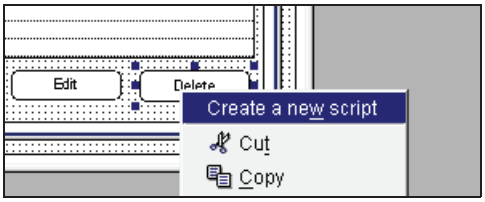


Figure 19-44 Creating a new script

2. In the script window, type the code that update the list. On Palm applications we use the FrmGotoForm() PalmOS function to reload the current form and therefore to update the list, as shown in Figure 19-45 on page 644. On Windows CE applications, there is a MAB-specific Java function named repopulate(), available on each form, that refreshes all the data-bound controls. We use this function for the Windows CE version of the script, as shown in Figure 19-46 on page 644.

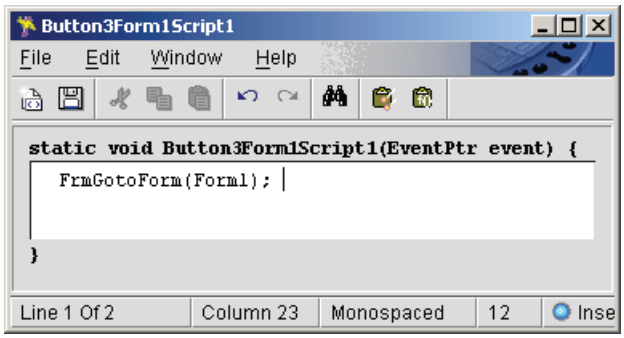


Figure 19-45 Script code for Palm applications

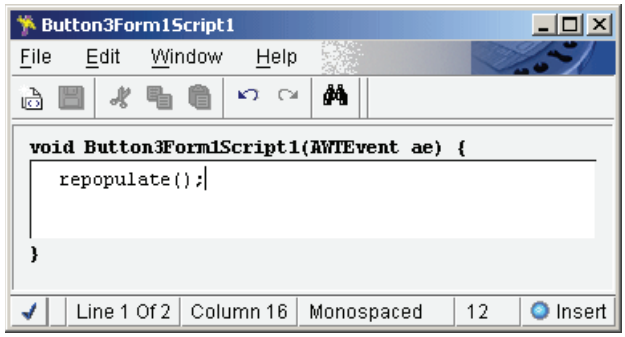


Figure 19-46 Script code for Windows CE applications

Click **File -> Save script** to save the changes and then click **File -> Close** to close the script window.

20

Synchronizing with DB2 databases

This chapter describes the configuration steps necessary to carry out DB2 Everyplace Synchronization with back-end DB2 databases. The following topics are covered:

- Overview
- DB2 Everyplace Synchronization Server configuration
- Everyplace Client configuration
- Sample application scenario
- Synchronize with remote databases
- Hints and tips

Note: In this chapter, the root directory that WebSphere is installed to is referred to as “WebSphere_Install_Dir”. It should be replaced by the actual directory name in real cases.

© Copyright IBM Corp. 2003. All rights reserved.

20.1 Architecture overview

The diagram shown in Figure 20-1 depicts the relational database synchronization environment for JDBC subscription types.

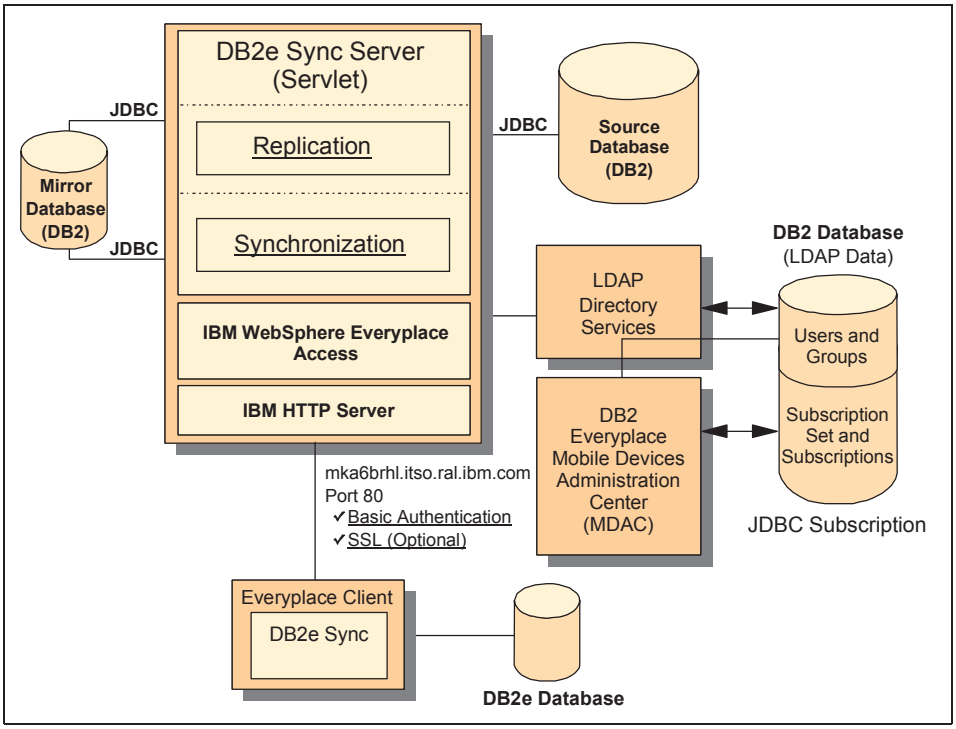


Figure 20-1 DB2 Everyplace synchronization for JDBC subscriptions

The Pocket PC device sits on the IBM Everyplace Client, which is the unified client for PIM and relational database synchronization. The Palm device has a stand-alone client for database synchronization called the ISync Client.

On the server side, IBM HTTP Server handles incoming HTTP requests, and passes those destined for WebSphere Application Server via a plug-in. WebSphere Portal rides on top of the WebSphere Application Server. It provides administration portlets to manage portlets as well as users and groups. User and group information is stored within LDAP.

Also on the server, DB2 Everyplace periodically replicates the back-end databases to mirror databases; this process for JDBC subscription type is also illustrated in Figure 20-1. DB2 Everyplace Mobile Devices Administration Center (MDAC), together with WebSphere Portal, provides the complete administration

functionalities for DB2 Everyplace Sync Server. MDAC stores its configuration information in a DB2 database (DSYCTLDB).

When synchronization is initiated at the client, the device sends the request to IBM HTTP Server, optionally through SSL. The user is authenticated against the user registry in LDAP. Once the user is authenticated, the DB2 Everyplace Sync Server is invoked and data synchronization is carried out between the client and the mirror databases.

20.1.1 DB2 Everyplace

DB2 Everyplace consists of two main components:

- DB2 Everyplace database
- DB2 Everyplace Sync Server

DB2 Everyplace is the database engine installed on the mobile device. DB2 Everyplace Sync Server carries out bi-directional synchronization of data between the database on the mobile device and the source database on the server.

For synchronization of the relational database from the server to the mobile device, the selected data is replicated periodically to a mirror (or mid-tier) database, which acts as a temporary repository for the data. A subset of the mirror data is moved to the database on the mobile device. On the other hand, for synchronization from the mobile device to the server, data is also moved into the mirror database first, then gets replicated to the server periodically.

DB2 Everyplace synchronization is controlled by DB2 Everyplace Sync Server, which has been integrated into WebSphere Everyplace Access' Everyplace Synchronization Server.

In order to set up synchronization, the following entities must be defined beforehand:

- Group
- User
- Subscription set
- Subscription

A user must belong to a group. A subscription must belong to at least one subscription set. A group subscribes to one or more subscription sets for data synchronization.

20.1.2 IBM Everyplace Client

IBM Everyplace Client is a unified client application package for personal digital assistants (PDAs). On Pocket PC devices, Everyplace Client provides a common interface that supports synchronization, security, device management, offline Portal content, offline Domino applications, and DB2 Everyplace database synchronization.

Note: For Palm devicesm almost all the functionality of Everyplace Client is provided but in separate applications.

20.1.3 DB2 Everyplace Sync Server

DB2 Everyplace Sync Server has been integrated into WebSphere Everyplace Access. It is deployed in WebSphere Application Server as a Web module containing several servlets.

DB2e Sync Server is HTTP based. Mobile devices can establish either a wireless or wired connection to synchronize data over the Internet, a wireless network, intranet, local area network (LAN), or wide area network (WAN).

20.2 Before you start

Before you start configuring and testing relational database synchronization, it is important to make sure IBM DB2 Everyplace Server and Everyplace Synchronization Server are running without error. Do the following:

1. Open WebSphere Application Server's Administrative Console by selecting **Start -> Programs -> IBM WebSphere -> Application Server V4.0 -> Administrator's Console**.
2. When the console is ready, make sure IBM DB2 Everyplace Server and Everyplace Synchronization Server are running and there are no error messages in the Event Message window.
3. To verify the DB2 Everyplace Sync servlet is running, open a browser and type the following into the address box:

   ```
   http://hostname/db2e/db2erdb
   ```

   Enter the login information when prompted. You should see the page shown in Figure 20-2 on page 649, which displays information about the sync servlet (MDSSServlet).

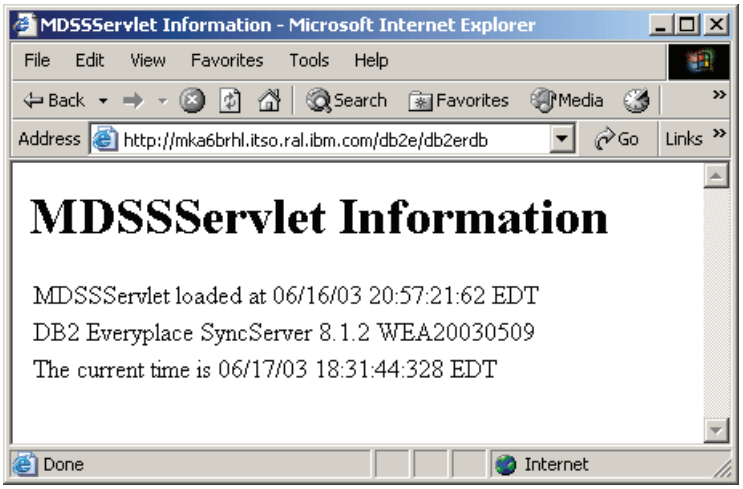


Figure 20-2 DB2 Everyplace sync servlet

If you see this page, you are ready to proceed.

4. If this page is not shown, check the following:

   a. Check whether the following file is present:

      WebSphere_Install_Dir\IBMSyncServer\db2e\Server\lib\dsysync.jar

   b. In WebSphere Application Server Administrative Console, click **IBM DB2 Everyplace Server**. In the right pane, click the **JVM Settings** tab. Check the classpaths and make sure you have this line in the classpaths:

      WebSphere_Install_Dir\IBMSyncServer\db2e\Server\properties

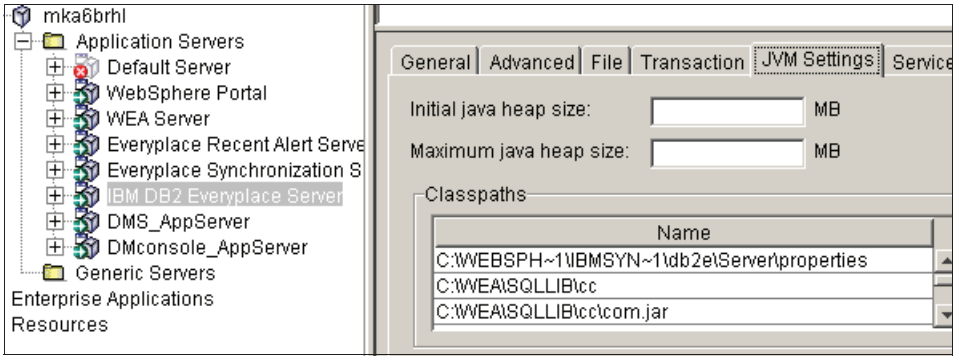


Figure 20-3 Check DB2 Everyplace Server's classpaths

20.3 Server configuration

For relational database synchronization, the server configuration and management activities are handled in two places: user and group management is done in WebSphere Portal, while subscription and subscription sets are managed by the Mobile Devices Administration Center (MDAC).

20.3.1 Creating users and groups

Everyplace Synchronization Server supports multiple synchronization groups. Synchronization groups must be created using WebSphere Portal's administration portlets. A user must be a member of a synchronization group in order to synchronize with the DB2 Everyplace Sync Server. When these synchronization groups are in use, their names are stored in the file DSYLDAP.properties, located at:

WebSphere_Install_Dir/IBMSyncServer/db2e/Server/properties/com/ibm/mobile services

under the key "SYNCGROUP".

To use the relational database adapter, a special user and group configuration is required:

- Each group must have a prefix of DB2e (case-sensitive), such as *DB2e_myemployees*.
- DB2e groups should not be members of any synchronization group defined in the DSYLDAP.properties file.
- Relational Database synchronization users must be a member of one of the synchronization groups defined in the DSYLDAP.properties file.
- Relational Database synchronization users must also be a member of one group that begins with DB2e.
- Relational Database synchronization users can belong to only one DB2e group, but the user may belong to other groups in the Portal environment including other synchronization groups.

The following steps demonstrate how to create groups and users for relational database synchronization:

1. To create a synchronization group:
   c. Log in to the WebSphere Everyplace Access server as an administrator, for example wpsadmin.
   d. Select the **Portal Administration** page group.

e. Select the **Users and Groups -> Manage Groups** option.

f. Enter the synchronization group name in the Group Name field, for example `SyncGroup`.

g. Click **Create Group**.

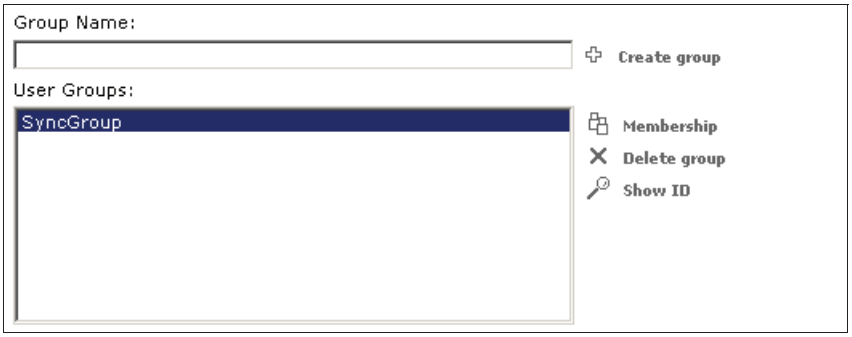


Figure 20-4 Create synchronization group

The group should be created and appear in the User Groups field.

h. Follow similar steps to create more synchronization groups.

Important: The IBM DB2 Everyplace Server uses the group “SyncGroup” by default. If you choose to use a different name, for example “AllSyncUsers”, you will need to start up MDAC using your synchronization group, and then restart the IBM DB2 Everyplace Server.

The group name can be confirmed by looking at the last line of the following file:

WebSphere_Install_Dir\IBMSyncServer\db2e\Server\properties\com\ibm\mobileservices\DSYLDAP.properties

For example: `SYNCGROUP=AllSyncUsers`

We recommend having a parent synchronization group that contains all synchronization users, and to use this group when starting MDAC.

After creating the desired synchronization group(s), users who are expected to synchronize with the database(s) must be added to the synchronization group(s).

2. To add users to synchronization group(s):

   a. Log in to the WebSphere Everyplace Access server as an administrator, for example wpsadmin.

b. Select the **Portal Administration** page group.

c. Select **Users and Groups -> Manage Groups**.

d. Search to find the synchronization group.

e. Select the synchronization group.

f. Click **Membership**.

g. Select **Add users to group**.

h. Use the Name is field to search for users.

i. Select the user(s) to add from the Search Results field.

j. Click **Add to group**.

As mentioned earlier, relational database synchronization users must belong to both synchronization group(s) and DB2e groups (relational database adapter group).

3. To create a relational database adapter group:

   a. Log in to the WebSphere Everyplace Access server as an administrator, for example wpsadmin.

   b. Select the **Portal Administration** page group.

   c. Select **Users and Groups** -> **Manage Groups**.

   d. Enter the synchronization group name in the Group Name field, for example `DB2e_employee`.

   Note: The group name must begin with DB2e (case sensitive).

   e. Click **Create Group**.

   Tip: After adding or removing users to a DB2e group, we recommend resetting that user in MDAC.

4. To add users to relational database adapter group(s):

   a. Log in to the WebSphere Everyplace Access server as an administrator, for example wpsadmin.

   b. Select the **Portal Administration** page group.

   c. Select **Users and Groups -> Manage User Groups**.

   d. Search to find the synchronization group.

   e. Select the synchronization group.

f. Click **Membership**.

g. Select **Add users to group**.

h. Use the Name is field to search for users.

i. Select the user(s) to add from the Search Results field.

j. Click **Add to group**.

Note: For relational database synchronization to work, user(s) must exist in both synchronization group(s) and one relational database adapter group.

20.3.2 Creating subscription and subscription set

Synchronization-related information for a relational database is defined as a subscription. Subscriptions are grouped into subscription sets that the user groups can subscribe to. We have our users and groups, so now we must define the necessary subscriptions and subscription sets to make the synchronization work.

1. To create a subscription:

   a. Click **Start -> Programs -> IBM Everyplace Synchronization Server -> Launch MDAC**.

   The DB2 Control Center automatically opens. As the MDAC retrieves the users and groups information from LDAP, a WebSphere Portal Server LDAP Logon window pops up and asks for the necessary information to retrieve data from LDAP (see Figure 20-5 on page 654).

   Note: Mobile Devices Administration Center starts the DB2 Control Center when starting. Therefore you may be prompted for a DB2 administrator ID and password the first time you launch the Mobile Devices Administration Center. This is most likely to occur when the WebSphere Application Server database is configured as a remote database. You can enter the login information, or cancel the dialog until it disappears.

   Enter the LDAP administrator's user name and password, and specify the synchronization groups defined through WebSphere Portal Server. We suggest having a parent group that contains all synchronization users, However multiple synchronization groups may be entered, separated by semicolons. Click **OK**.

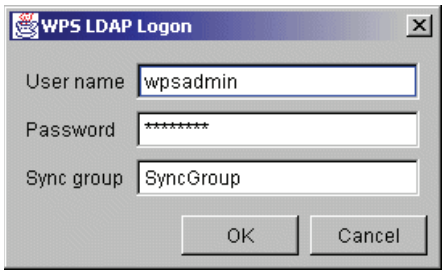


Figure 20-5 WebSphere Portal Server LDAP Logon window

Note: The Sync group name is case-sensitive.

The Mobile Devices Administration Center launches.

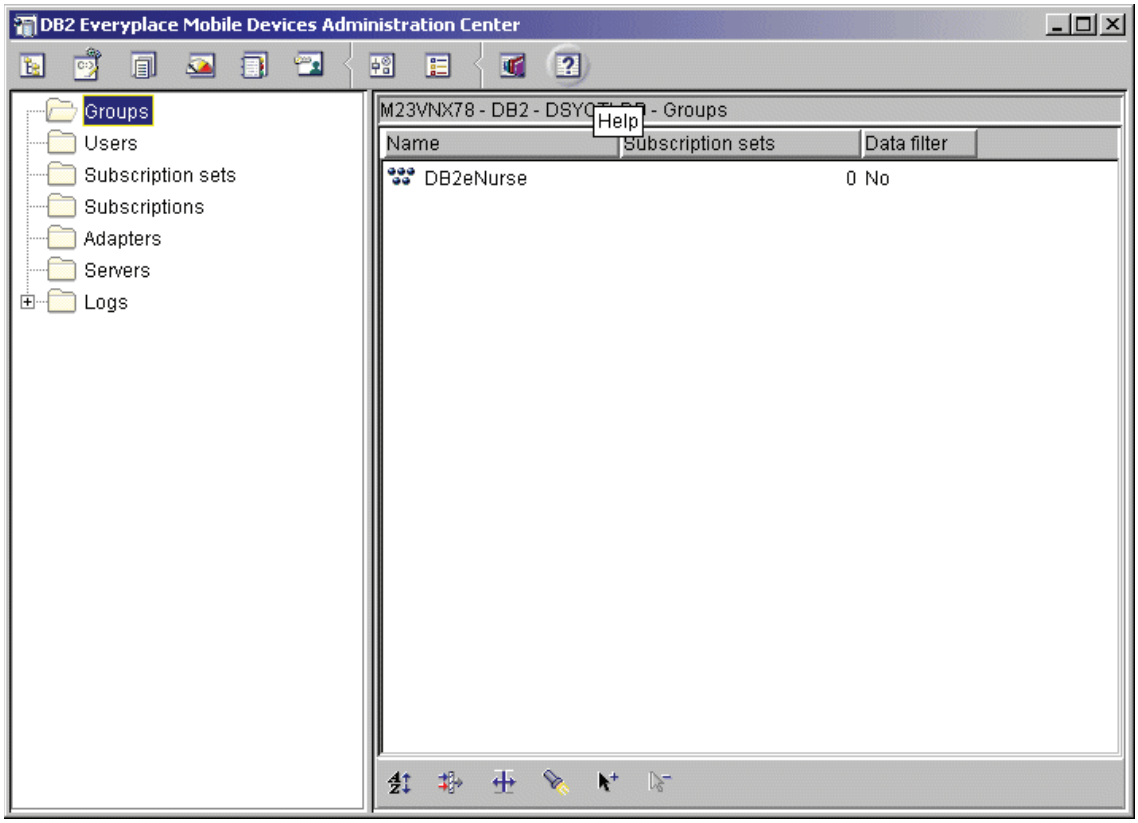


Figure 20-6 Mobile Device Administration Center

b. Click the **Groups** and **Users** folders to check that the DB2e group(s) and user(s) are imported into the MDAC.

c. Right-click **Subscription -> Create -> Table Subscription -> JDBC Subscription**.

Note: JDBC Subscription is used in this scenario. Other types of subscription will be covered in later sections.

The Create JDBC Subscription wizard opens.

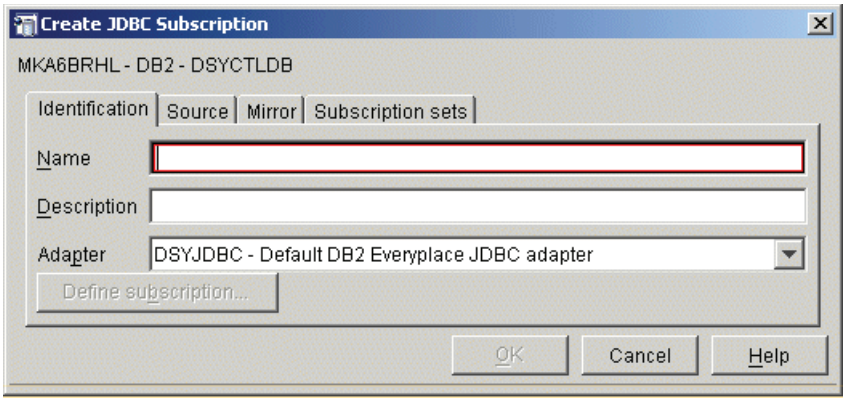


Figure 20-7 Create JDBC Subscription wizard

d. Enter the name of the subscription and, optionally, a description. Click the **Source** tab.

e. In the Source tab, the source database information should be provided.

   i. In the Database URL field, enter the source database URL, for example `jdbc:db2:VNURSE`. Or you can also use the **...** button besides this field to browse and choose the source database (only for DB2 databases).

   ii. In the Driver field, use the down arrow to choose the desired JDBC driver from the list.

   iii. In the User ID field, provide the user ID that can be used to access this database.

   iv. In the Password and Verify Password fields, enter the password.

   v. You can click **Test Connection** to verify the connection to the database.

   Figure 20-8 on page 656 shows the source database configured.

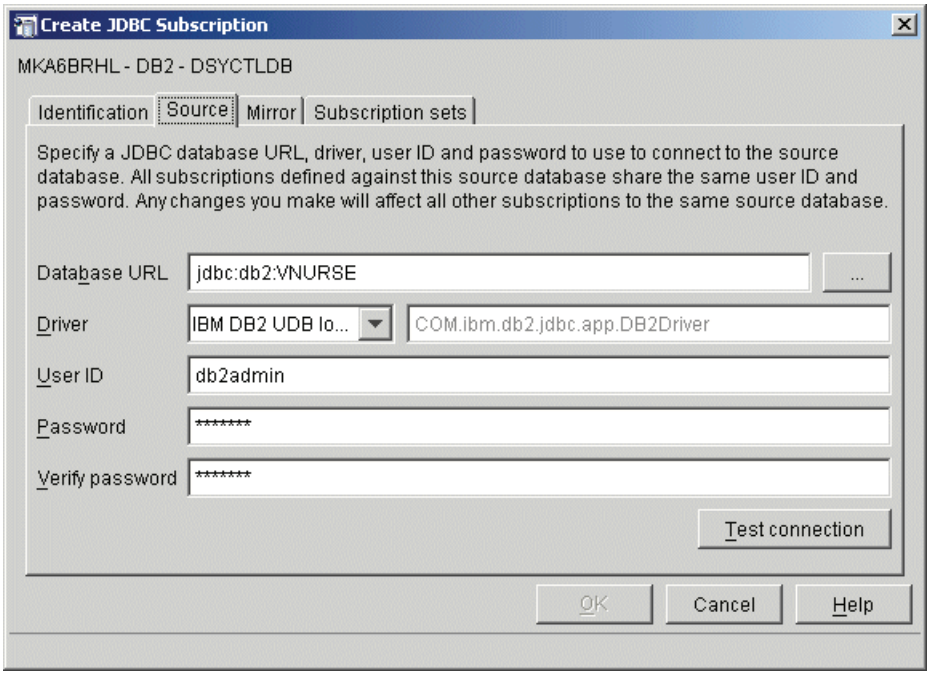


Figure 20-8 Create JDBC Subscription - define source database

f. When done, click the **Mirror** tab.

   In this tab, the mirror database information should be provided.

   i. In the Database URL field, enter the mirror database URL, for example `jdbc:db2:M_VNURSE`. Or you can also use the ... button besides this field to browse and choose the source database.

   ii. In the User ID field, provide the user ID that can be used to access this database.

   iii. In the Password and Verify Password fields, enter the password.

   iv. You can click **Test Connection** to verify connection to the database.

   v. When done, click the **Identification** tab.

   Figure 20-9 on page 657 shows the mirror database configured.

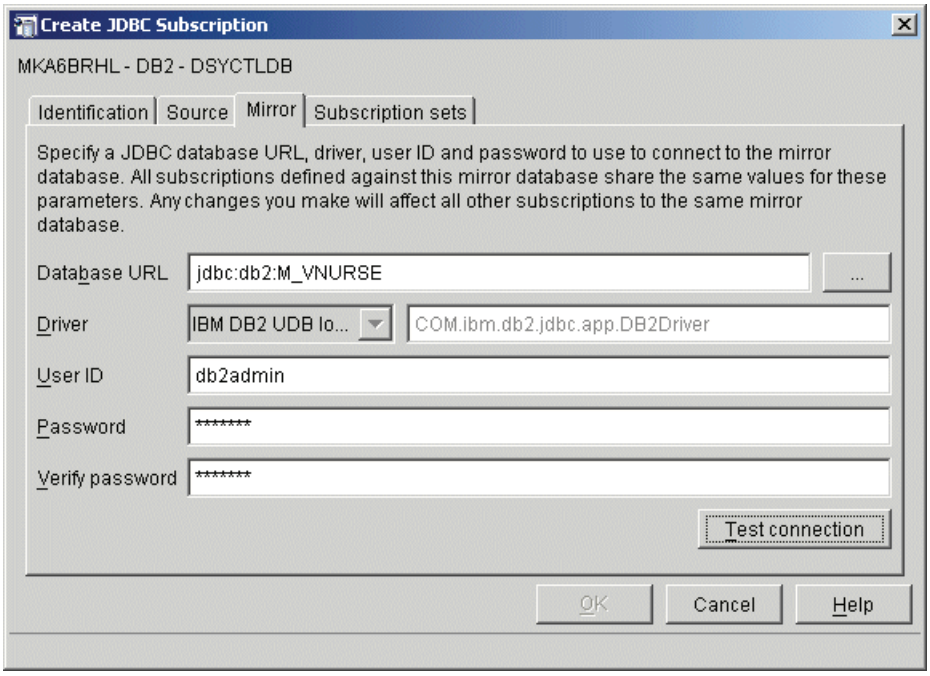


Figure 20-9 Create JDBC Subscription - define mirror database

g. On the Identification tab, click the **Define subscription** button.

   i. The Define Replication Subscription window is shown. Click **Add**.

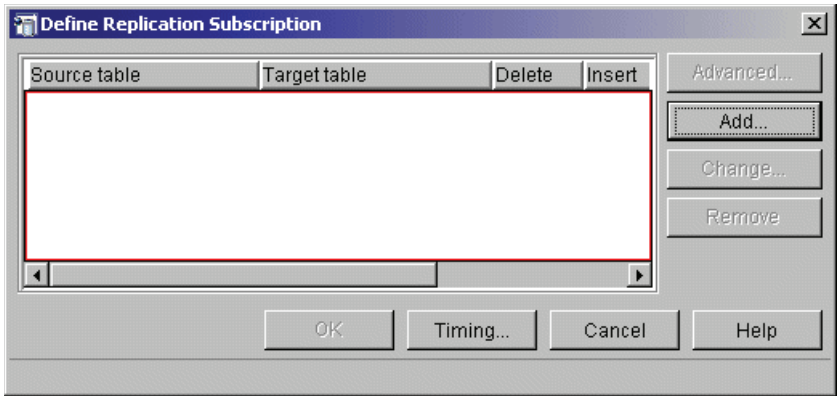


Figure 20-10 Create JDBC Subscription - define subscription

   ii. The Add Table window is displayed.

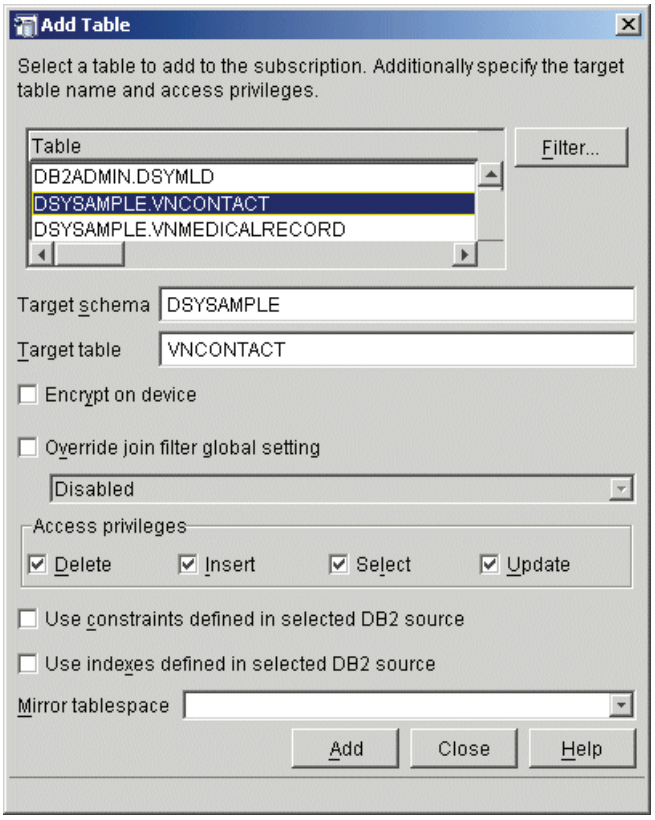


Figure 20-11 Create JDBC Subscription - define subscription - add table

iii. Select the table to be synchronized. The Target schema and Target table fields are filled in automatically; accept the default or change it to anything desirable.

iv. Click the appropriate **Access Privileges**. Click **Add**.

v. If there is more than one table to be synchronized, select more tables to add. When done, click **Close**.

The tables have been added to the subscription now.

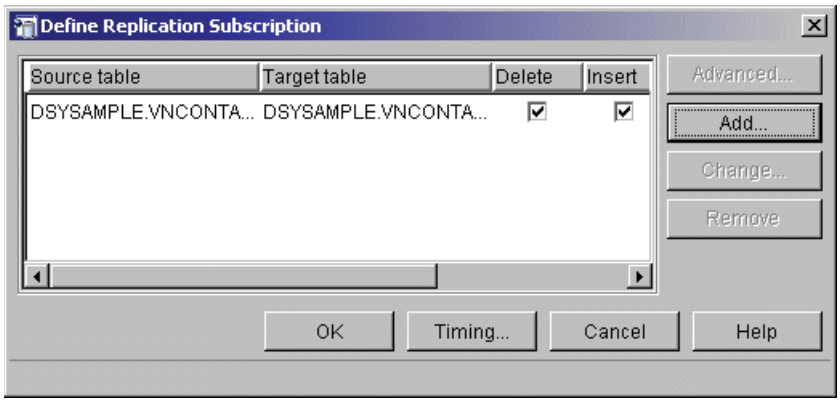


Figure 20-12 Create JDBC Subscription - define subscription

vi. Click the **Timing** button to adjust the replication frequency.

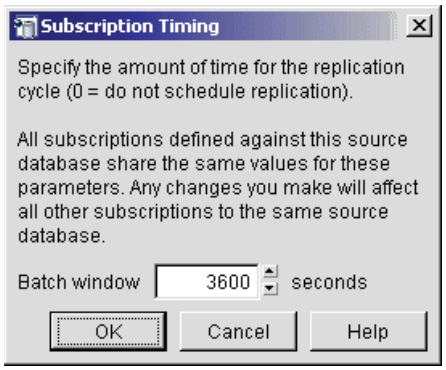


Figure 20-13 Adjust replication frequency

vii. Click **OK** and **OK** to return to the Create JDBC Subscription wizard.

h. Click the **Subscription sets** tab.

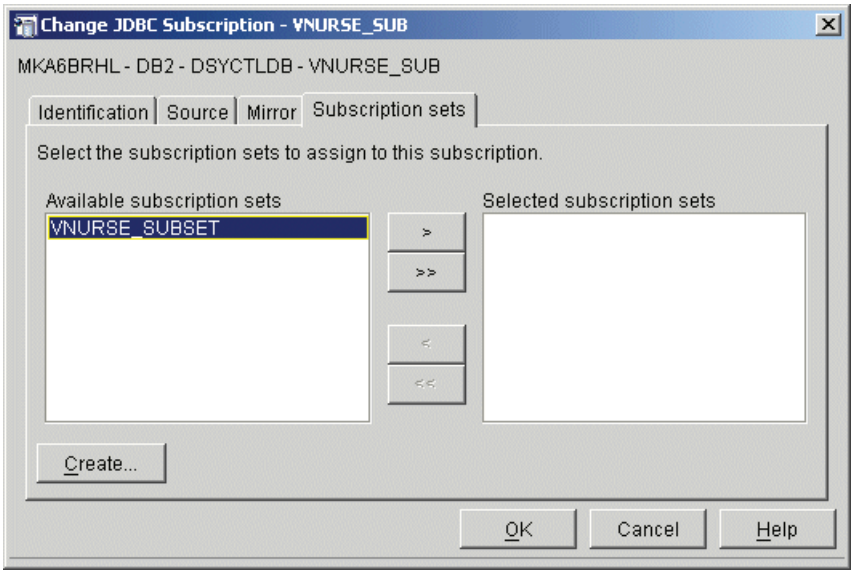


Figure 20-14 Create JDBC subscription - define subscription set

i. Choose from the available subscription sets in the left pane and click **>** to put it into the right pane. The subscription sets shown in the right pane will include this newly created subscription.

ii. You can also choose to create a new subscription set by clicking the **Create** button. The Create Subscription Set wizard will be covered later.

i. Now all the information have been gathered for the subscription. Click **OK**.

j. The new subscription should appear in the right pane when clicking the **Subscriptions** folder. It can always be altered by right-clicking the subscription and choosing **Edit**.

2. To create a subscription set:

a. With MDAC open, right-click **Subscription sets -> Create -> Table Subscription -> JDBC Subscription**.

The Create Subscription Set wizard opens.

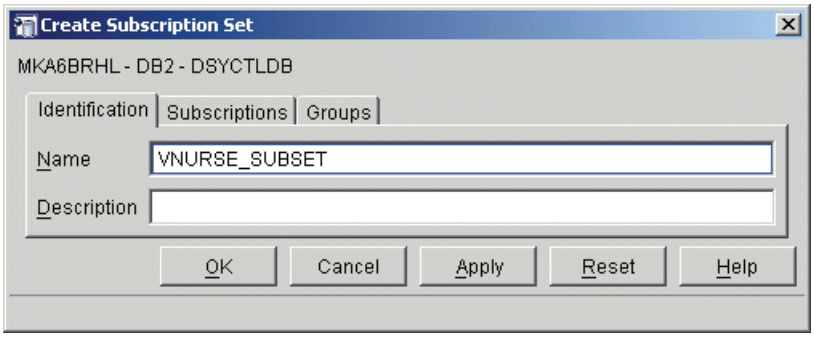


Figure 20-15 Create Subscription Set wizard

b. On the Identification tab, enter the name and description for this subscription set. Click the **Subscriptions** tab.

c. Choose from the available subscriptions in the left pane and click **>** to include it in this subscription set, or click **>>** to choose all.

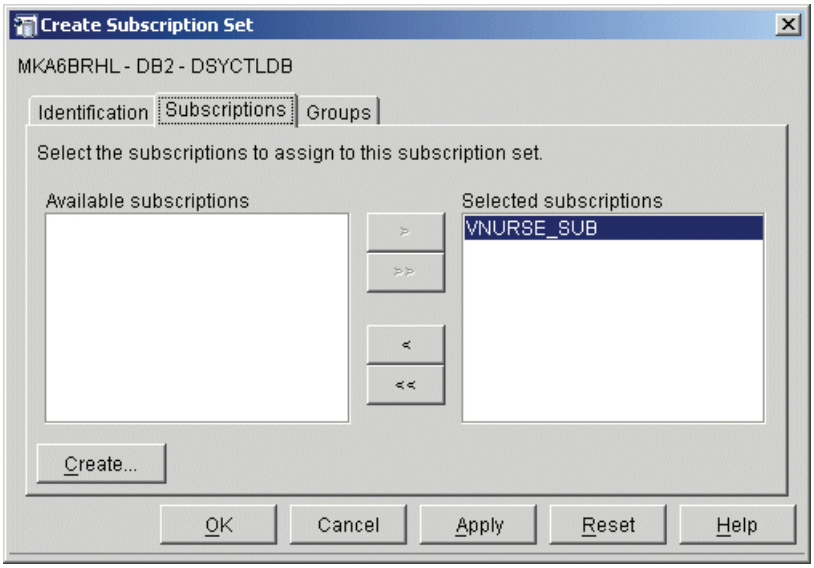


Figure 20-16 Create Subscription Set - select subscription to include

d. You can also click the **Create** button to create a new subscription, following the steps described in the previous section.

e. When done, click the **Groups** tab.

f. Choose from the available groups in the left pane and click **>** to subscribe it to this subscription set, or click **>>** to select all of the groups.

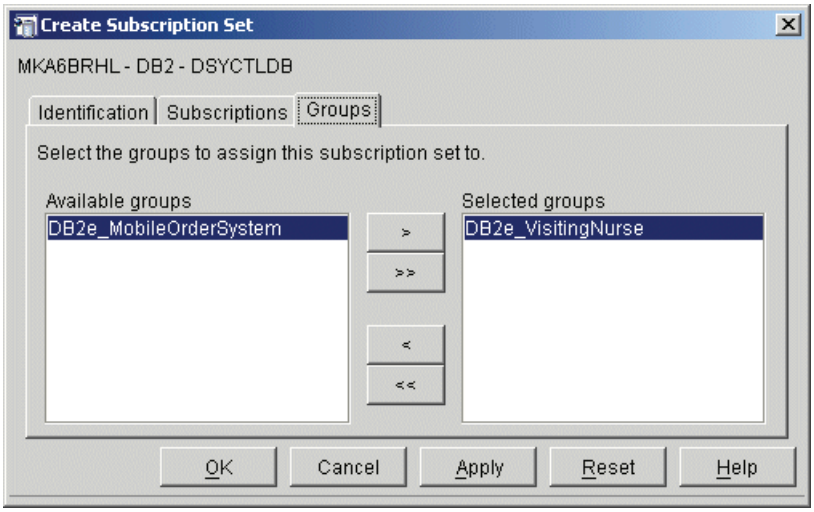


Figure 20-17 Create Subscription Set - select groups

g. Enough information has been collected to create a subscription set. Click **OK**.

h. The newly created subscription set should appear in the right pane of MDAC when clicking the **Subscription sets** folder. It can be easily modified or examined by right-clicking the subscription set and choosing **Edit**.

i. Now click the **Groups** folder. In the right pane, for each group, the number appearing in the Subscription sets column indicates the number of subscription sets this group is subscribing to.

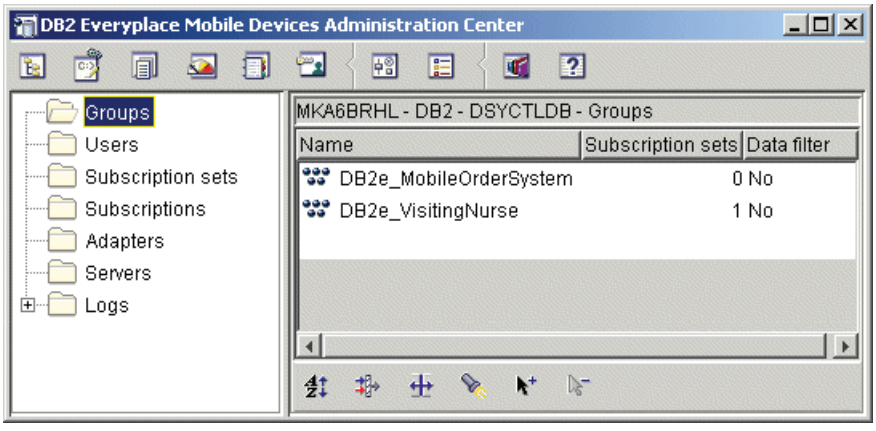


Figure 20-18 MDAC - groups

The association with the subscription sets can be easily changed by right-clicking the group and choosing **Edit**. Select or deselect subscription sets in the Subscription sets tab.

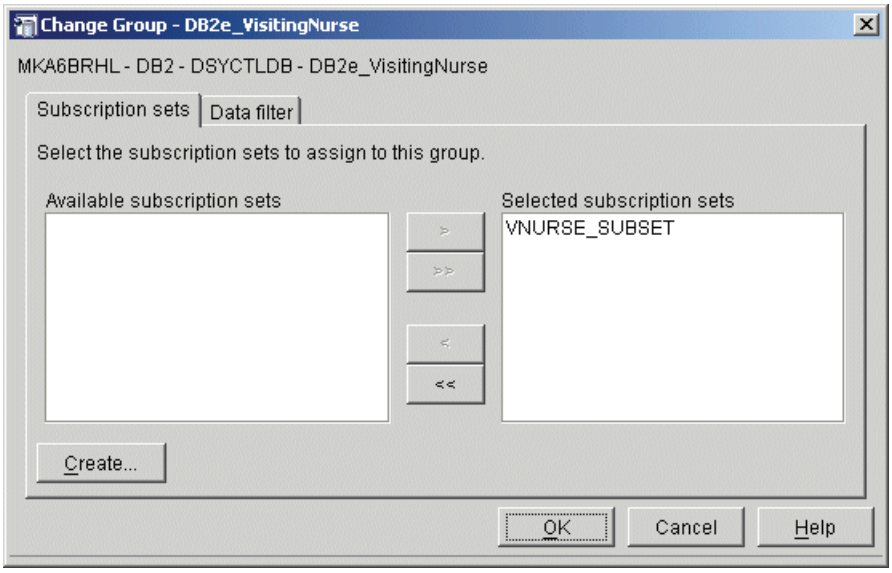


Figure 20-19 MDAC - groups - change subscription sets

20.4 Binding LDAP and MDAC

As discussed in previous sections, users and groups are created in WebSphere Portal and maintained in LDAP. The Mobile Devices Administration Center or MDAC retrieves this information from LDAP and keeps a copy in its control database DSYCTLDB.

In MDAC, it is possible to refresh this binding if any changes take place in WebSphere Portal.

- To refresh group information, right-click the **Groups** folder and select **Refresh WPS LDAP groups**.
- To refresh user information, right-click the **Users** folder and select **Refresh WPS LDAP users**.

By doing the refreshing, if new users or groups are added, they will be brought into the MDAC and its control database.

If any user is removed from the sync group in WebSphere Portal, upon refreshing the LDAP users, MDAC will warn you of the inconsistency and prompt you to delete the user from MDAC's control database.

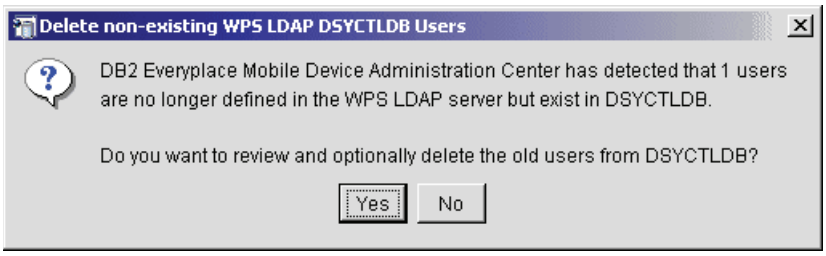


Figure 20-20 MDAC prompts for inconsistency in user information

Click **Yes** to delete the user from MDAC.

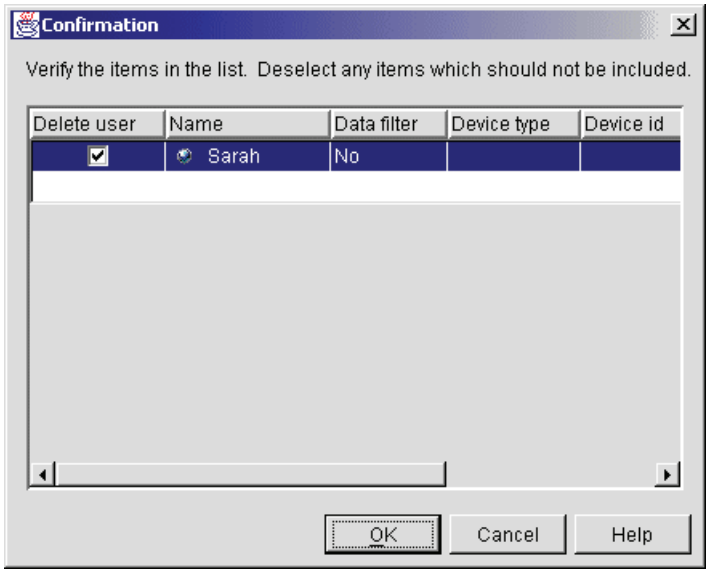


Figure 20-21 Confirmation for deleting user from MDAC

Click **OK** to confirm the deletion.

If any DB2e group is removed from WebSphere Portal, upon refreshing the Portal Server LDAP groups, MDAC will warn you of the inconsistency and prompt you to delete the group from MDAC's control database.

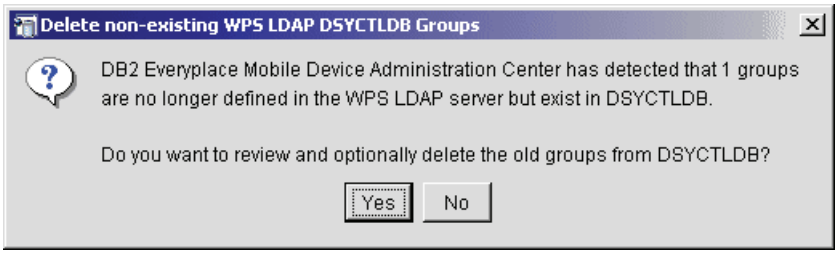


Figure 20-22 MDAC prompts for inconsistency in group information

Click **Yes** to delete the group from MDAC.

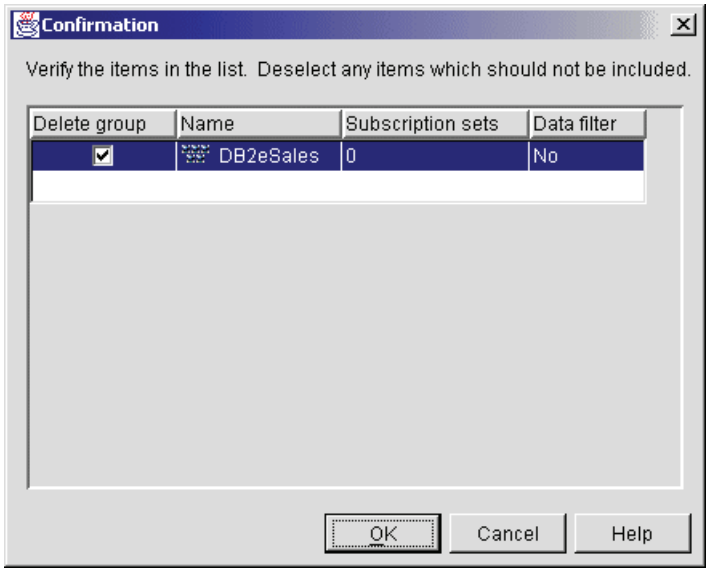


Figure 20-23 Confirmation for deleting user from MDAC

Click **OK** to confirm the deletion.

20.5 DB2 Everyplace Client configuration

The DB2 Everyplace Client is a component of the Everyplace Client. For general information about the Everyplace Client refer to 1.2.4, “Everyplace Client” on page 14. For information about the installation and configuration of the Everyplace Client refer to Chapter 4, “Everyplace Client” on page 81.

Follow these steps to configure database synchronization for a particular user, using the DB2 Everyplace Client:

1. Start the DB2 Everyplace Client. On Pocket PC devices you can start the client using the integrated interface or by selecting **Start -> Programs -> DB2 Everyplace SyncServer -> isyncui**. On Palm devices you have to select the **IBM Sync** icon.
2. On Pocket PC devices, select **File -> Settings** to open the Settings window. On Palm devices select **Options -> Settings**.
3. For both platforms, enter meaningful values for the Server IP, Port number, User ID and Password fields.
4. (Optional) Select **Advanced** on the Settings window. You can set the SSL support, the timeout for the synchronization session, the detail level for the log file and the target directory for the database files. Also for Palm devices, you can select if you want to start the database synchronization when you start the HotSync synchronization. Click **OK** to close the Advanced settings window.
5. Click **OK** to close the Settings window.
6. To start the database synchronization, select **Synchronize**. You will see the synchronization progress window.
7. When the synchronization ends, you can review the logs for detailed information about the synchronization session. On Pocket PC devices select **File -> View Log**. On Palm devices select **Log**.

20.6 Sample application synchronization

IBM DB2 Everyplace provides a sample application that is installed automatically: the Visiting Nurse (VNURSE) application. This application simulates the scenario where a nurse visits some patients and takes their medical data. The nurse uses the VNURSE application to retrieve a patient's information, such as past medical record and contacts. The nurse also uses this application to record the newly collected medical data, and can synchronize with the back-end database to upload or download the latest updated information about the patients.

To get familiar with this sample application:

- Select **Programs -> DB2 Everyplace Samples -> VNURSE** for Pocket PC devices.
- Select **Nurse** in Applications for Palm devices.

To work with this sample application, there are also subscription and subscription sets in the Everyplace Synchronization Server. The subscription is JDBCSUB1 and the subscription set is SUBSCRIPTION_SET1.

Explore the defined subscription and subscription sets in MDAC to get familiar with them.

To try out the sample application VNURSE synchronization:

1. Create a synchronization group and add a user to it.
2. Create a relational database adapter group (DB2e group) and add the same user to this group as well.
3. In MDAC, add the DB2e group to subscription set SUBSCRIPTION_SET1.
4. Configure the Everyplace Client on the device as explained in 20.5, “DB2 Everyplace Client configuration” on page 665.
5. Start the DB2e ISync Client on the device.
6. Select **Synchronize**, as shown in Figure 20-24. The synchronization starts.

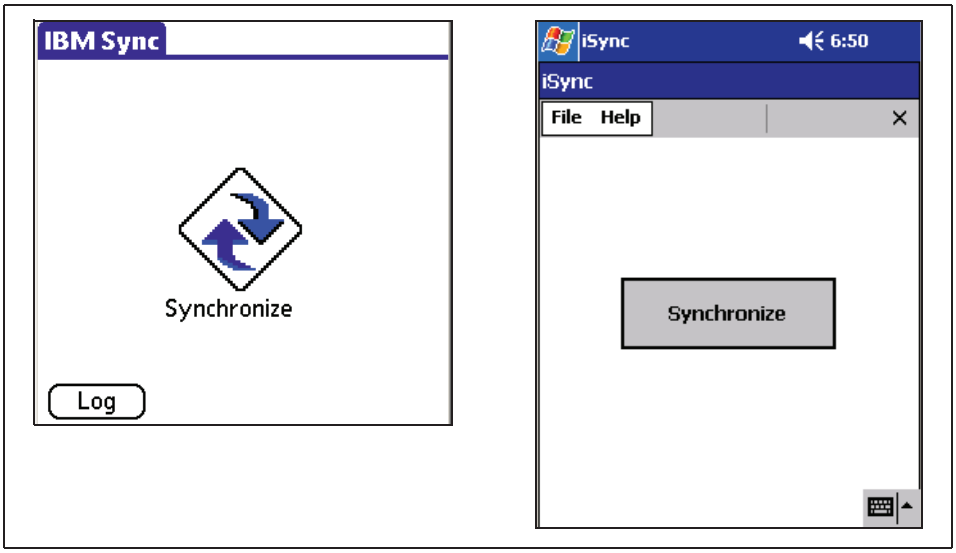


Figure 20-24 Executing DB2 Everyplace Sync

7. When the synchronization is completed, you see a message as shown in Figure 20-25 on page 668.

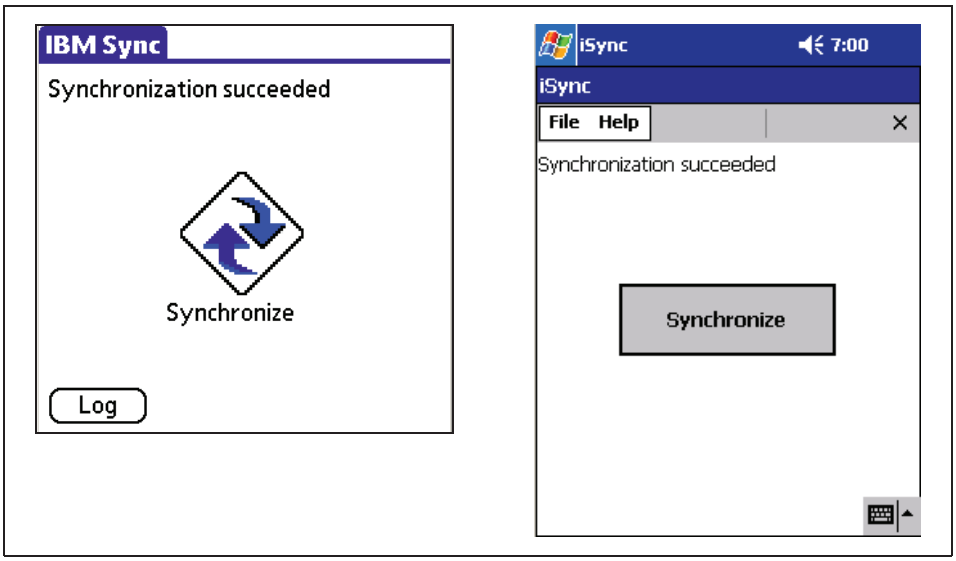


Figure 20-25 DB2 Everyplace Sync completes

8. If for any reason the synchronization failed, select **Log** on Palm devices or **File -> View logfile** on Pocket PC devices to get some information about the failure.
9. Try adding a new medical record using the VNURSE application, then perform the synchronization again.
10. Use the DB2 Control Center to view whether the new record appears in VNMEDICALRECORD table of the VNURSE database.

20.7 Verify the synchronization

After completing the synchronization, you may verify the synchronization results. On the server side, use the usual ways to query the back-end database to verify the changes are there. On the client, if you have an application that can retrieve and display the data from the databases, such as the sample VNURSE application, use it to verify the results.

Another generic way to verify the results is to use the DB2eCLP tool that comes with DB2 Everyplace:

1. Click **Programs -> DB2 Everyplace Samples -> DB2eCLP** on Pocket PC devices, or select **DB2eCLP** on Palm devices.
2. Select **OK** on the Welcome window. The DB2eCLP interface is shown.

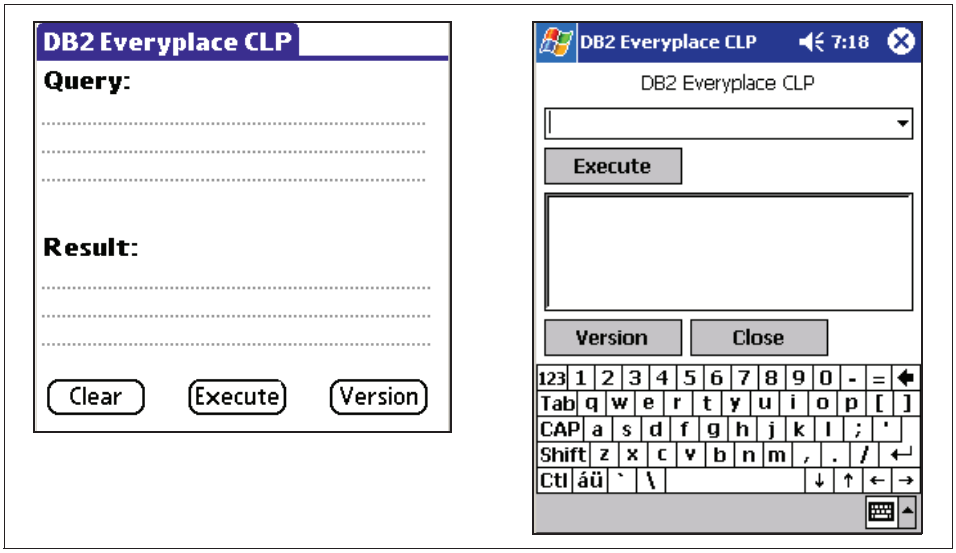


Figure 20-26 DB2eCLP

3. On Pocket PC, type in the top pane:

   ```
   connect to \database_path\
   ```

   Make sure the database path is followed by a backward slash (“\”). Otherwise, the connection will not be established. Then select **Execute**. On Palm devices this step is not necessary because, by default, the database is stored in only one place.

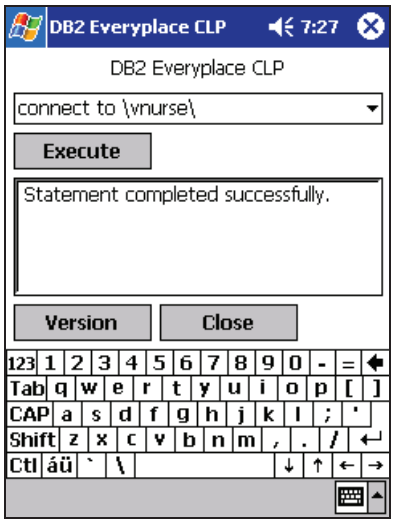


Figure 20-27 Connecting to the database in Pocket PC

4. Type the SQL statement in the top pane, for example:

   ```
   select * from vnmedicalrecord where patientid='900000001'
   ```

   Click **Execute** and the query results will be displayed in the bottom pane for you to verify.

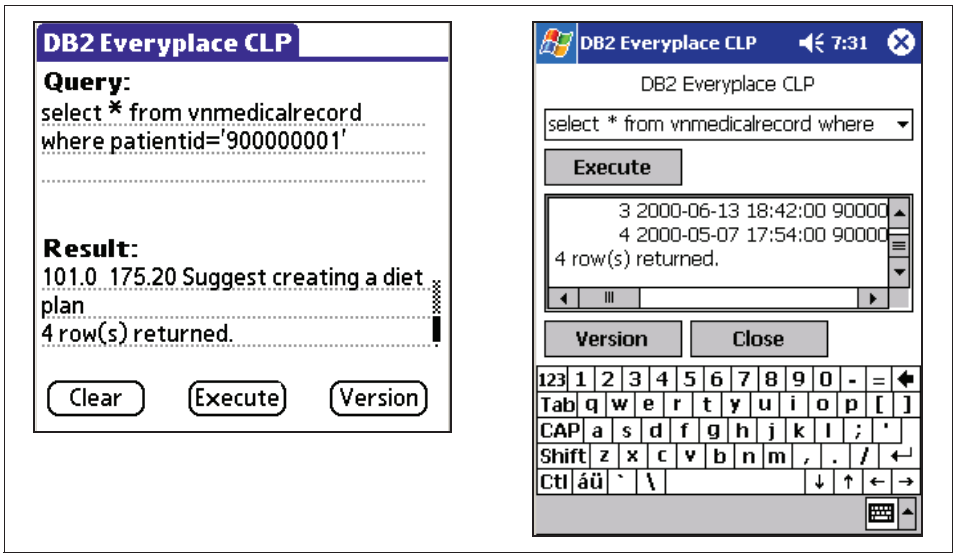


Figure 20-28 Using DB2eCLP

5. Type other necessary SQL commands to verify.
6. When done, make sure to exit the DB2eCLP by clicking the **Close** button to release the database connection. Otherwise, other DB2e applications will not be able to operate.

20.8 Synchronization using SSL

To prevent synchronization data from being exposed to unintended parties, it is often desirable to turn on security between the server and the client. In the case of DB2 Everyplace Synchronization, security could be enabled between:

- The HTTP server and the mobile client
- The HTTP server and the Application Server

For more information about security on the WebSphere Application Server platform, refer to the redbook *IBM WebSphere V4.0 Advanced Edition Security,* SG24-6520.

20.8.1 Enable server security

Refer to Chapter 5, “Everyplace Client secure connections” on page 115 for detailed steps about how to configure the SSL support between the HTTP Server and the mobile device.

20.8.2 Enable client security

Refer to Chapter 5, “Everyplace Client secure connections” on page 115 for information about how to configure SSL security using the Everyplace Client.

20.9 Synchronization with remote DB2 databases

In an enterprise environment, the databases are often stored on separate machines from the Everyplace Synchronization Server. This section demonstrates how to configure a subscription to synchronize with a remote DB2 database.

1. Before you start, make sure to run the **`db2jstrt`** command on the remote database machine so that the DB2 server can accept JDBC net connections.
2. Open the Mobile Administration Center.
3. Right-click **Subscriptions** and choose **Create -> Table subscription -> JDBC subscription**.

4. The Create JDBC Subscription wizard opens. Enter a name for the subscription and, optionally, a description. Click the **Source** tab.
5. Click the down arrow besides the Driver field and choose **IBM DB2 UDB Remote**. The Database URL field will change to a different format automatically.

   Fill in the Database URL field, where the host is the host name or IP address of the machine where the remote data source resides, and the port number is 6789 by default. Enter the database name.

   Specify the user name and password that can access the source database.

   Click the **Mirror** tab.

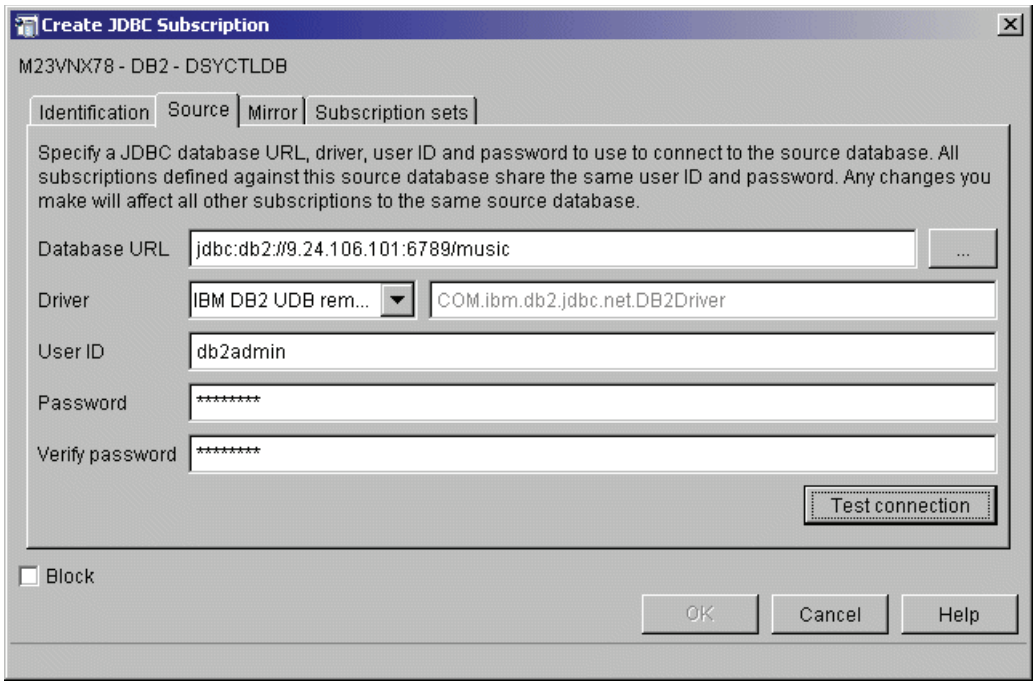


Figure 20-29 Specify remote data source

6. In the Mirror tab, specify the local mirror database and provide the user name and password used to access this database. Click the **Subscription sets** tab.
7. In the Subscription sets tab, specify the subscription sets this subscription should belong to. Click the **Identification** tab.
8. On the Identification tab, click the **Define subscription** button.
9. The Define Replication Subscription window is shown. Click **Add**.

10. Add the tables to be synchronized and click **Close**.
11. Modify the replication interval if necessary and click **OK** and **OK**.

    By now, the subscription to a remote data source is configured.
12. Click the **+** besides the Logs folder and click **Replication**. Sometimes it is necessary to refresh by right-clicking **Replication** and choosing **Refresh**. Make sure the replication for the newly created subscription is carried out successfully.

    You can also use the Control Center to view the contents of the mirror database to see whether the data has been replicated.
13. If the replication has no problem, this subscription is ready for use.

20.10 Types of subscription

In the previous sections, we use JDBC subscription for data synchronization. There are three possible types of subscriptions for relational database tables:

- DataPropagator™ subscription

  DataPropagator subscription is used only with DB2 back-end data sources. The DB2 data source is left intact, but separate steps must be taken to enable replication between source and mirror databases.
- JDBC subscription

  JDBC subscription provides users with access to data in any source database with a JDBC interface. Triggers are inserted into the source database.
- Upload subscription

  Upload subscription only allows the user to directly insert rows into a table on a source database. The source table may reside on any database that supports JDBC. There is no mirror database involved.

20.10.1 DataPropagator subscription

With DataProgator subscriptions, DB2 Everyplace Sync Server uses DB2 DataPropagator to replicate data between the source DB2 database and the mirror DB2 database. Figure 20-30 on page 674 illustrates this process.

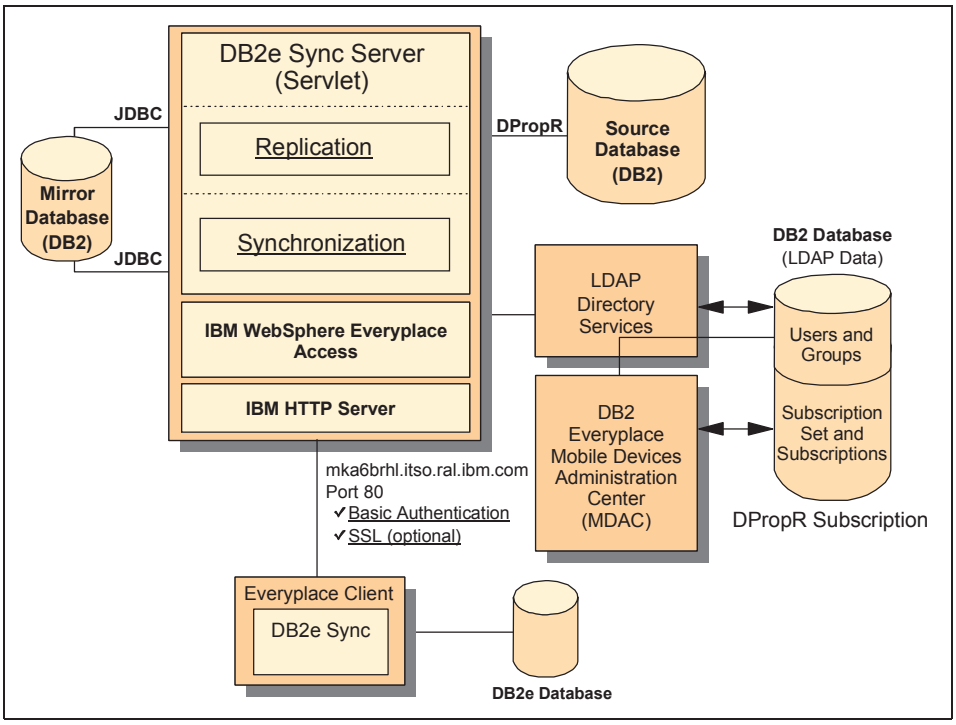


Figure 20-30 DataPropagator subscription - sample scenario

In the whole synchronization scenario, different subscription types only affect replications. Client configuration and usage steps remain the same.

Note: DataPropagator subscriptions can only be used with DB2 back-end databases.

Before creating DataProgator subscriptions, the source database table must be defined as a replication source in DB2:

1. Open the DB2 Control Center and Mobile Devices Administration Center by clicking **Start -> Programs -> IBM Everyplace Synchronization Server -> Launch MDAC**.
2. In the DB2 Control Center, expand the source database (for example, VNURSE) and select the **Tables** folder to list the available tables.
3. Right-click the synchronization source table, for example **VNMEDICALRECORD**.
4. Select **Define as replication source -> PE**.

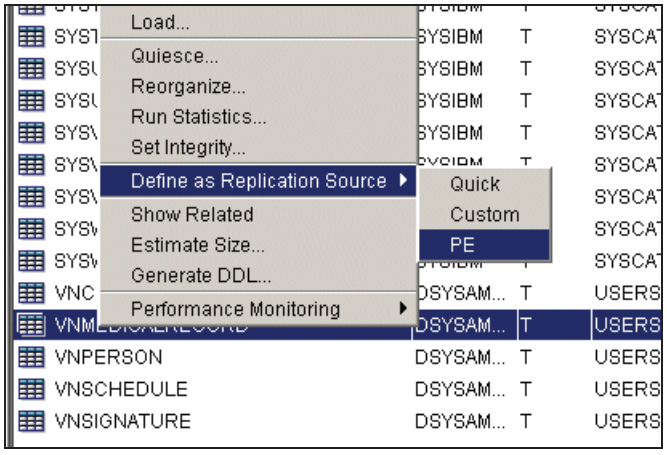


Figure 20-31 Define source database table as replication source

5. Select **Run Now**.

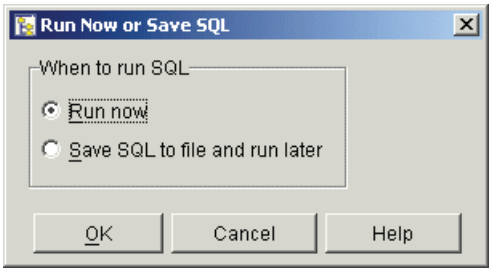


Figure 20-32 Run Now

6. Click **OK**.
7. Verify the replication source has been created by clicking the **Replication Sources** folder.

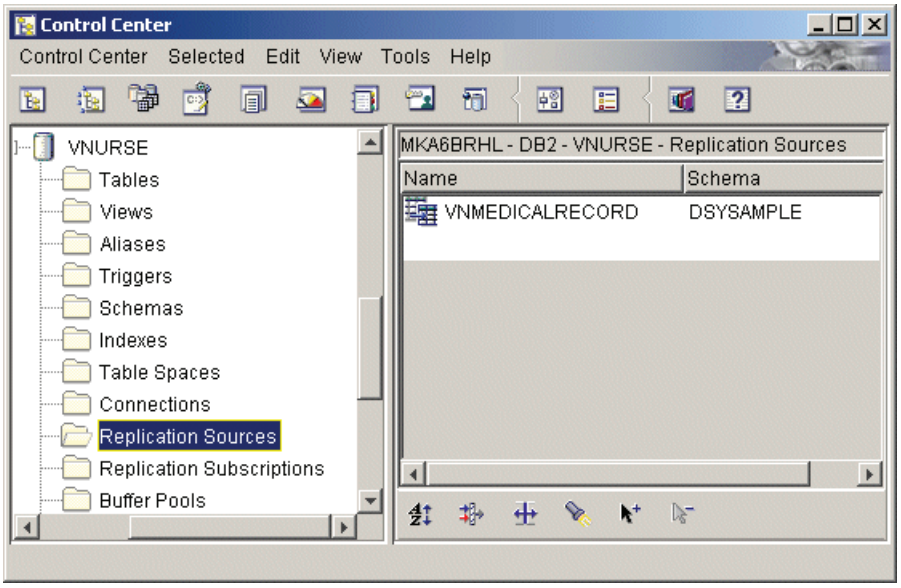


Figure 20-33 Table added as replication source

Now we can define the subscription.

8. Stop the DB2e SyncServer using the WebSphere Application Server Administrator Console, to avoid the warning message shown in Figure 20-34.

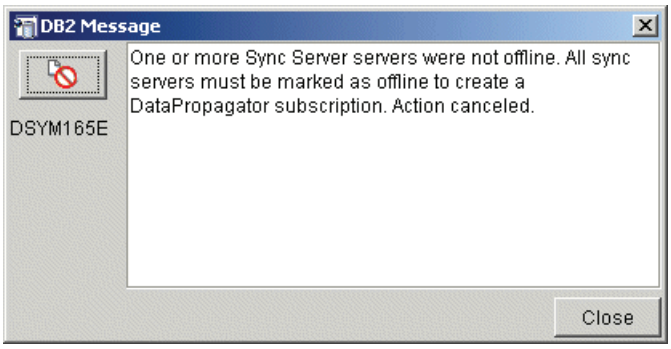


Figure 20-34 Warning about online sync server

9. In the Mobile Devices Administration Center, select **Subscription -> Create -> Table Subscription -> DataPropagator subscription**.

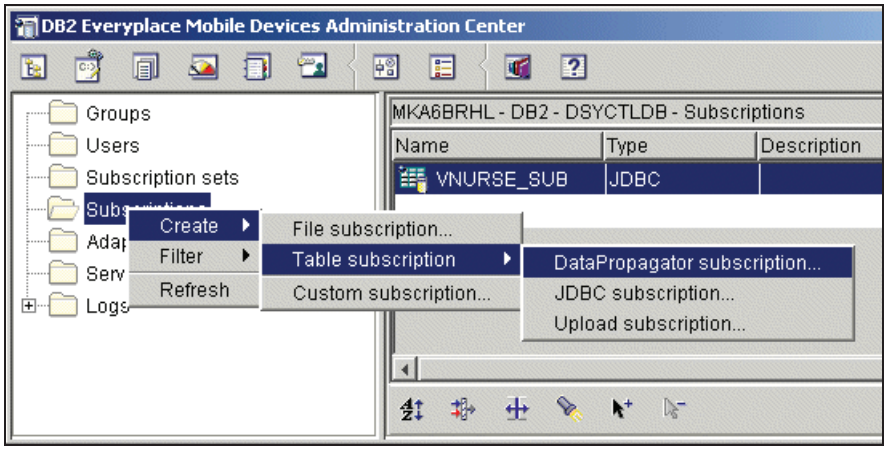


Figure 20-35 To create DataPropagator subscription in MDAC

10. The Create DataPropagator Subscription wizard opens. In the Identification tab, type a name for this subscription, for example `MedicalRecord`. Optionally, enter a description for this subscription.

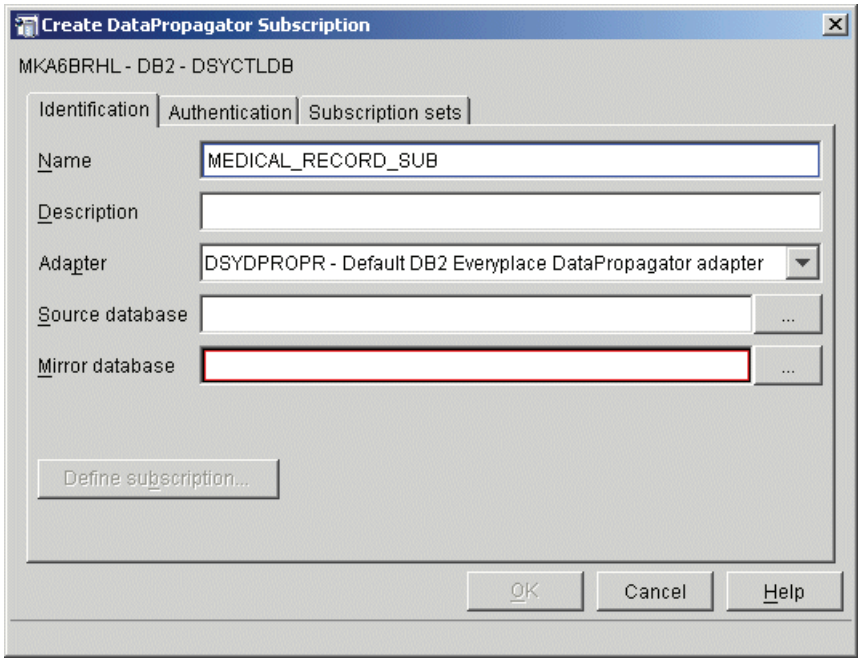


Figure 20-36 Create DataPropagator Subscription wizard

11. Click the **...** button next to the Source database field. Choose the source database from the list, and click **OK**.

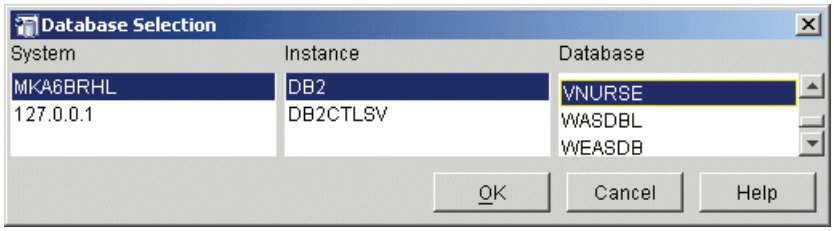


Figure 20-37 Choose source database

12. Click the **...** button next to the Mirror database field. Choose the mirror database from the list, and click **OK**. If a mirror database was not created earlier, click **Create** to launch the Create Database wizard.

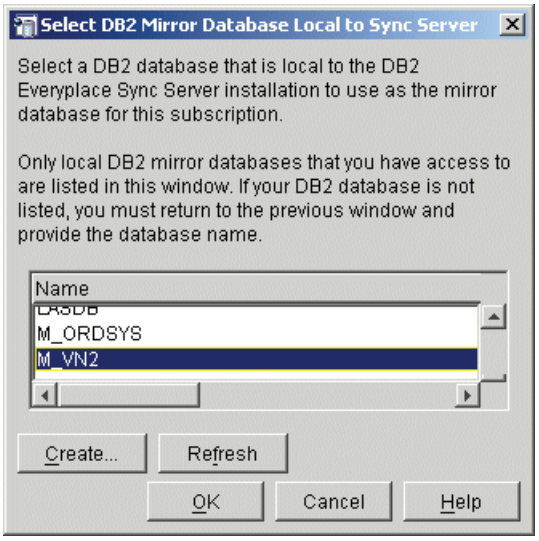


Figure 20-38 Set mirror database

13. Click the **Authentication** tab. Enter the user ID and password information for both source and mirror databases.

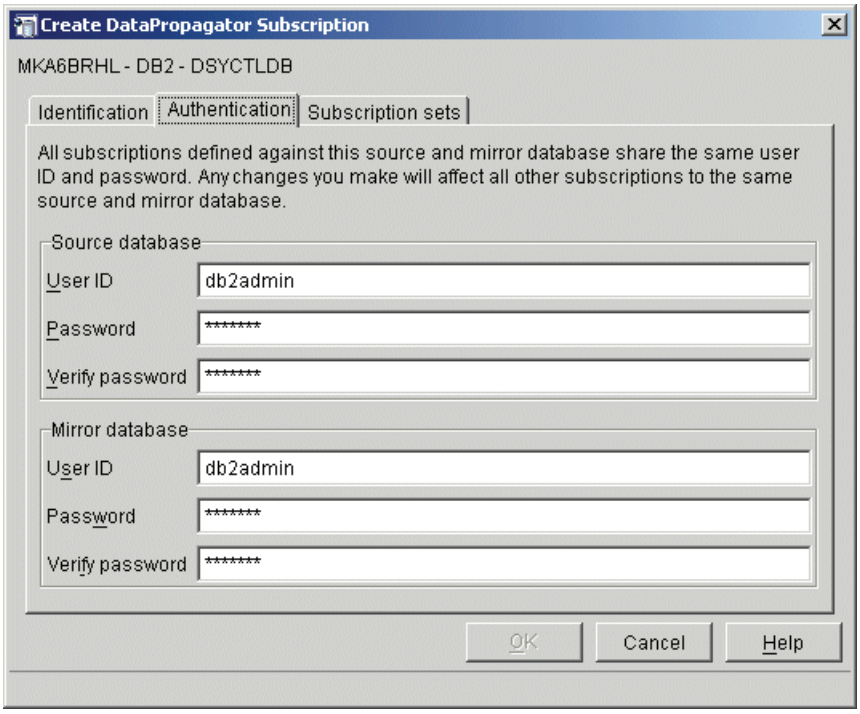


Figure 20-39 Enter authentication information for source and mirror databases

14. Click the **Subscription sets** tab. Select the subscription set(s) that this subscription will belong to and click **>** or **>>** to bring it to the Selected subscription sets pane. You can also click **Create** to create a new subscription set.

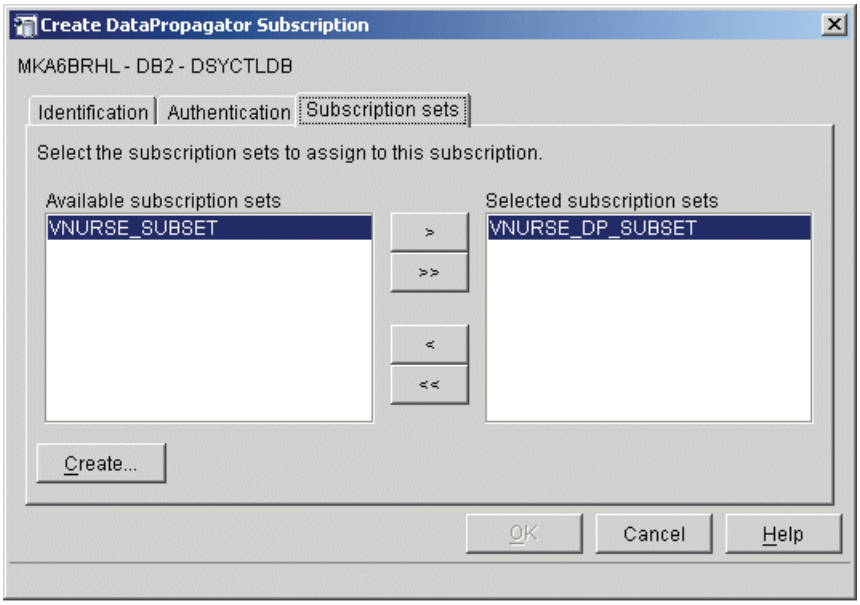


Figure 20-40 Select subscription set

15. Click the **Identification** tab again. Click the **Define subscription** button.

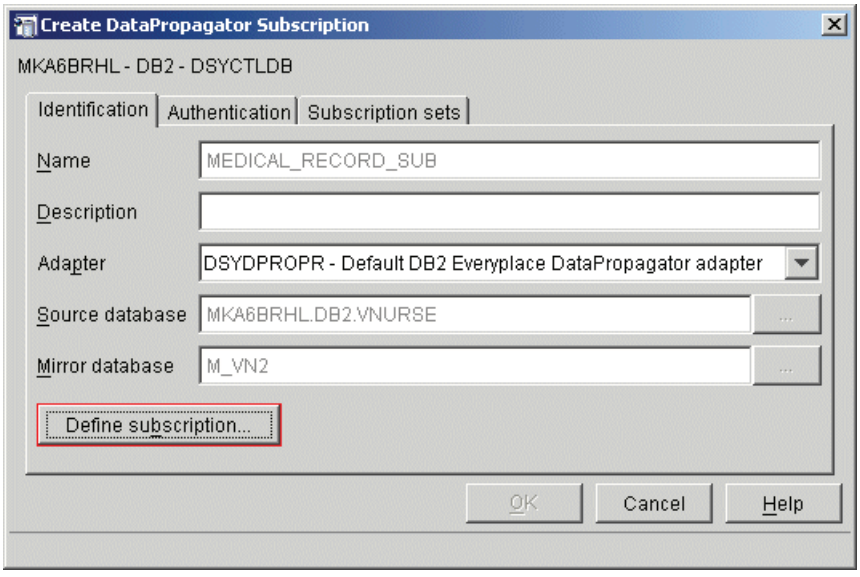


Figure 20-41 Define subscription

16. The Define Replication Subscription wizard opens.

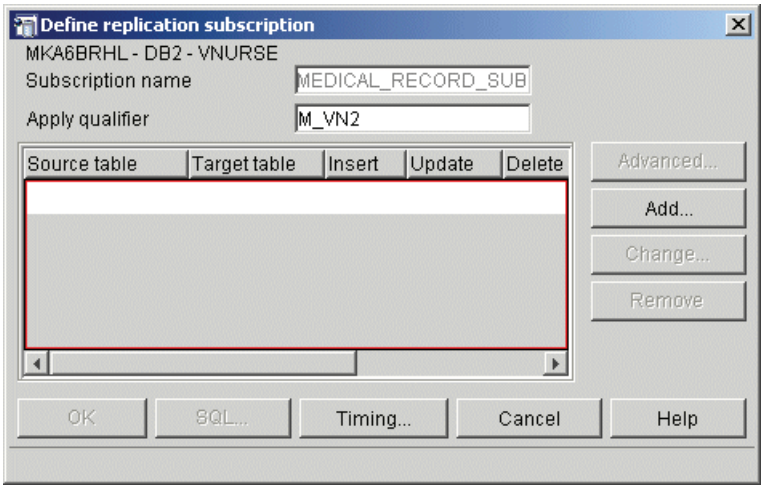


Figure 20-42 Define Replication Subscription wizard

17. Click the **Add** button to open the Add window.

18. The previously defined replication sources are displayed. Select the desired one and enter a name in the Target table field if it is different from the source. Click **Add**. Repeat for more replication sources. Click **Close** when done.

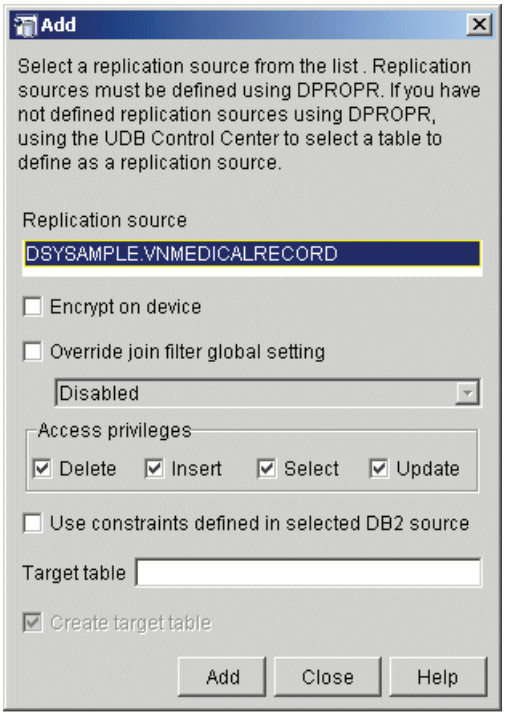


Figure 20-43 Add replication source

19. In the Define Replication Subscription wizard, click the **Timing** button to modify the replication frequency.

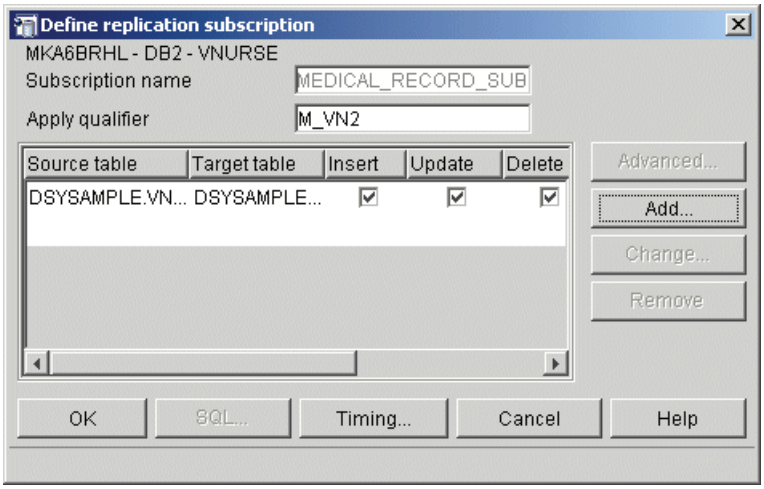


Figure 20-44 Define Replication Subscription wizard

20. Click **OK** and **OK**. Upon successful creation of the subscription, a DB2 message is shown.

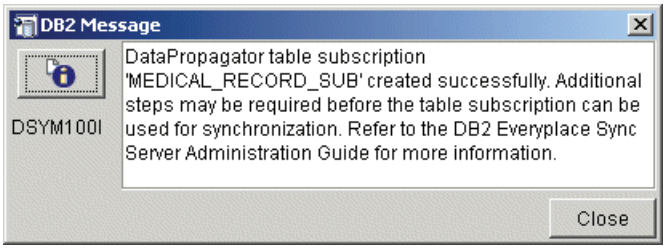


Figure 20-45 Subscription creation successful

Now there are some extra steps to take in order to bind the tables and start capture for the DataPropagator subscription:

1. Exit the DB2 Control Center and Mobile Devices Administration Center.
2. Open a DB2 command window by selecting **Start -> Programs -> IBM DB2 -> Command Window**.
3. Shut down all connected applications by entering:

   ```
   db2 force application all
   ```

   Note: If you are using the same DB2 database server to store the WebSphere Everyplace Access components configuration and to test the DataPropagator synchronization, be sure to stop all the components that use the database server before issuing the **`db2e force`** command. In a typical configuration this includes WebSphere Application Server, Everyplace Synchronization Server, and Directory Server.

4. In a DB2 command window, change to the bnd directory of the DB2 UDB, for example:

   ```
   cd c:\program files\SQLLIB\bnd
   ```

5. Then enter:

   ```
   bindcap db_name db2user db2userpassword
   ```

   This will change the database configuration parameter LOGRETAIN to CAPTURE. Replace `db_name` with the name of the database you would like to update, such as VNURSE in our example. Note that db2user must be a valid DB2 UDB user name and db2userpassword the corresponding password.

6. The same command must be run for the mirror database as well:

   ```
   bindcap mirror_db_name db2user db2userpassword
   ```

Replace `mirror_db_name` with the name of the mirror database you would like to update, such as M_VN2 in our example.

7. Start the capture process by typing in the DB2 command window:

   ```
   asnccp db_name
   ```

   Where `db_name` is the source database, for example VNURSE in our example. Leave the DB2 command window open. The capture process will run continuously in the background. If the capture process stops, repeat these steps.

8. The DataPropagator subscription has been fully configured now; check the Logs\Replication folder in MDAC to make sure replication succeeds.

Once the subscription is set up, IBM Everyplace Client can synchronize with the source database in exactly the same way with JDBC subscriptions.

For other resources of how to configure DataPropagator subscription, refer to *IBM DB2 Everyplace Sync Server Administration Guide Version 8 Release 1,* SC18-7186.

20.10.2 Upload subscription

The upload subscription process, as the name suggests, is one way only. It allows the client to insert rows into the source database, but not the other way around. There is no mirror database involved in the upload subscription process, which is illustrated in Figure 20-46 on page 685. In this sample scenario, a DB2 back-end database is used as the source database.

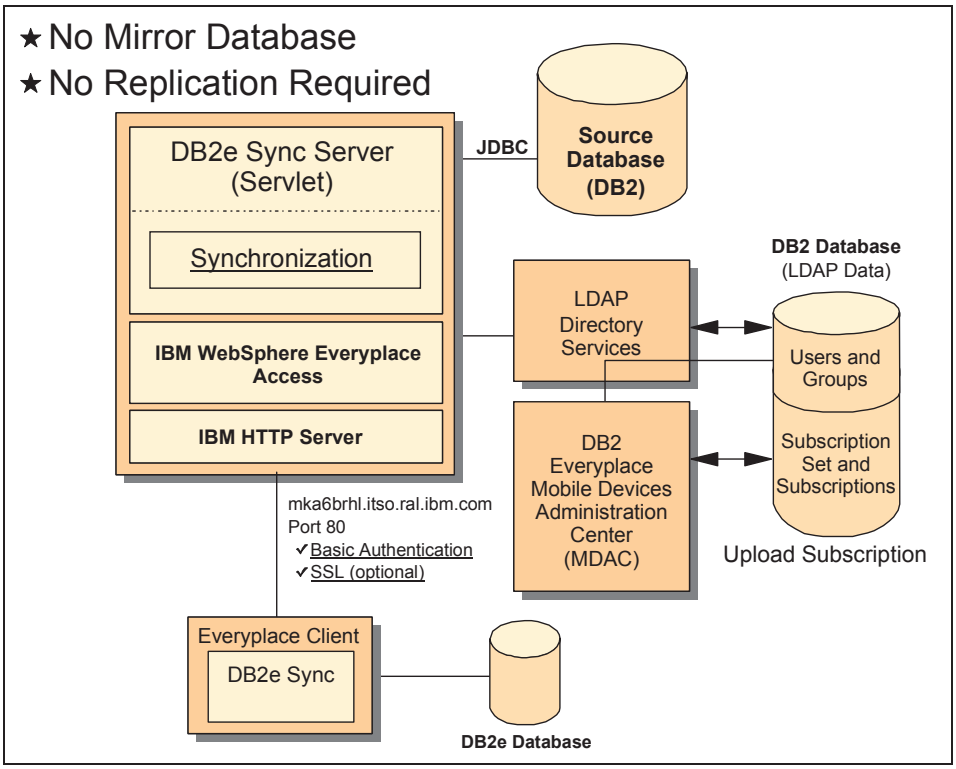


Figure 20-46 Upload subscription - sample scenario

To create an upload subscription:

1. Open the DB2 Control Center and Mobile Devices Administration Center by clicking **Start -> Programs -> IBM Everyplace Synchronization Server -> Launch MDAC**.
2. In the Mobile Devices Administration Center, select **Subscription -> Create -> Table Subscription -> Upload subscription**.

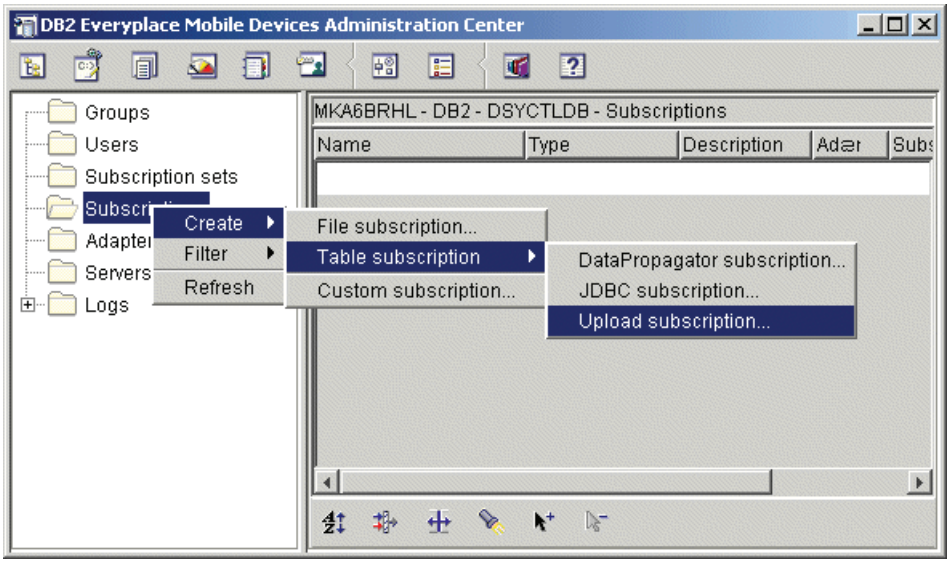


Figure 20-47 Create upload subscription

3. The Create Upload Subscription wizard opens. In the Identification tab, type a name for this subscription, for example `MedicalRecord`. Optionally, enter a description for this subscription.

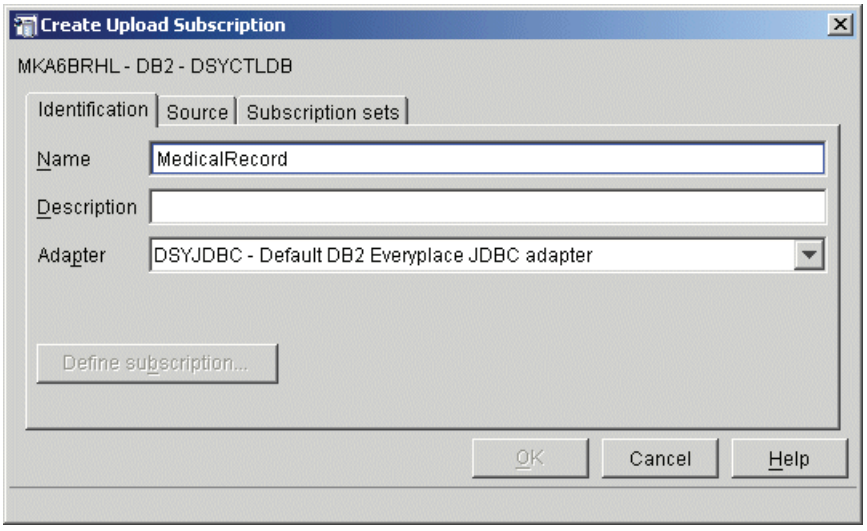


Figure 20-48 Name the subscription

4. Click the **Source** tab. Choose the suitable database driver by clicking the down arrow next to the Driver field.

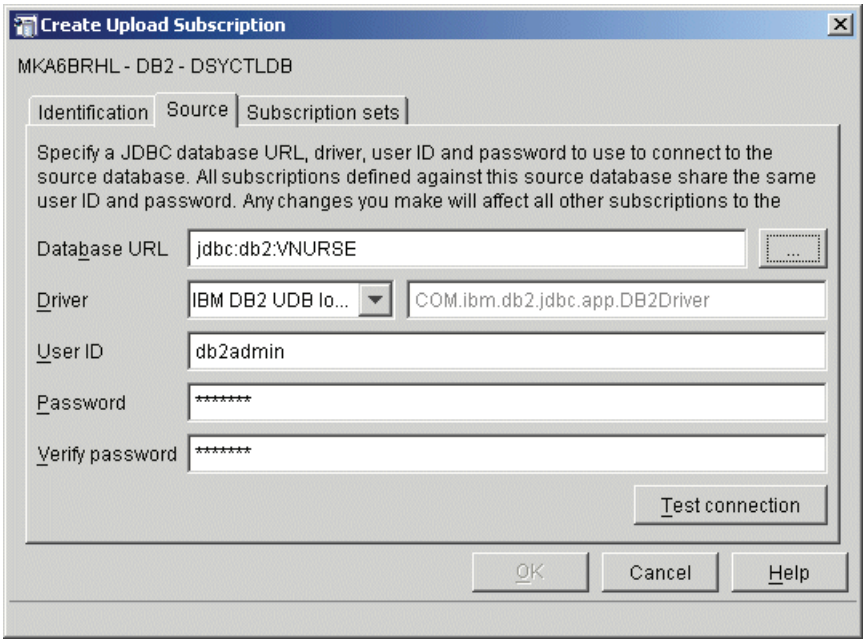


Figure 20-49 Setup source database

5. In the same window, enter the user ID and password that can access the source database.
6. Click the **Subscription sets** tab. Select the subscription set(s) this subscription will belong to and click **>** or **>>** to bring it to the Selected subscription sets pane. You can also click **Create** to create a new subscription set.

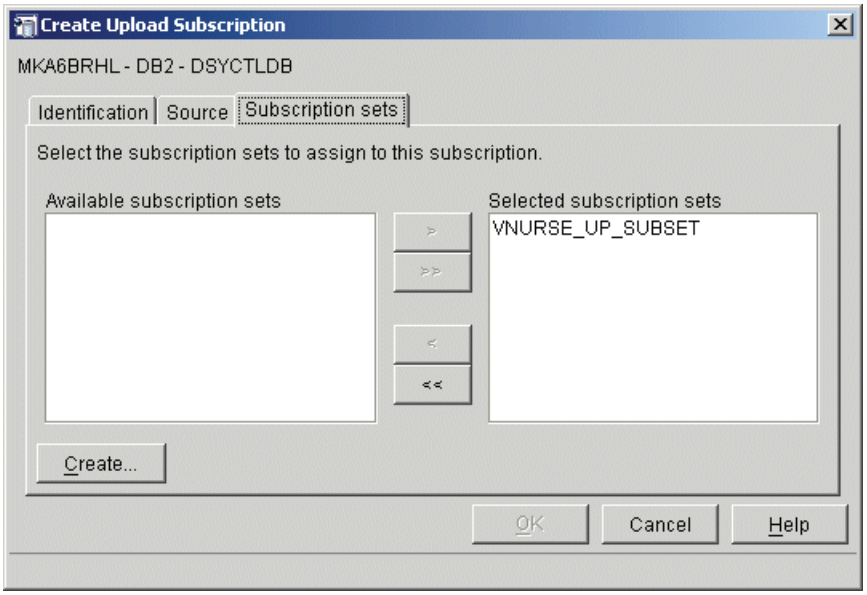


Figure 20-50 Select subscription set

7. Click the **Identification** tab again. Click the **Define subscription** button. The Define Upload Subscription wizard opens.

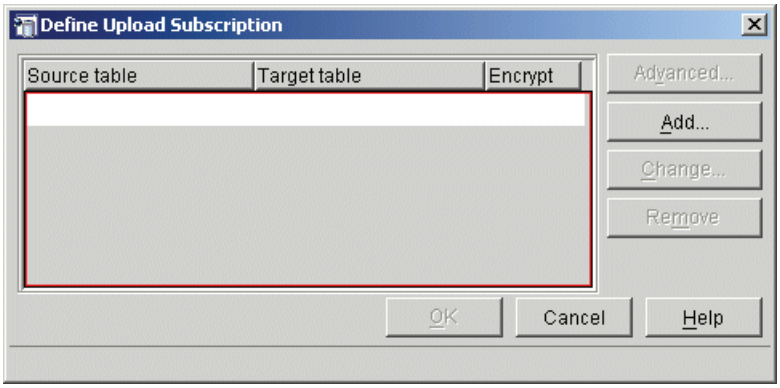


Figure 20-51 Define Upload Subscription wizard

8. Click the **Add** button to open the Add window.
9. Select the table into which rows will be inserted from the client. Change the Target schema and Target table name if they are different from the defaults. Click **Add**. Repeat for more tables. Click **Close** when done.

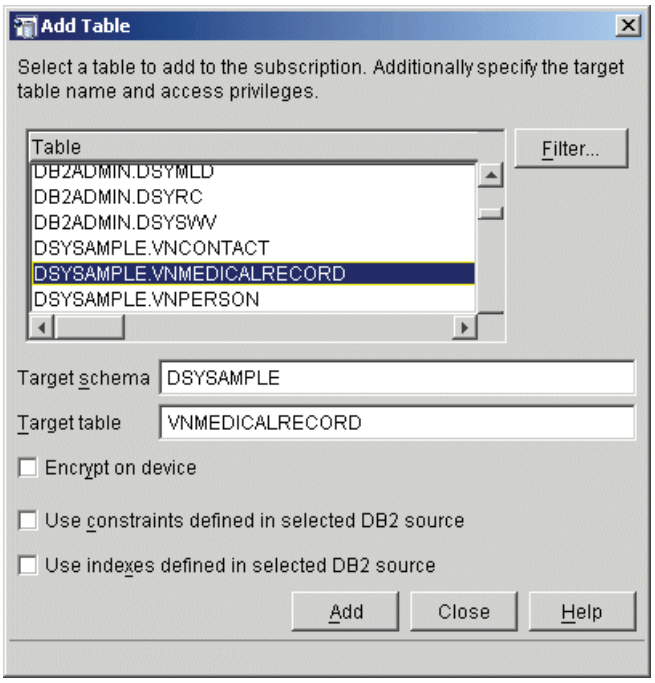


Figure 20-52 Select the table that is to receive data from client

10. Click **OK** and **OK**. The Upload Subscription will be created and shown in MDAC.

Once the subscription is set up, changes made to the DB2 Everyplace databases on the client will be uploaded into the source database through the use of IBM Everyplace Client. Changes made to the source database, on the other hand, will be propagated to the client database. IBM Everyplace Client usage is the same regardless of the different subscription type.

20.11 Filtering data from data sources

DB2 Everyplace Sync Server includes several filtering options for horizontal, vertical, and user-based filtering methods. Data filtering is essential to reduce traffic and optimize mobile device storage.

For details on how to configure these filters, refer to the *IBM DB2 Everyplace Sync Server Administration Guide Version 8 Release 1,* SC18-7186.

20.12 Debug and tracing

In this section, we include some useful information that you may need for problem determination when synchronizing DB2 Everyplace data.

20.12.1 Enable tracing

To turn on tracing for DB2 Everyplace Sync Server:

1. Open and edit the file:

   WebSphere_Install_Dir\IBMSyncServer\db2e\Server\properties\com\ibm\mobileservices\DSYGdflt.properties

2. Modify the key Trace.Level to:

   ```
   Trace.Level = *
   ```

3. Restart Everyplace Synchronization Server in the WebSphere Application Server Administrative Console.

20.12.2 Trace files

The trace files for DB2 Everyplace synchronization are located in:

WebSphere_Install_Dir\IBMSyncServer\db2e\Server\logs\dsy.n.trace

Where n=0,1, 2....n. Depending on the trace file size set in DSYGdflt.properties, when the first trace file (dsy.0.trace) hits the limit, a new trace file (dsy.1.trace) is generated, and so on.

There is another trace file in the \logs directory, dsyadmin.n.trace, where n=0, 1, 2...n; this is the trace file for MDAC.

20.12.3 DB2 Everyplace control database

All the information configured in MDAC is stored in a control database (DSYCTLDB) in DB2. If the control database gets corrupted, it can be dropped and recreated using the batch file:

WebSphere_install_dir\IBMSyncServer\db2e\Server\bin\dsyctldb.bat

20.13 Hints and tips

- It is always a good idea to test your connection when defining subscriptions.
- After defining a subscription, check in the MDAC under **Logs -> Replication** if there is any replication activity started and ended successfully for the newly defined mirror database. If there is not, the subscription is probably not configured properly and you will need to delete it and define it again.
- When the database is large, make sure to configure a big enough log file size for the mirror database, as all the records are processed in a single transaction. Otherwise, replication may fail.
- When the database is large and replication takes a much longer time, make sure to leave enough time between each replication. Otherwise, replication may fail.
- When synchronization fails with no apparent reasons, try resetting the user in MDAC by right-clicking the particular user and choose **Reset**.

21

Synchronizing with Oracle databases

This chapter describes how to configure the Oracle database as the synchronization data source for WebSphere Everyplace Access DB2 Everyplace.

© Copyright IBM Corp. 2003. All rights reserved.

21.1 Common grounds with DB2 data source

The relational database synchronization architecture was discussed in Chapter 20, “Synchronizing with DB2 databases” on page 645. This information remains valid in this chapter. The only difference is that the back-end database is Oracle now. Therefore, what is different here in operation is database replication. Now the data needs to be replicated into a DB2 mirror database from an Oracle data source. The process for JDBC subscription types is illustrated in Figure 21-1.

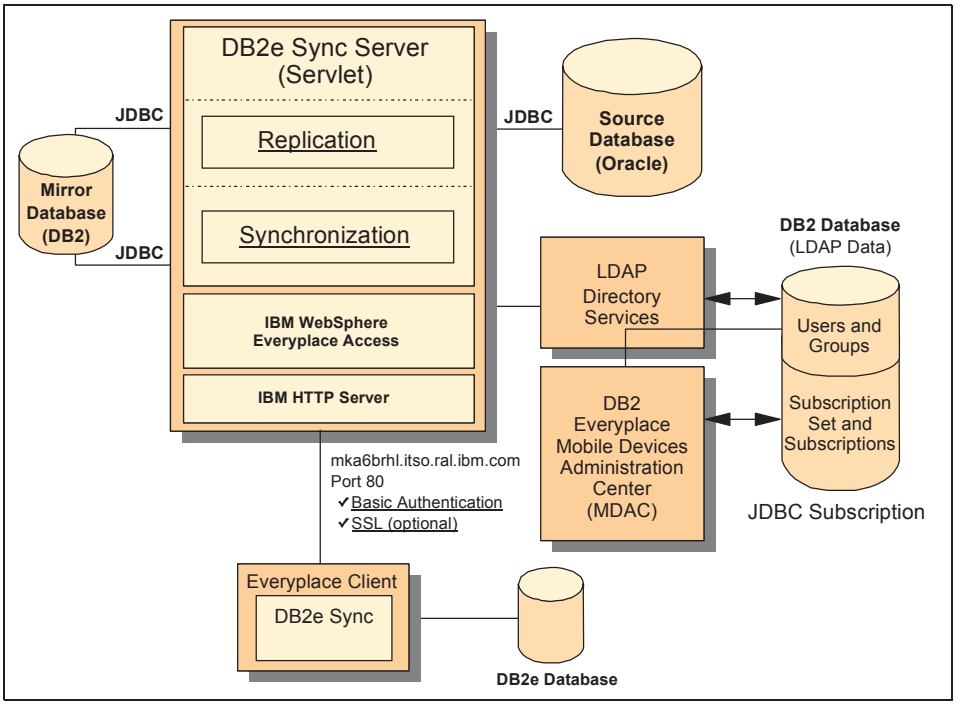


Figure 21-1 JDBC subscription synchronization with back-end Oracle database

For group and user creation and client configuration, refer to Chapter 20, “Synchronizing with DB2 databases” on page 645.

Note: This chapter only covers how to create subscriptions with an Oracle data source. Everything else remains the same as in the DB2 case.

21.2 Create a subscription with Oracle data source

In this section, we include a recommended procedure to install the JDBC driver to support an Oracle back-end database. Once this driver has been properly installed, you should be able to add JDBC and Upload synchronization subscriptions. However, the DataPropagator subscription type is not available when using a back-end Oracle database.

21.2.1 Add Oracle JDBC driver

In order to create a JDBC subscription with an Oracle database, DB2 Everyplace Server must be aware of how to load Oracle's JDBC driver. The following steps illustrate the changes to be made:

1. Exit the Mobile Devices Administration Center, if it is running.
2. Open WebSphere Application Server's Administrative Console by clicking **Start -> Programs -> IBM WebSphere -> Application Server V4.0 -> Administrator's Console**.
3. Stop the application server IBM DB2 Everyplace Server.
4. Locate the dsysetjavahome.bat file in the WebSphere_install_dir\IBMSyncServer\db2e\Server\bin directory and open it with a text editor.
5. Add the complete path of the Oracle JDBC driver to the SET_JDBC_DRV_CP line of the file and save the changes. For example:

   ```
   SET JDBC_DRV_CP=c:\oracle\ora92\jdbc\lib\classes12.zip
   ```

6. In the WebSphere Application Server Administrative Console, click **IBM DB2 Everyplace Server**. In the right pane, click the **JVM Settings** tab.

   Add the complete path of the Oracle JDBC driver into the list of classpaths.

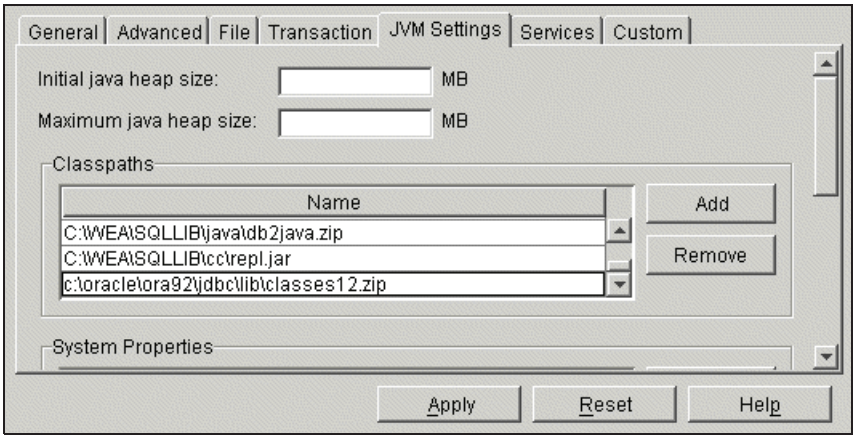


Figure 21-2 Add Oracle JDBC driver to DB2 Everyplace Server's classpath

7. Click **Apply**.
8. Restart the application server IBM DB2 Everyplace Server. Make sure there is no error in the Event Message pane.

21.2.2 Create a JDBC subscription

1. Open the Mobile Devices Administration Center by clicking **Start -> Programs -> IBM Everyplace Synchronization Server -> Launch MDAC**.
2. If there is no subscription set, create one following the steps described in 20.3.2, "Creating subscription and subscription set" on page 653.

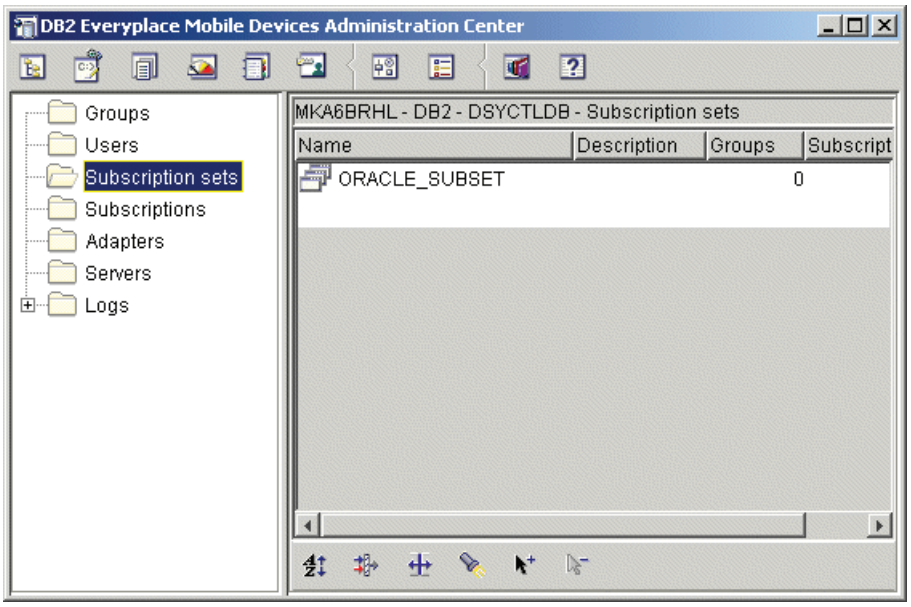


Figure 21-3 Create a subscription set

3. Right-click the **Subscriptions** folder and choose **Create -> Table subscription -> JDBC subscription**.
4. On the Identification tab, give the new subscription a name of your choice.

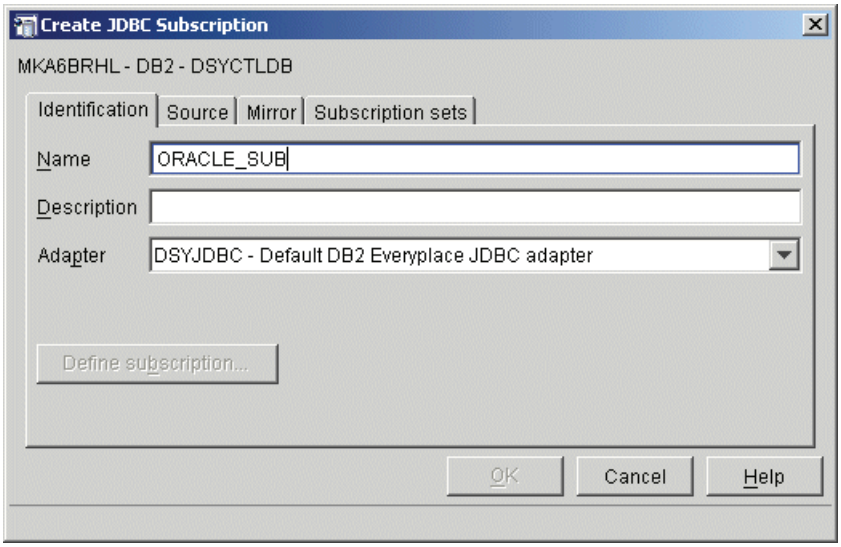


Figure 21-4 Name the subscription

5. Click the **Source** tab. Click the down arrow besides the Driver box and choose **Oracle** from the list. Observe that the format in the Database URL field changes.

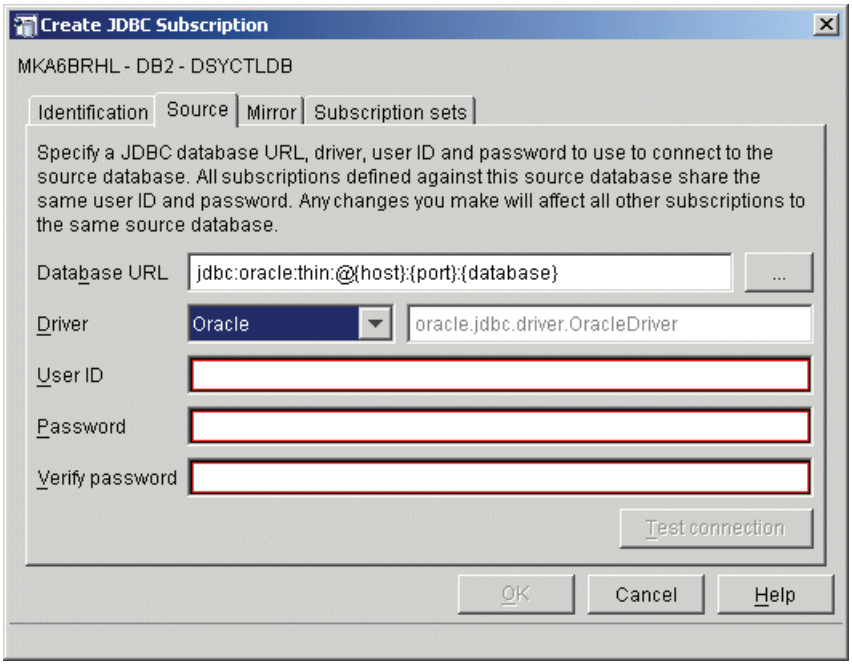


Figure 21-5 Choose Oracle as the driver

6. Enter the database URL for the Oracle data source, for example:

   ```
   jdbc:oracle:thin:@oracledbs.itso.ral.ibm.com:1521:sales
   ```

   Fill in the user ID and password that can access the database.

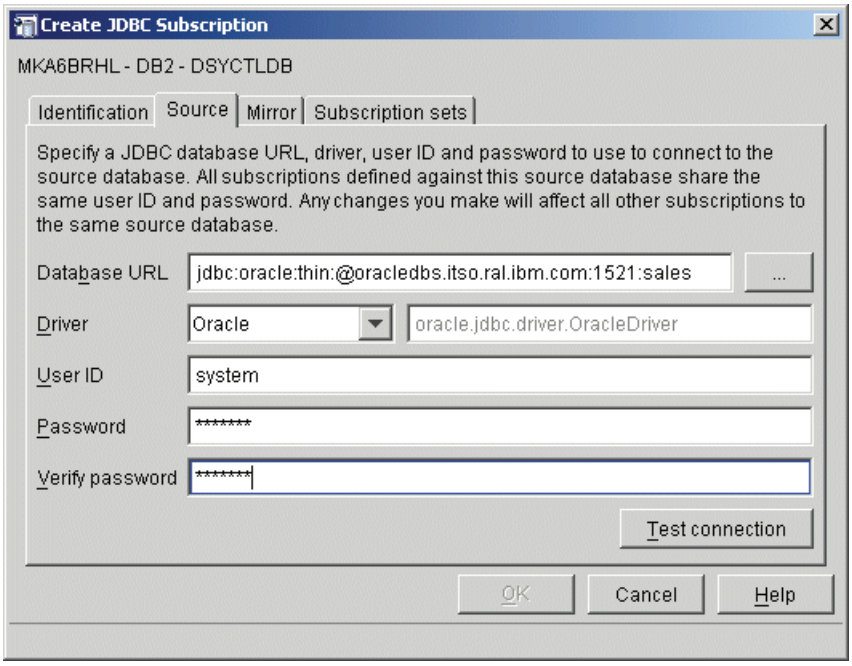


Figure 21-6 Fill in the source information

7. Click the **Test Connection** button to test whether a successful database connection can be established. If not, modify the information and try again.

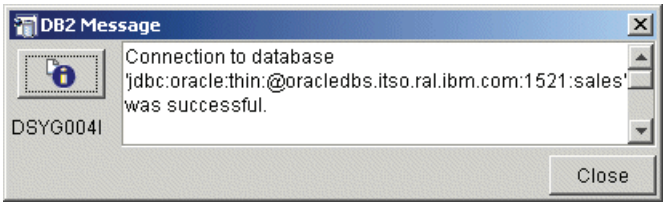


Figure 21-7 Test connection

8. Click the **Mirror** tab. Click the **...** button besides the Database URL box to choose a mirror database. Fill in the user ID and password information necessary to access the mirror database. Test the connection.

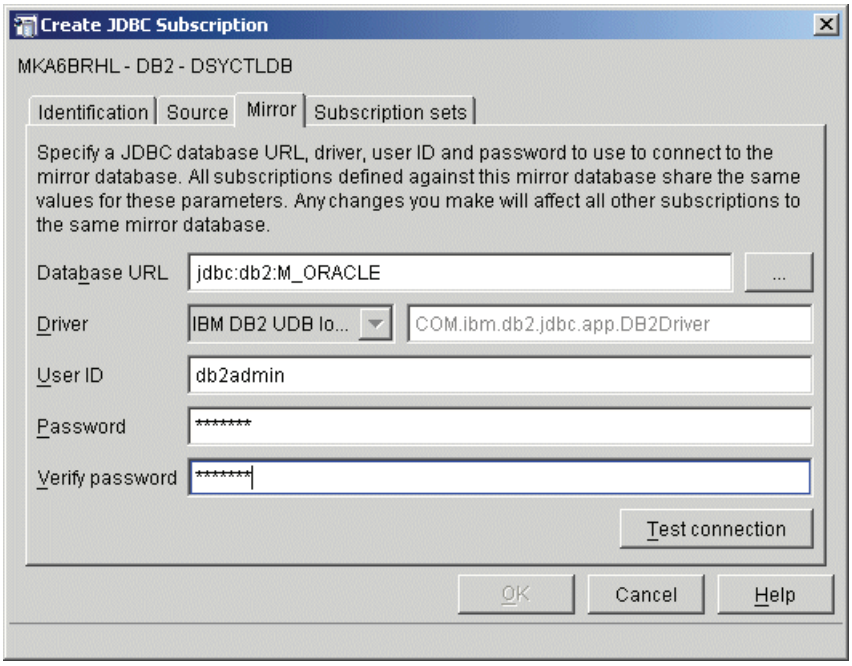


Figure 21-8 Configure mirror database

9. Click the **Subscription set** tab. Choose the subscription set this subscription will belong to and move it to the right pane.

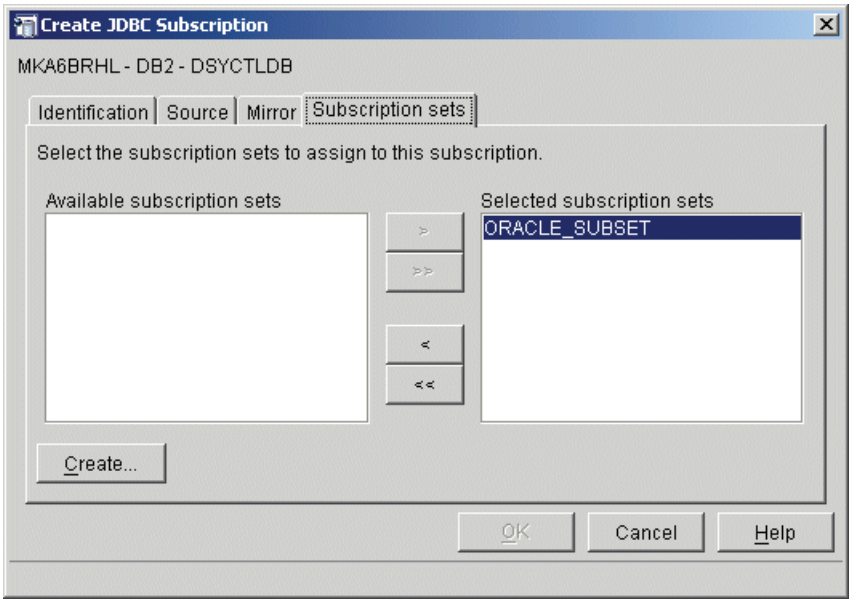


Figure 21-9 Choose subscription set

10. Go back to the Identification tab and click **Define subscription**.
11. Click **Add**. There is probably a message informing you that there are too many tables available to choose from. Click **Close**.

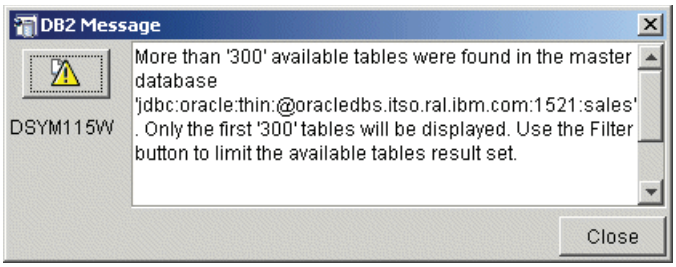


Figure 21-10 Too many tables to display

12. The available tables are shown. If you cannot find the table you want to use to do the configuration, click the **Filter** button.

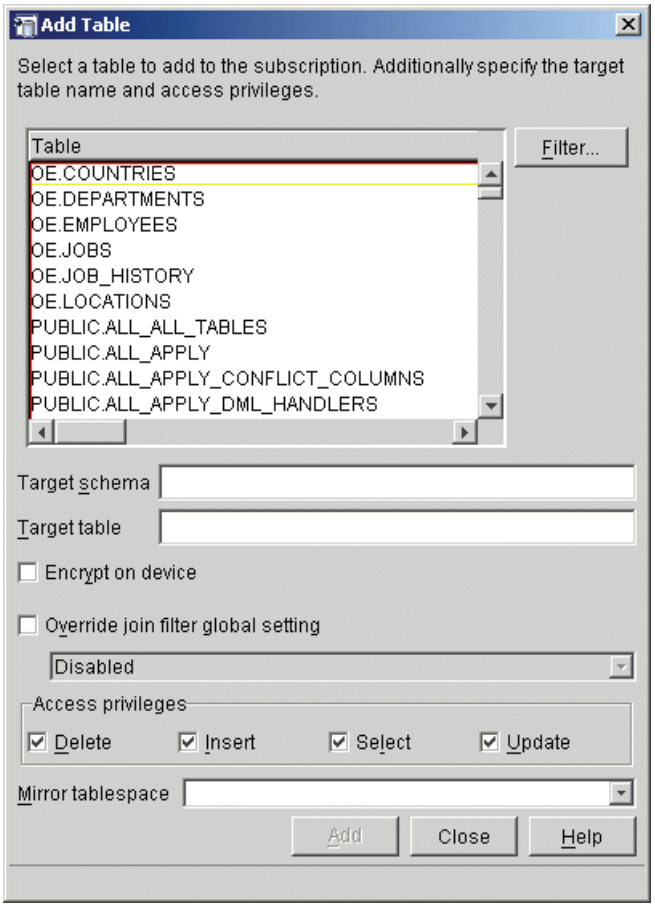


Figure 21-11 Available tables

13. Enter a suitable filter to reduce the scope. “%” is the wildcard character. Click **OK**.

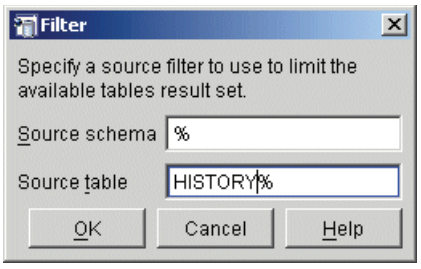


Figure 21-12 Filter the number of tables to view

14. Now the number of displayed tables is greatly reduced.

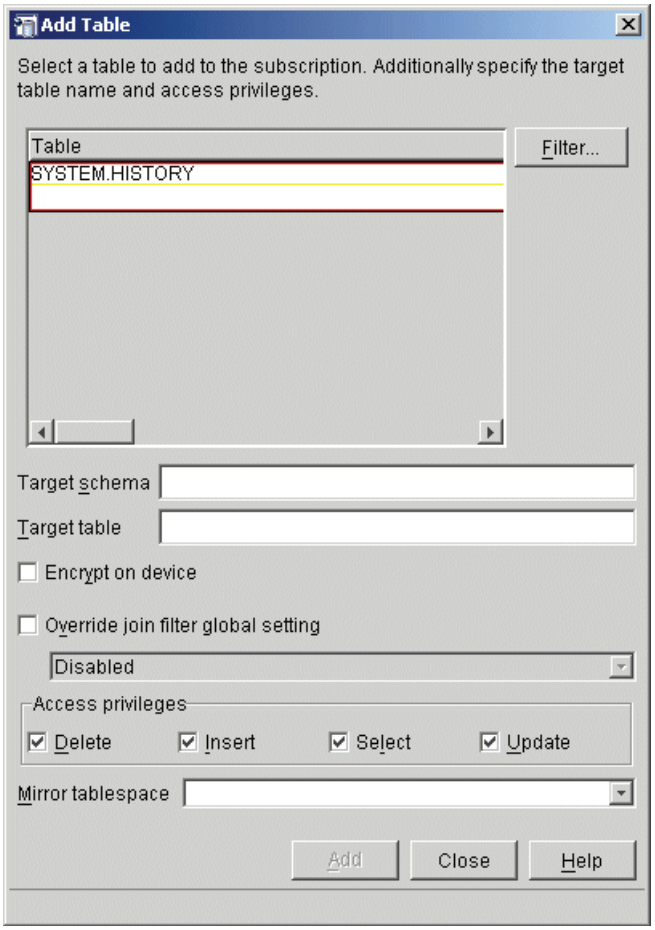


Figure 21-13 Filtered table view

15. Choose the table of interest and modify the Target schema and Target table name if necessary. Click **Add.**

16. Add more tables if desired. When done, click **Close**.

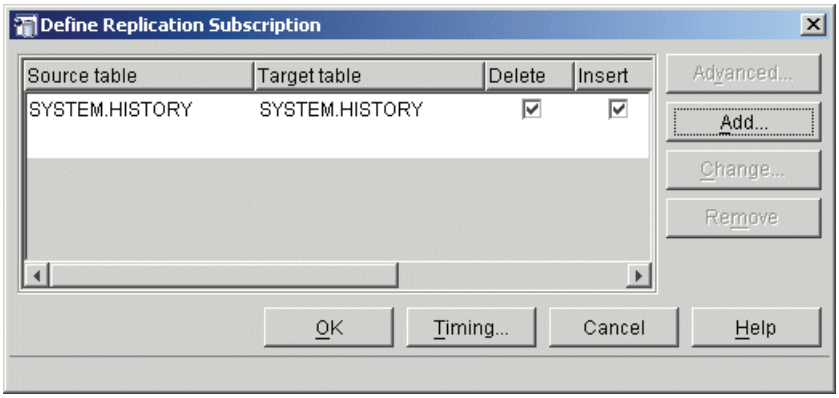


Figure 21-14 The table(s) to sync

17. Click **Timing** and modify the replication interval. For large tables, leave sufficient time.
18. Click **OK** and **OK** again.
19. If the source database is big, it takes a longer time to get the subscription created. When done, it returns to the MDAC window, with the created subscription shown in the right pane.

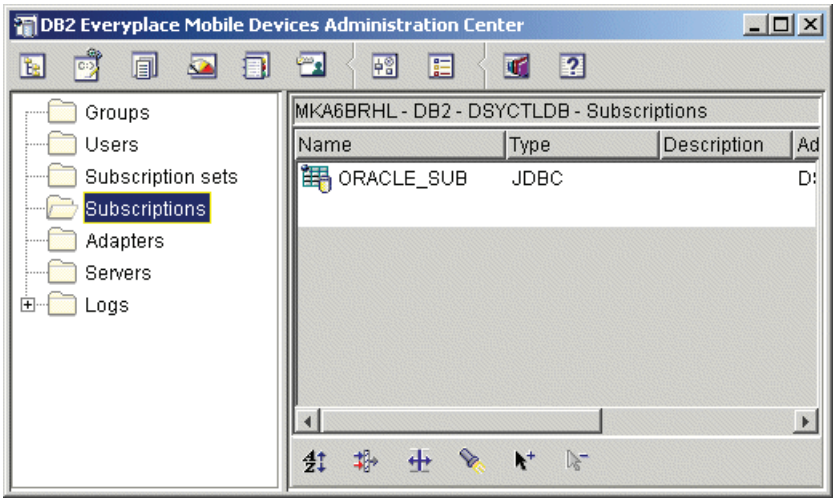


Figure 21-15 Subscription created

20. Right-click the **Replication** folder under Logs and choose **Refresh**. Make sure replication started and ended successfully for the mirror database you configured.

The subscription is now ready for use.

21.2.3 Create an upload subscription

As explained in 20.10.2, “Upload subscription” on page 684, upload subscription is for one-way data transfer between client and server databases. With Oracle source database, the client is able to upload changes in the DB2 Everyplace database into the Oracle source database. This scenario is illustrated in Figure 21-16.

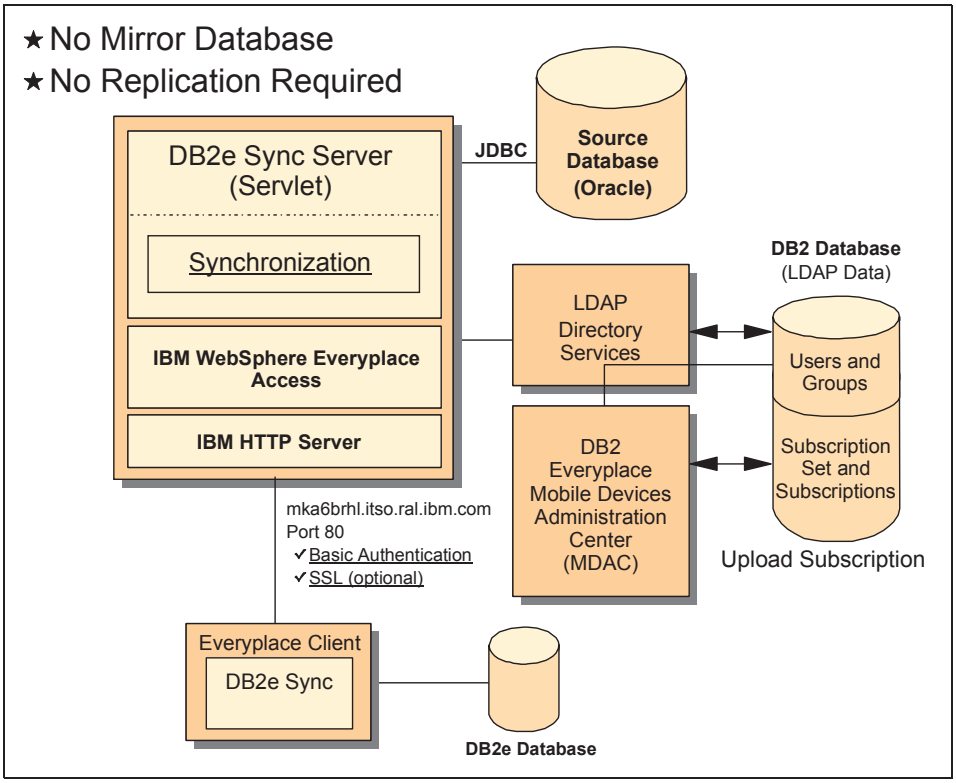


Figure 21-16 Upload subscription - sample scenario

You can create an upload subscription by following these steps:

1. Open the DB2 Control Center and Mobile Devices Administration Center by clicking **Start -> Programs -> IBM Everyplace Synchronization Server -> Launch MDAC**.
2. In Mobile Devices Administration Center, select **Subscription -> Create -> Table Subscription -> Upload subscription**.

3. The Create Upload Subscription wizard opens. In the Identification tab, type a name for this subscription, for example `SalesHistory`. Optionally, enter a description for this subscription.
4. Click the **Source** tab. Choose the **Oracle** database driver by clicking the down arrow next to the Driver field.

   Enter the correct database URL in the Database URL field, for example:

   ```
   jdbc:oracle:thin:@oracledbs.itso.ral.ibm.com:1521:sales
   ```

   Enter the user ID and password that can access the Oracle source database.

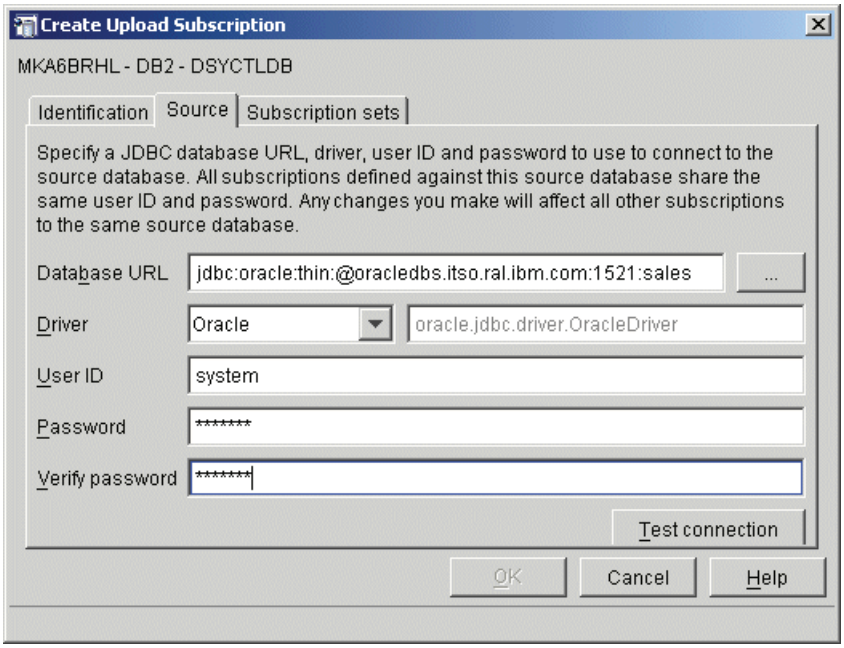


Figure 21-17 Define Oracle source database

5. Click the **Test connection** button to verify the connection can be established.
6. Click the **Subscription sets** tab. Select the subscription set(s) this subscription will belong to and click **>** or **>>** to bring it to the Selected subscription sets pane. You can also click **Create** to create a new subscription set.
7. Click the **Identification** tab again. Click the **Define subscription** button. The Define Upload Subscription wizard opens.
8. Click **Add**. There is probably a message informing you that there are too many tables available to choose from. Click **Close**.

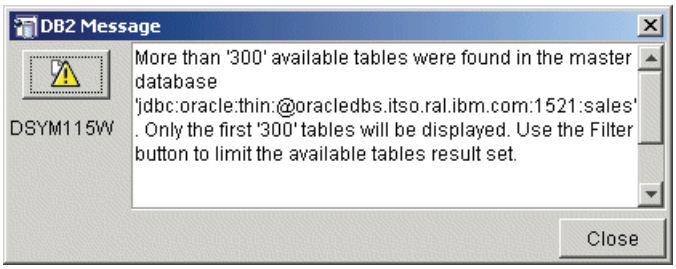


Figure 21-18 Too many tables to display

9. The available tables are shown. If you cannot find the table you want to configure with, click the **Filter** button.
10. Enter a suitable filter to reduce the scope. “%” is the wildcard character. Click **OK**.

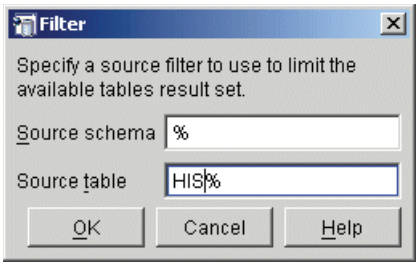


Figure 21-19 Filter the number of tables to view

11. Now the number of displayed tables is greatly reduced.

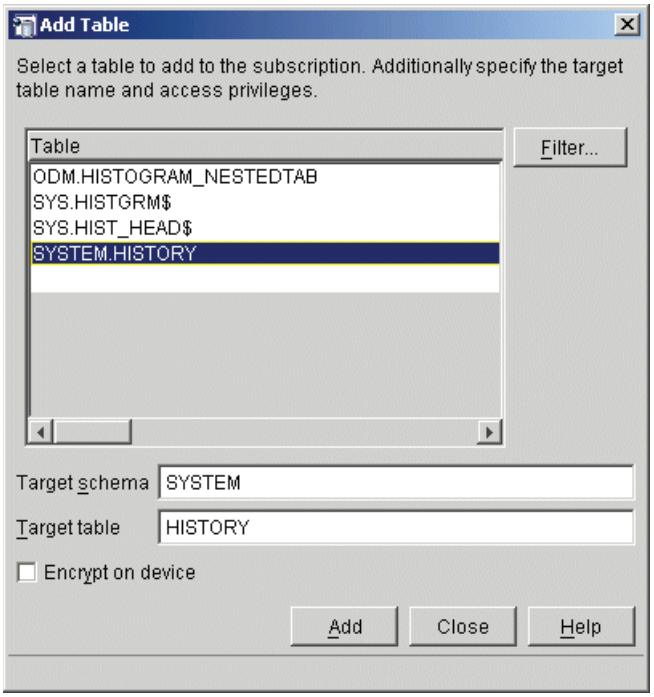


Figure 21-20 Filtered table view

12. Choose the table of interest and modify the Target schema and Target table name if necessary. Click **Add**.
13. Add more tables if desired. When done, click **Close**.
14. Click **OK** and **OK**. The upload subscription is created.

21.3 Sample dsysetjavahome.bat file

Example 21-1 Sample dsysetjavahome.bat file

```
@echo off
rem ------------------------------------------------------------------------
rem (C) COPYRIGHT International Business Machines Corp. 2000-2002
rem All Rights Reserved
rem
rem US Government Users Restricted Rights - Use, duplication or
rem disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
rem
rem dsysetjavahome.bat: set the java/jre path & classpath
rem ------------------------------------------------------------------------

for %%i in ("%DSYINSTDIR%") do (set DSY_INSTDIR=%%~sfi)
for %%i in ("%DSYSQLLIBINSTDIR%") do (set DSY_SQLLIBINSTDIR=%%~sfi)

SET JAVA_HOME=%WAS_HOME%\java
SET JAVA_HOME_MDAC=%DSY_SQLLIBINSTDIR%\java\java12\jdk\jre

rem for jdk 1.2.X and 1.3.X
SET JDK_EXTRA_CP=.

rem for jdk 1.1.X
rem SET
JDK_EXTRA_CP=%DSYSQLLIBINSTDIR%\java\jre\lib\i18n.jar;%DSYSQLLIBINSTDIR%\java\j
re\lib\rt.jar;%DSYSQLLIBINSTDIR%\java\swingall.jar

rem jdbc drivers
SET JDBC_DRV_CP=c:\oracle\ora92\jdbc\lib\classes12.zip
```

21.4 Synchronize with remote Oracle database

To create a subscription to synchronize with the Oracle data source that is on a separate machine, follow the same steps illustrated in 21.2, “Create a subscription with Oracle data source” on page 695.

21.5 Hints and tips

In addition to the hints and tips given in 20.13, “Hints and tips” on page 691, consider the following:

- If you get the following error:

  ```
  DSYD007, MDSS connection pool encountered the exception: unable to register
  JDBC driver for ......
  ```

  Check to make sure the Oracle JDBC driver is in the classpath of IBM DB2 Everyplace Server and that this server has been properly restarted with no errors.

- If you cannot make a test connection within MDAC to the Oracle database, check to make sure JDBC_DRV_CP is properly configured in dsysetjavahome.bat, and IBM DB2 Everyplace Server has been restarted to take in the change.

22

PIM and e-mail synchronization with Domino Server

This chapter discusses the use of the Everyplace Synchronization Server (ESS) to synchronize Personal Information Management (PIM) data with Lotus Domino Server. Topics covered in this chapter include:

- Overview
- How it works
- Configuration
- Usage
- Hints and tips
- Troubleshooting

© Copyright IBM Corp. 2003. All rights reserved.

22.1 Overview

There are often times when a mobile user requires information from an e-mail in his in-box. Likewise, there are also times when he needs access to his calendar to schedule an appointment or to find out what appointment he has next. How can he access this information when he is disconnected? With the use of a handheld device and the Everyplace Synchronization functionality, he can access and update his Personal Information Management (PIM) data while offline. This functionality is provided by the Everyplace Synchronization Server (ESS) component of Everyplace Access.

In this chapter, we discuss the procedure required to configure the ESS component for synchronization of PIM and e-mail data with the Lotus Domino Server.

22.2 How it works

For each user wishing to use the Everyplace Synchronization functionality, a WebSphere Everyplace Access user ID and a Lotus Domino Server user ID will be required. The WebSphere Everyplace Access user ID will be used to identify the user to WebSphere Everyplace Access and the Lotus Domino Server user ID will be used to identify the Lotus Domino Server account from which PIM and e-mail data will be retrieved.

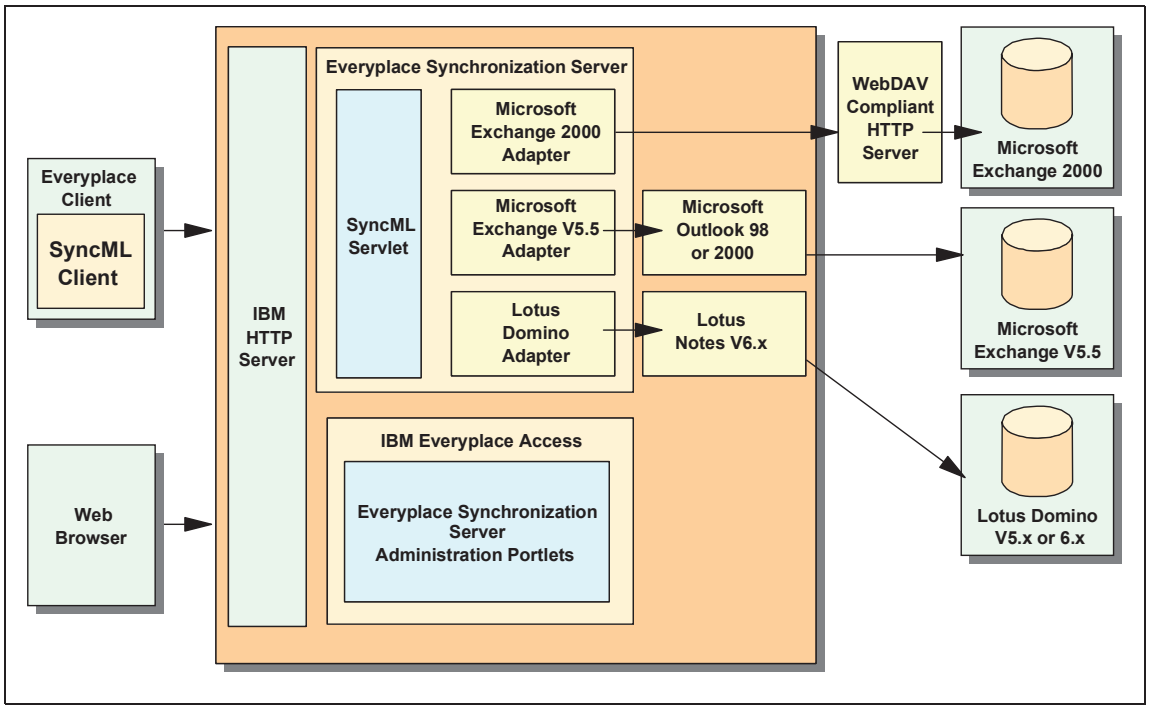


Figure 22-1 Everyplace Synchronization Server

As illustrated in Figure 22-1, during synchronization WebSphere Everyplace Access connects to the Lotus Domino Server through an adapter called the *Lotus Domino Adapter*. The adapter serves as an interface between WebSphere Everyplace Access and the Domino Server. It connects to Domino using an administrative user ID (Adapter Admin ID) and password with access to all Lotus Domino databases that are to be synchronized through the Everyplace Synchronization Server.

Once WebSphere Everyplace Access is connected to the Domino Server, WebSphere Everyplace Access uses the Lotus Domino user ID of the individual to retrieve PIM and e-mail data. However, before WebSphere Everyplace Access can do this:

- The Adapter Admin ID must be configured to have read access to the Public Name and Address Book (names.nsf) on the Lotus Domino Server. The Address Book must include the short name, user name and Internet password for every user who uses ESS to synchronize Lotus Domino documents.

- The Lotus Domino mail database for each user must grant *manager access* and *ability to delete* to the admin user ID. In addition to this, the maximum Internet name and password must be set to Editor or Manager.

After synchronization, the user can access and update his PIM and e-mail data. Any updates made will be stored on the client device until the next synchronization.

22.3 Configuration

Configuring the Everyplace Synchronization Server for synchronization with a Lotus Domino Server involves the following:

- Configuring the Lotus Domino Server where PIM and e-mail data will be retrieved from.
- Configuring the Lotus Notes client installed on the same machine as the Everyplace Synchronization Server. The Notes Client is used by the Lotus Domino Adapter to communicate with the Lotus Domino Server.
- Configuring the Lotus Notes user's database to grant access to the Adapter Admin ID.
- Configuring the Lotus Domino Adapter on the WebSphere Everyplace Access server. This may include setting up a default device profile.
- Configuring the WebSphere Everyplace Access user for synchronization with their Domino account. This may include setting up a user device profile.
- Configuring the WebSphere Everyplace Access Client on the client device that will be requesting the synchronization.

22.3.1 Lotus Domino Server configuration

The Lotus Domino Server where PIM and e-mail data will be retrieved from must be first be installed. Refer to Appendix H, "Lotus Domino Server and Sametime installation" on page 1257 for instructions on installing the Lotus Domino Server. It is recommended that you use Lotus Domino Version 5.0.11 or higher. Version 5.0.10 is known to be problematic with the Lotus Notes PIM portlets. Once the installation is complete, the following configuration steps can be performed.

Verify HTTP and DIIOP tasks are always started (optional)

This step is required only if you plan to use the Lotus Domino DIIOP PIM portlets. These portlets require HTTP and DIIOP to retrieve PIM and e-mail data. We need to verify that both the HTTP and DIIOP tasks are started when the Domino Server is started.

1. Open the file <Domino_root>\notes.ini, where <Domino_root> is the directory where Domino is installed on the server.
2. Search for the parameter `ServerTasks`.
3. Verify that the parameter values includes `HTTP` and `DIIOP`. If the values are not there, add them. For example:

   ```
   ServerTasks=<other tasks>,HTTP,DIIOP
   ```

4. Save your changes.
5. When you restart the Domino Server, you should see messages in the console indicating that these tasks have started.

Configure names.nsf database on Domino Server

We need to configure the Domain's Name and Address Book (names.nsf) to ensure the Adapter Admin ID has appropriate access to it.

1. Using the Lotus Domino Administrator client, select **File -> Database -> Open**.
2. Enter the name of the Domino Server and then click **Open**.

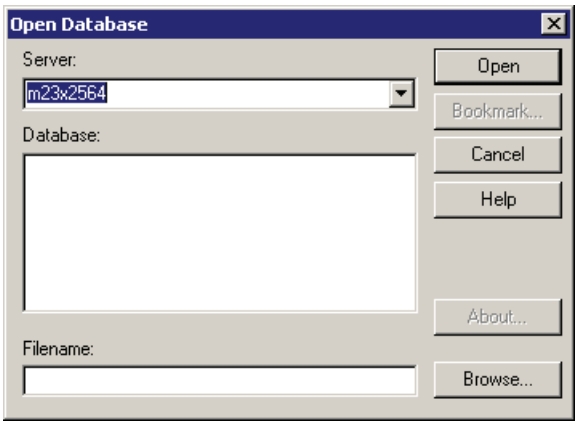


Figure 22-2 Enter Domino Server name

3. Select the Domain's Address Book (names.nsf) and click **Open**.

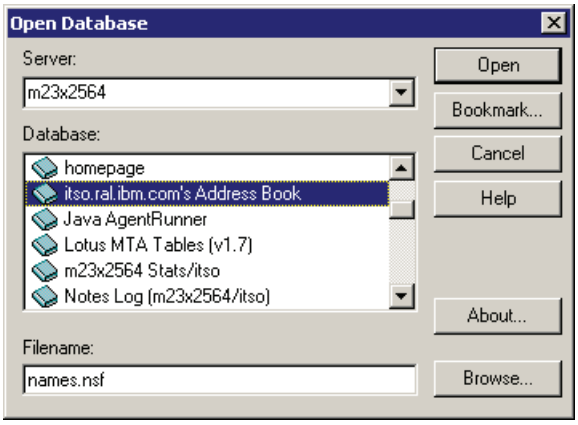


Figure 22-3 Select Domain's Address Book

4. On the left pane, select **Server -> Servers** and then double-click the Domino Server on the right pane.

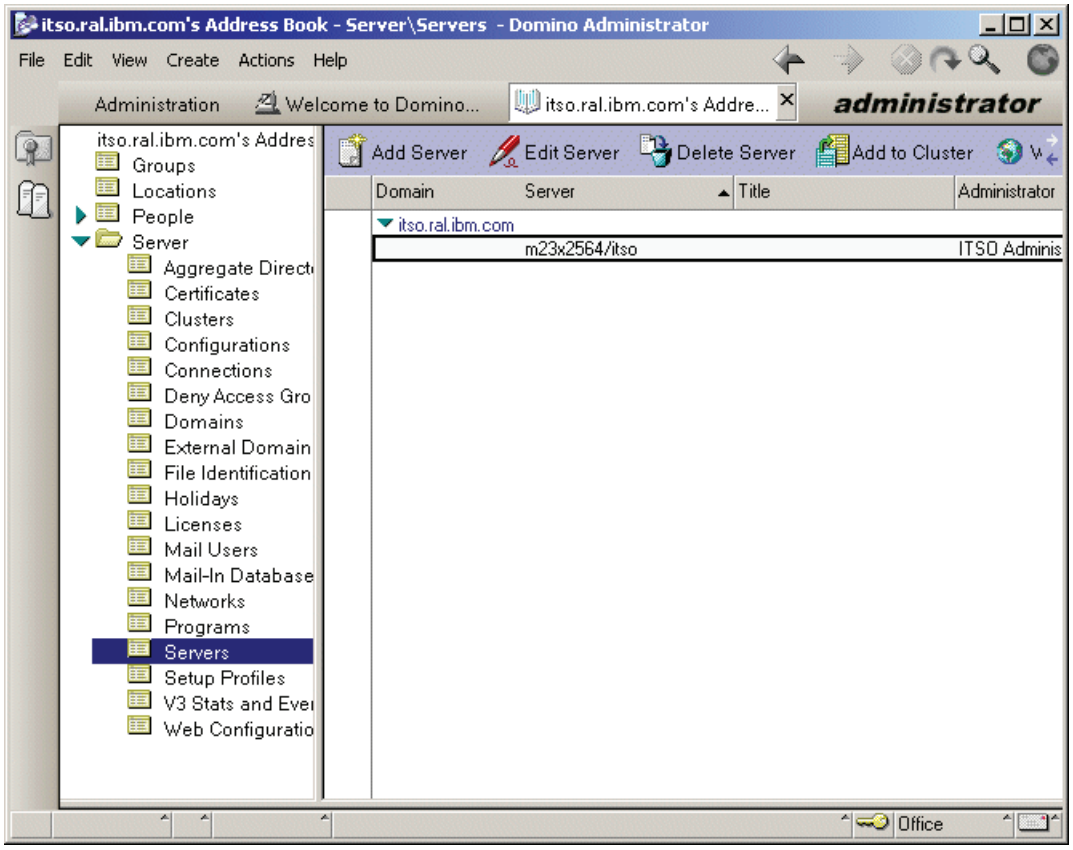


Figure 22-4 Double-click the Domino Server

5. Click the **Basics** tab. Add the Administrator group to the Administrators field. By default, the Administrator group is called Administrators unless you changed it during the installation of Domino. In our example, the Administrator group is ESS Admins because we changed it during the installation. Refer to Appendix H, “Lotus Domino Server and Sametime installation” on page 1257 for instructions on installing Domino Server.

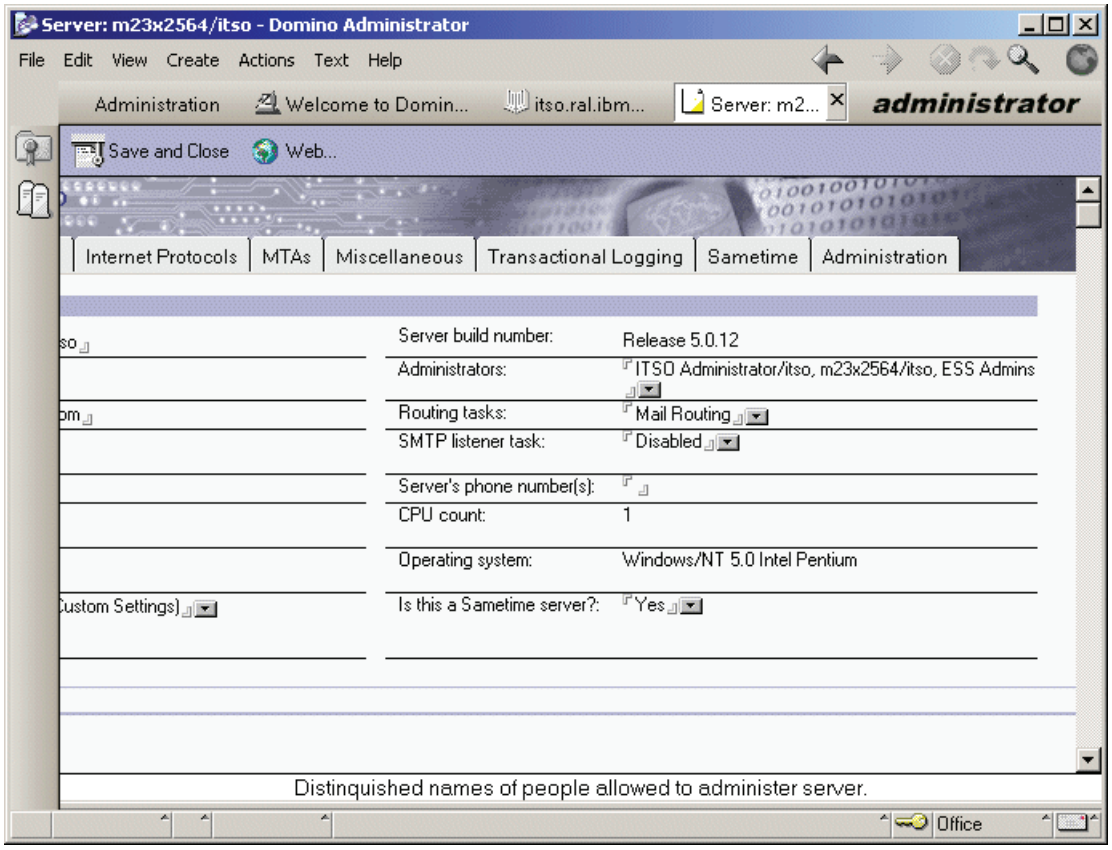


Figure 22-5 Basics tab

6. Verify that the Fully qualified Internet host name field contains the *fully qualified* host name of your Domino Server.

Important: If the host name is not fully qualified, the Everyplace Synchronization Server will not be able to connect to the Domino Server.

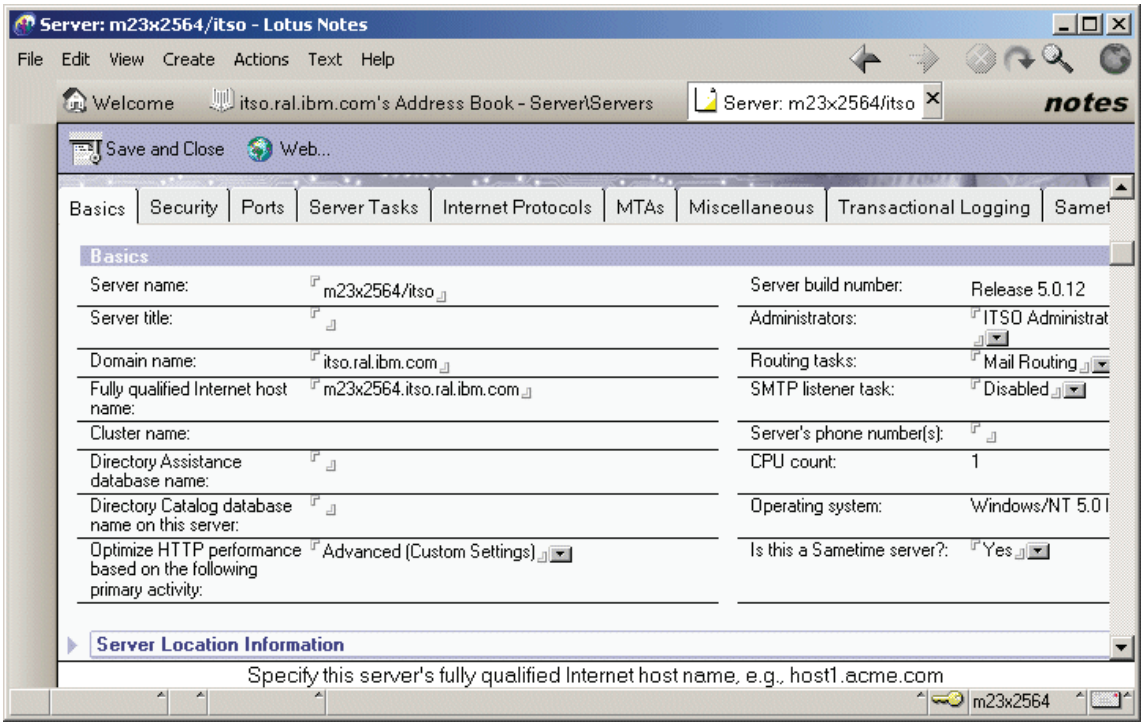


Figure 22-6 Basics tab - Fully qualified Internet host name

7. Click the **Security** tab. Add the Administrator group to the Create new databases field and the Create replica databases field.
8. Add * to all fields under Agent Restrictions.
9. Optional - Add * to all fields under Java/COM Restrictions. This step is only required if you plan to use the Lotus Domino DIIOP PIM portlets.

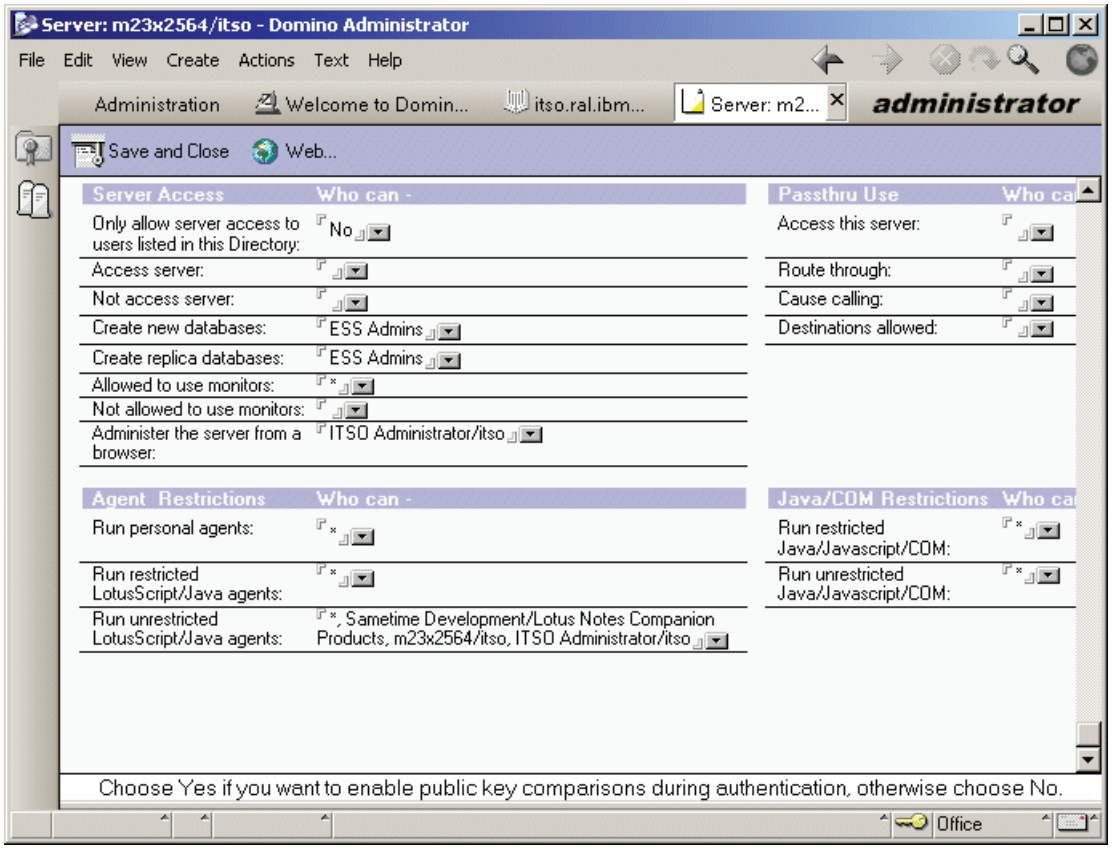


Figure 22-7 Security tab

10. Click the **Administration** tab. Add * to the Owner and Administrators field.

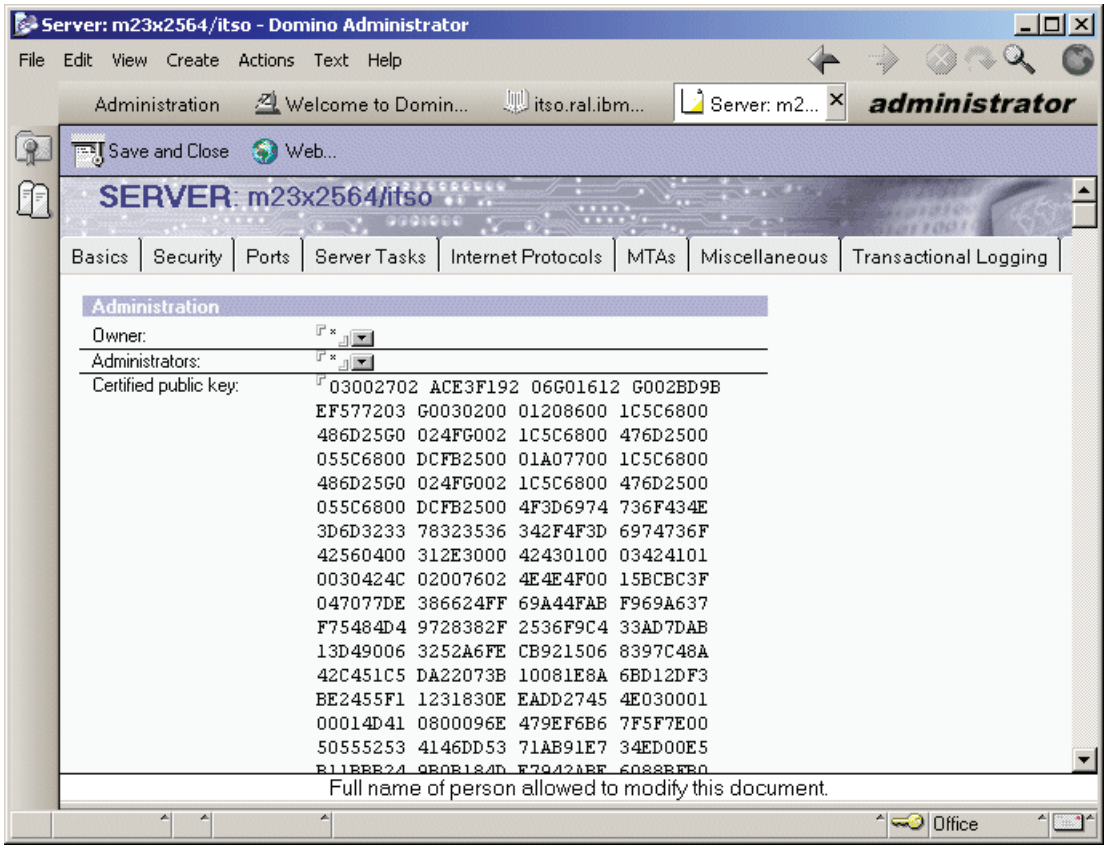


Figure 22-8 Administration tab

11. Click **Save and Close**.
12. Restart the Domino Server.

Create Adapter Admin User

We need to create a user that will be used by the Lotus Domino Adapter to connect to the Lotus Domino Server. This user will need to belong to the Administrators group.

1. Using the Lotus Domino Administrator client, Select **Administration -> People & Groups**.
2. Select **People -> Register** on the right side.

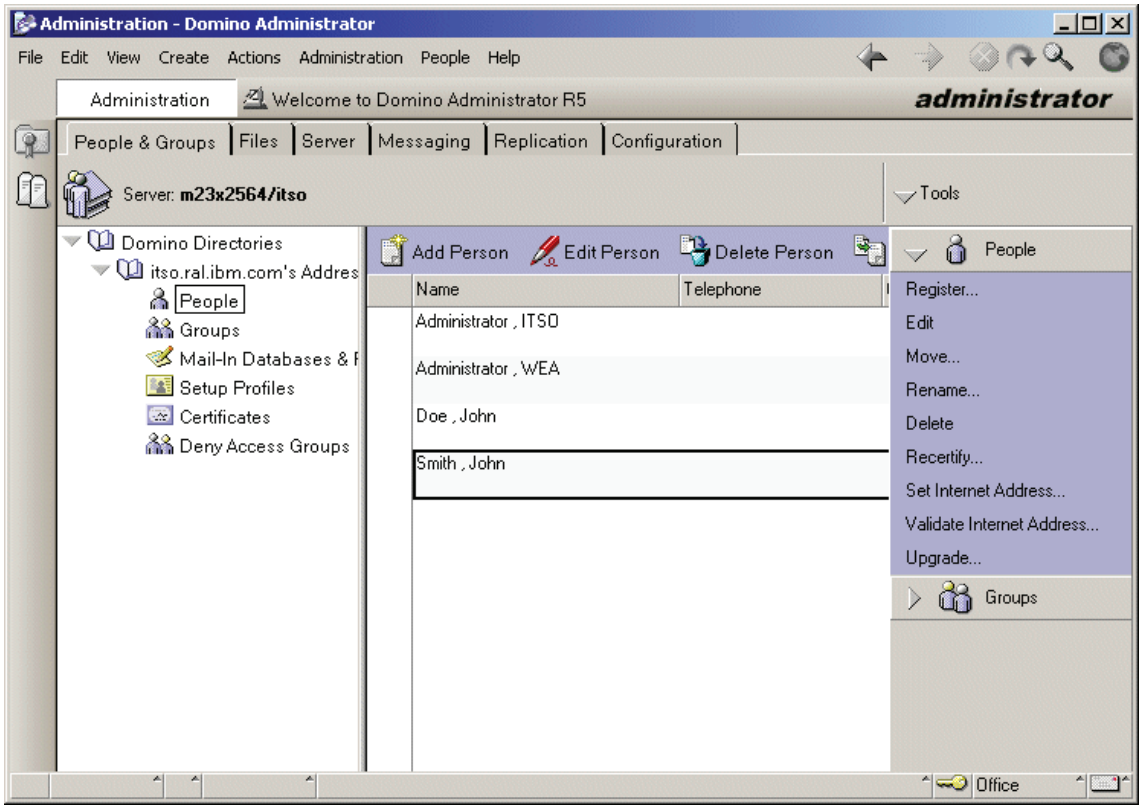


Figure 22-9 Registering a new user

3. It will ask you to enter the certifier password. The certifier password was set during the installation of Lotus Domino. Ensure a copy of the cert.id file from the Lotus Domino Server exists in the Lotus Domino Administrator client directory. Enter the password and click **OK**.
4. Select **Yes** to bypass the warning.
5. Select the **Advanced** check box. Enter the first name, the last name and the password. Select the **Set Internet password** check box. Enter the Internet domain. Change the user name on the Internet address if you do not like the generated one.

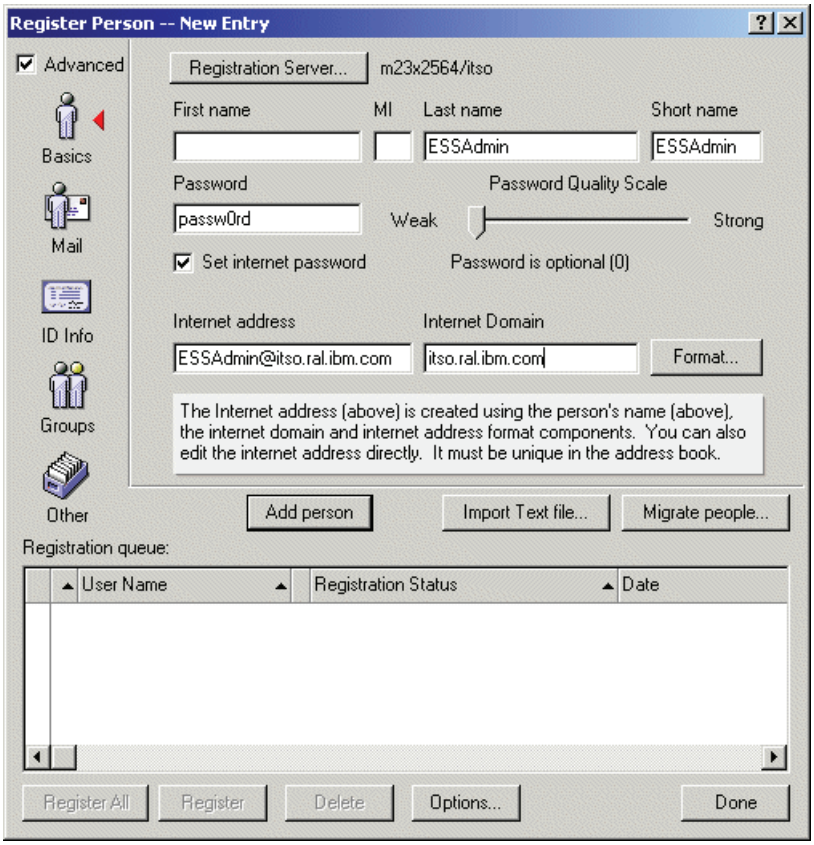


Figure 22-10 Register Person - Basics

6. Click **Mail** on the left pane. Click **Mail Server** and enter the Domino Server name. Select **iNotes Web Access** as the Mail file template.

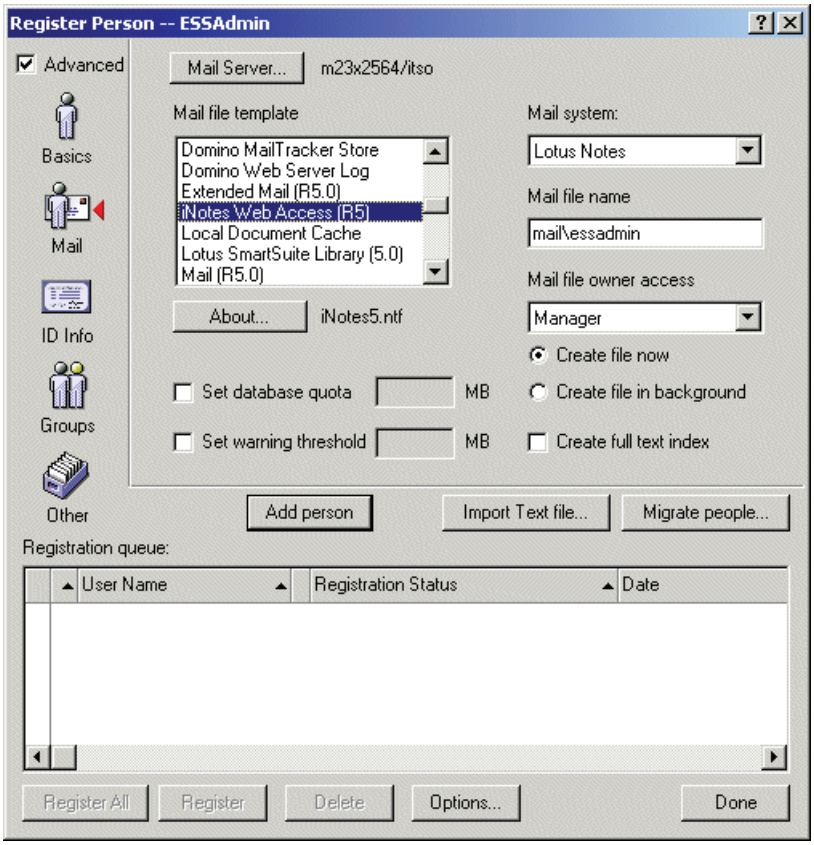


Figure 22-11 Register Person - Mail

7. Click **ID Info** on the left pane. Enter the appropriate security type and certificate expiration date. Select the **In file** check box under Location for storing user ID.

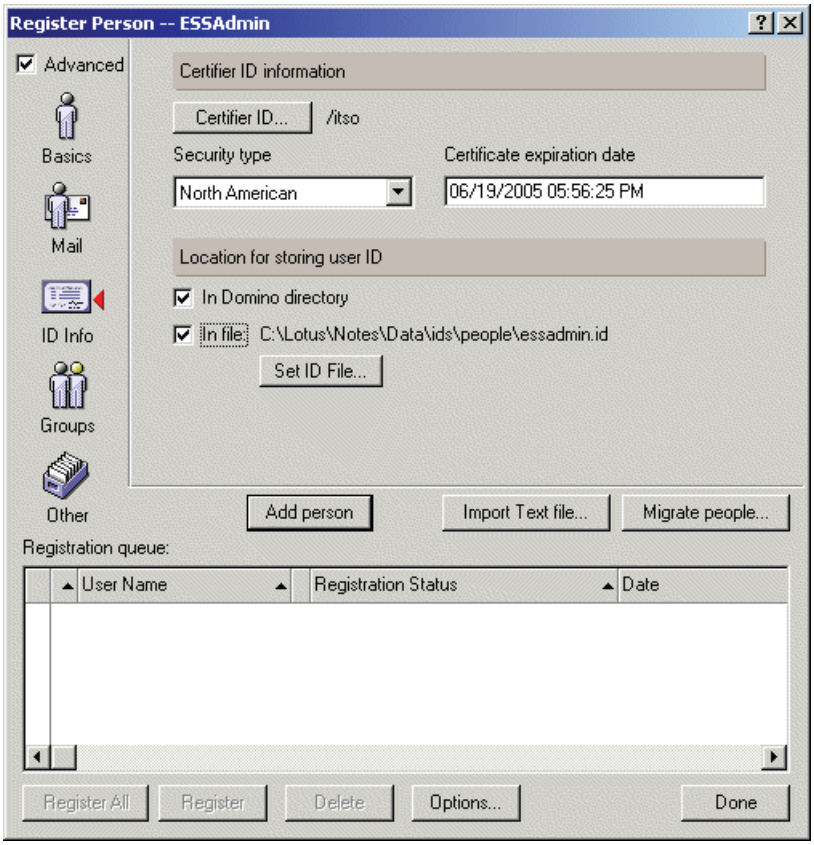


Figure 22-12 Register Person - ID Info

8. Click **Groups** in the left pane. Add the Administrator group to the Selected groups. In our example, the Administrator group is called ESS Admins.

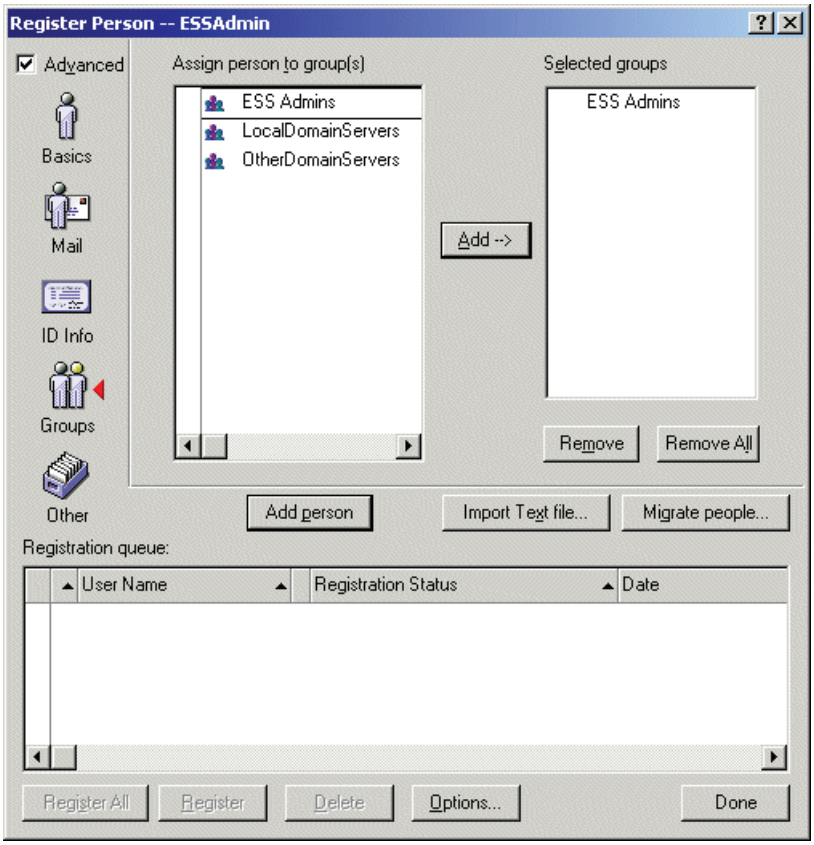


Figure 22-13 Register Person - Groups

9. Click **Add person**. Select user in the Registration queue and then click **Register**. Click **Done**.

Add Adapter Admin ID to ACL of mail.box database

The Adapter Admin ID needs to be added to the Access Control List (ACL) of the mail.box database.

1. Using the Lotus Domino Administrator client, select **File -> Database -> Open**.
2. In the Server field, enter the name of the Domino Server. In the Filename field, enter `mail.box` and then click **Open**.

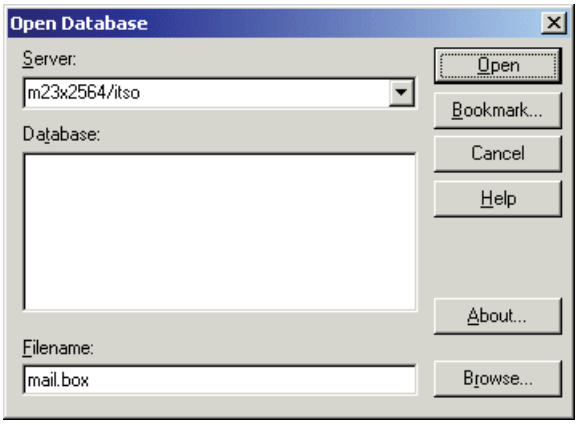


Figure 22-14 Open the mail.box database

3. Select **File -> Database -> Access Control**.
4. Click **Add** and then click the button on the right of the Person, server, or group field. Change the Address Book to the Domain Address Book by clicking the drop-down box at the top. Select the Adapter Admin ID and then click **Add**. Click **OK**.

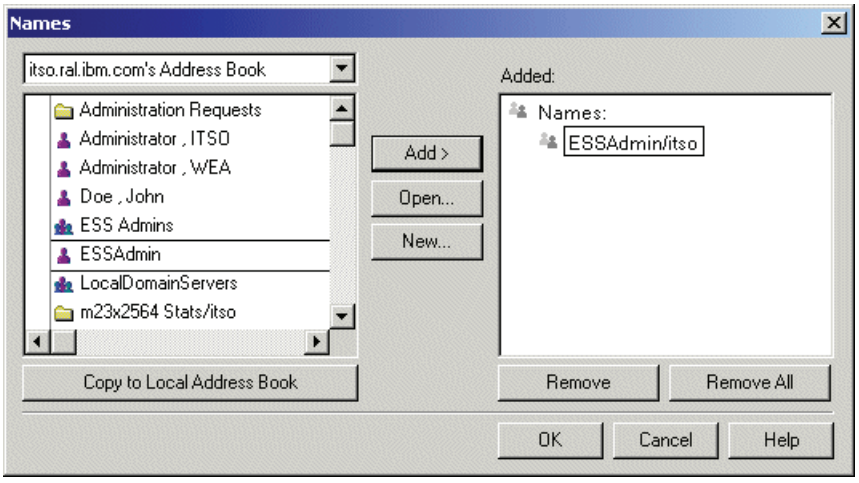


Figure 22-15 Add the Adapter Admin ID

5. The Adapter Admin ID should now appear on the Access Control List. Select **Person** from the User Type menu. Select **Manager** from the Access menu. Select the **Delete documents** check box.

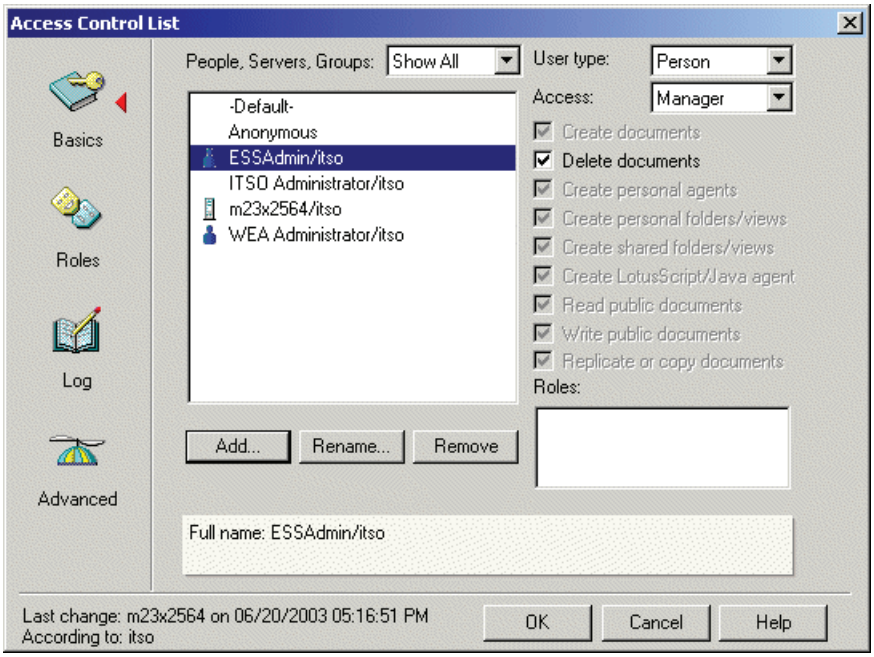


Figure 22-16 Access Control List - Basics

6. Click **Advance** in the left pane. Verify that the Maximum Internet name & password field is set to Editor or Manager.

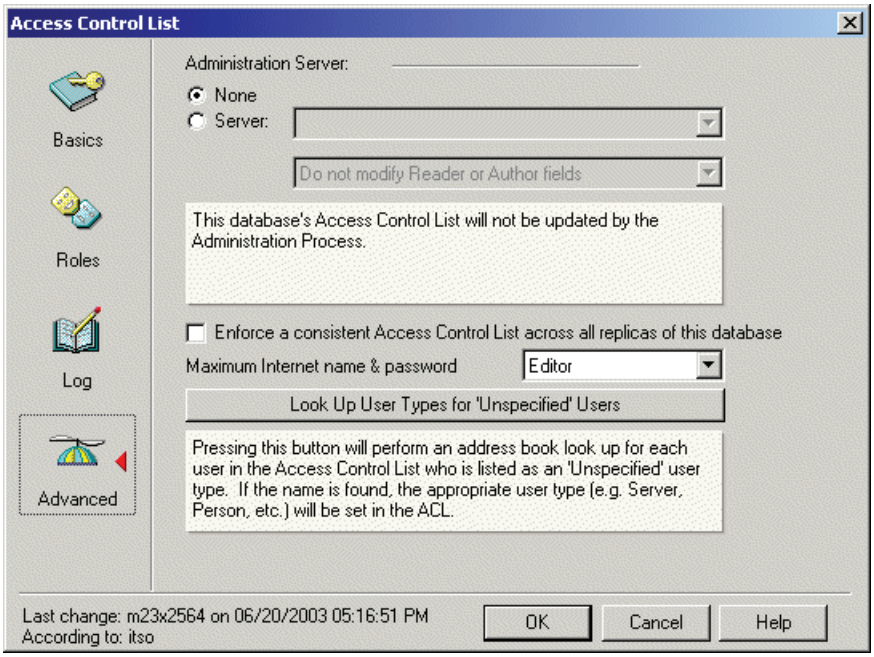


Figure 22-17 Access Control List - Advanced

7. Click **OK** and close the mail.box database.

22.3.2 Lotus Notes client installation and configuration

In order for the Lotus Domino Adapter to communicate with the Lotus Domino Server, a Lotus Notes client must be installed on the same machine as the WebSphere Everyplace Access server. Once installed, the Lotus Notes client needs to be correctly configured with the Adapter Admin ID.

Important: The Lotus Notes client must be installed before installing the Everyplace Synchronization Server.

Lotus Notes client installation

You should install Lotus Notes Version 6. This comes bundled with the WebSphere Everyplace Access CDs. The installation is straightforward:

1. Insert the CD and run the setup program.
2. Follow the on-screen instructions and accept the default settings right through to the end. The only thing to watch out for is when you are prompted for the

customer information. You will be asked whether to perform a Single User Install or a Multi-User Install. Select **Singe User Install**.

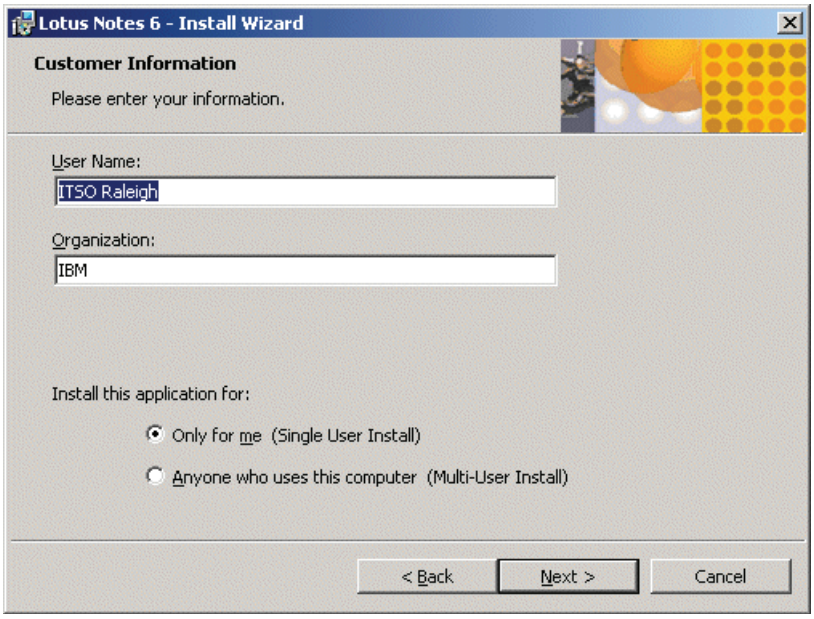


Figure 22-18 Lotus Notes 6 installation - customer information

Lotus Notes client configuration

Once the installation is complete, the Lotus Notes client needs to be configured with the Lotus Domino Adapter Admin ID.

1. Start the Lotus Notes client by clicking **Start -> Programs -> Lotus Applications -> Lotus Notes**.
2. Click **Next** on the Welcome window.
3. Enter the full name of your Adapter Admin ID and the Domino Server name. Click **Next**.

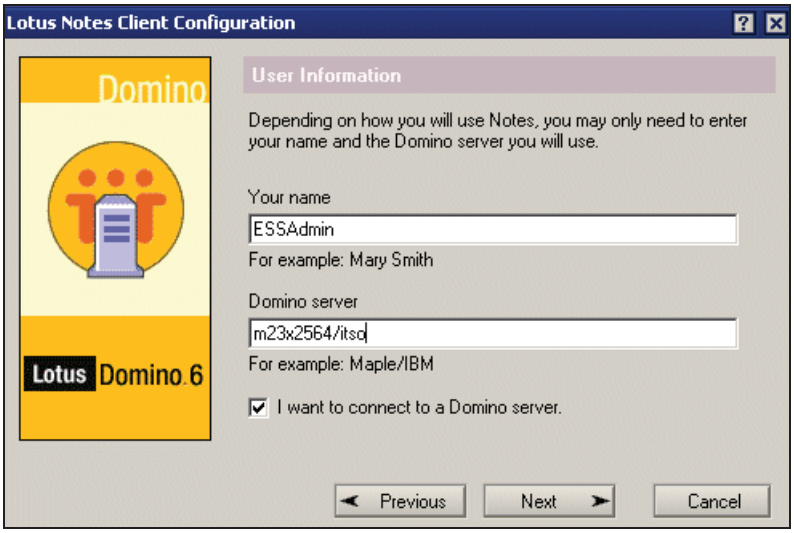


Figure 22-19 Lotus Notes client configuration - User Information

4. If Notes cannot find your ID file on the server, it will prompt you for an ID file. The ID file can be found in the <Domino_Administrator_dir>\Data\ids\people directory, where <Domino_Administrator_dir> is the directory where the Domino Administrator client is installed.

   Copy the ID file to a temporary directory on the WebSphere Everyplace Access server machine. Use the Browse button to locate the ID file. Then click **Next**.



Figure 22-20 Lotus Notes client configuration - Notes ID file

5. Click **Yes** if asked about copying the ID file to your data directory.
6. Enter the password for the Adapter Admin ID. Click **OK**.
7. Click **Next** on the Additional Services window.

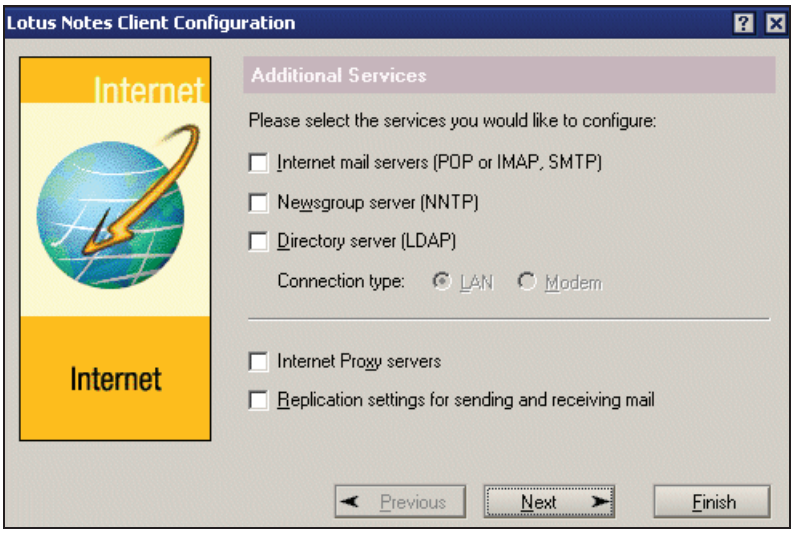


Figure 22-21 Lotus Notes client configuration - Additional Services

8. Click **OK** on the next window.

9. On the Setup window, select **No thanks, just give me the defaults**.
10. Close your Notes Client.

22.3.3 Lotus Notes User configuration

Every Lotus Notes User account that will be synchronized through ESS will need to grant appropriate access to the Adapter Admin ID. This can be done by the individual users or it can be done by the Domino Administrator using the Lotus Notes client that is installed along with the Lotus Domino Server.

1. If you are a user granting manager access to your database:
   a. Start up the Lotus Notes client and open your mailbox.
   b. Go to step 2.

   If you are a Domino Administrator granting manager access to a user's databases on behalf of the user:
   a. Start up the Lotus Notes client that was installed with the Lotus Domino Server. This can be done by running:

      `<Domino_root>\nlnotes.exe`

      where `<Domino_root>` is the root directory of Lotus Domino.
   b. Select **File -> Database -> Open**.
   c. Leave the Server field as `Local`. Locate the user's database under the mail directory. Click **Open**.

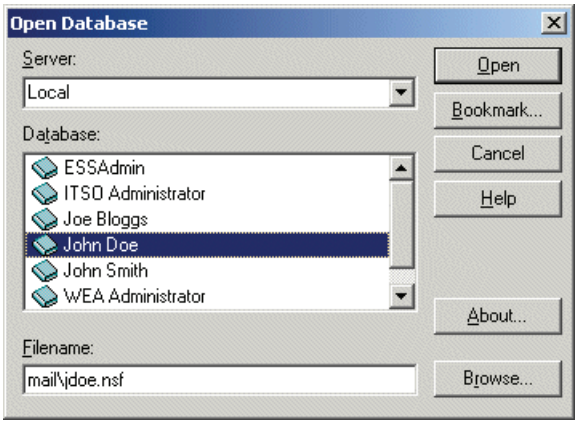


Figure 22-22 Open user's database

   d. Go to step 2.
2. Select **File -> Database -> Access Control**.

3. Click **Add** and then click the button to the right of the Person, server, or group field. Change the Address Book to the Domain Address Book by clicking the drop-down box at the top. Select the Adapter Admin ID and then click **Add**. Click **OK**.

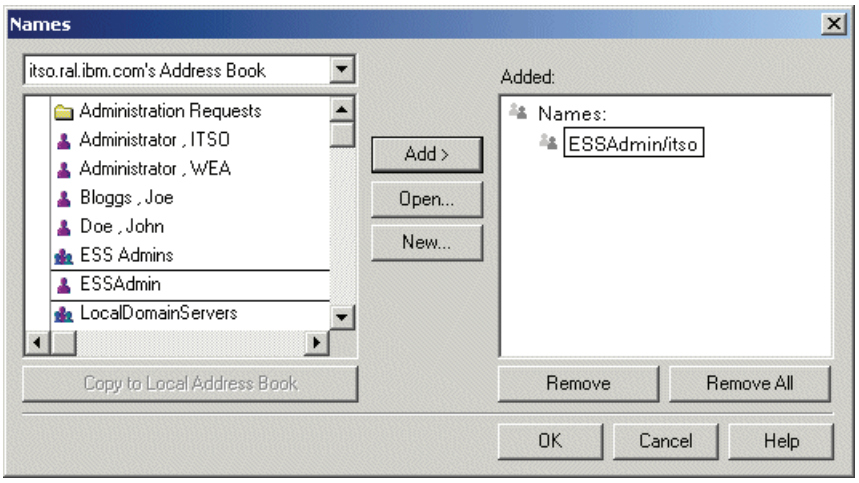


Figure 22-23 Add the Adapter Admin ID

4. The Adapter Admin ID should now appear on the Access Control List. Select **Person** from the User Type menu. Select **Manager** from the Access menu. Select the **Delete documents** check box.

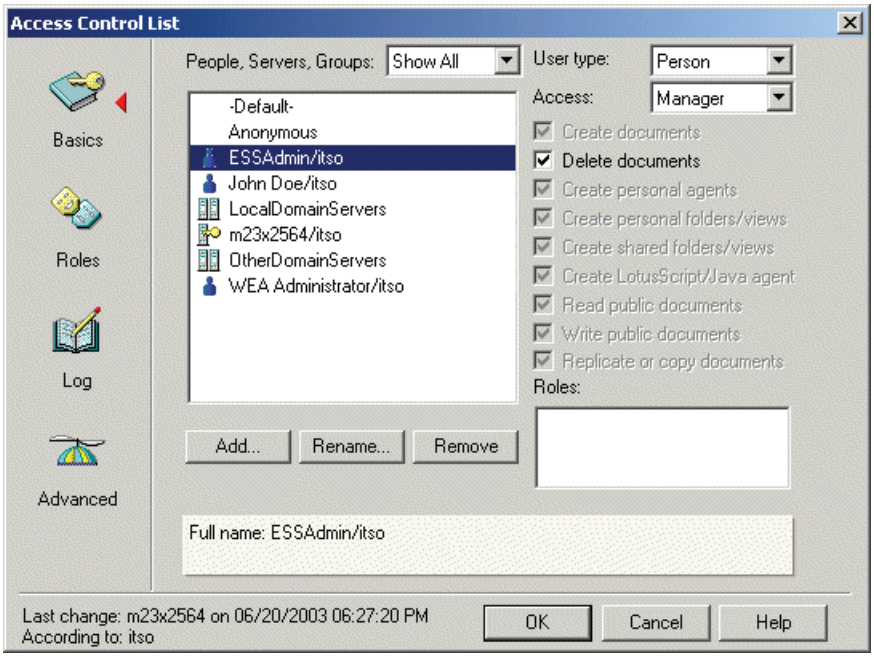


Figure 22-24 Access Control List - Basics

5. Click **Advance** in the left pane. Verify that the field Maximum Internet name & password is set to Editor or Manager.

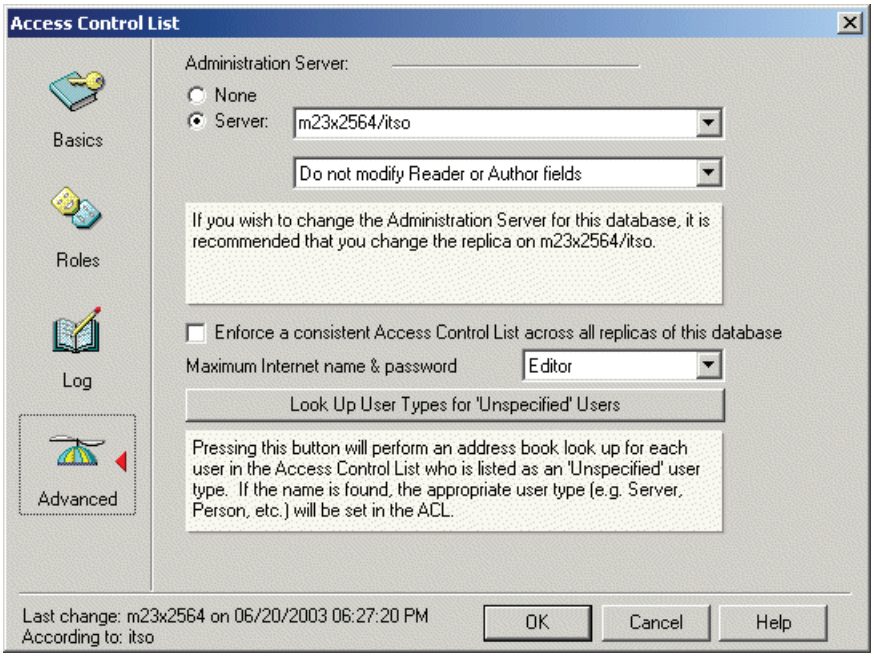


Figure 22-25 Access Control List - Advanced

6. Click **OK** and close the database.

22.3.4 WebSphere Everyplace Access server configuration

Once the local Lotus Notes client has been installed and configured, you can install your WebSphere Everyplace Access server. During the installation of the ESS component, the WebSphere Everyplace Access installation manager copies the *Domino mapping databases* to the data directory of the local Lotus Notes client. It is therefore important that the Notes Client be installed and configured with the Adapter Admin User before installing WebSphere Everyplace Access. After installing WebSphere Everyplace Access, we need to prepare the WebSphere Everyplace Access server for synchronization. This preparation involves:

- Creating WebSphere Everyplace Access users who will use the Synchronization feature of ESS.
- Configuring the Lotus Domino Adapter for synchronization with the Lotus Domino Server.
- Setting up a default device profile that will be inherited by each ESS user.

Create WebSphere Everyplace Access users for synchronization

Before users can synchronize their client device with PIM data using ESS, a WebSphere Everyplace Access user account needs to be created on the WebSphere Everyplace Access server. The WebSphere Everyplace Access user account will need to have the appropriate access rights on ESS-related portlets. Access to these portlets will allow the user to configure his synchronization settings. There are three ways to set up a user to have the appropriate access rights to these ESS-related portlets:

- Adding the user to the essusers or essadmins group. These two groups are created by the WebSphere Everyplace Access installer during installation.
- Adding an existing or new group as a member of the essusers or essadmins group. The user will be need to be a member of the group being added.
- Manually setting the access rights on the user or the group the user is a member of.

The users used in our examples were created as members of the group SyncGroup. SyncGroup was made a member of the essusers group. See Chapter 2, “Administration” on page 25 for instructions on creating and managing users and groups.

Configure Lotus Domino Adapter

The Lotus Domino Adapter needs to be configured with details of the Adapter Admin ID and names of all the servers the Adapter Admin ID has access to.

1. Log into WebSphere Everyplace Access as the Portal Administrator.
2. Select **Everyplace Synchronization -> Lotus Domino Adapter**.
3. During the installation of ESS, you should have been prompted for the Adapter Admin ID and password. Under Lotus Domino authenticator, verify that the Administrator ID and Administrator password is correct. Change if necessary.
4. During the installation of ESS, you should have been prompted for the details of one Domino Server. This Domino Server will be used to authenticate the Adapter Admin ID. Verify that this is correct. Change if necessary.

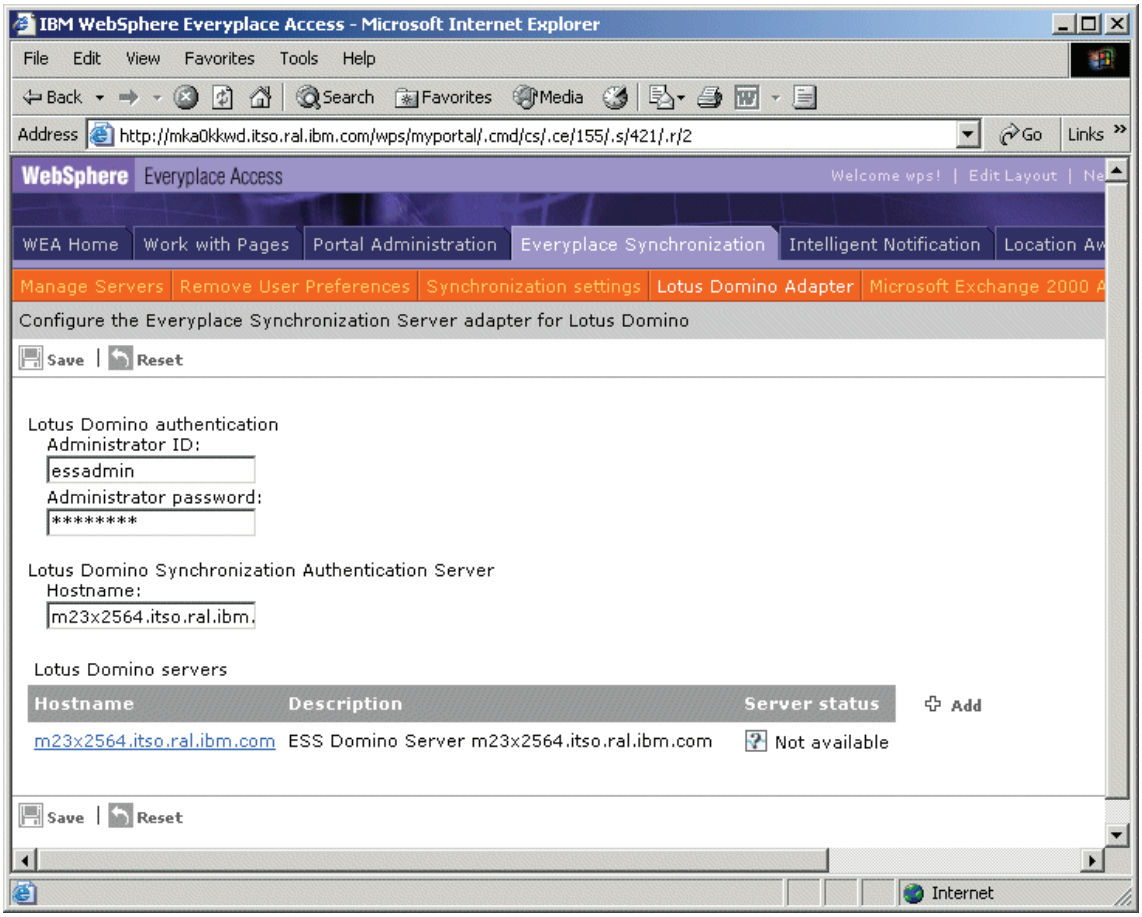


Figure 22-26 Configure Lotus Domino Adapter

5. During the installation of ESS, you should have been prompted for the details of one Domino Server. For our examples that follow, we will use the one Domino Server that was specified during the installation of ESS. Hence, this step will not be necessary if the Domino Server host name is correct.

   To add additional Domino Servers:

   a. You should see the name of the Domino Server added during the installation of ESS under the Lotus Domino Servers section. Click the **Add** button on the right side of this section.

   b. Enter the host name and description. Click **OK**.

To remove an existing Domino Server:

a. Under the Lotus Domino Servers section, click the Domino Server you wish to remove.

b. Click the **Remove this server** button.

6. Click **Save**.

7. Restart Everyplace Synchronization Server by clicking **Start -> Settings -> Control Panel -> Administrative Tools -> Services**. Locate the IBM Synchronization Server. Stop the service if it is running. Start the service.

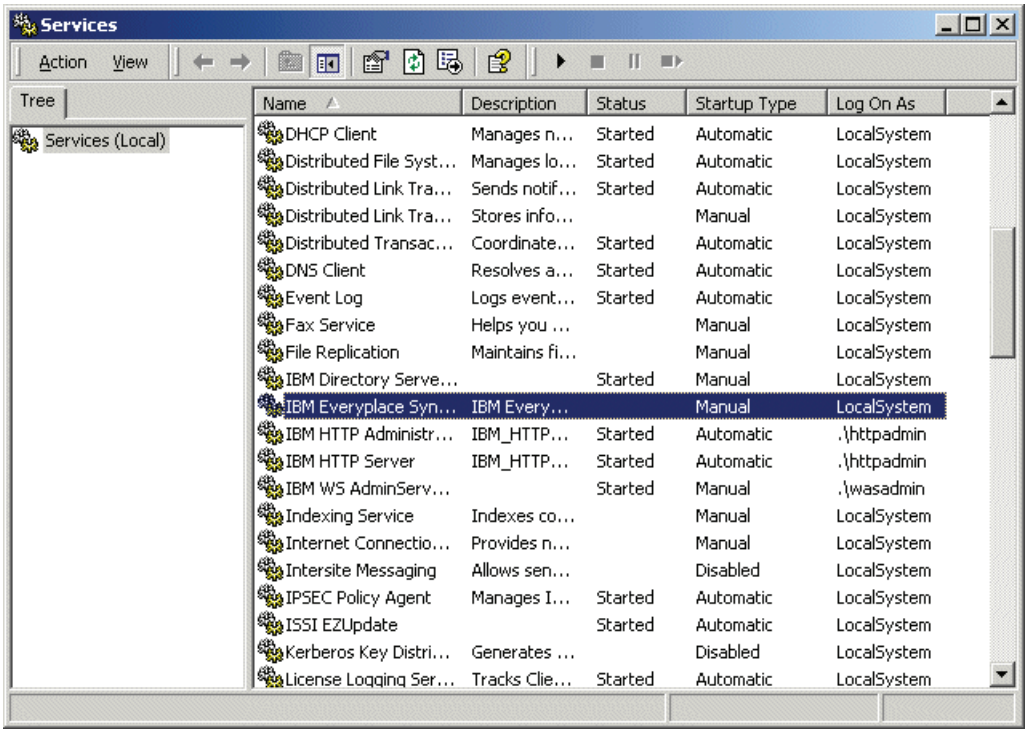


Figure 22-27 Restart the Everyplace Synchronization Server

8. Verify that the Domino Server has started by selecting **Everyplace Synchronization -> Manager Servers** or **Everyplace Synchronization -> Lotus Domino Adapter**. If the server has started, the word `Running` will be appear on the right side of your server under the Server status column.

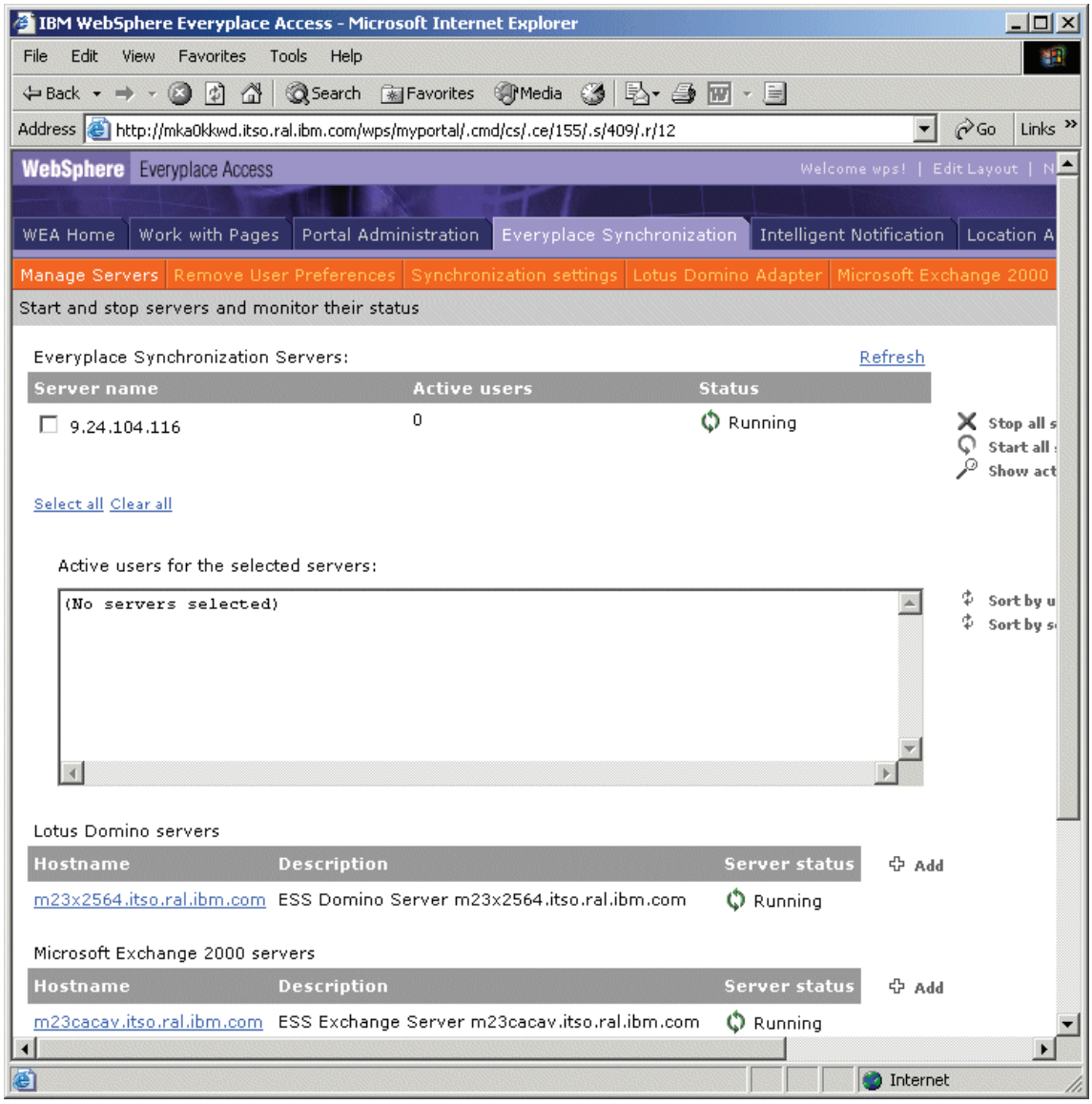


Figure 22-28 Manage Servers - Server Status

If the Domino Server has not started, check your settings for the Lotus Domino Adapter by selecting **Everyplace Synchronization -> Lotus Domino Adapter**. You may also want to check the ESS logs for errors. The ESS logs are located in <ESS_dir>\logs, where <ESS_dir> is the directory where ESS is installed.

You can also check the Domino Server console to see if there were any attempts to access the database of the Adapter Admin User. If ESS is correctly talking to the Domino Server, it should look something like Figure 22-29.

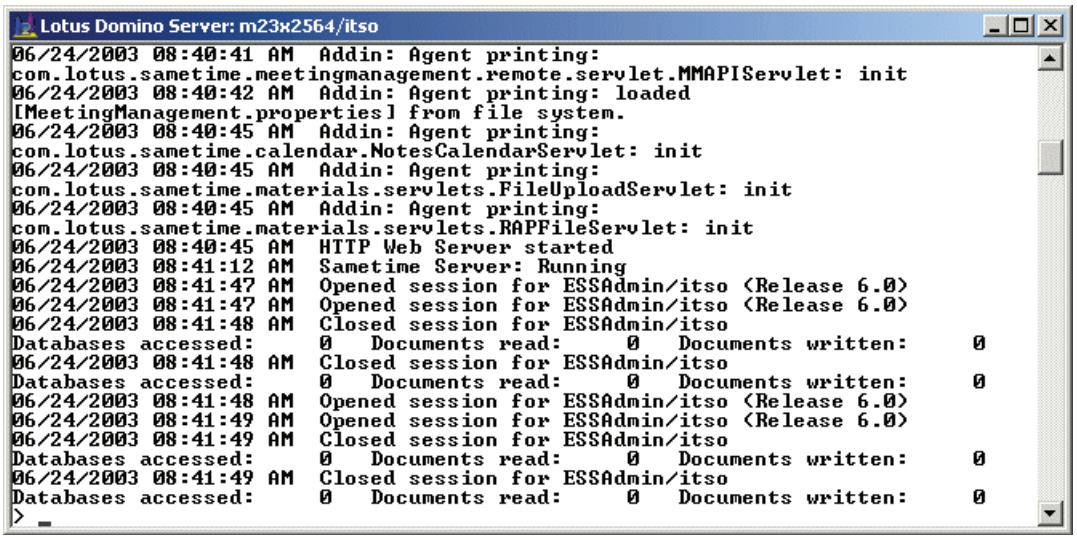


Figure 22-29 Lotus Domino Server console

If the server is not running and you cannot find anything wrong with your settings for the Lotus Domino Adapter, you might want to check that your Domino Server configuration is correct. Refer to 22.3.1, “Lotus Domino Server configuration” on page 714 for details on configuring your Domino Server.

Create default device profile

Before a user can begin synchronizing his Lotus Notes PIM and e-mail data, he must have a device profile associated with his WebSphere Everyplace Access user account. A device profile is a group of settings that limit the scope of the synchronized data. For user-specific device profiles, it also stores information about:

- The Domino Server the user wants to connect to.
- The user name and password of the Lotus Notes account whose PIM and e-mail data the user wants to synchronize.

A default device profile is a device profile that can be inherited by user-specific device profiles. It does not store information about the Domino Server name,

user name or password. It can only be created by a user who has permission to access the ESS Admin portlets.

To create a default device profile that can be inherited by user-specific device profiles, perform the following steps.

1. Log in as wpsadmin or somebody with access to ESS Admin portlets.
2. Select **Everyplace Synchronization -> Synchronization settings**.
3. Click **Create new device profile**.
4. Enter a name for your new profile in the Provide a name for your new profile field, and enter a description for the profile in the Provide a device description for your profile field. Click **Next**.

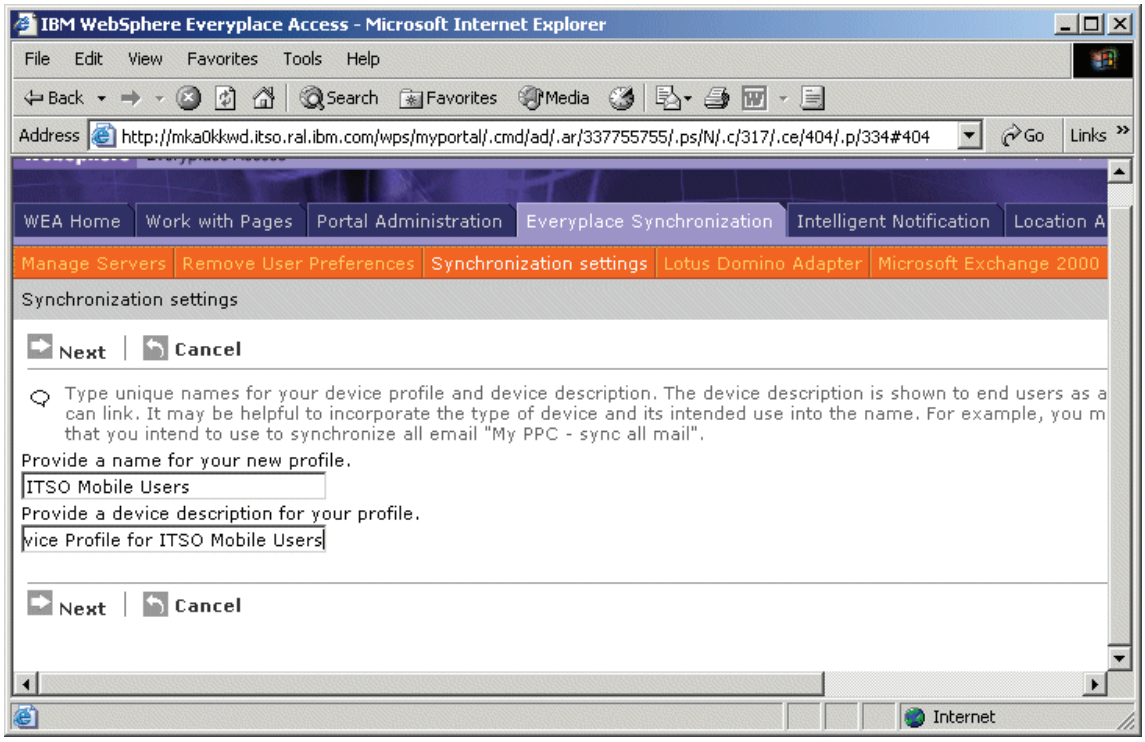


Figure 22-30 Enter Profile name and description

5. You will be presented with a list of PIM features that can have filters applied. We will start with the e-mail filter. Click the **E-mail** filter.
6. On the next window, you will see a number of filters that can be applied to e-mail synchronization. The top section of the page (the section under Creating profile) lists filters specific to the default device profile you are

creating. It will be empty to start with. The bottom section of the page (the section under Global filters) lists filters that will be applied to *all* e-mail synchronization.

We'll start with the Global filters. For the purposes of this example, we will demonstrate the use of the attachment filter to filter out attachments that do not have an extension of .doc and whose size is greater than 250 KB. Check the radio button **Include only files with the following extensions** and enter `doc` in the text box. Select **250 KB** from the drop-down list.

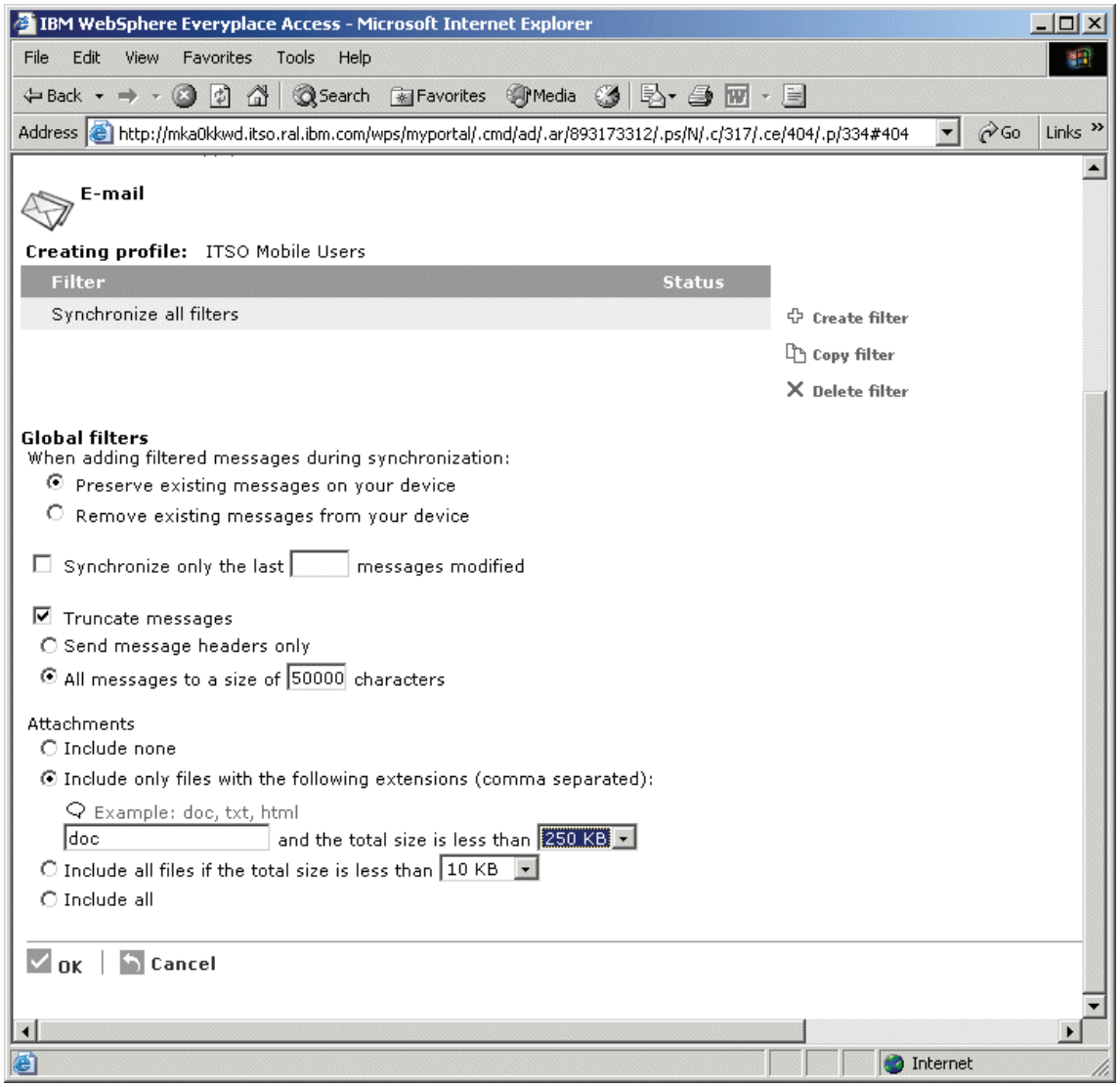


Figure 22-31 e-mail filters

7. We will now create a filter specific to the default device profile. Click **Create filter**.
8. Enter `Only e-mails received in the last week` in the text box under Provide a descriptive name for this filter.

9. Click **Date**. You will be presented with three drop-down lists. On the first drop-down list, select **Date received**. Select the radio button next to the second drop-down list and select **This week**.

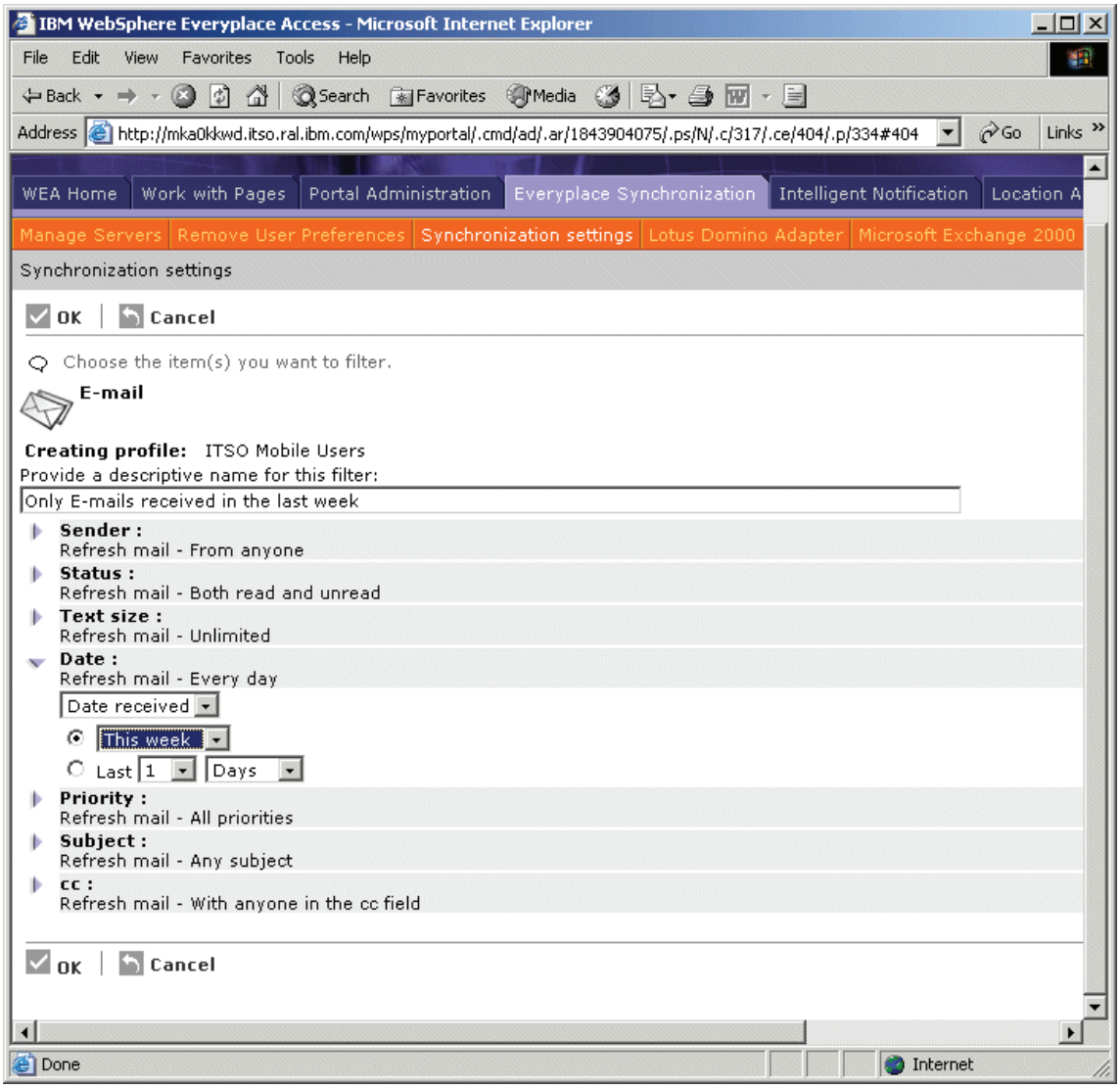


Figure 22-32 E-mail filters for default device profile

10. Click **OK**.
11. You will be taken back to the previous window. Click **OK**.

12. This completes our filter configuration for e-mail. We will now add filters for Calendar. Click the **Calendar** filter.
13. We will leave the Global filters as they are. Click **Create filter**.
14. Enter `Three Month Window` in the text box under Provide a descriptive name for this filter.
15. Click **Start date of entry**. Select **This time interval** from the drop-down list. The page will be refreshed and you will be presented with drop-down lists to define the boundary of your time interval. Use the drop-down lists to select **1 Months in the past to 2 Months in the future**.

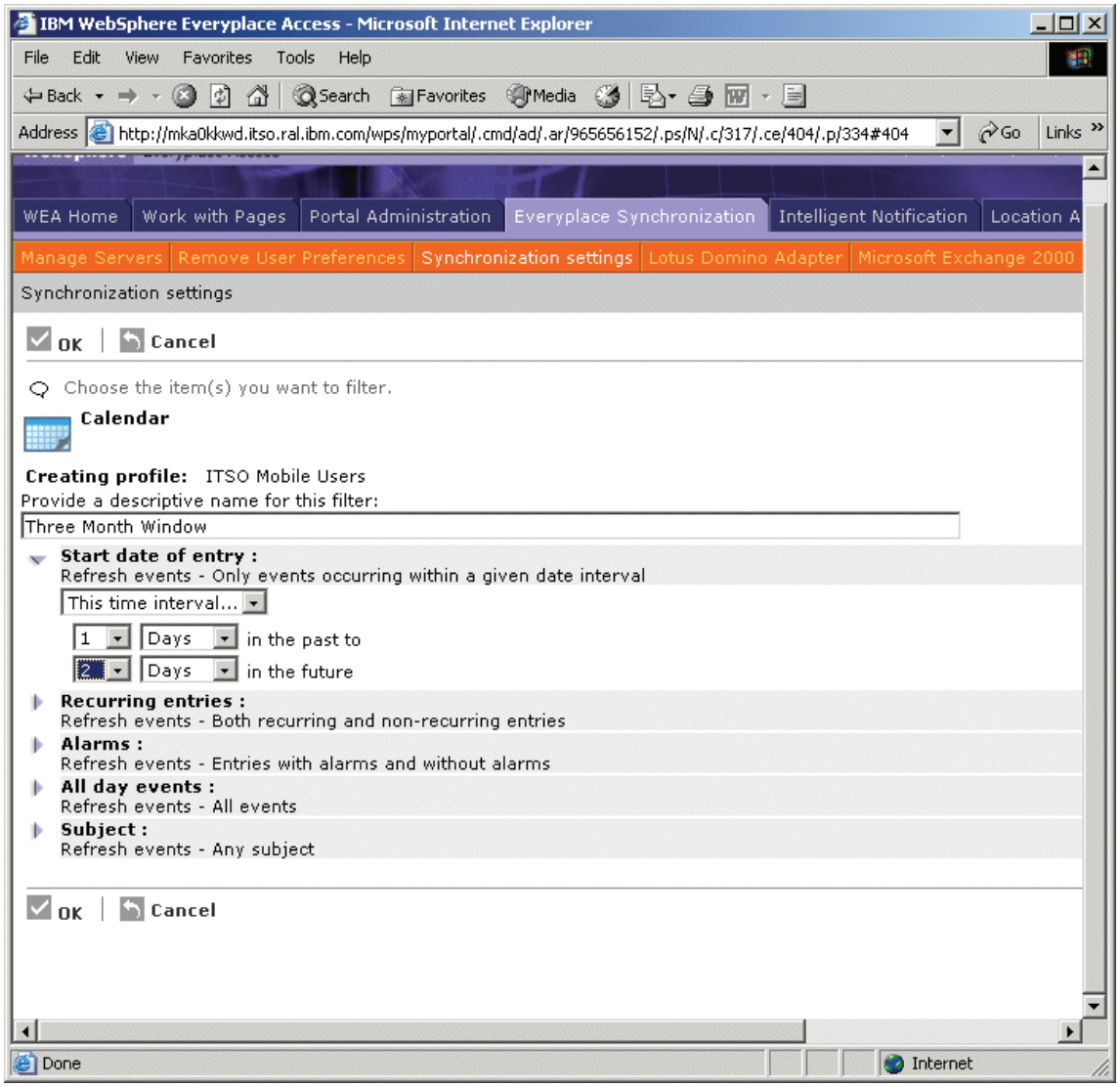


Figure 22-33 Calendar filters

16. Click **OK**.
17. You will be taken back to the previous window. Click **OK**.
18. We will not add any filters for Contacts. Click the **Tasks** filter.
19. We will leave the Global filters as they are. Click **Create filter**.

20. Enter `Only high priority tasks` in the text box under Provide a descriptive name for this filter.
21. Click **Priority** and select **High only** in the drop-down list.

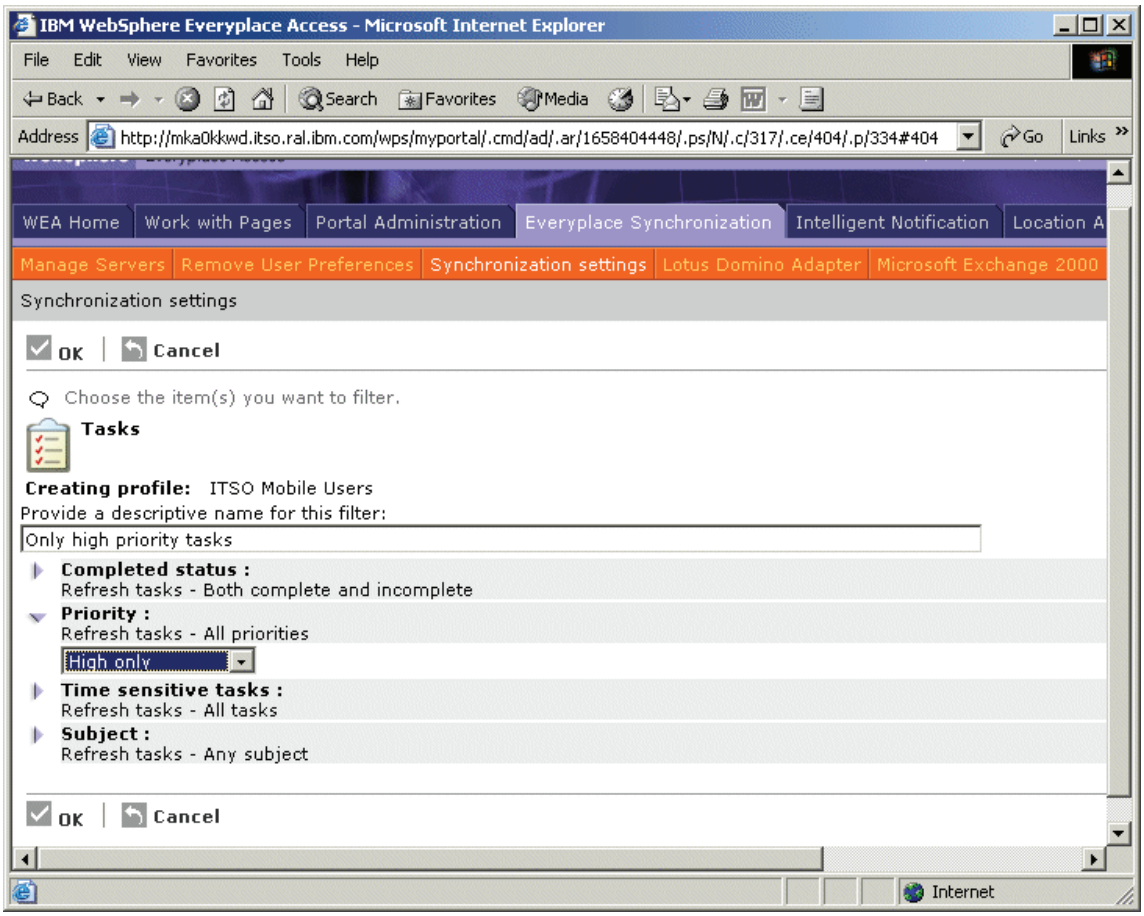


Figure 22-34 Tasks filter

22. Click **OK**
23. You will be taken back to the previous window. Click **OK**.
24. We will not add filters to Notes. Click **Next**.
25. You will be prompted to use information from either the server or the device when there is a conflict. Select **Server** and click **Next**.
26. You will be presented with a summary of your profile. You should see three active filters.

27. Click **OK** to save the profile. On the next window, you will see that the new default device profile (ITSO Mobile Users in this example) has been added to the list of available profiles.

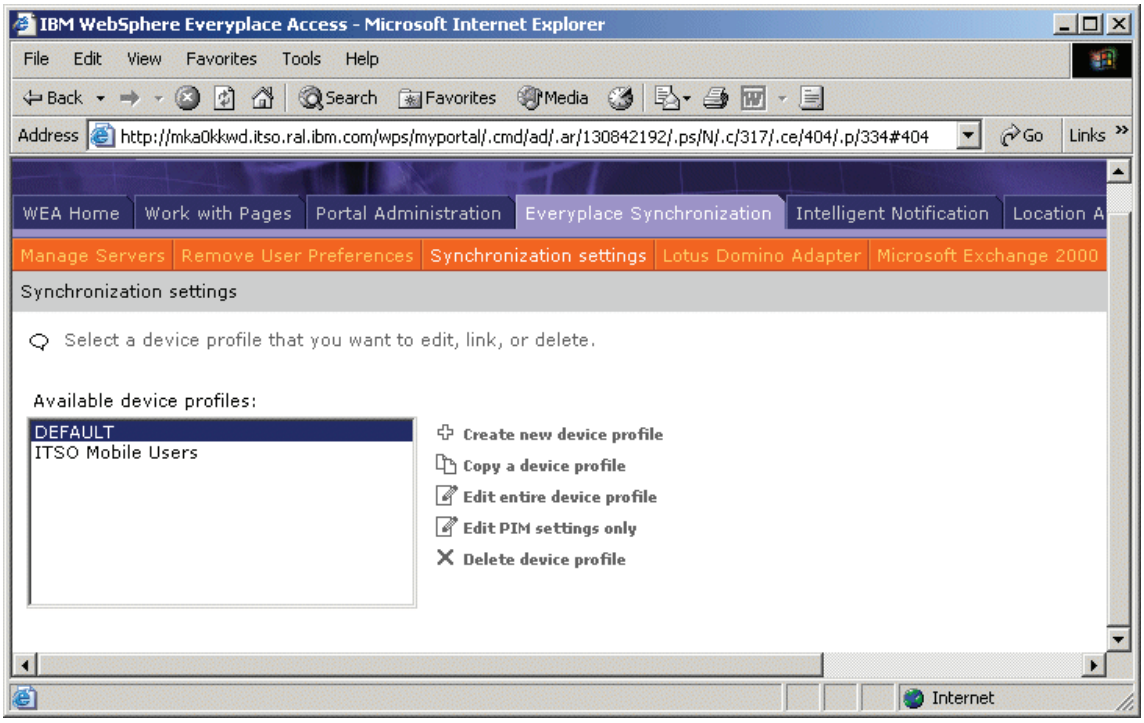


Figure 22-35 Available device profiles

22.3.5 WebSphere Everyplace Access user configuration

Before a user can start synchronizing his PIM data, he needs to configure his WebSphere Everyplace Access user account with the appropriate synchronization settings. This involves creating at least one user-specific device profile. For more information on device profiles, refer to “Create default device profile” on page 741.

The following shows how to create a user-specific device profile by creating a user device profile and linking it to a default device profile created previously:

1. Log in as the ESS user.
2. Select **WEA Home -> Configure**.
3. Click **Create new device profile**.
4. Enter the name of the new device profile and click **Next**.

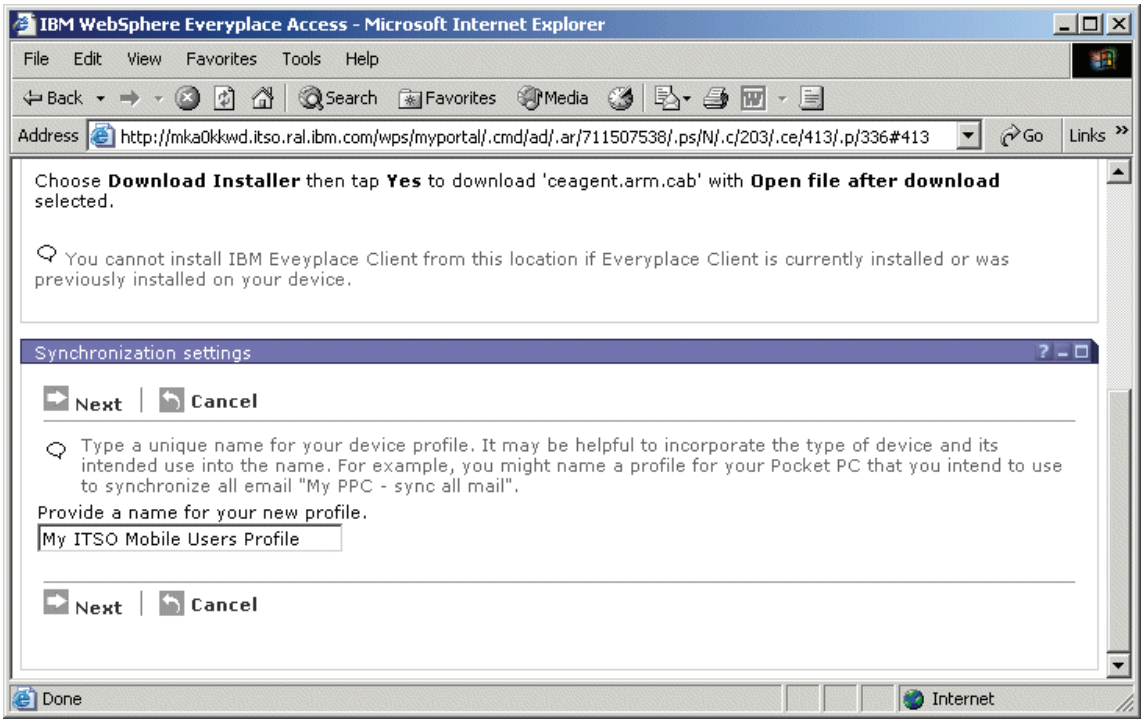


Figure 22-36 Enter device profile name

5. Select the Domino Server from which PIM data is to be retrieved. Enter the Lotus Notes user name and password. Select the appropriate Notes template. If you select **iNotes™**, you need to enter the database folder path.

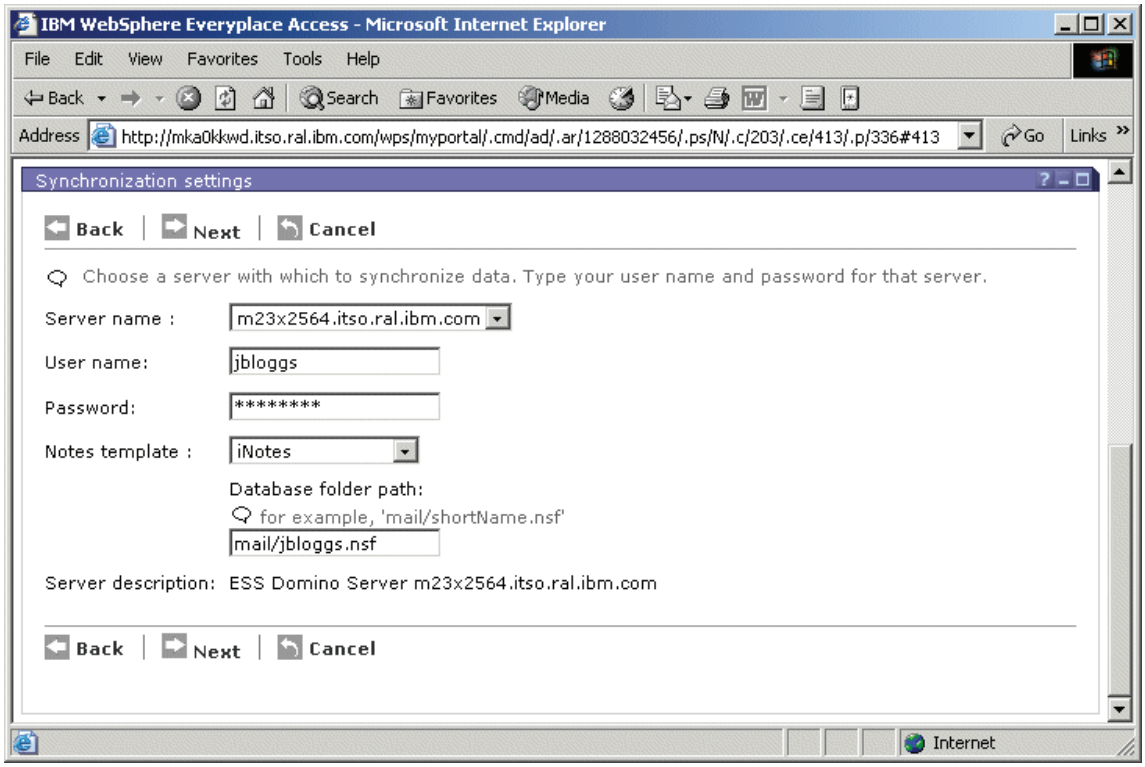


Figure 22-37 Synchronization Settings - Domino Server and user information

6. Click **Next**.
7. On the next window we are given an option to link our device profile to an existing default device profile. We will link it to the one we created earlier. Select the top radio button and select the default device profile in the list of available device profiles.

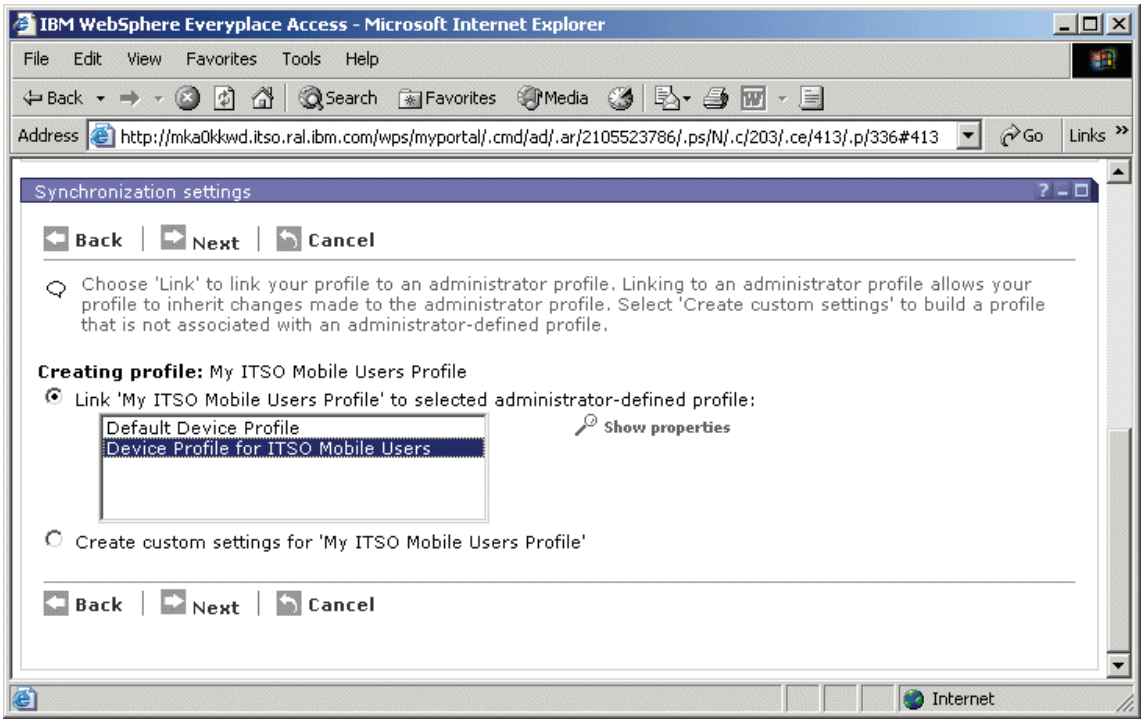


Figure 22-38 Link user device profile to an existing default device profile

8. Click **Next**. You will see a list of synchronization settings to be used on the *first* synchronization. Following the first synchronization, the default device profile that we had chosen to link with will be used to filter out what we don't want. On subsequent synchronizations, the default device profile (Mobile ITSO Users in our example) will be used. You can select to edit these initial synchronization settings by clicking the appropriate filter buttons. For the purposes of this example, we will not change these settings.

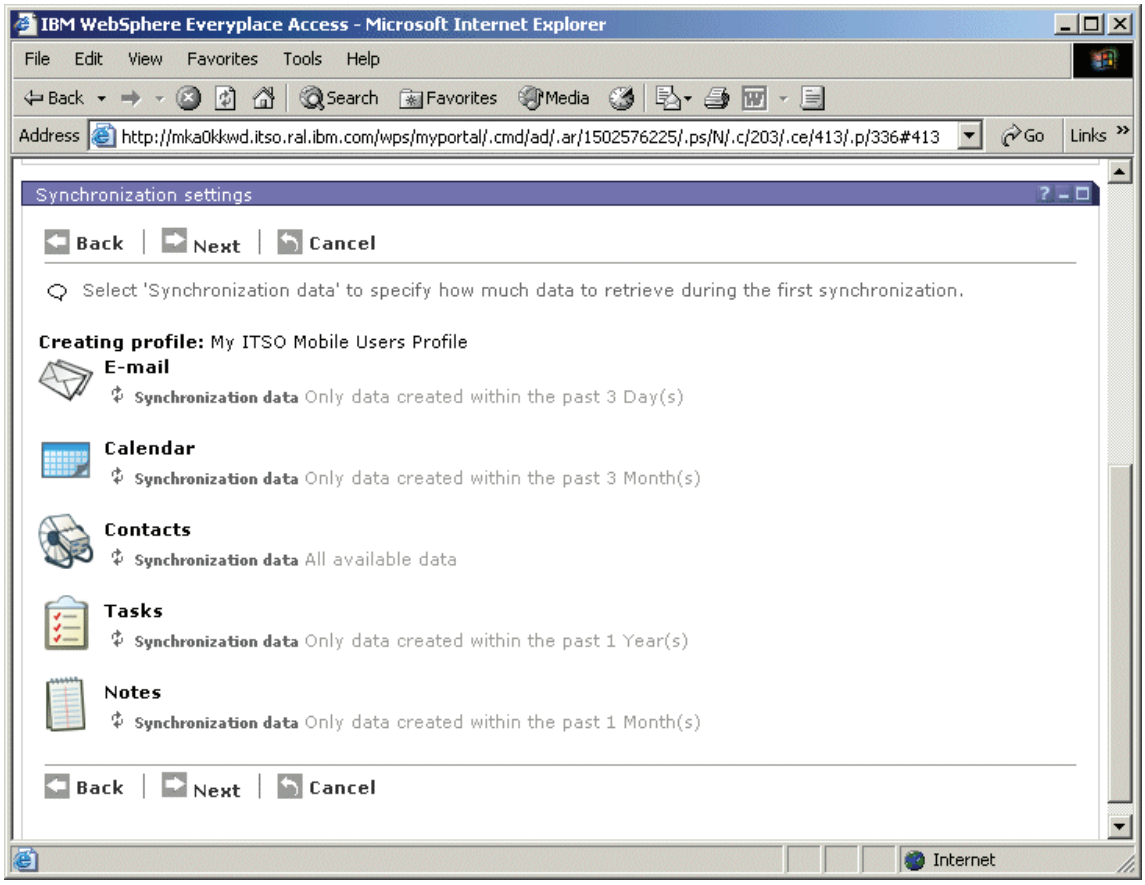


Figure 22-39 Initial synchronization settings

If we had not chosen to link to a default device profile, not only would we be given the opportunity to edit the initial synchronization settings, we would also be given the opportunity to create synchronization settings for subsequent synchronizations.

Click **Next**.

9. We now need to set the time zone and locality. Using the drop-down list, select the time zone and then click **Get locales for this time zone**. Using the second drop-down list, select the locale and click **Next**.
10. You will be presented with a summary of the profile. Click **OK**.
11. You will be brought back to the first page where you will see your newly created profile in the list of Available device profiles.

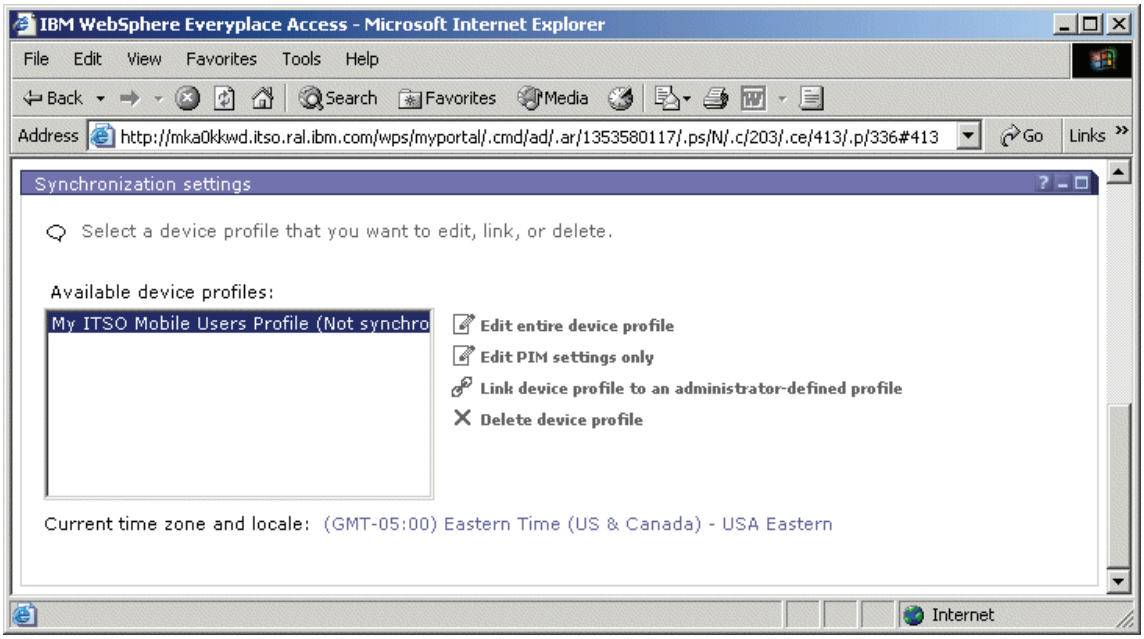


Figure 22-40 Available device profiles

22.3.6 WebSphere Everyplace Access Client configuration

The last step in the whole configuration process is to configure our client device to talk to the Everyplace Synchronization Server. Configuration of the client device will depend on the type of device we have. In this section, we cover both the Pocket PC and Palm device. For the Palm device, we will be using a Palm OS 5.2 device for our example. Palm OS 4.1 and 5.0 devices should work in a similar fashion. It is assumed that you have already installed the WebSphere Everyplace Access Client on these devices. Refer to Chapter 4, “Everyplace Client” on page 81 for more information on installing the WebSphere Everyplace Access Client.

Configuring the Pocket PC

Important: In order to use PIM and e-mail synchronization with the Pocket PC, you need to apply End User Update 3 (or later). This is a ROM image that can be downloaded from the manufacturer’s Web site. Instructions on applying the image should also be available at the manufacturer’s Web site.

1. Start the WebSphere Everyplace Access Client on the Pocket PC. **Start -> Everyplace Client**.

2. If prompted, enter the Portal login user name and password.
3. Select **Home** then **My settings** from the drop-down list.

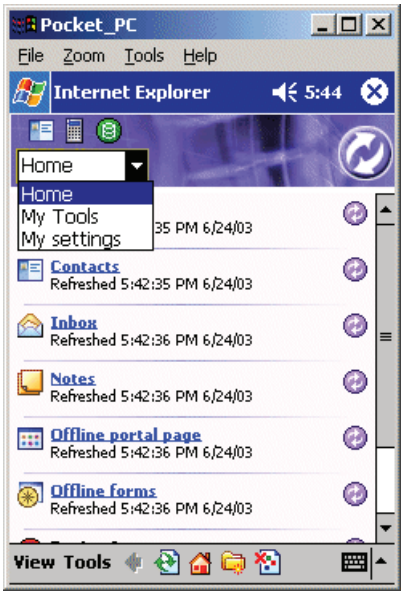


Figure 22-41 Select My settings

4. Using the drop-down list, select the **Active network profile**. For the purposes of this example, we will just edit the default network profile. Refer to Chapter 4, “Everyplace Client” on page 81 for more information about network profiles on the Pocket PC.
5. Enter the server name of the Everyplace Synchronization server.

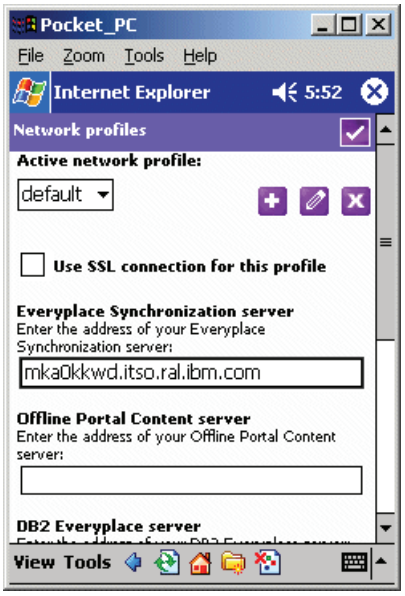


Figure 22-42 Network profiles

6. Select the checkmark in the top-right corner to save the network profile.
7. This completes the configuration of the Pocket PC for synchronization.

Configuring the Palm OS 5.2 device

1. For the purposes of this demonstration, we will be using VersaMail. If you intend to use VersaMail as your mail client, you must first create a VersaMail account before configuring the SyncML client. It is assumed that you have already installed VersaMail on your Palm device.

   On the **Applications** page, click **VersaMail**.
2. Select **Accounts -> Account Setup**.
3. Click **New** to create a new VersaMail account.
4. Enter the account name. Select **Other** as the mail service. Select **IMAP** as the protocol. Select **Synchronize Only Account**. Click **Next**.

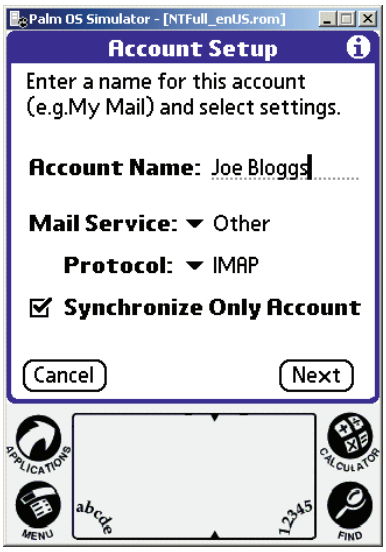


Figure 22-43 VersaMail - Account Setup

5. Enter the user name and password. It does not really matter what you enter here because the SyncML client will not use it. Click **Next**.
6. Enter the e-mail address. As for the Incoming Mail Server and the Outgoing Mail Server fields, it does not matter what you enter, since the SyncML client will not use it. Click **Next**.



Figure 22-44 VersaMail - Account Setup

7. Click **Done**. It should take you to the in-box of your new VersaMail account.
8. Now that we have created a VersaMail account to use for synchronization, we will begin configuring the SyncML client. Click the **SyncML** icon on the Applications page.
9. Click the horizontal bar at the top that reads **SyncML Client**. A drop-down list will appear. Select **PIM Server** under Preferences.



Figure 22-45 Synchronization Preferences

10. Enter the user name and password of the WebSphere Everyplace Access user. In the Server IP field, enter the WebSphere Everyplace Access server name or IP address.

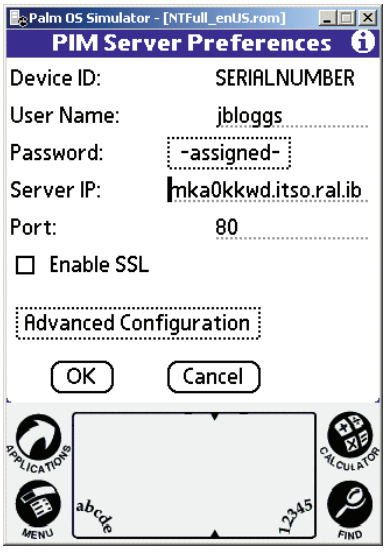


Figure 22-46 PIM Server Preferences

11. Click **OK**.

12. Click the horizontal bar at the top and this time select **PIM Subscriptions**.
13. You will be presented with a list of PIM components to configure. For each component there is a drop-down list to select Skip, Sync or Refresh.
    - Select **Skip** if you do not want to synchronize or refresh that component.
    - Select **Sync** if you want synchronization performed on that component when synchronization is initiated.
    - Select **Refresh** if you only want to either refresh the client with data from the server or refresh the server with data from the client.

      For the purposes of this example, we will not be exercising the Refresh option. On the Mail component, you can select either to use the **Standard Mail** client or **VersaMail**. Select **VersaMail** if you want support for e-mail attachments. The target field can stay as it is.

    For the purposes of this example, we will select **Sync** for all components and **VersaMail** for the mail client.

    When you select **VersaMail**, it will prompt you for the account name. Enter the account name you specified when you configured VersaMail.

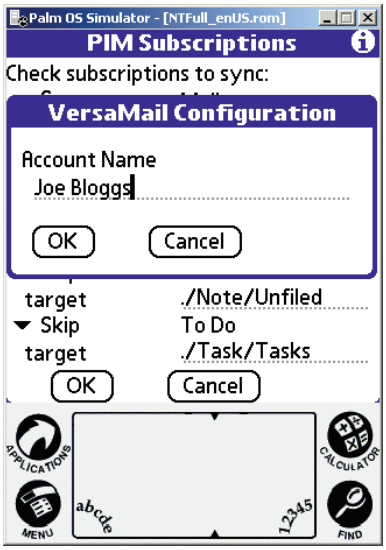


Figure 22-47 Enter e-mail account name



Figure 22-48 PIM Subscription settings

14. Click **OK** to save your settings.

15. This completes the configuration of the Palm OS 5.2 device for synchronization.

22.4 Usage

In this section, we demonstrate the use of the Everyplace Client to perform synchronization of PIM and e-mail data. We also discuss the Online PIM and e-mail portlets that are provided by Everyplace Access. Although not directly related to synchronization, these portlets are useful in verifying that the synchronization of data between the device and the servers are performed correctly. We will also be using these portlets to create some data for our synchronization demonstration.

22.4.1 Online PIM portlets

The Online PIM portlets provided by Everyplace Access allows a user to access his PIM data through a Portal-based interface. WebSphere Everyplace Access will act as a proxy to the actual PIM servers. Every user belonging to the essusers group, either directly or indirectly, will have access to these portlets. Before we can use these portlets to create and access data, we need to configure them with the appropriate PIM server and account details.

1. Log in as the ESS user.
2. Select **WEA Home -> Notes**. You will be presented with a Portal page with five portlets, one for each of the PIM components: Lotus Notes Mail, Lotus Notes Contacts, Lotus Notes Notebook, Lotus Notes Calendar, and Lotus Notes To Do List. Because these portlets have not been configured, they will contain some message telling you to edit the configuration details.

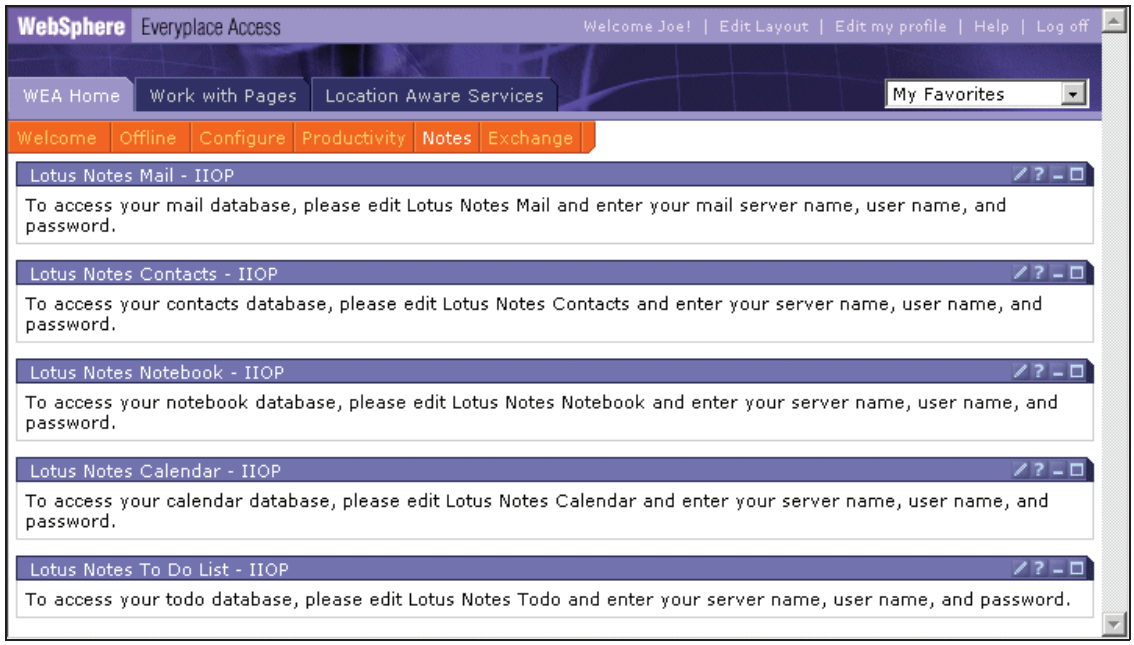


Figure 22-49 Lotus Notes PIM Portal page

3. We will now configure the Lotus Notes Mail portlet. Click the **Edit** button (the farthest left button on the portlet title bar).
4. If this is the first time you've clicked the **Edit** button, you will get a message that looks like Figure 22-50 on page 763.

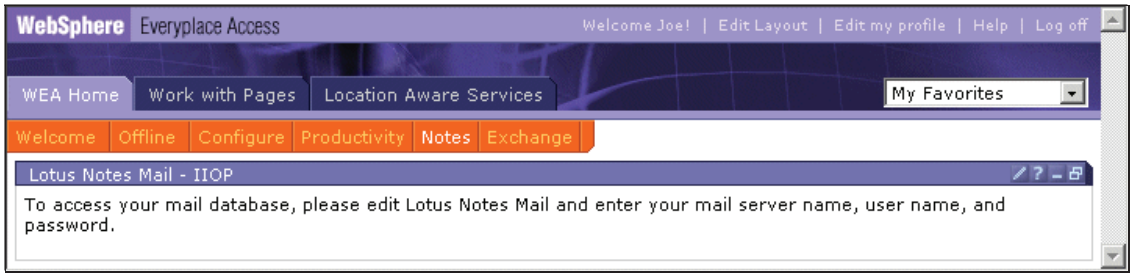


Figure 22-50 Message for first-time configuration

Click the **Edit** button again.

5. On the Configure Domino Server Parameters window, enter the server name, database folder path, user name and password. You can also set the number of items to display per page and whether to display the time and size of each e-mail.

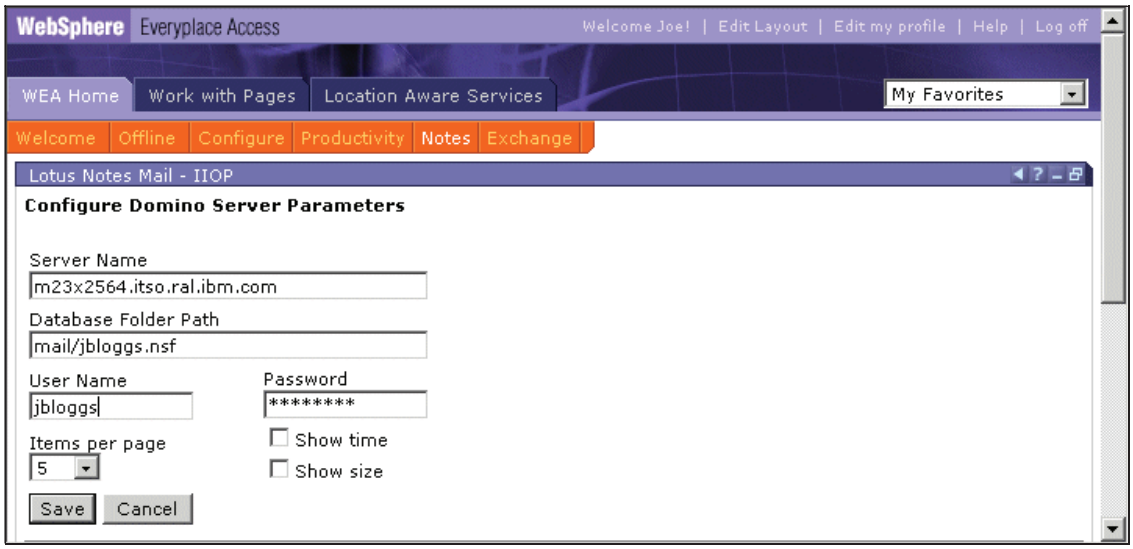


Figure 22-51 Configure Domino Server Parameters

6. Click **Save** to save your changes. You will be taken back to the previous window. The Lotus Notes Mail portlet will now show the in-box of the Lotus Notes e-mail account.

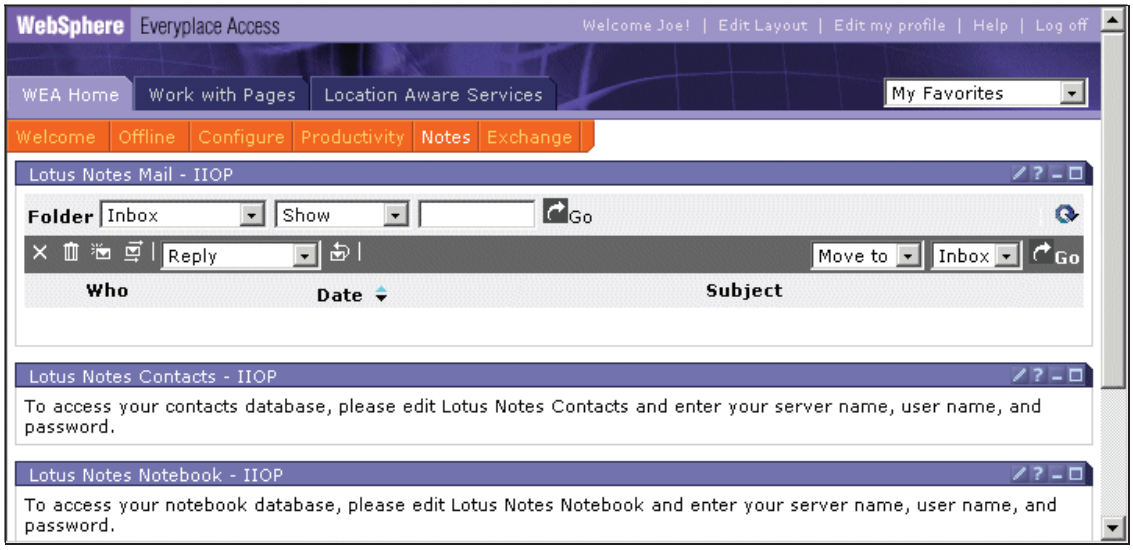


Figure 22-52 Lotus Notes Mail portlet

7. We will now configure the Lotus Notes Contacts portlet. Click the **Edit** button for that portlet.
8. If this is the first time you've clicked the **Edit** button, you will get a message that looks like Figure 22-53.

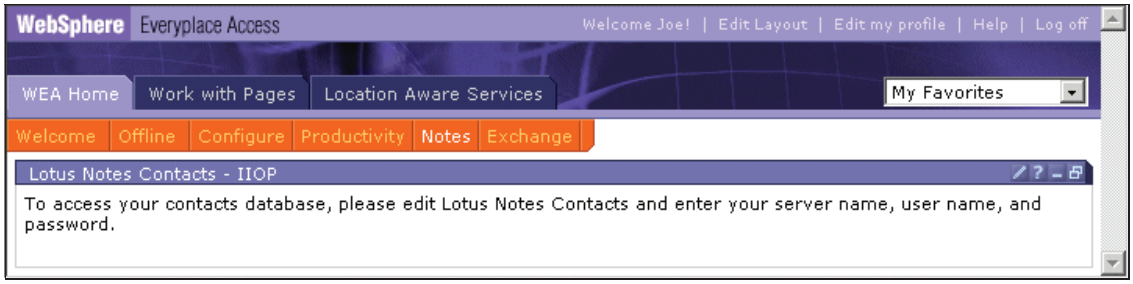


Figure 22-53 Message for first-time configuration

Click the **Edit** button again.

9. On the Configure Domino Server Parameters window, enter the server name, database folder path, user name and password. You can also set the number of items to display per page.

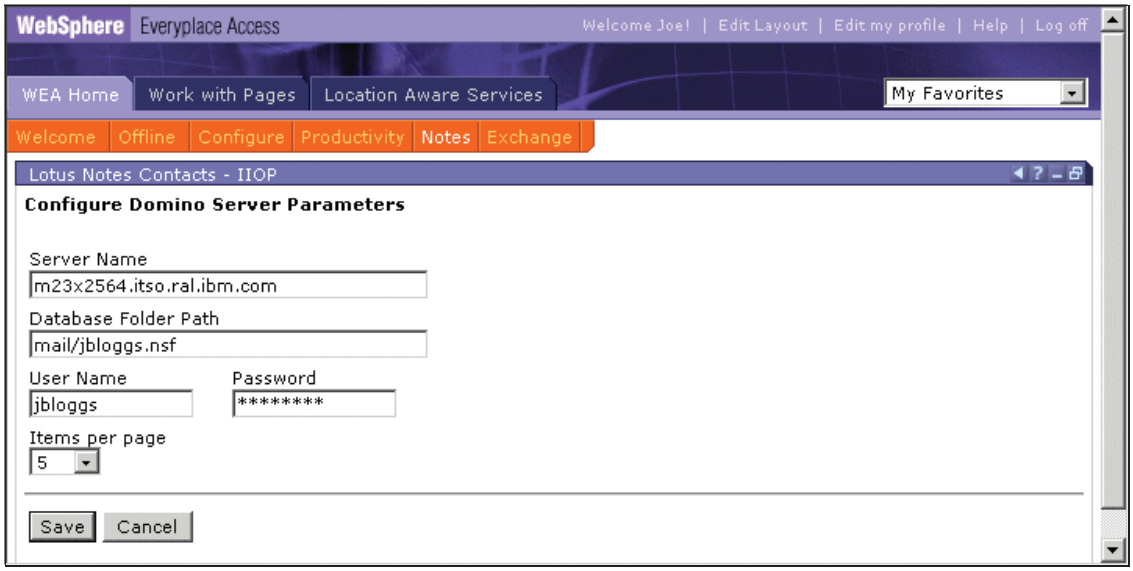


Figure 22-54 Configure Domino Server parameters

10. Click **Save** to save your settings. You will be taken back to the previous window. The Lotus Notes Contacts portlet will now show your contacts (if you have any).

11. Configuration of the other PIM portlets are similar to the configuration of the Calendar portlet. Once you've configured all the portlet, you should end up with a Portal page that looks like Figure 22-55 on page 766.

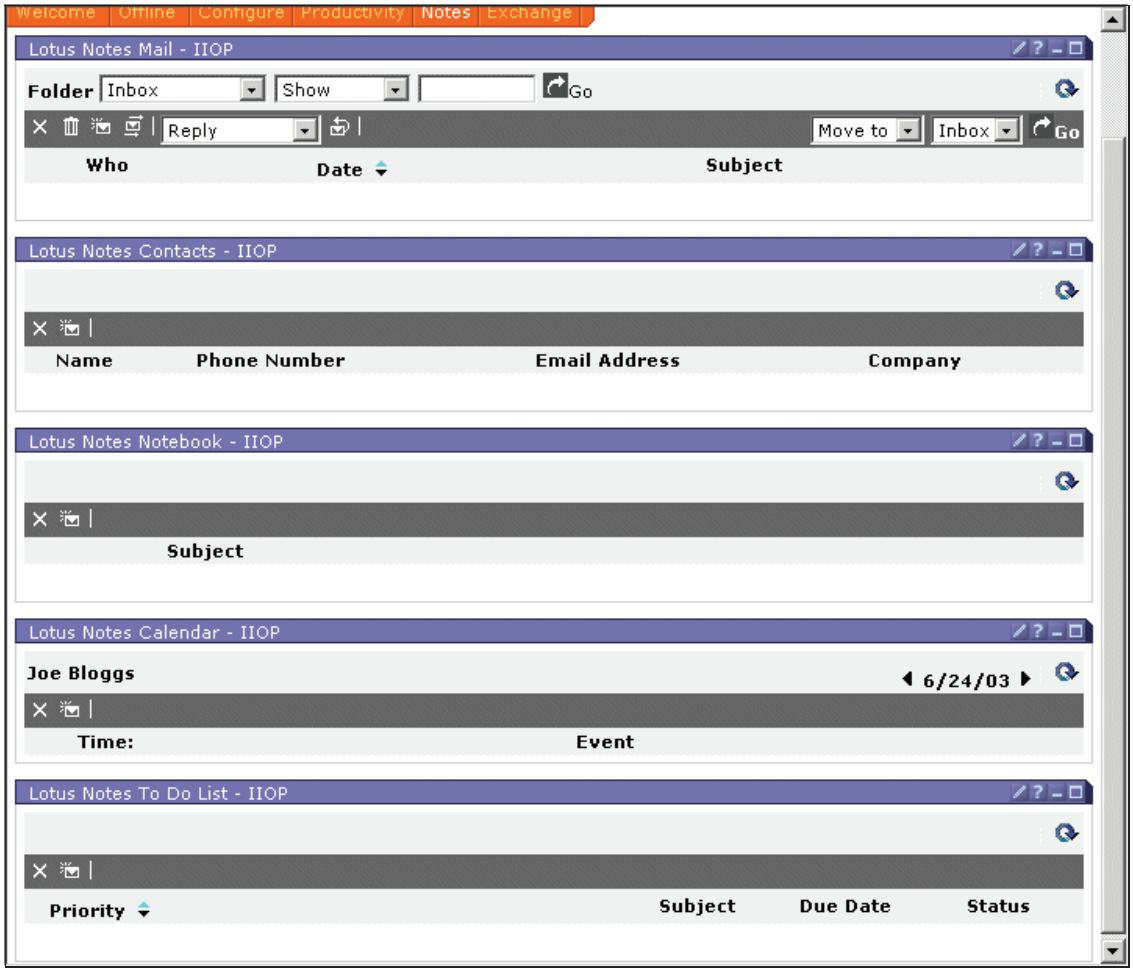


Figure 22-55 Notes Portal page showing all PIM portlets

22.4.2 Setting up synchronization scenarios

To demonstrate synchronization using ESS, we first need to set up some initial PIM data that can be synchronized. We will start by creating a contact, a note and a task through the online PIM portlets. We then send to our user a few e-mail notes with attachments using a Lotus Notes client. Finally, we schedule a recurring and non-recurring appointment using a Lotus Notes client.

Create contact

1. Log in as the ESS user.

2. Select **WEA Home -> Notes**.
3. In the Lotus Notes Contacts portlet, click the **New Contact** button. It is the button with an envelope inside, located on the right of the button with an X.

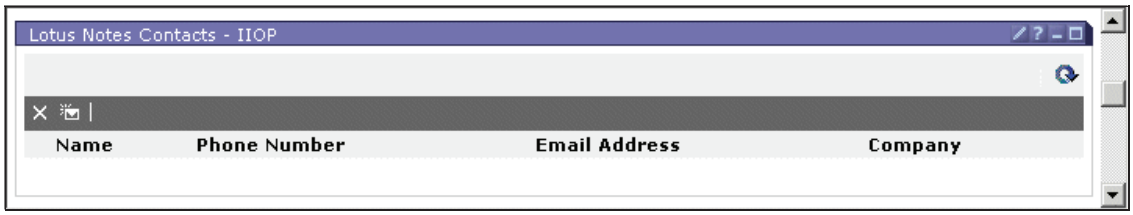


Figure 22-56 Online PIM portlet

4. You will be presented with a window containing fields to be filled in for the new contact. Enter at least the first name, last name, and office e-mail address.

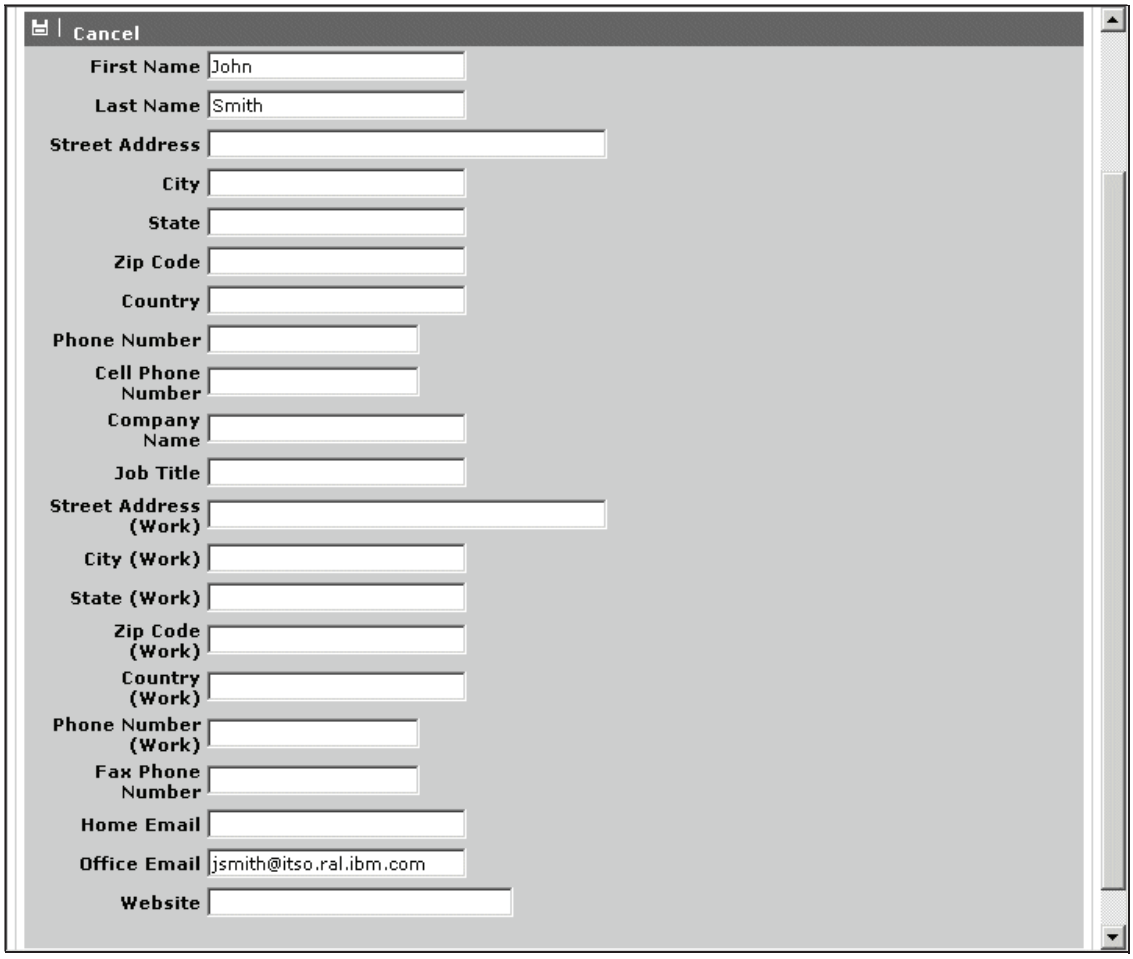


Figure 22-57 Enter new contact details

5. Click the **Save** button. It is the first button on the left of the grey horizontal bar above the First Name field.
6. You should now see your new contact in the Lotus Notes Contacts portlet.



Figure 22-58 Lotus Notes PIM portlet showing new contact

Create Note

1. Log in as the ESS user.
2. Select **WEA Home -> Notes**.
3. On the Lotus Notes Notebook portlet, click the **New Notebook Entry** button. It is the button with an envelope inside, located on the right of the button with an X.
4. You will be presented with a window to enter the subject and content of the new note. Enter something in the Subject field and something in the content box.

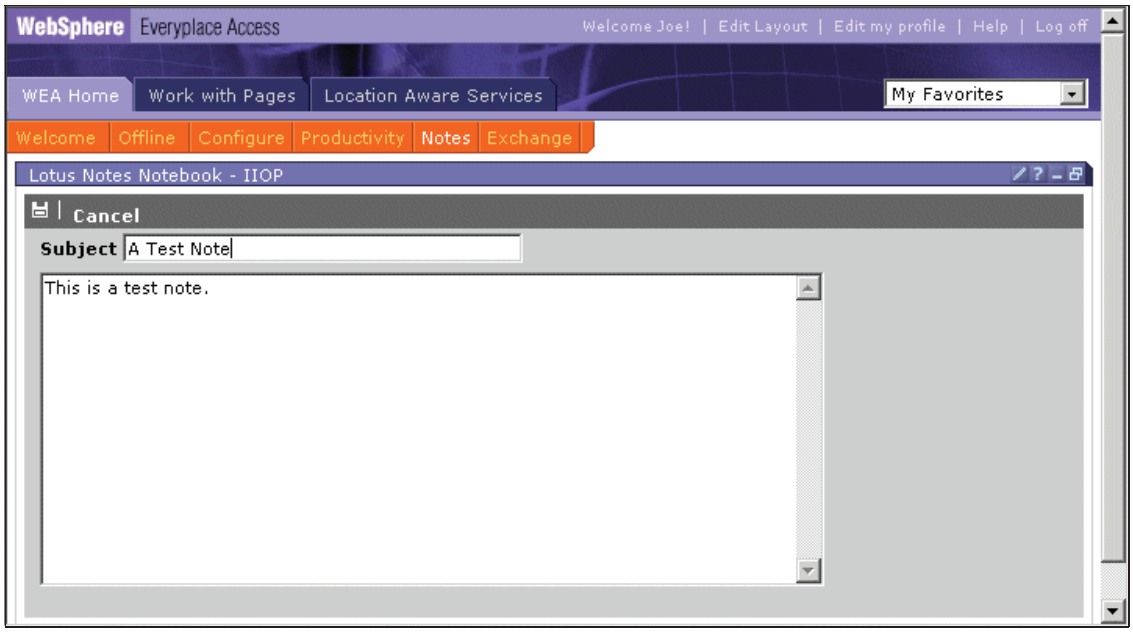


Figure 22-59 Enter subject and content of note

5. Click the **Save** button.
6. You should now see your new note in the Lotus Notes Notebook portlet.

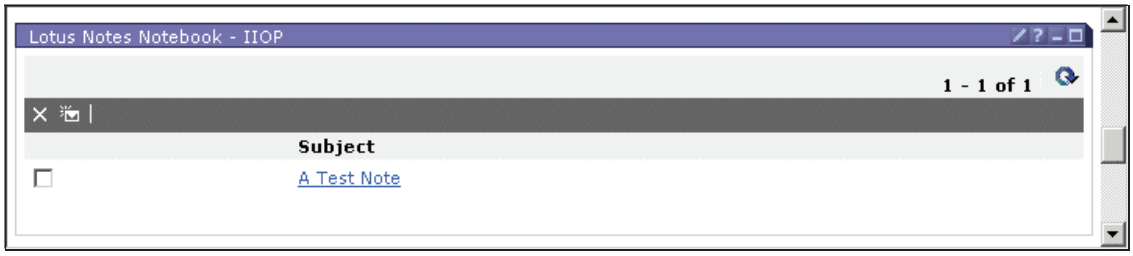


Figure 22-60 Lotus Notes Notebook portlet

Create task

1. Log in as the ESS user.
2. Select **WEA Home -> Notes**.
3. On the Lotus Notes To Do List portlet, click the **New Todo** button. It is the button with an envelope inside, located on the right of the button with an X.
4. You will be presented with a window to enter the subject, due date and importance the new task. Enter something in the Subject field. Set the due date to a day one week from now. Set the importance to Low.

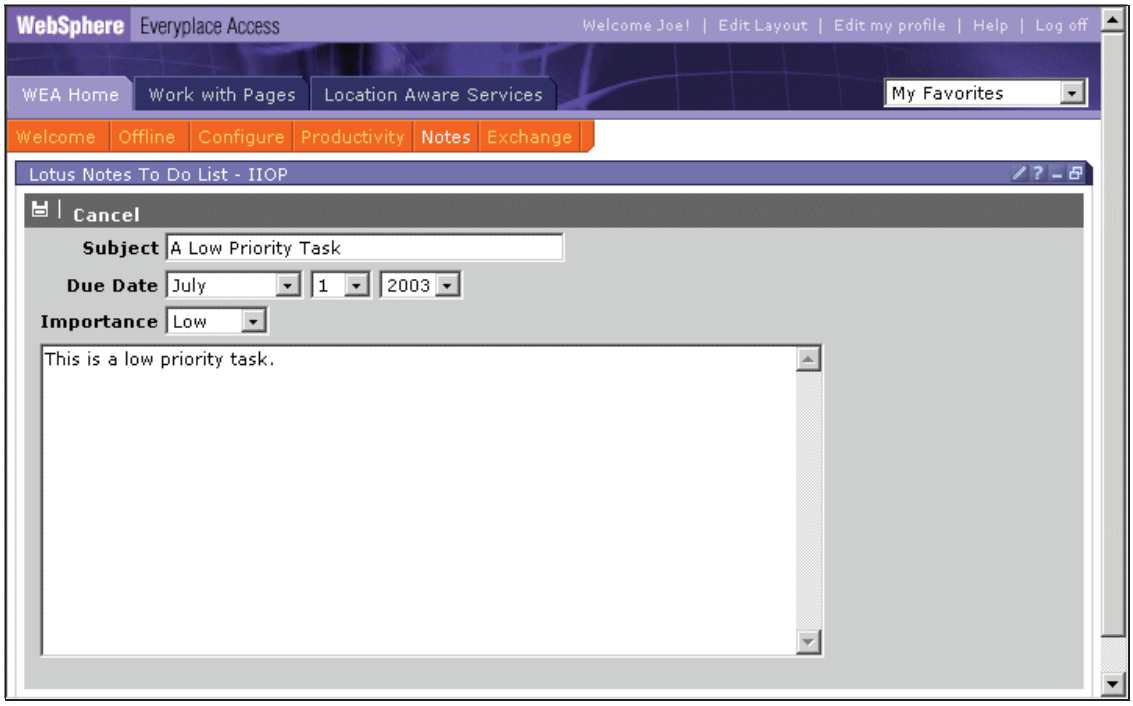


Figure 22-61 Enter subject and content of task

5. Click the **Save** button.
6. Repeat the process, but this time create a high priority task.

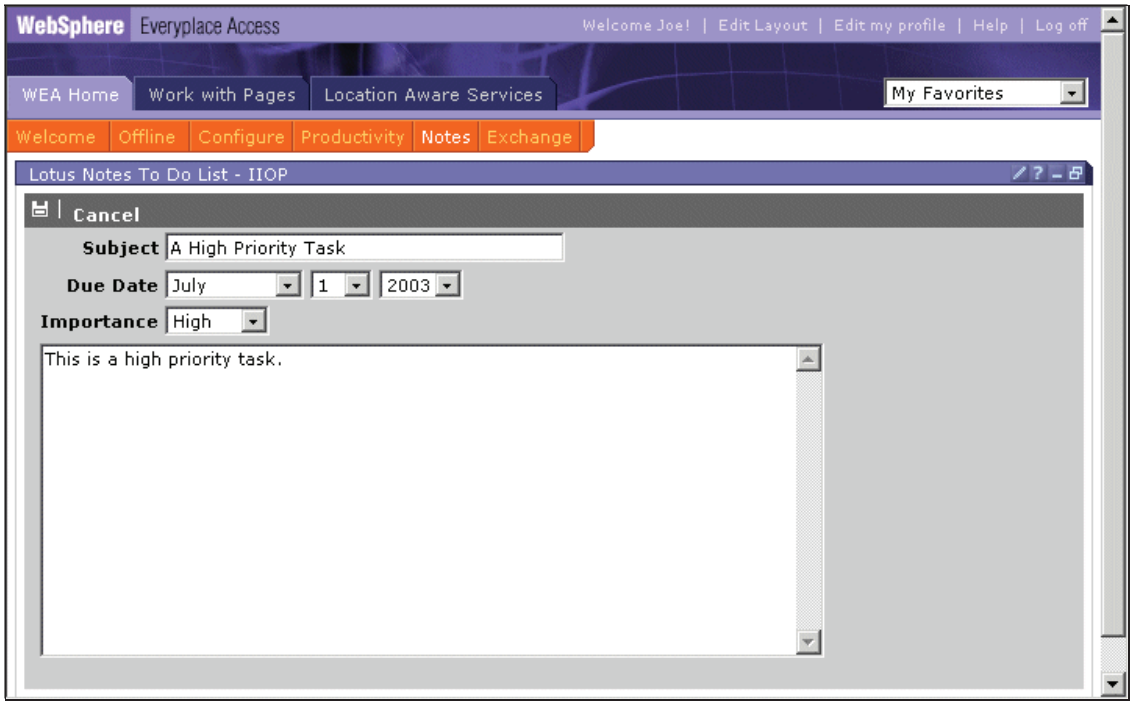


Figure 22-62 Enter subject and content of task

7. You should now see your new task in the Lotus Notes Notebook portlet.

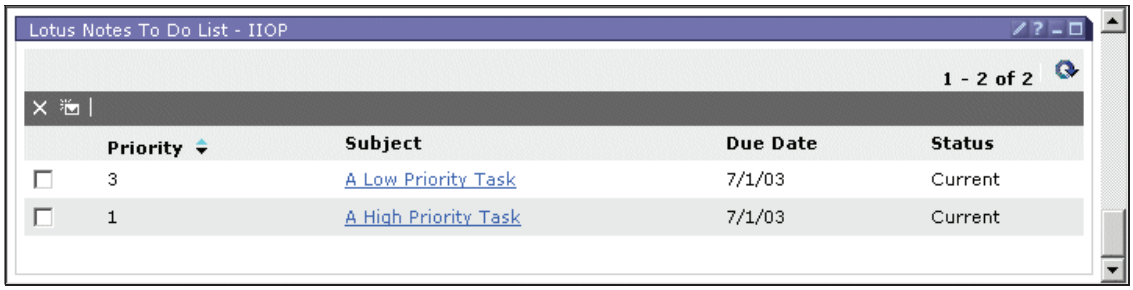


Figure 22-63 Lotus Notes To Do List portlet

Create e-mail

1. Start up a Lotus Notes client and log in as another user. We will be using this user to send e-mail to our ESS user.
2. Go to the Mail in-box and click **New Memo**.

3. Click the **To** entry field. This will move the cursor to this field. Click the **Address** button on the toolbar to open up the address book.

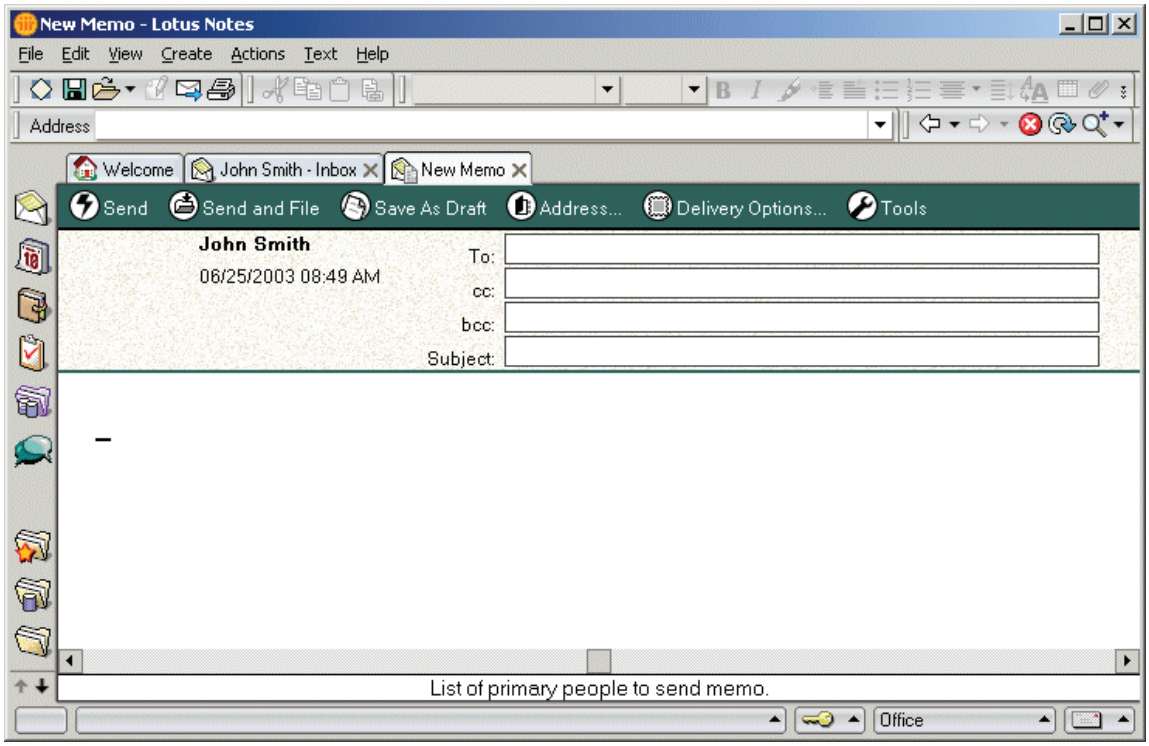


Figure 22-64 New Memo

4. Using the **Choose address book** drop-down list, select the domain address book.

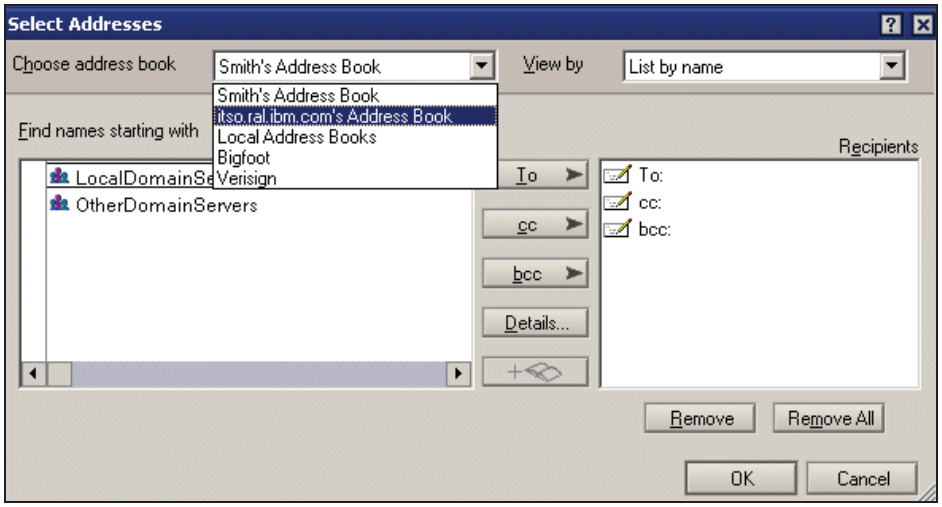


Figure 22-65 Select the Domain Address book

5. Select the ESS user and click the **To** button.

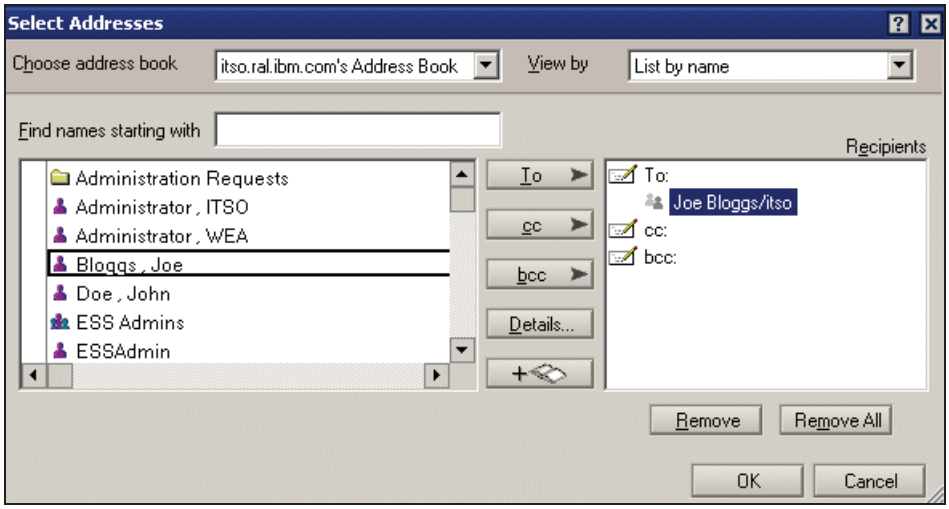


Figure 22-66 Select ESS User

6. Click **OK** to return to the E-mail window.
7. In the Subject field, enter `E-mail with an attachment of less than 250 K with a .DOC extension.`

8. Put your cursor on the message body and then click the **Attach** button (the paper clip button on the toolbar) or select **File -> Attach**. Locate a Microsoft Word document of less than 250 KB and attach it.

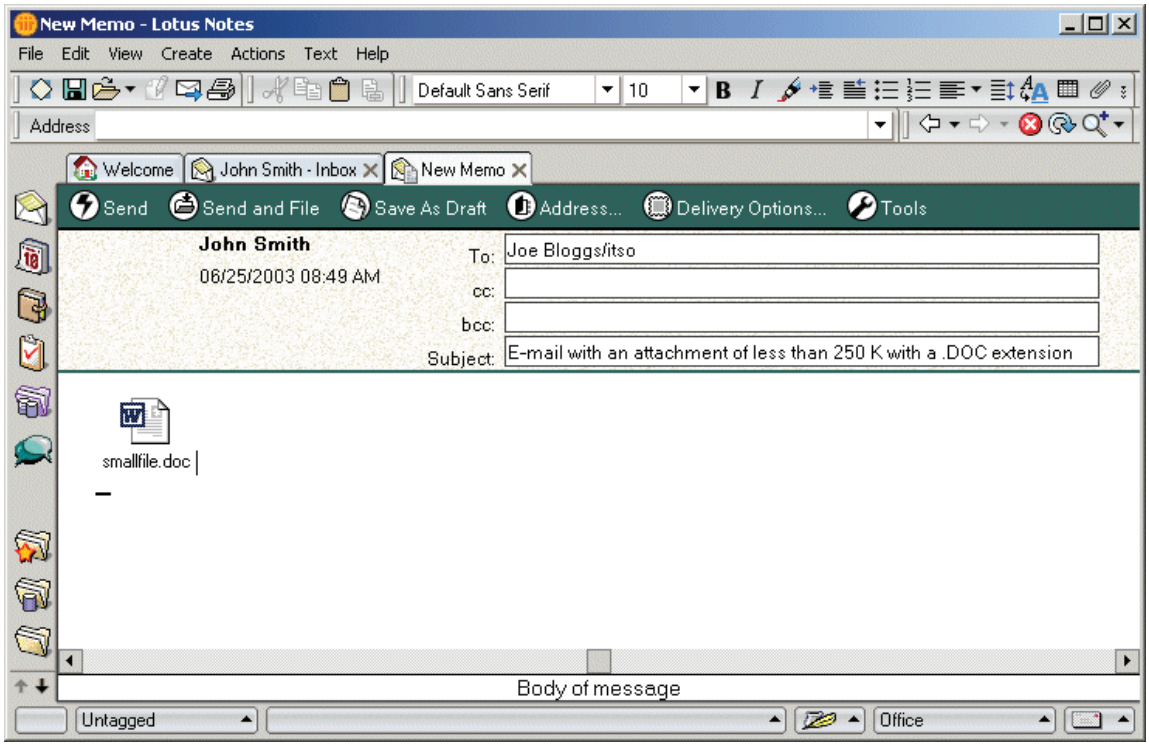


Figure 22-67 New Memo with attachment

9. Click **Send** to send the new e-mail.
10. Repeat the process, but this time:
    - Enter `E-mail with an attachment of less than 250 K with a .TXT extension` in the Subject field.
    - Attach a .txt document of less than 250 KB.
11. Finally, repeat the process again, but this time:
    - Enter `E-mail with an attachment of greater than 250 K with a .DOC extension` in the Subject field.
    - Attach a Microsoft Word document of greater than 250 KB.

Create Appointments

1. Start up a Lotus Notes client and log in as another user. We will be using this user to send meeting invitations to our ESS user.
2. Open up the user's calendar.
3. Click **Schedule a Meeting**.
4. Enter the meeting details:

Subject	`A non recurring appointment`
Location	`Room 101`
Begins	`<Tomorrow's date> 12:00 PM`
Ends	`<Tomorrow's date> 1:00 PM`
Description	`A non recurring appointment to discuss WEA Synchronization`

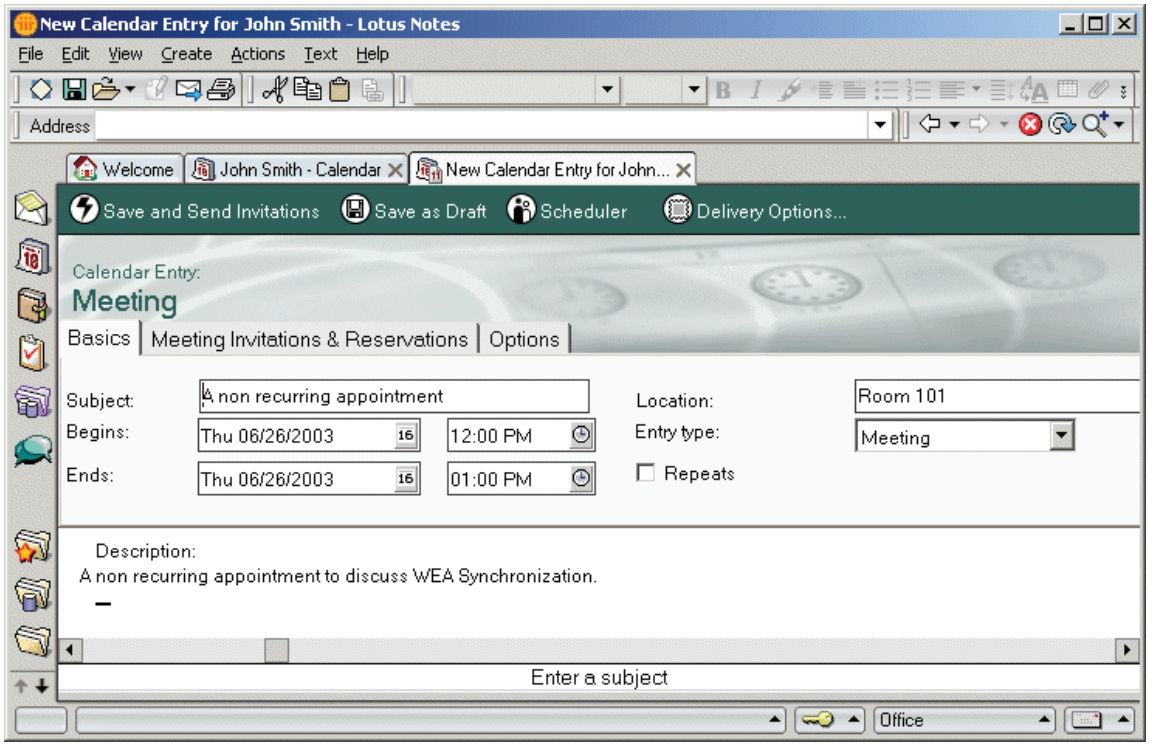


Figure 22-68 New Meeting - Basics

5. Click **Meeting Invitations & Reservations**.
6. In the Invite field, click the address book on the right.

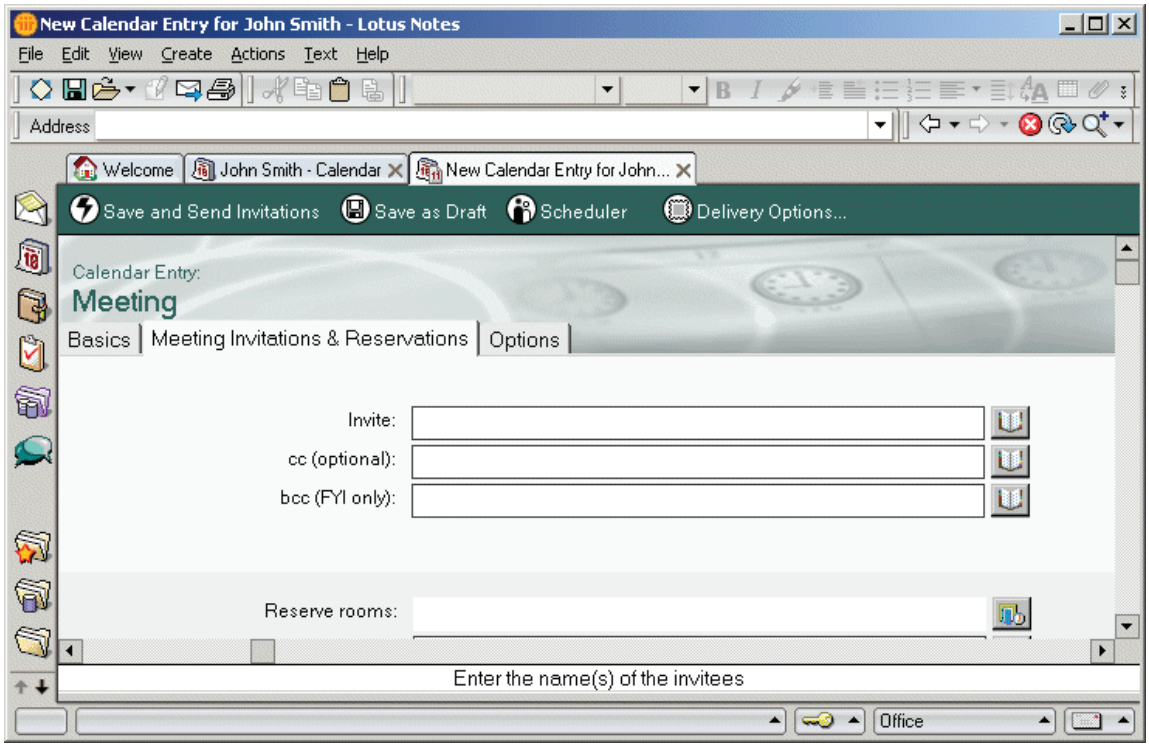


Figure 22-69 New Meeting - Invitations & Reservations

7. Add the ESS user to the To: list.

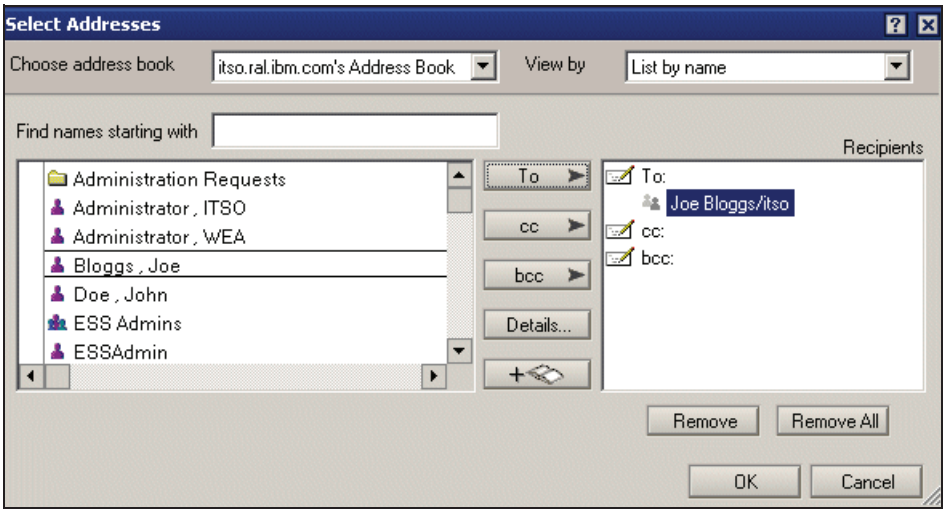


Figure 22-70 Add ESS user to To: list

8. Click **OK**.
9. Click **Save and Send Invitations**.

   Repeat the process, but this time when you enter the meeting details, enter:

   Subject `A recurring appointment`
   Location `Room 101`
   Begins `<Tomorrow's date> 3:00 PM`
   Ends `<Tomorrow's date> 4:00 PM`
   Description `A recurring appointment to discuss WEA Synchronization`

10. Click the **Repeats** check box.
11. Select **Weekly** and select the appropriate day of the week.

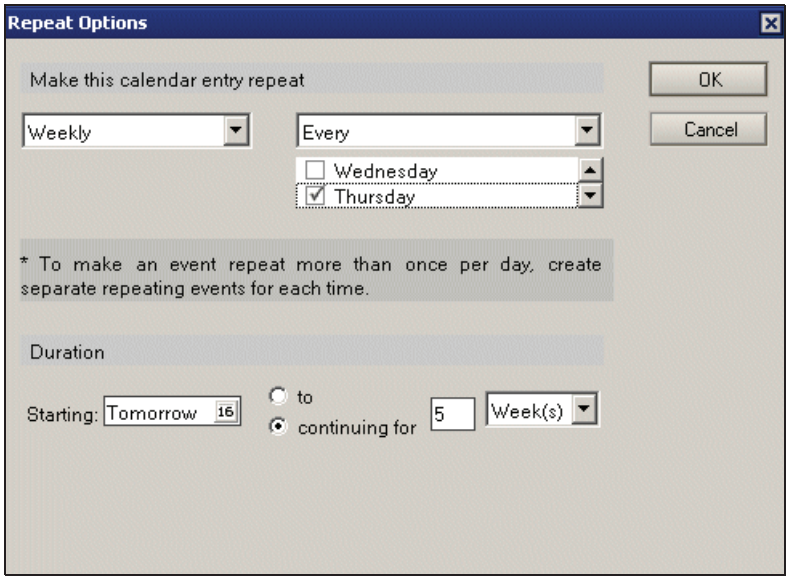


Figure 22-71 Repeat Options

12. Click **OK**.
13. Click the **Meeting Invitations & Reservations** tab and invite the ESS user as shown in steps 5 to 8.
14. Click **Save and Send Invitations**.
15. You have now created two appointments: one that is non-recurring and one that is recurring. They should both appear in the calendar of the user you used to create the appointments.

22.4.3 Performing synchronization

In this section, we demonstrate how synchronization can be initiated from both the Pocket PC and the Palm device. It is assumed that at this point you have already configured your client devices to point to the Everyplace Synchronization Server. Refer to 22.3.6, “WebSphere Everyplace Access Client configuration” on page 754 for more information on configuring your client device.

Synchronization with the Pocket PC

1. Launch the Everyplace Client by selecting **Start -> Everyplace Client**.
2. If prompted, enter the Portal login user name and password.
3. Select the home page by selecting **Home** on the drop-down list.

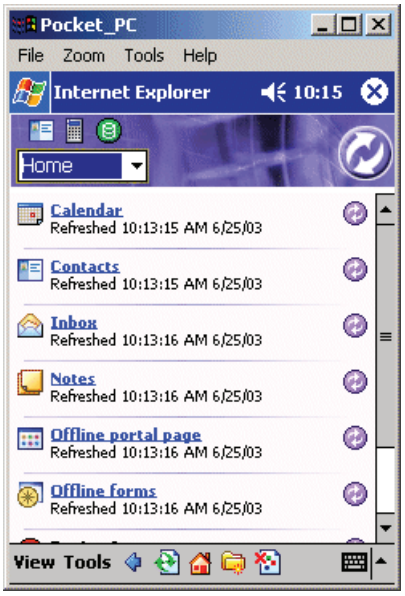


Figure 22-72 WebSphere Everyplace Access Client home page

4. You will be presented with a window with a list of components that can be synchronized. On the right side of each component is a Sync button for that component. Since we have several PIM components to synchronize, we will not invoke synchronization on individual components. Instead, we will invoke synchronization for all components. Click the blue **Sync** button at the top-right corner. This will initiate the synchronization for all components. The blue Sync button will switch to the synchronization progress indicator. Because this is the initial synchronization, it can take some time.

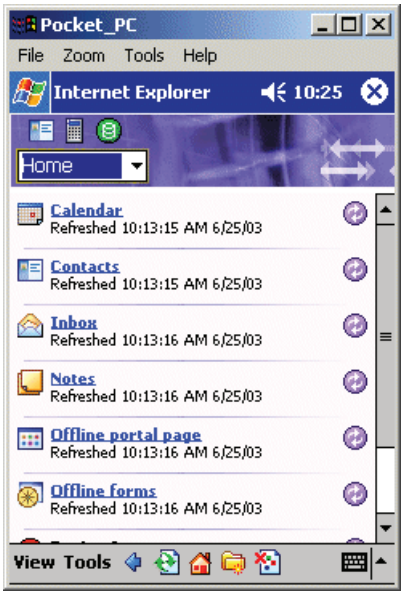


Figure 22-73 Pocket PC during synchronization

5. Once synchronized, it should look something like Figure 22-74.

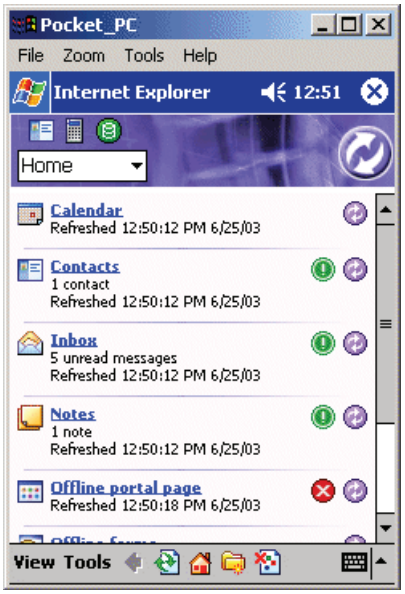


Figure 22-74 Everyplace home page after synchronization

Each component that contains new data should have an exclamation mark next to it to indicate new data.

6. We will start by examining the Lotus Notes Contacts data. Click **Contacts**.
7. You will see that the contact you added is now there.



Figure 22-75 Contacts list

8. Click the new contact and it will open up the details for that contact.



Figure 22-76 Contact details for new contact

9. Now go back to the Everyplace Client home page and click **Inbox**.
10. If you see a nearly empty window with no items, it is because we are looking at the in-box of the ActiveSync mailbox and not the SyncML mailbox.



Figure 22-77 ActiveSync in-box

11. Click **Inbox** on the grey horizontal bar. You will see a drop-down list showing both the ActiveSync and SyncML mailbox.

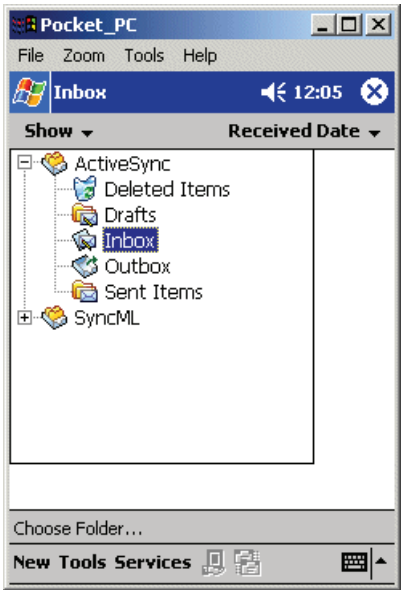


Figure 22-78 ActiveSync and SyncML mailbox

12. Expand the SyncML mailbox and click its in-box. You will then be presented with the in-box of the SyncML mailbox. You should see five e-mail notes. Two are meeting invitations from John Smith. The other three are e-mail notes sent from John Smith. Each of those three e-mail notes was sent with an attachment.

    However, if you look at the icon next to each e-mail item, you will notice that only one of them has a paper clip. In other words, only one of the three e-mail notes carries an attachment. This means that the default device profile has done its job. We created the default device profile (Mobile ITSO Users) to only accept attachments that have an extension of .DOC and are less than 250 KB. Of the three e-mail notes from John Smith, only one met the criteria.

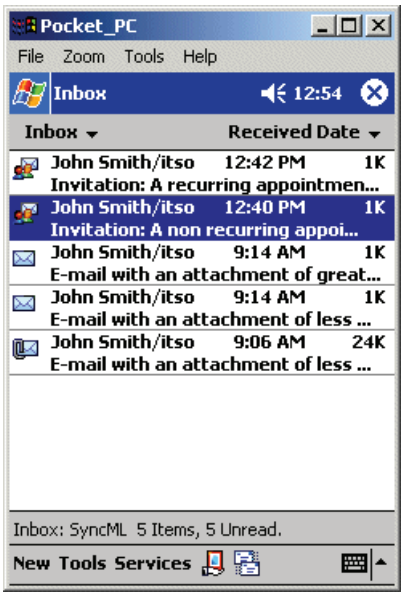


Figure 22-79 In-box of Joe Bloggs

13. Click the e-mail with the nonrecurring meeting invitation. You should see details of the meeting.
14. We want to accept this meeting. Click the button on the right of the Edit button. It contains a question mark, an X and a check.

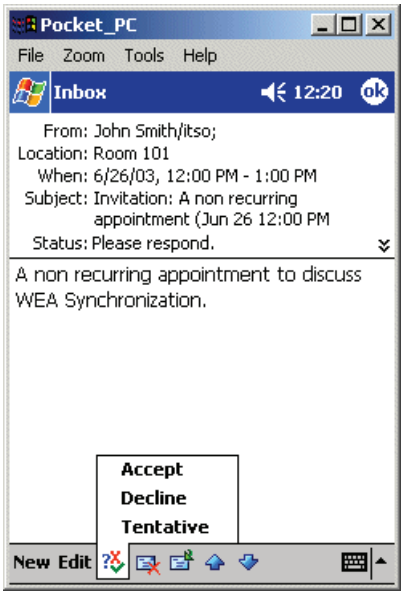


Figure 22-80 Meeting details

15. Click **Accept**. Click **OK** to send the response. You will be taken back to the in-box.
16. Click the e-mail with the subject `E-mail with an attachment of greater than 250 K with .DOC extension`. You will see a message in the e-mail body indicating that the attachment was removed because it wasn't less than 250 KB, as defined in our default device profile.

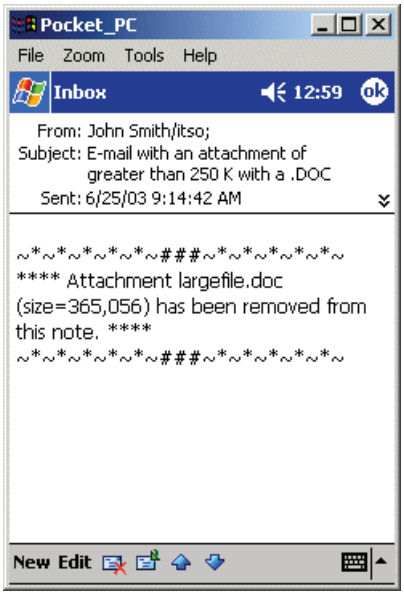


Figure 22-81 E-mail with attachment greater than 250 KB

17. Go back to the in-box and open the e-mail with the subject `E-mail with an attachment of less than 250 K with a .TXT extension`. Again you will see that the attachment was removed because it did not have a .doc extension.
18. Go back to the in-box and open the e-mail with the subject `E-mail with an attachment of less than 250 K with a .DOC extension`. This time, you will notice that the attachment was not removed because it was less than 250 KB and had a .doc extension.

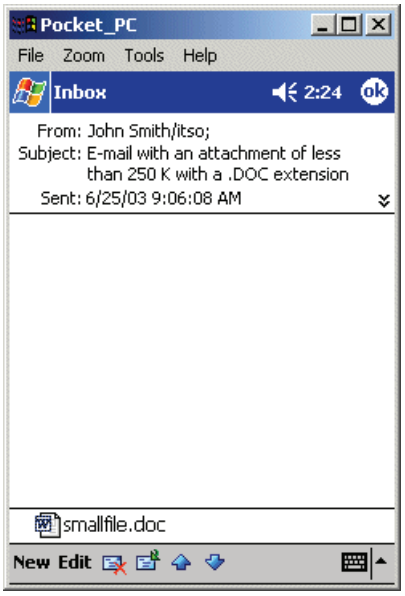


Figure 22-82 E-mail with attachment

19. Open the attachment (at the bottom of the window) by clicking it. It will be opened into Pocket Word.

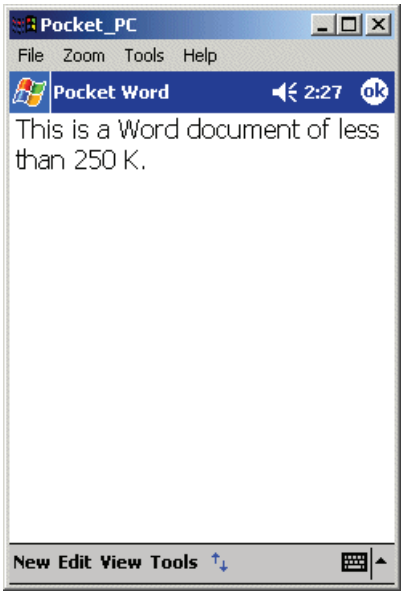


Figure 22-83 Attachment opened in Pocket Word

20. Return to the in-box. Return to the Everyplace Client home page. The page will refresh itself. You will notice that under Inbox, there is one unsent message. This is the acceptance of the nonrecurring meeting invitation. The message will be sent during the next synchronization.

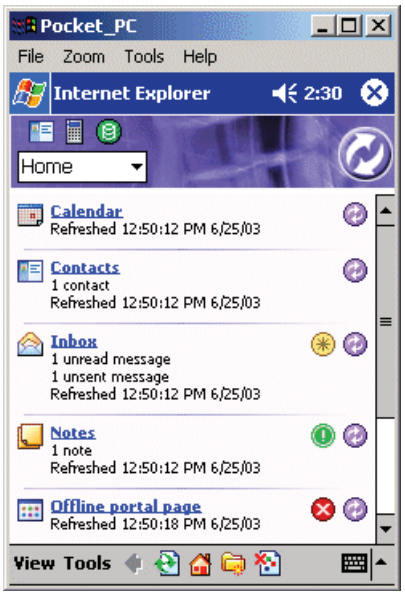


Figure 22-84 Everyplace Client home page

21. Click **Notes**. It may bring up a new note. If it does, click the **OK** button in the top-right corner. You should see the note you created using the PIM portlets.

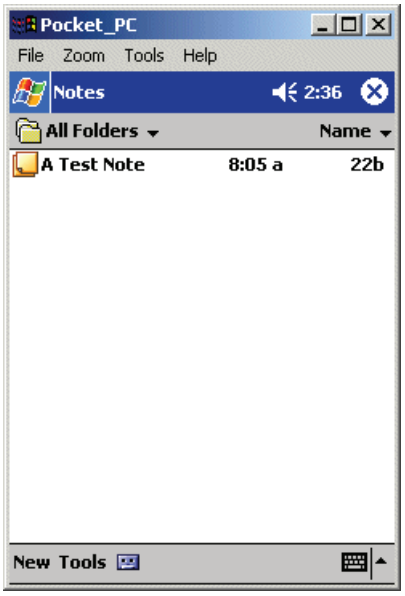


Figure 22-85 Notes Folder

22. Click the note you created. You should see the message in the note.

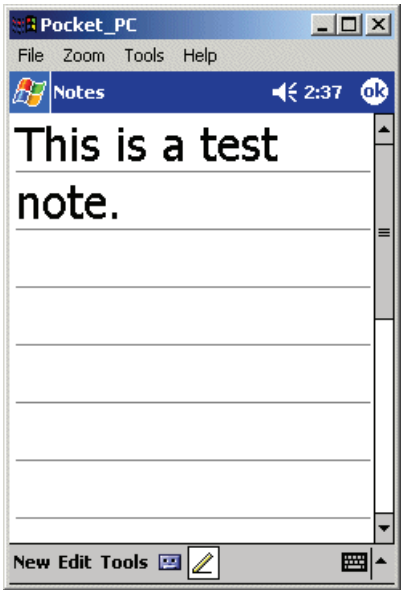


Figure 22-86 Content of note

23. Return to the Everyplace Client home page. Click **Tasks**.
24. You will see that there is only one task - the High Priority Task. The low-priority task we created using the PIM portlet did not meet the criteria defined in the default device profile.

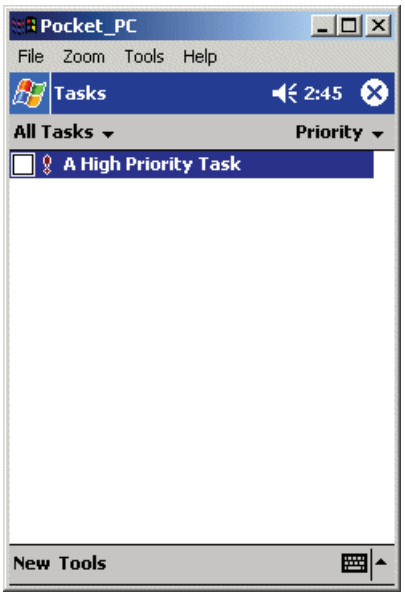


Figure 22-87 Task list

25. Click the task to open the content of the task.

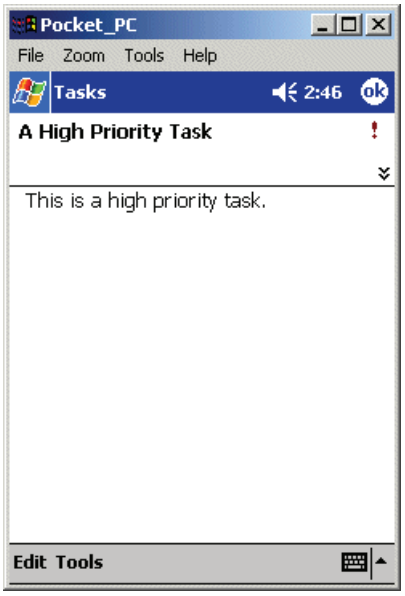


Figure 22-88 Content of task

26. We will now create an e-mail note and send it to another user. Click **Inbox**.
27. Click **New** in the bottom-left corner.
28. Enter the following details:

    To `jsmith@itso.ral.ibm.com` or e-mail address of intended recipient
    Subject `E-mail sent from Pocket PC`

    Enter something in the message body.

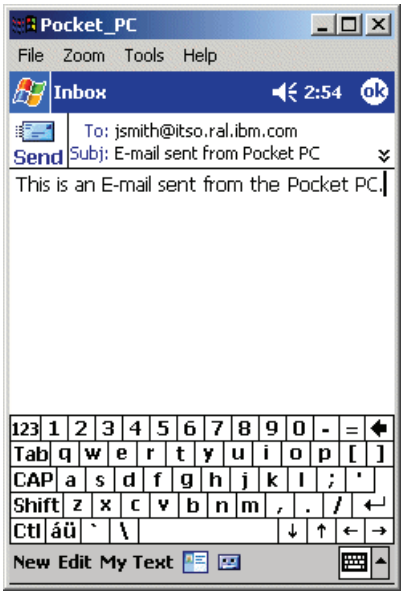


Figure 22-89 New message

29. Click **Send**.

30. Return to the Everyplace Client home page and click the blue **Sync** button to synchronize all components. During synchronization, the two unsent messages (the meeting acceptance and the e-mail created on the Pocket PC) will be sent to the intended recipients.

31. Acceptance of the nonrecurring appointment will result in a calendar entry being added in our calendar. Click **Calendar**. Because the appointment was scheduled for the next day, you will not see any calendar entries for today. Click the day that the appointment was created for. You should see an entry for the nonrecurring appointment.



Figure 22-90 Nonrecurring appointment

32. Click the appointment to view details of the appointment.

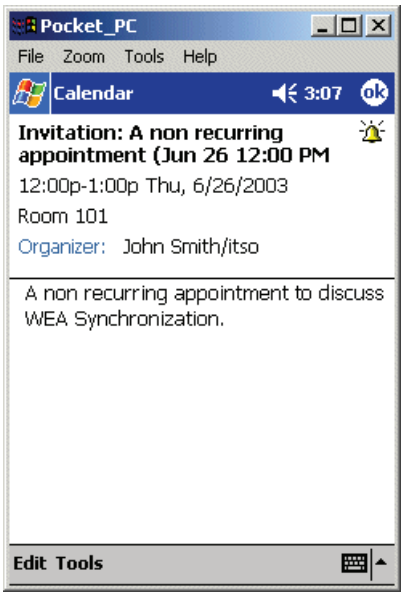


Figure 22-91 Appointment details

33. This completes our demonstration of synchronization with the Pocket PC.

Synchronization with the Palm OS 5.2 device

1. Click the **SyncML** icon.
2. Click the **Synchronize** button in the middle of the window.



Figure 22-92 SyncML Client

3. After synchronization is complete, you should see something like Figure 22-93 on page 796.

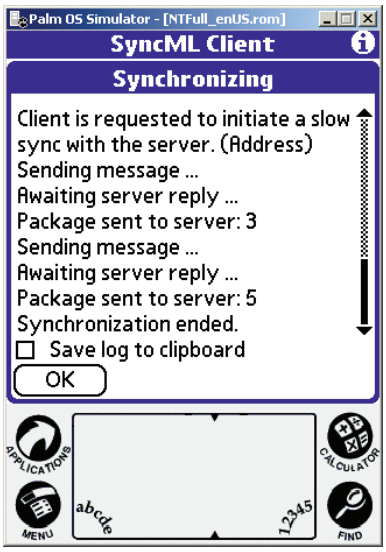


Figure 22-93 Synchronization Complete

4. Click **OK**.
5. Click **Applications** to return to the Applications page.
6. We will start by examining the address book. Click **Address**. You should see the contact that you added using the PIM portlet.



Figure 22-94 Address book

7. Open the contact by clicking it. You should now be able to see the details of the new contact. Click **Done**.



Figure 22-95 Contact details for new Contact

8. Click **Applications** to return to the Applications page.
9. Click **VersaMail**. You will be taken to the in-box of your ESS user. If you are in the in-box of another VersaMail account, change accounts by selecting **Accounts -> <The ESS User account>**.

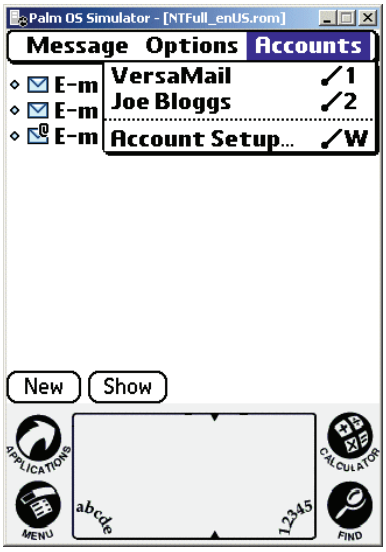


Figure 22-96 Changing VersaMail Account

10. In the in-box, you will notice that there are three e-mail messages instead of five. This is because there is no support for meeting invitations on Palm devices. The two invitations (recurring and nonrecurring) sent by John Smith are therefore not in this in-box. Each of the three e-mail notes in the in-box were sent with an attachment. However, if you look at the icon next to each e-mail item, you will notice that only one of them has a paper clip. In other words, only one of the three e-mail notes carries an attachment. This means that the default device profile has done its job. We created the default device profile (Mobile ITSO Users) to accept only those attachments that have an extension of .doc and are less than 250 KB. Of the three e-mail notes from John Smith, only one of them met these criteria.



Figure 22-97 In-box of Joe Bloggs

11. Select the e-mail with the subject `E-mail with an attachment of greater than 250 K with .DOC extension`. You will see a message in the e-mail body indicating that the attachment was removed because it was greater than 250 KB as defined in our default device profile.

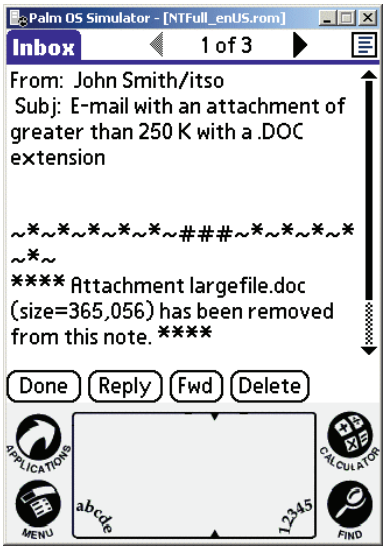


Figure 22-98 E-mail with attachment greater than 250 K

12. Go back to the in-box and open the e-mail with the subject `E-mail with an attachment of less than 250 K with a .TXT extension`. Again you will see that the attachment was removed because it did not have a .doc extension.
13. Go back to the in-box and open the e-mail with the subject `E-mail with an attachment of less than 250 K with a .DOC extension`. This time, you will notice that the attachment was not removed because it met the criteria of being less than 250 KB and having a .doc extension.

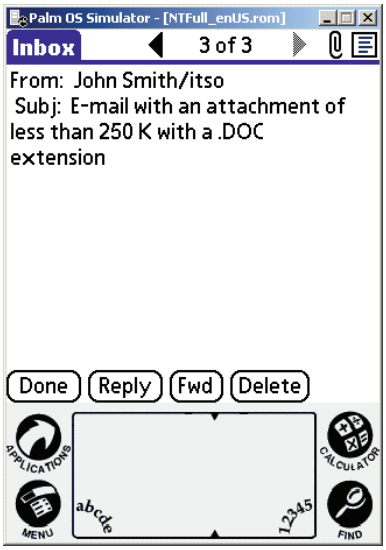


Figure 22-99 E-mail with attachment

14. Open the attachment by selecting the paper clip in the top-right corner. Click **View** on the message that comes up. You will see the attachment in a Microsoft Word Viewer.

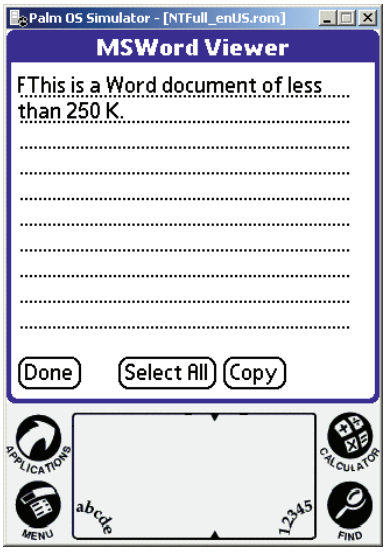


Figure 22-100 Attachment viewed using Microsoft Word Viewer

15. Click **Applications** to return to the Applications page.

16. Click **Memo Pad**. You will see the note you created using the PIM portlets. The content of the note will display in the subject. This might be a limitation in the Memo Pad application as it doesn't appear to support the concept of a subject in a note.



Figure 22-101 Memo list

17. Click the note to view its entire content.



Figure 22-102 Content of note

18. Click **Done**.
19. Click **Applications** to return to the Applications page.

20. Click **To Do List**. You will see that there is only one task - the High Priority Task. The low-priority task we created using the PIM portlet did not meet the criteria defined in the default device profile.



Figure 22-103 Task list

21. Click the Page icon on the right of the task to view its content.



Figure 22-104 Content of task

22. Click **Done**.
23. Click **Applications** to return to the Applications page.
24. We will now create an e-mail note and send it to another user. Click **VersaMail**.
25. Click **New**.
26. Enter the following details:

    To `jsmith@itso.ral.ibm.com` or e-mail address of intended recipient

    Subj `E-mail sent from Palm OS 5.2 device`

    Enter something in the message body.

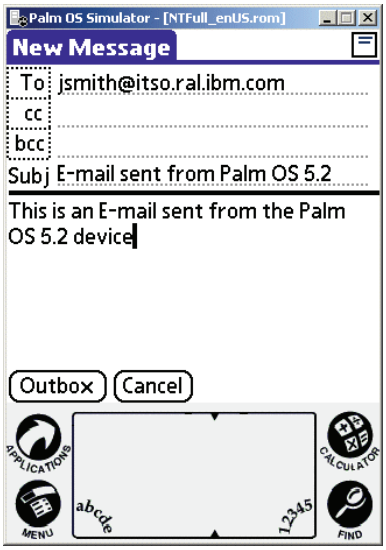


Figure 22-105 New message

27. Click **Outbox**.
28. Click **Applications** to return to the Applications page.
29. Click **SyncML** and initiate a synchronization. During synchronization, the unsent message will be sent to the intended recipient.
30. This completes our demonstration of synchronization with the Pocket PC.

22.4.4 Verifying Synchronization

We now need to verify the synchronization by checking that e-mail notes sent from the client devices reached their intended recipients. In our examples, all

e-mail notes sent from the client device were intended for John Smith. We will therefore check John Smith's in-box.

1. Using a Lotus Notes client, log in as user John Smith or the intended recipient of the e-mail notes sent from the client device.
2. Go to the in-box.

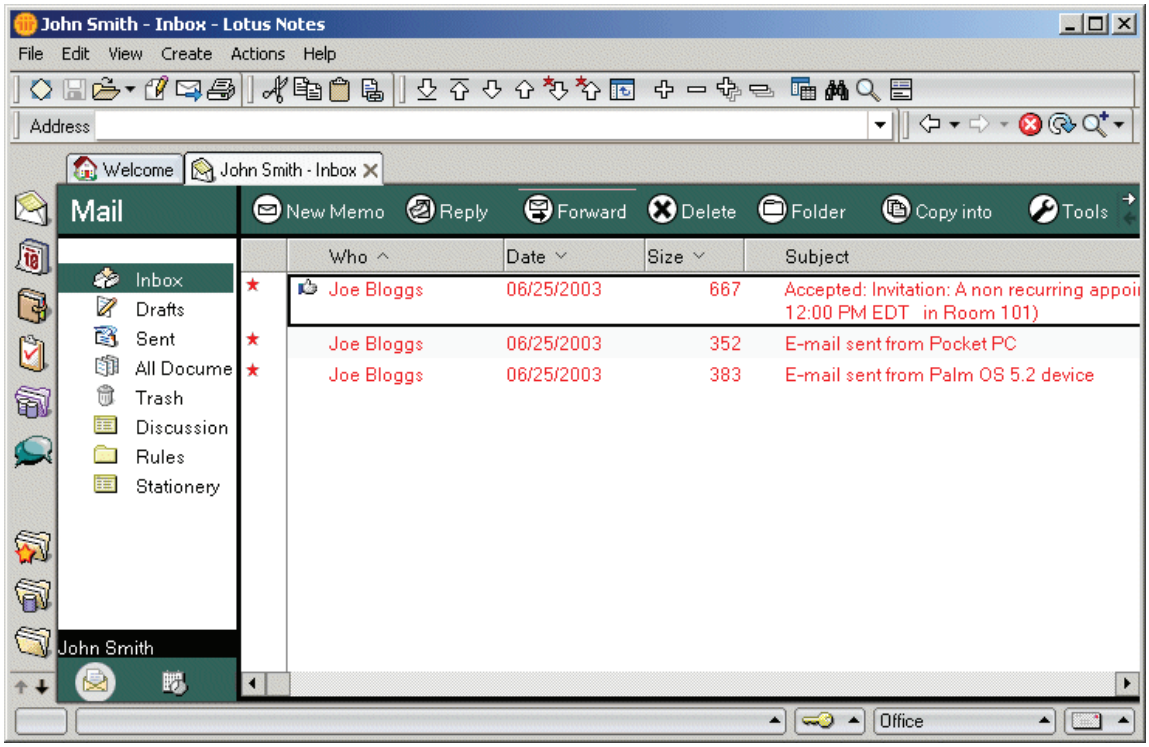


Figure 22-106 In-box of John Smith

3. You should see three e-mail notes from Joe Bloggs:
   - An invitation acceptance sent from the Pocket PC.
   - An e-mail sent from the Pocket PC.
   - An e-mail sent from the Palm device.

22.5 Hints and tips

22.5.1 Limitations

Before setting up the Everyplace Synchronization Server for synchronization with Lotus Domino, you should understand and be aware of the current limitations that exist on WebSphere Everyplace Access 4.3. Select **WebSphere Everyplace Access InfoCenter -> Everyplace Synchronization Server -> Planning -> Understanding current limitations** for a comprehensive listing of these limitations.

22.5.2 Troubleshooting

There are many things that can go wrong when configuring ESS with Lotus Domino. A good source of information on what to look for is the Troubleshooting section in the InfoCenter, which can be accessed by selecting **WebSphere Everyplace Access InfoCenter -> Everyplace Synchronization Server -> Troubleshooting**.

22.5.3 Log files

If you have carefully followed the instructions and still cannot get synchronization to work on your WebSphere Everyplace Access server, you may want to start looking at the log files. Log files are a great source of information on what happened or what went wrong during runtime. For descriptions of log files and how to use them, select **WebSphere Everyplace Access InfoCenter -> Everyplace Synchronization Server -> Administering -> Using logs**.

22.5.4 Tuning

If you find that synchronization is taking a long time, there are a few things that can be done to improve the performance of the Synchronization Server. Select **WebSphere Everyplace Access InfoCenter -> Everyplace Synchronization Server -> Tuning** for more information on optimizing your server.

22.5.5 Order does matter

It needs to be emphasized that the order in which you install your software components does matter. It is paramount that the local Lotus Notes client is installed and *configured* with the Adapter Admin user before installing the Everyplace Synchronization Server. Failure to do this may result in the Synchronization Server failing to connect to the Lotus Domino Server.

22.6 Resources

The WebSphere Everyplace Access InfoCenter provides fast, centralized access to information to help you install and configure WebSphere Everyplace Access. It comes bundled with WebSphere Everyplace Access and after installation, it can be accessed by selecting **Start -> Programs -> IBM WebSphere -> Everyplace Access -> InfoCenter**.

Alternatively, it can be accessed at:

`http://publib.boulder.ibm.com/pvc/wea/430/en/InfoCenter/index.html`

The WebSphere Everyplace Access support site is a great place to find information about:

- Bug fixes and updates to software.
- Updates to documentation.
- Hints and tips.
- Solutions to specific problems.

The support site can be accessed at:

`http://www-3.ibm.com/software/pervasive/products/support/ws_everyplace_access.shtml`

23

PIM and e-mail synchronization with Exchange 2000

In this chapter we describe the usage of the synchronization capabilities of the Everyplace Synchronization Server (ESS) for accessing Microsoft Exchange 2000 PIM and e-mail information.

Although it is not covered in this chapter, WebSphere Everyplace Access also provides PIM and e-mail synchronization with Microsoft Exchange V5.5 servers. For more information about ESS, refer to the InfoCenter.

© Copyright IBM Corp. 2003. All rights reserved.

23.1 Overview

Everyplace Synchronization Server (ESS), part of the WebSphere Everyplace Access product, is the component that allows you to synchronize PIM and e-mail information between Lotus Domino and Microsoft Exchange servers and mobile devices. Actually, it is a server running on top of WebSphere Application Server and uses the WebSphere Portal services for configuration and administrative task.

Figure 23-1 illustrates the architecture of the several components of the Everyplace Synchronization Server (ESS).

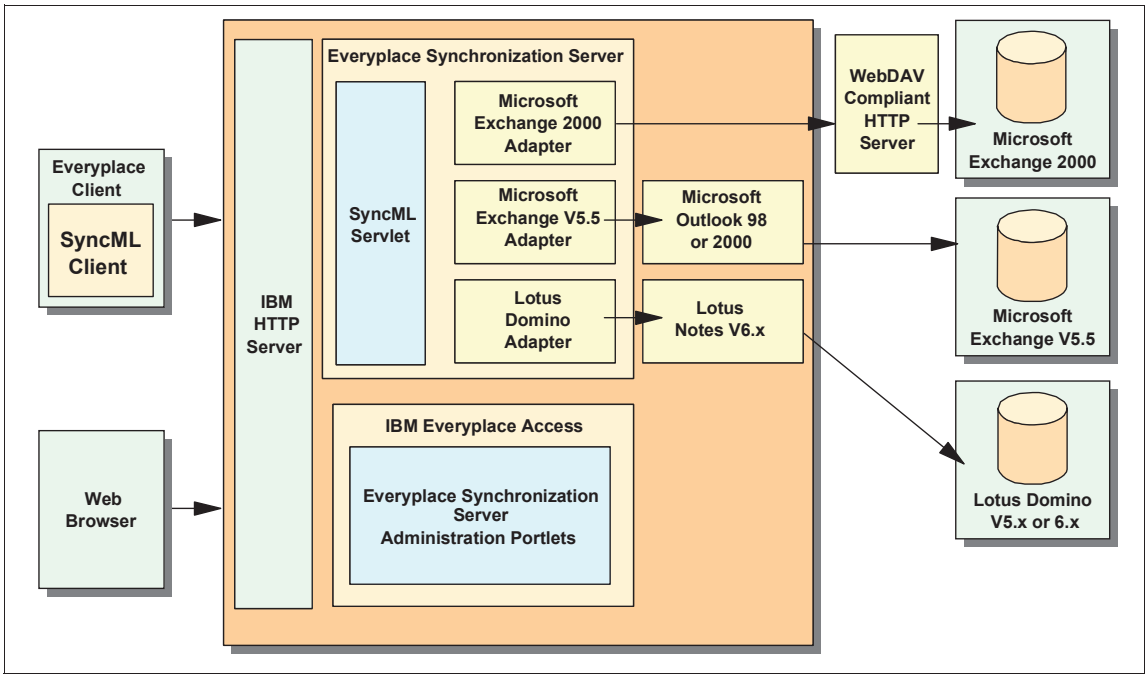


Figure 23-1 ESS architecture diagram

For synchronization to mobile devices, ESS uses the TrueSync technology. TrueSync is an embedded synchronization engine on which IBM has built a flexible adapter framework to provide synchronization with various servers, regardless of the platform or format of the data.

ESS provides several adapters to communicate with the supported servers:

- Lotus Domino Adapter
- Microsoft Exchange 2000 Adapter
- Microsoft Exchange V5.5 Adapter

TrueSync uses the SyncML 1.1.1 protocol to transport the data between ESS and mobile devices. SyncML is an open standard sponsored by the key players in the pervasive field: Ericsson, IBM, Lotus, Matsushita, Motorola, Nokia, Openwave, Starfish Software, and Symbian, and is supported by leading wireless companies. For more information about the SyncML protocol, visit the SyncML site at http://www.openmobilealliance.org/syncml/

On the client side, the Everyplace Client provides access to the device PIM and e-mail native databases and performs the client-side SyncML synchronization.

23.2 WP portlets for ESS

ESS comes with several administration portlets that you can use to manage the different components of the Everyplace Synchronization Server. The following portlets are deployed on WebSphere Portal when you install ESS:

- Synchronization Lotus Domino Adapter Administration
- Synchronization Microsoft Exchange 2000 Adapter Administration
- Synchronization Microsoft Exchange V5.5 Adapter Administration
- Synchronization Server Configuration
- Synchronization Server Management
- Synchronization Settings for Administrators
- Synchronization Settings for Users
- Synchronization User Preference Administration

The number of adapter administrator portlets depends on he adapters that you had installed when you installed ESS.

23.3 Synchronizing e-mail and PIM information

The section describes a sample scenario for synchronizing e-mail and PIM data on Palm and Pocket PC devices.

23.3.1 Exchange 2000 Server configuration

There are no special configuration tasks to perform to use the Exchange 2000 Server with ESS. Just make sure that you have the correct version of the Microsoft Exchange Server. In this WebSphere Everyplace Access release, the Exchange 2000 Service Pack 2 is supported.

In addition, ESS requires a Web-based Distributed Authoring and Versioning (WebDAV) compliant HTTP server (such as the Microsoft Internet Information Services) installed with the back-end server.

To verify that Microsoft IIS is installed and running, perform the following steps on the Exchange 2000 Server:

1. Select **Start -> Programs -> Administrative Tools -> Computer Management**
2. Expand **Services and Applications**
3. Expand **Internet Information Services**
4. Verify that the state for the Web server is Running.

Note: When you first create an Exchange 2000 user, not all of the default folders are created. Before end users can synchronize all of their folders with an Exchange 2000 Server, their Exchange accounts must be activated. To activate the account, the user must log in to his or her mailbox in Microsoft Outlook.

For more information about the Exchange 2000 Server, visit the Microsoft Exchange site at `http://www.microsoft.com/exchange/`.

23.3.2 Everyplace Synchronization Server configuration

When you install the WebSphere Everyplace Access server, and select the Exchange 2000 Adapter for installation, you have to enter the Microsoft Exchange 2000 Server name. Therefore, you already have a server configured for synchronization. To check the installed server, do the following:

1. Log in as an administrator user (`wpsadmin` for example) on your WebSphere Everyplace Access server.
2. Select **Everyplace Synchronization -> Manage Servers**. Note that the ESS service by default doesn't start when the machine does. If the ESS service is down, you will see the ESS server status as Not available, as shown in Figure 23-2.

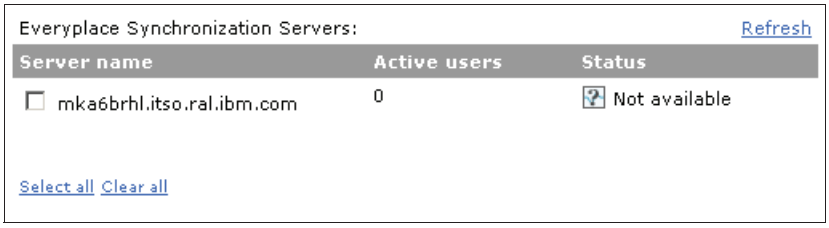


Figure 23-2 ESS not available

To start the ESS service, click **Start -> IBM Everyplace Synchronization Server-> Start Local Server** on your ESS server. Wait for the server to start.

Now click the **Refresh** link to refresh the server status, as shown in Figure 23-3.

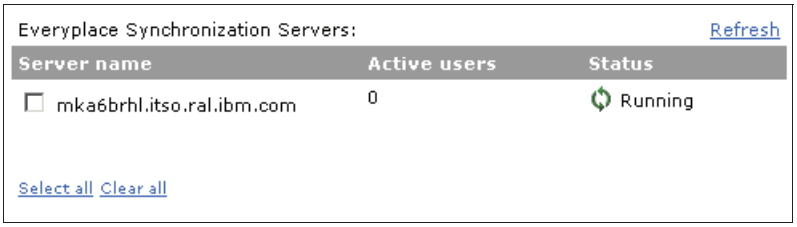


Figure 23-3 ESS available

3. On the same page, you see the active Exchange servers. The status of the server (or servers) should be Running, as shown in Figure 23-4.

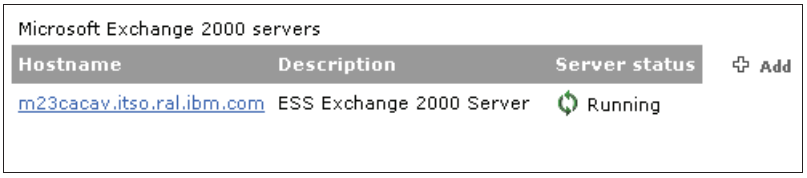


Figure 23-4 Exchange servers running

If your server status is Not available, contact your Exchange administrator to check for problems in the Exchange server.

4. If you wish, you can add more Exchange servers to your ESS installation. On the Exchange server list, shown in Figure 23-4, click **Add.** In the Add new Microsoft Exchange Server view, enter the server host name and a description of your server, as shown in Figure 23-5. Click **OK** to save the changes.

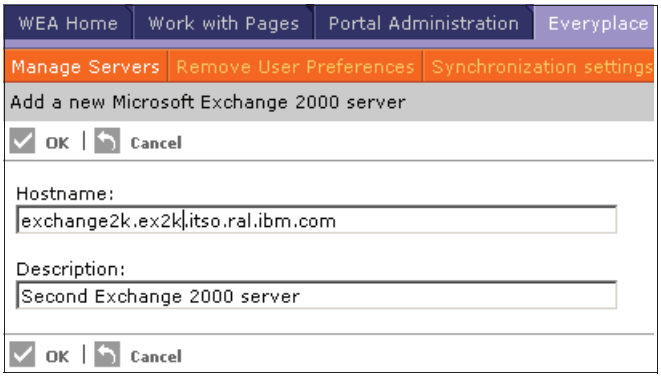


Figure 23-5 Adding a new Exchange server

23.3.3 Synchronization users and groups

By default, ESS defines two synchronization groups that you can use to manage the synchronization user rights:

- essusers
- essadmins

You can assign users to these groups directly, or assign groups to these groups. You can also create your own groups and set up their access rights to the synchronization portlets, using the WebSphere Portal Server Access Control List.

The essusers group only has access to the Synchronization Settings for Users portlet. The essadmins group has manage access to all the Administration portlets for ESS.

Then, to enable PIM and e-mail synchronization for a user, you could add him to to the essusers group, following these steps:

1. Log in the WebSphere Everyplace Access server as an administrator user (`wpsadmin` for example).
2. Select **Portal Administration -> Users and Groups -> Manage Groups**.
3. In the Name field, enter `essusers` as a search pattern and click **Get groups**, as shown in Figure 23-6.

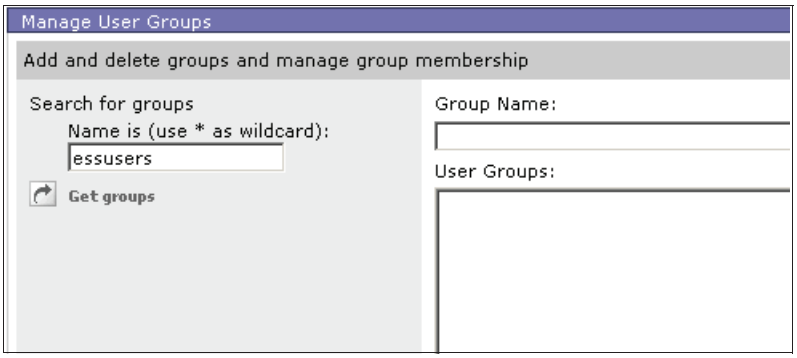


Figure 23-6 Looking for the essusers group

4. Select the **essgroups** in the User groups box as shown in Figure 23-7 on page 815. Click **Membership.**

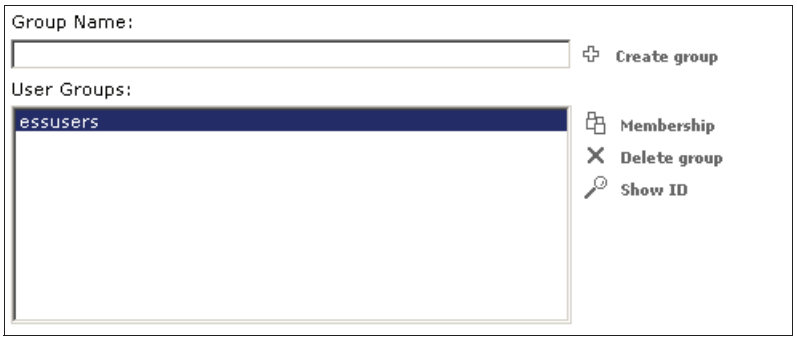


Figure 23-7 Membership for essusers group

5. You see all the users that are members of the essusers group (none in this case). To add a user (for example John Smith or John* for short), select **Add users to group** and in the Name field enter a pattern for the user or enter * to add all users as shown in Figure 23-8. Click **Go**.

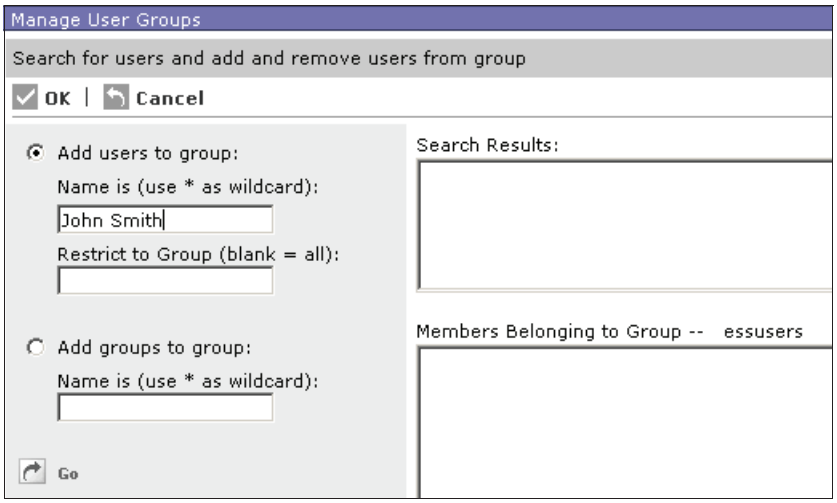


Figure 23-8 Looking for a user to add io the essusers group

6. In the Search results box you will see the users that were found. Select the users that you want to add, and click **Add to group**. You see the selected users in the Members belonging to Group box, as shown in Figure 23-9 on page 816. Click **OK**.

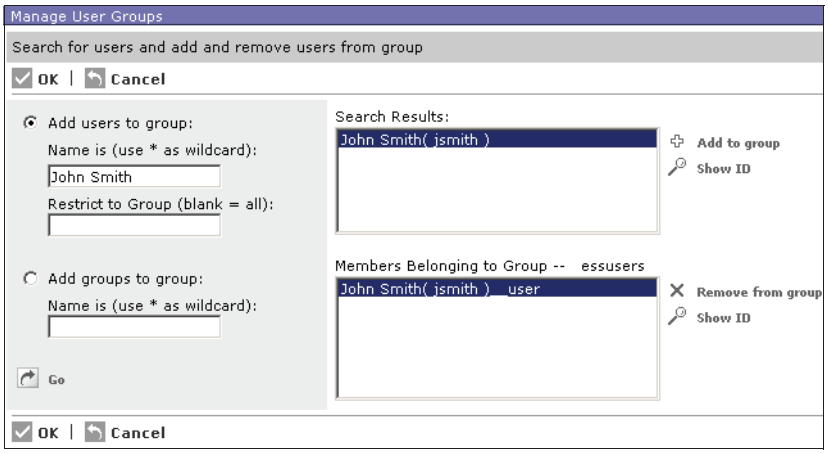


Figure 23-9 Adding a user to the essusers group

Now you have a user with access for configuring the synchronization settings.

23.3.4 Device profiles

A device profile must be created for each user who wants to perform PIM and e-mail synchronization. A device profile stores the synchronization settings for that user as well as the credentials to access the back-end Exchange server.

There are two types of device profiles:

- Administrator device profiles
- User device profiles

Administrator device profiles are created by an administrator user, and include settings that many users can share. On the other hand, a user device profile includes user-specific configuration, such as the Exchange server host name, user name, password and so on. However, a user device profile can be linked to an administrator device profile in order to provide default settings for that particular user. This default settings are defined by the administrator when he creates the profile. ESS comes with a default administrator device profile.

Another important device profile characteristic is the use of filters for the data to be synchronized. The Synchronization Server supports filtering options for five PIM data types: E-mail, Events/Calendar, Contacts, Tasks/To Dos, and Notes/Memos.

When a client synchronizes with the server, conflicts between server and client data can arise. You can define the behavior of the synchronization when such conflicts happen. Basically you have two choices:

- Server wins
- Client wins

This means that you can specify that the updated document on the device always overwrites the updated document on the server, or vice versa. If your PIM server is a Microsoft Exchange 2000 Server or Microsoft Exchange V5.5 Server, this setting causes one version of the document to overwrite the other document when there is a conflict between two versions of the same record.

To create a device profile for a user, follow these steps:

1. Make sure that the user belongs to the essusers group, or to a group that belongs to the essgroups group.
2. Log in into your WebSphere Everyplace Access server as the user (jsmith in this case). Select **WEA Home -> Configure**.
3. You will see the Synchronization settings portlet in the page, as shown in Figure 23-10. Note that there is no device profile available. Click **Create new device profile** to create one.



Figure 23-10 Creating a new device profile

4. Enter a name for the profile, as shown in Figure 23-11 on page 818. Click **Next**.

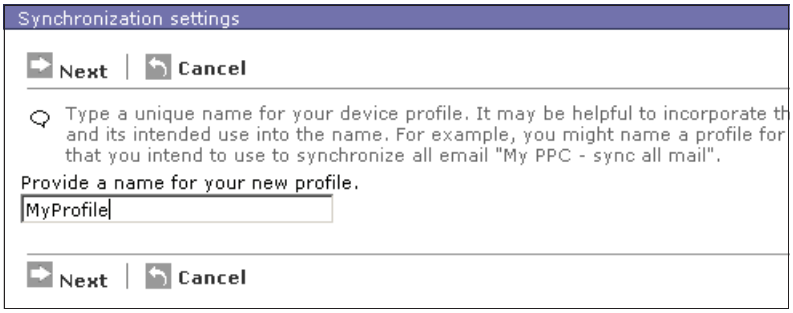


Figure 23-11 Naming the device profile

5. Select the Exchange server name where the user account resides. Enter the user name, password, mailbox name, and domain for the account, as shown in Figure 23-12. The mailbox name usually is the same as the user name and the domain is the Windows domain for the Exchange server (ITSO in this case). If you are not sure about these settings, consult your Exchange administrator. Click **Next**.

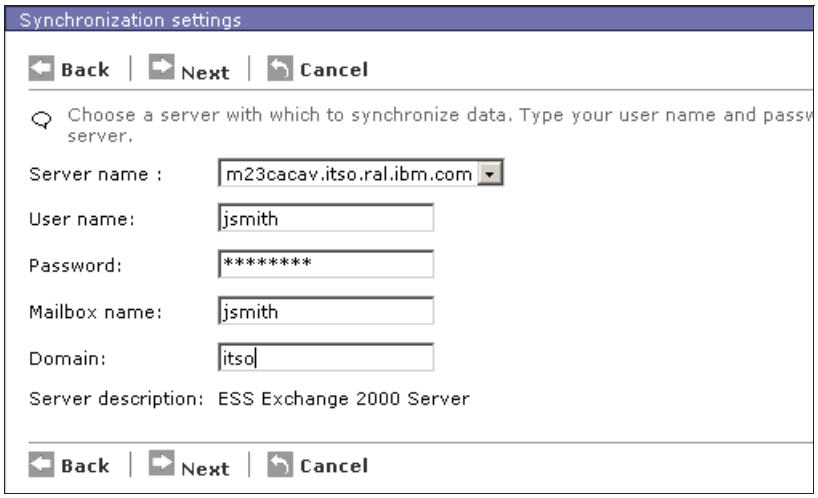


Figure 23-12 Settings for the Exchange account

6. In this step you can choose whether to link your profile to an administrator profile or to set all settings yourself. By default WebSphere Everyplace Access creates an administration profile. The administrator can modify it, or add a new one. In this case, choose the default profile and click **Next**.

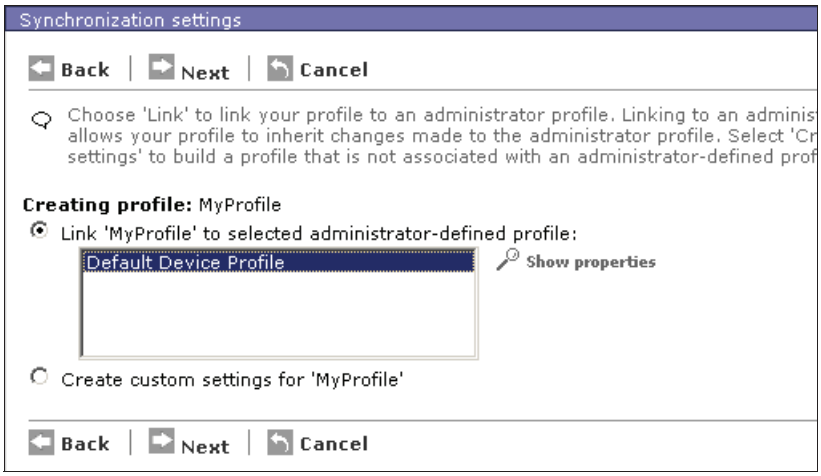


Figure 23-13 Linking the user device profile to an administrator profile

7. You can see Filters settings for the synchronization components. Each component settings is copied from the administrator profile (default in this case). Click **Next**.

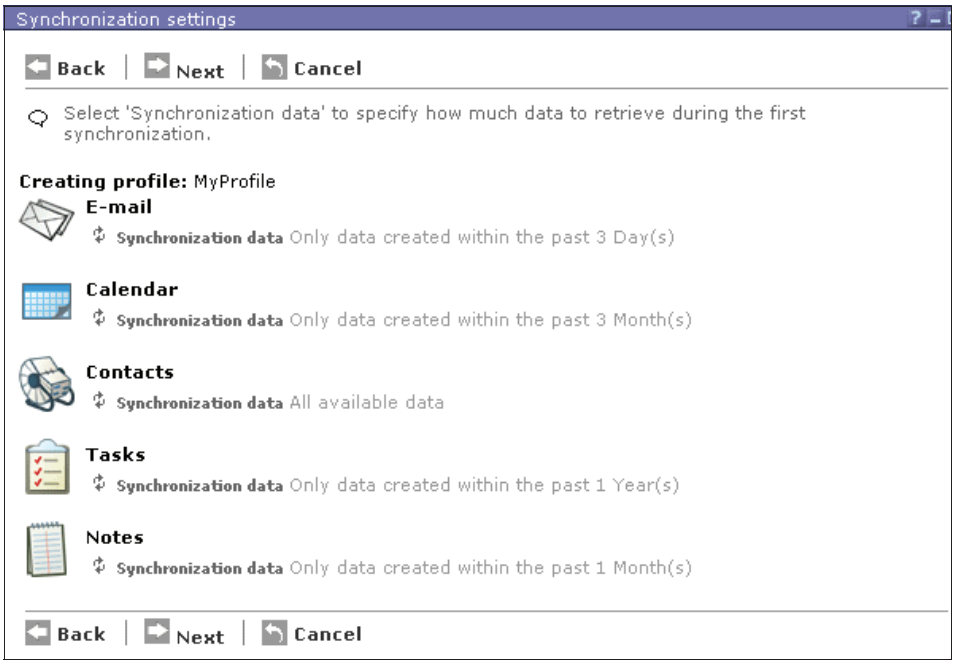


Figure 23-14 Synchronization settings for the profile

8. Choose a time zone and a locale for the profile. Click **Next**.



Figure 23-15 Choosing a time zone and locale for the profile

9. Now you can review the profile summary. If something is wrong, click **Back** to return to the appropriate page and correct it. If all the settings are correct, click **OK** to save the profile.

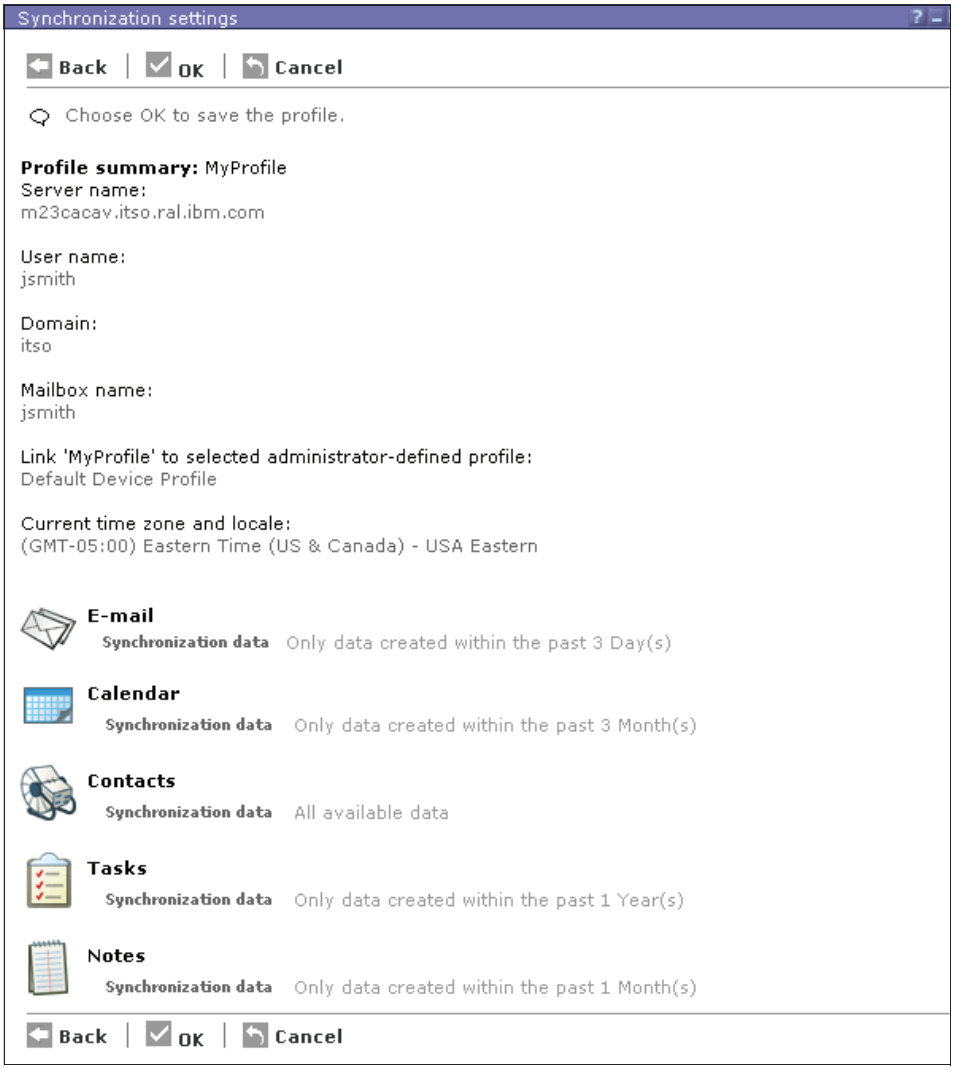


Figure 23-16 Summary for the profile

10. You have the device profile created, but not synchronized yet, as shown in Figure 23-17 on page 822. You can come back any time to this page to review or modify the device profile.

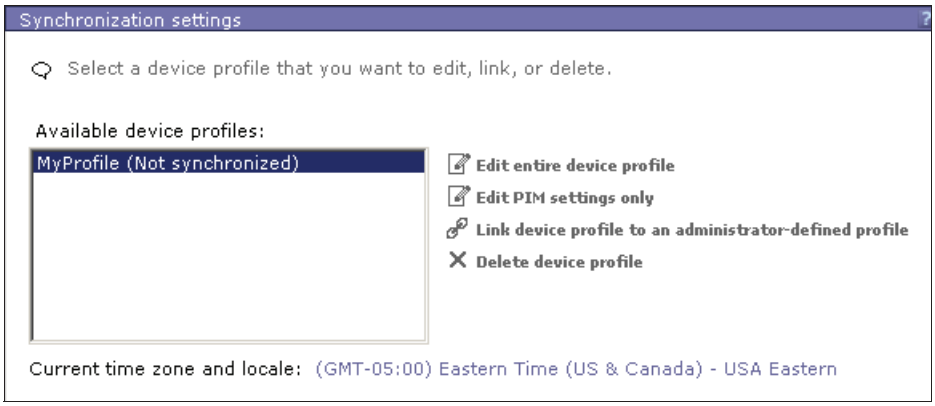


Figure 23-17 New device profile

Now that you have configured the server, you need to configure the client for synchronization.

23.3.5 Client configuration

You have to configure the WebSphere Everyplace Client with the correct settings to access to the synchronization server.

Not all the synchronization capabilities are available in all devices. Table 23-1 shows the features supported by the device operating system version.

Table 23-1 Everyplace Clients PIM and e-mail feature support per device

Operating System	PPC2002	Palm OS 3.5/4.0	Palm OS 4.1	Palm OS 5.0	Palm OS 5.2
SyncML Client	Yes	Yes	Yes	Yes	Yes
e-Mail sync with attachment	Yes[2]	Yes (no attach)	Yes[2] (VersaMail 2.0 Support)	Yes[2] (VersaMail 2.0 Support)	Yes[2] (VersaMail 2.0 Support)
PIM sync[1]	Yes	Yes	Yes	Yes	Yes
Invitation support[3]	Yes	No	No	No	No

Notes:

1. PIM services - calendar, contacts, to do list, memo.
2. Customer has to supply Attachment views.

3. Invitation support such as accept, reject, calendar update.

Palm client

On Palm devices, the client for PIM and e-mail synchronization is the SyncML Client application. You install this application when you install the Everyplace Client. Refer to Chapter 4, “Everyplace Client” on page 81 for information about how to install and configure the Everyplace Client on Palm devices.

For e-mail synchronization, the SyncML Client must be configured to work with a native mail client. There are two versions of a native mail program for Palm devices: Standard Mail and VersaMail 2.0. The SyncML Client can synchronize with both of them.

Configuring the SyncML Client with Standard Mail

To configure the SyncML Client with Palm devices that comes with Standard Mail, follow these steps:

1. Make sure that the SyncML Client is installed on your device.
2. Click the **SyncML** icon to open the SyncML client.
3. Select **Preferences -> PIM Server** to open the PIM Preferences window.
4. Enter the information for your server as shown in Figure 23-18. Click **OK**.

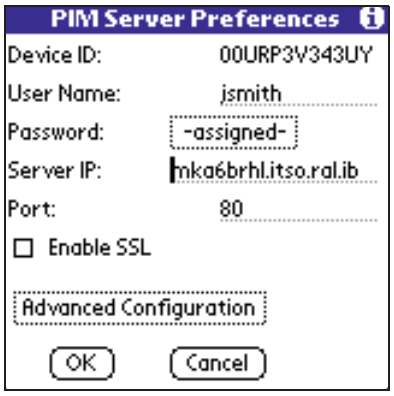


Figure 23-18 PIM Server Preferences

5. Select **Preferences -> PIM Subscriptions** o open the PIM Subscriptions window.
6. On the PIM subscriptions window, click **Mail Configuration**.
7. Select **Standard Mail** as shown in Figure 23-19 on page 824. Click **OK**.

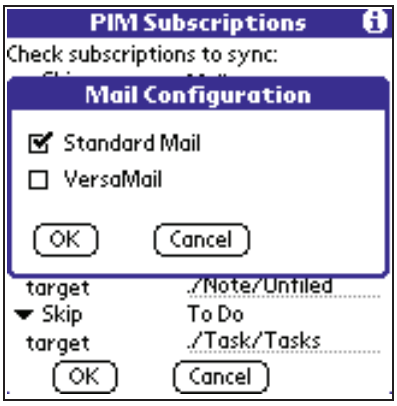


Figure 23-19 Selecting Standard Mail

8. Enter your name and your e-mail information, as shown in Figure 23-20. Click **OK**.



Figure 23-20 Entering the Mail Configuration information

9. Now we need to indicate to the SyncML client, the PIM information that we want to synchronize. For each PIM category you have three synchronization options, as shown in Figure 23-21 on page 825:
   - Skip: You do not want to synchronize this category
   - Sync: You want to synchronize this category
   - Refresh: You want to perform a complete refresh of the PIM category.



Figure 23-21 Synchronization options

In this case we select all the PIM categories for synchronization, as shown in Figure 23-22. Click **OK**.



Figure 23-22 Syncing all the PIM categories.

10. On the SyncML client main window, click **Synchronize** to start the synchronization session, as shown in Figure 23-23 on page 826.



Figure 23-23 SyncML Client main window

11. Wait while the SynML client synchronizes your PIM information with the ESS Server. At the end, you will see the `Synchronization ended` message, as shown in Figure 23-24. Click **OK**.

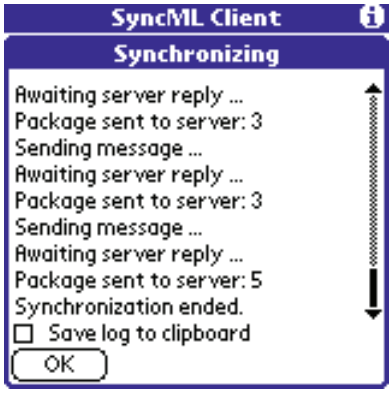


Figure 23-24 Synchronizing you PIM information

12. Close the SyncML client and start the Mail reader, by clicking the icon on the Palm Application window, as shown in Figure 23-25 on page 827.



Figure 23-25 Starting the Standard Mail reader

You will see the mail synchronized from the Exchange server, as shown in Figure 23-26.



Figure 23-26 Mail synchronized from the Exchange server

Configuring the SyncML Client with VersaMail 2.0

The configuration steps for the SyncML Client with VersaMail 2.0 are detailed in 22.3.6, “WebSphere Everyplace Access Client configuration” on page 754.

Pocket PC client

The configuration steps for the Pocket PC client are detailed in 22.3.6, “WebSphere Everyplace Access Client configuration” on page 754.

23.4 Hints and tips

If you don’t see your PIM data or e-mails, make sure that the Filter settings are configured correctly. For example, if the e-mail filter is configured to delivery to the device only the e-mail from the last 3 days, you won’t see any mail older than 3 days ago.

24

Instant messaging

This chapter discusses the messaging service offered in Everyplace Access, the components that are needed for enabling instant messaging service, and how to configure and use the Sametime client.

© Copyright IBM Corp. 2003. All rights reserved.

24.1 Overview

Instant messaging is the most popular means of communication in the connected world. Everyplace Access brings instant messaging to mobile devices. The Everyplace Access Client includes the Lotus Sametime 3.0 client, so that mobile and desktop users can chat with each other seamlessly.

With Sametime, users can set their current status, set alert preferences, manage users and groups, and so on. Figure 24-1 shows the Sametime client architecture.

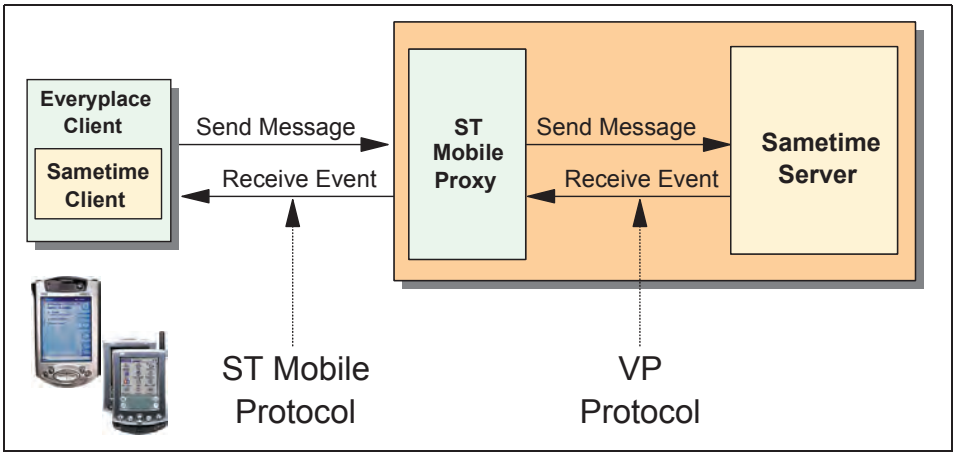


Figure 24-1 Sametime client architecture

The Sametime V3.0 client comes with the Sametime Mobile Protocol Stack, which talks to the Sametime Mobile Server using this protocol. The Sametime Mobile Server is an extension of the Sametime Server. Both the desktop user and the mobile user will connect to the same port on the Sametime Server. The Sametime Mobile (ST Mobile) proxy detects if the incoming request is from a mobile device or a normal desktop and establishes a session using specific protocols.

24.1.1 Sample scenario

This scenario shows how a Pocket PC device, a Palm device and the desktop user can use the Sametime client component provided with WebSphere Everyplace Access.

In this scenario we used the Lotus Domino Server 5.0.10 and Sametime 3.0 Server with Sametime Server Fix Pack 1. To enable mobile clients to connect, Lotus Sametime Mobile Extension 3.0 was also installed.

For instructions on how to set up Lotus Domino Server and Lotus Sametime Server, see Appendix H, “Lotus Domino Server and Sametime installation” on page 1257.

The Sametime Server has to be installed on the same machine as Lotus Domino Server.

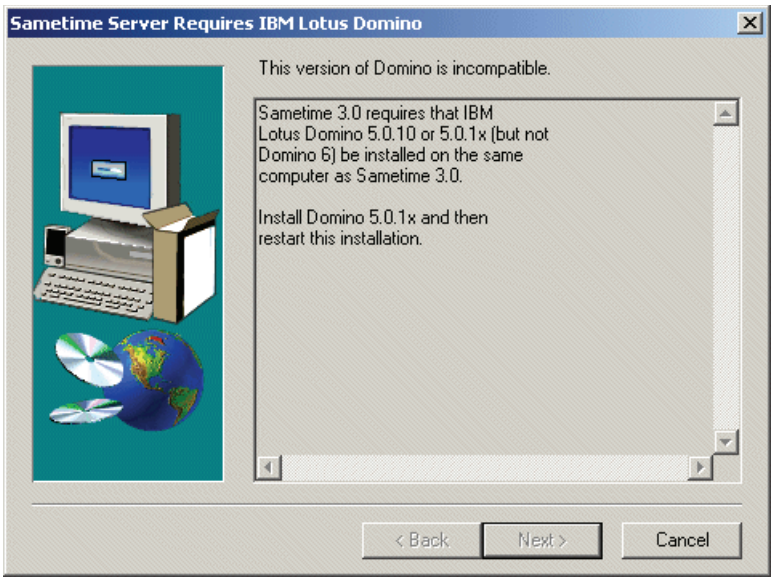


Figure 24-2 SametIme Server requires IBM Lotus Domino

In this scenario we will show Pocket PC, Palm and desktop users interacting with each other using Lotus Sametime.

Starting Sametime messaging client on Palm

1. To start the Sametime client on the Palm device, click **Applications** and click the **Sametime** icon.



Figure 24-3 Starting Sametime

2. This will bring up the login screen. Click the **Connectivity** tab.Enter the Sametime Server address and the port number on which the Sametime Server is listening and click **OK**.

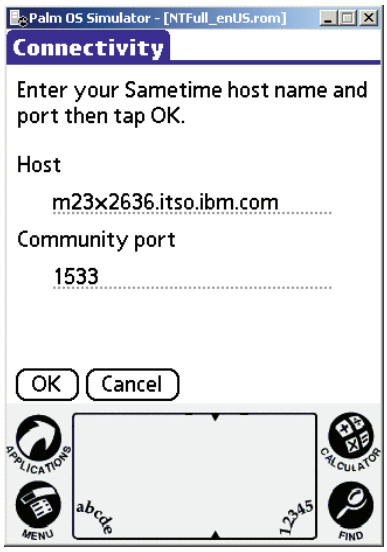


Figure 24-4 Connectivity

3. Enter the login details and click **Logon**.

Starting Sametime on Pocket PC

Click **Start** and click the **Sametime** icon as shown in Figure 24-5.



Figure 24-5 Starting Sametime on Pocket PC

Enter the connectivity information and the login information and click **Logon**.



Figure 24-6 Startup screen on Pocket PC

Adding users and chatting

For Palm devices, click **Menu-> Tools** and for Pocket PC devices click the **Tools** tab. This will bring up the screens shown in Figure 24-7 on page 835.

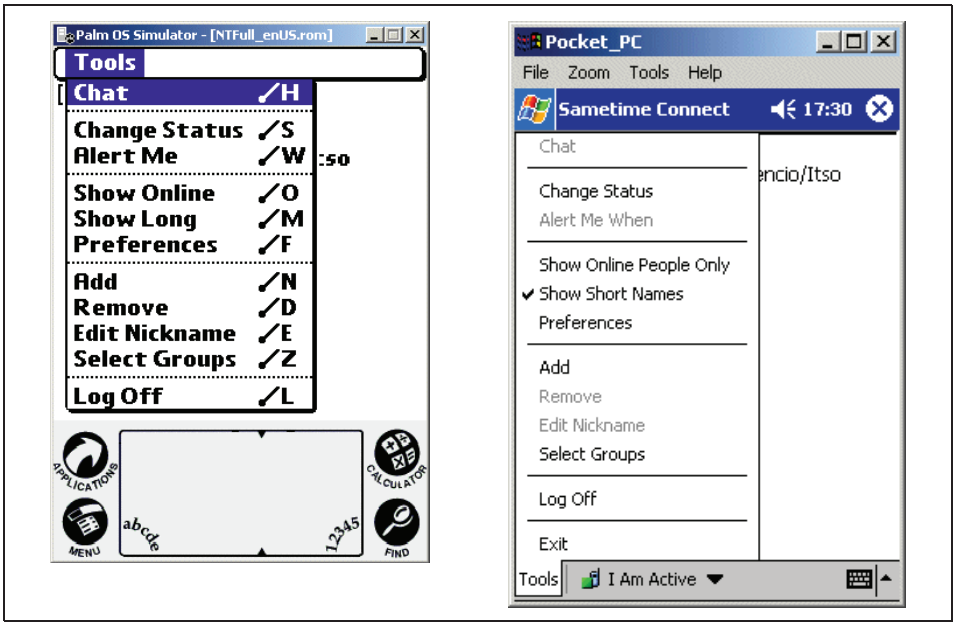


Figure 24-7 Tools

To add new users, click **Add**.

To chat with online users, click the user to chat. Then open **Tools** and click **Chat**.

Figure 24-8 on page 836 shows an ongoing chat session with Pocket PC, Palm, and desktop users.

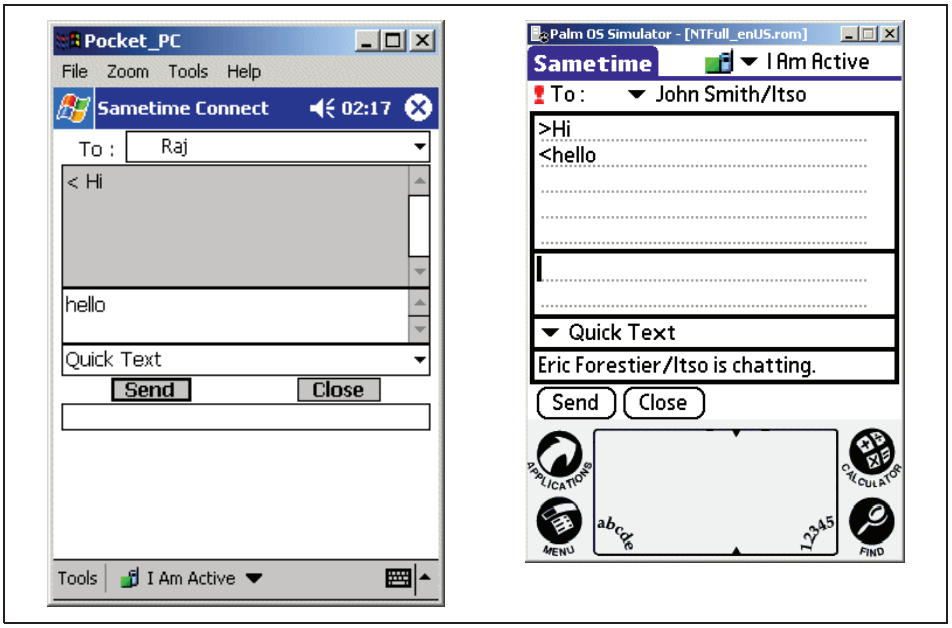


Figure 24-8 Chat session

Figure 24-9 on page 837 shows a desktop user initiating a chat session with a Palm user. The Palm user can switch to the new chat session by clicking the **To** drop-down list and selecting the appropriate user.

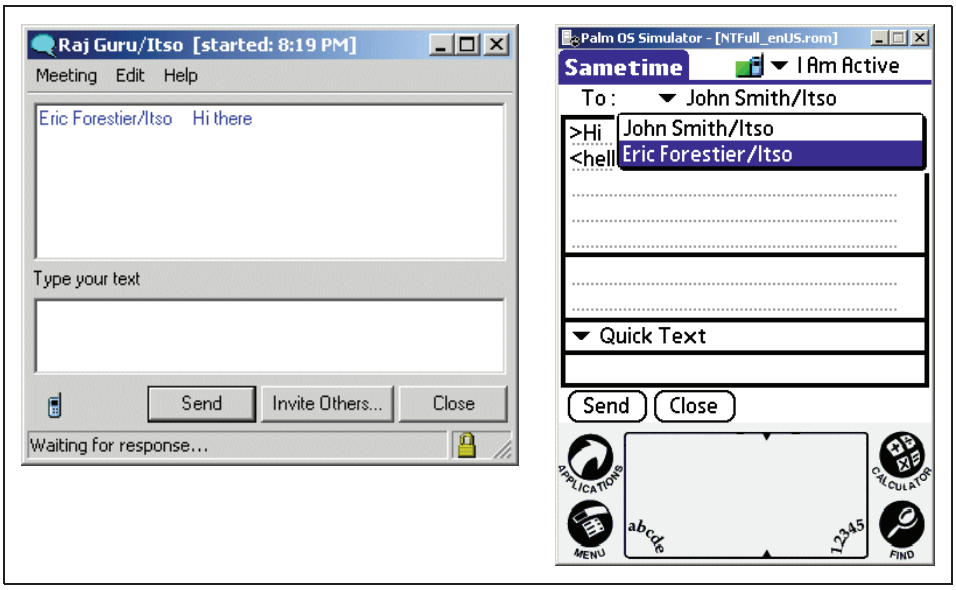


Figure 24-9 Chat sessions

25

Intelligent Notification Services (INS)

This chapter describes the Intelligent Notification Services (INS) within WebSphere Everyplace Access. Using sample scenarios, we provide instructions on how to administer INS, how to configure and develop extended services for users, and how users can benefit from these services.

Important: To ensure you have the latest code, you should visit the WebSphere Everyplace Access support Web site at http://www.ibm.com/software/pervasive/ws_everyplace_access/support/ to download the latest fix packs and code fixes. Apply these fixes prior to using the samples within this redbook.

© Copyright IBM Corp. 2003. All rights reserved.

25.1 Overview

Intelligent Notification Services notifies users via multiple delivery channels, based on user preferences and subscriptions.

Notifications are sent to users via delivery channels, which can include such mechanisms as portlets, Lotus Sametime, SMTP e-mail, and Short Messaging Service (SMS).

Intelligent Notification Services users can set preferences that affect where, when, how, by whom, and with which delivery channel the users are notified.

25.1.1 Concepts

Intelligent Notification Services can be divided into building blocks as illustrated in Figure 25-1.

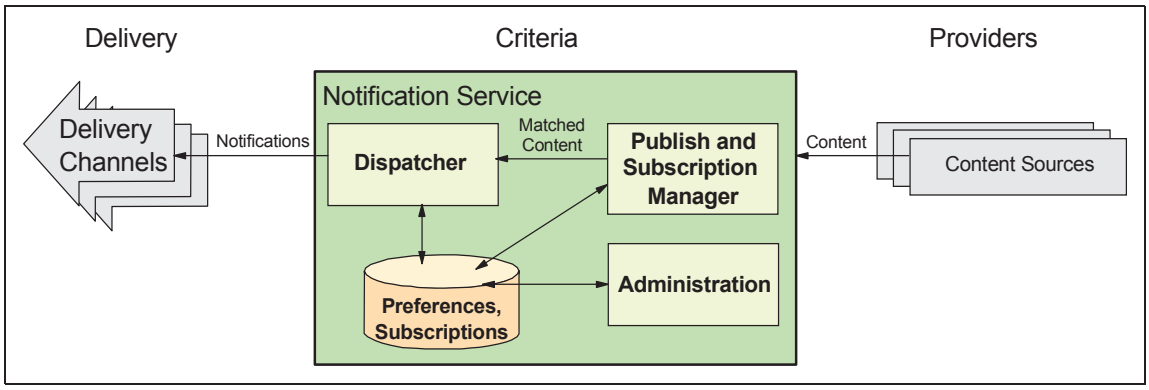


Figure 25-1 Intelligent Notification Services building blocks

Content is received from various sources such as content suppliers, applications or databases.

Content is received by the Publish and Subscription Manager, which compares it to user subscriptions and provides matched content to the Dispatcher.

Dispatcher delivers matched content to users based on their preferences.

Delivery channels provide interfaces to user devices such as Sametime, SMS, WAP, or SMTP.

25.1.2 Components

Intelligent Notification Services is made of the components illustrated in Figure 25-2.

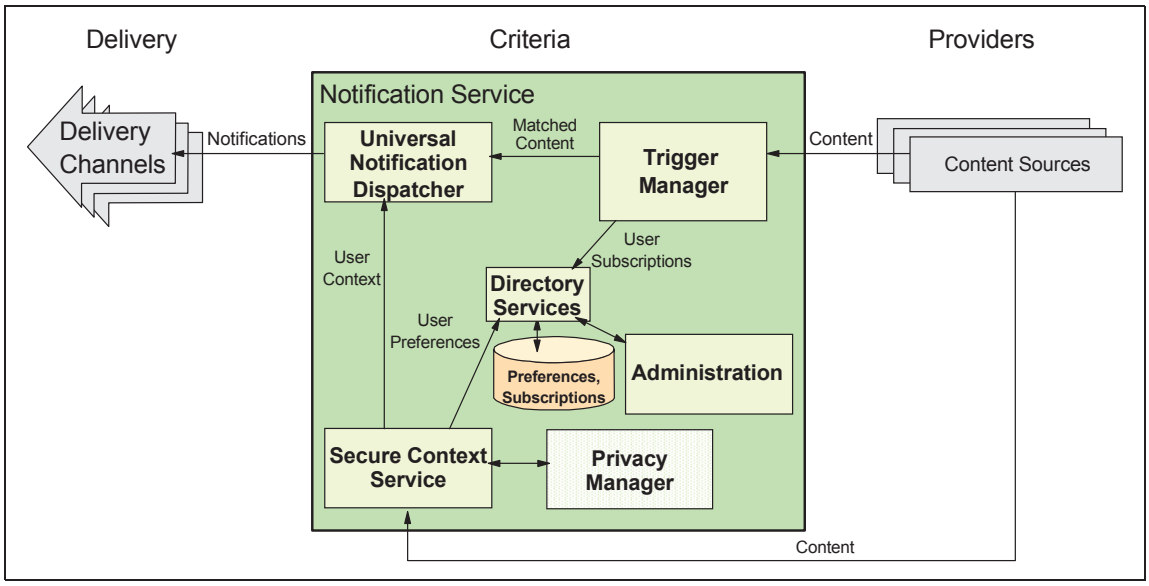


Figure 25-2 Intelligent Notification Services components

Administration

Intelligent Notification Services administration is performed via portlets as follows:

- Administrative portlets are used by the INS administrator to manage servers, configure gateway adapters, remove preferences for deleted users, and configure e-mail subscriptions.
- User portlets are used by Intelligent Notification Services users to manage delivery channels, manage their user groups, and specify rules.
- The Message Center portlet is used by an Intelligent Notification Services user to view messages, delete messages, and send simple notifications to other Intelligent Notification Services users.

Directory Services Engine

Directory Services Engine provides management services for configuration and runtime parameters, such as host names, ports, and configuration parameters for all of the servers.

Content adapter

Content adapter is an application that preprocesses input from data sources, converts captured data to Trigger Manager format, and publishes data to Trigger Manager for matching against user subscriptions.

Several content adapters are provided with Intelligent Notification Services, including content adapters for XML, RSS News, Lotus Notes e-mail, and Microsoft Exchange e-mail.

Trigger Manager

The Trigger Manager manages user subscriptions.The subscriptions specify filters to match against content from the content adapters. The Trigger Manager matches the content against subscription filters. When a match occurs, Trigger Manager calls the appropriate trigger handler, which notifies the appropriate user via the Universal Notification Dispatcher.

Universal Notification Dispatcher

The Universal Notification Dispatcher receives requests to send notifications both from other users and applications and from the Trigger Manager. It then dispatches notifications based on subscriber delivery preferences stored in a database by both the Services Preferences Manager and the Secure Context Server.

The Universal Notification Dispatcher interfaces with these delivery channels through gateway adapters. There is a gateway adapter for each type of delivery channel that Intelligent Notification supports. Each type of delivery channel requires that the message be in a specific format. Gateway adapters use transcoders to transform notifications into the proper format for each delivery channel.

Intelligent Notification Services provides the following gateway adapters, as illustrated in Figure 25-3 on page 843:

- Sametime instant messaging transcoder
- Simple Message Transfer Protocol (SMTP) transcoder
- Short Message Service (SMS) transcoder
- Wireless Application Protocol (WAP) transcoder

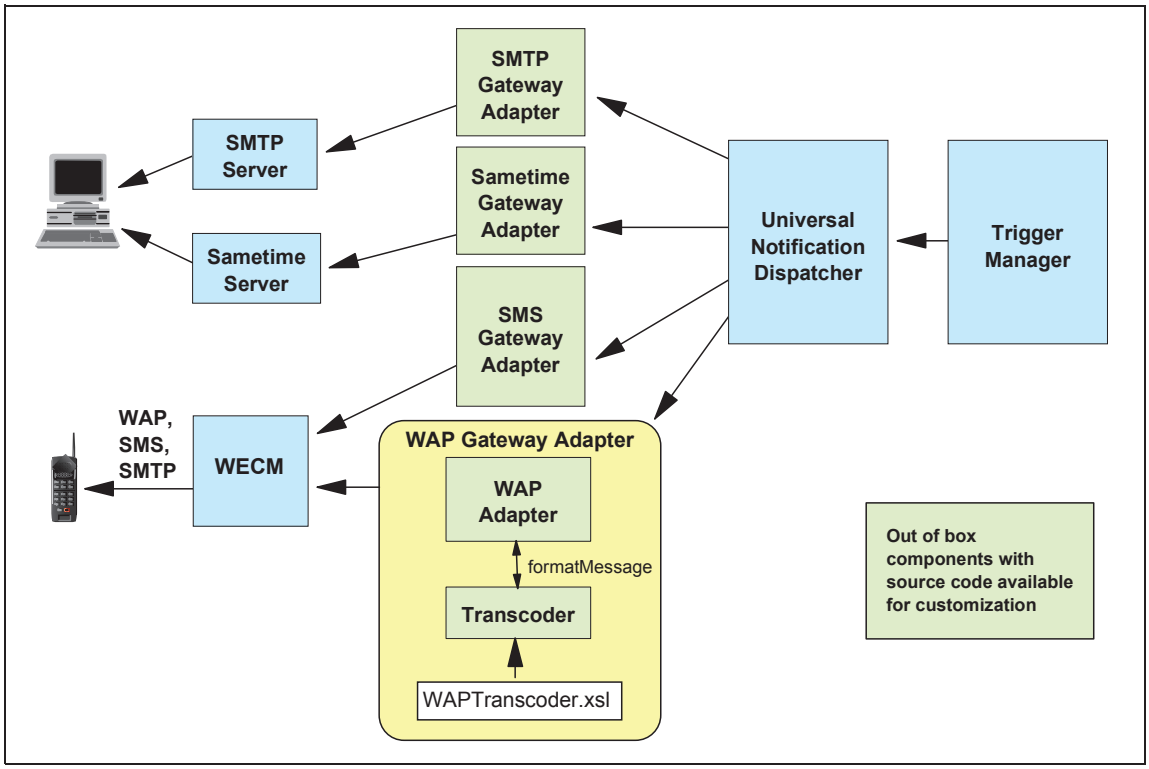


Figure 25-3 Intelligent Notification Services provided gateway adapters

Secure Context Server

Note: This component is currently not used by Intelligent Notification Services.

The Secure Context Server provides user context information thanks to context drivers.

Context drivers gather and distribute a specific category of user context information. Context information is information about a user's context, such as the user's physical location, online presence, or what activity the user is currently engaged in.

For example, one context driver is responsible for determining a user's Sametime availability, another is responsible for knowing a user's physical location, and a third is responsible for knowing whether a user is in a meeting.

The context driver gets context information from its associated information source and then stores that information in a context cache.

Note: With this release Intelligent Notification Services provides a location context driver sample.

Privacy Manager

Note: This component is currently not used by Intelligent Notification Services.

The user Privacy Manager provides management services for user privacy policies. The Secure Context Server queries the Privacy Manager for privacy constraints.

The Privacy Manager protects user-sensitive information, and to send messages only to devices the user has authorized the sender to use.

25.1.3 INS application programming interfaces

Intelligent Notification Services provide Java APIs that allow the development of custom code that can perform the following tasks:

- Subscribe to content (via TriggerManager class)
- Publish content (via TriggerManager class)
- Define trigger handlers (via TriggerHandlerV2 class)
- Manage subscriptions (via TriggerManager class)
- Send notifications (via NotificationService class)
- Custom gateway adapters

Intelligent Notification Services provides sample code that demonstrates how to use the external APIs.

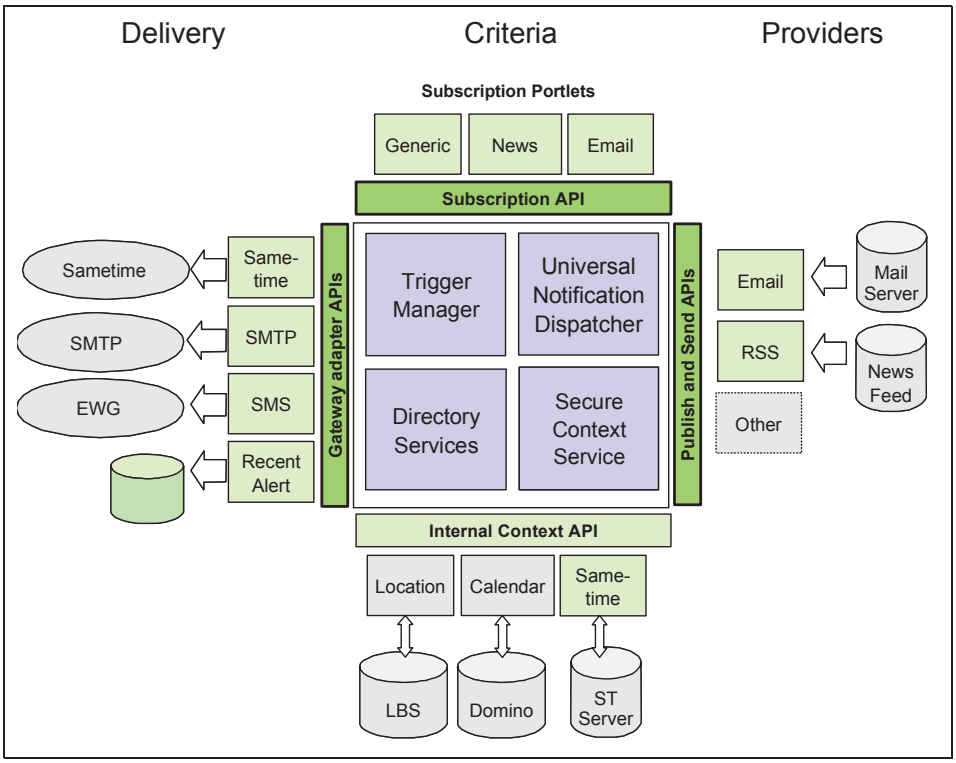


Figure 25-4 Intelligent Notification Services application programming interfaces

25.2 Intelligent Notification Services enablement

The following WebSphere Everyplace Access post-install steps need to be performed to enable users for Intelligent Notification Services:

1. Run the **`updatePortletCfg`** if you are running in a distributed environment.
2. Start the Intelligent Notification Services administration servers:
   - Intelligent Notification Services administration server
   - Intelligent Notification Services host administration server

   Note: When running in a distributed environment, there must be one Intelligent Notification Services administration server for the whole platform and one Intelligent Notification Services host administration server per machine where Intelligent Notification Services reside.

3. Configure the Intelligent Notification Services servers

4. Start the servers.
5. Create Intelligent Notification Services users.

25.2.1 Update portlet configuration (distributed environment only)

1. Open a DOS command prompt window.
2. Change to directory C:\WebSphere\INS\bin.
3. Type the command to update portlet configuration:

```
updatePortletCfg db_hostname jdbc_db_port db_name db_userid db_password
db2admin_hostname admin_port dse_port
```

For example:

```
updatePortletCfg host.location.company.com 0000 insdb2 db2admin db2admin
db2 adminhost.location.company.com
```

25.2.2 Starting INS administration servers

Intelligent Notification Services are administered within WebSphere Everyplace Access via the Intelligent Notification Services administration portlets. Most of the Intelligent Notification Services servers can be started from the INS Administration portlet. Yet, the INS administration server and the INS host administration server need to be started from a DOS window prior to any other INS step.

Starting INS administration server

1. Ensure that the JDBC listener is running locally. The JDBC listener must be running on the computer on which DB2 is installed. For DB2 installed on Windows:
   a. Select **Start -> Settings ->Control Panel -> Administrative Tools -> Services**.
   b. Scroll to the entry for JDBC.
   c. Check to see that the startup type is Automatic.

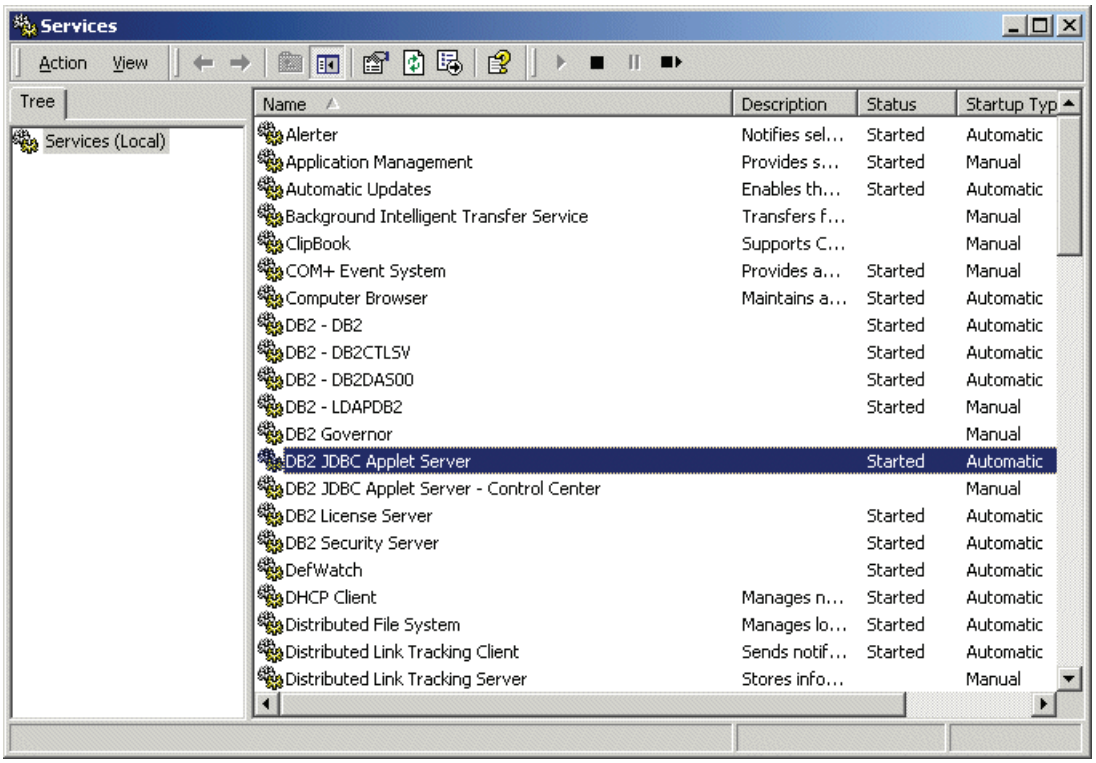


Figure 25-5 Checking JDBC listener

d. If the startup type is not Automatic, right-click the **JDBC** entry to bring up the properties. Change the startup type to Automatic. When you restart Windows, the JDBC listener is started automatically.

2. Open a DOS command prompt window.

> **Note:** You may alternatively select **Start -> Programs -> IBM Intelligent Notification services -> Start Administration Component**.

3. Change to directory C:\WebSphere\INS\bin.
4. Type the command **startADM <*short host name*>** (`mka6brky` in the shown example). You should see the following message:

```
ADM0004I The Administration server has started.
```

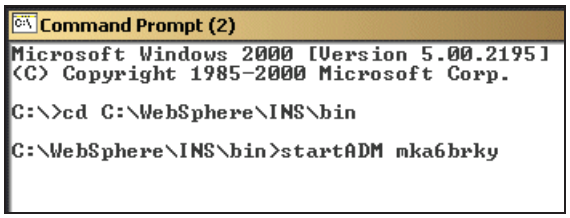


Figure 25-6 Starting INS administration server

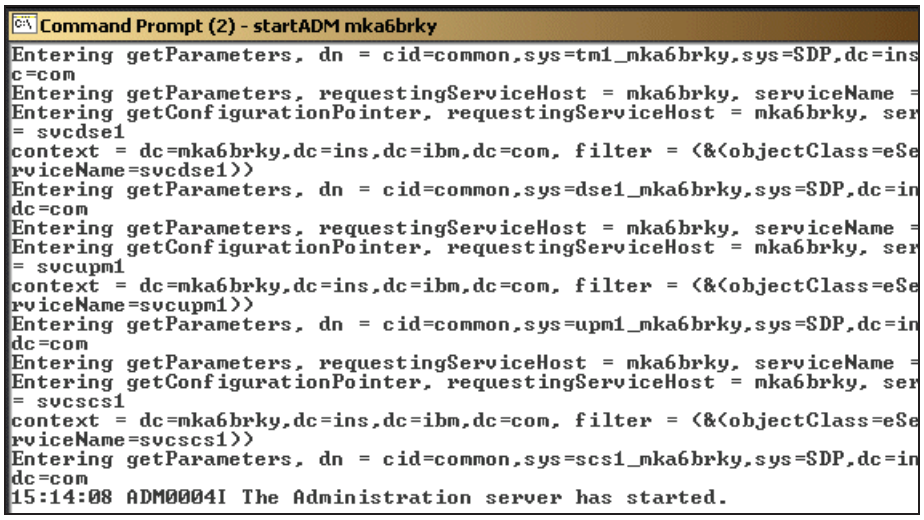


Figure 25-7 INS administration server starting log

Important: If you do not keep this window opened, the INS administration server would stop.

Starting INS host administration server

1. Open a DOS command prompt window.

Note: You may alternatively select **Start -> Programs -> IBM Intelligent Notification services -> Start HA**.

2. Change to directory C:\WebSphere\INS\bin.

3. Type the command **startHA**.

   You should see the following messages:

```
Starting up HostAdministrator
Listening on port 55013
LOG0004E No handlers are registered with the logger.
```

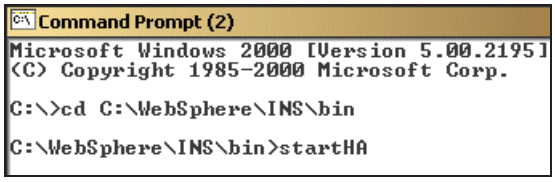


Figure 25-8 Starting INS host administration server

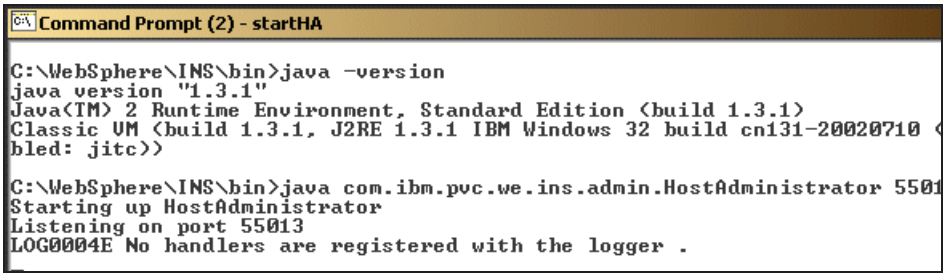


Figure 25-9 INS host administration starting log

Important: Keep this window open or the INS Host administration server will stop.

25.2.3 INS server configuration

In this section, we provide a sample INS server configuration.

25.2.4 Starting INS servers

Log into the Portal as an administrative user (most likely user `wpsadmin` with password `wpsadmin`), and select **Intelligent Notification -> Administration -> Manage Servers** portlet as shown underlined in Figure 25-10 on page 850.

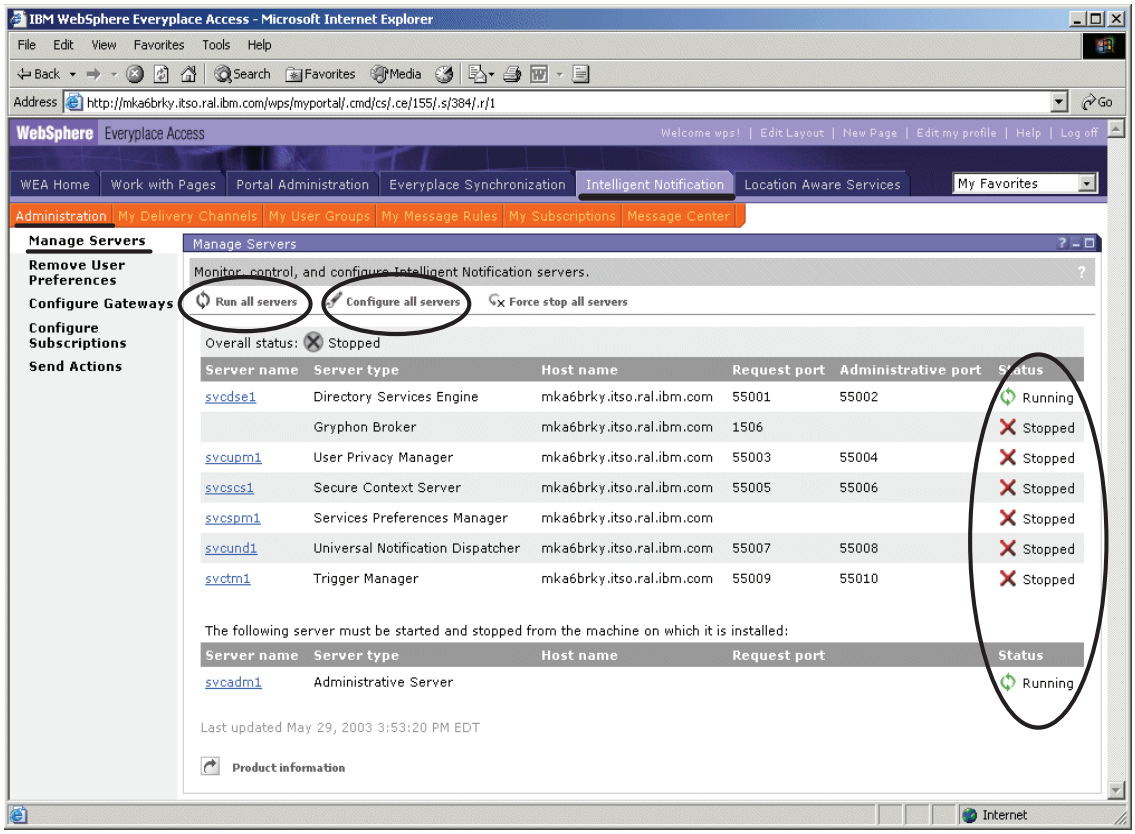


Figure 25-10 INS Manage Servers portlet

As circled on the right of Figure 25-10, you can see that Directory Services and Administrative Server are running and all other servers are stopped.

When all servers are in Stopped or Pause state, click **Run all servers** (circled at the top of Figure 25-10) and wait until all the servers are in the Running state.

Overall status: Running

Server name	Server type	Host name	Request port	Administrative port	Status
svcdse1	Directory Services Engine	mka6brky.itso.ral.ibm.com	55001	55002	Running
	Gryphon Broker	mka6brky.itso.ral.ibm.com	1506		Running
svcupm1	User Privacy Manager	mka6brky.itso.ral.ibm.com	55003	55004	Running
svcscs1	Secure Context Server	mka6brky.itso.ral.ibm.com	55005	55006	Running
svcspm1	Services Preferences Manager	mka6brky.itso.ral.ibm.com			Running
svcund1	Universal Notification Dispatcher	mka6brky.itso.ral.ibm.com	55007	55008	Running
svctm1	Trigger Manager	mka6brky.itso.ral.ibm.com	55009	55010	Running

The following server must be started and stopped from the machine on which it is installed:

Server name	Server type	Host name	Request port	Status
svcadm1	Administrative Server			Running

Figure 25-11 INS started servers

25.2.5 Enable INS users

Enabling a given user as an INS user means making it part of the INSUsersGroup.

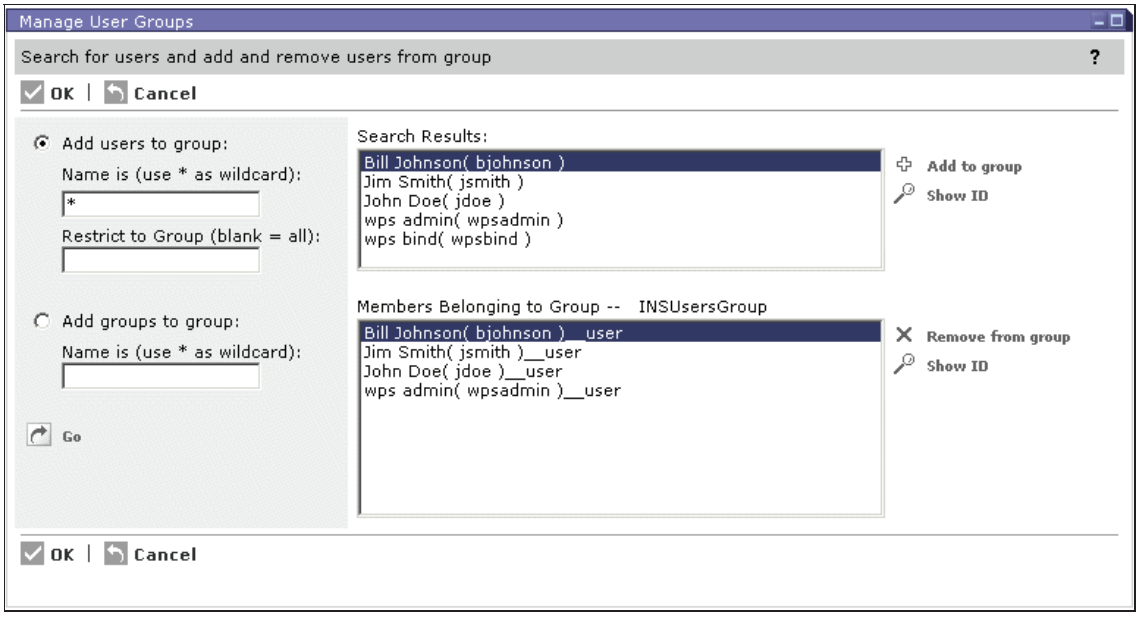


Figure 25-12 Enabling users for INS

Important: Once part of the INSUsersGroup, each user has to log in to WebSphere Everyplace Access 4.3 server once to complete the INS user initialization process, such as INS database and LDAP settings.

25.3 INS implementation

INS usage means the implementation of new notification channels through the provided APIs and the configuration of INS and its users for these new and built-in channels.

25.3.1 Defining the needs

The following scenarios show the Intelligent Notifications Services of WebSphere Everyplace Access.

Scenario 1

A user named *John Doe* wants to have the INS log file updated when getting an urgent Microsoft Exchange 2000 mail from his manager, whom we will name *Jim Smith*.

Scenario 2

Jim Smith wants to receive an instant message if connected or an e-mail note when weather condition changes.

Scenario 3

Bill Johnson wants to receive a WAP Push message for news about his favorite soccer team.

In all the scenarios, the Message Center portlet gets updated.

25.3.2 Analyzing the needs

These scenarios have the following needs:

- The needed content sources are:
  - Microsoft Exchange 2000 Server
  - Weather
  - news Soccer

  Subscriptions and Content Adapters for these content sources will have to be configured and started.

- The needed delivery channels are:
  - INS log file
  - Sametime
  - E-mail
  - WAP Push
  - Message Center portlet

  Gateway Adapters for these delivery channels will have to be configured and started.
- User John Doe needs to be configured for:
  - One INS log file delivery channel
  - One Message Center delivery channel
  - One Microsoft Exchange subscription
- User Jim Smith needs to be configured for:
  - One Sametime delivery channel
  - One SMTP mail delivery channel
  - One Message Center delivery channel
  - One Weather subscription
- User Bill Johnson needs to be configured for:
  - One WAP delivery channel
  - One Message Center delivery channel
  - One news subscription

25.4 INS development: Custom Delivery Channel

In Scenario 1, notification is delivered to the INS log file, which is not a delivery channel delivered by INS. Therefore, it requires the development of a custom gateway adapter, which means development and deployment of the following elements:

- Gateway adapter: Interfaces to the delivery channel
- Transcoder and Transcoder stylesheet: Formats the notification for delivery
- Delivery channel portlet: Enables users to add delivery channels of the custom type and specify properties for those delivery channels

25.4.1 Custom gateway adapter

The gateway adapter provides an interface to a particular type of delivery channel. Notifications are received by the gateway adapter, transformed in a way that fits the delivery channel, then sent to the delivery channel.

To develop a custom gateway adapter, a Java class must be written that implements the API interface class DynamicGateway. Its purpose is to format the notification according to the custom delivery channel specifications and then send it to the delivery channel via the appropriate interface.

Notification formatting is done by a transcoding engine that is part of the Universal Notification Dispatcher. This engine may be used in two different ways:

- Via the implementation of a Java class that extends the DynamicTranscoder class.
- Or via the creation of a custom XSL stylesheet for the transcoder to use.

In either case, a notification is received in NotificationML format and must be transformed into a string that is understandable by the delivery channel.

Notification ML is the markup language used to format a notification. It is a form of XML.

The transcoding engine knows which technique to use based on the gw.properties file provided with the custom gateway adapter, and discussed later in this chapter.

Note: Even when the XSL is used, an empty Java class must be provided.

The gw.properties file contains the information needed by INS to configure and install the custom gateway adapter, to be stored in the LDAP Directory Server. It is made of two sections: one for the common gateway parameters, and one for the parameters that are specific to each gateway adapter, such as a host name and port number, or an administrative user and password.

Sample code

Sample code of a gateway adapter can be found in the directory C:\WebSphere\INS\samples\adapters. Since the INS Message Center and the server-initiated action are implemented as custom gateway adapters, you may also want to look for their implementation in the directories C:\WebSphere\INS\UNDGateways\RecentAlerts and C:\WebSphere\INS\UNDGateways\ServerInitiatedActions.

Development

Developing the gateway adapter using the WebSphere Everyplace Access 4.3 provided in WebSphere Studio Site Developer is recommended, on a machine other than the WebSphere Everyplace Access 4.3 server.

To install WebSphere Studio Site Developer, see Appendix G, “Portlet development platform installation” on page 1211.

Important: The Java classes may be packaged in a single or in different Java class files, but *should not* be part of any Java package.

For the sake of clarity, the following assumes these Java classes are packaged into two different files.

Writing this class with WebSphere Studio Site Developer means creating a new Java project.

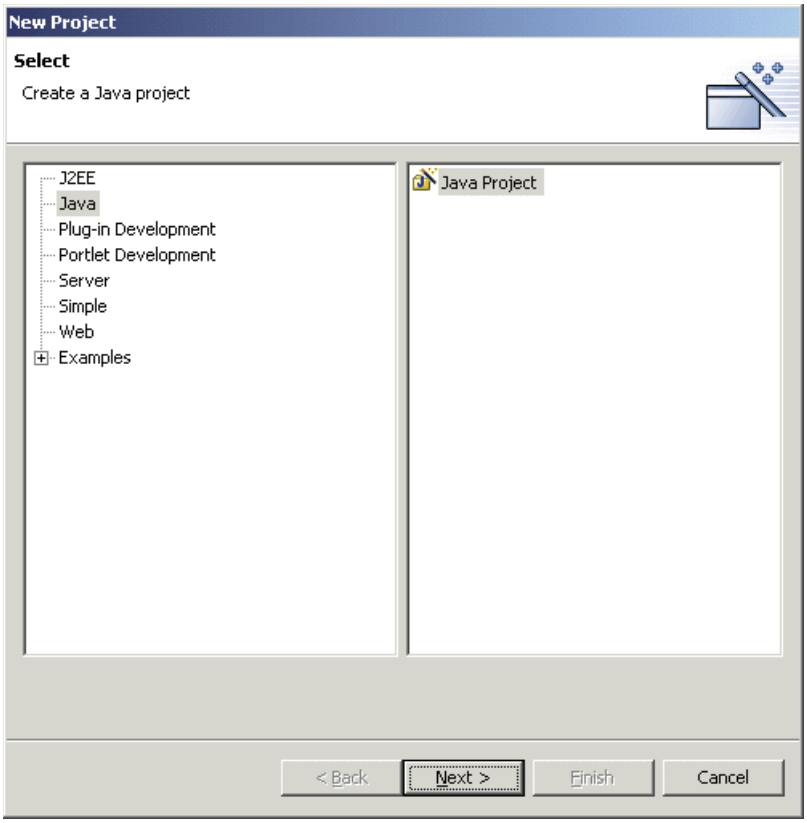


Figure 25-13 Creating a new Java project

The Java classpath must contain the INS JAR files ins.jar and jlog.jar, copied from the directory C:\WebSphere\INS\lib of the WebSphere Everyplace Access 4.3 server.

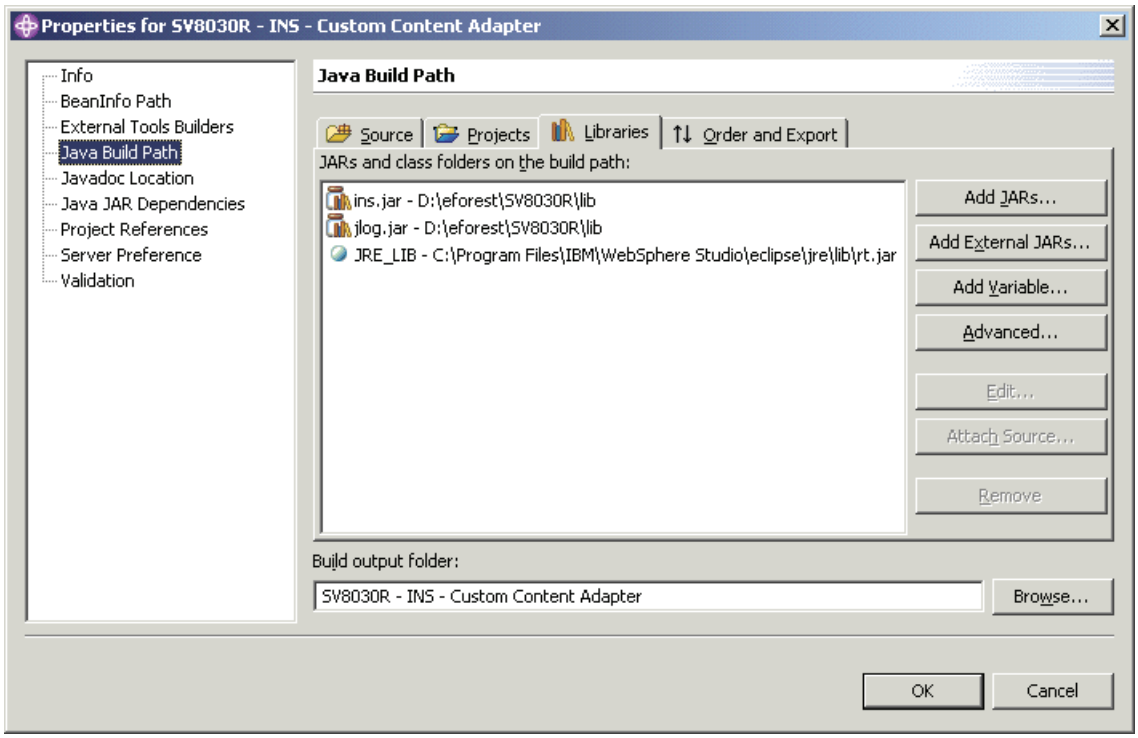


Figure 25-14 Java classpath

Custom gateway adapter class

First the Java class MyCustomGatewayAdapter.java is created, which implements the DynamicGateway interface:

- The header instructions (see Example 25-1) consist only of the required import packages, without any `package` instruction. The imported INS packages are part of the ins.jar file.

Example 25-1 Header instructions

```
import java.lang.*;
import java.util.*;

import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.*;
import com.ibm.pvc.we.ins.und.server.adaptors.*;
import com.ibm.pvc.we.ins.und.server.dispatcher.*;
```

- The class declaration (Example 25-2 on page 857) implements the DynamicGateway interface as well as com.ibm.loggin.IRecordType for logging purposes.

Example 25-2 Class declaration

```
public class MyCustomGatewayAdapter implements DynamicGateway,
   com.ibm.logging.IRecordType
{
```

- Internal variables used further in the class are shown in Example 25-3.

Example 25-3 Internal variables

```
   private DynamicTranscoder currenttc;
   private DeviceAddress deviceAddr;
   private Msg msg;
   private String msgbody;
   private String dbname;
private boolean loaded;
private Factory myFactory;
```

- The class constructor, used to initialize internal variables, is shown in Example 25-4.

Example 25-4 Class constructor

```
public MyCustomGatewayAdapter()
{
   // Initialize variables for this class
   dbname = null;
   loaded = false;

}
```

- The method setProperties is implemented from the DynamicGateway interface. This method allows the reading of the properties from the specific section of the gw.properties file. See Example 25-5.

Example 25-5 setProperties

```
public void setProperties(Properties properties)
{
   if (Logging._isTracing)
      Logging._tracer.text(TYPE_WARN, this, "setProperties", “Entry”);

   // Read gw.properties file
   if (!loaded)
   {
      dbname = properties.getProperty("dbname");
      loaded = true;
   }
}
```

- The method setFactory is implemented to instantiate the transcoder. See Example 25-6.

Example 25-6 setFactory

```
public void setFactory(Factory factory)
{
   if (Logging._isTracing)
      Logging._tracer.text(TYPE_WARN, this, "setFactory", "Entry");
   myFactory = factory;
}
```

- The method sendMessage is the primary method that is called by INS when notification is delivered through this gateway. According to the specifics of this delivery channel, It formats the message via the formatMessage method, then sends it to the delivery channel. See Example 25-7.

Example 25-7 sendMessage

```
public int sendMessage(DeviceAddress addr, Msg msg)
{
   if (Logging._isTracing)
      Logging._tracer.text(TYPE_WARN, this, "sendMessage", "Entry");

   // Initialize variables for this session
   deviceAddr = addr;
   currenttc  = null;

   // Transcoder and message body are set in the formatMessage method
   msgbody  = null;
   this.msg = msg;
   formatMessage(addr, msg);

   // send the message
   int rc = send(addr, msg);

   return rc;
}
```

- The method formatMessage actually instantiate the transcoder, then sends the message to the appropriate transcoder to format the notification according to the device. The returned formatted notification will then be used by the sending method to actually send the notification to the device. See Example 25-8.

Example 25-8 formatMessage

```
public void formatMessage(DeviceAddress addr, Msg notificationMsg)
{
```

```
   if (Logging._isTracing)
      Logging._tracer.text(TYPE_WARN, this, "formatMessage", "Entry");

   // get proper transcoder from tc_collection
   // The transcoder key name is build from the information
   // found in the gw.properties file
   try
   {
      Class myClass = myFactory.getTranscoder();
      currenttc= (DynamicTranscoder) myClass.newInstance();
      currenttc.setFactory(myFactory);
   }
   catch (Exception w)
   {
      Logging._tracer.text(TYPE_ERROR, this, "formatMessage",
         "Could not create a new instance of MY_CUSTOM transcoder class");
      w.printStackTrace();
   }

   if (currenttc != null)
      msgbody=currenttc.format(addr,notificationMsg);
}
```

- The send internal method sends the notification to the delivery channel according to its specifications (write to INS log in this example). See Example 25-9.

Example 25-9 send

```
private int send(DeviceAddress addr, Msg msg)
{
   if (Logging._isTracing)
   {
      Logging._tracer.text(TYPE_WARN, this, "send", "Entry");

      // write notification to the output log
      Logging._tracer.text(TYPE_DEFAULT_MESSAGE, this, "send",
         "###### INS LOG CUSTOM DELIVERY CHANNEL : Message Start ######");
      Logging._tracer.text(TYPE_DEFAULT_MESSAGE, this, "send",
         msg.toString());
      Logging._tracer.text(TYPE_DEFAULT_MESSAGE, this, "send",
         "####### INS LOG CUSTOM DELIVERY CHANNEL : Message End #######");
   }

   // always return OK
   return 0;
}
```

Custom transcoder class

The second Java class MyCustomTranscoder.java is created, which extends the DynamicTranscoder interface.

Note: INS provides two different classes for implementing a transcoder: the DynamicTranscoder class and the GenericTranscoder class:

- The DynamicTranscoder class is used to implement dynamically added custom transcoders such as the one we are currently describing.
- The GenericTranscoder class is used internally by core INS to declare the built-in delivered transcoders, which are WAP, Messaging, Sametime, and e-mail.

- The header instructions consist only of the required import packages, without any package instruction. The imported INS packages are part of the ins.jar. See Example 25-10.

Example 25-10 Header instructions

```
import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.*;
import com.ibm.pvc.we.ins.und.server.dispatcher.*;
import com.ibm.pvc.we.ins.GeneralConstants;
```

- The class declaration extends the GenericTranscoder class. See Example 25-11.

Example 25-11 Class declaration

```
public class MyCustomTranscoder extends DynamicTranscoder
{
```

- The class constructor calls the superclass constructor. See Example 25-12.

Example 25-12 Class constructor

```
    public MyCustomTranscoder()
    {
       super();
    }
```

- The method `setFactory` will be used further to actually instantiate a new transcoder. See Example 25-13.

Example 25-13 setFactory

```
private Factory myFactory;

public void setFactory(Factory f)
```

```
   {
      myFactory = f;
   }
```

- The method formatMessage transforms the notification from NotificationML format into a String that can be read by the delivery channel. See Example 25-14.

Example 25-14 formatMessage

```
   public String formatMessage(DeviceAddress addr, Msg msg)
   {
      if (Logging._isTracing)
         Logging._tracer.text(TYPE_WARN, this, "formatMessage", "entry");

      // CAN BE EMPTY!

      // format message for this specific delivery channel

      // Create a string to hold the original Notification ML
      String notificationML=msg.extractML();
      if (Logging._isTracing)
         Logging._tracer.text(TYPE_WARN, this,
          "formatMessage", "received following XML: " + notificationML);

      // Create a StringBuffer to build the newly formatted message
      StringBuffer b=new StringBuffer();

      // Build formatted message
      b.append("msg from: ").append(msg.fromUserid).append("\n");
      b.append("msg to: ").append(msg.toUserid).append("\n");
      b.append("msg subject: ").append(msg.subject).append("\n");

      // Return newly formatted message string
      return b.toString();
   }
```

- The method setStylesheet will be used by INS to instantiate this transcoder. See Example 25-15.

Example 25-15 setStylesheet

```
public void setStylesheet()
{
   // stylesheet will be set only once during transcoder object life time
   // if no stylesheet is provided, stylesheet = "NULL"
   if ( stylesheet == null )
   {
      try
```

```
        {
           stylesheet = "notNull";
           //create transformer
           TransformerFactory tFactory = TransformerFactory.newInstance();
           transformer = tFactory.newTransformer(new
            StreamSource(myFactory.getStyleSheet()));
        }
        catch (TransformerConfigurationException tce)
        {
           tce.printStackTrace();
           if (Logging._isTracing)
           {
              Logging._tracer.exception(TRANSCODERS,this,"setStylehsheet",tce);
              Logging._tracer.stackTrace(TRANSCODERS,this,"setStylesheet");
           }
        }
     }
}
```

Custom Transcoder XSL file

The transcoder XSL stylesheet is used to transform the notification in a format suitable to the delivery channel.

The notification is written in NotificationML. The specifications for NotificationML are described in Table 25-1.

Table 25-1 NotificationML specifications

Tag	Description
<message>...</message>	Notification
<to>...</to>	User ID of message recipient
<from>...</from>	User ID of message sender
<subject>...</subject>	Subject of message
<text>...</text>	Text message body

The XSL stylesheet that transforms the notification should look like Example 25-16.

Example 25-16 Custom transcoder XSL template file

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:output encoding="utf-8" omit-xml-declaration="yes" method="text"/>
<xsl:template match="message">
```

```
   From: <xsl:value-of select="from"/>
   Sub: <xsl:value-of select="subject"/>
   Msg: <xsl:value-of select="text" />
   <xsl:for-each select="url">
      <xsl:text>URL:</xsl:text>
      <xsl:value-of select="." />
   </xsl:for-each>
<xsl:text>This text is closing the notification</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

For instance, let's say the NotificationML is as shown in Example 25-17.

Example 25-17 NotificationML sample

```
<message>
   <to>delivery channels</to>
   <from>jdoe@ins.realm</from>
   <subject>WEATHER</subject>
   <text>
      Surf City NC 12:56 PM EST on March 27, 2001
      Forecast for Wednesday 03/28/01
      Forecast Condition: Mostly Sunny
      Forecast high: near 60
      Forecast low: 40s
   </text>
   <url>http://www.weather.com</url>
```

The transformed notification will then be as shown in Example 25-18.

Example 25-18 Transformed notification sample

```
From: jdoe@ins.realm
Sub: WEATHER
Msg: Surf City NC 12:56 PM EST on March 27, 2001
Forecast for Wednesday 03/28/01
Forecast Condition: Mostly Sunny
Forecast high: near 60
Forecast low: 40s
URL: http://www.weather.com
This text is closing the notification
```

Custom gateway properties file

The gw.properties file contains the information needed by INS to configure and load the custom gateway adapter.

In its common section, it provides the identification of the gateway adapter, the names of the Java classes used for formatting and transcoding, whether it will use the XSL or the Java class, and some internal parameters.

The common section parameters must be specified in the form:

```
<LDAPsettingID>=<propertyFieldName>=<parameterValue>
```

The gateway specific parameters must be specified in the form:

```
<parameterName>=<value>
```

The common section parameters are as follows:

- `property ibm-undDeviceAdaptorLink`:
  - The property field name identifies the delivery channel protocol. It should be made of three parts separated by the # sign:
    - First part is the Protocol type
    - Second part is the Protocol
    - Third part is the Protocol version (optional)

    These three parts must match the ones found in the corresponding user's delivery channel. The third part is set to none in LDAP when not present.
  - The parameter value is the name of the class handling this gateway adapter.
- `property ibm-undDeviceTranscoderLink`:
  - The property field name is an identifier for the used transcoder, which must end with string '_TC'.
  - The parameter value is the name of the class handling this gateway transcoder. It may be the same as previous class name.
- `property ibm-undStyleSheetLink`:
  - The property field name is an identifier for the used transcoder stylesheet, which must end with string '_SS'.
  - The parameter value specifies either the class or the XSL file that will be used for message formatting.
- The following are internal parameters and should not be changed:
  - `ibm-undSupportsStringCOD=false`
  - `ibm-undAdaptorNumberOfRetries=2`
  - `ibm-undAdaptorRetryPeriod=40`

Example 25-19 gw.properties file

```
# UND Required
ibm-undDeviceAdaptorLink=MyChannel#MyChannel=MyCustomGatewayAdapter
```

```
ibm-undDeviceTranscoderLink=MYCUSTOM_TC=MyCustomTranscoder
ibm-undStyleSheetLink=MYCUSTOM_SS=MyCustomTranscoder.class
ibm-undSupportsStringCOD=false
ibm-undAdaptorNumberOfRetries=2
ibm-undAdaptorRetryPeriod=40

# Gateway specific
dbname=MYDB
```

Deployment

The following steps must be performed to deploy the custom gateway adapter into the WebSphere Everyplace Access 4.3 server:

1. Create the directory C:\WebSphere\INS\UNDGateways\MyGateway.
2. Copy files MyCustomGatewayAdapter.class, MyCustomTranscoder.class, MyCustomTranscoder.xls, and gw.properties into the directory C:\WebSphere\INS\UNDGateways\MyGateway.
3. From the WebSphere Everyplace Access 4.3 Portal, log in as an administrator, select **Intelligent Notification -> Administration -> Manage Servers**, then stop and restart all the INS servers.

Note: The custom delivery channels other than the Message Center will not appear in the Configure Gateways portlet.

25.4.2 Delivery channel portlet

The delivery channel portlet allows the user to add, configure, modify, and delete delivery channels. The delivery channel portlet defines:

- The delivery channel name
- The delivery channel type
- Times when the delivery channel is unavailable
- Any other channel-specific parameter, such as user ID or telephone number

Custom delivery channels are added via the DeliveryChannelManager.addDeliveryChannel method.

A custom channel delivery portlet is composed of:

- A controller Java class extending DeliveryChannelBaseController and implementing ActionListener
- A bean Java class extending DeliveryChannelBean and implementing Serializable
- A delivery channel add JSP used for adding delivery channels

- A delivery channel view JSP used to list the current delivery channels
- A subscription manager JSP used to manage the subscriptions

The provided channel delivery portlets are:

- Sametime delivery channels
- SMTP e-mail delivery channels
- Message center delivery channels
- WAP delivery channels
- SMS delivery channels
- Server-initiated actions delivery channels

Sample code

Sample code for custom channel delivery portlet is found in the WebSphere Everyplace Access 4.3 server directory C:\WebSphere\INS\samples\portlets. The following files are included:

- *DeliveryChannelBaseController.java:

  Logic handling the delivery channel portlets for Sametime, and SMTP samples.

- *DeliveryChannelBean.java:

  Delivery channel data storage bean for generic, news, stock, and weather samples.

Development

It is suggested that you use WebSphere Studio Site Developer on a machine other than the WebSphere Everyplace Access V4.3 server to develop the portlet.

The WebSphere Everyplace Access V4.3 Portal Toolkit should be installed in WebSphere Studio Site Developer to allow portlet development. To install WebSphere Studio Site Developer and/or the WebSphere Everyplace Access 4.3 Portal Toolkit, refer to Appendix G, “Portlet development platform installation” on page 1211.

Creating Java project

1. Create a new Java JSP portlet project. See Figure 25-15 on page 867.

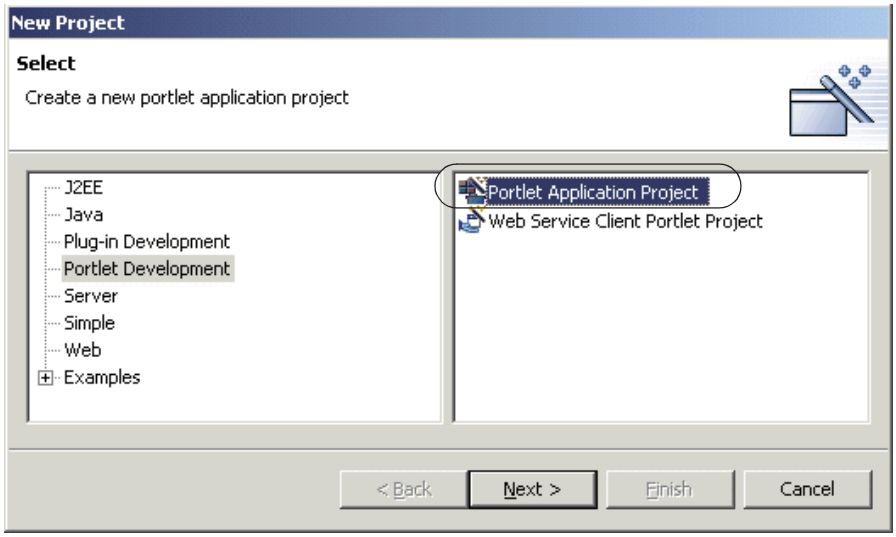


Figure 25-15 New portlet application project

2. Define the portlet project. For example see Figure 25-16.

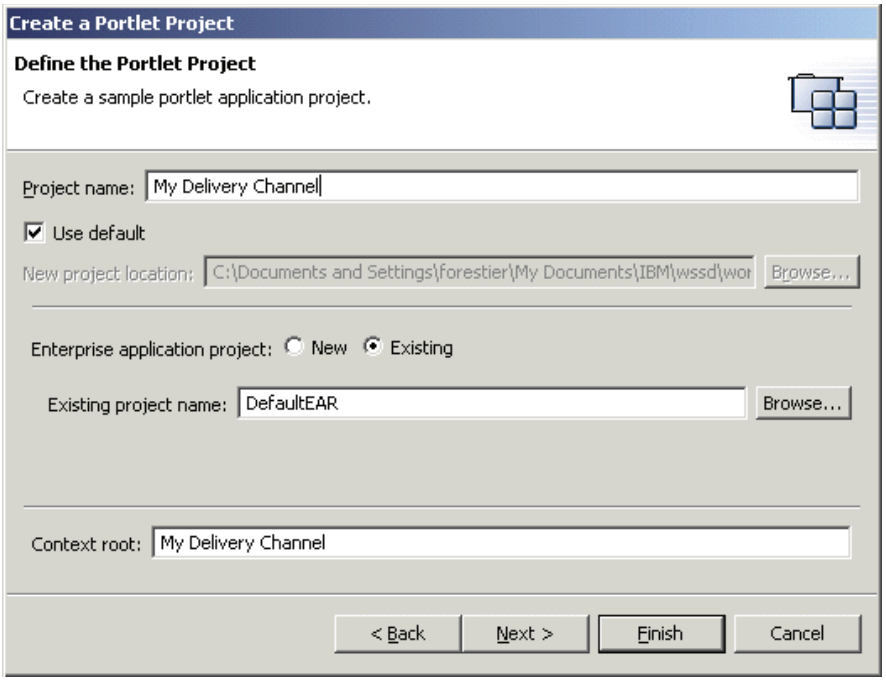


Figure 25-16 Define the new project

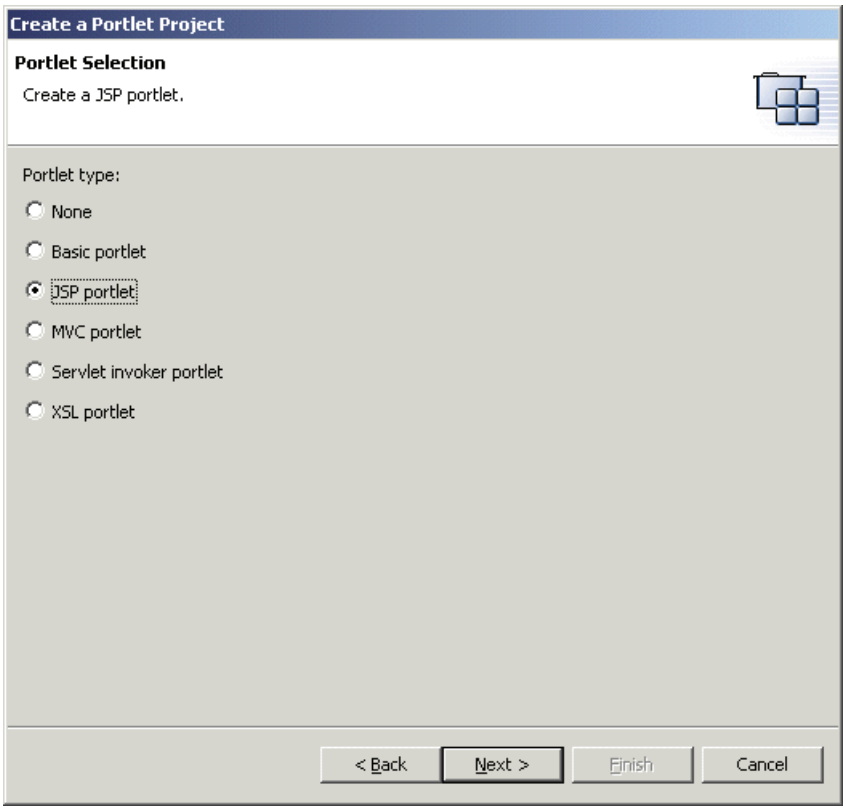


Figure 25-17 JSP Java Project

3. Delete the JSP directory and the default package (including Dummy.java), which will not be used.

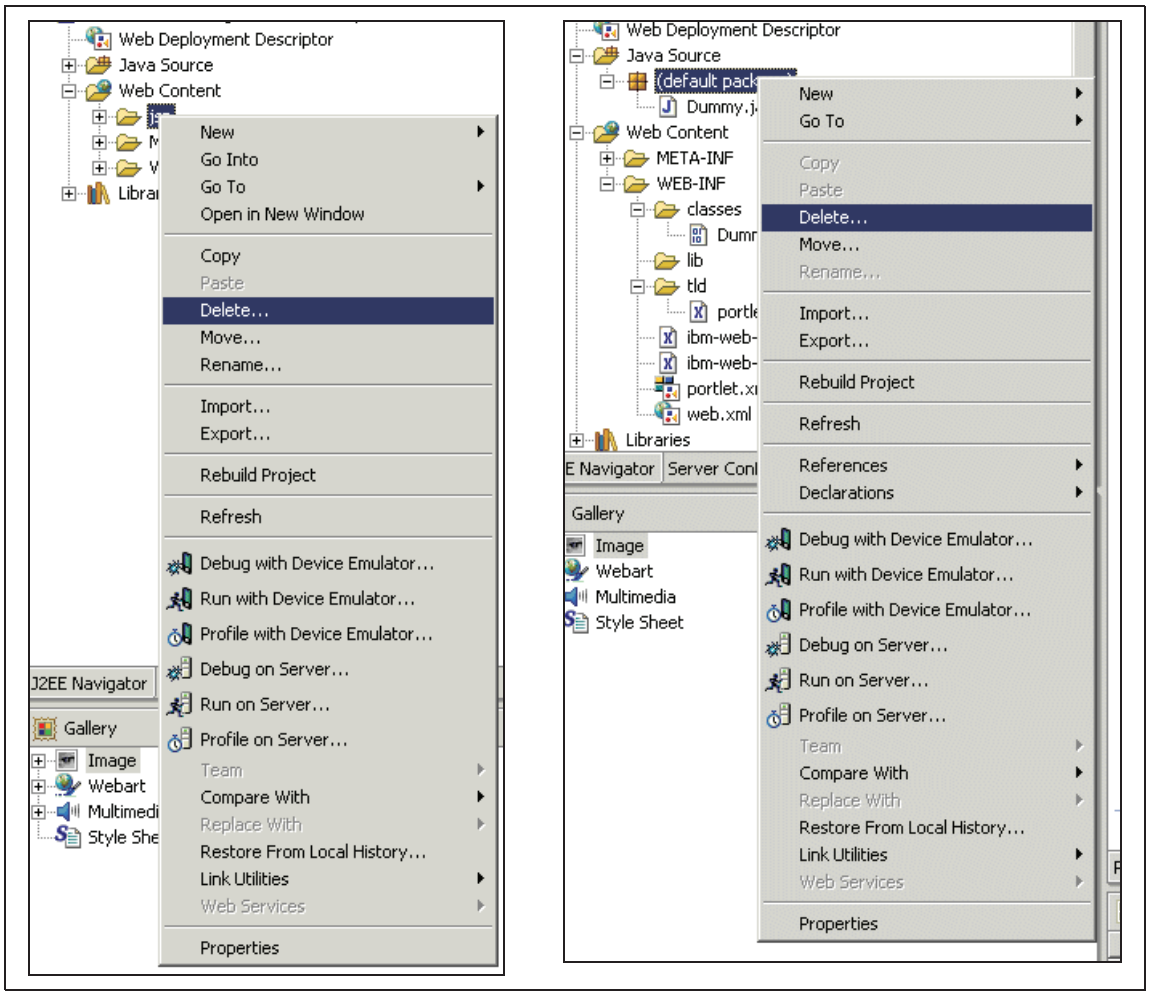


Figure 25-18 Delete JSP directory and the default package in the project

Importing necessary libraries

The portlet libraries directory (WEB-INF/lib) must contain:

- The JAR file ins.jar, which is copied from the directory C:\WebSphere\INS\lib of the WebSphere Everyplace Access 4.3 server.
- The following JAR files, which are extracted from the WAR file C:\WebSphere\INS\install\INSPortlet.war of the Everyplace Access server:
  - `INSPortlet.jar`
  - `insSNMP.jar`
  - `wpsStyleBean.jar`

Import those files into the Project folder WEB-INF/lib. The Java classpath is then updated accordingly.

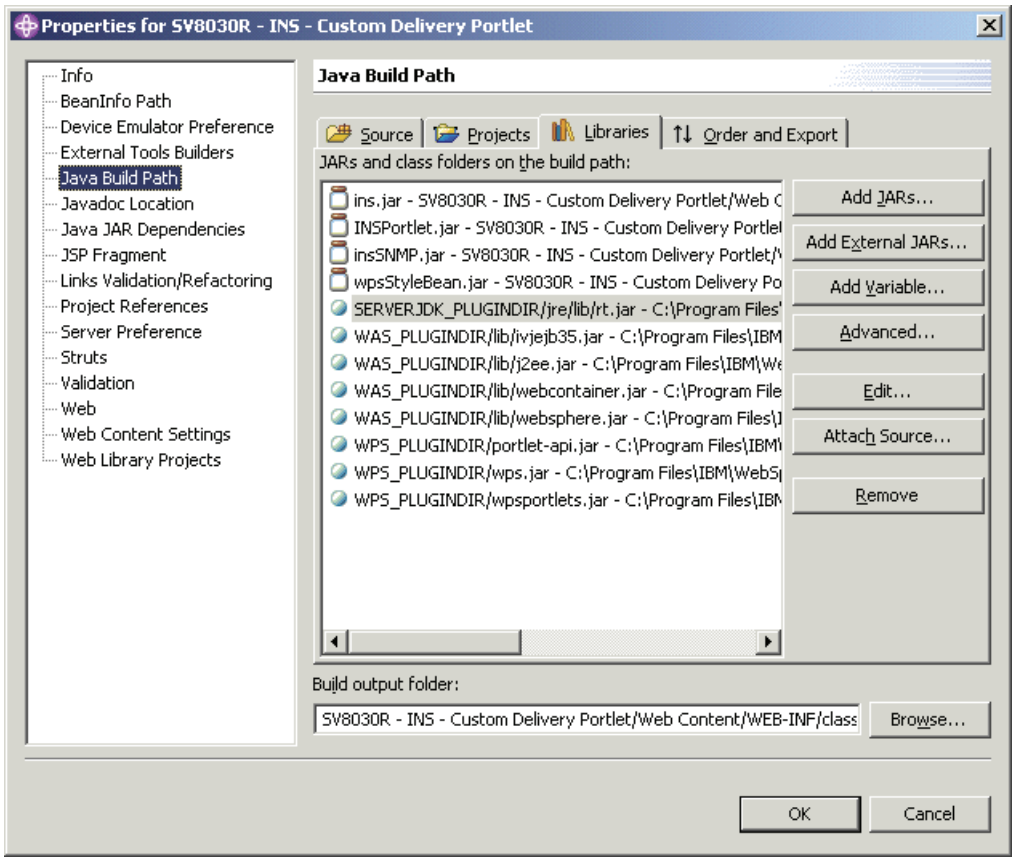


Figure 25-19 Java project classpath

Importing and modifying JSP'

1. Create the following directories by right-clicking **WEB-INF**, then selecting **New -> Folder**:
   - /Web Content/WEB-INF/INS
   - /Web Content/WEB-INF/INS/html
   - /Web Content/WEB-INF/INS/pda
   - /Web Content/WEB-INF/INS/pda/textonly
   - /Web Content/WEB-INF/INS/wml

2. Extracts the following files from the C:\WebSphere\INS\install\INSPortlet.war file of the WebSphere Everyplace Access 4.3 server into your WebSphere Studio Site Developer machine:
   - WEB-INF\INS\html\SmtpDeliveryChannelHelp.jsp
   - WEB-INF\INS\html\DCTemplateAdd.jsp
   - WEB-INF\INS\html\DCTemplateView.jsp
   - WEB-INF\INS\pda\DCTemplateView.jsp
   - WEB-INF\INS\pda\textonly\DCTemplateView.jsp
   - WEB-INF\INS\wml\DCTemplateView.jsp
3. Import the above files into your WebSphere Studio Site Developer project.
4. In your WebSphere Studio Site Developer project, rename the file SmtpDeliveryChannelHelp.jsp to DCTemplateHelp.jsp along with the links to this JSP.
5. Edit the file DCTemplateHelp.jsp and write any help message, for example as follows:

```
<h2 class="help"> My Delivery Channel Help </h2>
<p class="help">When a message is received, it is forwarded to the INS log
file. </p>
```

6. Save the JSP file.

Importing and modifying Java classes

1. Create a new package, for example com.ibm.itso.ins (Figure 25-20).

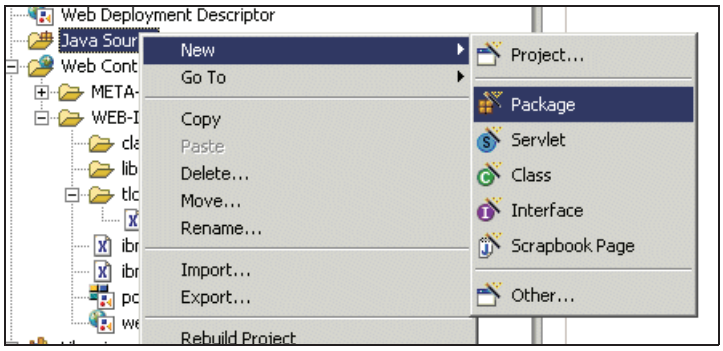


Figure 25-20 Create new package

2. Copy the following files from directory C:\WebSphere\INS\samples\portlets of the WebSphere Everyplace Access 4.3 server into your WebSphere Studio Site Developer machine:
   - SmtpChannelDeliveryController.java
   - SmtpChannelDeliveryController

3. Import those files under newly created package of your project by right-clicking your package, then selecting **Import -> File System**, then selecting the directory where you copied the sample files.

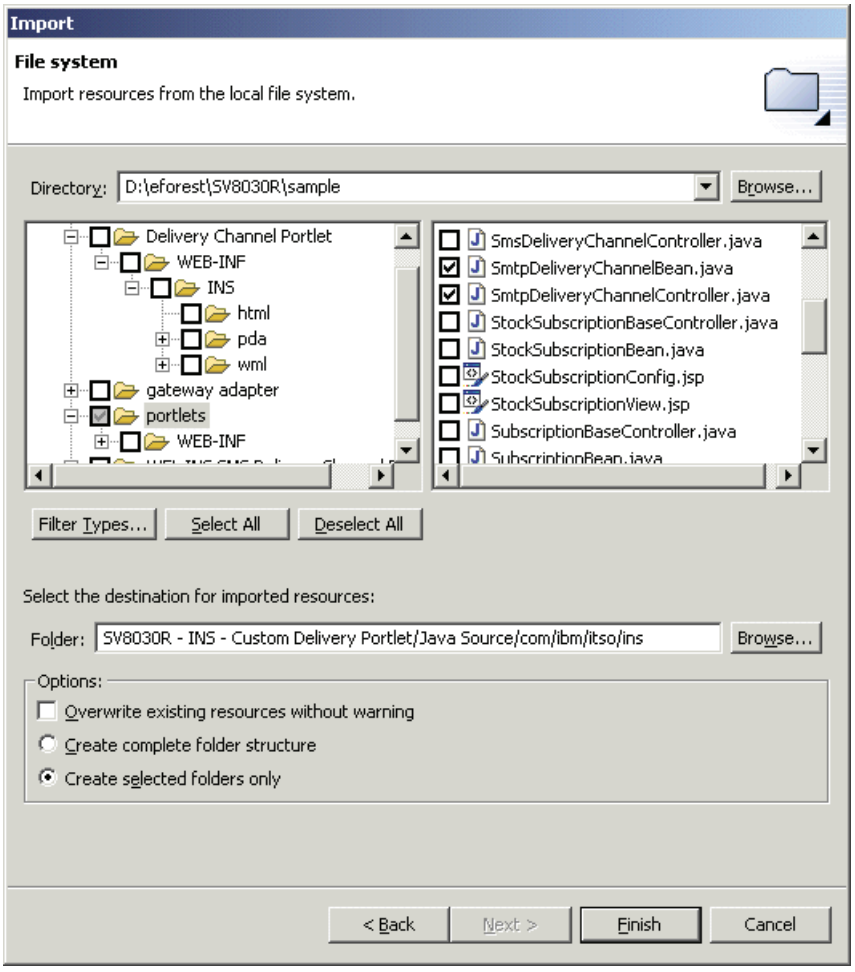


Figure 25-21 Import SMTP sample classes

4. Rename your classes, for example MyDeliveryChannelBean and MyChannelBaseController.

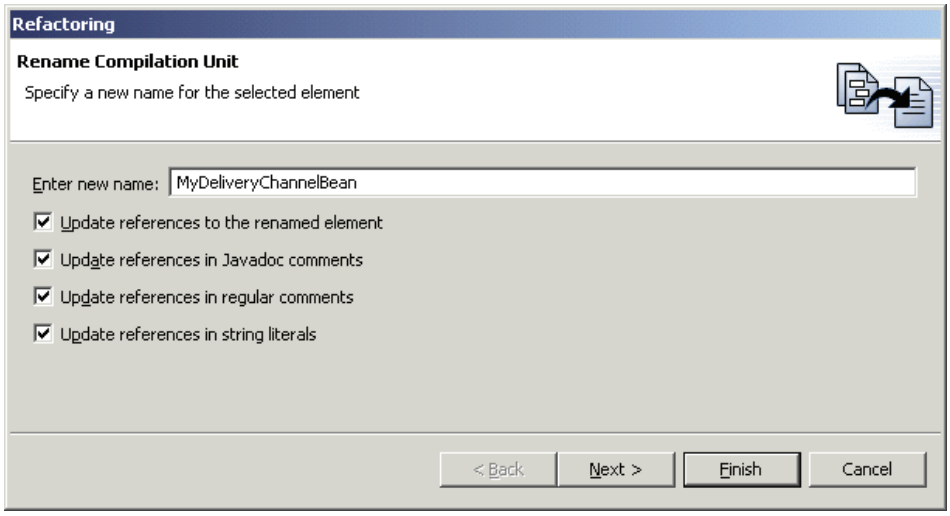


Figure 25-22 Renaming classes

5. On each class:
   - Rename the package name to your own (for example, com.ibm.itso.ins).
   - Import the package com.ibm.pvc.we.ins.portlets.*.
   - Replace any call to /WEB-INF/INS/DCSmtpAdd.jsp by /WEB-INF/INS/DCTemplateAdd.jsp.
   - Replace any call to /WEB-INF/INS/SmtpDeliveryChannelHelp.jsp by /WEB-INF/INS/DCTemplateHelp.jsp.
   - Replace any call to class SmtpDeliveryChannelBean by MyDeliveryChannelBean.
   - Remove any block of code dealing with variable dcAddress, which deals with the delivery channel address that we do not set in this example.
   - Since we removed the delivery channel setting part, we have to replace it with one default setting, such as the following:

     ```
     dcBean.setAddress(dcManager.getFullyQualifiedUserId());
     ```

     which needs to be added in the code after all the error checking has been done, just before deleting the variable dcManager.
   - The above instruction implies the processing of an associated exception, to be added on the catch section after it:

     ```
     catch(GetFailureException gfe)
     {
        if(log.isErrorEnabled())
     ```

```
            log.error("MyChannelBaseController(actionPerformed): exception
        adding DC", gfe);
           request.setAttribute("errorId", "dcBaseErr.error_saving_dc");
           request.setAttribute("dcBean", dcBean);
           request.setAttribute(MODE_KEY, modeParameter);
        }
```

6. On MyDeliveryChannelBean class constructor, delete following lines dealing with LDAP:

```
this.putProperty("ibm-deviceIDType", "3");
this.putProperty("ibm-appProtocolVersion", "none");
//this.putProperty("ibm-undIpServicePort", "25");
//this.putProperty("ibm-undserver", "raleigh.ibm.com");
```

7. Save the class files.

Importing the images

1. Create the following directories by right-clicking **Web Content**, then selecting **New -> Folder**:
   - Web Content/images
   - Web Content/images/INS
2. Extract the following files from the WAR file C:\WebSphere\INS\install\INSPortlet.war of the WebSphere Everyplace Access 4.3 server into your WebSphere Studio Site Developer machine:
   - images\INS\add.gif
   - images\INS\delete.gif
   - images\INS\delete_box.gif
   - images\INS\dot.gif
   - images\INS\error.gif
   - images\INS\header_cancel.gif
   - images\INS\header_divider.gif
   - images\INS\header_ok.gif
   - images\INS\information.gif
3. Import the above files into your WebSphere Studio Site Developer project.

Modifying the Web deployment XML descriptor

1. Open the web.xml file.
2. Select the **Source** tab.
3. Modify the <servlet-class> descriptor to your controller class:

```
<servlet-class>com.ibm.itso.ins.MyChannelBaseController</servlet-class>
```

4. Save the web.xml file.

Exporting the portlet

1. Rebuild the project by right-clicking the project, then selecting **Rebuild project**.
2. Export the project as a WAR file (for example, mydeliverychannel.war) by right-clicking the project, then selecting **Export -> WAR File**.

Deployment

Tip: We suggest you access the target WebSphere Everyplace Access 4.3 server from the Internet browser local to the machine that contains the WAR file to be installed.

1. From your WebSphere Studio Site Developer machine, log on to the WebSphere Everyplace Access 4.3 as an administrator, select **Portal Administration -> Install Portlets -> Install Portlets**, then browse your WAR file.

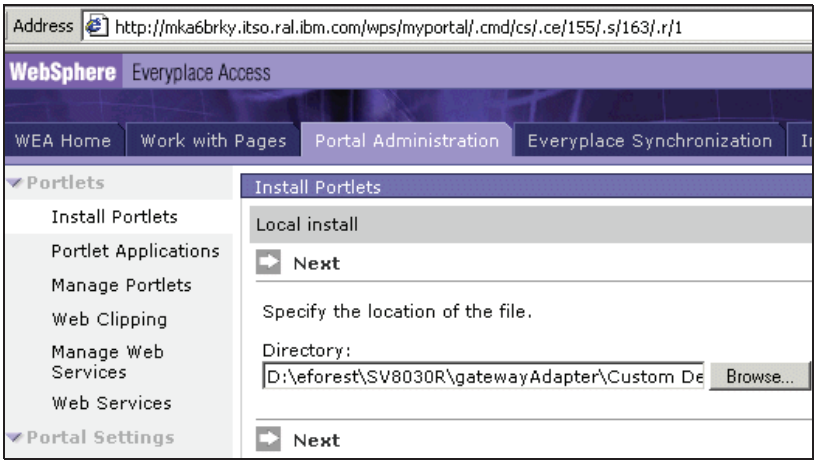


Figure 25-23 Installing my delivery channel portlet

2. Select **Next -> Install**. The portlet should successfully install.

Note: Should you need to update the portlet:

1. Select **Portal Administration -> Portlet Applications -> Manage Portlet Applications**
2. Select the WAR file to be updated
3. Click the **Update** icon
4. Browse your WAR file
5. Select **Next -> Install**

The portlet should successfully install.

6. Select **Portal Administration -> Access Control List,** click **Special groups -> All authenticated users**, click **Name contains**, type `custom delivery portlet`, and click **Go**.
7. Next to your portlet, click **Edit**, then **Save**.

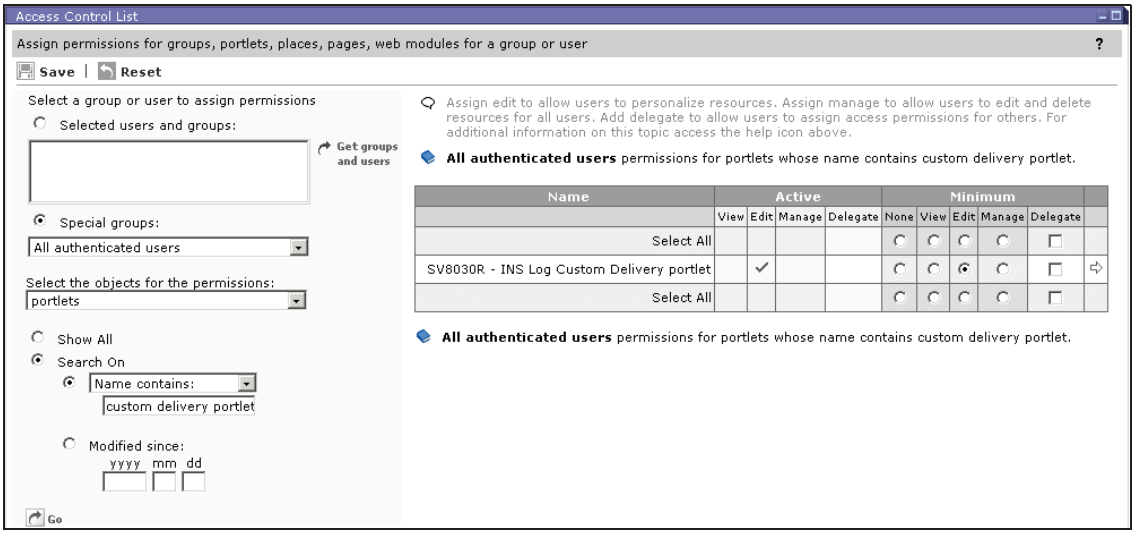


Figure 25-24 Granting Edit access for my delivery channel portlet to all logged users

8. Select **Work with Pages -> Edit Layout,** then select **Intelligent Notification -> My delivery Channels**.

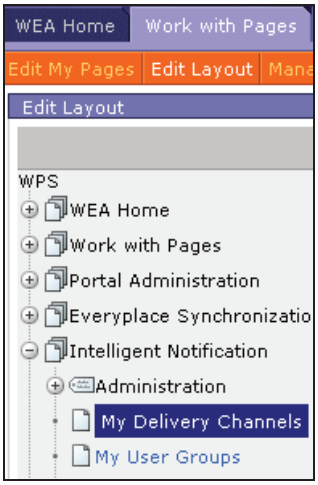


Figure 25-25 Editing My Delivery Channels layout

9. Click **Show layout control**.

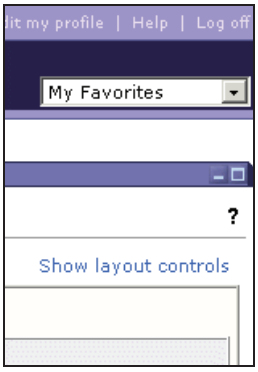


Figure 25-26 Show layout controls

10. Click the **Add column container** icon of the first container.

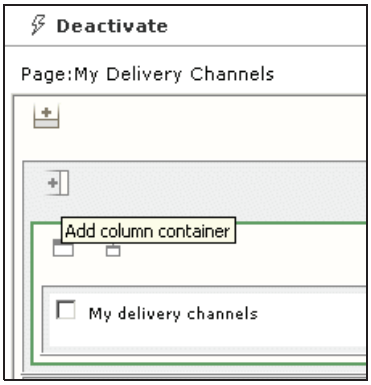


Figure 25-27 Add column container

11. Click the **Add portlet** icon of the right container.

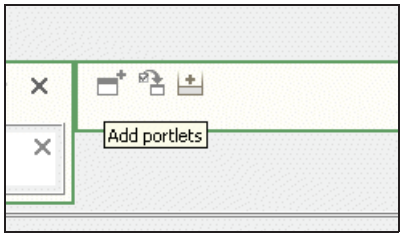


Figure 25-28 Add a portlet

12. Click **Name contains**, type `custom delivery portlet`, and click **Go**. Check your portlet, then click **OK**.

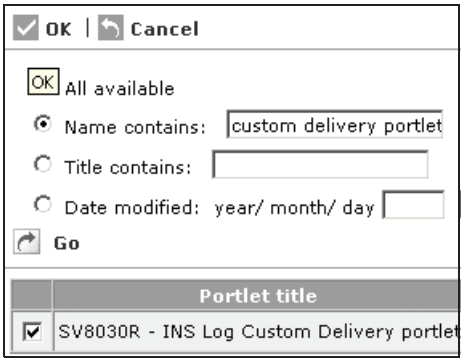


Figure 25-29 Check custom delivery portlet

13. Click the **Set column width** icon (presently not set), and set it to 50% for each of the column container of first row container.

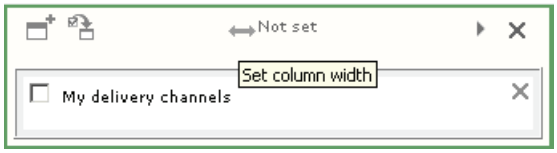


Figure 25-30 Set column width to 50%

14. Activate your page.

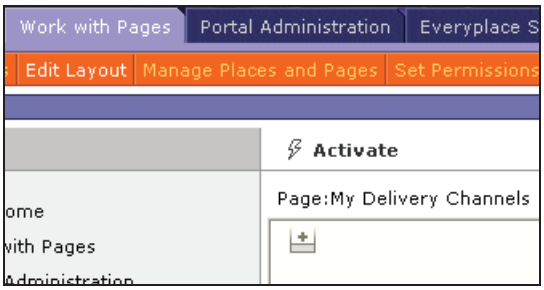


Figure 25-31 Activate the page

15. Select **Intelligent Notification -> My Delivery Channels**. You should see your delivery portlet in the first portlet row.

> **Note:** If your portlet does not show correctly, you may have to log out and in again to see it without error.

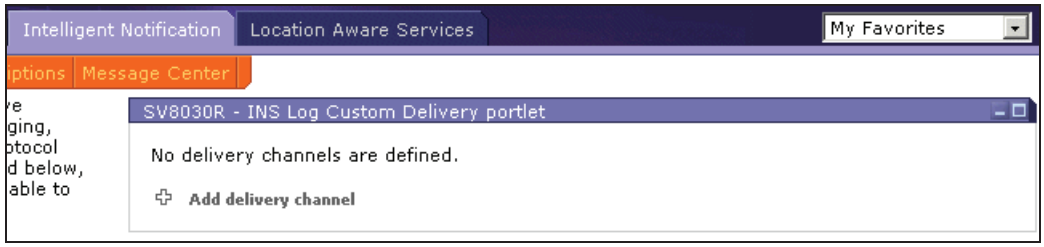


Figure 25-32 Deployed custom delivery channel portlet

25.5 INS development: Custom subscription

In scenario 2, Jim Smith wants to be notified when the weather condition changes, which is not a delivered subscription. Therefore, it requires the development of the following to enable Jim Smith to subscribe to a weather source:

- A custom content adapter that will capture data from the weather source and convert it to a format that the Trigger Manager can read.
- A custom trigger handler associated to this weather subscription.
- A custom subscription portlet to allow a subscription from the user.

25.5.1 Custom content adapter

The custom content adapter converts the information source data into the correct XML format conforming to one of the Trigger Manager DTDs, as follows:

1. Connect to the Trigger Manager.
2. Transform incoming data into an XML document.
3. Translate this XML document into a format suitable for the Trigger Manager, using an XSL stylesheet.
4. Publish the XML document to the Trigger Manager.

The provided content adapters are:

- RSS News content adapter, which retrieves news in RSS format from news Web sites
- Lotus Notes e-mail content adapter
- Microsoft Exchange content adapter
- XML content adapter, used by the news, stock and weather samples

Sample code

Sample code for custom content adapters is found in the following WebSphere Everyplace Access 4.3 server directories:

- C:\WebSphere\INS\samples for the XML content adaptation sample
- C:\WebSphere\INS\samples\email for the Exchange content adaptation sample
- C:\WebSphere\INS\samples\news XML files for the news sample using XML content adaptation
- C:\WebSphere\INS\samples\stock XML files for the stocks sample using XML content adaptation

- C:\WebSphere\INS\samples\weather XML files for the weather sample using XML content adaptation

The sample codes retrieve content data from sample XML files and publish them to the Trigger Manager.

Development

A Java class must be written with a Java classpath that should contain the INS jar file ins.jar, xalan.jar and jlog.jar copied from the directory C:\WebSphere\INS\lib of the WebSphere Everyplace Access 4.3 server.

1. Use the sample code file C:\WebSphere\INS\samples\email\XmailContentAdapter.java
2. Update the sample code to suit your own needs, as follows:
   a. Retrieve the host name and port number of the Trigger Manager.

Example 25-20 Retrieve Trigger Manager host name and port number

```
private String IQproperties = "IQ";

PropertyResourceBundle pr = (PropertyResourceBundle)
PropertyResourceBundle.getBundle(IQproperties);
String IQueuehost = pr.getString("iqSocketClientStub.host");
String IQueueport = pr.getString("iqSocketClientStub.port");
```

   b. Connect to the Trigger Manager.

Example 25-21 Connect to the Trigger Manager

```
manager = TriggerManager.create(IQueuehost+":"+IQueueport);
```

   c. Get data from the content source. This depends on the type of source you are connecting to.
   d. Format the data into XML readable by the Trigger Manager. This can be done in one of the following ways:
      - Transform the incoming data into the appropriate XML format
      - Publish an XML document along with the associated XSL stylesheet that will format the XML accordingly

      The Trigger Manager accepts the XML as a document or as a URL to this document.

      The three Trigger Manager DTDs defining the allowed XML formats are as follows:

Example 25-22 ContentBytesMessage.dtd

```
<!ELEMENT ContentBytesMessage (ContentTopic?,
(ContentProperties|ContentQueryable), (ContentBytes?)) >

<!ELEMENT ContentTopic (topic)*>
<!ELEMENT topic (#PCDATA)>

<!ELEMENT ContentProperties (nameval+)>
<!ELEMENT nameval (#PCDATA)>
<!ATTLIST nameval name CDATA #REQUIRED
                          type (boolean|byte|int|short|long|float|double)
#REQUIRED>

<!ELEMENT ContentQueryable (#PCDATA)>

<!ELEMENT ContentBytes (#PCDATA)>
<!ATTLIST ContentBytes url CDATA #IMPLIED>
```

Example 25-23 ContentMessage.dtd

```
<!ELEMENT ContentMessage (ContentTopic?, (ContentProperties|ContentQueryable))
>

<!ELEMENT ContentTopic (topic*)>
<!ELEMENT topic (#PCDATA)>

<!ELEMENT ContentProperties (nameval+)>
<!ELEMENT nameval (#PCDATA)>
<!ATTLIST nameval name CDATA #REQUIRED
                 type (boolean|byte|int|short|long|float|double|string)
#REQUIRED>

<!ELEMENT ContentQueryable (#PCDATA)>
```

Example 25-24 ContentTextMessage.dtd

```
<!ELEMENT ContentTextMessage (ContentSource?,
(ContentProperties|ContentQueryable), (ContentText?) >

<!ELEMENT ContentSource (name*)>
<!ELEMENT name (#PCDATA)>

<!ELEMENT ContentProperties (nameval+)>
<!ELEMENT nameval (#PCDATA)>
<!ATTLIST ContentProperties name CDATA #REQUIRED
                          type (boolean|byte|int|short|long|float|double)
#REQUIRED>

<!ELEMENT ContentQueryable (#PCDATA)>
```

```
<!ELEMENT ContentText (#PCDATA)>
<!ATTLIST ContentText url CDATA #IMPLIED>
```

e. Publish the XML document to the Trigger Manager via the `publishContent` method, which can be done in five ways:

- publish XML under a content source name:

  ```
  publishContent(String contentSource, String xml)
  ```

- publish XML and associated XSL under a content source name

  ```
  publishContent(String contentSource, String xsl, String xml)
  ```

- publish XML URL under a content source name

  ```
  publishContent(String contentSource, URL urlXml)
  ```

- publish XML and associated XSL URL under a content source name

  ```
  publishContent(String contentSource, URL urlXsl, String xml)
  ```

- publish XML URL and associated XSL URL under a content source name

  ```
  publishContent(String contentSource, URL urlXsl, URL urlXml, String
  xml)
  ```

Example 25-25 Publishing XML content

```
manager.publishContent(contentSource,insXmlDoc);
```

Deployment

The content adapter must be run every time the content source data is fetched. Retrieving this content source data depends on its type. Most likely, the content data will be run periodically to ensure regular update of the content source.

The delivered content adapters give some samples of how to deploy a content adapter. They can be found in C:\WebSphere\INS\apps:

- The startNMAIL.bat file launches the Notes content adapter.
- The startRSS.bat file launches the RSS content adapter.
- The startXMAIL.bat file launches the Exchange content adapter.

25.5.2 Custom trigger handler

The custom trigger handler notifies a given user via the Universal Notification Dispatcher when a match occurs.

A new subscription is added to the Trigger Manager via the SubscriptionManager.addSubscription method.

Content is published to the Trigger Manager via the TriggerManager.publishContent method.

A custom trigger handler is composed of a trigger handler Java class and a trigger handler XML file descriptor.

The provided trigger handlers are:

- RSS News
- Lotus Notes e-mail
- Microsoft Exchange e-mail
- News sample
- Weather sample
- Stocks sample

Sample code

Sample code for the custom trigger handler is found in the following WebSphere Everyplace Access 4.3 server locations:

- Trigger handler Java class sample files:
  - C:\WebSphere\INS\samples\news\NewsHandlerV2.java for news
  - C:\WebSphere\INS\samples\stock\StockHandlerV2.java for stocks
  - C:\WebSphere\INS\samples\weather\WeatherHandlerV2.java for weather
- Trigger handler XML file descriptor sample files:
  - C:\WebSphere\INS\insapps\samples\trigglets.xml for the descriptor
  - C:\WebSphere\INS\insapps\samples\trigglets.dtd for its DTD

Development

A Java class must be written that extends Java class TriggerHandlerV2, with a Java classpath that should contain the INS jar file ins.jar, xlalan.jar and jlog.jar copied from directory C:\WebSphere\INS\lib of the WebSphere Everyplace Access 4.3 server.

1. Use the sample code file C:\WebSphere\INS\samples\weather\WeatherHandlerV2.java.
2. Update the sample code to suit your own needs, as follows:
   a. The main class must extend the TriggerHandlerV2 class.
   b. The doGet(TriggerRequest req, TriggerResponse rsp) method should be implemented to retrieve information from the request parameters (the subscription matching parameters) and put it in the response object.

Example 25-26 doGet() method

```
public void doGet(TriggerRequest req, TriggerResponse resp) throws
TriggerException
{
   Iterator reqKeys;
   reqKeys = req.keys();
   while (reqKeys.hasNext())
   {
      reqParm = (String)reqKeys.next();

      if (reqParm.equals("City"))
         resp.put(reqParm, City);
      else if (reqParm.equals("State"))
         resp.put(reqParm, State);
      else if (reqParm.equals("WeatherReportOption"))
         resp.put(reqParm, WeatherReportOption);
      else if (reqParm.equals("NotificationOption"))
         resp.put(reqParm, NotificationOption);
      else if (reqParm.equals("ContentStorage"))
         resp.put(reqParm, ContentStorage);
      else if (reqParm.equals("DeviceNames"))
         resp.put(reqParm, DeviceNames);
      else if (reqParm.equals("ChannelOrderOption"))
         resp.put(reqParm, multidevice);
   }
}
```

c. The doPut(TriggerRequest req, TriggerResponse rsp) method should be implemented to add a new subscription (trigger) to the Trigger Manager.

   The method is declared.

Example 25-27 doPut() method declaration

```
public void doPut(TriggerRequest req) throws TriggerException
{
```

Then all the subscription parameters are saved.

Example 25-28 doPut() method: subscription parameter saving

```
   reqParm = req.get("userid");// User ID (user@realm)
   if ((reqParm == null) || (reqParm.trim().equals("")))
      throw new TriggerException(messages.getString("USERID_ERR"));
   else
   {  // To and From fields for the notification request
      toUserid = reqParm;
      fromUserid = reqParm;
```

```
   }
   reqParm = req.get("City");
   if ((reqParm == null) || (reqParm.trim().equals("")))
      throw new TriggerException(messages.getString("REQUIRED_ERR_CITY"));
   else
   {
      City = reqParm;
   }
   reqParm = req.get("State");
   if ((reqParm == null) || (reqParm.trim().equals("")))
      throw new TriggerException(messages.getString("REQUIRED_ERR_STATE"));
   else
   {
      State = reqParm;
   }
   reqParm = req.get("WeatherReportOption");
   if ((reqParm == null) || (reqParm.trim().equals("")))
      throw new TriggerException(messages.getString("REQUIRED_ERR_OPTION"));
   else
   {
      WeatherReportOption = reqParm;
   }
   reqParm = req.get("NotificationOption");
   if ((reqParm != null) && !(reqParm.trim().equals(""))) // Default is "once"
   {
      NotificationOption = reqParm;
   }
   reqParm = req.get("ContentStorage");
   if ((reqParm != null) && !(reqParm.trim().equals("")))// Default is "don't
save"
   {
      ContentStorage = reqParm;
   }
   reqParm = req.get("DeviceNames"); // Device(s) (i.e. MAIL, IM, SMS)
   if ((reqParm == null) || (reqParm.trim().equals("")))
      throw new TriggerException(messages.getString("REQUIRED_ERR_DEVICE"));
   else
   {
      DeviceNames = reqParm;
   }
   reqParm = req.get("ChannelOrderOption");// Multi device option ("all" or
"any")
   if ((reqParm != null) && !(reqParm.trim().equals("")))// Default option is
"all"
   {
      multidevice = reqParm;
   }
```

Then the subscription matching selector is built.

Example 25-29 doPut() method: subscription matching selector

```
   // Create the selector statement
   selector = "(cityname = '" + City + "') AND (citystate = '" + State + "')";
   if (WeatherReportOption.equals(current))
      selector = selector + " AND (forecastday IS NULL)";
   else
      selector = selector + " AND (forecastday IS NOT NULL)";
```

Then create, save and activate the subscription via its selector.

Example 25-30 doPut() method: subscription creation

```
try
   {
      sid = subscribe(source, selector);// Create the subscription
      save(); // Save the subscription to the trigger store
      activate(); // Activate the subscription
   }
   catch (IQueueException iqe)
   {
      throw new TriggerException("addSub exception: " + iqe.getMessage());
   }
```

3. The doPost(TriggerRequest req, TriggerResponse rsp) method should be implemented to update an existing subscription, which consists of updating the correct parameter, then updating, then saving the subscription.

Example 25-31 doPost() method

```
public void doPost(TriggerRequest req, TriggerResponse resp) throws
TriggerException
{
   Iterator reqKeys;
   reqKeys = req.keys();
   while (reqKeys.hasNext())
   {
   reqParm = (String)reqKeys.next();
   if (reqParm.equals("City"))
      City = req.get(reqParm);
   else if (reqParm.equals("State"))
      State = req.get(reqParm);
   else if (reqParm.equals("WeatherReportOption"))
      WeatherReportOption = req.get(reqParm);
   else if (reqParm.equals("NotificationOption"))
      NotificationOption = req.get(reqParm);
   else if (reqParm.equals("ContentStorage"))
      ContentStorage = req.get(reqParm);
   else if (reqParm.equals("DeviceNames"))
      DeviceNames = req.get(reqParm);
```

```
   else if (reqParm.equals("ChannelOrderOption"))
      multidevice = req.get(reqParm);
   }

   // Create the selector statement
   selector = "(cityname = '" + City + "') AND (citystate = '" + State + "')";
   if (WeatherReportOption.equals(current))
      selector = selector + " AND (forecastday IS NULL)";
   else
      selector = selector + " AND (forecastday IS NOT NULL)";

   // update and save the subscription
   try
   {
      update(sid, source, selector);
      save();
   }
   catch (IQueueException iqe)
   {
      throw new TriggerException("updSub exception" + iqe.getMessage());
   }
}
```

d. The doDelete(TriggerRequest req) method should be implemented to delete an existing subscription.

Example 25-32 doDelete() method

```
public void doDelete(TriggerRequest req) throws TriggerException
{
   remove();
}
```

e. The handleMatch(TriggerRequest req, TriggerResponse rsp) method should be implemented to build the notification XML when a match occurs.

   The method is declared.

Example 25-33 handleMatch() method declaration

```
public void handleMatch(ContentMessage cmsg)
{
```

Then the XML document is built.

Example 25-34 handleMatch() method: Build XML document

```
String notificationString;
StringBuffer message = new StringBuffer();
```

```
message.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
message.append("<message>\n");
message.append("<to>" + toUserid + "</to>\n");
message.append("<from>" + toUserid + "</from>\n");
message.append("<subject>WEATHER</subject>\n");
message.append("<text>" + cmsg.getString("cityname") + " " +
cmsg.getString("citystate") + " ");
message.append("Temperature: " + cmsg.getString("temperature") + "\n");
message.append("Wind: " + cmsg.getString("wind") + "\n");
message.append("Humidity: " + cmsg.getString("humidity") + "\n");
message.append("</text>\n");
message.append("</message>");

notificationString = message.toString();
```

Then the subscription is kept or removed depending on its settings.

Example 25-35 handleMatch() method: keep or remove subscription

```
//Set the Trigger Option based on user input
if (NotificationOption.equals("once"))
   remove();
else if (NotificationOption.equals("always and persist"))
{
   try
   {
      save();
   }
   catch (IQueueException iqe)
   {
      System.out.println("saveSub exception" + iqe.getMessage());
   }
}
```

Then the notification is submitted to all the subscribed devices.

Example 25-36 handleMatch() method: Notification submission

```
// Submit the notification
ANotificationService ns = getNotificationService();
DeliveryOptionsExtended dopts = new DeliveryOptionsExtended();
StringTokenizer tokens = new StringTokenizer(DeviceNames, "+");

while ( tokens.hasMoreTokens() )
{
   String token = tokens.nextToken().trim();
   dopts.addDevice(token);
}

dopts.setPriority(priority);
```

```
dopts.setMultiDevices(multidevice);

try
{
   int rc = ns.sendMessage(notificationString, dopts, null);
}
catch (Exception e)
{
   e.printStackTrace();
}
```

Deployment

The trigger handler must be deployed as follows

1. On the WebSphere Everyplace Access 4.3 server, go to the C:\WebSphere\INS\ins\apps directory.
2. Create your own subdirectory.
3. Create the ./classes subdirectory to your directory and place your custom trigger handler classes in this subdirectory.
4. Compose the triglets.xml file (see Example 25-38 on page 910) and place it into your root subdirectory, along with its DTD (shown in Example 25-39 on page 911).

   This file allows the Trigger Manager to locate the trigger handler class based on the URN passed with the SubscriptionManager.addSubscription and TriggerManager.addTrigger methods.

Example 25-37 trigglets.xml file

```
<?xml version='1.0'?>
<!DOCTYPE triglet-app SYSTEM "./triglets.dtd">
<triglet-app>
   <triglet>
      <triglet-name>WeatherHandler Sample Version 2.0</triglet-name>
      <description>Weather subscription Sample handler</description>
      <triglet-class>WeatherHandlerV2</triglet-class>
      <init-param>
         <param-name>SubscriptionsPerTriglet</param-name>
         <param-value>1</param-value>
      </init-param>
      <init-param>
         <param-name>TrigletTimeToLiveMillis</param-name>
         <param-value>1209600000</param-value>
      </init-param>
   </triglet>

   <triglet-mapping>
```

```
      <triglet-name>WeatherHandler Sample Version 2.0</triglet-name>
      <url-pattern>/WeatherHandler</url-pattern>
   </triglet-mapping>
</triglet-app>
```

Example 25-38 trigglets.dtd file

```
<!ELEMENT triglet-app (description?, context-param*, triglet*,
triglet-mapping*) >

<!ELEMENT description (#PCDATA)>

<!ELEMENT context-param (param-name, param-value, description?)>

<!ELEMENT init-param (param-name, param-value)>
<!ELEMENT param-name (#PCDATA)>
<!ELEMENT param-value (#PCDATA)>


<!ELEMENT triglet (triglet-name, description?, triglet-class, init-param*)>
<!ELEMENT triglet-name (#PCDATA)>
<!ELEMENT triglet-class (#PCDATA)>

<!ELEMENT triglet-mapping (triglet-name, url-pattern)>
<!ELEMENT url-pattern (#PCDATA)>
```

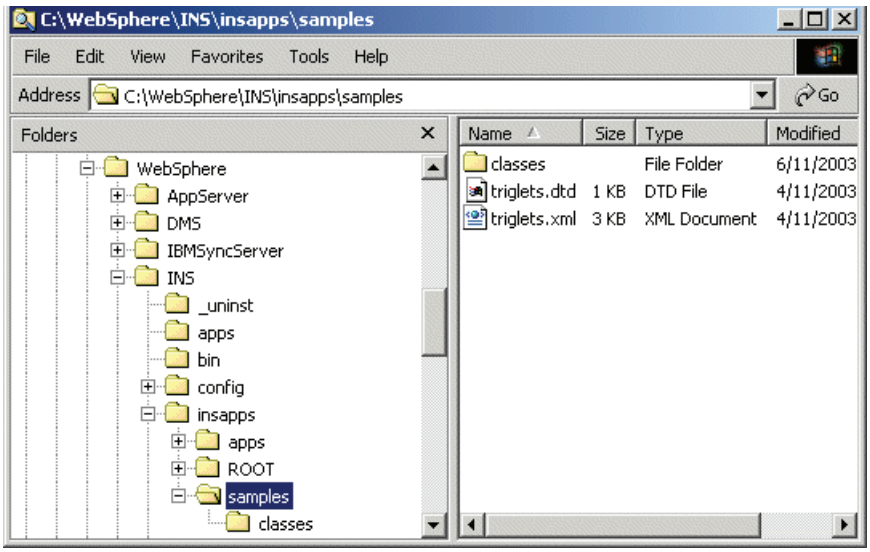


Figure 25-33 Deployment directory root content sample

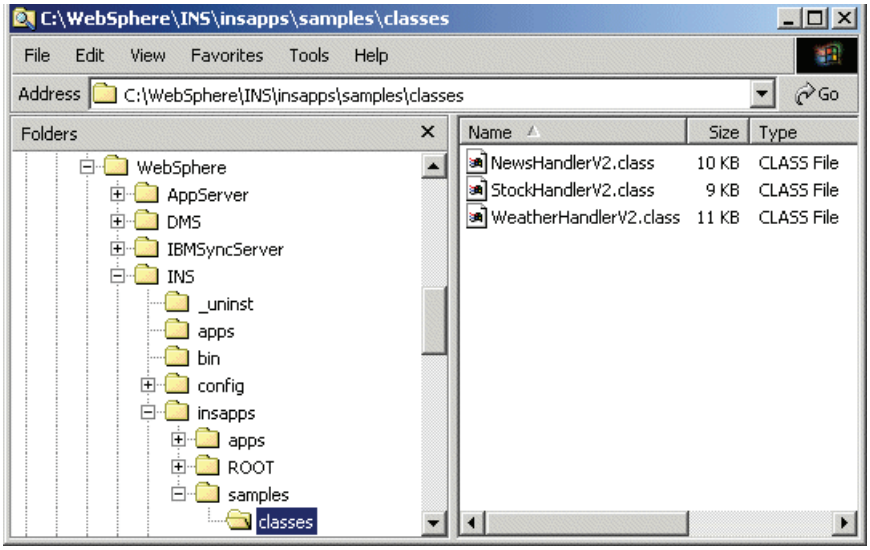


Figure 25-34 Deployment directory classes content sample

25.5.3 Custom subscription portlet

The custom subscription portlet allows the user to subscribe to content source notifications.

The user should be able to add, edit and delete a subscription. One subscription should define:

- The content matching criteria
- Whether notification should be sent once or always
- The selected delivery channels
- Whether all or first successful delivery channel are notified

A new subscription is added to the Trigger Manager via the SubscriptionManager.addSubscription method.

A custom subscription portlet is composed of:

- A controller Java class extending SubscriptionBaseController and implementing ActionListener
- A bean Java class extending Serializable and implementing SubscriptionBean
- A configuration JSP used for adding and editing subscriptions
- A subscription view JSP used to list the current subscriptions

- A subscription manager JSP used to manage the subscriptions

The provided subscription portlets are:

- RSS News
- Lotus Notes e-mail
- Microsoft Exchange e-mail
- News sample
- Weather sample
- Stocks sample

Sample code

Sample code for the custom subscription portlet is found in the WebSphere Everyplace Access 4.3 server directory C:\WebSphere\INS\samples\portlets. The following files are included:

- *SubscriptionBaseController.java:

  Logic handling the subscription portlets for generic, news, stock and weather samples.

- *SubscriptionBean.java:

  Subscription data storage bean for generic, news, stock and weather samples.

- *SubscriptionConfig.jsp:

  Subscription edition JSP for generic, news, stock and weather samples.

- *SubscriptionView.jsp:

  Subscription view JSP for generic, news, stock and weather samples.

Development

We suggest you use WebSphere Studio Site Developer along with the WebSphere Everyplace Access 4.3 Portal Toolkit to develop the custom subscription portlet.

We will describe this portlet by using the weather sample.

Refer to 25.4.2, “Delivery channel portlet” on page 865 for details on using WebSphere Studio Site Developer.

Importing sample into WebSphere Studio Site Developer

We suggest you use WebSphere Studio Site Developer on a machine other than the WebSphere Everyplace Access 4.3 server.

1. Open WebSphere Studio Site Developer.
2. Create a new Java JSP portlet project.

3. Delete the directory Web-content/jsp and Java Source/(default package).
4. The portlet libraries directory (WEB-INF/lib) must contain:
   - The following JAR file copied from directory C:\WebSphere\INS\lib of the WebSphere Everyplace Access 4.3 server:
     - ins.jar
     - jlog.jar
   - The following JAR files extracted from the WAR file C:\WebSphere\INS\install\INSPortlet.war of the WebSphere Everyplace Access 4.3 server:
     - INSPortlet.jar
     - insSNMP.jar

   Import these files into the Project folder WEB-INF/lib. The Java classpath is then updated accordingly.
5. Create the following directories by right-clicking **WEB-INF**, then selecting **New -> Folder**:
   - Web Content/WEB-INF/INS
   - Web Content/WEB-INF/INS/html
   - Web Content/WEB-INF/INS/pda
   - Web Content/WEB-INF/INS/pda/textonly
   - Web Content/WEB-INF/INS/wml
6. Extract the following files from the WAR file C:\WebSphere\INS\install\INSPortlet.war of the WebSphere Everyplace Access 4.3 server into your WebSphere Studio Site Developer machine.:
   - WEB-INF\INS\html\DeliveryChannelOrdering.jsp
   - WEB-INF\INS\html\WeatherSubscriptionConfig.jsp
   - WEB-INF\INS\html\WeatherSubscriptionHelp.jsp
   - WEB-INF\INS\html\WeatherSubscriptionView.jsp
   - WEB-INF\INS\pda\DeliveryChannelOrdering.jsp
   - WEB-INF\INS\pda\WeatherSubscriptionConfig.jsp
   - WEB-INF\INS\pda\WeatherSubscriptionView.jsp
   - WEB-INF\INS\pda\textonly\DeliveryChannelOrdering.jsp
   - WEB-INF\INS\pda\textonly\WeatherSubscriptionConfig.jsp
   - WEB-INF\INS\pda\textonly\WeatherSubscriptionView.jsp
   - WEB-INF\INS\wml\DeliveryChannelOrdering.jsp
   - WEB-INF\INS\wml\WeatherSubscriptionConfig.jsp
   - WEB-INF\INS\wml\WeatherSubscriptionView.jsp
7. Import the above files into your WebSphere Studio Site Developer project.
8. Create a new package, com.ibm.pvc.we.ins.portlets.

9. Copy the following files from directory C:\WebSphere\INS\samples\portlets of the WebSphere Everyplace Access 4.3 server into your WebSphere Studio Site Developer machine:
   - WeatherSubscriptionBaseController.java
   - WeatherSubscriptionBean.java
10. Import them under the newly created package.
11. Rename them to:
   - MyWeatherSubscriptionBaseController.java
   - MyWeatherSubscriptionBean.java
12. Edit MyWeatherSubscriptionBaseController.java, adding an INS Data Access instantiation instruction in the init method as shown in Example 25-40 on page 912.

Example 25-39 Add INSDataAccess instantiation

```
public void init(PortletConfig portletConfig) throws UnavailableException
{
   super.init( portletConfig );
   INSDataAccess.init(log);
```

13. Create the following directories:
   - Web Content/images
   - Web Content/images/INS
14. Extract the following files from the WAR file C:\WebSphere\INS\install\INSPortlet.war of the WebSphere Everyplace Access 4.3 server into your WebSphere Studio Site Developer machine:
   - images\INS\add.gif
   - images\INS\delete_box.gif
   - images\INS\divider2.gif
   - images\INS\dot.gif
   - images\INS\edit.gif
   - images\INS\header_cancel.gif
   - images\INS\header_ok.gif
   - images\INS\information.gif
   - images\INS\msg_warning.gif
   - images\INS\INSbanner.jpg
15. Import the above files into your WebSphere Studio Site Developer project.
16. Edit the web.xml file:
   - Change the Web application ID to yours
   - Change the servlet and display names to yours
   - Change the <servlet-class> descriptor to your controller class

– Modify the servlet mapping accordingly
– Remove the <welcome-file-list> tag
– Save the web.xml file

The web.xml file should then look like Example 25-41 on page 912.

Example 25-40 servlet descriptor web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.2//EN" "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app id="MySubsciptions">
   <display-name>SV8030R - INS Custom Subscription</display-name>
   <servlet id="Servlet_1">
      <servlet-name>My Wheather Subscriptions</servlet-name>
      <display-name>My Wheather Subscriptions</display-name>

<servlet-class>com.ibm.pvc.we.ins.portlets.MyWeatherSubscriptionBaseController<
/servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>My Wheather Subscriptions</servlet-name>
      <url-pattern>/My_Wheather_Subscriptions/*</url-pattern>
   </servlet-mapping>
</web-app>
```

17. Update the sample code to suit your own needs.
18. Rebuild the project, and export it as a WAR file.
19. Deploy the portlet into the WebSphere Everyplace Access 4.3 server.

Subscription base controller Java class

This class controls the portlet itself. It extends SubscriptionBaseController.

Example 25-41 Subscription base controller class declaration

```
public class WeatherSubscriptionBaseController extends
SubscriptionBaseController
{
```

1. The init method calls its superclass, then initializes all the instance variables.

Example 25-42 Subscription base controller init() method

```
public void init(PortletConfig portletConfig) throws UnavailableException
{
   super.init( portletConfig );
   INSDataAccess.init(log);
   contentSource = "weather";
   triggerHandlerURN = "/samples/WeatherHandler";
```

```
   subKeys = new String[]{MySubscriptionBean.KEY_NOTIFICATION_OPTION,
      SubscriptionBean.KEY_CONTENT_STORAGE,
      SubscriptionBean.KEY_DEVICE_NAMES,
      SubscriptionBean.KEY_CHANNEL_ORDER_OPTION,
      WeatherSubscriptionBean.CITY,
      WeatherSubscriptionBean.STATE,
      WeatherSubscriptionBean.WEATHER_REPORT_OPTION
      };

   config_jsp = jspBaseDir + "/WeatherSubscriptionConfig.jsp";
   help_jsp = jspBaseDir + "/WeatherSubscriptionHelp.jsp";
   view_jsp = jspBaseDir + "/WeatherSubscriptionView.jsp";

   if (log.isDebugEnabled())
   {
      log.debug("MyWeatherSubscriptionBaseController(init):
         triggerHandlerURN= " + triggerHandlerURN);
      for (int i=0;i<subKeys.length; i++)
      {
         log.debug("MyWeatherSubscriptionBaseController(init); subKeys = "
            + i + " " + subKeys[i]);
      }
   }
}
```

2. The inherited method createSubscriptionBeanFromRequestData() is changed according to the specifications of the weather content adapter as follows:

   a. Declare the method (see Example 25-44 on page 914).

Example 25-43 create SubscriptionBeanFromRequestData() method declaration

```
public void createSubscriptionBeanFromRequestData(PortletRequest request,
   SubscriptionBean newSub)
      throws PortletException
{
   if (log.isDebugEnabled())
   {
      log.debug(contentSource +
         "SubscriptionBaseController(entering
createSubscriptionBeanFromRequestData method");
   }
```

   b. Get the specific content parameters (see Example 25-45 on page 914).

Example 25-44 Get specific content parameters

```
   String city         = request.getParameter("city");
   String state        = request.getParameter("state");
   String reportOption = "forecast";
```

```
   reportOption        = request.getParameter("reportOption");
```

c. Get the subscription type (see Example 25-46 on page 916).

Example 25-45 Get subscription type

```
   String notificationOption = "once";
   notificationOption        = request.getParameter("notificationOption");
   String channelOrderOption = request.getParameter("channelOrderOption");

   String contentStorageOption = "don't save";
```

d. Get the delivery channels declared for this subscription (see Example 25-46 on page 916).

Example 25-46 Get associated delivery channels

```
   String selectedDCNames = "";
   selectedDCNames = getSelectedDCNamesFromRequest(request);
```

e. Build the notification (see Example 25-48 on page 917).

Example 25-47 Build notification

```
   if (   notificationOption != null && contentStorageOption != null
       && city               != null && state                != null
       && reportOption       != null && selectedDCNames      != null
       && channelOrderOption != null)
   {
      newSub.setNotificationOption(notificationOption);
      newSub.setContentStorage(contentStorageOption);
      newSub.setDeviceNames(selectedDCNames);
      newSub.putProperty(SubscriptionBean.KEY_CHANNEL_ORDER_OPTION,
         channelOrderOption);
      newSub.putProperty(MyWeatherSubscriptionBean.CITY,city);
      newSub.putProperty(MyWeatherSubscriptionBean.STATE,state);
      newSub.putProperty(MyWeatherSubscriptionBean.WEATHER_REPORT_OPTION,
         reportOption);
   }
}
```

3. All other methods listed below are not changed from the SubscriptionBaseController generic Java class for our example:
   - includeJSP
   - doView

     Called when the portlet is in view mode.

- setSubIndexBeansToSession

  Called by doView for the default view mode to instantiate the bean.

- filterChannels

  Filters out server-initiated action channels for normal subscriptions.

- doAdd

  Called by doView when adding a subscription to get the delivery channels and call the subscription configuration JSP.

- doModify

  Called by doView when modifying a subscription to retrieve the subscription, get the delivery channels, and call the subscription configuration JSP.

- doChannelOrdering

  Called by doView when modifying the order of the delivery channels to get the delivery channels and call the channel ordering JSP.

- doHelp

  Called when the portlet is in help mode.

- actionPerformed

  Called when an action is performed on the portlet to call doChannelOrderAction, doMoveUpChannelAction, or doMoveDownChannelAction accordingly. You may want to override this method when implementing more actions than the default ones.

- doChannelOrderAction

  Called by actionPerformed when ordering of the channels was requested.

- doMoveUpChannelAction

  Called by actionPerformed when channel has to move up.

- doMoveDownChannelAction

  Called by actionPerformed when channel has to move down.

- doAddSubscriptionAction

  Called by the actionPerformed method when an add action is performed to instantiate a bean and create a subscription.

- doModifySubscriptionAction

  Called by the actionPerformed method when a modify action is performed to instantiate a bean and modify the subscription.

– doDeleteSubscriptionAction

  Called by the actionPerformed method when a modify action is performed to instantiate a bean and delete the subscription.

– getSelectedDCNamesFromRequest

  Called by createSubscriptionBeanFromRequestData to get the delivery channels associated to the subscription.

– addToDCBufferIfNeeded

  Called by getSelectedDCNamesFromRequest to add more channels.

– shouldSwitchToOrderingMode

  Called by doAdd and doModify to decide whether channels should be re-ordered.

– saveTemporateDataInBean

  Called by shouldSwitchToOrderingMode to buffer user input data.

– printBeanPropertiesForDebugging

  Debug purpose method.

Subscription bean Java class

This class is the Model part of the MVC architecture of the subscription portlet. It extends the SubscriptionBean class, which contains the variables, getters and setters of the core subscription properties:

- Notification option
- Content storage option
- Channel order option
- Preferred delivery channels
- Subscription name
- Content source

The SubscriptionBean class may be extended to add any specific subscription properties and their associated getters and setters.

Subscription view JSP

This JSP allows the user to view existing subscriptions, add new ones, and modify existing ones. The main instructions of this page, shown in the following examples from the sample WeatherSubscriptionView.jsp, are as follows:

1. Get the current subscriptions.

Example 25-48 Get current subscriptions from portlet session

```
<%
   SubscriptionIndexBean[] indexObjects =
```

```
      (SubscriptionIndexBean[])
        portletRequest.getPortletSession().getAttribute
          (SubscriptionBaseController.SUB_INDEX_BEANS);
%>
```

2. Prompt the user to add a subscription if none exist.

Example 25-49 Display prompt to add subscription when none exist

```
<!--if there is no existing subscription  -->
<% if (indexObjects == null || indexObjects.length <= 0)  { %>
   <TR BGCOLOR="#DEF2FD">
      <td>
         <SPAN class="wpsPortletHead">
            <portletAPI:text
              key="Sub_No_Subcription_Currently"
              bundle="nls.INS.PortletMessages" />
         </SPAN>
      </TD>
   </TR>
   <TR>
      <TD>
         <IMG
           SRC='<%=portletResponse.encodeURL("/images/INS/information.gif")%>'
           border="0"
           alt="<portletAPI:text
           key="InfoImage"
           bundle="nls.INS.PortletMessages" />" >
         <SPAN class="wpsPortletText">
            <portletAPI:text
              key="Sub_add_NewSubscription_Prompt"
              bundle="nls.INS.PortletMessages" />
         </SPAN>
      </TD>
   </TR>

<!--if there are already subscriptions    -->
<% } else {%>
   <TR BGCOLOR="#DEF2FD">
      <td>
         <SPAN class="wpsPortletHead">
            <portletAPI:text
              key="Sub_citystate"
              bundle="nls.INS.PortletMessages" />
         </SPAN>
      </td>
      <td>
         <SPAN class="wpsPortletHead">
            <portletAPI:text
```

```
                key="Sub_condition"
                bundle="nls.INS.PortletMessages" />
            </SPAN>
        </td>
    </TR>
```

3. For each existing subscription bean, display its name as a modify link, which changes the portlet into MOFIDY_MODE when clicked.

Example 25-50 Display existing instructions with modify link

```
<% boolean toggle = true;
   for (int i = 0; i < indexObjects.length;i++, toggle = !toggle)
   {
      SubscriptionIndexBean indexBean =
(SubscriptionIndexBean)indexObjects[i];
      SubscriptionBean subBean =
(SubscriptionBean)indexBean.getSubscriptionBean();

      PortletURI modifyURI = portletResponse.createURI();
      modifyURI.addParameter(SubscriptionBaseController.MODIFY_MODE, "true");
      modifyURI.addParameter(SubscriptionBaseController.SUB_BEAN_INDEX,
String.valueOf(indexBean.getIndex()));

      StringBuffer sb = new StringBuffer();
      String city = (String)subBean.getProperty(WeatherSubscriptionBean.CITY);
      String state =
(String)subBean.getProperty(WeatherSubscriptionBean.STATE);
      sb.append(city);
      sb.append(",");
      sb.append(state);
      String option =
(String)subBean.getProperty(WeatherSubscriptionBean.WEATHER_REPORT_OPTION);
%>

<TR CLASS="<%=toggle? "wpsTableRow": "wpsTableShdRow"%>">
   <TD>
      <A style="text-decoration: none;"
        HREF="<%=modifyURI%>">
        <%=sb.toString()%>
      </A>
   </TD>

.... some other JSP instructions ....

<%
   }
%>
```

4. Optionally retrieve other subscription parameters to further identify the subscription to the user.

Example 25-51 Optional parameter retrieval

```
<%
   if ("currentCondition".equalsIgnoreCase(option))
   {
%>
<portletAPI:text key="Sub_current_condition" bundle="nls.INS.PortletMessages"
/>
<%
   }
   else
   {
%>
<portletAPI:text key="Sub_forecast" bundle="nls.INS.PortletMessages" />
<%
   }
%>
```

5. Create an `add` link that switches the portlet in the ADD_MODE when clicked.

Example 25-52 Add link

```
<%
   PortletURI addURI = portletResponse.createURI();
   addURI.addParameter(SubscriptionBaseController.ADD_MODE, "true");
%>

.... some other JSP instructions ....

<A style="text-decoration: none;" href="<%= addURI%>">
   <IMG SRC='<%=portletResponse.encodeURL("/images/INS/add.gif")%>'
     title="add"
     alt="<portletAPI:text key="Sub_add" bundle="nls.INS.PortletMessages" /> "
     border="0"/>
    
   <SPAN class="wpsDialogIconText">
      <portletAPI:text key="Sub_add" bundle="nls.INS.PortletMessages" />
   </SPAN>
</A>
```

Subscription configuration JSP

This JSP allows the user to edit subscription parameters, and is displayed when the subscription needs to be edited. The main instructions of this page, shown in the following from the sample WeatherSubscriptionView.jsp, are as follows:

1. Declare javascript variables for all the subscription parameters

Example 25-53 Subscription parameter variables

```
<%
   String notificationOption = "";
   String notifyOnce= "checked";
   String notifyAlways= " ";
   String channelOrderOption = "";
   String orderChannel ="";
   String notOrderChannel = "checked";

   String contentStorageOption = "";
   String saveContent = " ";
   String notSaveContent = "checked";

   String city = "";
   String state = "";
   String reportOption = "";
   String currentCondition = " ";
   String forecast = "checked";

   String addFailed    = (String)portletRequest.getAttribute("add_failed");
   String modifyFailed = (String)portletRequest.getAttribute("modify_failed");
   String deleteFailed = (String)portletRequest.getAttribute("delete_failed");
   String inputError   = (String)portletRequest.getAttribute("input_error");
   String serverError  = (String)portletRequest.getAttribute("server_error");

   Locale aLocale = (Locale) portletRequest.getLocale();
   String lan = aLocale.getLanguage();
   .....
```

2. Get the available delivery channels.

Example 25-54 Get available delivery channels

```
   DeliveryChannelBean[] dcBeans =
     (DeliveryChannelBean[])
       portletRequest.getPortletSession().getAttribute("allDCBeans");


   if (portletRequest.getPortletSession().getAttribute
      ("savedOrderedAllDCBeans") != null)
   {
      dcBeans =
       (DeliveryChannelBean[])
       portletRequest.getPortletSession().getAttribute
       ("savedOrderedAllDCBeans");
   }
```

3. Get the selected subscription bean.

Example 25-55 Get selected subscription bean

```
SubscriptionBean subBean =
  (SubscriptionBean)portletRequest.getAttribute("selectedSubBean");
```

4. When there are existing subscriptions, populate the bean with their parameters.

Example 25-56 Populate selected subscription bean if not empty

```
if (subBean != null)
{
   notificationOption = subBean.getNotificationOption();
   contentStorageOption= subBean.getContentStorage();
   .... some other subscription parameters ....
```

5. Create a `Save` and a `Cancel` button.

Example 25-57 Create save and cancel buttons

```
<% ....
   PortletURI cancelURI = portletResponse.createURI();
   PortletURI deleteURI = portletResponse.createURI();
   PortletURI saveURI = (PortletURI)portletRequest.getAttribute("saveURI");
   ....
%>

.... some other JSP instructions ....

<input type="image"
     name="<portletAPI:encodeNamespace value='OK'/>"
     value="<portletAPI:text key="Sub_OK" bundle="nls.INS.PortletMessages" />"
     src= "<%=portletResponse.encodeURL("/images/INS/header_ok.gif")%>"
     align="absmiddle"
     alt="<portletAPI:text key="Sub_OK" bundle="nls.INS.PortletMessages" />"
     border="0"/>
   <a style="text-decoration: none;"
     href="javascript:document.forms.<portletAPI:encodeNamespace
       value='WeatherSubscriptionConfig'/>.submit()">
      <span class="wpsTaskIconText">
         <portletAPI:text
           key="Sub_OK" bundle="nls.INS.PortletMessages" />
      </span>
   </a>
   <img src="<%= portletResponse.encodeURL("/images/INS/divider2.gif")%>"
     align="absmiddle" border="0" alt=""/>
   <a style="text-decoration: none;" href="<%= cancelURI %>">
      <img
        src="<%=
          portletResponse.encodeURL("/images/INS/header_cancel.gif") %>"
```

```
        align="absmiddle" border="0"
        alt="<portletAPI:text key="Sub_Cancel"
        bundle="nls.INS.PortletMessages"/>"/>
      <span class="wpsTaskIconText">
         <portletAPI:text
           key="Sub_Cancel" bundle="nls.INS.PortletMessages"/>
      </span>
   </a>
```

6. When there are existing subscriptions, create a Delete and a Save URI button.

Example 25-58 Create delete and save buttons for existing subscriptions

```
<%
   ....
   String subBeanIndex =
     (String)portletRequest.getAttribute
     (SubscriptionBaseController.SUB_BEAN_INDEX);

   if (subBeanIndex != null)
   {
      saveURI.addParameter
        (SubscriptionBaseController.SUB_BEAN_INDEX, subBeanIndex);
      deleteURI.addParameter
        (SubscriptionBaseController.SUB_BEAN_INDEX, subBeanIndex);
   }
   ....
%>

.... some other JSP instructions ....

<% if (subBeanIndex != null) { %>
   <img src="<%= portletResponse.encodeURL("/images/INS/divider2.gif")%>"
     align="absmiddle" border="0" alt=""/>
   <a style="text-decoration: none;" href="<%= deleteURI %>">
      <img src="<%= portletResponse.encodeURL("/images/INS/delete_box.gif")
%>"
        align="absmiddle" border="0"
        alt="<portletAPI:text key="Sub_Delete_Subscription"
          bundle="nls.INS.PortletMessages"/>"/>
      <span class="wpsTaskIconText">
         <portletAPI:text key="Sub_Delete_Subscription"
           bundle="nls.INS.PortletMessages"/>
      </span>
   </a>
<%}%>
```

7. Display other subscription parameters and allow the user to modify them.

Example 25-59 Subscription parameter display and change sample

```
<tr>
   <td >
      <span class="wpsPortletText">
         <portletAPI:text key="Sub_weather_report_option"
           bundle="nls.INS.PortletMessages" />
      </span>
   </td>
</tr>

<tr>
   <td>
      <input type="radio" value="currentCondition" id="current"
        name="<portletAPI:encodeNamespace value="reportOption"/>"
        <%=currentCondition%>>
      <label for="current">
         <span class="wpsPortletText">
            <portletAPI:text key="Sub_current_condition"
              bundle="nls.INS.PortletMessages" />
         </span>
      </label>
   </td>
</tr>

<tr>
   <td>
      <input type="radio" value="forecast" id="forecast"
        name="<portletAPI:encodeNamespace value="reportOption"/>"
        <%=forecast%>>
      <label for="forecast">
         <span class="wpsPortletText">
            <portletAPI:text key="Sub_forecast"
              bundle="nls.INS.PortletMessages" />
         </span>
      </label>
   </td>
</tr>
```

Delivery channel ordering JSP

This WebSphere Everyplace Access 4.3 core JSP allows the user to change the order in which their available delivery channels are prompted for notification. There is no need for any customization of this JSP.

Deployment

This procedure looks like the delivery channel portlet deployment procedure, which is described in “Deployment” on page 875.

1. Package the portlet as a WAR file.
2. Install the portlet in the WebSphere Portal.
3. Give INS users access to this portlet.
4. Add this portlet to the Intelligent Notification Services My Subscriptions page.

25.6 INS server configuration

In this section, we provide a sample INS server configuration.

25.6.1 E-mail subscription configuration

The INS e-mail subscription application needs to be configured to poll the Microsoft Exchange 2000 and Domino Servers as follows:

1. Log on to WebSphere Everyplace Access server as an administrator, and select **Intelligent Notification -> Administration -> Configure Subscriptions** portlet.
2. Enter the Domino Server's fully qualified address, administration ID and password.
3. Enter the Microsoft Exchange 2000 Server administration ID and password.

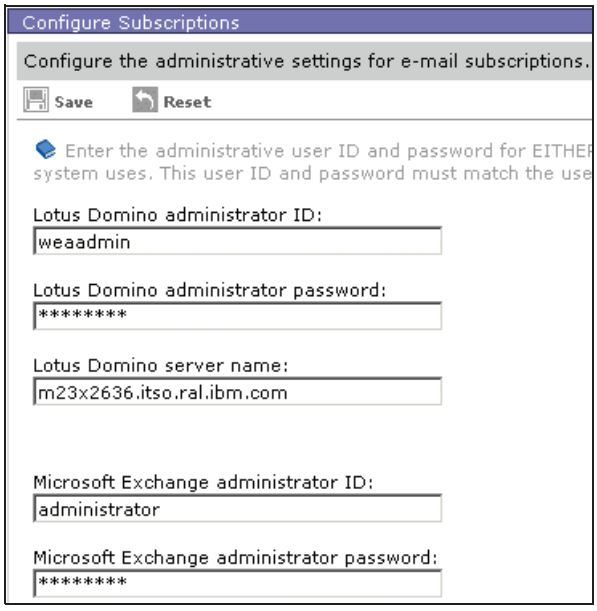


Figure 25-35 Configuring INS e-mail subscription

4. Click **Save** to save the settings
5. Select **Intelligent Notification -> Administration -> Manage Servers** portlet
6. Click **Pause all servers**, then **OK** to stop the INS servers.

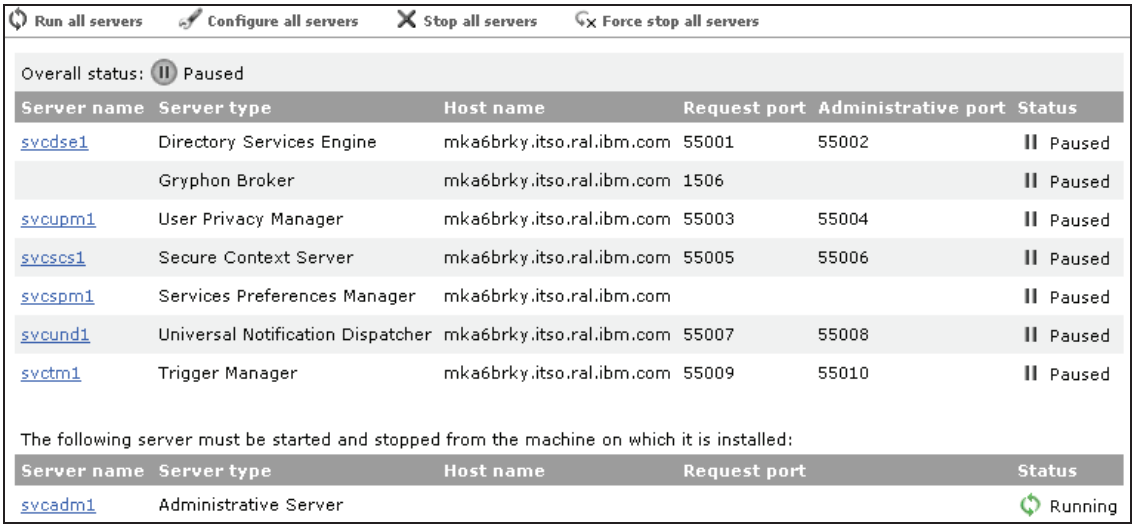


Figure 25-36 Pause all INS servers

7. Once they are all in Paused state (except the Administrative Server), click **Configure all servers**, then **OK** to re-configure the INS servers.
8. Check the server CPU. Once CPU has gone back to almost null, click **Run all servers** to re-start the INS servers. You can alternatively stop, then restart all the servers.

Note: You may notice that only the Domino Server address is required, not the Exchange server address.

This is because when configuring the Domino subscription, the administrator mail file is required, so it needs the name of the host where it resides.

For Exchange, the administrator's server address is not needed because WebSphere Everyplace Access will only rely on the administrator's password of the server where the user's mailbox is located.

This implies that all the Exchange servers housing the user mailboxes must have the same administrator's user and password.

25.6.2 Content adapters configuration

The following tasks need to be done on the Lotus Domino Server:

- Enable DIIOP verification
- Enable JavaScript
- Give Notes administrator ACL read access to the users

Start the WebSphere Everyplace Access 4.3 server

1. On the WebSphere Everyplace Access 4.3 server, edit the file C:\WebSphere\INS\apps\NMAILStartup.properties and check that the settings are correct.
2. Open a DOS windows and change to directory C:\WebSphere\INS\apps.
3. Type the command **startNMAIL**.

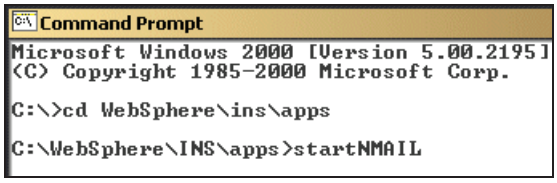


Figure 25-37 Starting Lotus Notes e-mail content adapter

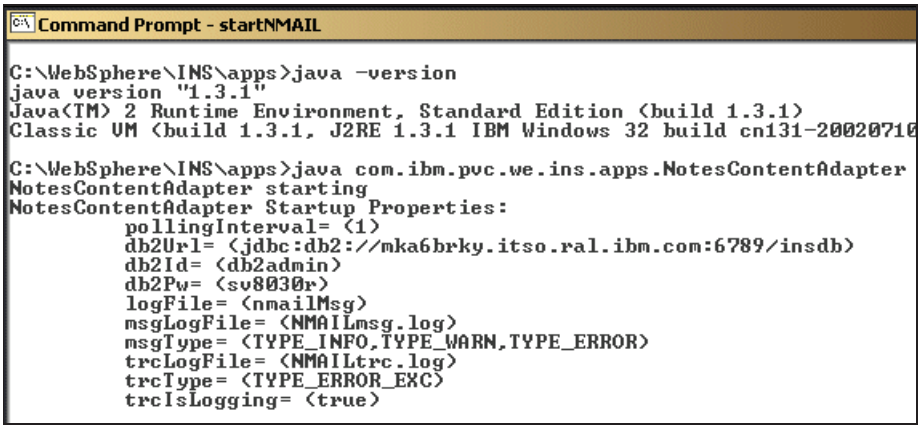


Figure 25-38 Lotus Notes content adapter starting log

Microsoft Exchange e-mail content adapter

To do Microsoft Exchange server settings, perform these steps:

1. Check the Microsoft Exchange Server to confirm that the user mailboxes have the proper access rights:

a. On the Microsoft Exchange Server, open the Exchange System Manager (select **Start -> Programs -> Microsoft Exchange -> System Manager**)

b. Select **Servers-> *hostname* ->First Storage Group -> Mailbox Store (*hostname*)**, right-click **Properties**, select the **Security** tab and check that all the permission settings are allowed for the Administrator user.

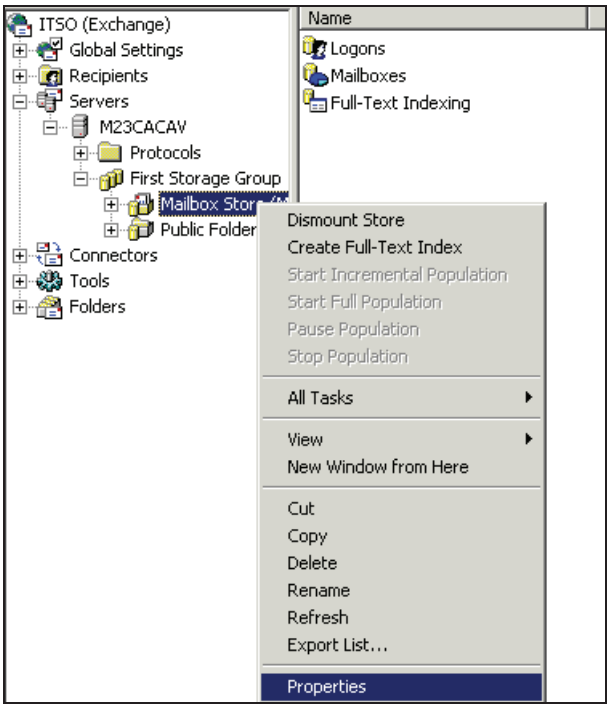


Figure 25-39 Selecting mailbox store properties

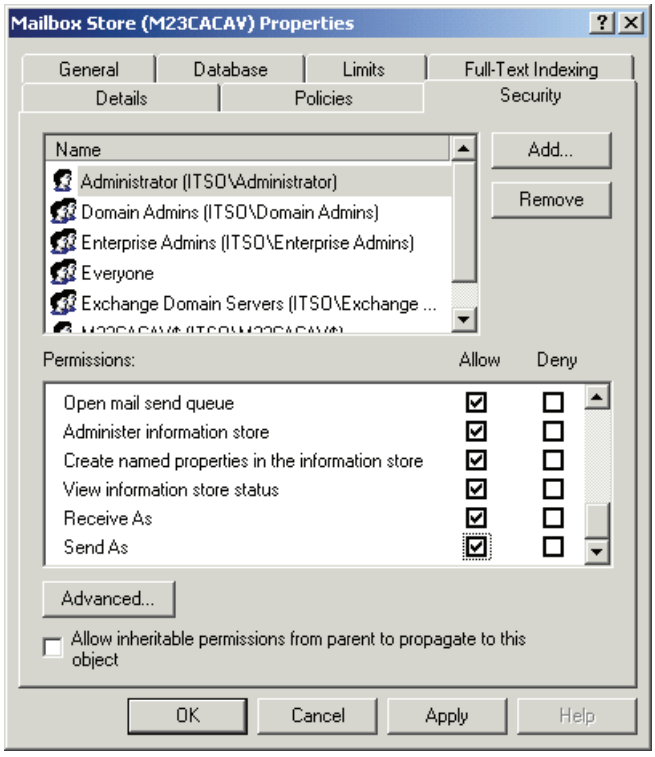


Figure 25-40 Mailbox store properties

You may have to uncheck **Allow inheritable permissions from parent to propagate to this object** to be able to get the two last rights, Receive As and Send As. In such case, click **Copy** in the window shown in Figure 25-41.

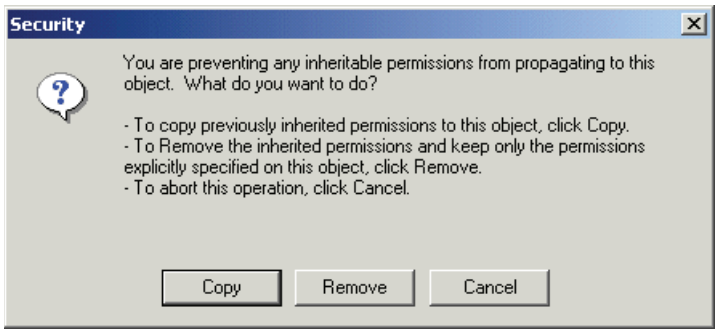


Figure 25-41 Copy inherited permissions to the object

c. You may need to restart the Microsoft Exchange Server.

WebSphere Everyplace Access 4.3 server starting

1. On the WebSphere Everyplace Access 4.3 server, edit the file C:\WebSphere\INS\apps\XMAILStartup.properties and check that the settings are correct.
2. Open a DOS windows and change to directory C:\WebSphere\INS\apps.
3. Type the command **startXMAIL**.



Figure 25-42 Starting Microsoft Exchange content adapter

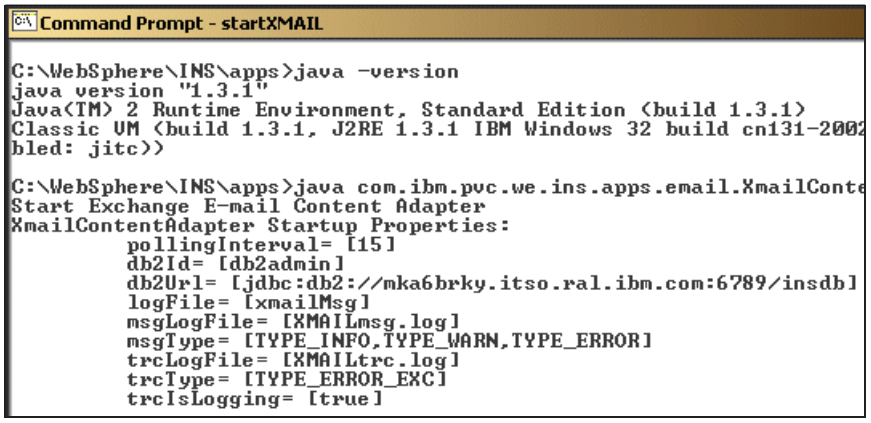


Figure 25-43 Microsoft Exchange content adapter starting log

Important: Keep this window opened, or the Microsoft Exchange Content Adapter will stop.

Re-run this command each time you restart the INS servers.

RSS News content adapter

The RSS News content adapter polls the news content sources for data in RSS (Rich Site Summary) format. The retrieved data consists of a channel of titles,

links, and news article short description. The content source is a Web site. Configuring and starting RSS News content adapter consists of:

- Finding the content source channel.
- Updating accordingly the RSS Subscription portlet parameters.
- Updating accordingly the RSSStartup.properties file.
- Starting the RSS content adapter.

Find the content source channel

Let's consider the example of the Moreover RSS news provider site:

1. Open the RSS site:

   `http://w.moreover.com/categories/category_list_rss.html`

2. Pick several news feeds to be added to the RSS News portlet, for example IBM news and soccer news. For each of the corresponding URLs, note the channel title.

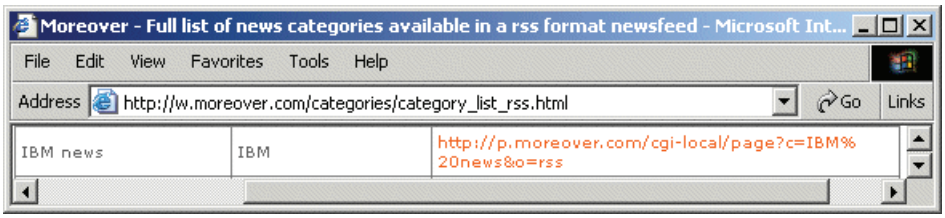


Figure 25-44 Choose an RSS site



Figure 25-45 Getting RSS site channel title

– To obtain IBM news, set the following:
  - RSS site: `http://p.moreover.com/cgi-local/page?c=IBM%20news&o=rss`
  - Channel title: `Moreover - IBM news`
– For soccer news, set the following:
  - RSS site: `http://p.moreover.com/cgi-local/page?c=Sports%3A%20soccer%20news&o=rss`
  - Channel title: `Moreover - Sports: soccer news`

Update the RSS Subscription portlet

1. Log on as an administrator to the WebSphere Everyplace Access 4.3 server and select **Portal Administration-> Portlets -> Manage Portlets**.
2. Select the Intelligent Notification RSS Subscriptions portlet, and click **Modify parameters**.
3. For each RSS channel to be added, add the parameters rssFeedsX and rssFeedsDisplayX, where X is a sequential number starting at 0 and being incrementing to each added channel.

   For each added channel, the rssFeedsX parameter must match the RSS channel title, whereas the rssFeedsDisplayX parameter identifies with your own words the RSS channels.

   Enter the required parameters, then select **Save** to save your changes, then **Cancel** to return.

   – IBM news:

rssFeeds0	Moreover - IBM news
rssFeedsDisplay0	My IBM news

   – Soccer news:

rssFeeds1	Moreover - Sports: soccer news
rssFeedsDisplay1	My soccer news

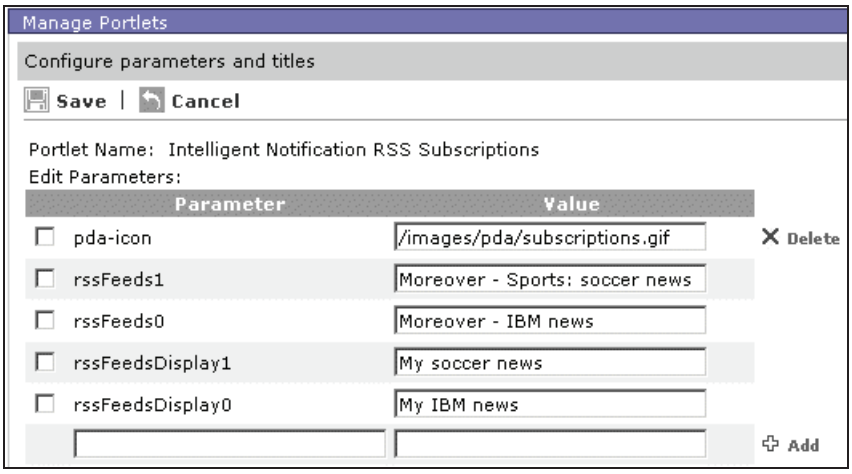


Figure 25-46 INS RSS subscriptions portlet parameters

Update the RSS content adapter properties file

On the WebSphere Everyplace Access 4.3 server, edit the file C:\WebSphere\INS\apps\RSSStartup.properties:

1. Check that the default settings are correct (such as parameter pollingInterval).
2. The last line of the file should be the rssFeeds parameter. Enter the URLs of each of the added channels, separated by a comma.

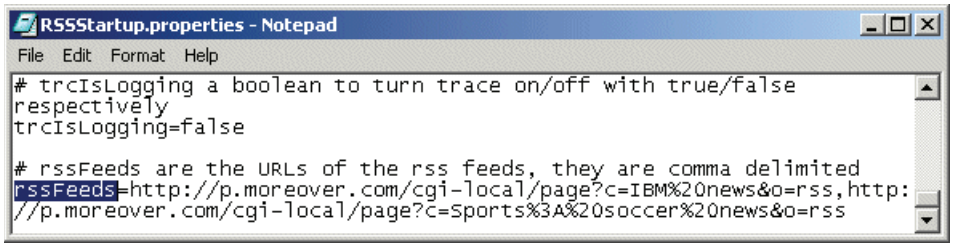


Figure 25-47 rssFeeds properties value

Start the RSS content adapter

1. Open a DOS windows and change to directory C:\WebSphere\INS\apps.
2. Type the command **`startRSS RSSStartup`**.

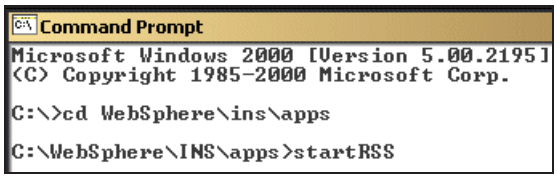


Figure 25-48 Starting RSS News content adapter



Figure 25-49 RSS news content adapter starting log

Important: \Keep this window opened, or the RSS news content adapter will stop.

Re-run this command each time you restart the INS servers.

25.6.3 Gateway adapters configuration

Standard gateway adapters are configured via the INS administration portlets, whereas custom gateway adapters need to be manually installed.

Among the provide INS gateway adapters, mail, messaging, Sametime and WAP gateway adapters are standard, whereas the message center gateway adapter is a custom one.

Configuring standard gateway adapters

1. Log in to the WebSphere Portal as an Intelligent Notification administrator (for example `wpsadmin`).
2. Select **Intelligent Notification**.

3. Select **Administration -> Configure Gateways**. A list of gateway adapters appears.
4. Configure a standard gateway adapter by selecting it from the list, then clicking **Edit**. A list of editable configuration parameters appears.
   – Mail gateway: Enter the fully qualified host name of the SMTP server.

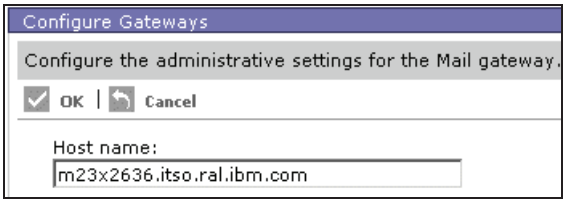


Figure 25-50 Mail gateway configuration

   – Messaging gateway: Enter the fully qualified URL of the SMS Gateway along with the port on which the SMS gateway listens to the incoming requests. When the SMS gateway is WebSphere Everyplace Connection Manager, this port ought to be 13131.

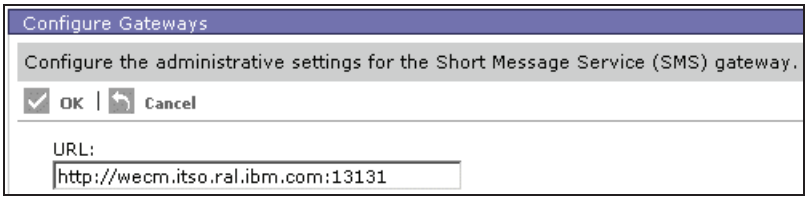


Figure 25-51 Messaging gateway configuration

   – Sametime gateway: Enter the fully qualified host name of the Sametime Server and the administrative ID and password of the Domino Server.

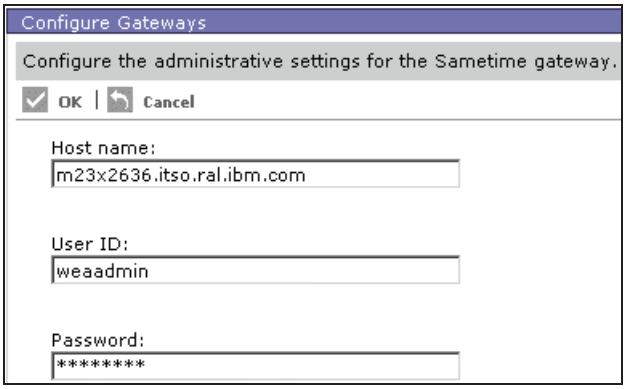


Figure 25-52 Sametime gateway configuration

– WAP gateway: Enter the fully qualified URL of the WAP Gateway along with the port on which the WAP gateway listens to the incoming requests. When the WAP gateway is WebSphere Everyplace Connection Manager, this port ought to be 13131.

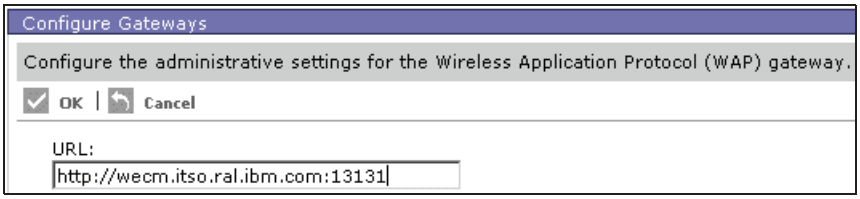


Figure 25-53 WAP gateway configuration

5. Edit the configuration parameters.
6. Click **OK** to save your changes to the configuration parameters.
7. Restart the INS servers via the Manage Servers portlet to stop and restart all servers to ensure that the servers are stopped and started in the correct order.

Modifying WAP gateway adapter to accept target IP addresses

When a developer wants to test their notification on a phone, they may want to use an emulator such as a Nokia toolkit when sending WAP Push messages to a phone. This is what is needed for scenario 3.

This requires a change in the INS universal dispatcher code, to make it accept an IPv4 address instead of the usual phone number, which is a PLMN address.

The sample code is delivered on the WebSphere Everyplace Access V4.3 server to directory C:\WebSphere\INS\samples\adaptors, file WAP.java.

Note: This code change does not work for SMS gateway, because the WebSphere Everyplace Connection Manager Push Proxy API requires the PLMN format for SMS delivery. Instead, a mapping table needs to be implemented on the WebSphere Everyplace Connection Manager side to map dummy phone numbers to the actual target IP addresses.

The procedure is as follows:

1. To indicate the WebSphere Everyplace Connection Manager Push Proxy API that an IPv4 address is used instead of a regular PLMN address, we use the send() method of the class, as shown in Example 25-60 and Example 25-61.

Example 25-60 Original PLMN set address line

```
p_address.setAddress( _phoneNumber, PushAddress.PLMN, PAPUser.getHost());
```

Example 25-61 Changed IPv4 set address line

```
p_address.setAddress( _phoneNumber, PushAddress.IPV4, PAPUser.getHost());
```

Note: This code change has to be made in conjunction with the corresponding LDAP change described in “WAP delivery channel” on page 933.

2. Once the change is done, save and compile the file. The compilation class path should contain the JAR files ins.jar, push.jar, and jlog.jar, which are located in C:\WebSphere\INS\lib directory of the WebSphere Everyplace Access 4.3 server.
3. Then from root directory C:\WebSphere\INS, create the directory structure com\ibm\pvc\we\ins\und\server\adaptors (which the package to which the changed class pertains), and place the changed class in the adaptors directory.
4. Then restart the INS servers.

Configuring custom gateway adapters

For example, execute the following steps to change the configuration of the message center gateway adapter:

1. Open the file C:\WebSphere\INS\UNDGateways\RecentAlerts\gw.properties.
2. Check the properties and change them accordingly. Note that in most cases the default settings should fit your needs.

25.7 INS user configuration

For the purposes of the scenarios, users John Doe, Jim Smith, and Bill Johnson are created as follows via the WebSphere Everyplace Access Sign Up icon:

- John Doe

  WEA ID jdoe
  First name John
  Last name Doe
  Email jdoe@itso.ral.ibm.com

- Jim Smith

  WEA ID jsmith
  First name Jim
  Last name Smith
  Email jsmith@itso.ral.ibm.com

- Bill Johnson

  WEA ID bjohnson
  First name Bill
  Last name Johnson
  Email bjohnson@itso.ral.ibm.com

These users are then made part of the Everyplace Access group INSUsersGroup, as shown in Figure 25-12 on page 851.

Scenario 1 user configuration

In Scenario 1, John Doe wants to have the INS log file updated when he receives an urgent Microsoft Exchange 2000 mail from his manager Jim Smith.

Therefore, John Doe needs to have a Microsoft Exchange e-mail subscription, and to define one INS log delivery channel.

The delivery channel needs to be defined first, since it will have to be checked when defining the subscription.

INS log file portlet delivery channel

Log into WebSphere Everyplace Access as user John Doe and select **Intelligent Notification -> My Delivery Channels -> INS Log Custom Delivery -> Add delivery channel** and fill in the custom delivery portlet required fields.

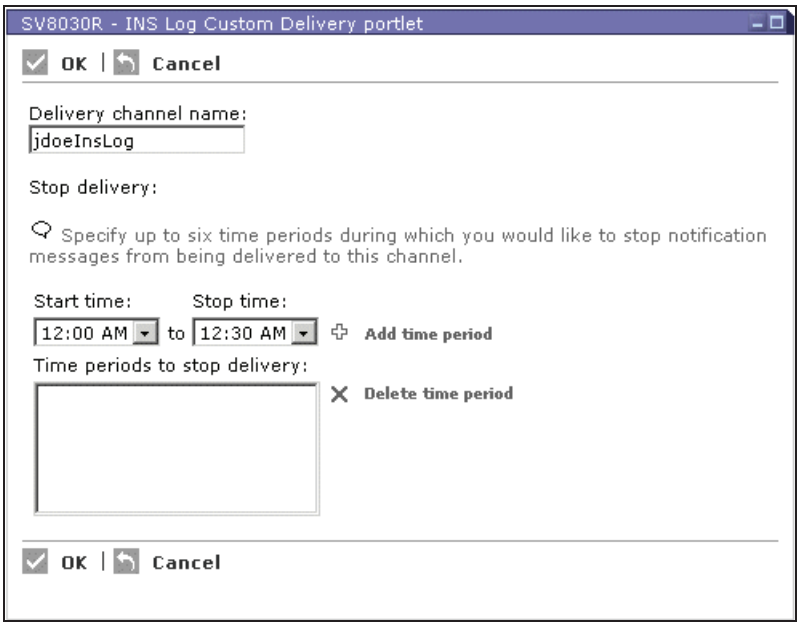


Figure 25-54 Configuring one INS log delivery channel



Figure 25-55 Configured INS log delivery channels

Microsoft Exchange subscription

1. Log into WebSphere Everyplace Access as user John Doe and select **Intelligent Notification -> My Subscriptions -> My Microsoft Exchange subscriptions** portlet. The Exchange e-mail account first needs to be configured.



Figure 25-56 My Microsoft Exchange subscription portlet initial window

2. Click **Setup your e-mail account**, fill in the required fields, and click **OK**.



Figure 25-57 Microsoft Exchange subscription e-mail configuration

> **Note:** The user's Exchange server name is required because it may differ from one user to another.
>
> Likewise, for a Lotus Notes subscription, the user's Domino Server name would be required for a Lotus Notes subscription, because it may differ from the one of the Domino administrator.

3. Select **Intelligent Notification -> My Subscriptions -> My Microsoft Exchange subscriptions** portlet, select **Add subscription**, and fill the required fields to create a subscription that, when urgent mail from Jim Smith occurs, delivers a notification to John Does's custom delivery channel as well as the default John Doe's message center.

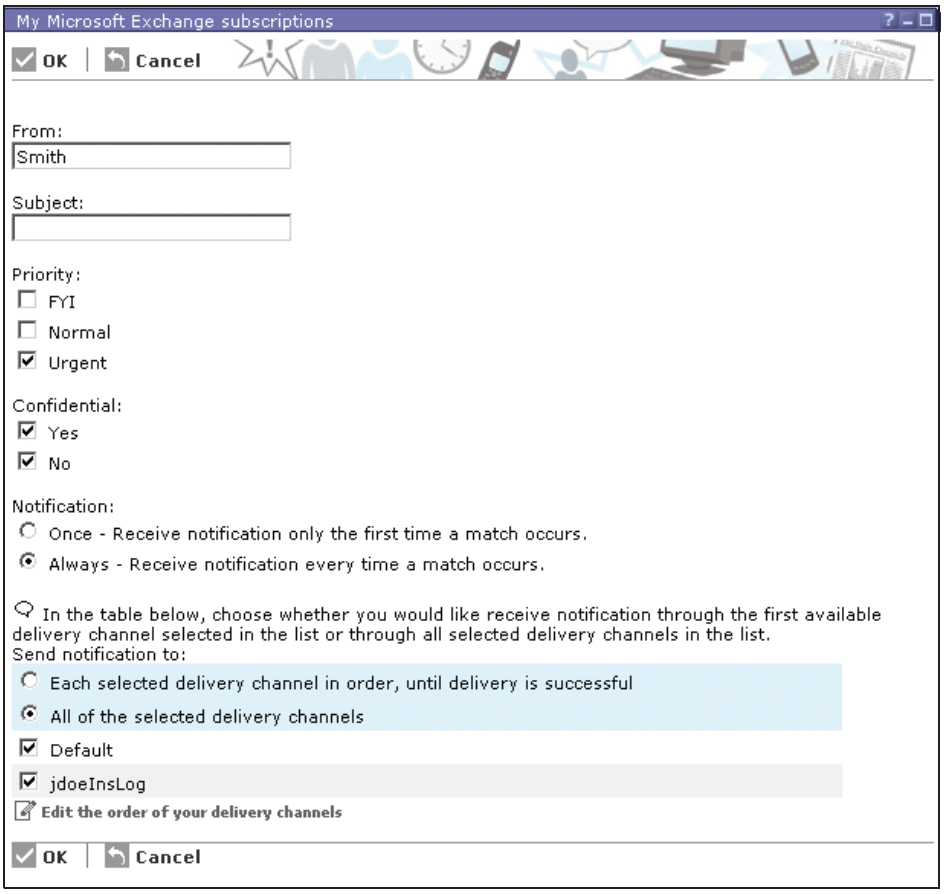


Figure 25-58 Add urgent mail from Jim Smith exchange subscription

Triggering the notification

Jim Smith send an urgent Exchange note to John Doe.

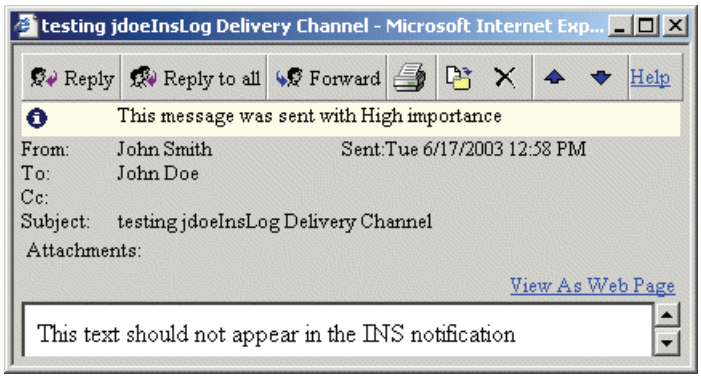


Figure 25-59 Urgent mail from Jim Smith to John Doe

John Doe's message center then shows the notification.

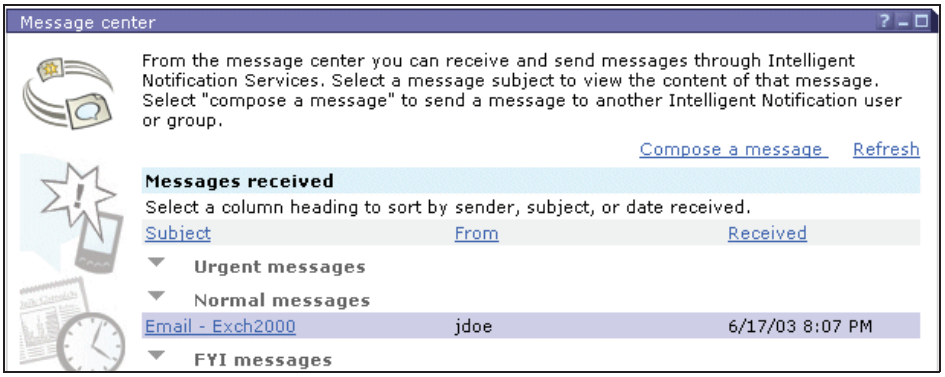


Figure 25-60 Message center triggered notification



Figure 25-61 Message center expanded notification

John Doe's INS Log delivery channel then shows the notification.

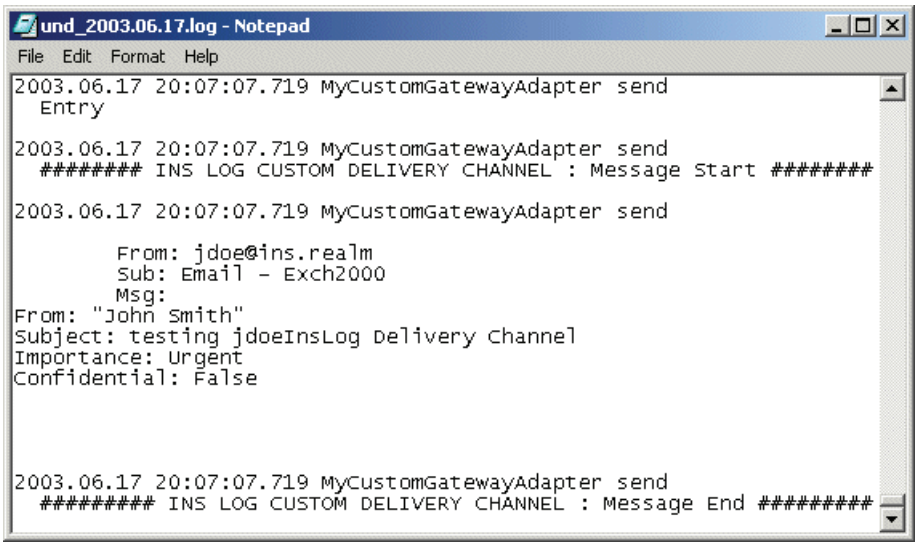


Figure 25-62 Universal notification dispatcher LOG file showing custom delivery channel

Scenario 2 user configuration

In Scenario 2, Jim Smith wants to receive an instant message when connected or an e-mail when weather condition changes.

Therefore, Jim Smith needs to have a weather subscription, and to define one Sametime and one SMTP e-mail delivery channels.

The delivery channels need to be defined first, since they will have to be checked when defining the subscription.

Sametime delivery channel

Note: The Messaging gateway must have already been configured. Refer to “Configuring standard gateway adapters” on page 917.

Log into WebSphere Everyplace Access as user Jim Smith and select **Intelligent Notification -> My Delivery Channels -> Sametime delivery channels** portlet, select **Add delivery channel** and fill in the Sametime delivery portlet required fields.

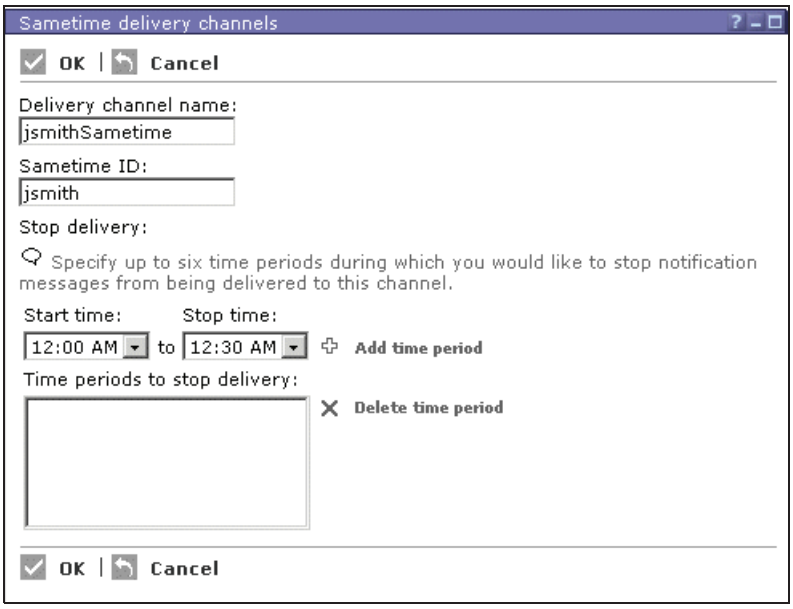


Figure 25-63 Configuring one Sametime delivery channel



Figure 25-64 Configured Sametime delivery channel

SMTP e-mail delivery channel

Note: The Messaging gateway must have already been configured. Refer to “Configuring standard gateway adapters” on page 917.

Log into WebSphere Everyplace Access as user Jim Smith and select **Intelligent Notification -> My Delivery Channels -> SMTP mail delivery channels -> Add delivery channel** and fill in the Sametime delivery portlet required fields.

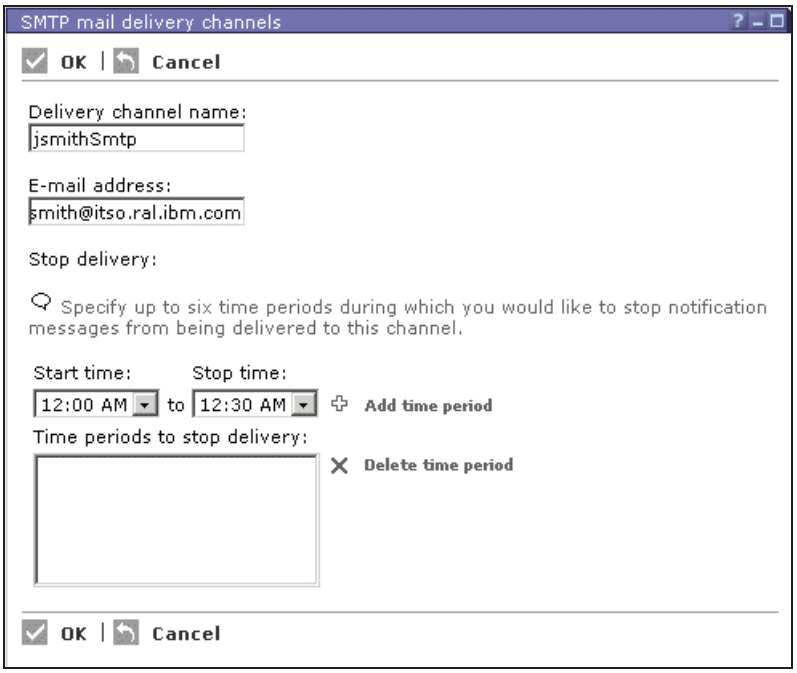


Figure 25-65 Configure one SMTP mail delivery channel



Figure 25-66 Configured SMTP mail delivery channel

Weather subscription

For this scenario, we will use the provided weather sample content feeding.

This sample uses the XML content adapter to parse input XML files. Some weather forecast sample XML files are provided in directory C:\WebSphere\INS\samples\weather of the WebSphere Everyplace Access 4.3 server.

The sample files weather2a.xml and weather2b.xml show that there are forecast hits after current date 03/27/2001 for Surf City, NC.

Therefore, we will subscribe for a forecast of Surf City, NC, and should obtain two notifications:

- Mostly sunny, highs near 60, lows lower 40s.
- Mostly cloudy with a chance of showers. Breezy, highs upper 60s.

Log into WebSphere Everyplace Access as user Jim Smith and select **Intelligent Notification -> My Subscriptions -> My Weather subscriptions** portlet, select **Add subscription**, and fill in the required fields to create a subscription that, when a weather forecast for Surf City, NC occurs, delivers a notification to Jim Smith's Sametime delivery channel as well as Jim Smith's SMTP mail delivery channel.

1. Fill in the City and State fields
2. Select the **Forecast** radio button.
3. Select **Always - Receive notification every time a match occurs**
4. Select **Each selected delivery channel in order, until delivery is successful**
5. Check Jim Smith's Sametime and SMTP mail delivery channels
6. Select **Edit the order of your delivery channels** if necessary

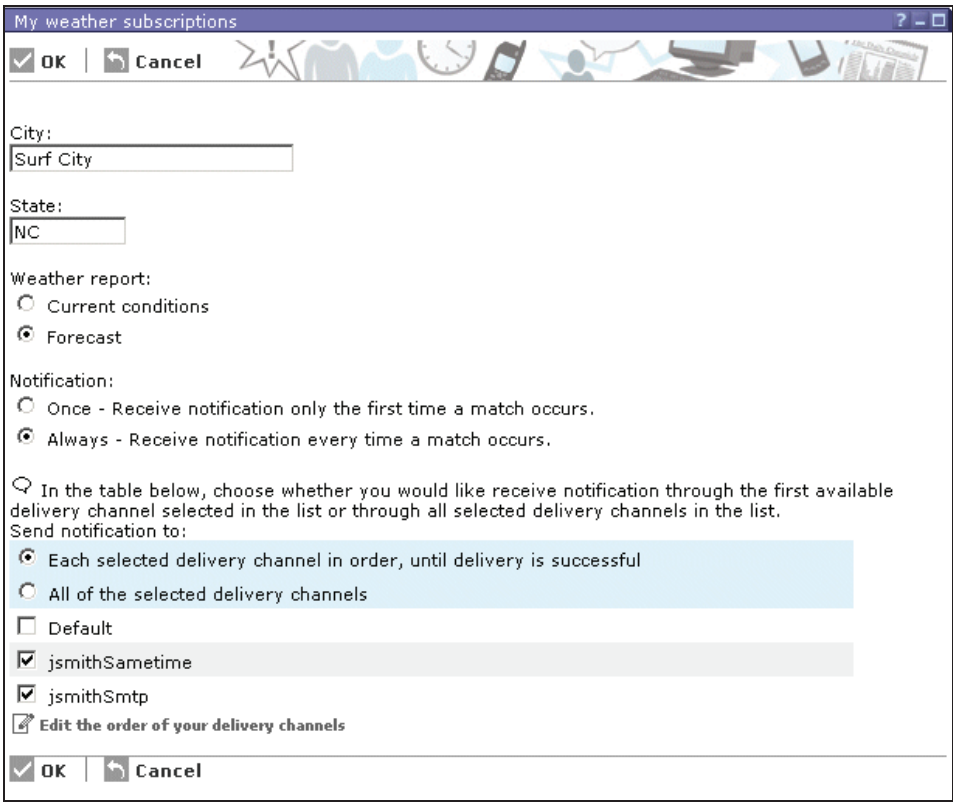


Figure 25-67 Configuring on weather subscription

Triggering the subscription

To simulate the reception of a weather forecast, the XML content adapter needs to be fed with a sample weather XML file.

1. On the WebSphere Everyplace Access 4.3 server, open a DOS window and change to the directory C:\WebSphere\INS\bin.
2. Type the command **`contentfeed weather weather`**

This will launch Java class XMLContentAdapter, which will parse XML files found in the directory C:\WebSphere\INS\samples\weather and feed it with weather forecast information about Surf City, NC, and therefore trigger a notification to Jim Smith.

When Sametime is connected, Jim Smith will not receive any Exchange mail, and will receive a Sametime notification, as shown in Figure 25-68 on page 931.

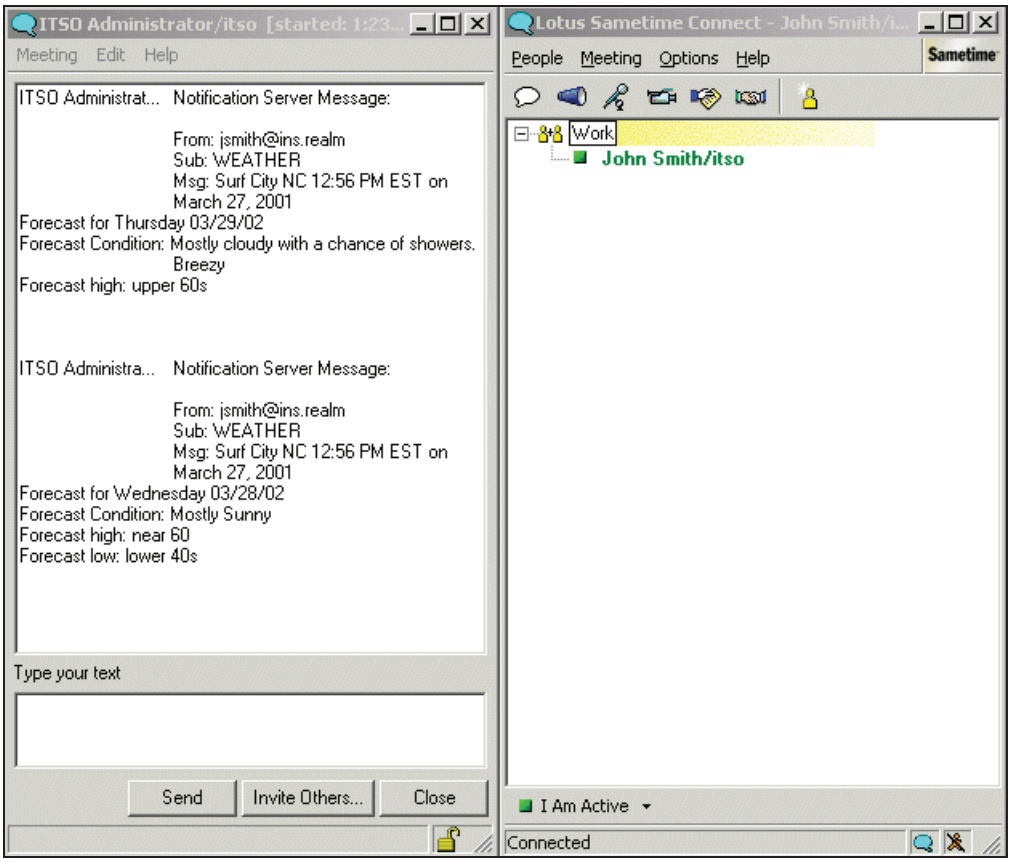


Figure 25-68 Weather Sametime notification

When not connected, Jim Smith will receive two Exchange notes, as shown in Figure 25-69 on page 932 and Figure 25-70 on page 932.

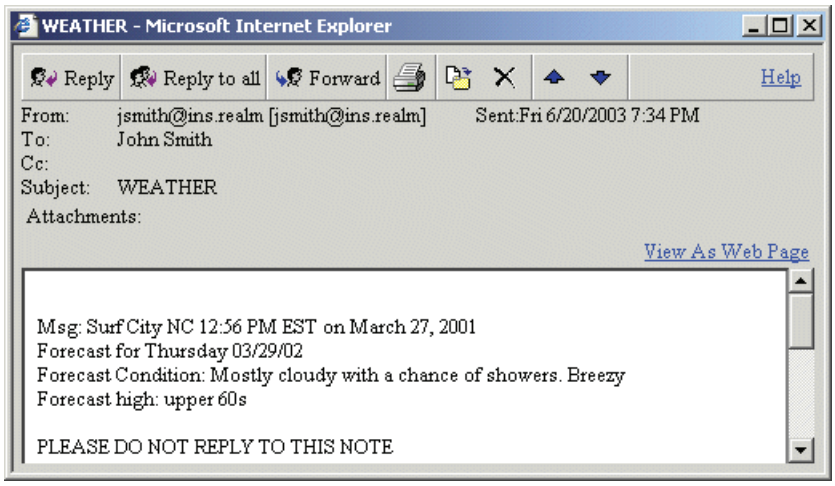


Figure 25-69 Weather Exchange mail notification

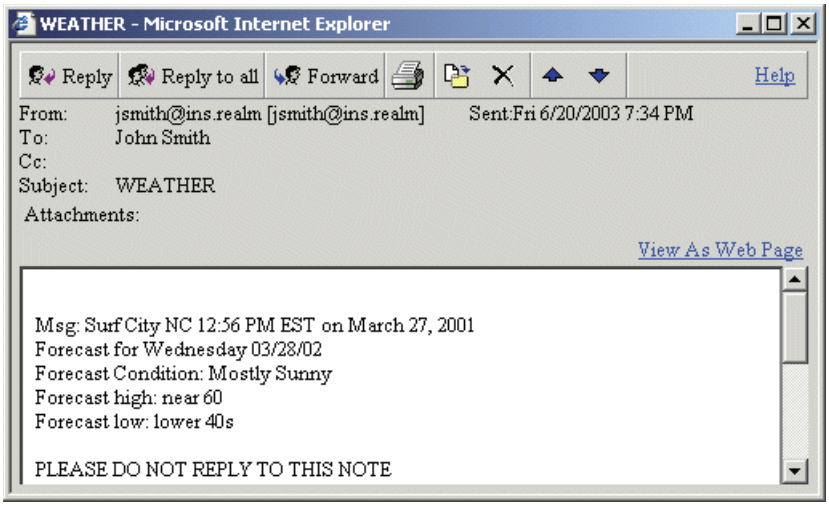


Figure 25-70 Weather Exchange mail notification

Scenario 3 user configuration

In Scenario 3, Bill Johnson wants to receive a WAP Push message for news about his favorite soccer team.

Therefore, Bill Johnson needs an RSS news subscription, and to define one WAP delivery channel.

The delivery channel needs to be defined first, since it will have to be checked when defining the subscription.

WAP delivery channel

Note: The Messaging gateway must have already been configured. Refer to “Configuring standard gateway adapters” on page 917 and “Modifying WAP gateway adapter to accept target IP addresses” on page 919.

Since we will be using a Nokia toolkit, the address of the device will be given as an IP address instead of a phone number. The WAP delivery channel portlet expects a phone number, and automatically adds a leading “+” to the phone number. Since we have changed the WAP delivery channel in “Modifying WAP gateway adapter to accept target IP addresses” on page 919, we need to remove the leading “+” once the portlet has processed the delivery channel creation. This will be done directly into LDAP as specified below.

The steps to create a WAP delivery channel with an IP addressing device are as follows:

1. Log into WebSphere Everyplace Access as user Jim Smith and select **Intelligent Notification -> My Delivery Channels-> WAP delivery channels** portlet, select **Add delivery channel**, and fill in the WAP delivery portlet required fields.

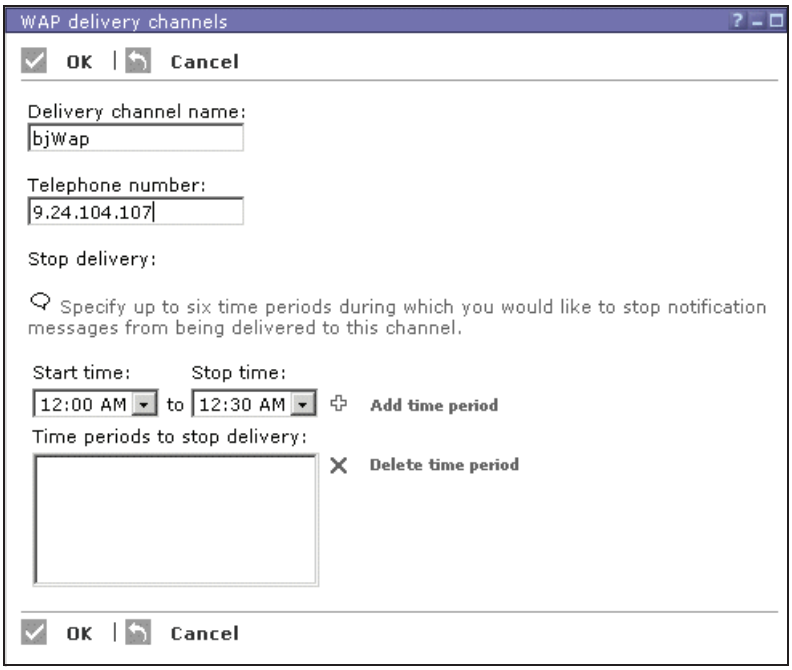


Figure 25-71 Configure one WAP delivery channel



Figure 25-72 Configured WAP delivery channel

2. Log into the LDAP server and retrieve the above delivery channel in the LDAP tree.

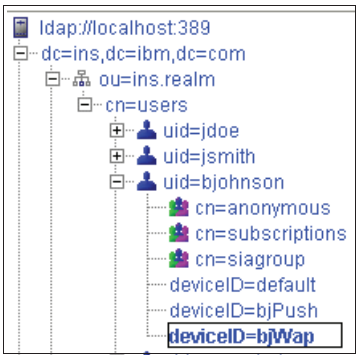


Figure 25-73 WAP delivery channel LDAP settings

3. Double-click the corresponding device ID, and remove the leading “+” from field ibm-appDeviceAddress, as shown in Figure 25-74 on page 936.

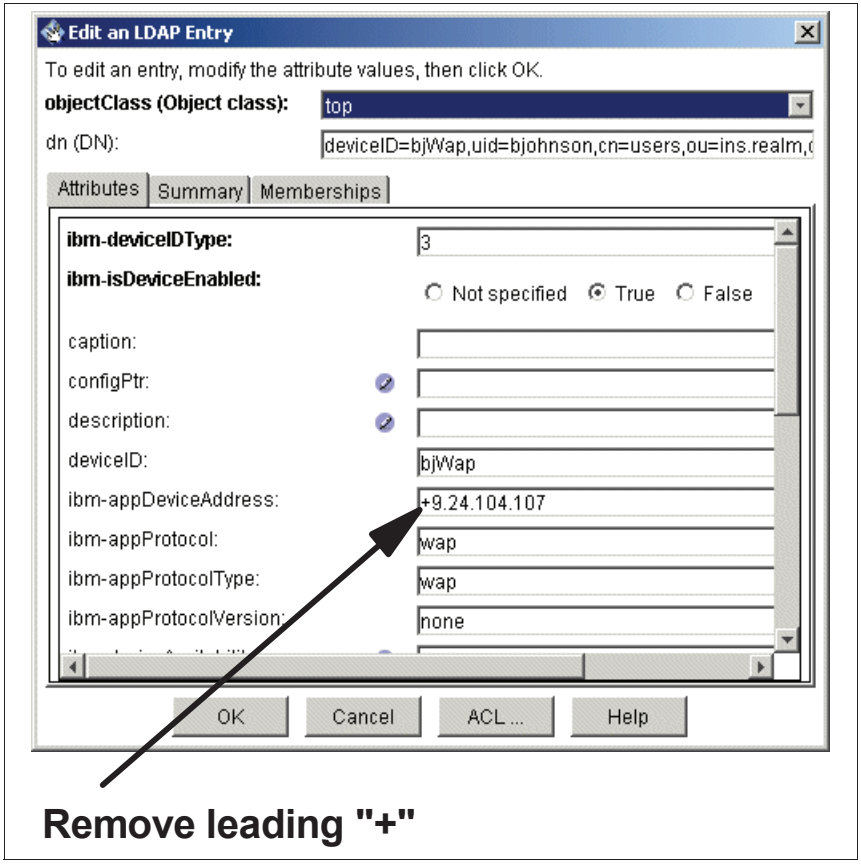


Figure 25-74 Removing leading "+" in LDAP

RSS news subscription

For this scenario, we will use the RSS news content adapter. Refer to "RSS News content adapter" on page 913 for details on how to configure the RSS news content adapter so that the soccer news appears on the RSS news subscription portlet.

Log into WebSphere Everyplace Access as user Bill Johnson and select **Intelligent Notification -> My Subscriptions -> My RSS newssubscriptions** portlet, select **Add subscription**, and fill in the required fields to create a subscription that, when some news about the Arsenal team occurs, delivers a notification to Bill Johnson's WAP delivery channel.

1. In the News source drop-down menu, select **My soccer news**.
2. In the Subject contains field, enter **Arsenal**.
3. Select **Always - Receive notification every time a match occurs**.

4. Select **All of the selected delivery channels**.
5. Check Bill Johnson's WAP delivery channel.

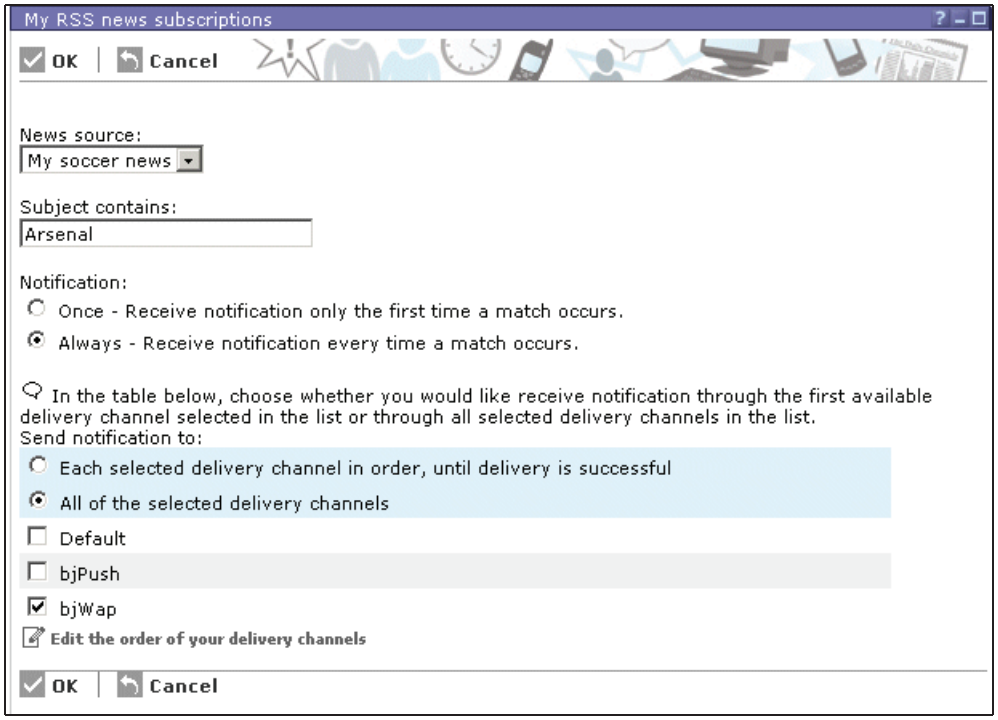


Figure 25-75 Adding an RSS news subscription

Triggering the subscription

The subscription is triggered whenever some news dealing with Arsenal occurs.

As of now, the Nokia toolkit does not format received WAP Push messages containing WML cards. Therefore, the reception of this message will be seen in the toolkit's Push inbox.

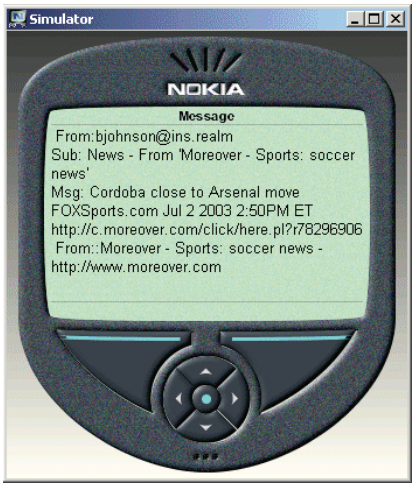


Figure 25-76 Received message seen via the device simulator

25.8 Using the message center

In addition to being a possible delivery channel, the message center can be used to send notifications to another user or a group of users. An INS user must be the member of your user group before he or she can send an INS message to you.

Once you have created a user group, INS allows you to categorize users of this group and give them specific access to certain delivery channels. The message rules define the type of notification access of a user group.

Note: Any regular INS user can also send messages from the message center to other users regardless of how the user was created or is part of any user group.

To describe these INS features, we will use a scenario in which John Doe wants to communicate via the message center with his co-workers Jim Smith and Bill Johnson.

Then John Doe wants to add a rule that says that urgent messages from his co-workers should be delivered to Sametime.

25.8.1 Create a user group

- Log in to WebSphere Everyplace Access 4.3 as user `jdoe` and select **Intelligent Notification -> My user groups -> Add user group**.



Figure 25-77 Selecting add user group

- Select the name of the new group myCo-workers in the Name f the user group field, then click **Add member**.

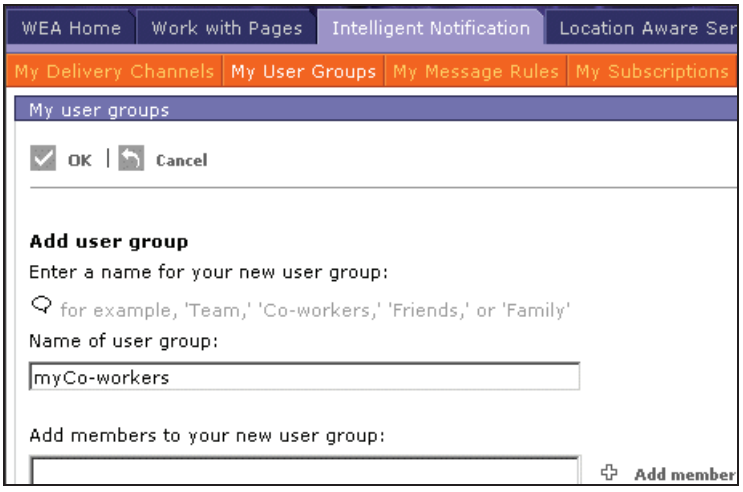


Figure 25-78 Typing user group name and requesting member addition

- On the Add members to user group window, type * in the User name is field to get all the INS users and click **Go**. A list of the INS users displays. Add Jim Smith and Bill Johnson to the user group by selecting both of them one after the other and clicking **Add to list**, then clicking **OK**, then **OK** again.

Note: Only those INS users that actually have INS data (that is, they have accessed any of their INS portlets at least once after having been added to INSUsersGroup) will appear in the search result.

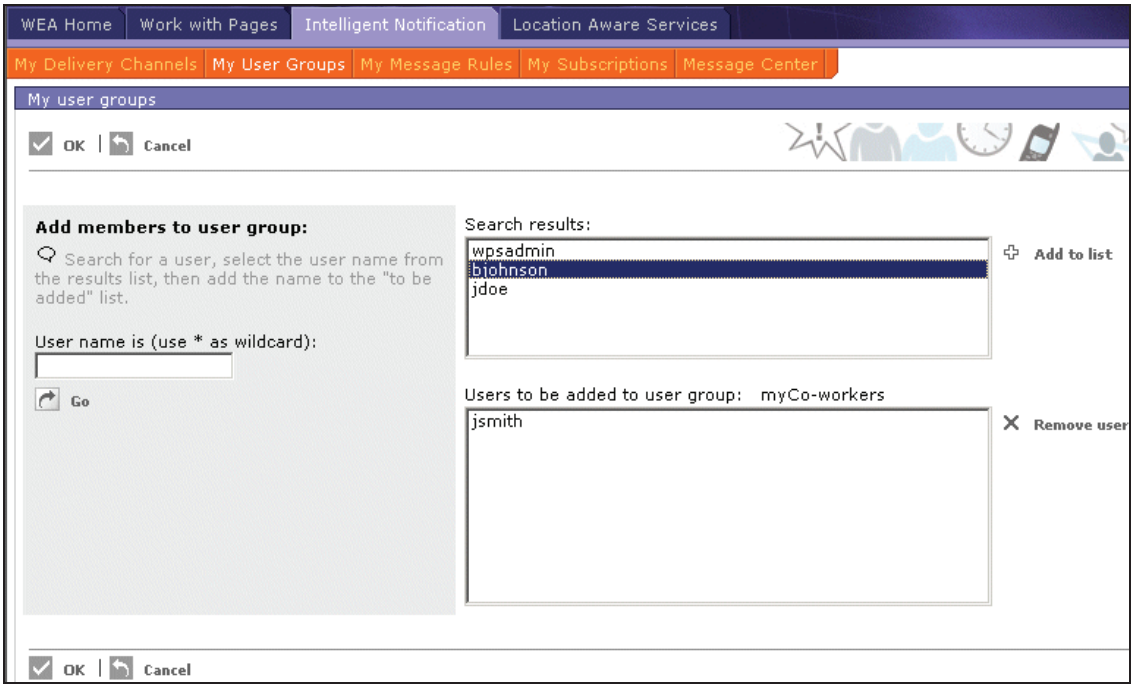


Figure 25-79 Adding INS users to a user group

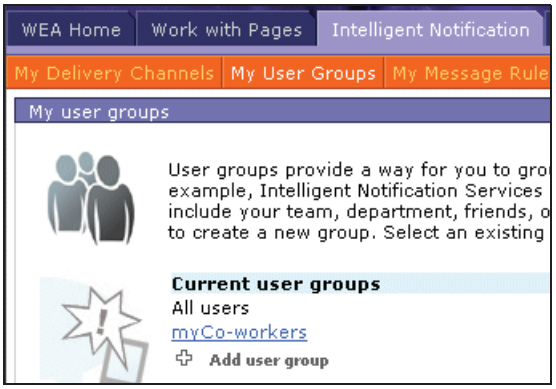


Figure 25-80 INS user group added

25.8.2 Create a message rule

- Log in to WebSphere Everyplace Access 4.3 as user `jdoe`, select **Intelligent Notification -> My user rules -> Add rule** under the Urgent message rules section.

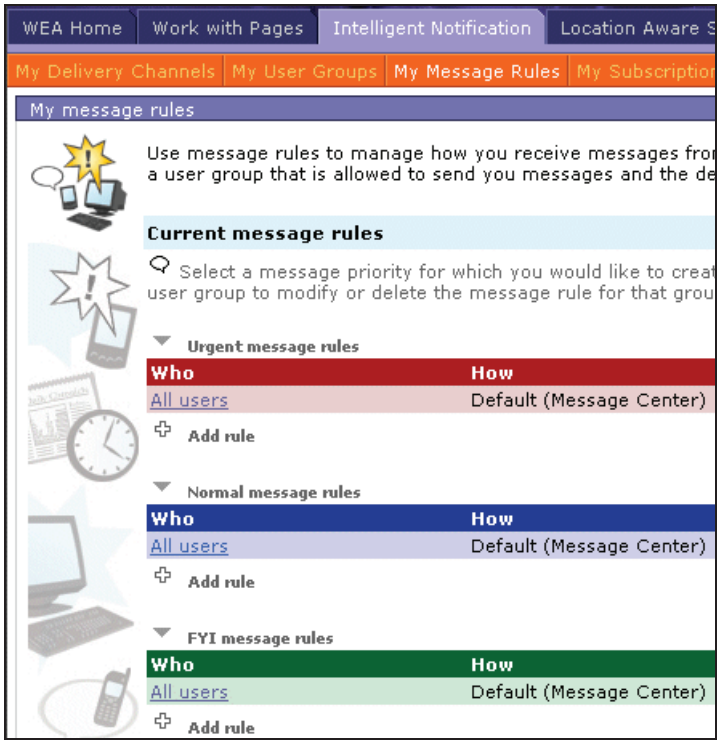


Figure 25-81 Adding a message rule

- In the Add a message rule window, select user group **myCo-workers**, then check the Sametime delivery channel **jdoeSametime**, then click **OK**.

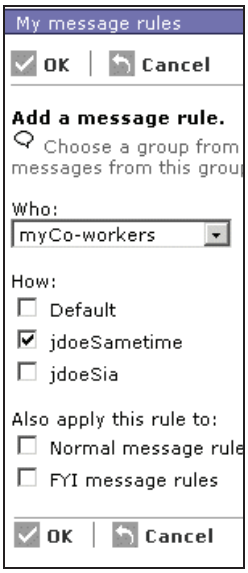


Figure 25-82 Adding a message rule (continued)

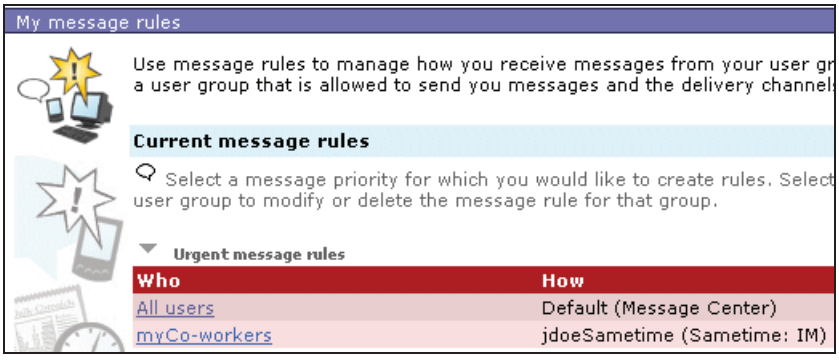


Figure 25-83 User message rule added

25.8.3 Enabling the delivery channels

With WebSphere Everyplace Access, the INS internal message center content adapter delivers notification by default only to the message center. This can be corrected via the correct setting of a parameter of the INS message center portlet as follows:

1. Log into WebSphere Everyplace Access 4.3 as the administrator, select **Portal Administration -> Portlets -> Manage portlets**, select the first

Intelligent Notification Message Center portlet, and click **Modify parameters**.

2. The list of portlet parameters appears. You need to add or modify the parameter UND.PREF.DEVICE and set it to *.

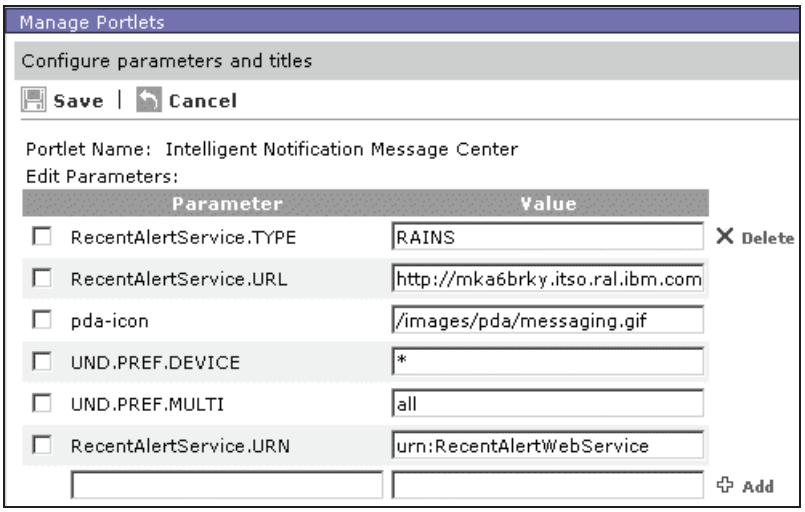


Figure 25-84 Setting UND.PREF.DEVICE in INS message center portlet

3. You may have to restart WebSphere Portal Server.

25.8.4 Triggering message rules

The previous section now allows user John Doe to receive an instant message when one person in his INS group sends him an urgent message from the message center. Say Bill Johnson sends an urgent message via his message center to John Doe. Then John will receive both this message on his message center and an instant message on his Sametime user.

This is performed as follows:

- Bill Johnson logs in to the WebSphere Everyplace Access 4.3 server, selects **Intelligent Notification -> Message Center -> Compose a message**.
- He fills in the required fields, then selects **OK**.

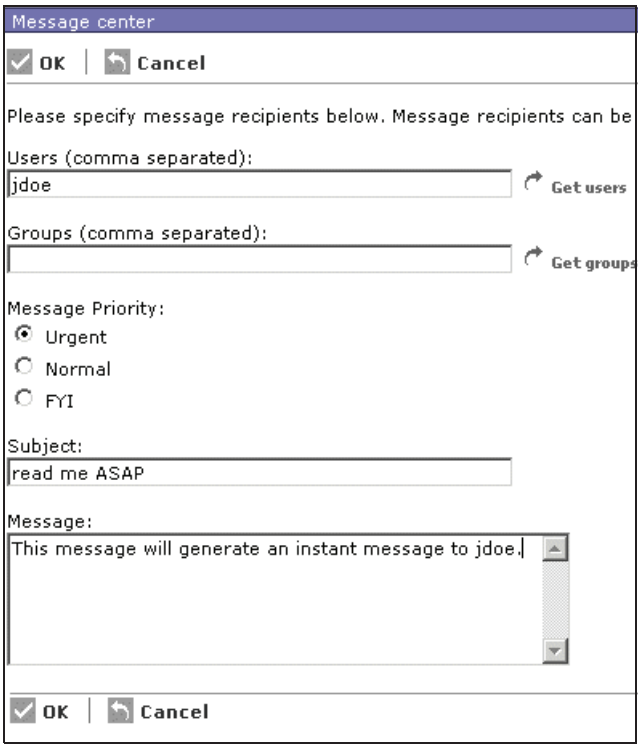


Figure 25-85 Sending an urgent message to John Doe

- An instant message is received by John Doe, if connected on Sametime on his desktop.

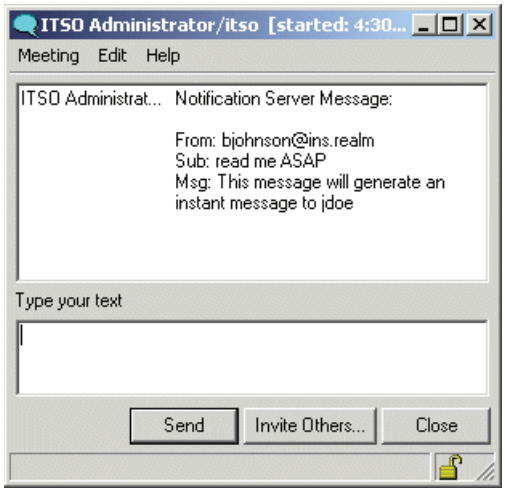


Figure 25-86 Sametime received by jdoe

- Then John Doe logs into the Portal to check the generated message from Bill.

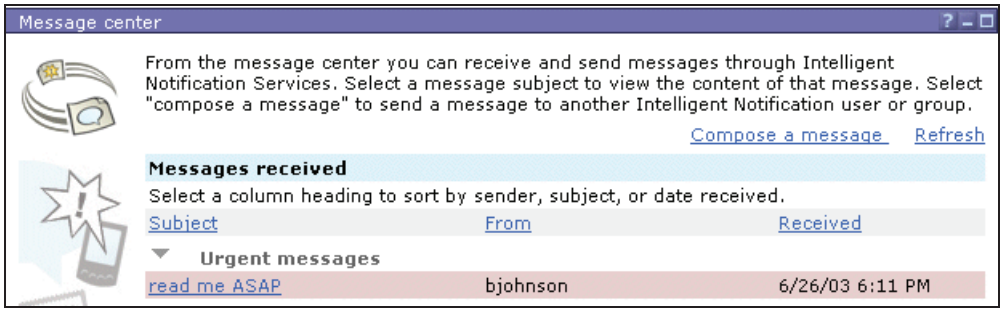


Figure 25-87 Received message center message



Figure 25-88 Received message center message content

25.9 INS enablement and configuration check list

As a summary, the following steps need to be performed when enabling INS and configuring INS in a WebSphere Everyplace Access server:

On the INS Server:

1. Run **updatePortletCfg** if in a distributed environment
2. Start INS administration server
3. Start INS host administration server
4. Start all INS servers
5. Enable INS users
6. Subscription configuration and starting
7. Content adapter configuration and starting
8. Gateway adapter configuration

In the INS user:

1. Delivery channel configuration
2. Subscription configuration

25.10 Viewing INS LDAP settings

The three sample scenarios presented in this chapter use the LDAP tree structure shown in Figure 25-89 on page 947.

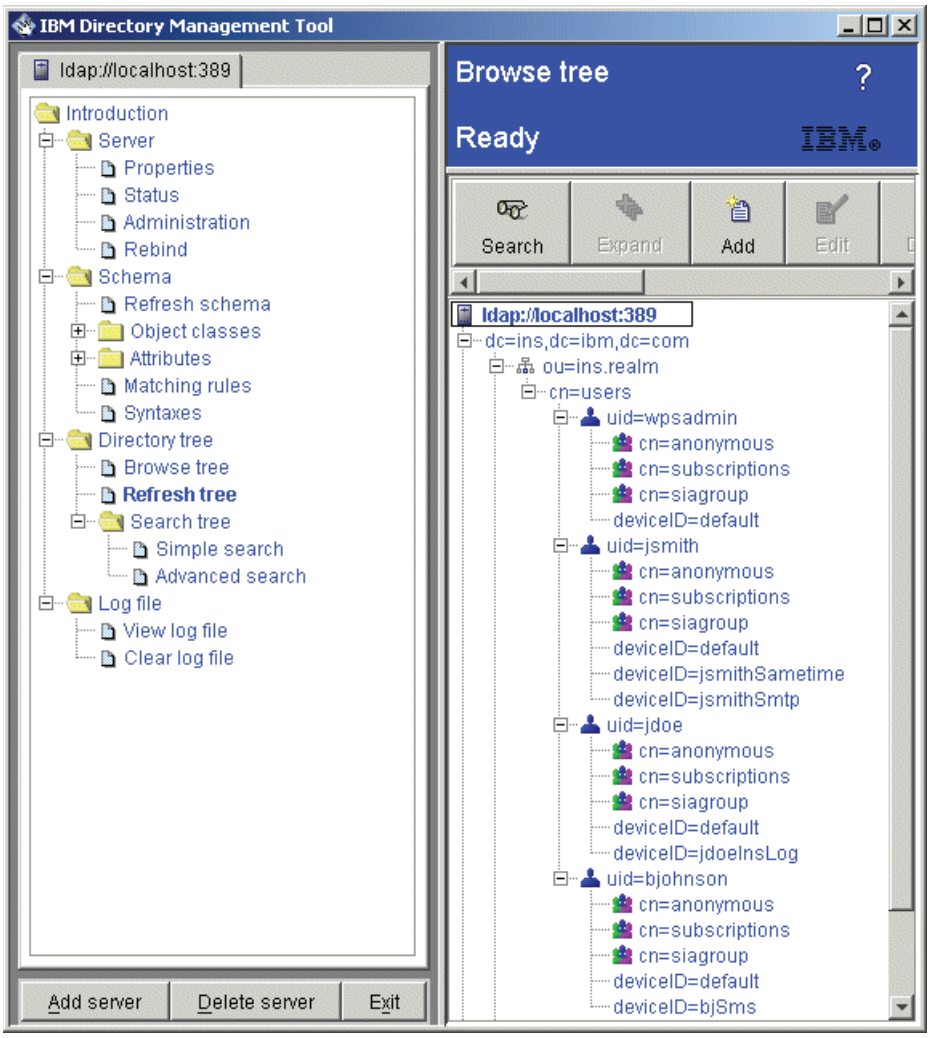


Figure 25-89 INS LDAP structure

The following may be seen in Figure 25-89:

- INS has its own LDAP suffix, and the suffix for INS is the hardcoded `dc=ins,dc=ibm,dc=com`, whatever the names in the WebSphere Everyplace Access LDAP suffix.

Note: This is the reason why when a user has been removed from the WEA `INSUsersGroup` group, you must select **Intelligent Notification -> Administration -> Remove User Preferences** and remove this user from the dc=ins,dc=ibm,dc=com branch.

- Under branch dc=ins,dc=ibm,dc=com -> ou=ins.admin -> cn=users, all user branches (including uid=wpsadmin) have at least four entries:
  - Three common names entries (cn=anonymous, cn=subscriptions, cn=siagroup)
  - At least one deviceID entry. The deviceID entries describe the delivery channels for that user.
    - All users have the default delivery channel defined by default.
    - User jsmith has also defined delivery channels jsmithSametime and jsmithSmtp.
    - User jdoe has also defined delivery channel jdoeInsLog.
    - User bjohnson has also defined delivery channel bjSms.
- The deviceID=default entry describes the default delivery channel, the message center, which is installed as a custom delivery channel. Its main values are:
  - deviceID=default
  - ibm-appDeviceAddress=jsmith@ins.realm
  - ibm-appProtocol=MessageCenter
  - ibm-appProtocolType=MessageCenter
  - ibm-appProtocolVersion=none

  The ibm-appDeviceAddress value will be copied in the From field of the notifications (look at the jsmith notification in Figure 25-68 on page 931, Figure 25-69 on page 932, and Figure 25-70 on page 932).

  The ibm-appProtocol* values have been built from property ibm-undDeviceAdaptorLink of the message center gw.properties file found in WebSphere Everyplace Access 4.3 server directory C:\WebSphere\INS\UNDGateways\RecentAlerts.

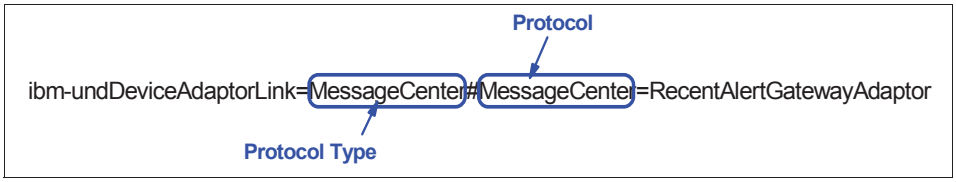


Figure 25-90 ibm-undDeviceAdaptorLink property for default delivery channel

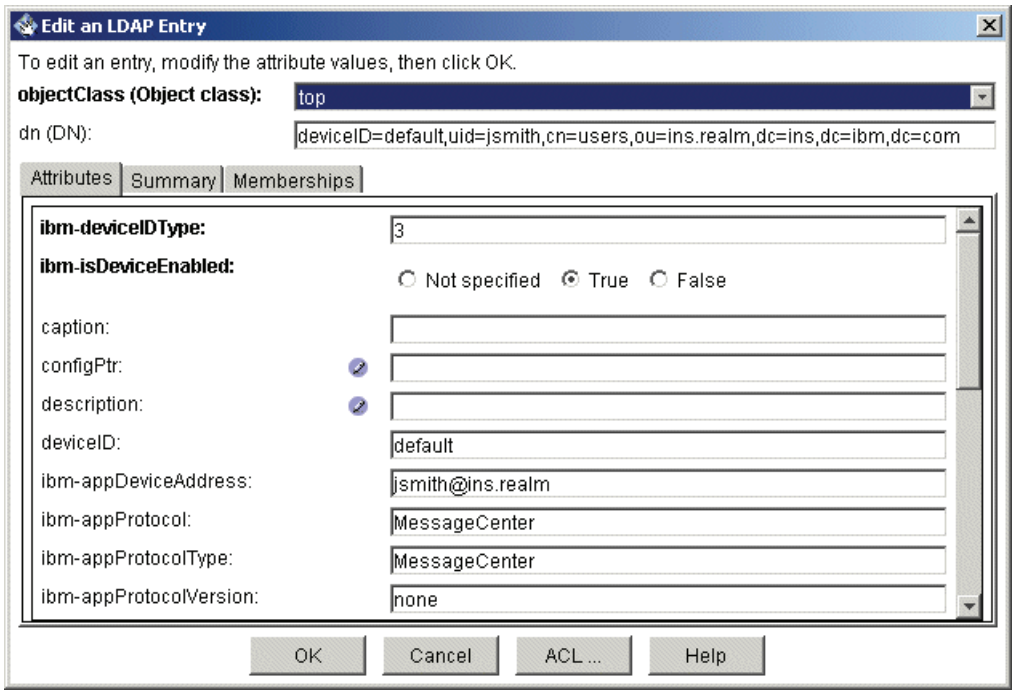


Figure 25-91 Message center delivery channel LDAP entry for jsmith

- Likewise:
  - The deviceID=jsmithSametime entry describes the built-in Sametime delivery channel for jsmith.
  - The deviceID=jsmithSmtp entry describes the built-in SMTP mail delivery channel for jsmith.
  - The deviceID=jdoeInsLog entry describes the custom INS log delivery channel for jdoe.

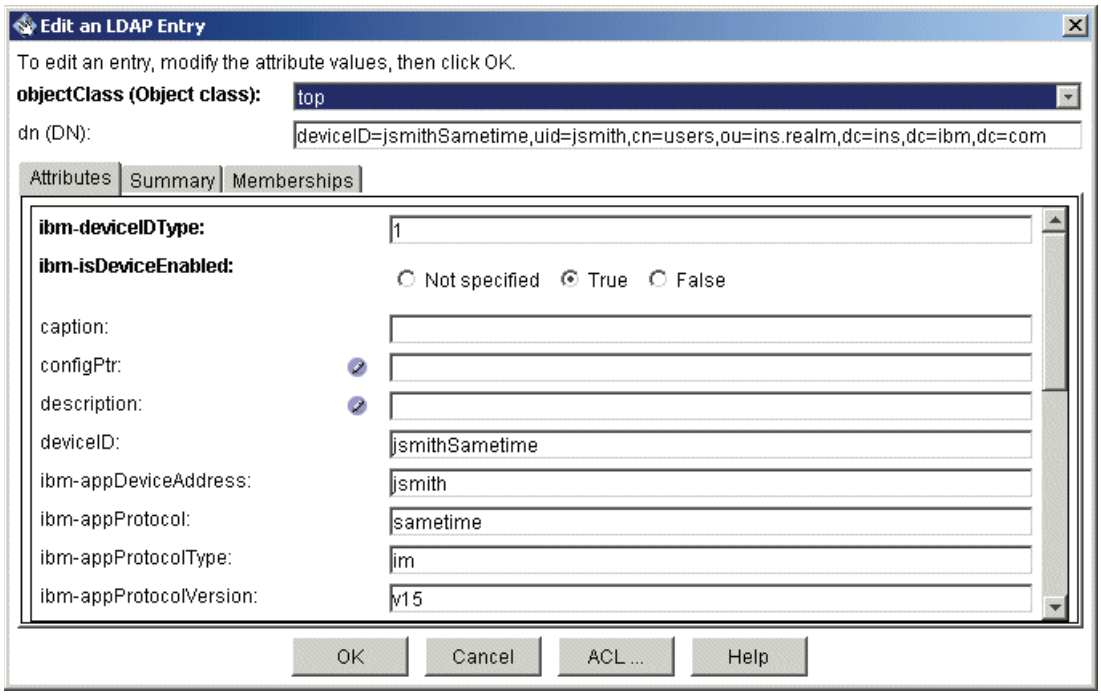


Figure 25-92 Sametime delivery channel LDAP entry for jsmith

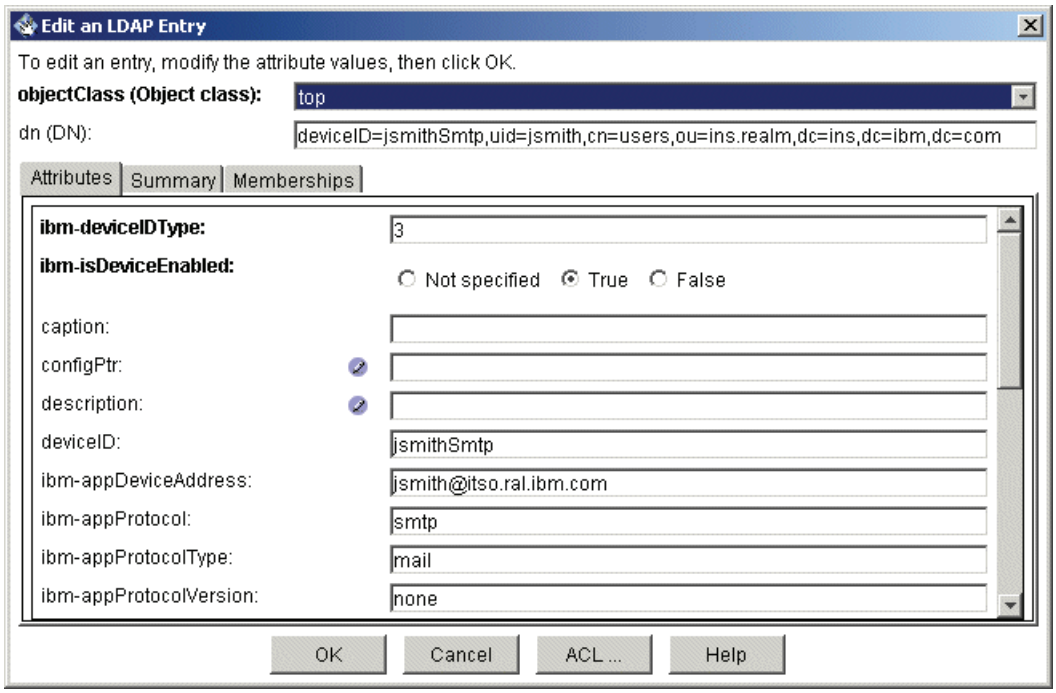


Figure 25-93 SMTP mail delivery channel LDAP entry for jsmith

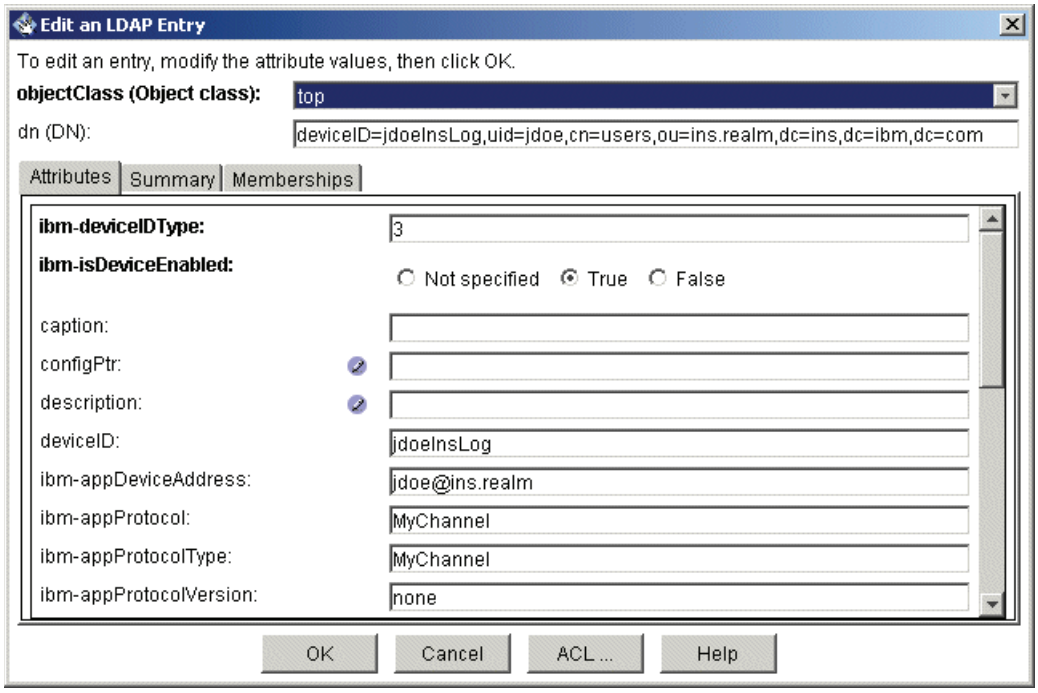


Figure 25-94 INS log delivery channel LDAP entry for jsmith

- The built-in delivery channels specific parameters may be located by expanding **dc=ins,dc=ibm,dc=com -> sys=SDP -> sys=und1_*hostname* -> cn=und adaptors**. See Figure 25-95 on page 953.

  The built-in delivery channel common parameters and custom delivery channel deployment location parameters are located by expanding **dc=ins,dc=ibm,dc=com -> sys=SDP -> sys=und1_*hostname* -> cid=common.**

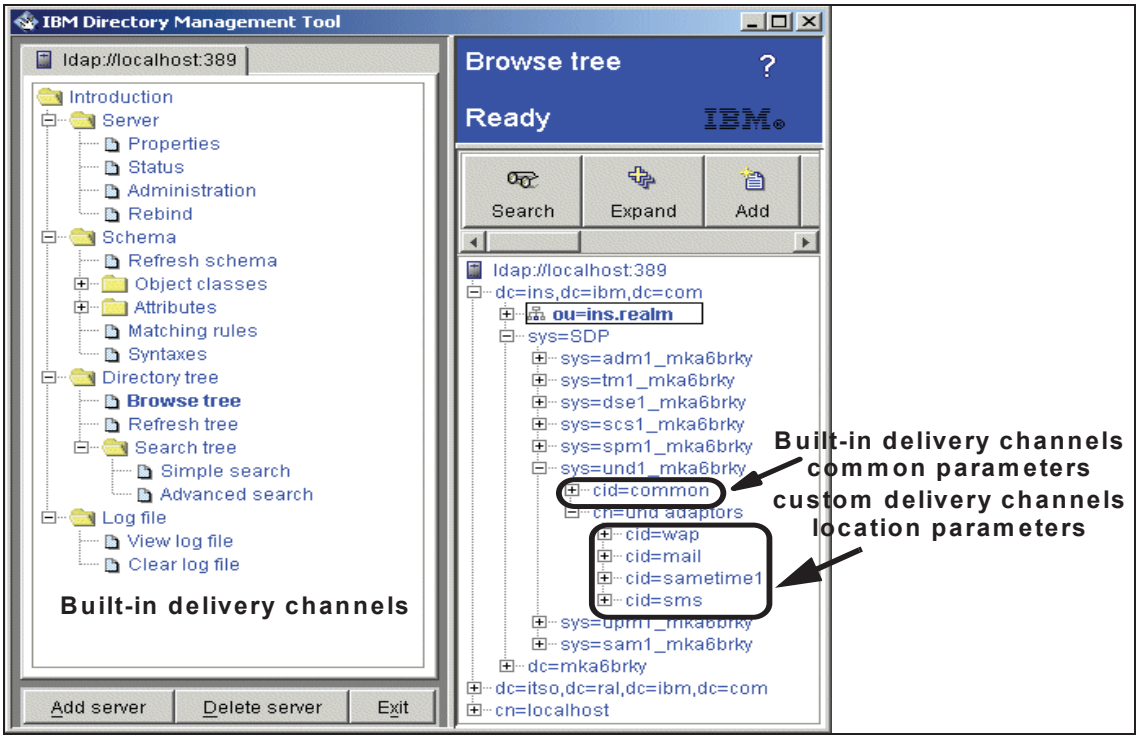


Figure 25-95 delivery channels shown in LDAP

25.11 Troubleshooting

This section provides useful information for troubleshooting INS runtime problems you may encounter.

25.11.1 INS log files

There are two kinds of log files with INS: the console and each INS server's log file. The console is the startHA DOS window.

Each server can be set its own log file Select **Intelligent Notification -> Administration -> Manage Servers** portlet. Clicking each server allows you to tune the log and trace settings, as shown in the example of the universal notification dispatcher settings shown in Figure 25-96 on page 954.

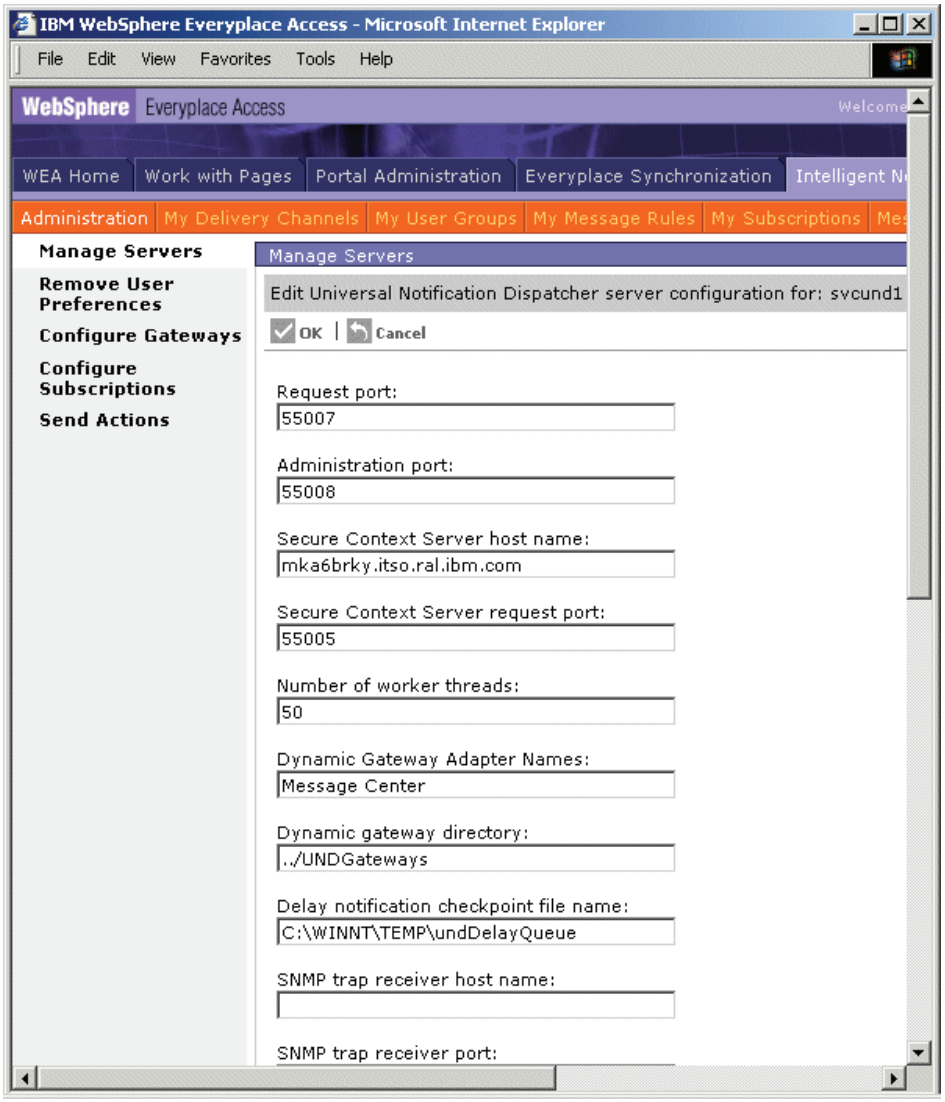


Figure 25-96 Universal notification dispatcher server configuration - part 1

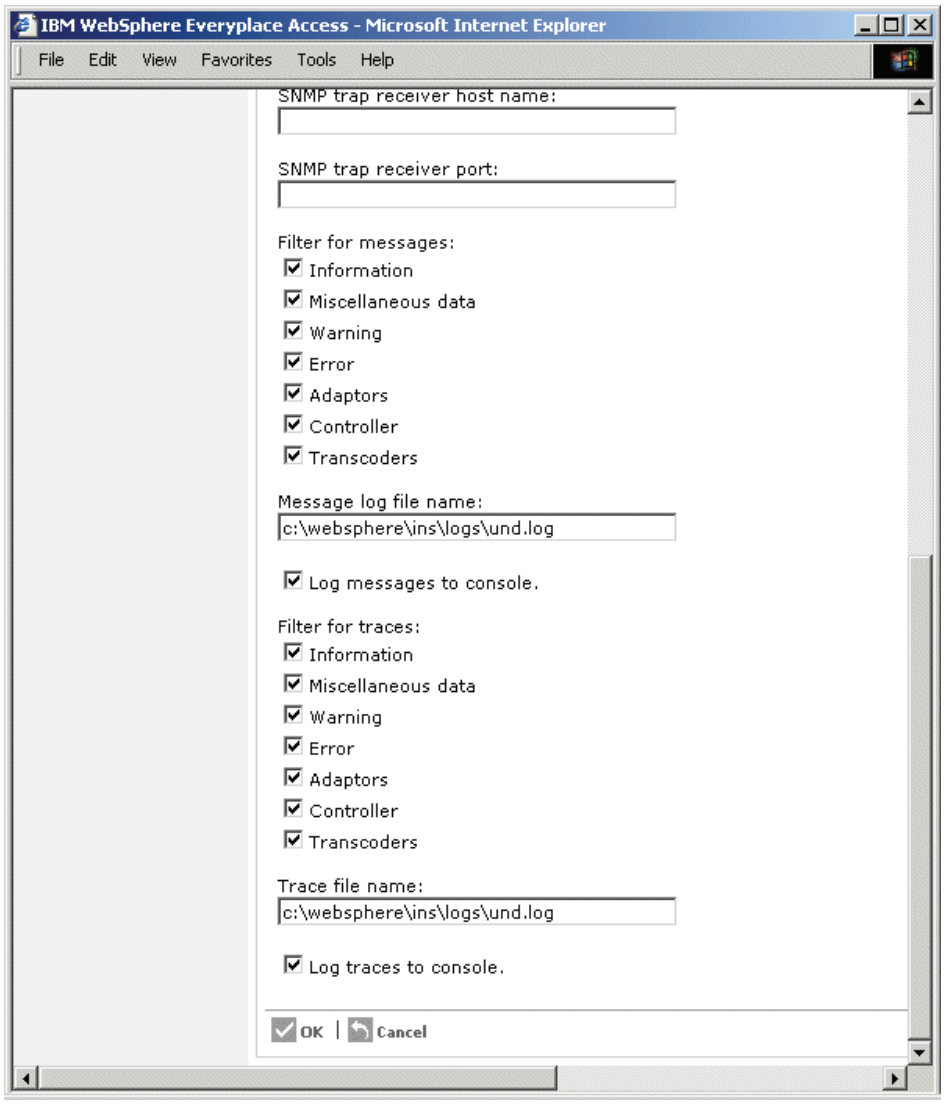


Figure 25-97 Universal notification dispatcher server configuration - part 2

It is advisable to store all the INS LOG files in a single location, such as C:\WebSphere\INS\logs.

When troubleshooting a given problem, once the failing server has been identified, it is advisable to log the traces of this server also in the console, so that the chronology of all the events that occurred may be seen.

25.11.2 Suggested actions

This section provides a list of common failures along with the associated suggested action.

All notifications do not work for a given user

In such case, this may mean that the LDAP data for this user may have been corrupted. LDAP data for a given user may be suppressed by selecting **Intelligent Notification -> Administration -> Remove User Preferences**.

One notification does not work for all users

This may be because the INS servers have been restarted since the last time the content adapter polling program (such as startNMAIL, startXMAIL, or startRSS) was run. In such case, re-start the content adapter polling program (startNMAIL and startXMAIL).

This may be because the polling interval for the given content adapter is too high. In such case, edit the content adapter properties file (for example C:\WebSphere\INS\apps\NMAILStartup.properties) and check that parameter pollingInterval (whose unit is the minute) fits your needs.

26

Server Initiated Actions (SIA)

This chapter describes the Server Initiated Actions (SIA) function in WebSphere Everyplace Access. The SIA feature relies on INS notification mechanisms to allow the server to send notifications to the user devices through their preferred delivery channels so that, under the covers, the device performs the actions it was requested by the server in the notification. As of now, the type of action that can be performed is device synchronization.

Important: To ensure you have the latest code, you should visit the WebSphere Everyplace Access support Web site at http://www.ibm.com/software/pervasive/ws_everyplace_access/support/to download the latest fix packs and code fixes. Apply these fixes prior to using the samples within this redbook.

© Copyright IBM Corp. 2003. All rights reserved.

26.1 Overview

There are two ways of using the SIA feature:

1. Users want to automatically synchronize their devices when a particular mail is received, for example when they receive an urgent mail from their manager. In such cases, only e-mail synchronization is requested.
2. The administrator wants a given user, or group of users, to have their devices synchronized. For example, they want a DMS job to run on their devices. The solicited synchronizations are:
   - DMS synchronization
   - DB2e synchronization
   - Offline synchronization
   - PIM synchronization
   - E-mail synchronization
   - All synchronizations

Note: The Server Initiated Actions function does not ensure proper delivery of the notification. Should anything prevent the notification to be delivered, it is discarded and not retried.

26.1.1 Description

SIA works on top of INS and consists of two main parts:

1. An SIA delivery channel. This special delivery channel receives notifications from INS as a regular delivery channel, then feeds transformed SIA notifications back into INS. This delivery channel has two components:
   - The SIA gateway adapter receives regular INS notification messages and translates them into the proper server-initiated actions format to be sent to the device.
   - The SIA user delivery channel portlet allows the user to configure the SIA notification channels.
2. An administrative portlet that allows an administrator to send SIA notifications to users or group of users.

SIA providers

SIA presently supports three sources for notifications:

- Notifications sent by the administrative portlet
- Notifications sent by the Notes content adapter
- Notifications sent by the Exchange 2000 content adapter

Note: Exchange Version 5.5 is not presently supported. However, it is expected to be supported in a future release.

SIA Delivery channels

SIA presently sends SMS or Sametime notifications and relies on companion products: WebSphere Everyplace Connection Manager to transport the SMS notifications and Domino Server to transport Sametime notifications.

SIA presently supports Pocket PC 2002 devices for Sametime notifications, and Pocket PC 2002 Phone Edition for SMS notifications.

Note: Palm support is expected to be included in a future WebSphere Everyplace Access release.

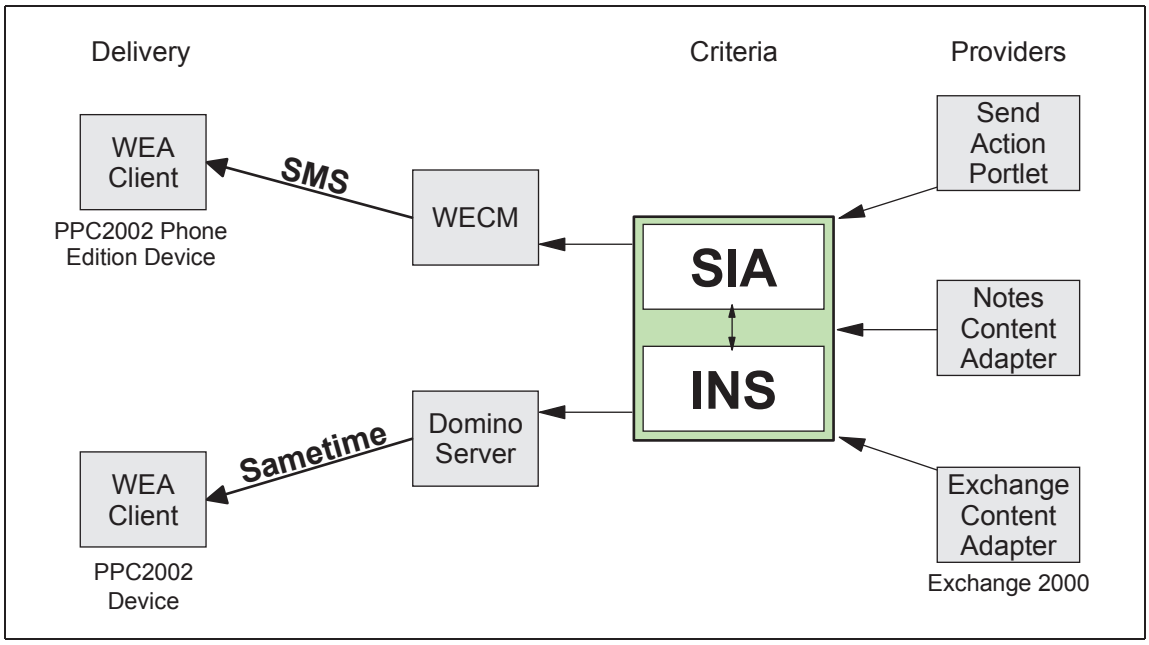


Figure 26-1 SIA functional overview

Note: It is possible to use an e-mail delivery channel for SMS SIA notification, provided the SMS addressing is given by an e-mail address.

SIA data flow

The different data flows occurring during an SIA notification are illustrated in Figure 26-2.

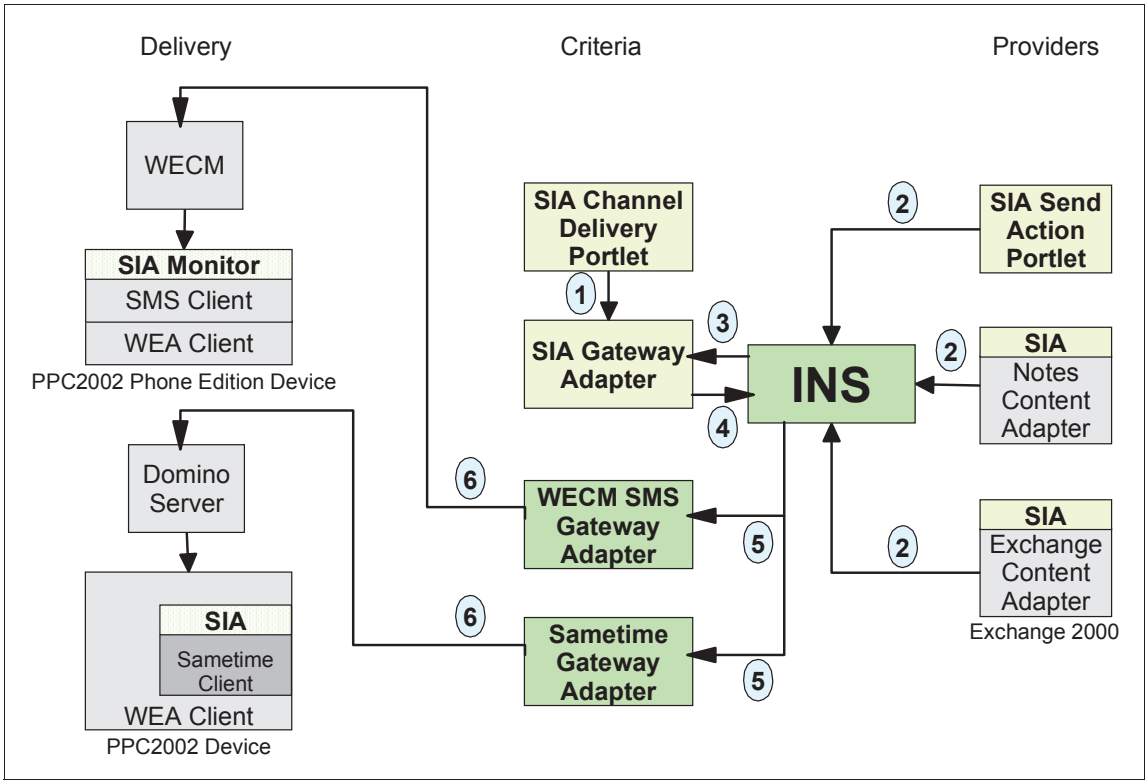


Figure 26-2 SIA data flows

1. The SIA gateway adapter gets configured by the INS user using the SIA channel delivery portlet, which allows setting of the SMS and/or Sametime delivery channels used by SIA.

Note: As opposed to a regular subscription, the SIA does not use any subscription portlet to be configured. All the information needed by SIA for a given user is set inside the SIA channel delivery portlet. Hence, this portlet looks like a combination of the regular subscription portlet with a regular channel delivery portlet.

2. The SIA notification gets triggered in two possible ways:
   - WebSphere Everyplace Access administrator uses the SIA send action portlet to trigger a solicited synchronization of the device.

- The INS user has configured an e-mail subscription with one SIA delivery channel, to trigger e-mail synchronization on the device.

  When such an e-mail arrives, the corresponding content adapter runs a specific SIA process to generate the notification to INS.

3. The notification is sent by the INS to the SIA delivery channel, which re-formats the synchronization request according to the target delivery channel, SMS, or Sametime.
4. The re-formatted synchronization request goes back to INS for delivery.
5. The appropriate delivery channel is triggered by INS to deliver the notification.
6. INS relies on WebSphere Everyplace Connection Manager to deliver SMS to the device, and on Domino Server to deliver the Sametime instant message to the device.

Client processing

- When the SIA synchronization request arrives on a device where the WebSphere Everyplace Access Client is not running, the request is discarded, therefore lost.
- When the SIA synchronization request is an SMS:
  - The device relies on a component called the SIA monitor, installed at WebSphere Everyplace Access Client installation time.

    Note: The SIA monitor is an application that registers towards the phone regular SMS client to receive the SMS. Since only one application is allowed to do such, an error message box will be displayed. In such case, the other application already registered towards the SMS client must be stopped and the device be reset to make the change effective.

  - The SIA monitor intercepts the SIA synchronization requests and forwards them to the WebSphere Everyplace Access Client for processing.
  - All the regular SMS are forwarded to the SMS inbox.
- When the SIA synchronization request is a Sametime instant message:
  - When the Sametime client is not running, the request is discarded, therefore lost.
  - The Sametime client intercepts the SIA synchronization requests and forwards them to the WebSphere Everyplace Access Client for processing.
  - All the regular Sametime instant messages are forwarded to the Sametime connect client.

26.1.2 Prerequisite settings

When the WebSphere Everyplace Access 4.3 client is being installed on a Pocket PC 2002 Phone Edition for SMS SIA usage:

- During the WebSphere Everyplace Access Client installation, be sure to check the **Server Initiated Action (only Phone Edition)** item.

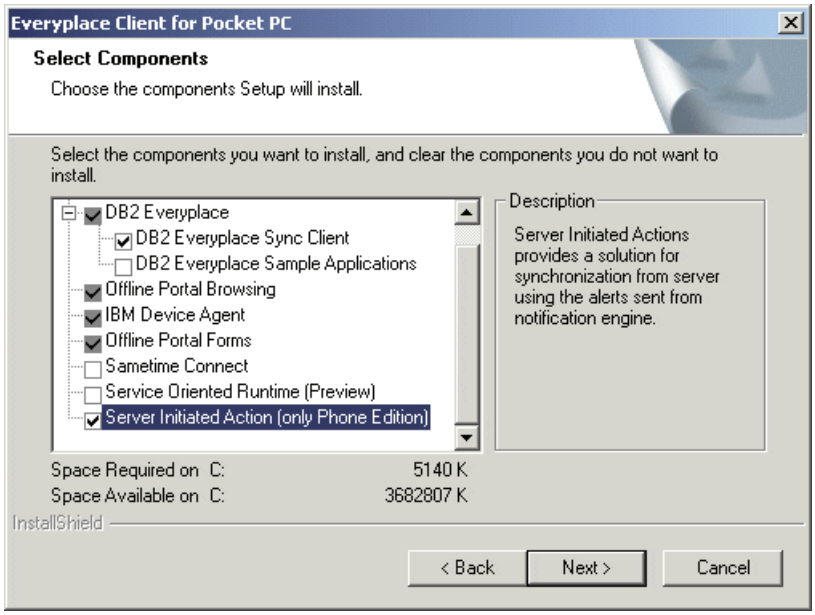


Figure 26-3 WebSphere Everyplace Access Client installation for SIA usage via SMS

- The device needs to be reset after installation. This is due to the SIA monitor, which needs to register to the SMS client.

Server settings

The following INS settings are required for proper SIA function:

- The e-mail subscription must have been configured for user-triggered server-initiated actions. Refer to 25.6.1, “E-mail subscription configuration” on page 908.
- The Notes and Exchange content adapters have to be started for user-triggered SIA. Refer to 25.6.2, “Content adapters configuration” on page 910.
- The Sametime and Messaging gateways have to be configured. Refer to 25.6.3, “Gateway adapters configuration” on page 917.

User settings

- One Sametime delivery channel must have been created for SIA usage via Sametime. Refer to “Sametime delivery channel” on page 926.

Note: The same Sametime delivery channel may be used both for regular INS notifications and for SIA notifications.

- One SMS delivery channel must be created for SIA usage via SMS. Refer to Chapter 25, “Intelligent Notification Services (INS)” on page 839.
- One e-mail subscription must have been created for user triggered SIA. Refer to Chapter 23, “PIM and e-mail synchronization with Exchange 2000” on page 809 for the Exchange example.
- One server-initiated actions delivery channel must have been created, that will use either SMS or Sametime or both delivery channels for SIA message delivery.

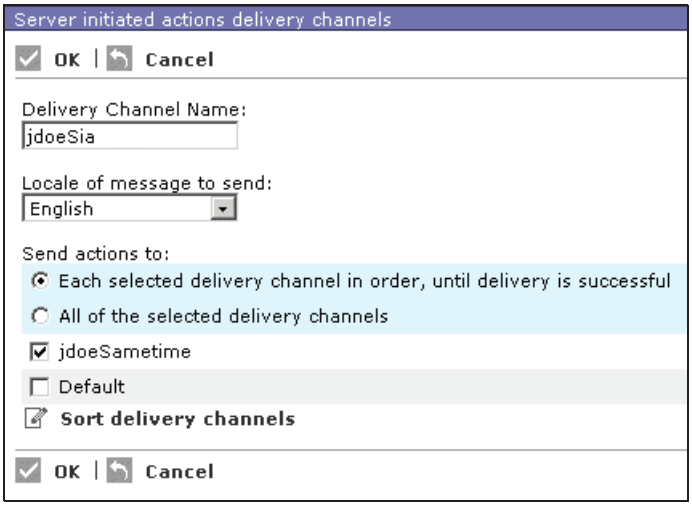


Figure 26-4 SIA delivery channel definition

Note: The locale field is here for historical reasons. It is expected to be removed in future WebSphere Everyplace Access releases.

- On the device, the WebSphere Everyplace Access Client must be started.

Important: When using Sametime delivery SIA, the Sametime client must be started within the WebSphere Everyplace Access Client.

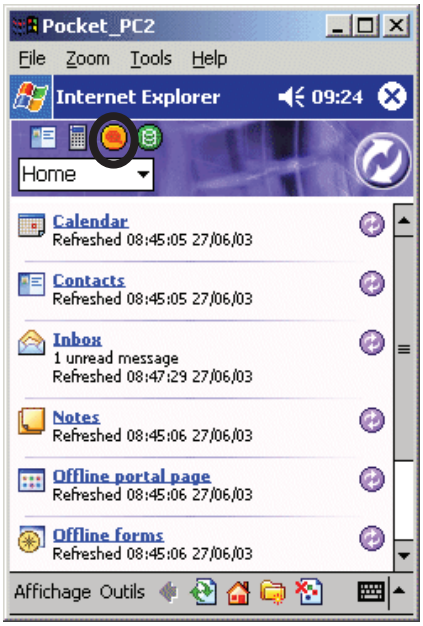


Figure 26-5 Click Sametime icon

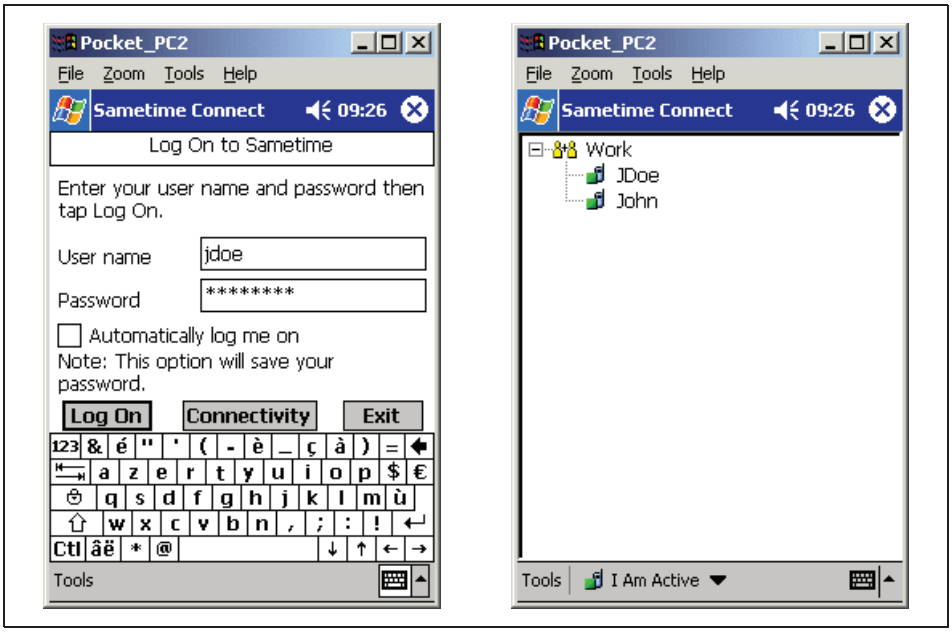


Figure 26-6 Log into Sametime and get connected

26.1.3 User defined server-initiated actions

We will illustrate the user SIA with a sample scenario where user John Smith wants to be automatically Sametime synchronized and Sametime notified whenever he receives an e-mail.

- The SIA delivery channel must use the Sametime delivery channel (refer to Figure 26-4 on page 963).
- The e-mail subscription must use both the Sametime and the SIA delivery channels.

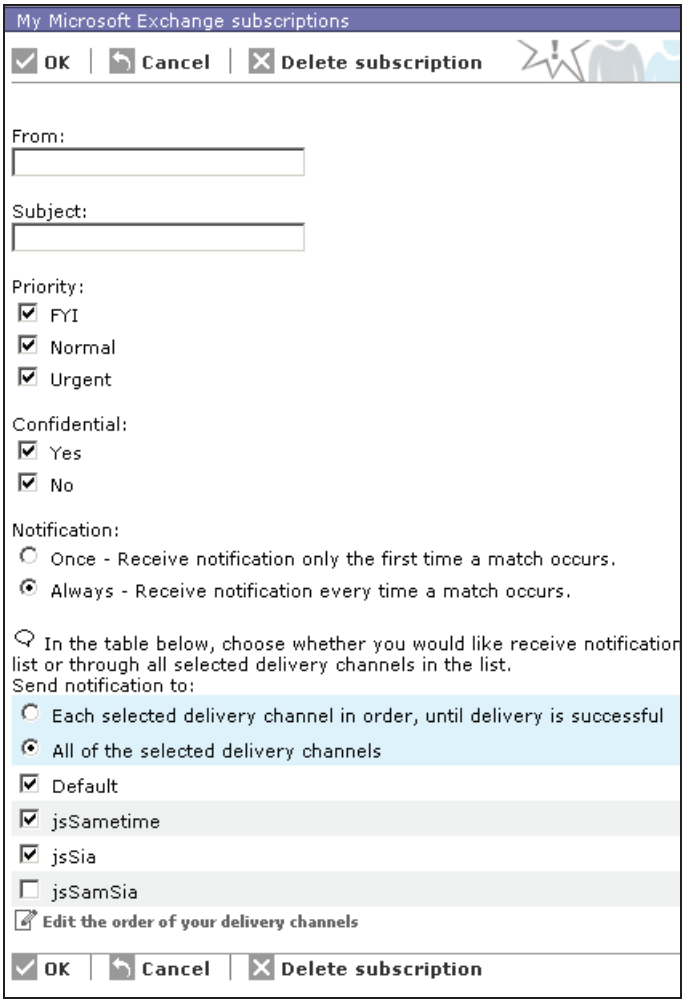


Figure 26-7 E-mail subscription for Sametime and SIA delivery

- To trigger the notifications, John Doe sends an e-mail to John Smith.

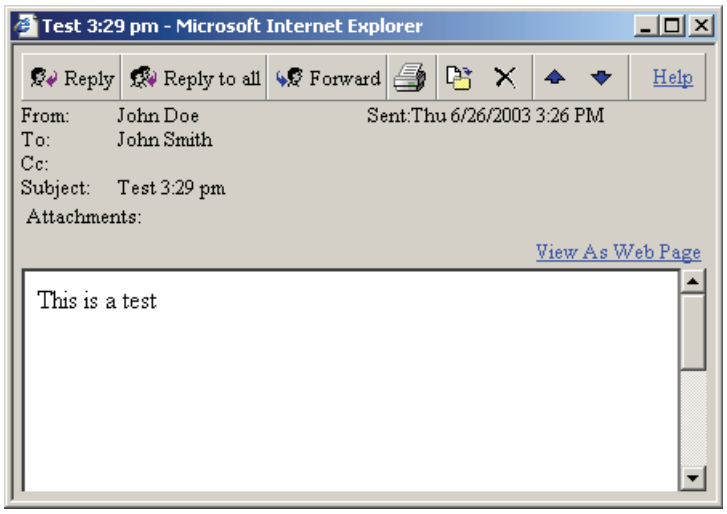


Figure 26-8 John Doe's e-mail to John Smith

- Then John Smith receives a Sametime notification that he received an e-mail.

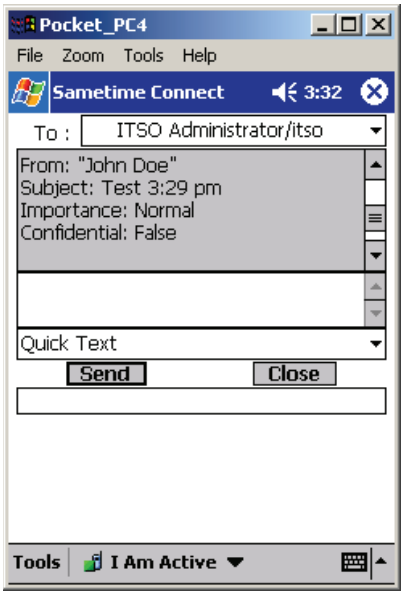


Figure 26-9 E-mail received Sametime notification

- Then, under the cover, the Pocket PC received the control Sametime message that makes it synchronize e-mail.

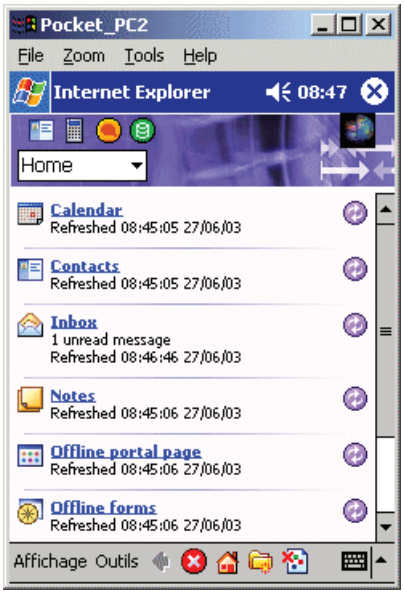


Figure 26-10 PPC synchronization triggered by user SIA

- Then John Smith can view the received message on its Pocket PC.

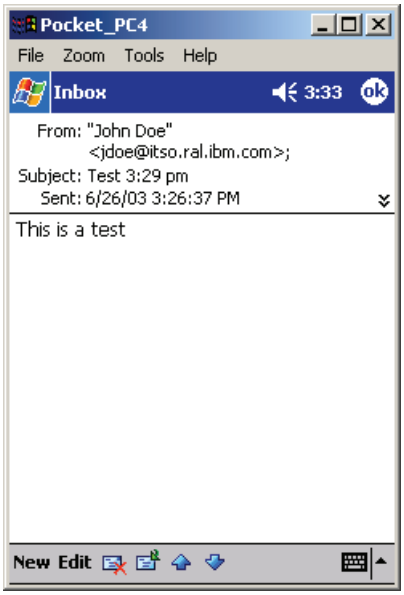


Figure 26-11 Received mail on PPC

26.1.4 Administrator-defined server-initiated actions

The administrator defines server-initiated actions by using the Send server initiated actions portlet.

The purpose of this portlet is to initiate actions to be performed by client devices. The portlet can be used to send an action to one or more users or groups of users.

If a recipient user has set up a delivery channel for server-initiated actions, they receive the action on the delivery channels which they have specified to receive actions from the Server-Initiated Actions portlet.

If the delivery channels are equipped with Server-Initiated Action client software, include in the WebSphere Everyplace Access 4.3 client, they use that software to perform the action that was sent from this portlet.

We will illustrate the administrator SIA with a sample scenario where the administrator sends a solicited e-mail synchronization to John Doe.

- The administrator, logged as `wpsadmin` into the WebSphere Everyplace Access server, clicks the **Intelligent Notification -> Administration -> Send server initiated actions** portlet to send the SIA to John Doe.

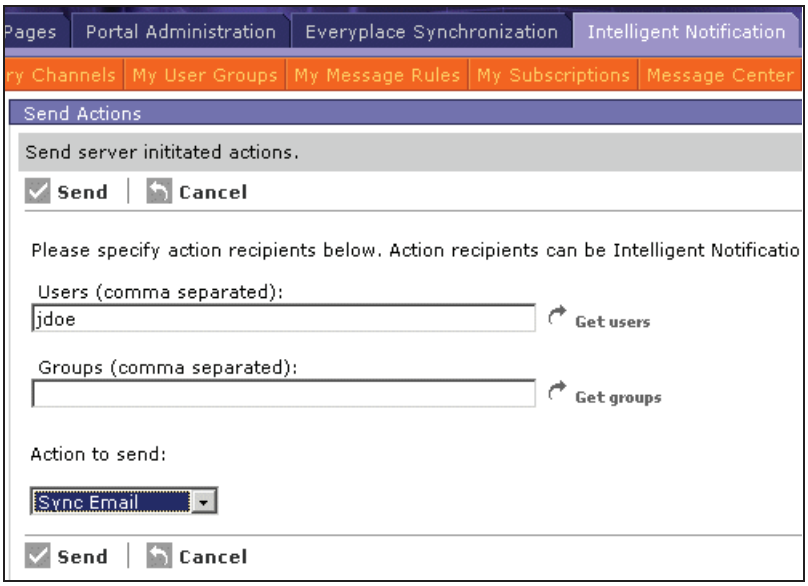


Figure 26-12 SIA send action

- Then, under the covers, the Pocket PC receives the control Sametime message that starts the e-mail synchronization process.

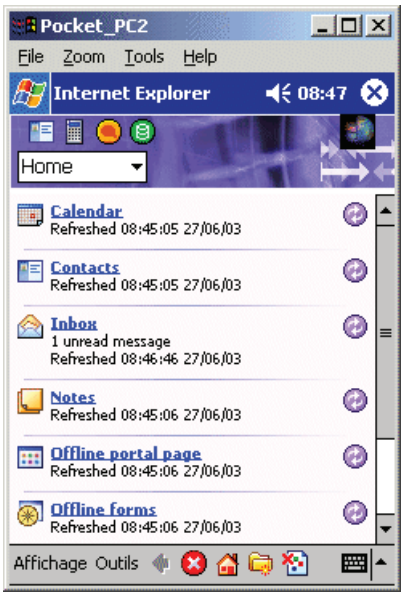


Figure 26-13 Pocket PC synchronization triggered by administrator SIA

26.2 Troubleshooting

When troubleshooting a given problem, once the failing server has been identified, it is advisable to log the traces of this server also in the console, so that the chronology of all the events that occurred may be seen.

Note: Since the Start Initiated Actions (SIA) function internally uses Intelligent Notifications Services (INS), see also 25.11, “Troubleshooting” on page 953 for details on INS troubleshooting.

SIA log files

SIA delivery channel gw.properties, file, found in the directory C:\WebSphere\INS\UNDGateways\ServerInitiatedActions, contains two properties that allow you to trace on a given file.

Example 26-1 SIA gw.properties trace properties

```
sia_LOG=true
sia_LOG_FILE=c:\\Websphere\\INS\\logs\\siaLog.log
```

27

Location Aware Services

This chapter describes the Location Aware Services provided in Everyplace Access. In this chapter we introduce you to Location Aware Services offered in Everyplace Access. Location Aware Services is a middleware solution that allows application providers to use the location-based services from different providers. Location Aware Services provide application developers an easy-to-use, yet powerful, application programming interface (API) to enable location awareness in their applications.

© Copyright IBM Corp. 2003. All rights reserved.

27.1 Introduction

The increasing use of pervasive devices gives us tremendous scope for providing value-added solutions to these users. This does not mean just making the existing applications available on these devices, considering the small screen size, low bandwidth, low processing capabilities and so on. Making these applications location aware can dramatically improve the adoption rate, since it would cater to specific needs of the mobile user. Also, a new class of applications can be provided to these users. Enabling location awareness in your applications can be complex and time consuming since this involves collaborating with several vendors and aggregating technologies from various sources.

Location Aware Services in Everyplace Access enables rapid development and deployment of applications that are location aware.

Everyplace Access provides a single standards-based API for the Location Aware Services. The reason for providing this API is that application developers can program to services and not to individual providers. The advantage of programming to services is that there is no risk of being tied to a single provider for services.

Device position is key to providing Location Aware Services. Everyplace Access does not provide the device position but depends on the carriers and the device. Location Aware Services can talk to any service provider or devices that have GPS to get a device's location.

The purpose of Location Aware Services is to provide:

1. Provider transparency
   a. Ability to program to a single API for location services.

      Location Aware Services provides developers of location-aware applications with a single API for common location services. The API defines call interfaces, the types of all objects passed in the calls, and the set of error codes that services can use to signal the application. Behind the Location Aware Services API, a set of service adapters translates the Location Aware Services API to the API of a particular service provider. As a result, the application is now independent of the actual provider of a location service.

   b. Transparent authentication of the application and/or user to the service provider

      In general, service providers require an application to authenticate before invoking a service. Authentication mechanisms differ from provider to provider, and credentials must be obtained, stored, and periodically renewed. Location Aware Services and the service adapters cooperate to

manage the authentication process, transparently to the application programmers.

c. Local/remote transparency

   Location service providers often make their services available both over the Internet and as software to be installed at the application provider's site. If installed locally, there will be no need to authenticate, and a different invocation mechanism can be used. Location Aware Services makes these differences transparent to the application. It is not aware whether or not the service is implemented locally or remotely.

2. Provider Differentiation

Service adapters implementing a common API mask differences between providers at the API level, but services from different providers will differ in other ways. For example, one provider's location-based directory service might offer a wider coverage region than another provider's, or one provider's mapping service might offer color images for WAP phones while another does not. Two services can differ in their quality of service, for example, accuracy of position or level of detail in maps. Location Aware Services offers applications a high-level way to differentiate between implementations on the basis of the capabilities that concern it. The application then can select the service implementation that offers the desired capabilities.

Location Aware Services also allows applications flexibility in their requests for services. For example, an application that supplies maps to WAP phones might be able to use images in either WBMP or BMP format, but would prefer WBMP. Location Aware Services enables applications to express not only requirements but preferences as well.

3. Dynamic Service Binding

Third-party location services often are offered as network services, and, like all network services, can become unavailable occasionally, whether due to network outage or a failure at the provider's site. Provider transparency enables Location Aware Services to bind another service dynamically to the application, allowing the application to continue without interruption. Location Aware Services selects a replacement service based on the requirements the application supplied in its original request for service.

Dynamic service binding also is used in a function sometimes referred to as location roaming. By nature, a location-based service is associated with a certain geographical area. Certain location-based services can be highly localized, that is, apply to a limited geographic area. As examples, a traffic-congestion reporting service might offer information for certain metropolitan areas, or a directory service might be specialized for a particular city. Therefore, a location-based service that is valid for one location might not

be valid for another. If a user of a location-based service moves to another area, that service might eventually become invalid.

27.2 Architecture

Figure 27-1 illustrates the architecture of the Location Aware Services function provided by WebSphere Everyplace Access.

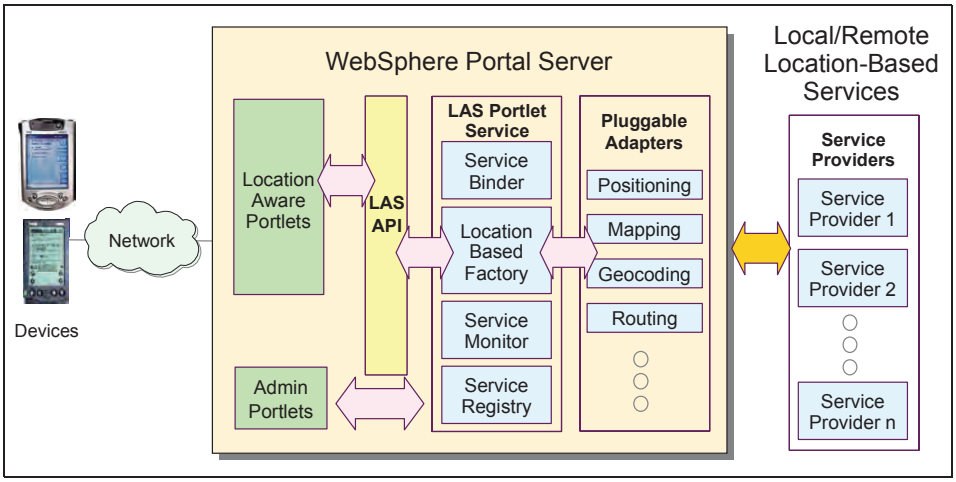


Figure 27-1 LAS Architecture

Location Aware Services provide a single API for location services. In addition to defining service interfaces, the API defines classes for representing locations, parameters for requesting maps, route information, and so on.

Location Aware Services currently support the following services:

- **Geocoding** provides the geographical coordinates for a provided address.
- **Reverse-geocoding** provides the nearest address to provided geographical coordinates.
- **Mapping** provides a map of a given location.
- **Routing** provides path from one provided point to another.
- **Directory** provides points of interest near a given location.

27.2.1 Service adapters

Applications use the different services through service adapters. Location Aware Services masks the service adapters from the application developer and provides a single interface to program.

The role of a service adapter includes:

- Interface translation: Translates the interface provided by the service provider to a Location Aware Services interface.
- Service invocation and error handling: Adapters invoke services depending on the service provider. It could for example be through TCP/IP Sockets, HTTP requests, and so on. Adapters also act like RPC stubs serializing and deserializing objects used by location services. Adapters insulate the application developer from the service protocols.
- Authentication with service providers: Adapters have to handle the authentication mechanism as required by the service provider.

In order to enable services through Everyplace Access, service providers have to write their own adapters. Adapters must implement specific interfaces provided by Location Aware Services.

Adapters for the following service providers are already provided with Everyplace Access:

- Webraska
- MapInfo
- Go2Map
- LIF
- IBM Location Server

27.2.2 LAS portlet service

The core functionality of the Location Aware Services are handled by the portlet service, which comprises several components and is the core component in the LAS architecture.

A portlet service is a discoverable extension to the Portal functionality. The reason behind implementing Location Aware Services as a portlet service is that the life cycle is managed by the Portal and does not have container restrictions placed on portlets. Application portlets can query the container for Location Aware Services and use the service without knowing the implementation or concern themselves with its life cycle management.

To understand more about portlet services, see for example *IBM WebSphere Portal V4 Developer's Handbook*, SG24-6897.

The function of the Location Aware Services portlet is:

1. Provide appropriate service to the application based on the preferences given by the application through the properties file. This functionality is taken care by the Location Service Factory.
2. Determine service failures based on service failover policies and take appropriate action. This functionality is handled by the service binder object.
3. Determine the ordering of service providers who are offering the same service. If Location Aware Services could not determine which service provider to use, then the order set by the administrator for the services is used.

Location Service Factory

As the name implies, the role of Location Service Factory is to create instances of Location Aware Services adapters when requested. The request for adapters can come from two sources: applications, via the service binder objects and internally via service monitor object, when it decides that the particular service adapter is needed.

The Location Service Factory makes use of the parameters supplied to determine which service provider adapter must be instantiated. For example, for map services, the parameters might include the coverage region, the format of the maps, etc. The Location Service Factory based on these parameters provides the best fit service provider. If there is a conflict due to several providers providing the requested features, then the order set by the administrator is taken.

Service Binder

Service Binder objects provide the point at which Location Aware Services mediates access to location services. To the application, these objects offer the interfaces of specific location services. Internally, they make use of Location Service Factory functions to instantiate adapters. Service Binder also handles service failure based on failover policies and replaces services.

Service Monitor

This monitors the status of all the current services and notifies the Service Binder if any of the service fails so that it can be replaced by other service providers.

Service Registry

The Service Registry is a repository of service descriptions and other service data, including authentication credentials. Service descriptions contain information Location Aware Services required to manage the service, including a unique name (within the Location Aware Services installation) and the name of

the Java class implementing the service adapter, as well as information describing the capabilities of the service.

27.2.3 Administration portlets

Location Aware Services provides an administration portlets through which services can be managed. Administration portlets allow you to do the following tasks:

- Add, remove, enable, disable service providers
- Modify order of service availability
- Configure properties for service configuration profile
- Configure properties for service providers configuration profile

27.3 Installation, configuration and verification

In this section we provide information about installation, configuration, and verification of Location Aware Services.

27.3.1 Installation

Location Aware Services can be installed as a part of the Everyplace Core Services. User Everyplace Access 4.3 setup manager to install the Location Aware Services. Installation of the Everyplace Services require an existing Portal server.

If you are using a remote DB2 database, then you will need to create a Location Aware Services database with the schema name of Atlas. Also, ensure DB2 JDBC service is running on the DB2 server.

27.3.2 Configuration

The configuration of Location Aware Services can be done through Administration portlet and the Server settings portlet. See 27.4, “Administration portlets” on page 979 for more information on configuring using the Administration portlet. See 27.6, “Application development” on page 996 for more information on configuring using the server settings portlet.

27.3.3 Verifying the installation

To make sure Location Aware Services is correctly installed, do the following:

1. Check if the Location Aware Services database is correctly created. Use control center to check if the LAS database was created. Connect to the LAS

database. Open the tables in the LAS database and check if tables are created with the LAS prefix having the schema of Atlas.

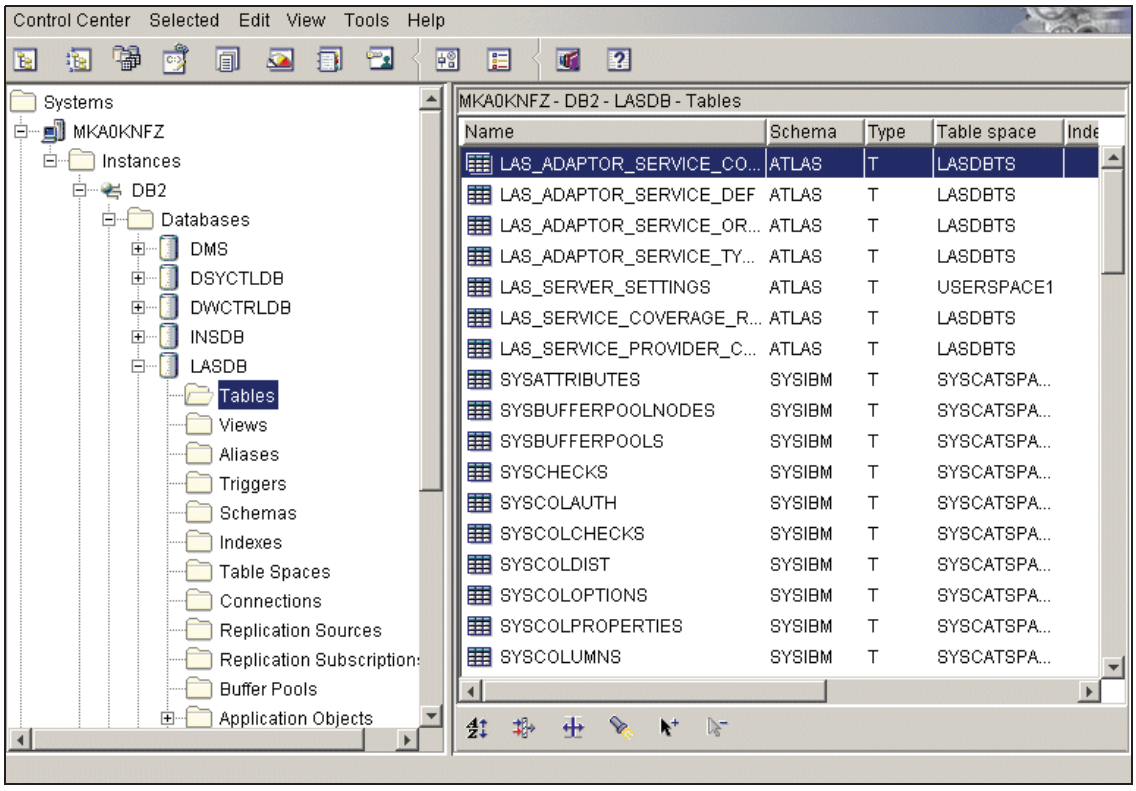


Figure 27-2 LAS database

2. Check if the Location Aware Services system property file was created. This file should be located at <WPS_HOME>/app/wps.ear/wps.war/WEB-INF/conf/atlas.properties.
3. Verify if the Location Aware Services registered itself as a portlet service. Open the <WAS_HOME>/lib/app/config/services/PortletServiceRegistryServices.properties file and check if the following lines are present:

```
com.ibm.atlas.portlet.service.LocationFactoryService =
com.ibm.atlas.portlet.service.impl.LocationFactoryServiceImpl
com.ibm.atlas.portlet.service.impl.LocationFactoryServiceImpl.factory =
com.ibm.atlas.portlet.service.factory.LocationPortletServiceFactory
com.ibm.atlas.portlet.service.impl.LocationFactoryServiceImpl.coverageRegio
n =
```

```
com.ibm.atlas.portlet.service.impl.LocationFactoryServiceImpl.providerName
=
com.ibm.atlas.portlet.service.impl.LocationFactoryServiceImpl.HTTPProxy =
```

4. Make sure the Location Aware Services installer has modified the <WAS_HOME>/lib/app/com/ibm/pvc/ent/cds/CommonDataService.properties file. Check if the following lines were added in the file:

```
#atlas application
CommonDataService.atlas=com.ibm.pvc.ent.cds.base.BaseCommonDataService
atlas.Version=LAS1.1.20030514005943
atlas.DataServiceMapper=com.ibm.pvc.ent.cds.atlas.AtlasDataServiceMapper
atlas.DataSourceInitialFactory=com.ibm.websphere.naming.WsnInitialContextFa
ctory
atlas.DataSourceProviderURL=IIOP://localhost:900
atlas.DataSourceName=jdbc/lasDS

#atlas server setting application
CommonDataService.atlasServerSetting=com.ibm.pvc.ent.cds.base.BaseCommonDat
aService
atlasServerSetting.Version=LAS1.1.20030514005943
atlasServerSetting.DataServiceMapper=com.ibm.pvc.ent.cds.atlas.AtlasServerS
ettingMapper
atlasServerSetting.DataSourceInitialFactory=com.ibm.websphere.naming.WsnIni
tialContextFactory
atlasServerSetting.DataSourceProviderURL=IIOP://localhost:900
atlasServerSetting.DataSourceName=jdbc/lasDS
```

5. Check if the atlas.jar (Location Aware Services Core) file is present in <WAS_HOME>/lib/app.

27.4 Administration portlets

Location Aware Services provides administration portlets to configure the different services. This section will show you how to administer the Location Aware Services.

Note: While configuring, if you navigate to a different page and come back, the changes will be lost.

27.4.1 Adding a custom service provider

Adding a new service provider involves developing a new adapter for the service provider and adding them into the Location Aware Services.

For example, to add a new service in Everyplace Access, follow these steps:

1. Select **Manage Service** page in the Location Aware Services page.
2. Click the **Add** button.

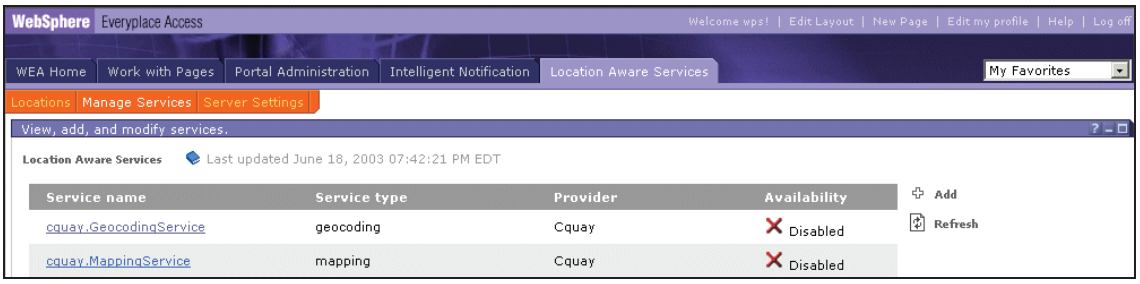


Figure 27-3 Adding new service provider

3. Enter details for this service:
   - Service Name.
   - Provider Name.
   - Service Type. The service type has to be one of the six service types supported by Everyplace Access.
   - Coverage Region type. If a polygon is selected as the coverage region type, you will need a customized Graphical Markup Language (GML) file. See the Open GIS Consortium Web site for more information on GML: `http://www.opengis.org`.
   - Select the **Enable service availability to portlets** check box to enable the service.

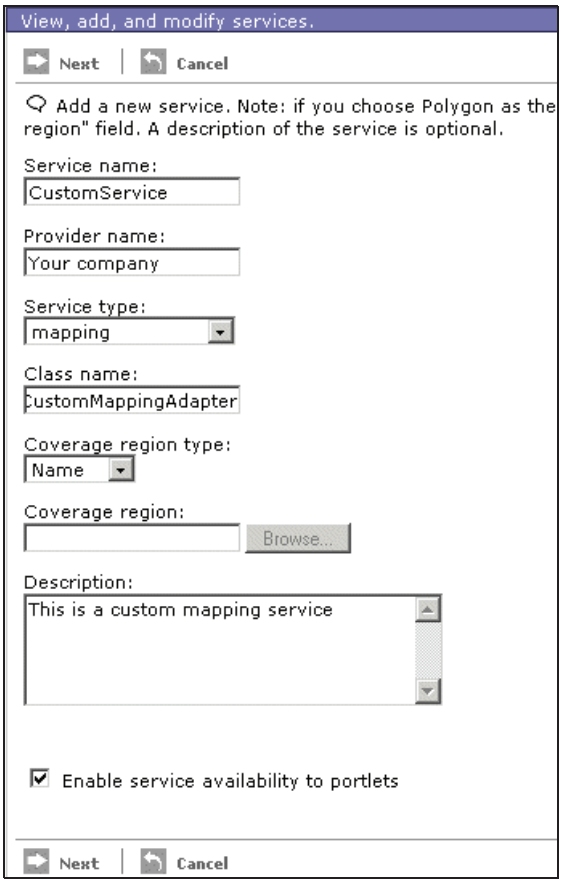


Figure 27-4 Configuration properties for adding new service

d. Click **Next**.

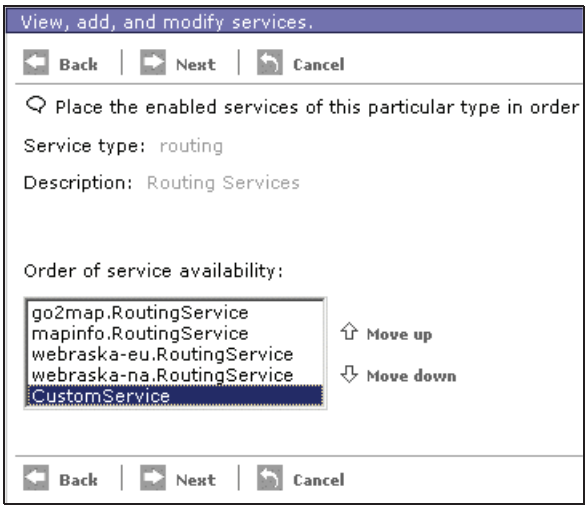


Figure 27-5 Order of service availability

e. Select the order of the availability. The Location Aware Services engine will search for the services based on this order. If you want your service to be searched first, move your service to the top of the list. Click **Next**.

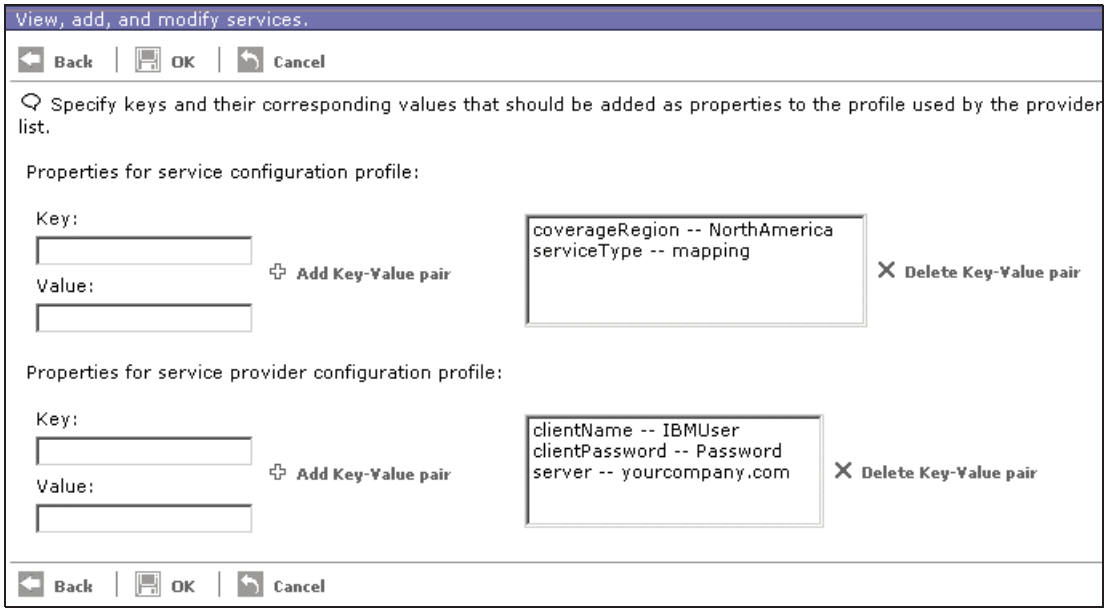


Figure 27-6 Service and service provider configuration properties

f. Add the properties for the service configuration profile. A service configuration profile properties are used to describe the characteristics of the service. The Location Aware application can use the profile to select the services that it wants to use. For example, coverageRegion, providerName, and serviceType are service parameters that can be entered here. The properties should be entered as key-value pairs. Enter the key and value and click Add key-value pair to add the properties. To modify the properties, the key-value pair has to be removed and added again.

g. Add the properties for the service provider configuration profile. These properties are used to initialize the adapter. For example, the URL of the service provider, clientName, clientPassword, and userLanguagePreference are service provider parameters. The properties are to entered as key-value pairs. Click **OK**.

27.4.2 Modifying the properties of a service and service provider

To modify the properties of a service provider, select **Location Aware Services** and open the **Manage Services** tab.

Click the service name you want to configure. You can modify the adapter class, coverage region, service provider configuration profile or the service configuration profile.

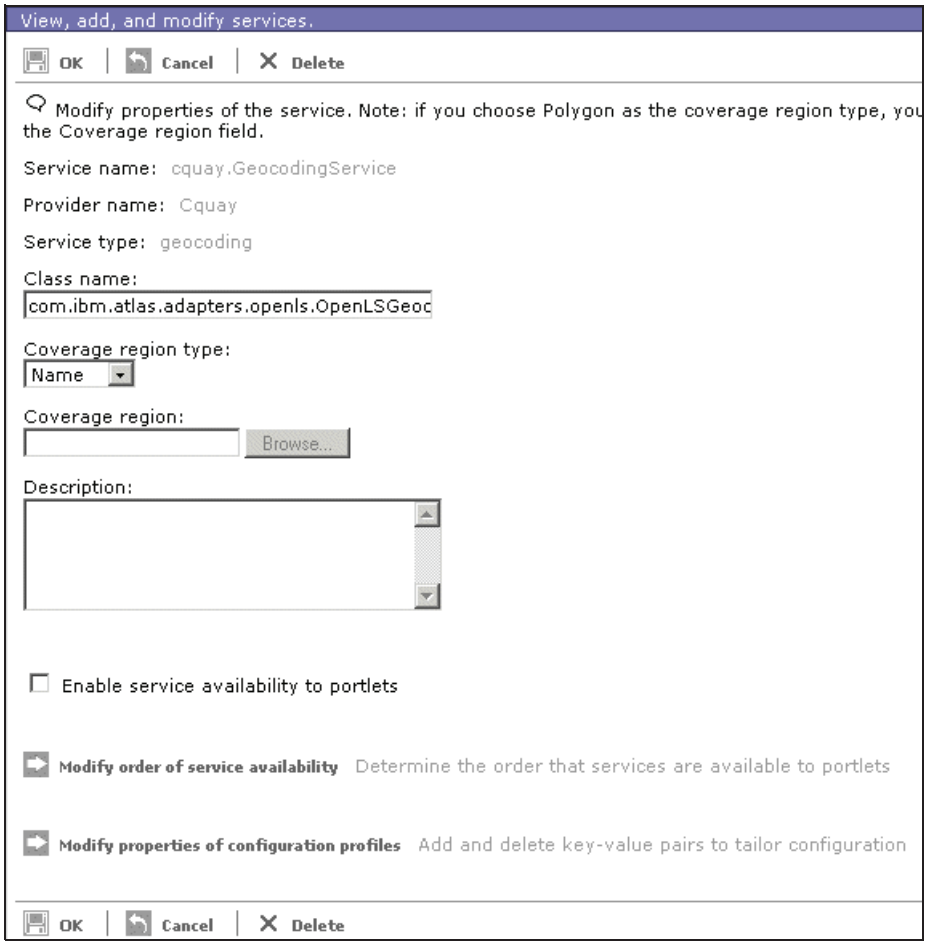


Figure 27-7 Modifying the properties

27.4.3 Enabling and disabling the service

If the service is enabled, then this service can be used by other portlets. To enable or disable a service, select **Location Aware Services** and open the **Manage Services** tab. Click the service that you want to enable or disable. This should bring up the window shown in Figure 27-7. Check or un-check the **Enable service availability to portlets** check box to enable or disable the service.

27.5 Sample Location Aware Services application

A sample Location Aware Services portlet is shipped along with Everyplace Access. This portlet demonstrates all the available features of Location Aware Services. A Location Aware Services adapter has to be configured before using the features of the portlet. In the example we have used Webraska as the service provider. In this section we will see the capabilities of location aware services.

Before the portlet can be used, at least one service provider must be configured for each service. The steps required to configure a service provider are shown in 27.4, “Administration portlets” on page 979.

Restriction: If you are in the middle of a task, such as retrieving a map, you will lose information you entered if you execute any of the following events:

- Navigate to another place or page before the task is completed.
- Clicking the Maximize or Minimize button on the title bar.
- Clicking the Edit button on the title bar.

27.5.1 Getting a map of a location

Navigate to Location Aware Services and open the **Locations** tab.

Enter the address of the location and click **Get Map**.

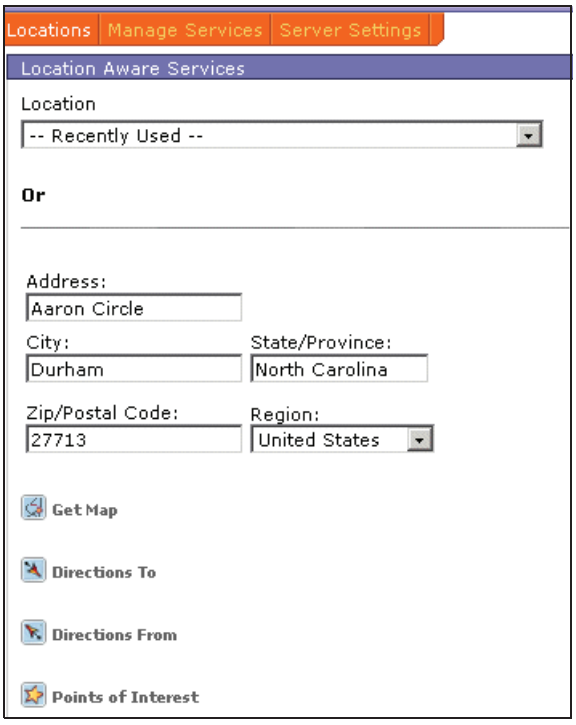


Figure 27-8 Retrieving a map

This should bring up a map as shown in Figure 27-9 on page 987. After the map is visible, you should be able to zoom in, zoom out, move and re-center the map.

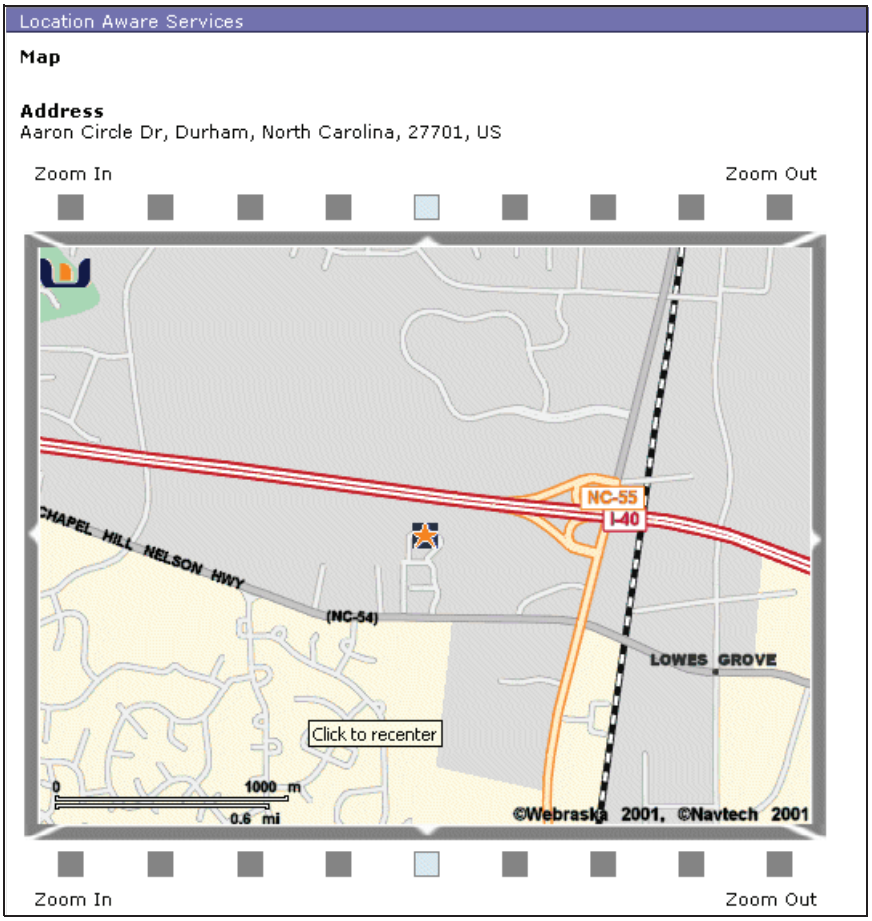


Figure 27-9 Map

27.5.2 Adding locations for persistence

Frequently used locations can be added to the portlet so they get persisted. To add locations, click the portlet's **Edit** button.

You can add a new location by clicking **Add new Location**. Or you can add one of the recently used addresses. We will add using the recently used address. Click **Add to My Locations**.

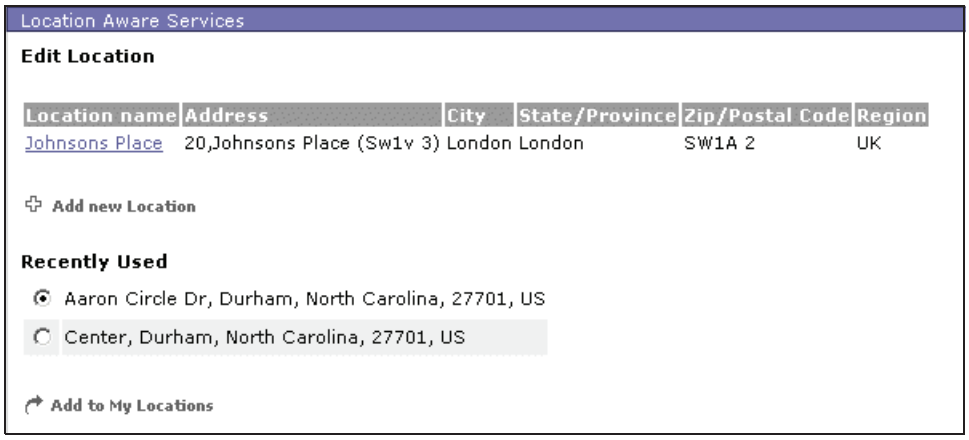


Figure 27-10 Adding new location

Enter the location name and click **Ok**.



Figure 27-11 Adding a new location

Go back to the portlet view mode and click the **Location** drop-down list. You should be able to see the location just added.

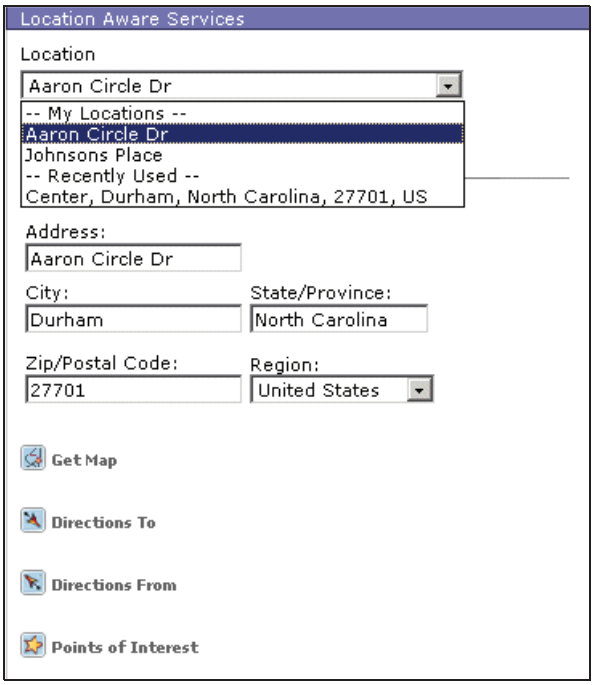


Figure 27-12 My Location list

27.5.3 Getting directions

The Location Aware Services portlet can be used to get directions from one point of interest to another. Also reverse directions between the same locations and a map showing the directions can also be obtained.

Enter the address from where you want the direction in the Location Aware Services portlet and click **Directions To.** This should bring up the window shown in Figure 27-13 on page 990.

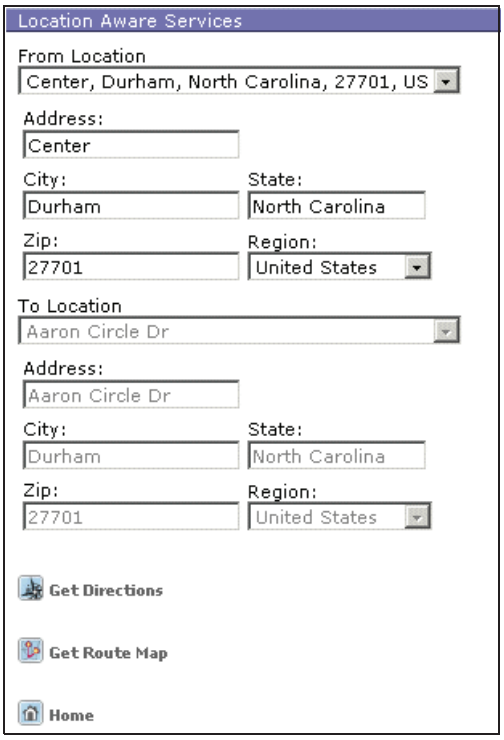


Figure 27-13 Directions

Enter the address to where you want the directions and click **Get Directions**.

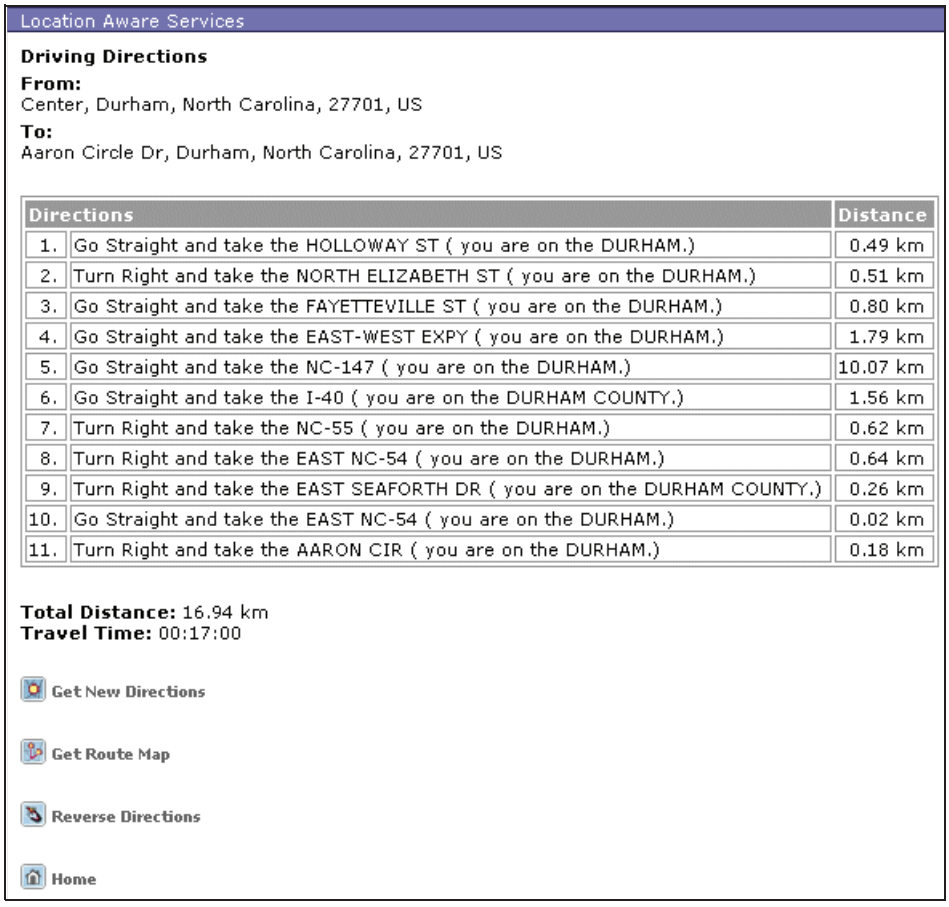


Figure 27-14 Driving directions

To get a map of the directions, click **Get Route Map**.

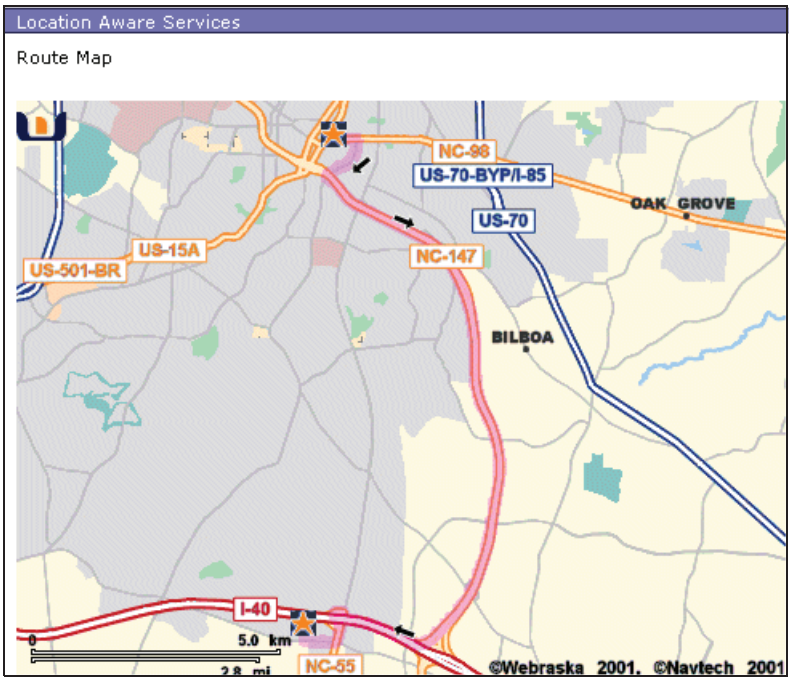


Figure 27-15 Route map

27.5.4 Getting points of interest near location

To get the points of interest near a location, enter the location address and click **Points of Interest**.

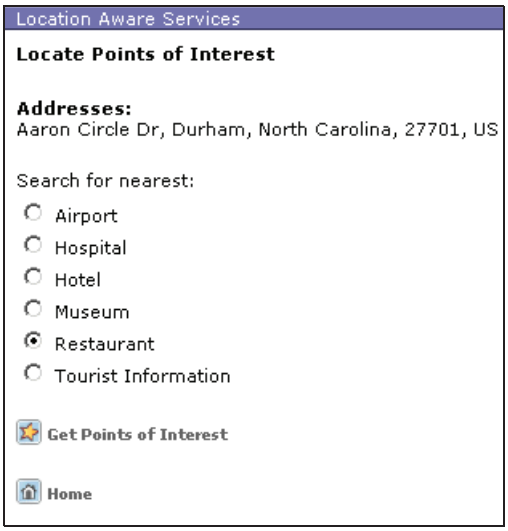


Figure 27-16 Points of Interest

Select the type of your interest and click **Get points of Interest**.

Location Aware Services

Back

Points of Interest

Addresses:
Aaron Circle Dr, Durham, North Carolina, 27701, US

Results		Distance from location
1. Chick-Fil-A	Residence Inn Blvd, Durham, North Carolina, 27701, US	Unknown
2. Oh Brian's	Durham, North Carolina, 27701, US	Unknown
3. Burger King	Durham, North Carolina, 27701, US	Unknown
4. It's A Weiner	Durham, North Carolina, 27701, US	Unknown
5. Kfc	Durham, North Carolina, 27701, US	Unknown
6. Wendy's	Durham, North Carolina, 27701, US	Unknown
7. Taco Bell	Durham, North Carolina, 27701, US	Unknown
8. Shor Szechuan Chinese Restaurant	Durham, North Carolina, 27701, US	Unknown
9. Pizza Hut	Durham, North Carolina, 27701, US	Unknown

Figure 27-17 Restaurants near the point of interest

27.5.5 Location Aware Services on devices

This sections shows the working of a Location Aware Services portlet on pervasive devices. As an example we have chosen a Pocket PC device.

1. In the Everyplace Client, click the drop-down list and select **Location Aware Services**. Click the **Locations** portlet.

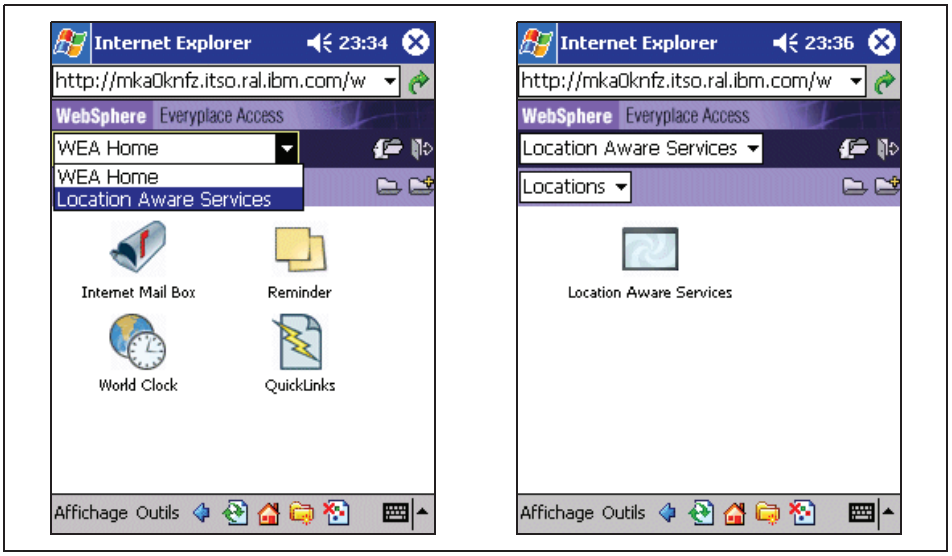


Figure 27-18 Location Aware Services on Pocket PC

Enter the location details and click **Get Map**.

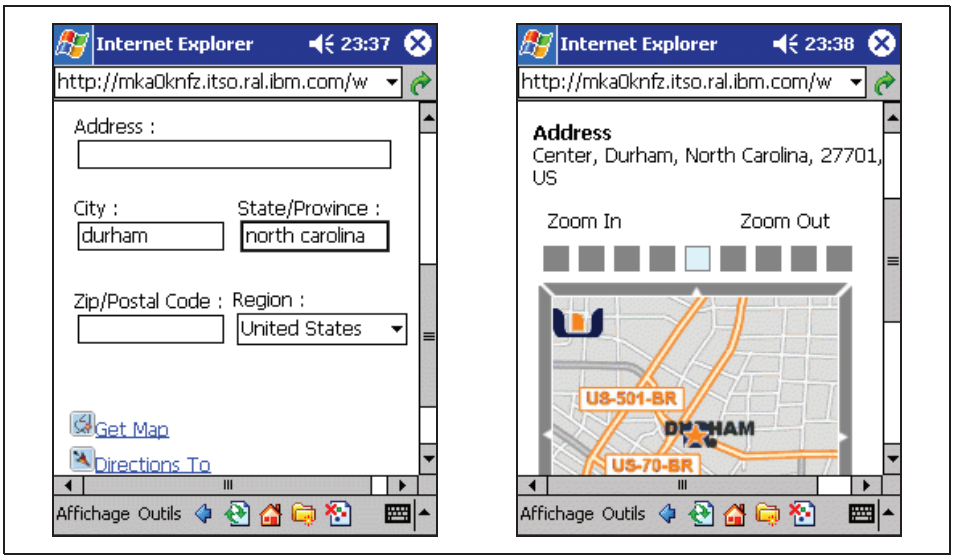


Figure 27-19 Requesting Map

Scroll down and select **Points of Interest**. Figure 27-20 shows the restaurants near the location.

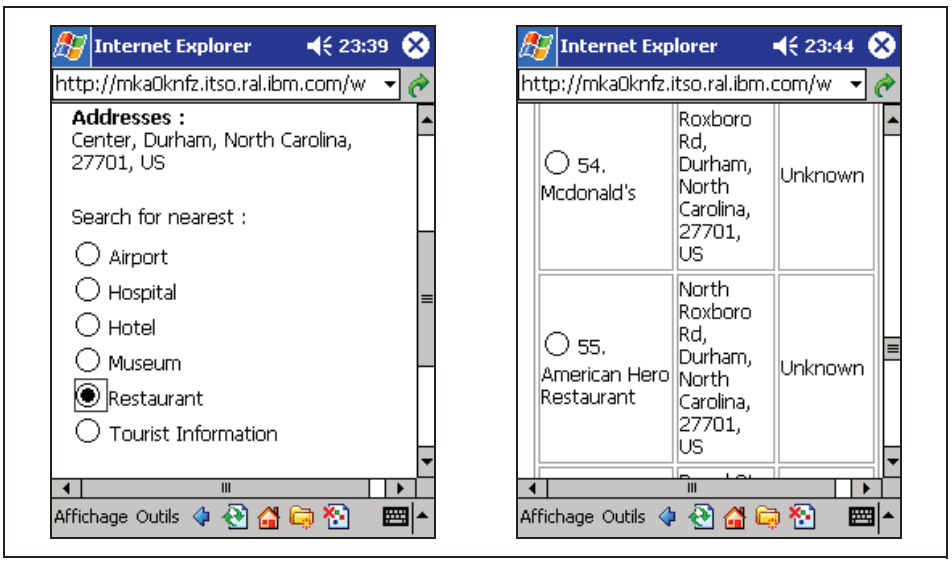


Figure 27-20 Points of interest

27.6 Application development

Location-aware applications can be developed as portlets or as stand alone Java Applications.

27.6.1 Location Aware Services API for application development

LAS provides an application developer API. The LAS Java API provides a flexible interface that allows simple access to straight forward services, while still providing access to the more sophisticated capabilities offered by some services. The documentation for the API can be located in the WebSphere installed directory <WebSphere>/LAS/Doc/API.

The following table lists the important package.

Package	Description
com.ibm.atlas	The Atlas application programming interface.
com.ibm.atlas.directory	This package consists of classes for making directory searches for points of interest and business places (“yellow pages”).
com.ibm.atlas.geocoding	Package consists of classes for specifying address, locations to get Geocoded address. This package also consists of classes for reverse geocoding a given Geocoded address. It consists of necessary data types for making requests and getting responses.
com.ibm.atlas.geometry	Package defines several geometrical shapes, points and coordinates used by the Atlas.
com.ibm.atlas.mapping	Package consists of classes for requesting maps, specifying the desired features of the map
com.ibm.atlas.positioning	Package consists of classes for requesting the position of the device. Consists of data types for making requests and holding the responses.
com.ibm.atlas.routing	Package consists of classes for requesting routes and necessary data types for making requests and getting responses.
oom.ibm.atlas.uom	(Units of measure) Package defines classes associated with unit of distance, speed, angles, time and so on.

27.6.2 Developing Location Aware Java application

In this section we will discuss how to develop Java application using the LAS API.

The following steps have to be performed before actually developing the application:

1. Add all the JAR files in <WebSphere>LAS/las/lib into your project.
2. Find the atlas.properties file and add it to your project.
   a. Modify the atlas.properties file. This is a global file for the Atlas Location Aware Services middleware. By default, Atlas loads this file from com.ibm.atlas.properties file, but may be pointed to directly by the system property ATLAS_PROPERTIES.
   b. Modify the servicePropertiesRoot to point to the folder where the LAS adapter properties files are located. LAS adapter properties files are located in the directory <WebSphere>/LAS/las/service_properties.
   c. Modify the trace and log level accordingly. Setting trace on and to high will slow down the system performances.
   d. Setting useConsoleLog or useConsoleTrace to true causes the log and trace messages to be written to the Portal server log and trace file, or the messages will be written in the file specified by logFileName and traceFileName properties.
   e. ClientName and Password are clear text values used to gain access to services when no corresponding properties are provided in individual service properties file.
   f. Atlas gives a default service failover policy file. But you could write your own service failover policy class, which implements the ServiceFailoverPolicy interface to define how the system should behave when a service fails. If you define your own service failover class for a particular service, this should be specified by the <service-type>FailoverPolicyClass.
   g. Specify the proxy the service adapters have to use by the HTTPProxy property.

Example 27-1 atlas.properties file

```
logDirectory=C:\\WebSphere\\LAS\\logs
logFileName=atlas.log
traceFileName=atlas.trace
registryClassname=com.ibm.atlas.system.PropertiesFileServiceRegistry
servicePropertiesRoot=C:\\WebSphere\\LAS\\las\\service_properties
useConsoleLog=false
useConsoleTrace=false
# Set to one of 'warn', 'info' or 'error'.
logLevel = warn
#Set to one of "high", "medium", "low", or "none".
traceLevel = none
```

```
clientName = dummy
clientPassword = dummy
defaultUserLanguagePreference = en
#<service-type>FailoverPolicyClass =
#
# Specifies the default failover policy class. If not set, defaults to
# com.ibm.atlas.system.DefaultServiceFailoverPolicy.
#
#defaultFailoverPolicyClass =
#Coordinate reference system classes
CRS.WGS84.className = com.ibm.atlas.geometry.WGS84ReferenceSystem
CRS.EPSG:4326.className = com.ibm.atlas.geometry.WGS84ReferenceSystem
HTTPProxy=
```

3. Modify the service properties and the properties file of the service provider. The properties file must be located in the directory as specified in the atlas.properties file. For the adapters provided along with Location Aware Services, the properties file are under the directory <WebSphere>/LAS/las/service_properties.

   Service properties file is used by the Location Aware Services Engine to select the appropriate adapter depending on the serviceType and serviceName (service provider). Also, this properties file specifies the location of the initPropertiesFile. Example 27-2 shows the service properties file for the mapping service provided by Webraska.

   The service provider properties file contains information required by the adapter. The parameters in the properties file depend on the service provider. Example 27-3 shows the Webraska adapter's properties file.

Note: To learn about the different parameters required by the adapters provided by Everyplace Access, see the InfoCenter (select **Location Aware Services -> Managing -> Administrative Services -> Properties for Service Provider Configuration Profile**.

Example 27-2 Mapping Service properties

```
serviceType = mapping
className = com.ibm.atlas.adapters.webraska.WebraskaMappingService
serviceName = webraska-na.MappingService
initPropertiesFile = webraska-na/init.properties
```

Example 27-3 Webraska service init properties

```
server = dummy.webraska.com
path = /gns
clientName = demo
clientPassword = demo
```

```
connectionTimeout = -1
#geolang = en
```

Initialize the service. This will invoke the Atlas core components, which will connect to the appropriate service provider using the property files. The code allocates two Location Aware Services used in the application: geocoding and mapping. The Location Aware Services provide access to multiple providers of the location services. Location Aware Services searches among the registered services for the one that fits the specifications provided in the properties file, such as the geography, and returns the first match. GeocodingService is defined in the package com.ibm.atlas.Geocoding. GeocodingService determines the location of the given address. MappingService is defined in the package com.ibm.atlas.Mapping. MappingService returns a map of the given location.

```
GeocodingService geocodingService = new GeocodingService(geoProps);
MappingService mappingService = new MappingService(mapProps);
```

The Construct and Address object is used by GeocodingService to obtain the location. The address is defined in the package com.ibm.atlas.GeoCoding. This class represents an address and is loosely based on RFC 2426 and ISO/ISE 11180.

GeocodingService returns the location of the given address. If there are multiple locations matching the given address, all the locations are returned. Multiple locations can be found, for example if the entered address was not unique. For example, the Geocoding address `City Center, Durham` gave multiple locations such as `Durham Oregon, Durham Washington, Durham Indiana` and so on.

Example 27-4 Constructing the address and getting the Geocoded location

```
//Construct the Address
Address addr = new Address();;
addr.setStreet(street);
addr.setCity(city);
addr.setState(state);
addr.setCountry(country);

GeocodedAddress[] locations = geocodingService.geocode(addr);
```

The given locations could be presented to the user to select the best match, or the application can choose to give the first match. In our application, the first location is chosen and the map is provided. We create a boundary of 5 square kilometers around the point of interest.

```
Point pt = geoAddress[0].getPoint();
```

```
Box bounds = new Box(pt.getCoordinates(), 5000.0, 5000.0);
```

A MappingOptions object contains the options for creating the map image, such as the length and height given in number of pixels. The format of the map data should be returned, such as GIF, JPG, and so on. The application requests a map image of 240x240 pixels and in the GIF format.

Example 27-5 Mapping options

```
MappingOptions options = new MappingOptions(240, 240, "gif");
```

A MapOverlay object is used to specify what information should be overlaid on the requested map. The MapOverlay object could be used to overlay a single or multiple point of interests on the map, routing information on the map, and so on. The application requests that the point of location be overlaid on the map.

Example 27-6 MapOverlay

```
MapOverlay [] overlays = new MapOverlay[] {new MapOverlay(pt)};
```

Finally the MappingService is used to request the map specifying the bounds, the mapping options, and the overlay. The Map object represents a map including image data, width, height, and format, and the geographical area covered by the map.

Example 27-7 Requesting the Map

```
Map map = mappingService.getMap(bounds, options, overlays);
```

The Map.getData() returns the map as a byte array. This can be used to render the map on a canvas, save it as a GIF file and render it as an HTML and so on.

27.6.3 Developing Location Aware portlet application

In this section we show you how to develop portlets using the Location Aware Services provided by Everyplace. We will develop a simple portlet that will fetch the map of the given address and display the map. WebSphere Studio Site Developer was used to develop the portlet.

1. Open WebSphere Studio Site Developer to add a new portlet application.

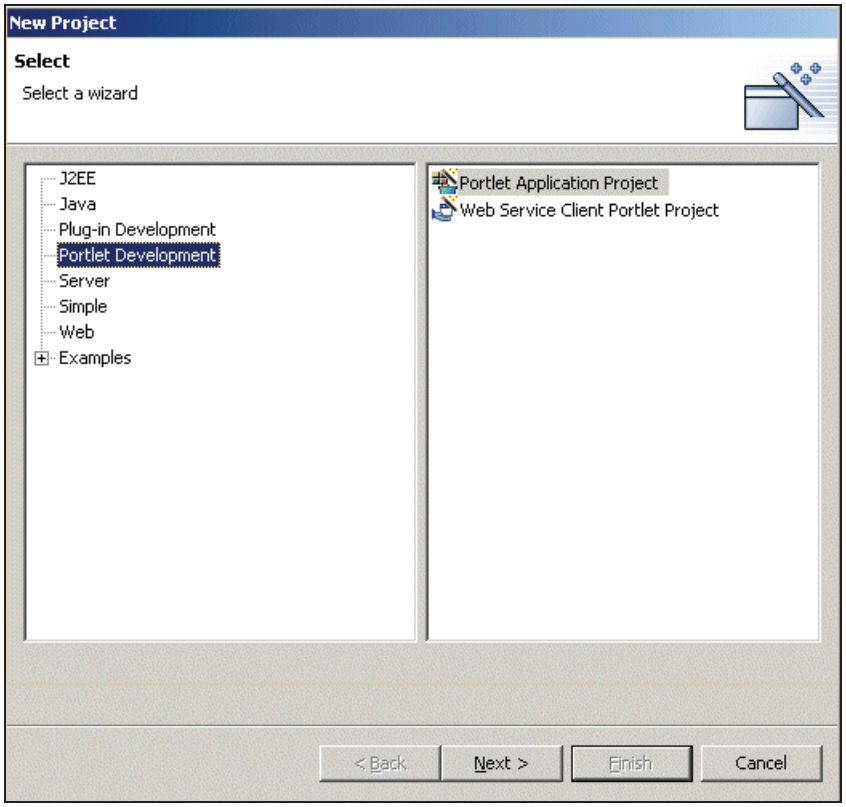


Figure 27-21 Create a new project

2. Enter the project name and click **Next**.

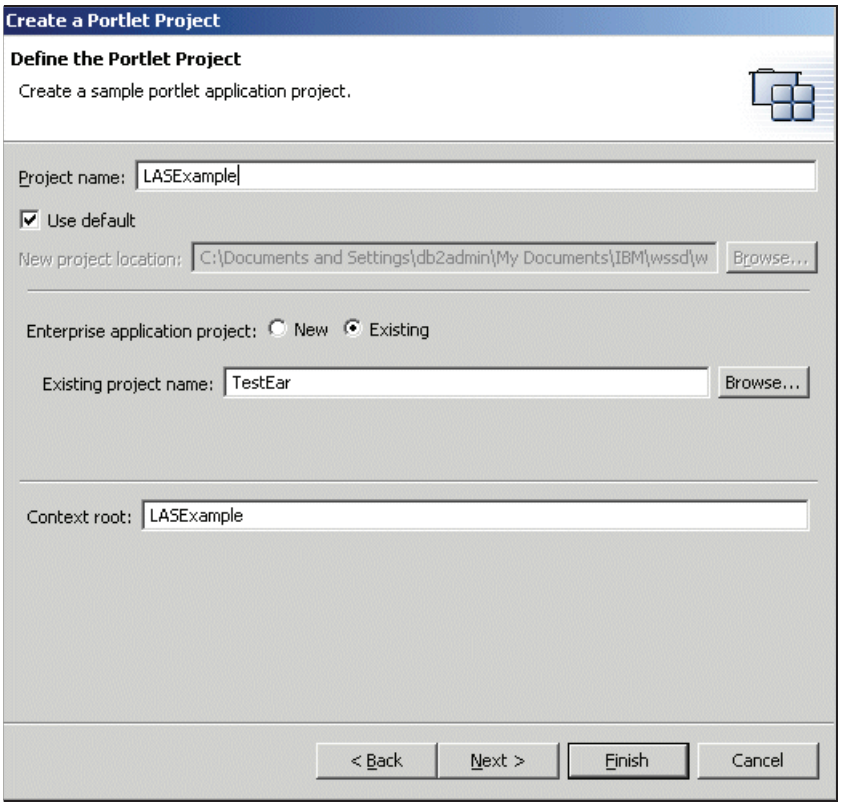


Figure 27-22 Create a portlet project

3. Verify the portlet parameters and click **Finish**.

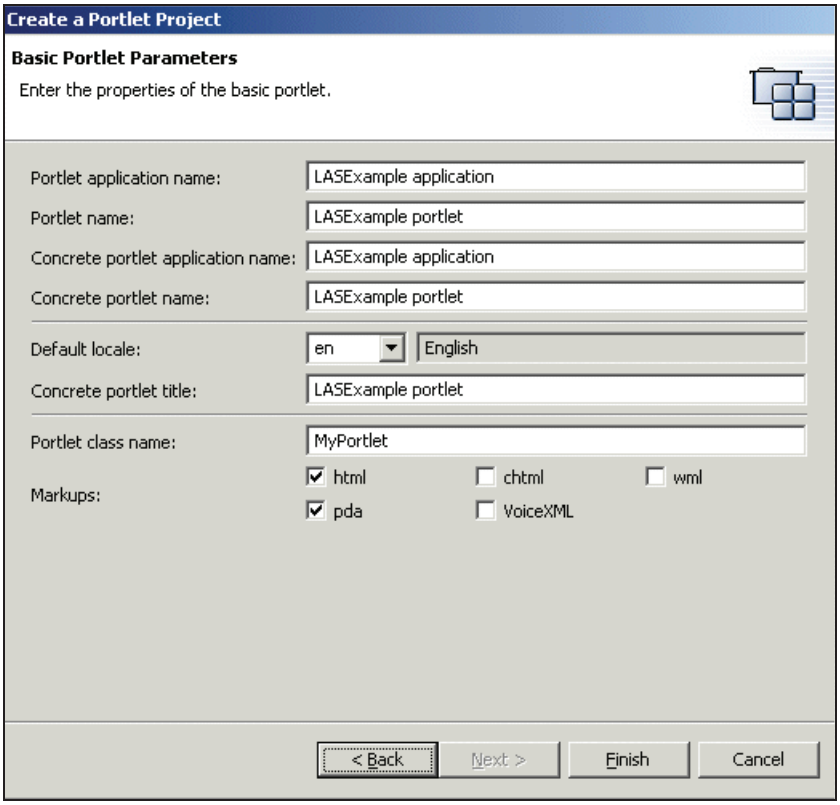


Figure 27-23 Basic portlet parameters

4. Right-click the project and select **Properties**. Select **Libraries** and click **Add External Jars.** Add atlas.jar into your project, which is in the path <WebSphere>/LAS/las/lib.

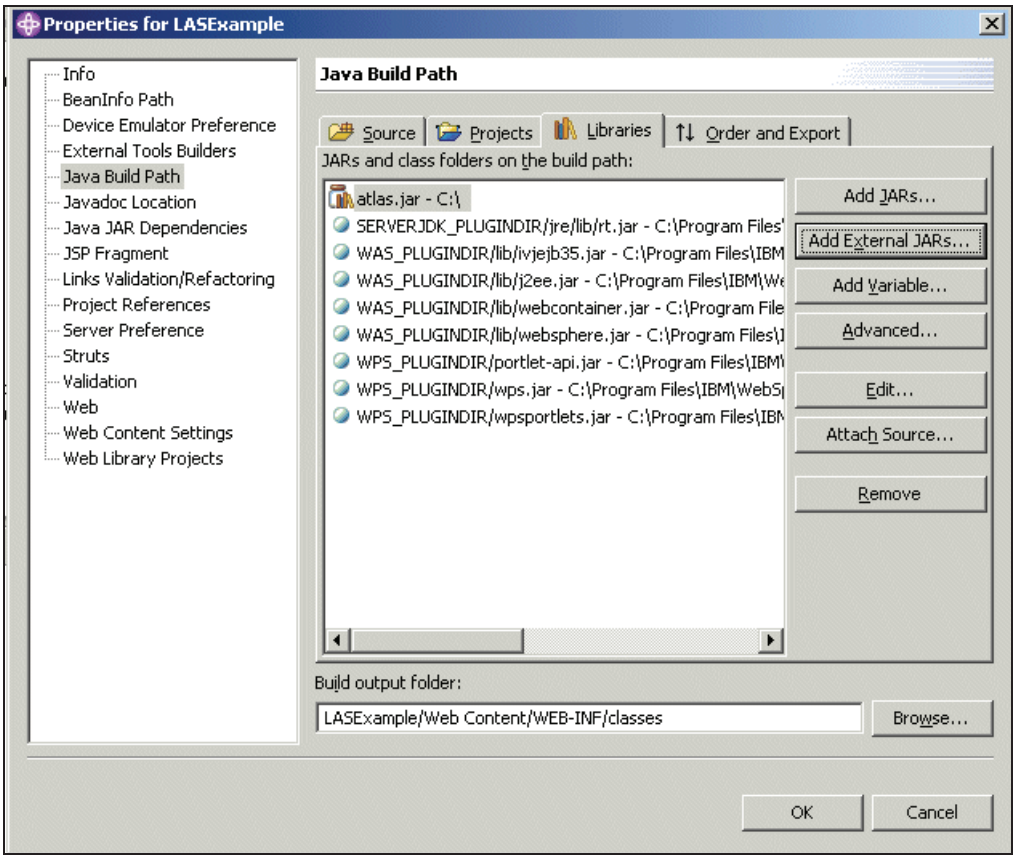


Figure 27-24 Adding atlas.jar to the project classpath

5. Add a GetMap.jsp to your project. This JSP file presents the user with the fields to enter the address.

Example 27-8 GetMap.jsp

```
<%@ page contentType="text/html" %>
<FORM action='<%=request.getAttribute("GetMap") %>' METHOD="POST"> Enter the
Location Details to get the Map<BR>
Street :<INPUT
   type="text" name="Street" size="26"><BR>
City : <INPUT type="text" name="City" size="20"> State :<INPUT
   type="text" name="State" size="20"><BR>PIN :
<INPUT type="text" name="Pin" size="20"> Country :<INPUT type="text"
   name="Country" size="20"><BR>
<BR>
```

```
<INPUT type="submit" name="GetMap"
value="GetMap"></FORM>
```

6. Add the ShowMap.jsp. This JSP file is used to display the map along with the ServletRender servlet.

Example 27-9 ShowMap.jsp

```
<%@ page contentType="text/html" %>
<%@taglib uri="/WEB-INF/tld/portlet.tld" prefix="portlet"%>
<portlet:init/>
<IMG source='<%=encodeURL("images/"+portletRequest.getSessionID()+".gif")) >'>
<FORM action='<%=request.getAttribute("OK") %>'><INPUT type="submit" name="OK"
value="OK"></FORM>
```

7. Add the Error.jsp. This JSP will be displayed if any exception occurs during the process of getting the map.

Example 27-10 Error.jsp

```
<%@ page contentType="text/html" %>
<P>The given address was not found.</P>
<FORM action='<%=request.getAttribute("OK") %>'><INPUT type="submit"
name="back" value="OK"></FORM>
```

8. Location Aware Services is implemented as a portlet service.

   The init method makes a call to get the Location Service Factory, which can then be used to create other services.

   The doView method displays different JSPs depending upon the state. The states get changed in the actionPerformed method.

   The actionPerformed method carries out the action depending upon the action event received.

   The GetMap event, is generated by the GetMap.jsp, which gets the input parameters and generates this event. The actionPerformed method receives this event, gets the address of the location, and loads up the Geocoding and Mapping services using the Location Service Factory. Then the address will be Geocoded and a map of the location will be requested using the MappingService. The map is then saved as a file under the images directory. This file will be associated with the session ID of the user and will be deleted when the user logs out.

Example 27-11 MyPortlet.java

```
package portlet;

import java.io.*;
```

```
import javax.servlet.RequestDispatcher;
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlet.event.ActionEvent;
import org.apache.jetspeed.portlet.event.ActionListener;
import com.ibm.atlas.system.*;
import com.ibm.atlas.portlet.service.*;
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlet.event.*;
import com.ibm.atlas.geocoding.*;
import com.ibm.atlas.geometry.*;
import com.ibm.atlas.mapping.*;

public class MyPortlet extends PortletAdapter implements ActionListener{

   private LocationFactoryService locFactorySvc=null;

   public void init(PortletConfig portletConfig) throws UnavailableException {
      super.init(portletConfig);
      try{
         locFactorySvc=(LocationFactoryService)
getPortletConfig().getContext().getService(com.ibm.atlas.portlet.service.Locati
onFactoryService.class);
      }catch (Exception e)
      {
         e.printStackTrace();
      }
      getPortletLog().debug("Location Portlet Init..successful");
   }

   public void doView(PortletRequest request, PortletResponse response) throws
PortletException, IOException {
      String includeFile="GetMap.jsp";
      String state= (String)
request.getPortletSession().getAttribute("state");

      if (state==null)
      {
         state="GetMap";
         request.getPortletSession().setAttribute("state",state);
      }

      if (state.equals("ShowMap"))
      {
         includeFile="ShowMap.jsp";
         try{
            RequestDispatcher
rd=this.getServletContext().getRequestDispatcher("/RenderMap");
            rd.include(request,response);
         }catch(Exception e)
```

```
         {
            e.printStackTrace();
         }
      }
      if (state.equals("Error")) includeFile="Error.jsp";
      getPortletLog().debug("State is "+state);

request.setAttribute("GetMap",generateURI(response,"GetMap").toString());
      request.setAttribute("OK",generateURI(response,"OK").toString());
       // Invoke the JSP to render
      getPortletConfig().getContext().include("/jsp/" + includeFile, request,
response);
   }

   public void actionPerformed(ActionEvent event)
   {
      String action=event.getActionString();
      PortletRequest request=event.getRequest();
      PortletSession session= request.getPortletSession();
      if (action.equals("GetMap"))
      {
         try
         {
            String temp;
            GeocodingService geoSvc = (GeocodingService)
locFactorySvc.getLocationService(GeocodingService.class);
            Address address=new Address();
            if ((temp=request.getParameter("Street"))!=null)
               address.setStreet(temp);
            if ((temp=request.getParameter("City"))!=null)
               address.setCity(temp);
            if ((temp=request.getParameter("State"))!=null)
               address.setState(temp);
            if ((temp=request.getParameter("Country"))!=null)
               address.setCountry(temp);
            if ((temp=request.getParameter("Pin"))!=null)
               address.setPostalCode(temp);
            getPortletLog().debug(address.toString());
            GeocodedAddress []geoAddress= geoSvc.geocode(address);
            MappingService mapSvc = (MappingService)
locFactorySvc.getLocationService(MappingService.class);

            Point pt = geoAddress[0].getPoint();
            // Get a map of the 5km-square area around the point
            Box bounds = new Box(pt.getCoordinates(), 5000.0, 5000.0);
            // The map image should be 240x240 pixels, in the GIF format
            MappingOptions options = new MappingOptions(240, 240, "gif");
            // Request that a generic "point" icon be placed at the point
            MapOverlay [] overlays = new MapOverlay[] {new MapOverlay(pt)};
```

```
            Map map = mapSvc.getMap(bounds, options, overlays);
            String path= getContext().getConfig().getRealPath("\images");
            FileOutputStream fos=new
FileOutputStream(path+"\"+session.getId()+".gif");
            fos.write(map.getData());
            fos.close();
            }catch(Exception e)
         {
            getPortletLog().debug(e.toString());
            session.setAttribute("state","Error");
            return;
         }
         session.setAttribute("state","ShowMap");
      }
      if (action.equals("OK"))
      {
         session.setAttribute("state","GetMap");
      }
   }
   public PortletURI generateURI(PortletResponse response, String action)
   {
      PortletURI tempURI = response.createURI();
      tempURI.addAction(action);
      return tempURI;
   }
   private String getJspExtension(PortletRequest request) {
      // Return "jsv" for Voice JSP, otherwise return "jsp"
      return request.getClient().getMarkupName().equals("vxml") ? "jsv" :
"jsp";
   }
   public void logout(PortletSession arg0) throws PortletException
   {
      super.logout(arg0);
      String path=getConfig().getContext().getRealPath("\images");
      File file=new File(path+"\"+arg0.getId()+".gif");
      if (file.exists())
         file.delete();
   }
}
```

27.7 Adapter development

Location Aware Services uses adapters that can be dynamically plugged in for enabling services from service providers. Service providers should write custom adapters that communicate with their infrastructure to provide the necessary

service. Location Aware Services provides adapters for several service providers. See 27.2.1, “Service adapters” on page 975 for a list.

In order to enable Location Aware Services to use your custom adapter to provide the service, you should develop the adapter in accordance with the interfaces defined and register the adapter for the specific service.

All service adapters for Location Aware Services must extend the com.ibm.atlas.adapters.LocationServiceAdapter class.

The LocationServiceAdapter class is an abstract class that contains four methods:

- public void init(LocationServiceConfig config)
- public LocationServiceConfig getConfig()
- public void setClientOptions(ServiceRequestOptions options)
- public abstract ServiceCapabilities getCapabilities()

The Location Service Factory calls the init method when the adapter is first created. The adapter can override this method to perform any required initialization procedures, including authentication if it is session-based, or testing communication with the remote provider's site. If the adapter overrides this method, it should call super.init first.

The setClientOptions method sets the clientName, clientPassword and userLanguagePreference properties in the ServiceRequestOptions object to values given in the adapter's init properties or, if not given there, to defaults specified in the atlas.properties file. All adapters must call this method when their main service methods are invoked.

The getCapabilities method returns a service-specific subclass of ServiceCapabilities. A DirectoryServiceAdapter must return a DirectoryCapabilities object, a GeocodingServiceAdapter must return a GeocodingCapabilities object, and so on. Since the ServiceCapabilities subclasses provide constructors that take a Properties object as a parameter, each adapter's responsibility is to retrieve the service's properties from the Service Registry and pass them to the constructor. Thus, a MappingService adapter's getCapabilities method would typically be implemented as follows.

```
ServiceCapabilities getCapabilities() {
ServiceRegistry svcReg = config.getFactory().getServiceRegistry();
Properties svcProps = svcReg.getServiceProperties(config.getName());
return new MappingCapabilities(svcProps);
}
```

For location service providers that offer two or more of the Location Aware Services, it may be desirable to create a superclass that is extended by all the

provider's service adapters. This superclass would extend LocationServiceAdapter, and would typically handle the low-level communication with the service provider that will be common to all the adapters.

Depending on the type of service, service adapters should implement one of the service interfaces. These service interfaces are defined in the com.ibm.adapters package. The following interfaces are included in this package:

- DevicePositionServiceAdapter
- DirectoryServiceAdapter
- GeocodingServiceAdapter
- MappingServiceAdapter
- ReverseGeocodingServiceAdapter
- RoutingServiceAdapter

27.7.1 Development

The following steps are necessary for developing a custom adapter. We will develop a dummy mapping adapter using WebSphere Studio Site Developer.

1. Open WebSphere Studio Site Developer and click **New Project**. Select a Java project and click **Next**.
2. Enter a project name and click **Next**.
3. In the Java Settings pane, click the **Libraries** tab and select **Add External JARS**. Add the atlas.jar file, which can be found in the <WebSphere>/LAS/las/lib folder. Click **Finish**.
4. Click your project and create a new Java Class. Set the super class as com.ibm.atlas.adapters.LocationServiceAdapter. Set the interface as com.ibm.atlas.adapters.MappingServiceAdapter. Click **Finish**.

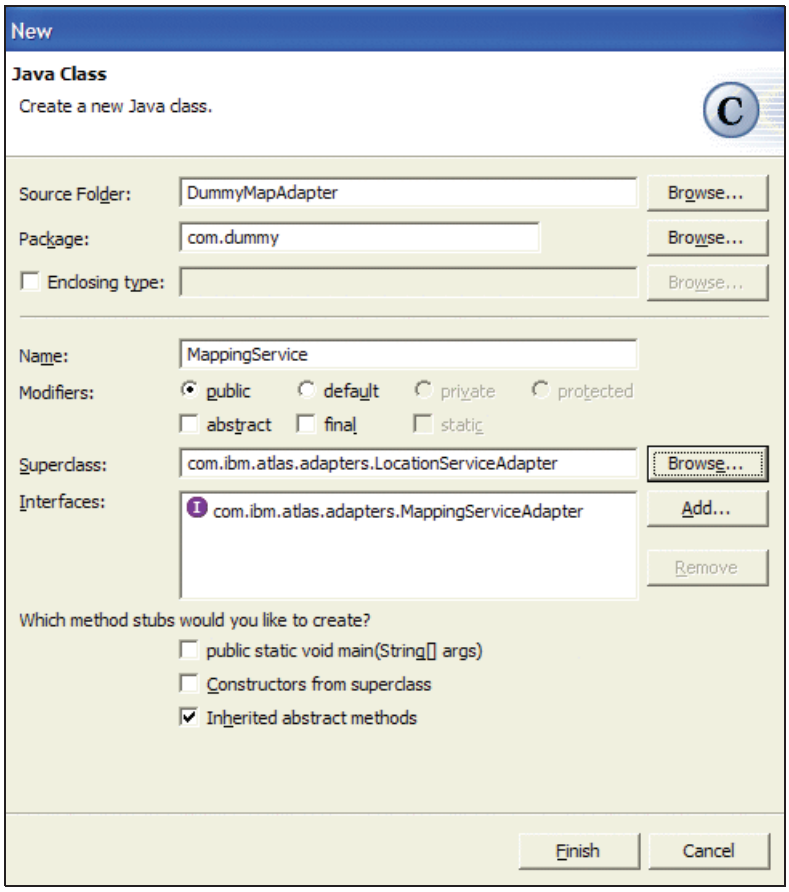


Figure 27-25 Dummy adapter class

5. This should get a skeleton code shown in Example 27-12. Modify the GetMap methods to return the map. The MapOverlay object contains the necessary information about the location and the details required for the map.

Example 27-12 Dummy map adapter

```
package com.dummy
import com.ibm.atlas.ServiceCapabilities;
import com.ibm.atlas.ServiceFailureException;
import com.ibm.atlas.ServiceRequestException;
import com.ibm.atlas.adapters.LocationServiceAdapter;
import com.ibm.atlas.adapters.MappingServiceAdapter;
import com.ibm.atlas.geometry.Box;
import com.ibm.atlas.geometry.PointCircle;
import com.ibm.atlas.mapping.Map;
```

```
import com.ibm.atlas.mapping.MapOverlay;
import com.ibm.atlas.mapping.MappingCapabilities;
import com.ibm.atlas.mapping.MappingOptions;
import com.ibm.atlas.system.ServiceRegistry;

public class MappingService
   extends LocationServiceAdapter
   implements MappingServiceAdapter {


public ServiceCapabilities getCapabilities() {
       ServiceRegistry serviceRegistry =
config.getFactory().getServiceRegistry();
       java.util.Properties serviceProperties =
serviceRegistry.getServiceProperties(config.getName());
       return new MappingCapabilities(serviceProperties);

public Map getMap(Box arg0, MappingOptions arg1, MapOverlay[] arg2)
       throws ServiceFailureException, ServiceRequestException {
          Map map=new Map();

          // Use the Mapping Options, Map Overlay to get the Map Data
          return map;
   }


public Map getMap(
       PointCircle arg0,
       int arg1,
       MappingOptions arg2,
       MapOverlay[] arg3)
       throws ServiceFailureException, ServiceRequestException {
       Map map=new Map();
          byte [] buffer=null;

          // Use the PointCircle, Mapping Options, Map Overlay to get the Map
Data
          return map;
   }
}
```

27.7.2 Registering the custom adapter

After the adapter class is ready, it has to be registered with the Location Aware Services. When registering an adapter, you must specify such information as the class to be loaded, service provider properties, the service type and so on.

Location Aware Services provides an Administration portlet through which the custom adapters can be added and configured. For details about registering and configuring your custom adapter, see 27.4.1, “Adding a custom service provider” on page 979.

During the registration of the service adapter, a class has to specified that will be loaded by the Location Aware Services if the service adapter was matched and selected for service. This class has to put in the directory which is in the class path of the WebSphere Application Server. If the class was com.dummy.GeoCoding, then this class has to be present in <WAS_HOME>/lib/com/dummy directory.

27.8 Troubleshooting

The Server Setting portlet can be used to set the Message log and Trace log levels.

Messages may be written either to a separate file or console. If console is selected, messages are written to a Portal message log file that can be found in <wps home>/log/appserver-out.log. If writing to a file is selected, the messages will be written to a specified file found in <wps home>/log.

There are three message level settings:

- **Errors**: Logs errors that are critical but Location Aware Services is still operational.
- **Warning and Error**: Includes error messages along with minor problems.
- **Information, Warning and Error**: Includes basic information messages along with warnings and errors.

Tracing is a troubleshooting tool but decreases system performance, so it should be enabled only when necessary. Tracing can either be disabled or levels set to low, medium, or high.

Trace information can either be logged to a file or console. When console is selected, the trace information is logged into the file <wps_home>/log/appserver-err.log. If writing to a file is selected, the trace information is logged into a specified file found in the path <wps_home>/log.

Select Location Aware Services and select the **Server Settings** tab. This should bring up the window in Figure 27-26 on page 1014, where trace and message settings can be configured.

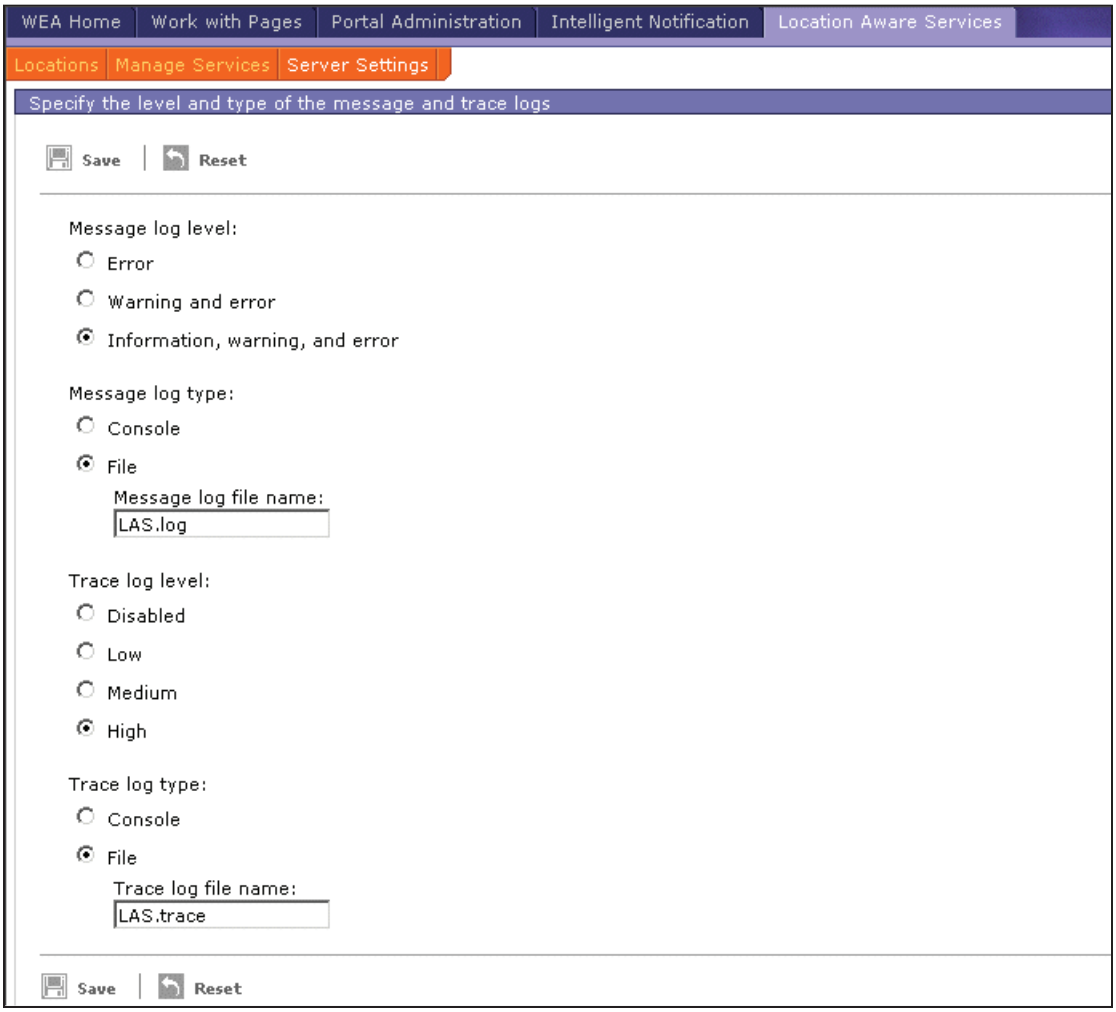


Figure 27-26 Server Settings portlet

28

Device Services

This chapter describes the Device Services offered in WebSphere Everyplace Access V4.3.

In this chapter we explain how Device Services are integrated in WebSphere Everyplace Access V4.3 and how to use Device Management portlets. We help you to understand Device Services through scenarios that are common in device management, such as bootstrapping new devices with a WebSphere Everyplace Access V4.3 Client, configuring devices, taking inventory, and software distribution.

© Copyright IBM Corp. 2003. All rights reserved.

28.1 Device Services overview

Device management is critical because of the increase in the number of mobile devices on the market and the growing complexity of the software in these devices.

Device Services offered in WebSphere Everyplace Access V4.3 facilitates management of software on mobile devices. WebSphere Everyplace Access integrates with the Tivoli Device Manager to provide device management services.

Note: Currently Pocket PC and Palm-based devices are supported. Future versions of WebSphere Everyplace Access will support SyncML-based devices.

28.1.1 Architecture

Figure 28-1 shows the different components of the device management services in WebSphere Everyplace Access. In this section each component is briefly explained.

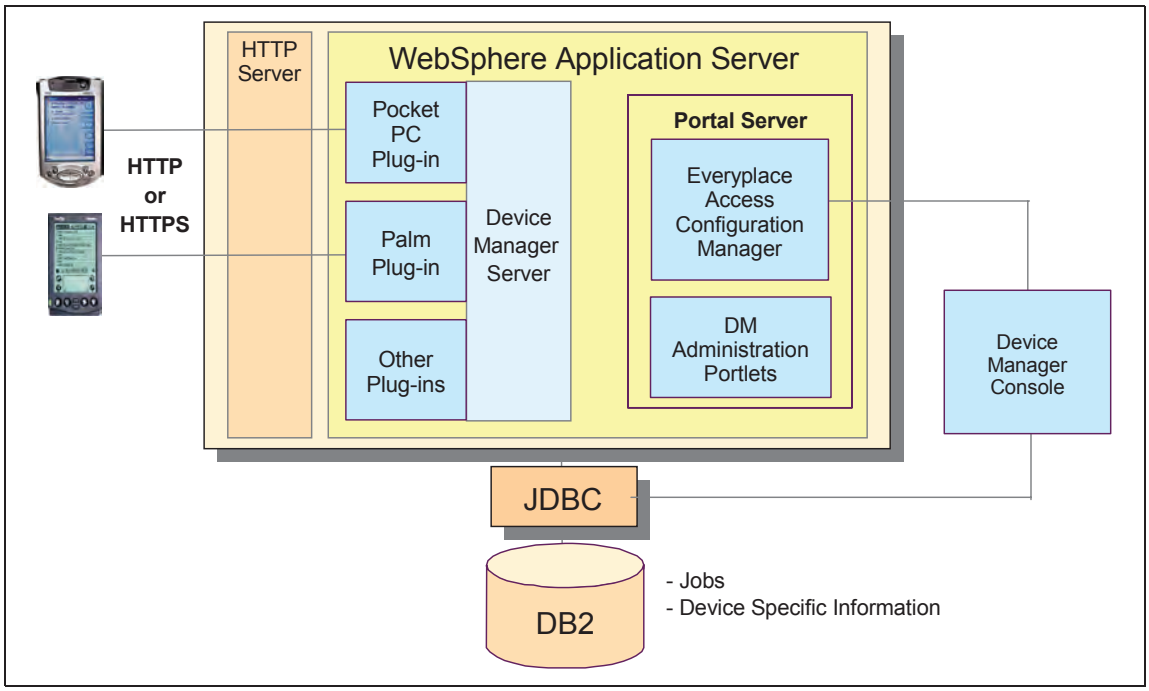


Figure 28-1 WebSphere Everyplace Access Device Services

28.1.2 Device Services

The Device Services is a set of servlets executing in the application server. Device Services execute the management tasks (jobs) on the connected devices.

Device Management Server includes a DM servlet that ensures that the device is enrolled with the server. For enrolled devices, the DM servlet checks if there are any pending jobs for the device and processes them after prioritizing the jobs. The DM servlet processes the jobs using the device plug-ins that interact with the device.

Device plug-ins are responsible for device identification, communication with the device, processing jobs on the device, and high-level management of tasks. A plug-in is provided for different classes of devices and is extensible to support new devices types. The following plug-ins are provided in the Device Services:

- Pocket PC plug-in
- Palm plug-in
- SyncML plug-in (Technology preview only)

28.1.3 Device Services Agent

The Device Services Agent resides on the device. The Device Agent is responsible for executing the commands sent by the Device Management Server. The Client connects and interacts with the Device Management Server using a proprietary protocol. The Device Services Agent is installed along with the WebSphere Everyplace Access Client. The Device Agent can be configured and controlled by the Everyplace Client UI. The Device Agent need not be connected to the server all the time. When the Client Agent connects, the server identifies the client and executes the jobs pending for this device.

28.1.4 Device Services database

The DeviceServices database is a relational database that stores all the device management information. This database stores device-specific information, including the jobs for the devices. The Device Services database is created with all the necessary tables, views and queries.

28.1.5 Device Management portlets

Device Management portlets are installed with the WebSphere Everyplace Access core services. Device Management portlets provide an extensible framework for adding new job portlets. There are two types of portlets: administration and user.

Administration portlets

Administration portlets allow administrators to create and manage jobs. WebSphere Everyplace Access provides the following portlets:

- The Device Management - Job portlet allows administrators to create jobs.
- The Everyplace Client Installer portlet enables administrators to create an install job for installing WebSphere Everyplace Access Client.
- The Device Management - Configure portlet enables administrators to create configuration files that store connection profiles. Connection profiles define the host names of the various servers in a WebSphere Everyplace Access environment.

Note: It is expected that future releases of WebSphere Everyplace Access will include more portlets for administration.

User portlets

With user portlets, users with access to an Everyplace Access Portal server can install and configure WebSphere Everyplace Access Clients on their devices.

28.1.6 Device Services console

The Device Services console is a Java utility that the administrator can use to administer and monitor the Device Management Server. The console provides functionality to register and view devices, create jobs for viewing or modifying device or software properties, view the status of the jobs, and so on.

The Device Services console can be downloaded from the Device Management Server and installed on any system. This allows administrators of the devices to control devices from various terminals and not necessarily on the server. The Device Services console requires a DB2 database client for connecting to the Device Management database.

The first time the Device Services console is started, it must be able to connect to the Device Management Server to login and retrieve the properties file. Device Services console creates a backup of the properties file and subsequent start up will not need to connect to the Device Management Server for login and will connect directly to the Device Services database for administration.

Note: The Device Services console connects to the WebSphere Everyplace Access Subscription Manager to authenticate the user.

With the Device Services console, the following types of jobs can be created:

- Software Distribution job. With this job we can distribute software packages to clients.
- Inventory Collection job. With this job we can query the device and check all the installed software on the device.
- Configuration job. With this job we can configure the browser settings, PPP connection settings, TCP/IP settings, mail server settings, and Device Management Client settings on the device.

28.2 Setting the stage

Before we begin with the different scenarios of device management, certain configuration has to be done.

28.2.1 Creating a device management user group

For details about adding and viewing groups and users, see Chapter 2, “Administration” on page 25. For example:

1. Check if the user group dmadmins exists. This should have been automatically been created during the installation of WebSphere Everyplace Access.
2. If not, create a user group with the name dmadmins. The name should be exactly the same. Otherwise, Device Management Services will not work.
3. Add the Portal administrator (usually wpsadmin) as a member of the group.

Note: When using Device Management Services, you must log in as the administrator who is a member of the dmadmins group.

Permissions must be given to this user group to view certain places, pages, and portlets. For information about changing portlet permissions, see Chapter 2, “Administration” on page 25.

1. View permission should be given to WebSphere Everyplace Access Home place.
2. View permission should be given to Administration and Configure pages under WebSphere Everyplace Access Home.
3. Manage permission should be given to the following portlets:
   - Device Management - Configure

– Device Management - Jobs

28.2.2 Starting the Device Services console

A Device Services console can be downloaded from the Device Management Server and launched on any system. To download the Device Services console, enter the following path:

```
http://dmserver.yourco.com/dmconsole/DMConsole
```

On the server machine, the Device Services console can be launched from the WebSphere installed path. Go to WebSphere\DMS\console and click **DMconsole.bat**.

When the Device Services console is launched, a command window shown in Figure 28-2 will be displayed.

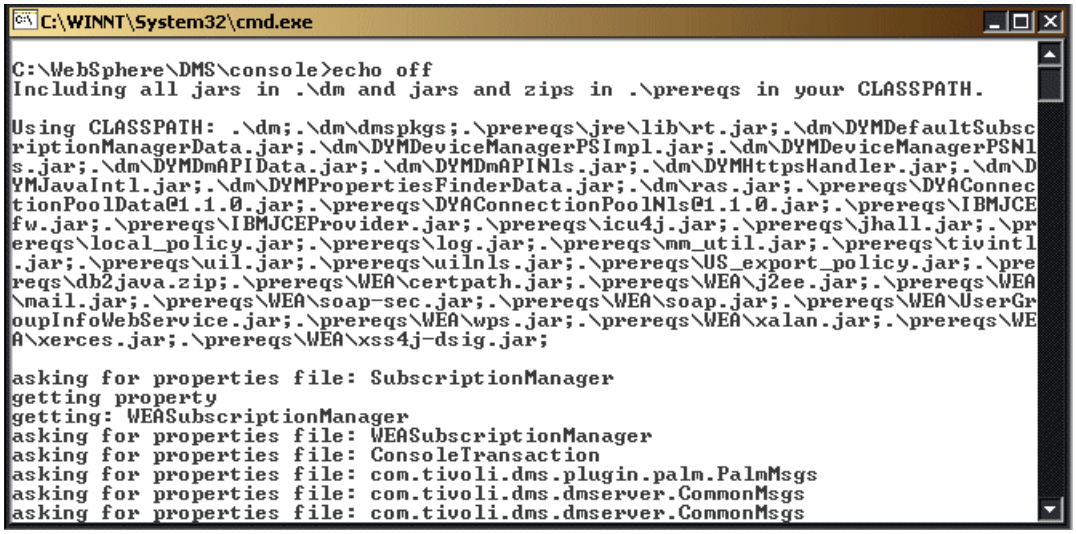


Figure 28-2 Device Services console command pop-up window

Enter the administration user ID and password in the login prompt. The user must be a part of the dmadmins user group that was created in the previous section.

If the login was successful, the window in Figure 28-3 on page 1021 will be displayed.

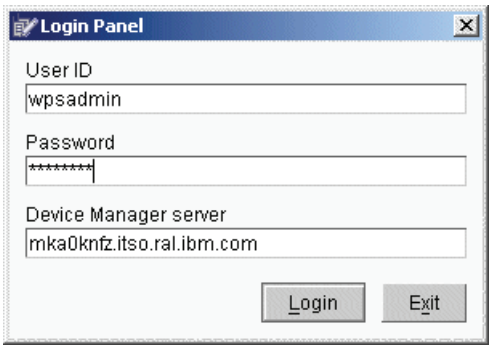


Figure 28-3 Login prompt

If the console is connected to the Device Management Server and if the user was authenticated and part of dmadmins group, the window shown in Figure 28-4 will be displayed.



Figure 28-4 Device Services console

28.3 Scenarios

In this section, we provide sample scenarios using Device Services.

28.3.1 Bootstrapping and configuring Everyplace Access Client

WebSphere Everyplace Access Device Services can be used to install and configure a WebSphere Everyplace Access Client on the device. This scenario will show how to use the Device Management portlets provided in WebSphere Everyplace Access.

Note: This scenario is applicable only to Pocket PC-based devices and only to devices that do not have Everyplace Client installed.

Device management administration

1. Log into the Portal server using the user ID and password of the administrator who is a part of the dmadmins group.
2. Select the **WEA Home** tab.
3. Select the **Administration** tab.
4. Configuration files include device connection profiles that help users connect to server-side applications. A configuration file can hold multiple connection profiles.

Creation of configuration file and device connection profile

1. Enter a unique name for the new configuration file in the edit box below New File in the Device Management - Configure portlet and click the **New** button.

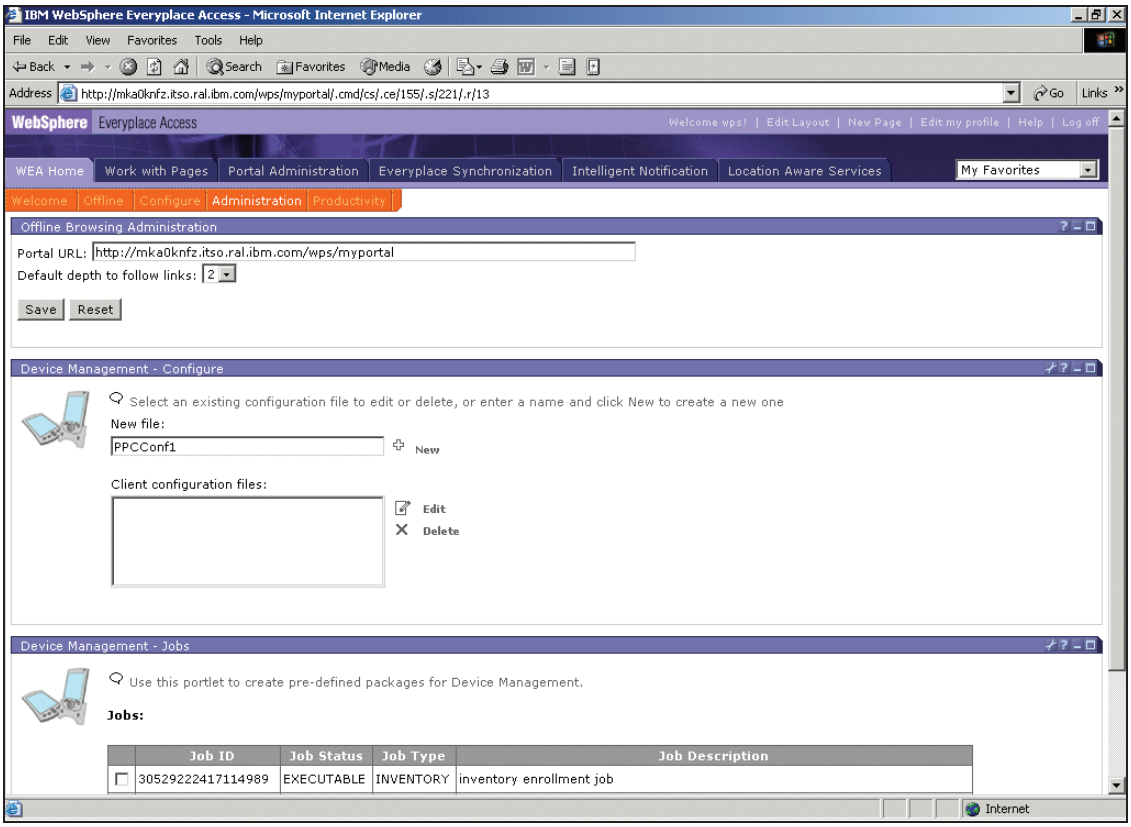


Figure 28-5 New configuration file

2. Select an existing device connection profile or create a new one by clicking **New**.

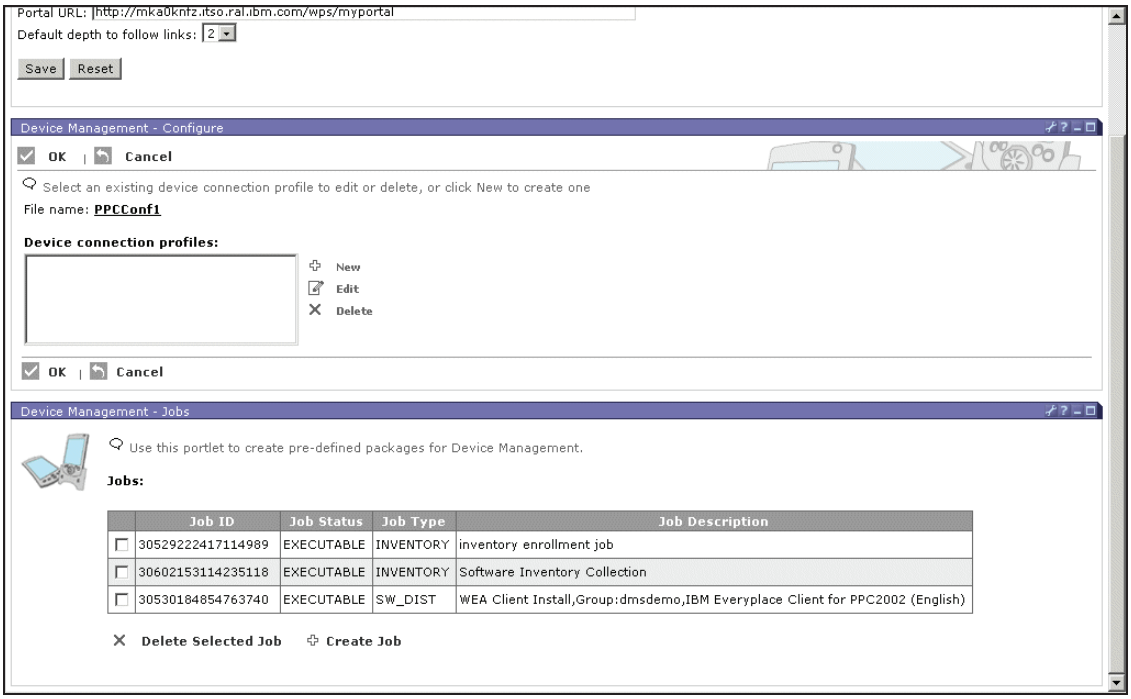


Figure 28-6 New device connection profile

3. Enter a unique profile name and modify or verify the server information in the profile and click **OK**.

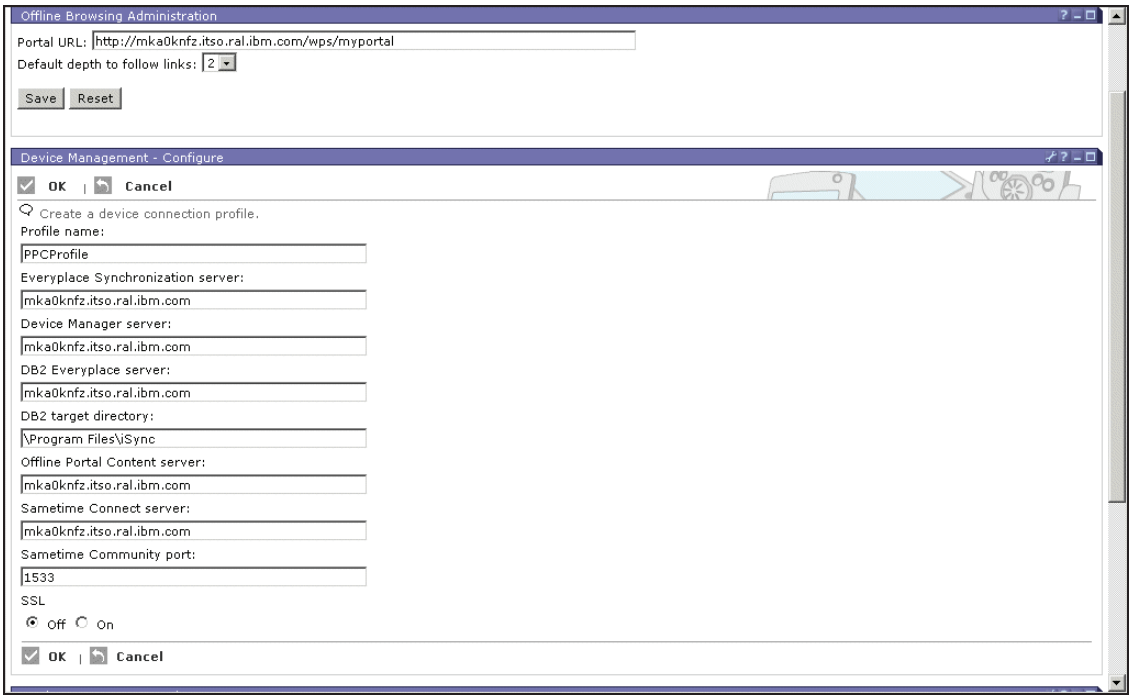


Figure 28-7 Configuring a device connection profile

4. In the Device Management - Configure portlet, select the default profile for the configuration and click **OK**.

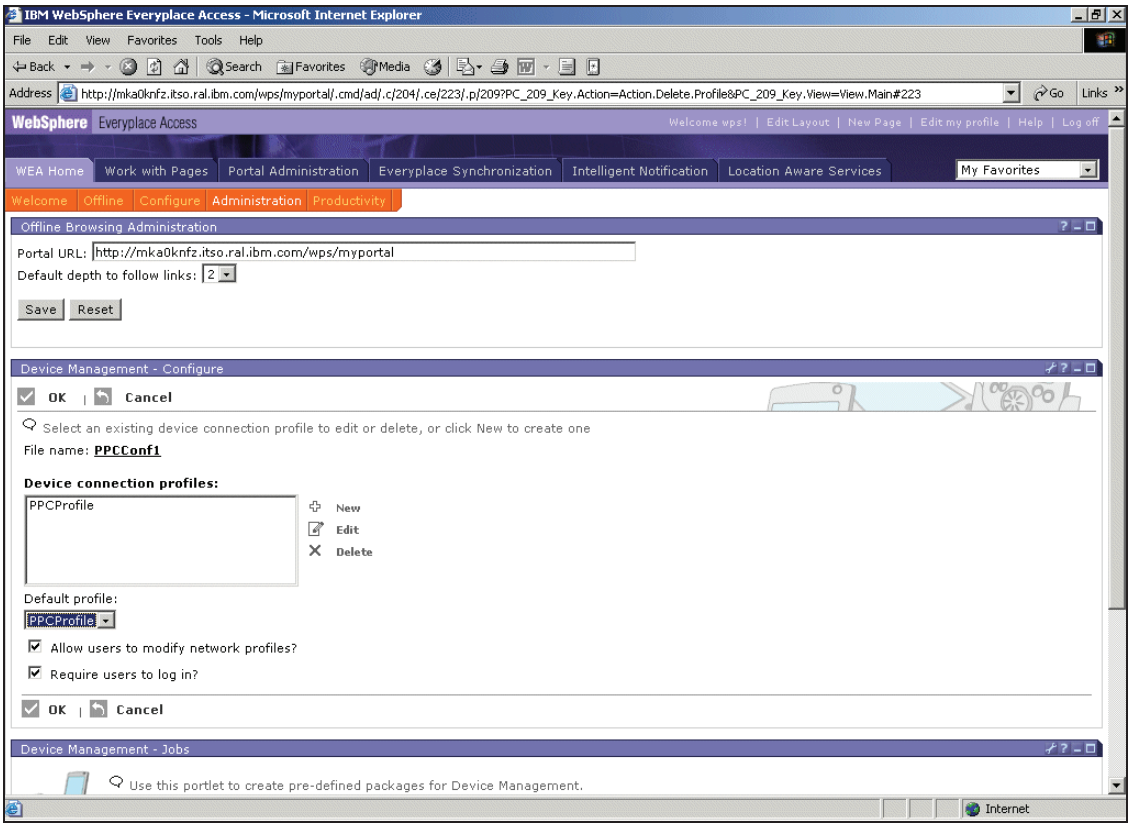


Figure 28-8 Selecting a default connection profile

Uploading the Pocket PC client installation files

1. On the Device Management - Jobs portlet, click the **Configure** button, which should bring up the window shown in Figure 28-8.

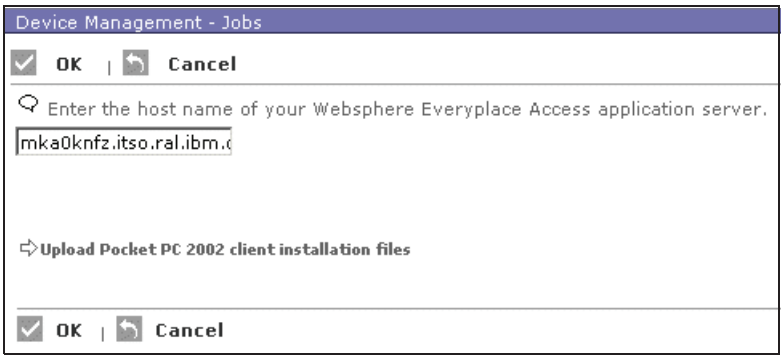


Figure 28-9 Upload Pocket PC 2002 client installation files

2. Click **Upload Pocket PC 2002 Client installation files.** Disk 7 of the WebSphere Everyplace Access Installation CD provides the necessary Pocket PC 2002 files for installing the WebSphere Everyplace Access Client. Enter the names of the necessary files and click **Upload**.

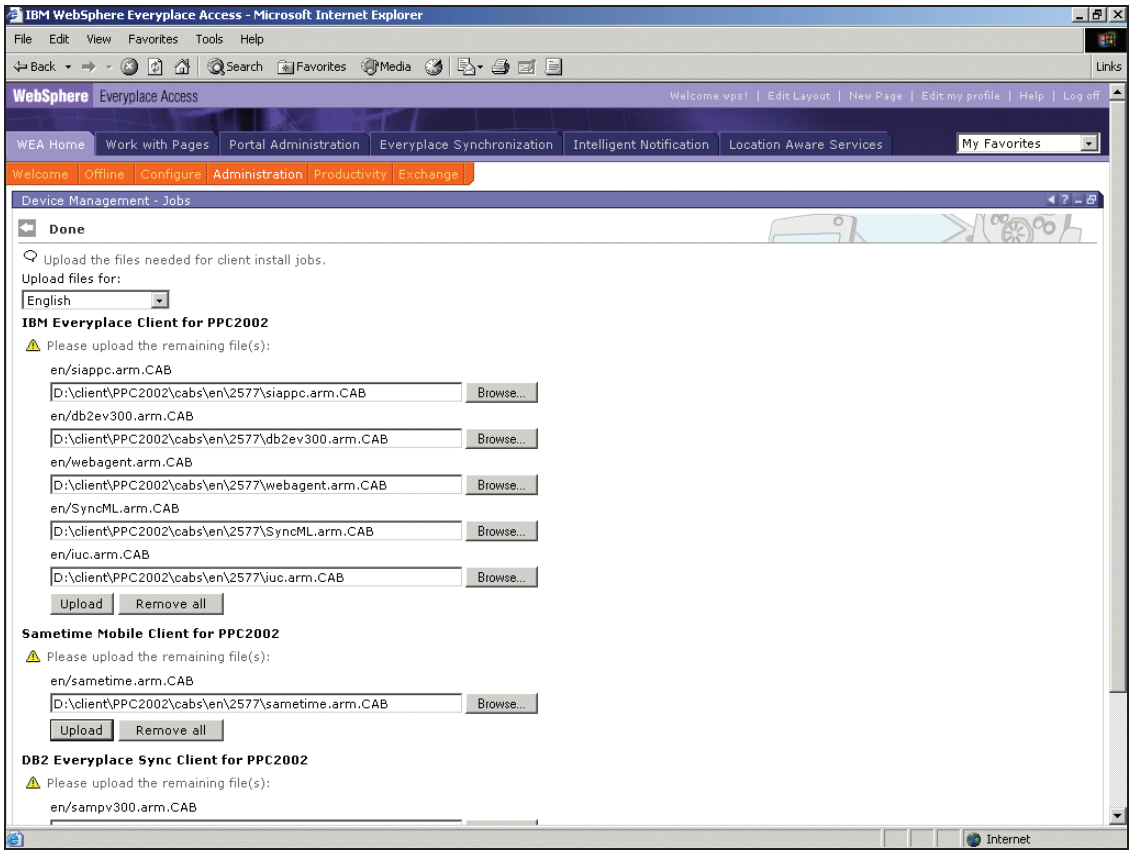


Figure 28-10 Uploading the Pocket PC 2002 client installation files

3. After you click **OK**, you should get the window shown in Figure 28-11 on page 1029.

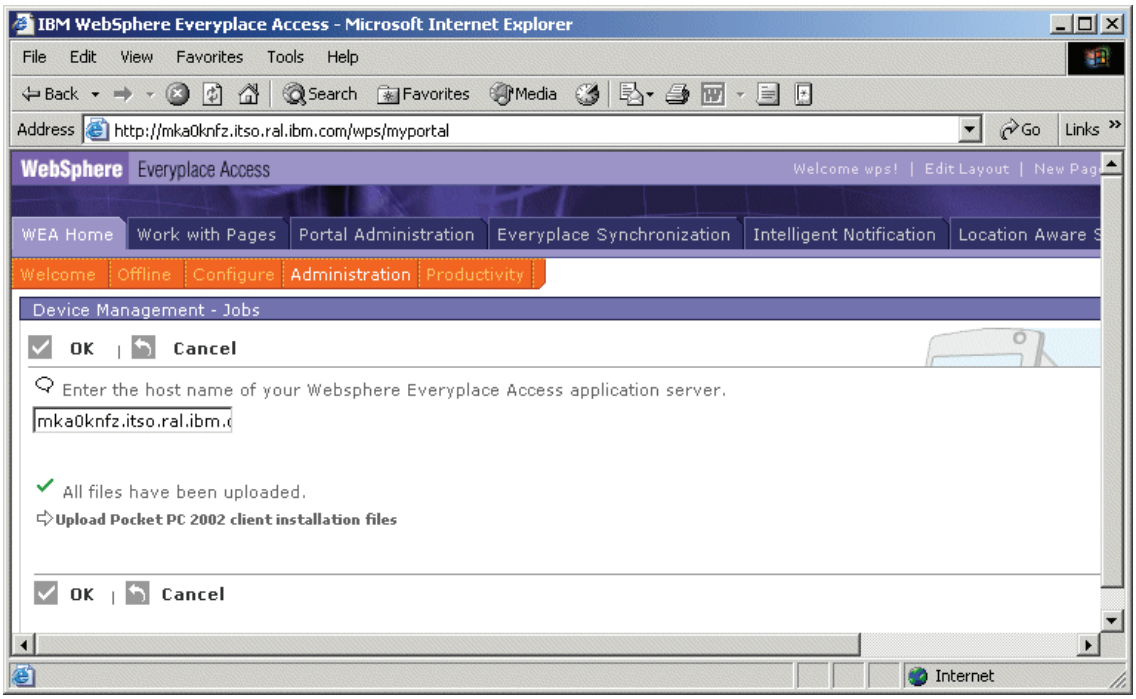


Figure 28-11 All files have been uploaded

Creation of an Everyplace Client installer job

1. In the Device Management - Jobs portlet, click the **Create Job** button. Select **WEA Client Install** as the job type.

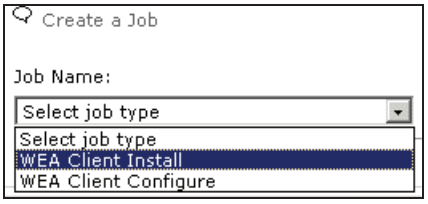


Figure 28-12 Install job type

2. Click **OK**, which should bring up the screen shown in Figure 28-13 on page 1030. Select the job for the appropriate user group and click **OK**.

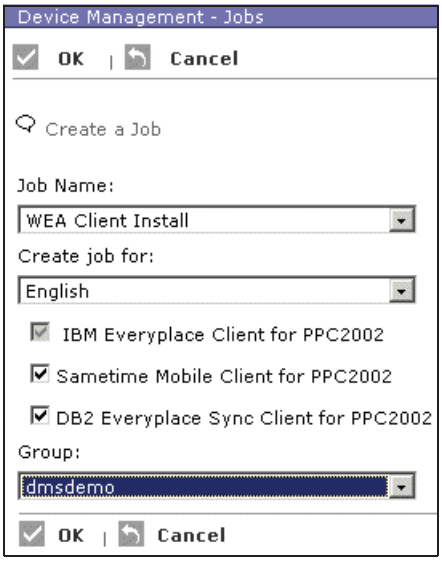


Figure 28-13 Install job configuration

Creating a configuration job

1. In the Device Management - Jobs portlet, click the **Create Job** button. Select **WEA Client Configure** as the job type.
2. Select the configuration file and the group of users for which this job is applicable and click **OK**.

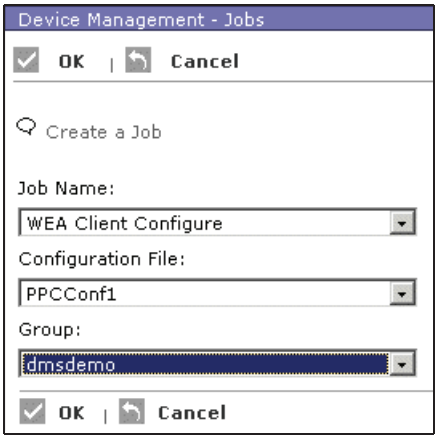


Figure 28-14 WebSphere Everyplace Access Client configure job

3. If the jobs appear in the list in the Device Management - Jobs portlet, the server- side configuration is over.

Steps to be performed on the Pocket PC

1. Open the browser on the Pocket PC device and enter the Portal server URL, for example:

   ```
   http://yourserver.com/wps/portal
   ```

2. Enter the user name and password. The user should be a part of the user group for whom the install and configure job has been scheduled. Click **Login**.

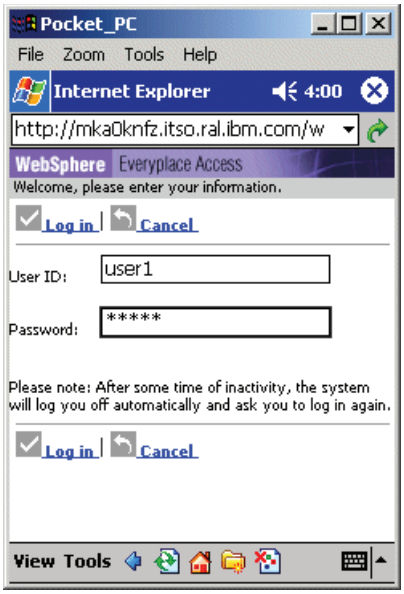


Figure 28-15 WebSphere Everyplace Access Login

3. You will see the WebSphere Everyplace Access Home page. Select **Configure** in the drop-down list. Click **Everyplace Client Installer**.

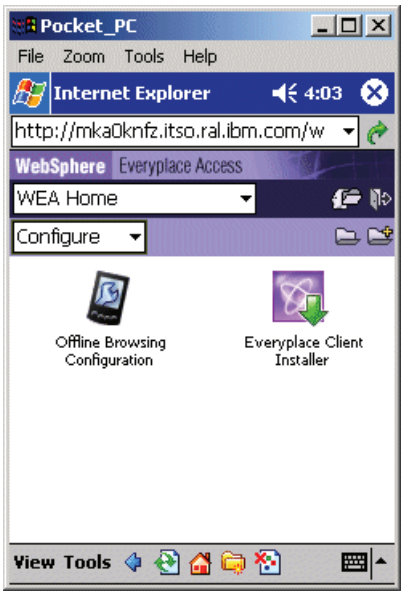


Figure 28-16 WebSphere Everyplace Access Client Installer

4. Click the **Download Installer** button.

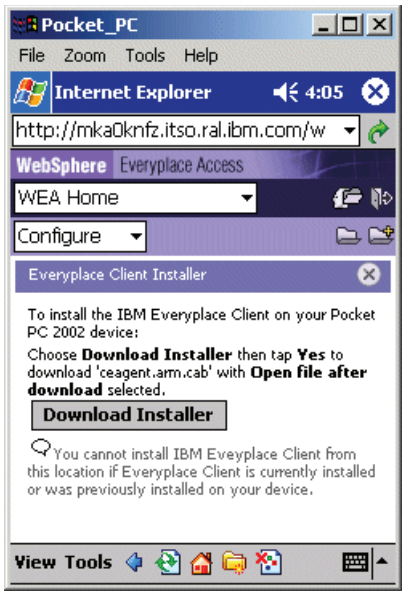


Figure 28-17 Download Installer

5. Click **Yes**.



Figure 28-18 Download WebSphere Everyplace Access Setup confirmation

6. The installation will take some time. The installation will download the necessary installation files and launch it automatically. The display will show the status of each job being performed. The screen shown in Figure 28-19 on page 1034 should appear on the Pocket PC.



Figure 28-19 Current status

7. The first time the device connects to the Device Services, the Device Services servlet routes the call to the enrollment server, which enrolls the device with the user. The user is the user ID with which you logged into the Portal server. The first job that is performed is an inventory job. Subsequently the Device Management Server will check if there were any jobs scheduled for this device class and user group and continue processing them. The Everyplace Client files are downloaded and launched automatically. Once the installation and configuration of the WebSphere Everyplace Access Client is completed successfully, you will see the screen shown in Figure 28-20 on page 1035.

.



Figure 28-20 WebSphere Everyplace Access Client login screen

8. Enter the user ID and password and log in. This will bring up the screen shown in Figure 28-21.

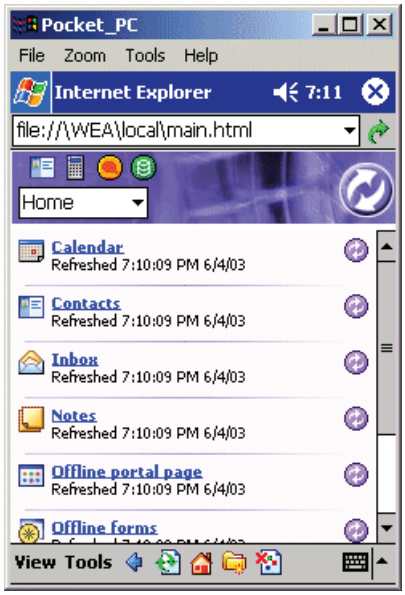


Figure 28-21 WebSphere Everyplace Access Client home page

9. Select **My Settings**. Scroll down to network profiles and check the network configuration profile. It should show the profile you had configured for this device with your server addresses.

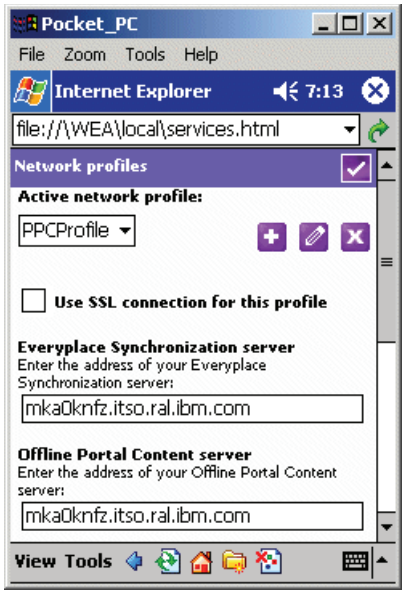


Figure 28-22 Network profile

10. The WebSphere Everyplace Access Client is installed and configured. You are all set for the next scenario.

28.3.2 Software distribution

Distributing software to devices is the most common application of device management. In this scenario, we will show you how to use the Device Services console and the necessary steps that you will have to perform in order to distribute software to devices.

Distributing software requires you to:

1. Create a software package
2. Register the software with the Device Management Server
3. Create a job for distributing it.

As an example we show you the suggested steps for software distribution when distributing a DB2e application on a Pocket PC device. For details about DB2e application development, see Chapter 18, “DB2 Everyplace applications with WebSphere Studio Device Developer” on page 529.

Creating the software package

A software package consists of the application-related files along with two other files: the meta package definition file and the package definition file.

The meta package definition file is the definition file for the entire software package. It lists all the application packages which makes a software package.

For each application package in the software package, the package definition file lists the files that make up the application package, such as graphics files, database files, compiled files, and distribution properties for that application package, such as running a setup file after the files are downloaded and allocating disk space.

For detailed instructions on what constitutes a meta package definition file and package definition file, look into the appropriate plug-in section in the Device Services Library.

Applications to be distributed to Pocket PC devices are packaged as CAB files. Palm device applications *.PRC and databases *.PDB can be distributed.

Creating a software package

1. The software package has to be accessible by the HTTP Server. Create a folder under the document root of your HTTP Server Program Files/IBM HTTP Server/htdocs. We created a folder called DMSSoftware.
2. Copy your application CAB files to the DMSSoftware directory. Our application has only one CAB file, called WinCEMobileOrderSystem.CAB.
3. Create the package definition files for your applications.

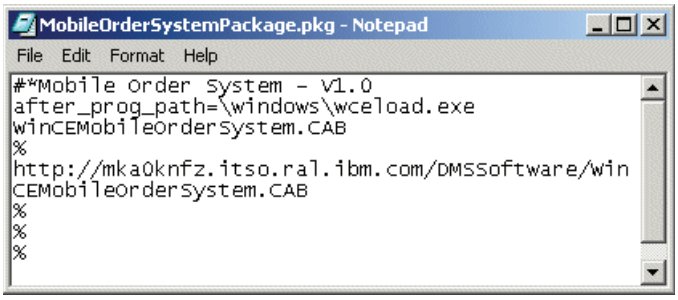


Figure 28-23 Package definition file

4. Create the meta package definition file. There should be only one meta package definition file for the entire software package.

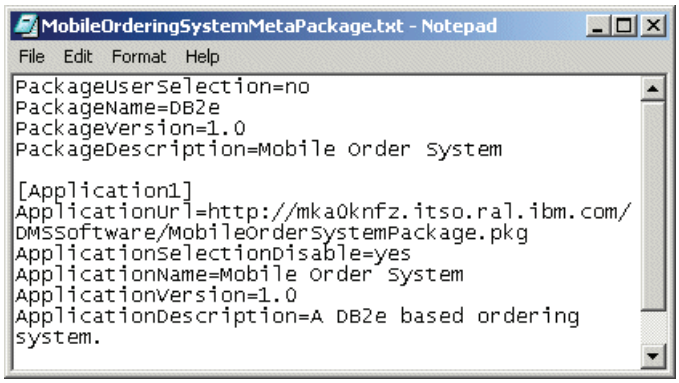


Figure 28-24 Meta package definition file

5. The package definition files and meta package definition file should be copied to the DMSSoftware directory.

Registering the software with the Device Management Server

1. Open the Device Services console. The software is created for a specific device class. Click **Device Classes.** Right-click the appropriate device class where you are registering the software and select **New Software**.

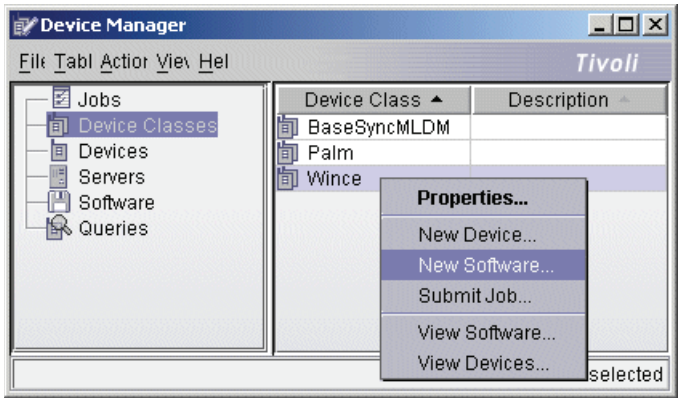


Figure 28-25 Registering new software

2. Enter the new software properties.

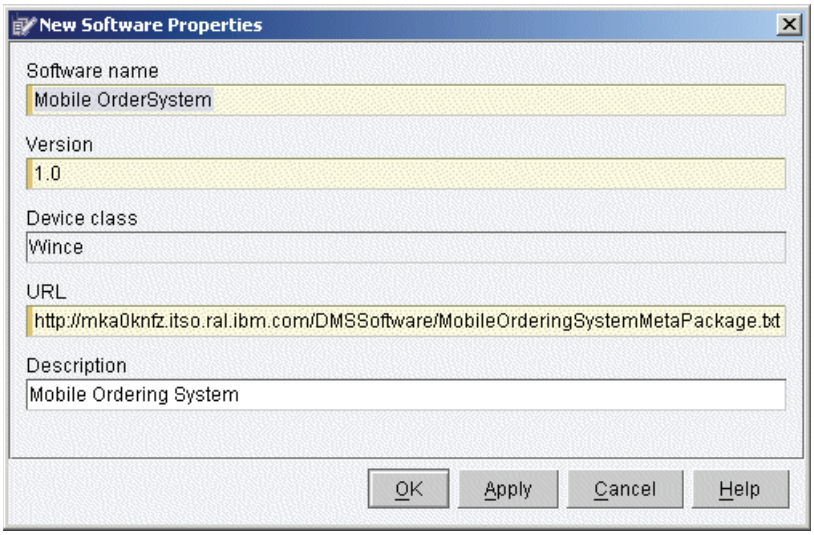


Figure 28-26 New software properties

Note: The URL should be the fully qualified URL for the meta package definition file.

3. Click **OK** and check if the software is registered by clicking the **Software** tab and select **Return Anything**.

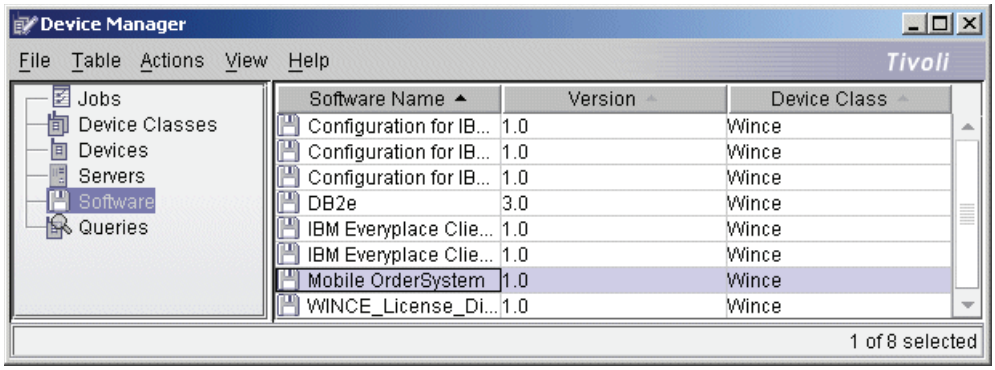


Figure 28-27 Registered software

Creating a software distribution job

With the Device Services console, you can create jobs for specific devices, user groups, device classes, enrolled devices, and so on. You can create the following types of jobs:

- Inventory collection job
- Device configuration job
- Software Distribution job

We will be creating a software distribution job for the user group dmsdemo and for Windows CE devices.

To create a software distribution job, do the following:

1. In the Device Services console, right-click **Jobs** and select **Submit Job**.
2. Select **Device Filtered by query**, or by owner group or both. Select Windows CE as the device class, Owner Group and select currently enrolled devices only. Click **Next**.

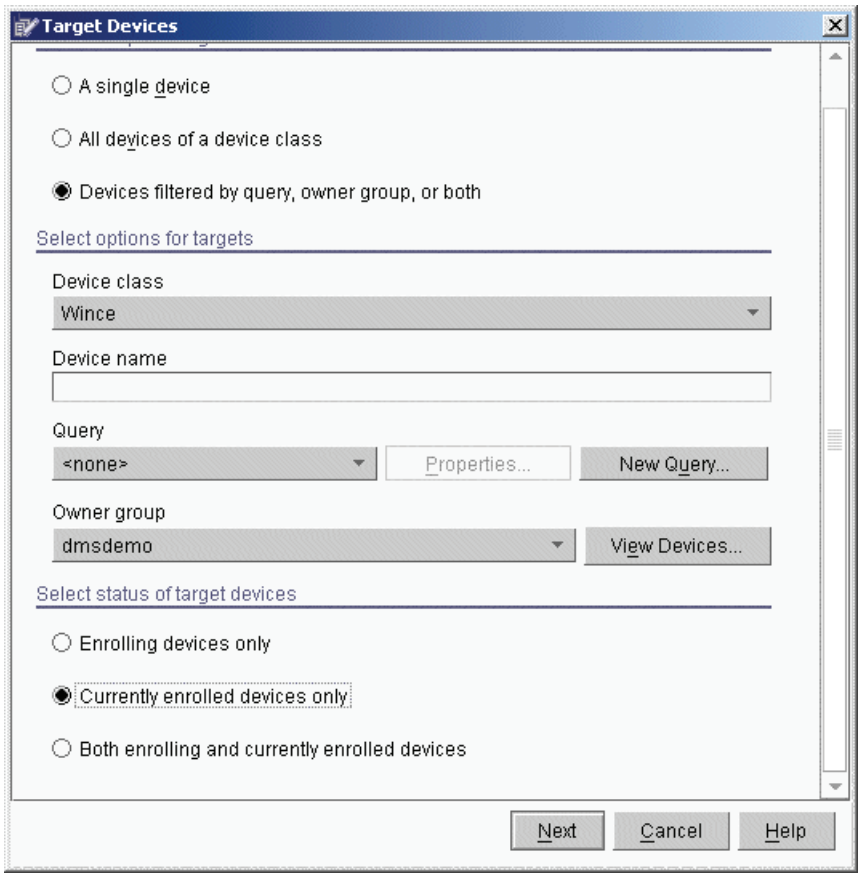


Figure 28-28 Target devices

3. Select **Software Distribution** as the job type. Enter a description and click **Next**.

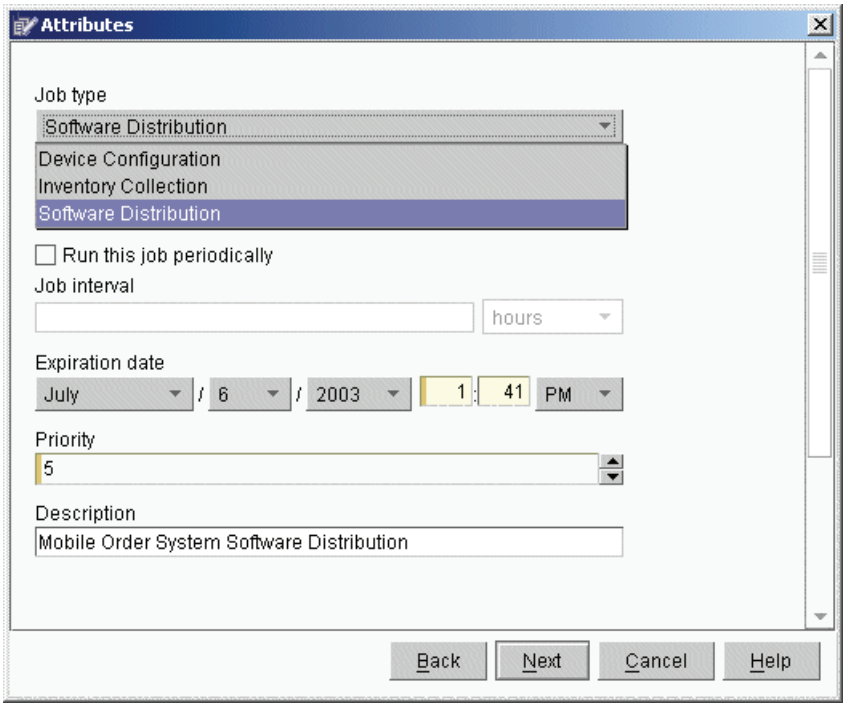


Figure 28-29 Attributes

4. Select the software that should be distributed and click **Next**.

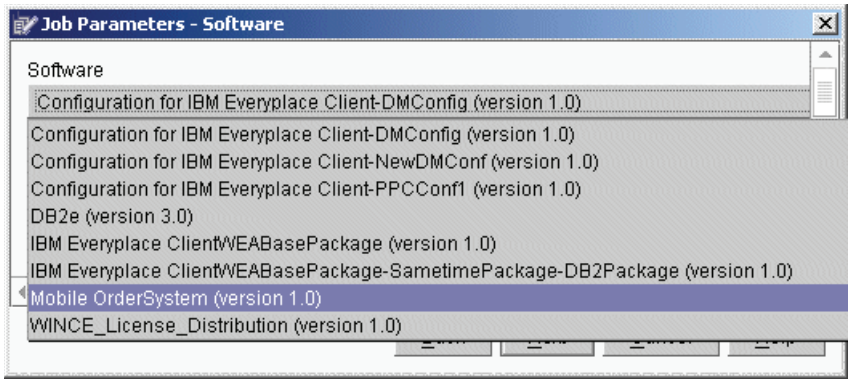


Figure 28-30 Software to be distributed

5. You will shown the job summary. Click **Next**. You will be shown a job submission status. Click **OK**.

Job progress

1. To verify the status of your job, click **Jobs**. Click **Target Device Class**. Select **Return anything with a device class that matches**. Select **WinCE** and click **OK**.

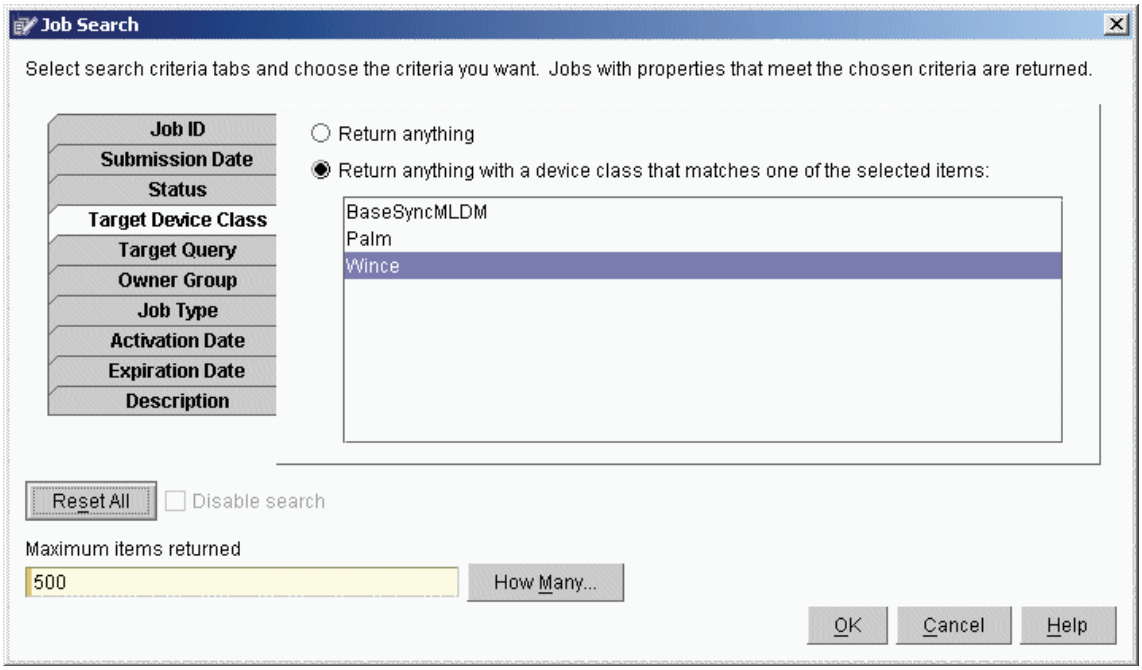


Figure 28-31 Job search

2. You should be able see the job in your list. Right-click the job and select the **View Job Progress Summary** option. This will show the current status of the job.

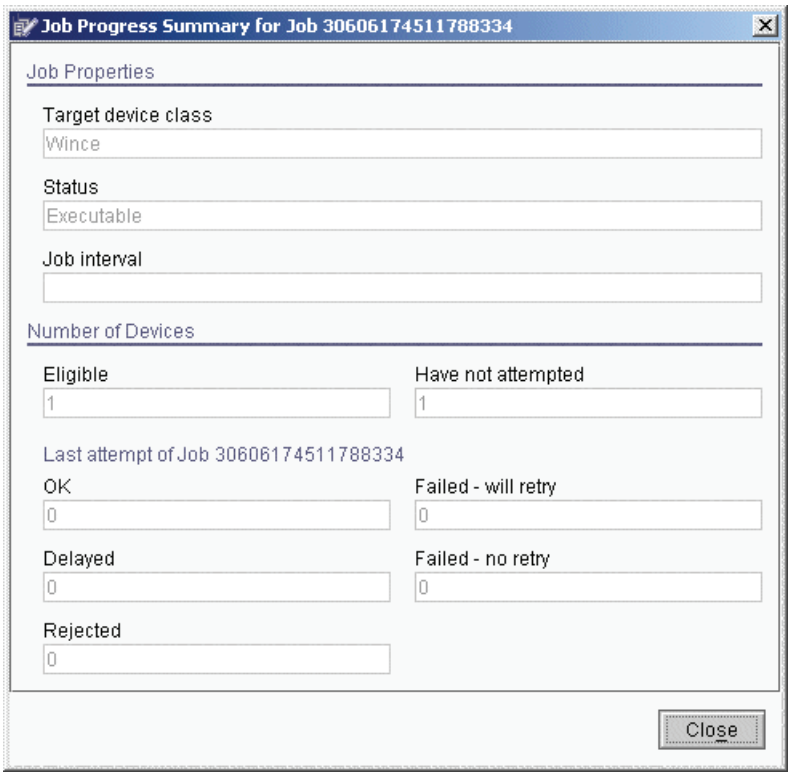


Figure 28-32 Job progress summary

Connect to Device Management Server using a Windows CE device

1. Start the Everyplace Client in the Pocket PC device. Enter the user ID and password and click **Login**.
2. In the Home tab, click **Device Agent**. This will start a device management session with the Device Management Server, which will execute the jobs applicable for this device.

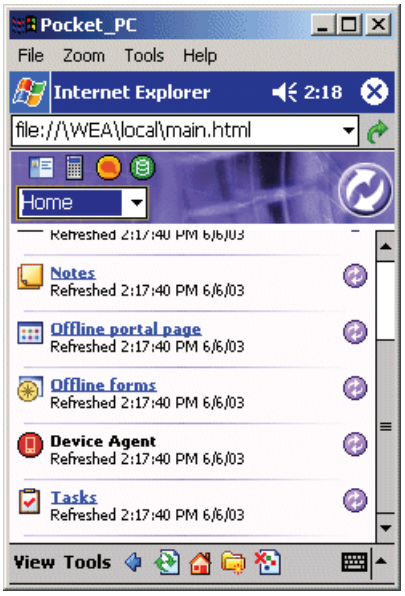


Figure 28-33 Device Agent

3. Verify if the application has been installed.



Figure 28-34 Verifying the install

4. Device Management Server administrators can check the status of their jobs through the Device Services console. You can see that the last attempt of the job was OK.

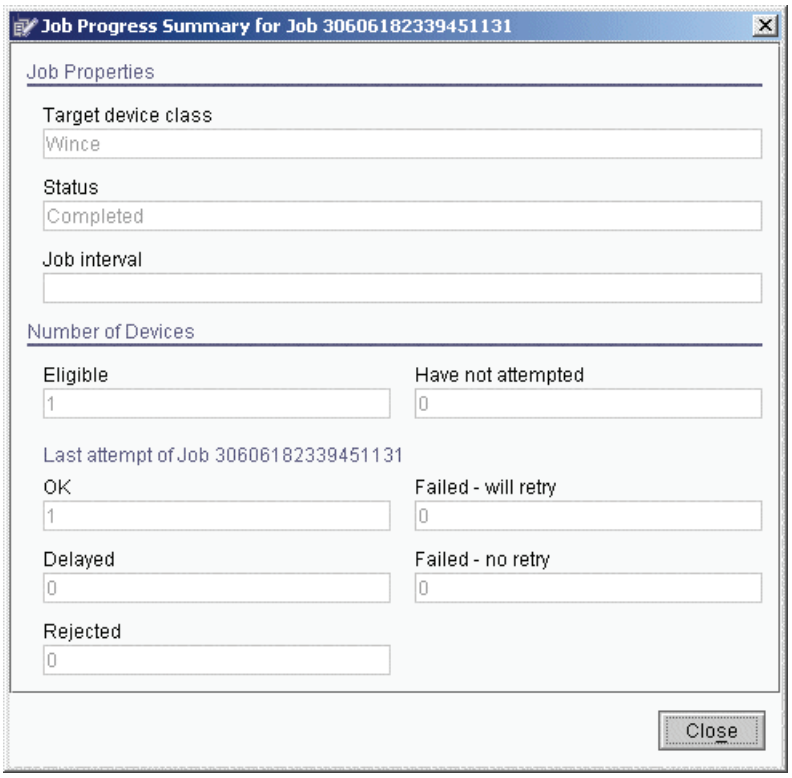


Figure 28-35 Job progress after executing the job on a device

28.3.3 Configuring SSL on a Palm device

With this scenario we will show how to work with Palm devices to connect to the Device Management Server. Also we will show you how to create a configuration job. We will be creating a job to enable Palm to connect using SSL.

Steps to connect to the Device Services

For information on how to install and configure Everyplace Access Client components on Palm devices, refer to Chapter 4, “Everyplace Client” on page 81.

Note: For the Device Agent to work on Palm devices, it is necessary to have IBM Device Services conduit for Palm OS installed on the host PC. The program is included in the IBM Everyplace Client CD.

1. Click the applications and click **IBM Agent**.

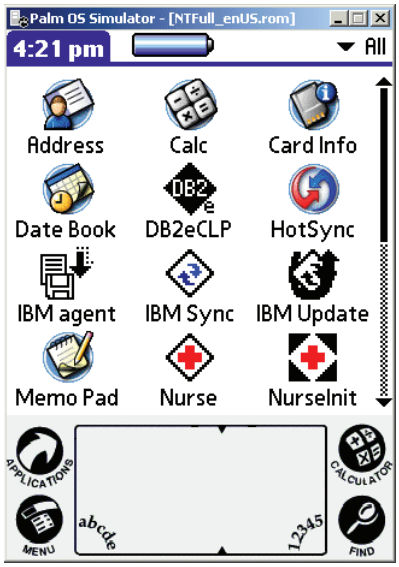


Figure 28-36 IBM Agent for Palm

2. Click the top of the next screen to configure the server and the login settings.

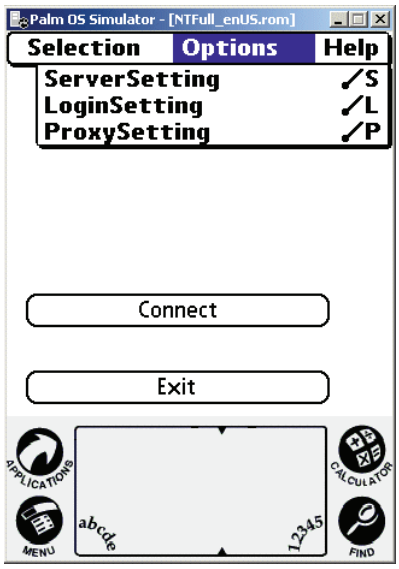


Figure 28-37 Settings

3. Select **Server Setting**, enter the address of your server, and click **OK**.

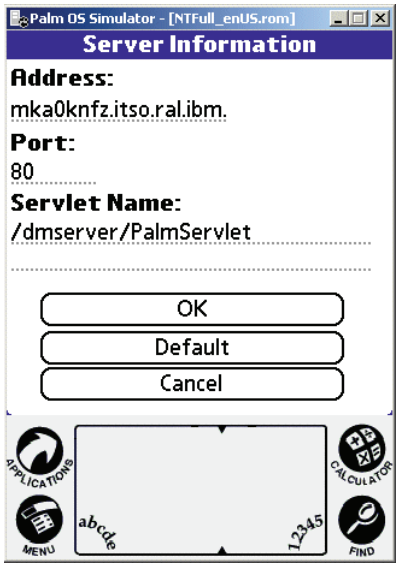


Figure 28-38 Server Settings

4. Select **Login Setting** and enter the user name and password of your account on the Everyplace Access server. Enter `HotSync` as the service name.



Figure 28-39 Login configuration

5. Click **OK**. This should bring up the following screen. Click **Connect**.

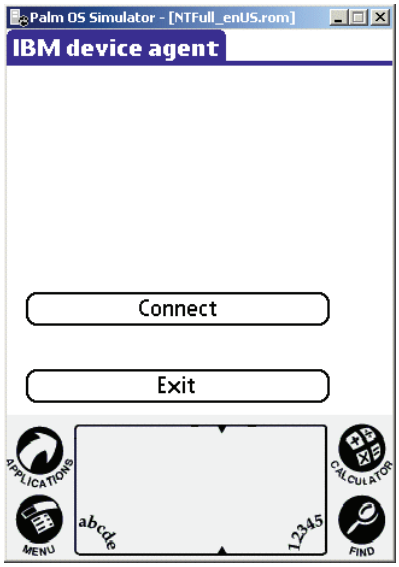


Figure 28-40 IBM Device Agent

6. This will connect tot the Device Management Server. Since this is the first time the device is connecting to the server, an inventory scan will be performed.

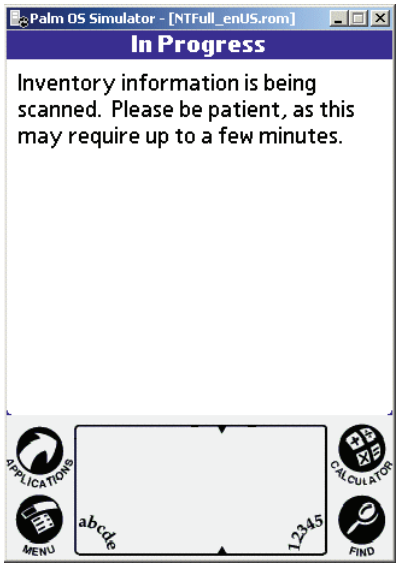


Figure 28-41 Device agent at work

7. You will get a message when the inventory scan is over.

Creating a SSL configuration job

1. Open the Device Services console. Click **Devices** and select **Palm devices** as shown in Figure 28-42 on page 1051.

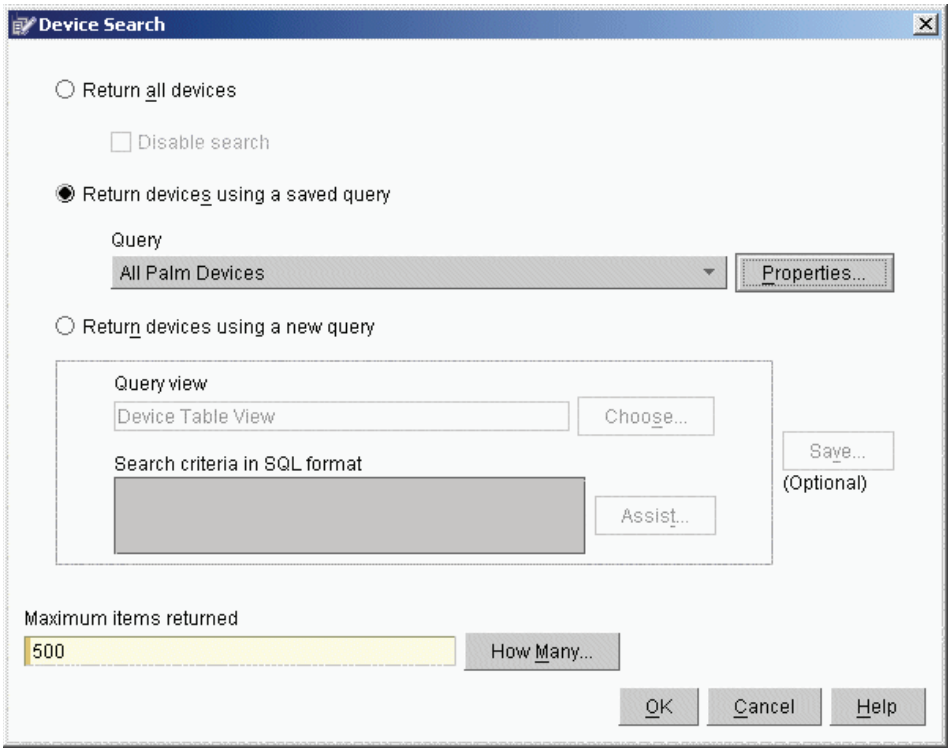


Figure 28-42 Querying Palm devices

2. Right-click the Palm device you want to configure SSL and click **Submit Job**.

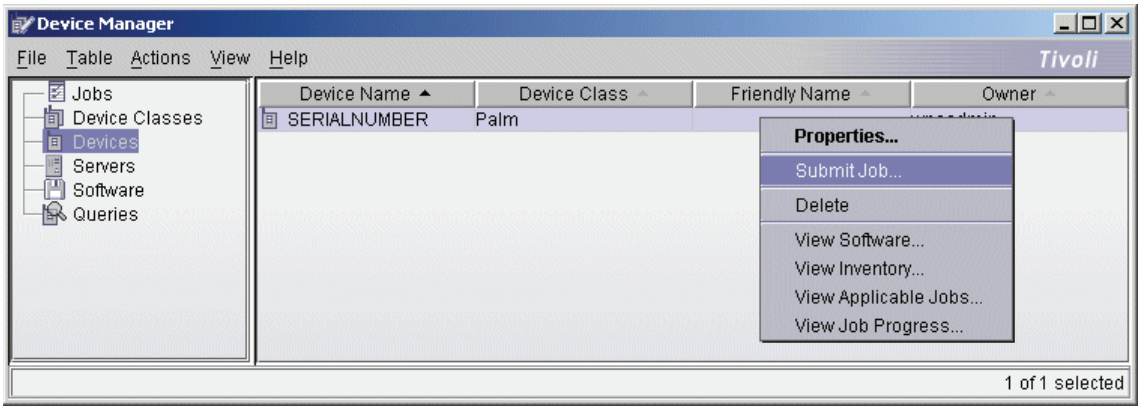


Figure 28-43 Creating a job

Tip: If you want to configure SSL on all the Palm devices that connect, select **Device Classes** and right-click the Palm devices and select **Submit Job**.

3. Select **Configuration job** as the job type. Enter a suitable description for the job. Click **Next**.

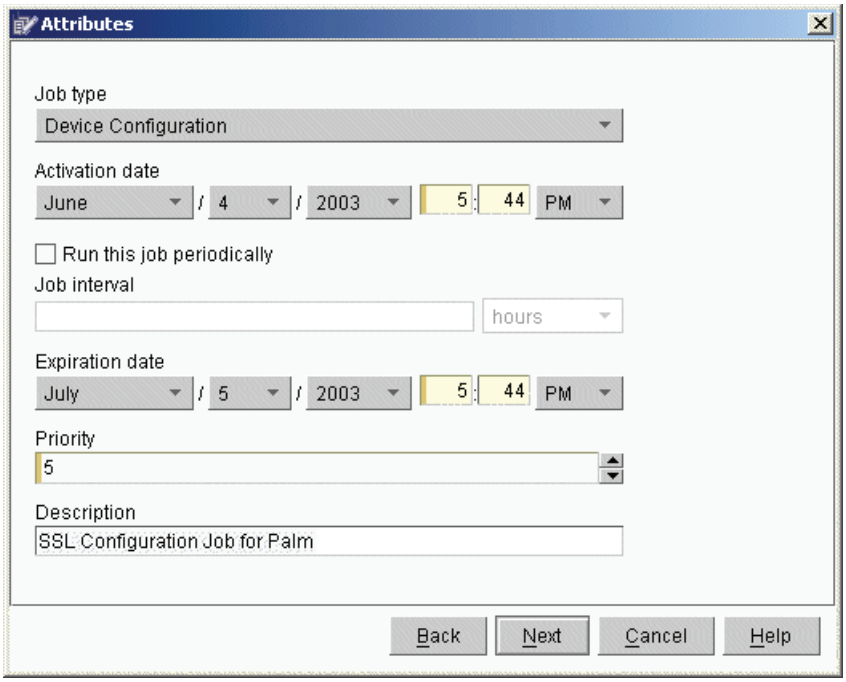


Figure 28-44 Job attributes

4. Select the **Agent** tab. In the SSL On drop-down list, select **On**. Enter 443 as the Management Server port and click **Next**.

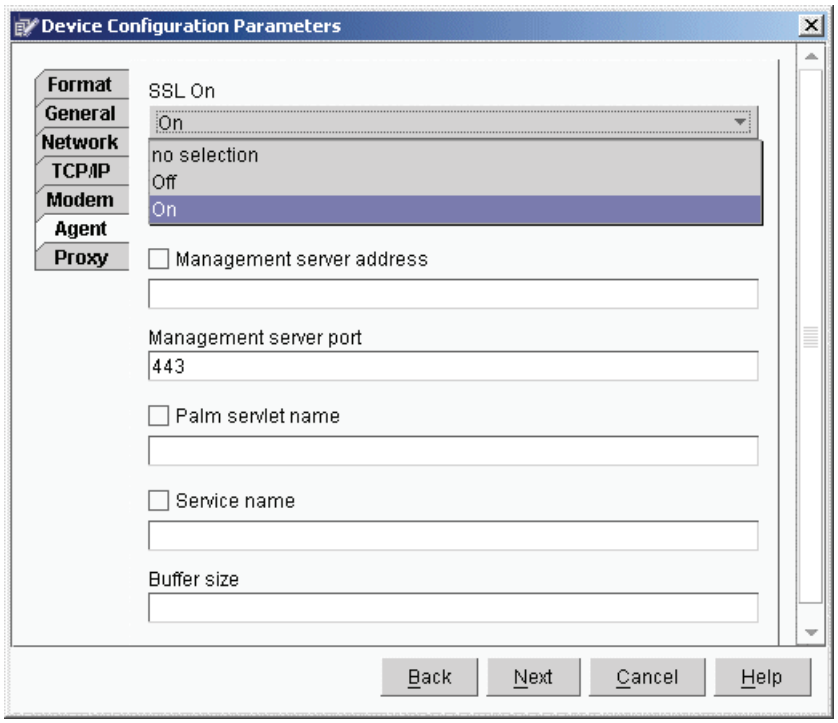


Figure 28-45 Device Agent configuration

5. Click **Next**. This will show you a window for which devices the job is submitted. Click **OK**. A message stating that the job has been successful comes up. Click **OK**.
6. Connect to the Device Services using the IBM Agent on the Palm device.
7. You will get the following messages:

```
Connect to Network Dispatcher
Connect to Management Server
Post Job Result
Complete
Configuration Job Complete
```

8. Exit and start the IBM Agent again. Check the Server Settings. Connect to the server to check if the IBM Agent can communicate using SSL on port 443.

28.4 Device management for Sharp Zaurus

You will need to install the Zaurus support in the Device Management Server prior to use the Device Agent on Zaurus devices. The tasks involved are first to create the database for Zaurus devices and then to install the support classes. Refer to the InfoCenter for detailed steps about these tasks.

28.5 Troubleshooting

There could be situations where you may need to view the Device Management Server traces to check the status of the server and the jobs submitted. By default the tracing is disabled. The trace log is primarily a development tool and should be enabled when troubleshooting. Enabling the trace log can affect your system performance. Extended use of trace logging can also consume large amounts of disk space. To manually enable tracing, do the following steps:

1. Go to the directory where WebSphere is installed and navigate to the subdirectory WebSphere\AppServer\installedApps\yourservername_DMS_WebApp.ear\dmserver.war\WEB-INF\classes.

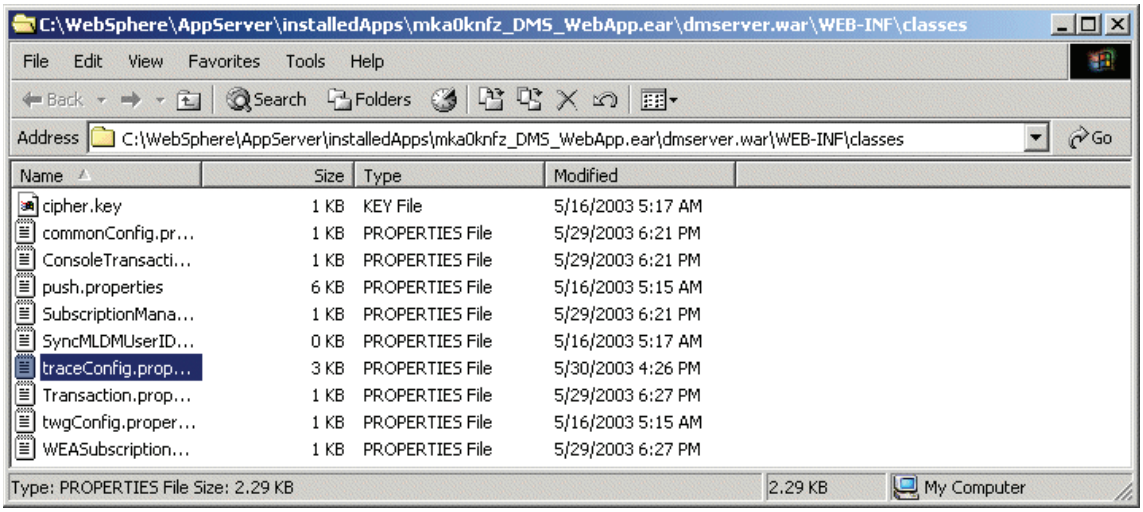


Figure 28-46 File path for enabling trace

2. Set Enable Trace to true.

```
#Global flag to turn tracing on
EnableTrace=true
```

Figure 28-47 Global flag for enabling tracing

3. Near the bottom of the file are the trace flags for each of the Device Services components. Set the appropriate flags on depending on the problem faced. For example if there is a problem with plug-ins you could set the traceEnabled.plugins to true.

```
#These flags determine if trace in enabled for the component
traceEnabled.dmserver=true
traceEnabled.event=false
traceEnabled.notification=false
traceEnabled.plugins=false
traceEnabled.resultscollector=false
traceEnabled.twgapi=false
traceEnabled.database=false
traceEnabled.enrollserver=false
traceEnabled.datconverter=false
traceEnabled.mcollect=false
#traceEnabled.test=false
```

Figure 28-48 Flags for different components of Device Services

4. Save your changes.
5. Open a command window and navigate to the directory WebSphere\DMS\bin.
6. Run the command **`server -app dmserver -trace.set`**. This will prompt the Device Services to re-read the trace settings and act on them.
7. Perform the device management tasks. To view the trace logs navigate to WebSphere\AppServer\logs and open the file DMS_stdout.log.

29

Everyplace Client API for Pocket PC 2002

In this chapter, we introduce you to the use of the Everyplace Client application programming interface for Pocket PC 2002 to develop a customized client for synchronization.

© Copyright IBM Corp. 2003. All rights reserved.

29.1 Overview

In this version of Everyplace Client, an application programming interface (API) has been defined for the Pocket PC 2002 device platform. With this API, the user application can synchronize data, use the stop request, and get requested status of components supported by Everyplace Client. The API is defined as URL format, and the user application must perform a GET method HTTP request to localhost (127.0.0.1) with port 8080 to send the request.

The user application does not need to include any programming interface headers or any link to a specific API link library.

29.2 Usage

To access this API, you can use any programming platform that supports HTTP development. For example, you can use the J2ME Connected Device Configuration (CDC) that implements the java.net.* package. WebSphere Studio Device Developer supports J2ME development and includes an implementation for the CDC configuration. Also it provides an implementation for the Personal Profile (PPC) that provides GUI support using the Abstract Window Toolkit (AWT).

Each command in the API is represented by a specific URL. You have to access this URL to execute the command and get the response code to obtain information about the command execution.

Note that the Everyplace Client application must be started and the user logged in when you are using the API.

29.2.1 Commands available

There are three commands available in the API. The following is a summary of these commands. For the complete reference, refer to the WebSphere Everyplace Access online library at:

http://www-3.ibm.com/software/pervasive/products/library/ws_everyplace_access.shtml

Sync command

You can use this command to start a synchronization session, specifying the components to synchronize. The basic syntax is:

```
http://127.0.0.1:8080/apisync?cmd=sync&comp=<component>[&comp=<component>...]
```

where `component` can take one of this values:

- `msmail`, for e-mail synchronization
- `contacts`, for Contacts synchronization
- calendar, for Calendar synchronization
- `todos`, for To Do (Tasks) synchronization
- `npad`, for Notes (Memo) synchronization
- `dmsagent`, for Device Manager (jobs) synchronization
- `db2eISync`, for DB2 Everyplace synchronization
- `offlineForms`, for Offline Forms synchronization
- `offlineportal`, for Offline Browsing synchronization

For DB2e synchronization, there is an additional syntax to specify a particular subscription to synchronize. Refer to the Everyplace API product reference documentation for more information.

Examples

- To synchronize e-mail:

  ```
  http://127.0.0.1:8080/apisync?cmd=sync&comp=msmail
  ```

- To synchronize e-mail and contacts:

  ```
  http://127.0.0.1:8080/apisync?cmd=sync&comp=msmail&comp=contacts
  ```

Stop command

You can use this command to stop a synchronization session currently in progress. If you have multiple components to synchronize, this command only stops the current and the subsequent components. The syntax is:

```
http://127.0.0.1:8080/apisync?cmd=stop
```

Status command

You can use this command to detect if a synchronization session is taking place. The status code is returned as an HTTP response code. The syntax is:

```
http://127.0.0.1:8080/apisync?cmd=getstatus
```

The return codes for the status command are shown in Table 29-1.

Table 29-1 Return codes for the status command

Code	Description
700	No sync is taking place
701	A sync is taking place
710	No connection was established

Code	Description
602	Last sync failed
No response	Communication error

29.3 Developing a custom synchronization client

As an example of the usage of the Synchronization API, we will develop a customized WebSphere Everyplace Access Client using the WebSphere Studio Device Developer software. For the HTTP support, we use the CDC configuration and for GUI support the PPC configuration.

You have to create a J2ME project that uses the CDC and PPC configurations. Also you have to configure the environment to execute the application. Refer to Chapter 18, “DB2 Everyplace applications with WebSphere Studio Device Developer” on page 529 for details about the use of the WebSphere Studio Device Developer to create a J2ME application for Pocket PC devices that use the CDC and PPC configurations.

29.3.1 Java classes for the customized client

The WebSphere Everyplace Access customized client is composed of four classes.

SyncException

This class encapsule the HTTP response error codes. For this example, all the codes listed in the class are considered as exceptions.

The code for the class is shown in Example 29-1.

Example 29-1 SyncException.java

```
package com.ibm.ral.itso.ppcapi;

/**
 * Custom exception for the sync API
 *
 * This class encapsule the error codes for the API
 *
 */
public class SyncException extends Exception {

   // Sync command return codes
   public final static int SYNC_FORBIDDEN = 403;
```

```
    public final static int SYNC_ERROR_INVALID_REQUEST = 601;
    public final static int SYNC_STATUS_SYNCING = 701;
    public final static int SYNC_STATUS_NO_CONNECTION = 710;

    // Stop command return codes
    public final static int STOP_STATUS_NO_CONNECTION = 710;

    private int errorCode;

    /**
     * Constructor for SyncException.
     */
    public SyncException(int errorCode) {
        super("Error: "+ errorCode);
        this.errorCode = errorCode;
    }


    /**
     * Returns the errorCode.
     * @return int
     */
    public int getErrorCode() {
        return errorCode;
    }

}
```

SyncEngine

This class encapsules the access to the HTTP engine with the corresponding commands in three methods: sync(), stop() and getStatus().

The HTTP access is located in the private method execute(). This method uses the java.net.URL class to make a connection to the HTTP engine, represented as an HTTPConnection object, and then uses the getResponseCode() method to obtain the response code for the corresponding request.

The sync and stop methods call the execute method and if the response is not OK, they throw a SyncException with the corresponding errorCode.

The code for the class is shown in Example 29-2.

Example 29-2 SyncEngine.java

```
package com.ibm.ral.itso.ppcapi;

import java.net.HttpURLConnection;
import java.net.URL;
```

```
import java.net.URLConnection;

/**
 * Represents the synchronization engine for the client.
 *
 * This class encapsules the net calls to access the sync engine.
 */
public class SyncEngine extends MyWEAClient {

   private final static int SYNC_OK = 200;
   private final static int STOP_OK = 200;

   // Status command return codes
   public final static int STATUS_IDLE = 700;
   public final static int STATUS_SYNCING = 701;
   public final static int STATUS_NO_CONNECTION = 710;
   public final static int STATUS_SYNC_FAILED = 602;

   private final static int NO_RESPONSE = -1;


   private final static String WEA_ENGINE_URL
      = "http://127.0.0.1:8080/apisync";
   private final static String SYNC_COMMAND = "sync";
   private final static String STOP_COMMAND = "stop";
   private final static String STATUS_COMMAND = "getstatus";

   /**
    * Execute an URL command using the java.net.URL class
    *
    * @param command The command to execute, can be SYNC_COMMAND ,
    * STOP_COMMAND or STATUS_COMMAND command.
    * @param parameters The parameters for the command as defined in the
    * Everyplace API reference
    */
   private static int execute(String command, String parameters) {

      int responseCode = NO_RESPONSE;

      if (parameters == null) {
         parameters = "";
      }

      try {
         String urlString = WEA_ENGINE_URL+"?cmd="+command+parameters;

         URL syncUrl = new URL(urlString);
         HttpURLConnection conn =
            (HttpURLConnection)syncUrl.openConnection();
```

```
            responseCode = conn.getResponseCode();
            conn.disconnect();
        }
        catch(Exception e) {
            e.printStackTrace();
        }

        return responseCode;
    }

    /**
     * Execute the sync command with parameters
     */
    public static void sync(String parameters) throws SyncException {

        int responseCode = execute(SYNC_COMMAND, parameters);

        if (responseCode != SYNC_OK) {
            throw new SyncException(responseCode);
        }
    }

    /**
     * Execute the stop command
     */
    public static void stop()  throws SyncException {
        int responseCode = execute(STOP_COMMAND, null);
        if (responseCode != SYNC_OK) {
            throw new SyncException(responseCode);
        }
    }

    /**
     * Return the status of a current synchronization
     */
    public final static int getStatus() {
        return execute(STATUS_COMMAND, null);
    }
}
```

MainFrame

This class contains the GUI for the client. The GUI has two parts:

- A panel for the components selection check box
- A panel for the Control Message and Synchronize button

When you choose one or more components to synchronize and click the **Synchronize** button, the command is built in the syncCommand() method and

then it is passed to the SyncEngine.sync() method. Immediately, the method checks continuously for the status code, and when it is different from SYNCING, the loop stops and the last status code is verified. If it is IDLE, it means that the synchronization was successful and if it is FAILED, the sync failed. Note that the API has no way to know why the synchronize session failed.

Example 29-3 MainFrame.java

```
package com.ibm.ral.itso.ppcapi;

import java.awt.Button;
import java.awt.Checkbox;
import java.awt.Component;
import java.awt.Frame;
import java.awt.Menu;
import java.awt.MenuBar;
import java.awt.MenuItem;
import java.awt.Panel;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * GUI for the customized client
 *
 */
public class MainFrame extends Frame {

     private Panel controlPanel = null;
     private Panel selectionPanel = null;

     private TextField messagesField = null;

     private Button syncButton = null;

     private Checkbox contactsCheckBox = null;
     private Checkbox calendarCheckBox = null;
     private Checkbox todoCheckBox = null;
     private Checkbox notesCheckBox = null;
     private Checkbox dmsCheckBox = null;
     private Checkbox db2eCheckBox = null;
     private Checkbox offlineFormsCheckBox = null;
     private Checkbox offlinePortalCheckBox = null;
     private Checkbox mailCheckBox = null;

     private MenuBar mainMenuBar = null;


    /**
```

```
 * Constructor for MainFrame
 *
 */
public MainFrame() {
   super();
   initialize();
}

/**
 * This method initializes the graphic controls
 *
 */
private void initialize() {
     this.add(getControlPanel(), java.awt.BorderLayout.NORTH);
     this.add(getSelectionPanel(), java.awt.BorderLayout.CENTER);
     this.setMenuBar(getMainMenuBar());
     this.setTitle("My WEA Client");

}

/**
 * This method initializes the main menu bar.
 *
 * The menu bar has an option for close the application
 */
private MenuBar getMainMenuBar() {
   if (mainMenuBar == null) {
      mainMenuBar = new MenuBar();
      mainMenuBar.add(new Menu("Actions"));
      mainMenuBar.getMenu(0).add(new MenuItem("Close"));
      mainMenuBar.getMenu(0).getItem(0).addActionListener(
         new ActionListener() {
               public void actionPerformed(ActionEvent event) {
                  closeCommand();
               }
         });
   }

   return mainMenuBar;
}

/**
 * This method initializes the Control panel
 *
 * The panel has the message field and the Synchronize button
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getControlPanel() {
```

```
   if(controlPanel == null) {
      controlPanel = new java.awt.Panel();
      java.awt.FlowLayout layFlowLayout1 = new java.awt.FlowLayout();
      layFlowLayout1.setAlignment(java.awt.FlowLayout.LEFT);
      controlPanel.setLayout(layFlowLayout1);
      controlPanel.add(getMessagesField(), null);
      controlPanel.add(getSyncButton(), null);
   }
   return controlPanel;
}

/**
 * This method initializes the Selection panel
 *
 * This panel contains all the checkboxs for choose the components
 * to synchronize.
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getSelectionPanel() {
   if(selectionPanel == null) {
      selectionPanel = new java.awt.Panel();
      java.awt.GridLayout layGridLayout2 = new java.awt.GridLayout();
      layGridLayout2.setRows(9);
      layGridLayout2.setColumns(1);
      layGridLayout2.setHgap(5);
      selectionPanel.setLayout(layGridLayout2);
      selectionPanel.add(getMailCheckBox(), null);
      selectionPanel.add(getContactsCheckBox(), null);
      selectionPanel.add(getCalendarCheckBox(), null);
      selectionPanel.add(getTodoCheckBox(), null);
      selectionPanel.add(getNotesCheckBox(), null);
      selectionPanel.add(getDmsCheckBox(), null);
      selectionPanel.add(getDb2eCheckBox(), null);
      selectionPanel.add(getOfflineFormsCheckBox(), null);
      selectionPanel.add(getOfflinePortalCheckBox(), null);
   }
   return selectionPanel;
}

/**
 * This method initializes the messages field
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getMessagesField() {
   if(messagesField == null) {
      messagesField = new java.awt.TextField();
      messagesField.setColumns(20);
```

```
      messagesField.setEditable(false);
      messagesField.setEnabled(false);
   }
   return messagesField;
}

/**
 * This method initializes the Synchronize button.
 *
 * Also it associate the syncCommand() method to the button
 *
 * @return java.awt.Button
 */
private java.awt.Button getSyncButton() {
   if(syncButton == null) {
      syncButton = new java.awt.Button();
      syncButton.setLabel("Synchronize");
      syncButton.setName("syncButton");
      syncButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent Event) {
            syncCommand();
         }
      });
   }
   return syncButton;
}

/**
 * This method initializes the Contacts checkbox
 *
 * @return java.awt.Checkbox
 */
private java.awt.Checkbox getContactsCheckBox() {
   if(contactsCheckBox == null) {
      contactsCheckBox = new java.awt.Checkbox();
      contactsCheckBox.setLabel("Contacts");
      contactsCheckBox.setFont(new java.awt.Font("sansserif", 1, 12));

   }
   return contactsCheckBox;
}

/**
 * This method initializes Calendar chechbox
 *
 * @return java.awt.Checkbox
 */
private java.awt.Checkbox getCalendarCheckBox() {
   if(calendarCheckBox == null) {
```

```
            calendarCheckBox = new java.awt.Checkbox();
            calendarCheckBox.setLabel("Calendar");
            calendarCheckBox.setFont(new java.awt.Font("sansserif", 1, 12));
        }
        return calendarCheckBox;
    }

    /**
     * This method initializes To Do checkbox
     *
     * @return java.awt.Checkbox
     */
    private java.awt.Checkbox getTodoCheckBox() {
        if(todoCheckBox == null) {
            todoCheckBox = new java.awt.Checkbox();
            todoCheckBox.setLabel("To Do");
            todoCheckBox.setFont(new java.awt.Font("sansserif", 1, 12));
        }
        return todoCheckBox;
    }

    /**
     * This method initializes Notes checkbox
     *
     * @return java.awt.Checkbox
     */
    private java.awt.Checkbox getNotesCheckBox() {
        if(notesCheckBox == null) {
            notesCheckBox = new java.awt.Checkbox();
            notesCheckBox.setLabel("Notes");
            notesCheckBox.setFont(new java.awt.Font("sansserif", 1, 12));
        }
        return notesCheckBox;
    }

    /**
     * This method initializes DMS checkbox
     *
     * @return java.awt.Checkbox
     */
    private java.awt.Checkbox getDmsCheckBox() {
        if(dmsCheckBox == null) {
            dmsCheckBox = new java.awt.Checkbox();
            dmsCheckBox.setLabel("Device Manager");
            dmsCheckBox.setFont(new java.awt.Font("sansserif", 1, 12));
        }
        return dmsCheckBox;
    }
```

```
/**
 * This method initializes DB2e checkbox
 *
 * @return java.awt.Checkbox
 */
private java.awt.Checkbox getDb2eCheckBox() {
   if(db2eCheckBox == null) {
      db2eCheckBox = new java.awt.Checkbox();
      db2eCheckBox.setLabel("DB2 Everyplace");
      db2eCheckBox.setFont(new java.awt.Font("sansserif", 1, 12));
   }
   return db2eCheckBox;
}

/**
 * This method initializes Offline Forms checkbox
 *
 * @return java.awt.Checkbox
 */
private java.awt.Checkbox getOfflineFormsCheckBox() {
   if(offlineFormsCheckBox == null) {
      offlineFormsCheckBox = new java.awt.Checkbox();
      offlineFormsCheckBox.setLabel("Offline Forms");
      offlineFormsCheckBox.setFont(new java.awt.Font("sansserif", 1, 12));
   }
   return offlineFormsCheckBox;
}

/**
 * This method initializes Offline Browsing checkbox
 *
 * @return java.awt.Checkbox
 */
private java.awt.Checkbox getOfflinePortalCheckBox() {
   if(offlinePortalCheckBox == null) {
      offlinePortalCheckBox = new java.awt.Checkbox();
      offlinePortalCheckBox.setLabel("Offline Portal");
      offlinePortalCheckBox.setFont(new java.awt.Font("sansserif", 1, 12));
   }
   return offlinePortalCheckBox;
}

/**
 * This method initializes Mail checkbox
 *
 * @return java.awt.Checkbox
 */
private java.awt.Checkbox getMailCheckBox() {
   if(mailCheckBox == null) {
```

```
            mailCheckBox = new java.awt.Checkbox();

            mailCheckBox.setLabel("Mail");
            mailCheckBox.setFont(new java.awt.Font("sansserif", 1, 12));
        }
        return mailCheckBox;
    }

    /**
     * Execute the synchronization of the components specified by the user
     *
     */
    private void syncCommand() {

        StringBuffer command = new StringBuffer();

        // Building the synchronization command
        if (getMailCheckBox().getState()) {
            command.append("&comp=msmail");
        }

        if (getContactsCheckBox().getState()) {
            command.append("&comp=contacts");
        }

        if (getCalendarCheckBox().getState()) {
            command.append("&comp=calendar");
        }

        if (getTodoCheckBox().getState()) {
            command.append("&comp=todos");
        }

        if (getNotesCheckBox().getState()) {
            command.append("&comp=npad");
        }

        if (getDmsCheckBox().getState()) {
            command.append("&comp=dmsagent");
        }

        if (getDb2eCheckBox().getState()) {
            command.append("&comp=db2eISync");
        }

        if (getOfflineFormsCheckBox().getState()) {
            command.append("&comp=offlineForms");
        }
```

```
if (getOfflinePortalCheckBox().getState()) {
   command.append("&comp=offlineportal");
}

if (command.toString().equals("")) {
   getMessagesField().setText("No sync requested");
   return;
}

// Disable all controls
Component[] components =  getSelectionPanel().getComponents();
for (int i = 0; i < components.length; i++) {
   if (components[i] instanceof Checkbox) {
      components[i].setEnabled(false);
   }
}
getSyncButton().setEnabled(false);


try {
   // Init the synchronization
   SyncEngine.sync(command.toString());

   getMessagesField().setText("Syncing...");
   int statusCode;

   // Wait for a state other than syncing
   while(true) {
      statusCode = SyncEngine.getStatus();
      if (statusCode != SyncEngine.STATUS_SYNCING) break;
   }

   String statusMessage;
   if (statusCode == SyncEngine.STATUS_IDLE) {
      statusMessage = "Sync successful.";
   }
   else {
      statusMessage = "Sync failed. No further information.";
   }
   getMessagesField().setText(statusMessage);
}
catch(SyncException e) {
   e.printStackTrace();
   int errorCode = e.getErrorCode();
   String errorMessage = "Sync failed. ";
   switch(errorCode) {
      case SyncException.SYNC_ERROR_INVALID_REQUEST:
         errorMessage += "Invalid request.";
         break;
```

```
                case SyncException.SYNC_FORBIDDEN:
                   errorMessage += "No log in.";
                   break;
                case SyncException.SYNC_STATUS_SYNCING:
                   errorMessage += "Another sync is taking place.";
                   break;
                case SyncException.SYNC_STATUS_NO_CONNECTION:
                   errorMessage += "No connection was established.";
                   break;
                default:
                   errorMessage += "No further information.";
             }
             getMessagesField().setText(errorMessage);
          }

          // Enable all controls
          for (int i = 0; i < components.length; i++) {
             if (components[i] instanceof Checkbox) {
                components[i].setEnabled(true);
             }
          }

          getSyncButton().setEnabled(true);
       }

       /**
        * Close the frame
        */
       private void closeCommand() {
          this.setVisible(false);
          this.dispose();
       }
}
```

MyWEAClient

This class starts the MainFrame class and serves as the application entry point.

The code for the class is shown in Example 29-4.

Example 29-4 MyWEAClient.java

```
package com.ibm.ral.itso.ppcapi;

import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;

/**
```

```
 * Main class for the WEA client
 *
 */
public class MyWEAClient {

   public static void main(String[] args) {
      new MainFrame().show();
   }

}
```

29.3.2 Running the sample

If you run the sample, you will see the GUI for the client as shown in Figure 29-1.

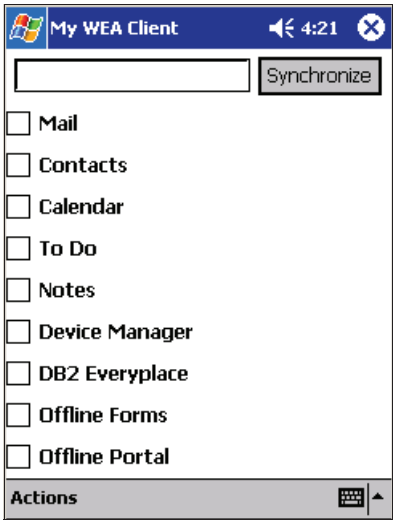


Figure 29-1 MyWEAClient

To start a synchronization session, select the components that you wish synchronize and click **Synchronize.** You will see the message field with the sync progress, as shown in Figure 29-2 on page 1074, and you may see the title of the client change.

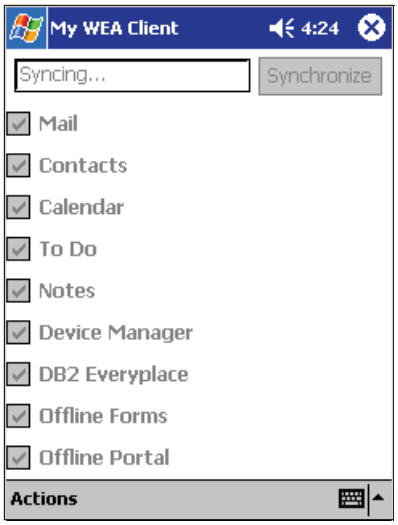


Figure 29-2 Syncing the selected components

At the end you will see the result of the synchronization as shown in Figure 29-3.



Figure 29-3 Sync result

If you go to the Everyplace Client unified interface, you can check that synchronization really was done.

29.4 Hints and tips

If you obtain a 600 HTTP response code, make sure that you typed the command correctly. For example, if you use:

```
http://127.0.0.1:8080/apisync?cmd=status
```

instead of:

```
http://127.0.0.1:8080/apisync?cmd=getstatus
```

you will receive the 600 error code and the synchronization request will fail.

A

Everyplace Access sample installation instructions

This appendix provides the instructions for installing WebSphere Everyplace Access Version 4.3 on a Windows 2000 Server platform. This sample installation and configuration can be used to run and exercise most of the sample scenarios included in this redbook.

© Copyright IBM Corp. 2003. All rights reserved.

System requirements

In this section we list the WebSphere Everyplace Access V4.3 installation requirements.

Hardware requirements

- Intel Pentium III 1 GHz processor (minimum)
- Memory of 1.5 GB (minimum); however 2 GB is recommended
- Disk space 580 MB for installation (minimum)
- 540 MB for data storage (minimum)
- CD-ROM drive
- Mouse or pointing device and keyboard
- TCP/IP network software installed
- SVGA or better display resolution

Note that these requirements are the minimum. In our case the installation takes around 5 GB of disk space.

Software requirements

- Windows 2000 (Server or Advanced Server) with Service Pack 3
- Lotus Client 6 or later and/or Microsoft Outlook for access to Lotus Domino Servers or Microsoft Exchange V5.5 servers

Planning for installation

Prior to installing WebSphere Everyplace Access, some initial setup is required:

1. Setting up TCP/IP: A fixed IP address is required. You can use the Microsoft Loopback Adapter if you are installing WebSphere Everyplace Access in a stand-alone machine. The machine must have a fully qualified domain name. You can test the network connectivity by pinging the server name from a command window.
2. (Optional) Installing Lotus Notes client V6 and/or Microsoft Outlook Client

Setup Manager

Setup Manager, the installation program that installs many of the Everyplace Access components, does the following:

- Automatically installs IBM JRE Version 1.3.1, which is required, if it does not exist on the machine.

- Checks for existing components in the environment. If a previous version of a component exists, it can be updated. If a current version of a component exists, that component is not installed.
- Pre-installation verification of the following:
  - Operating system and its prerequisites
  - RAM requirements
  - System paging settings
  - Disk space available (done after selecting components)
- Installs the following components and their corequisite components:
  - Everyplace Synchronization Server (this also installs DB2 Everyplace, WebSphere Portal, WebSphere Application Server Advanced Edition, IBM HTTP Server, and DB2 Universal Database)
  - IBM Directory Server (this also installs IBM HTTP Server and DB2 Universal Database)
  - WebSphere Portal (this also installs WebSphere Application Server Advanced Edition, IBM HTTP Server, and DB2 Universal Database)
  - Device Management Services
  - Intelligent Notification Services
  - Location Aware Services

Note: The following components are not installed by Everyplace Access Setup Manager. Each of these components has a separate installation program.

- IBM Everyplace Client
- Lotus Domino Enterprise Server
- Microsoft Exchange 2000 or 5.5
- WebSphere Studio Site Developer Advanced

Installation instructions

This section includes instructions for the installation of WebSphere Everyplace Access and other required components.

Installing Lotus Notes client

Important: If you are going to be accessing a Lotus Domino Server from your WebSphere Everyplace Access server, the Lotus Notes client must be installed on the server machine *first*. If this step is not completed prior to installing Everyplace Access, the connection between the Everyplace Access server and the Domino Server will fail.

If you are not using Lotus Domino, you can skip this part.

1. Install the Lotus Notes client, if it is not already installed.
2. Follow the on-screen instructions, accepting the default options, to install the Lotus Notes client.

Refer to Chapter 22, “PIM and e-mail synchronization with Domino Server” on page 711 for complete information about how to configure the Everyplace Sync Server to access a Domino Server.

Installing Microsoft Outlook

Important: If you are going to be accessing a Microsoft Exchange 5.5 server from your WebSphere Everyplace Access server, Microsoft Outlook must be installed *before* installing WebSphere Everyplace Access. If this step is not completed prior to installing Everyplace Access, the connection between the Everyplace Access server and the Exchange server will fail.

If you are not using a Microsoft Exchange V5.5 server, you can skip this part.

1. Install Microsoft Outlook, if it is not already Installed.
2. Go through the steps putting in the product code, license, and so on.
3. Select **Customize**.

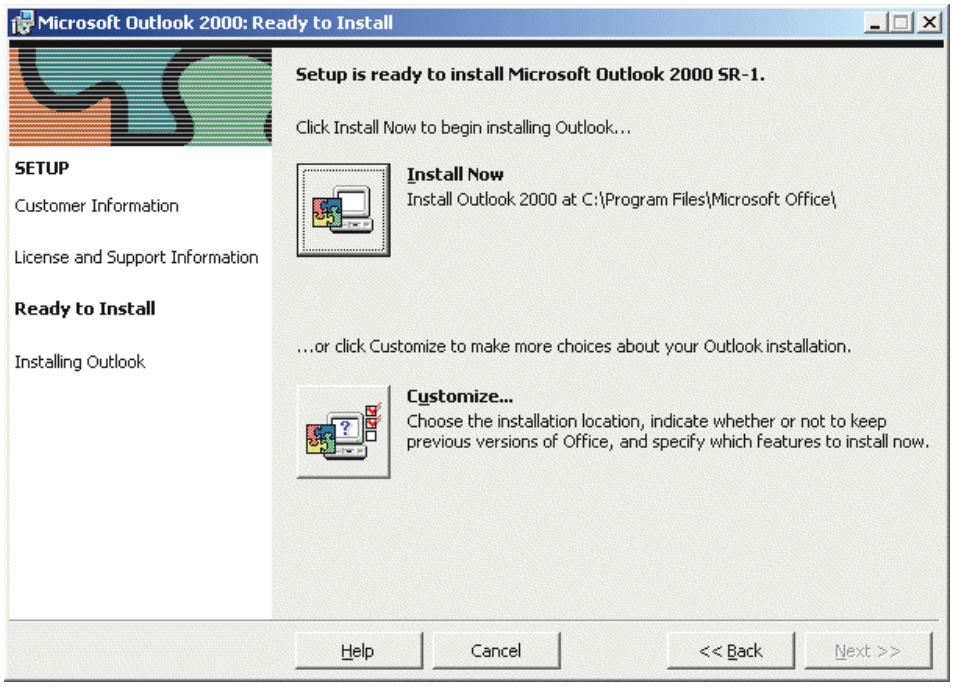


Figure A-1 Microsoft Outlook - select Customize

4. Select Microsoft Outlook for Windows only.

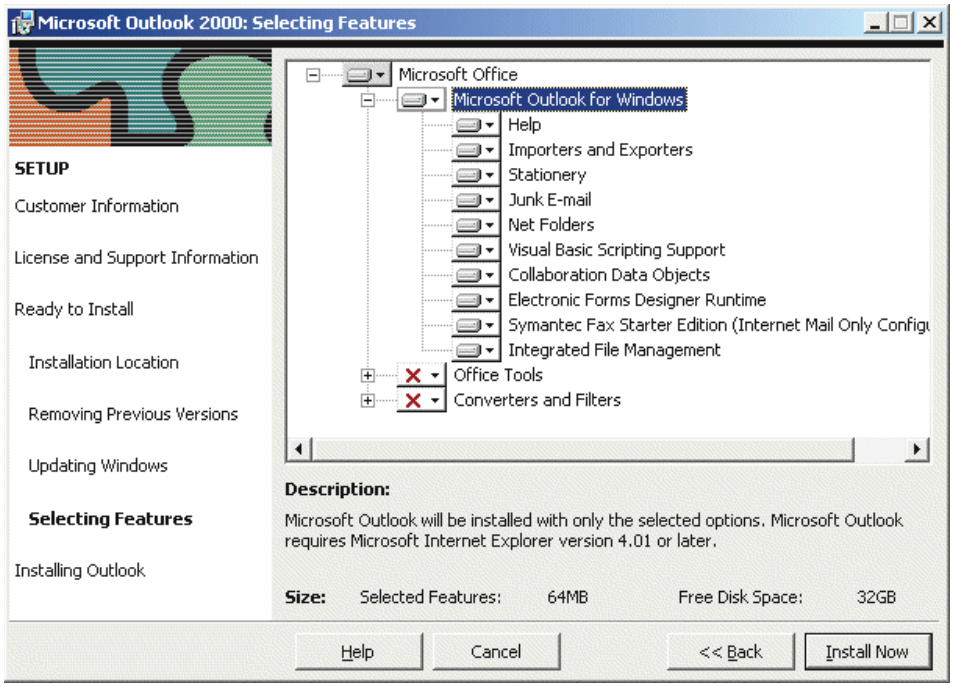


Figure A-2 Selecting Microsoft Outlook only

Making sure that port 80 is not used

The WebSphere Everyplace Access installer will install and configure the IBM HTTP Server on port 80. Before starting the installation, you have to make sure that port 80 is not used by any process in the system.

Open a command prompt. Issue the command **`netstat -an | more`.** Make sure there is no entry such as 0.0.0.0:80 on the Local Address row at the output. If there is none, you are all set to install WebSphere Everyplace Access.

If port 80 is used (there is a 0.0.0.0:80 entry at the output of the **`netstat -an`** command), you have to disable/uninstall the process which is using it. This would usually be an HTTP Server or Microsoft IIS.

Make sure IIS is not started. Open the Windows Services panel and look for IIS Admin Service. If it is started, stop it and disable it (highlight and right-click it, then set startup type to Disabled.

Setting the domain suffix

From the desktop, right-click **My Computer -> Properties**. Then on the Network Identification tab, click **Properties**.

Check that your NetBIOS computer name matches the host name you want to give to your machine. Then click **More**. In the DNS Suffix and NetBIOS Computer Name window, set your primary DNS suffix to the domain name your host pertains to (for example, `itso.ral.ibm.com`). Click **OK** to accept changes. Your computer needs to be rebooted to acknowledge the changes.

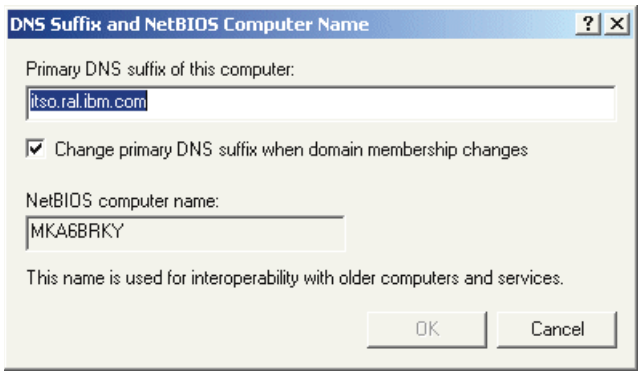


Figure A-3 Set DNS suffix

Starting the install and selecting the features to be installed

1. Log on as administrator using your password.

   Note: Make sure to log on to the local machine domain. This is not a network logon.

   The administrator user should have the following user rights:

   – Act as part of the operating system
   – Create a token object
   – Increase quotas
   – Replace a process level token

   These user rights may be checked by selecting **Start -> Settings -> Control Panel -> Administrative Tools -> Local Security Policy ->Local Policies -> User Rights Assignment**. You have to reboot your machine if one of these rights is missing and you need to add it.

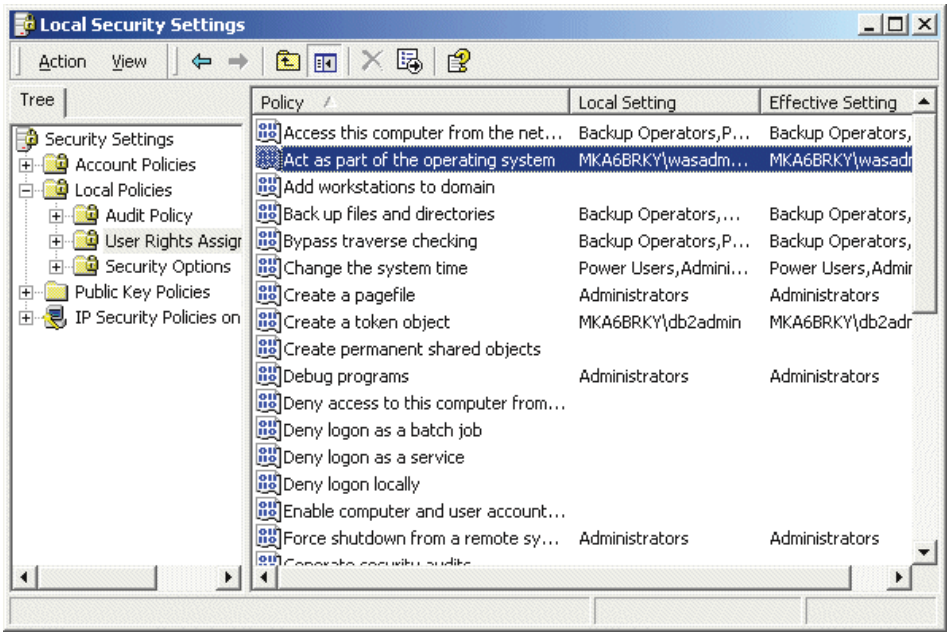


Figure A-4 Local security settings

2. Insert CD 1. The installer should autorun. However if it does not, you can run the Setup Manager by entering `D:\install.bat` (where `D` is the letter of your CD-ROM drive) on a command line.
3. The installer will check to see if there is an acceptable JVM available on your system to use during the install process. If not, it will install the IBM Developer kit for Windows, Java 2, 1.3.1.
4. If the installer needs to install the IBM JDK, the following prompt will be displayed. Select the language to install and click **OK** to continue.

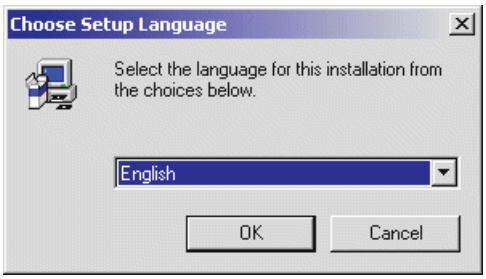


Figure A-5 Select install language

5. Next, follow the screen prompts. Accept the license agreement and the default values to install the IBM Developer Kit For Windows, Java 2 Technology Edition Version 1.3.1. If you are prompted with a message asking `Install this Java Runtime Environment as the system JVM?`, click **Yes**.

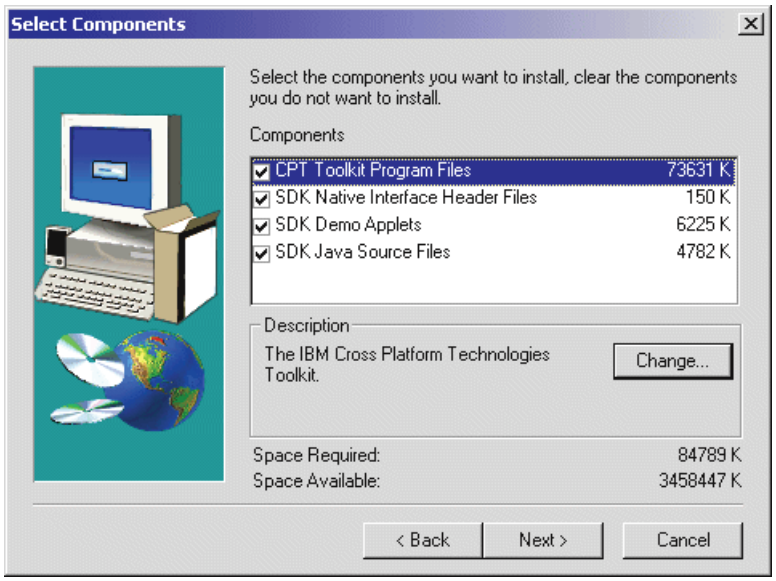


Figure A-6 Select Java components

6. Click **Finish**.
7. After the JDK has installed, If the following window does not appear after about 60 seconds, go back and run the install.bat file again.

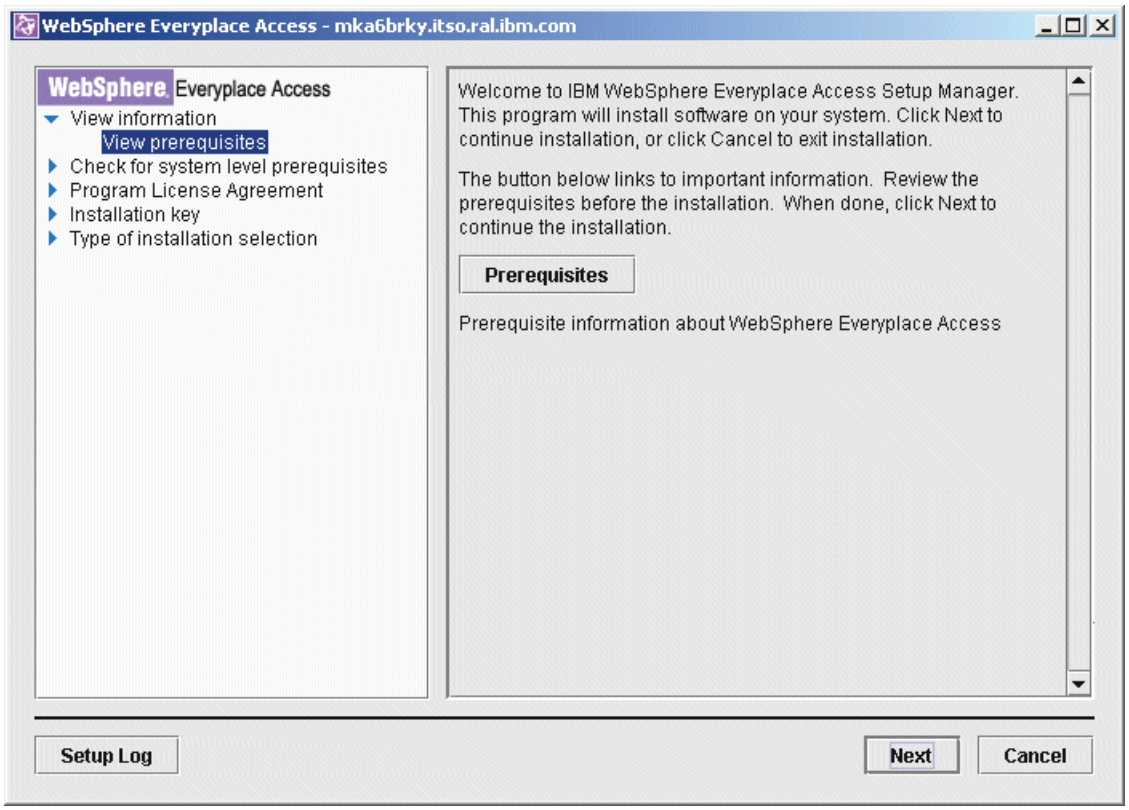


Figure A-7 Prerequisites

8. Continue by viewing the prerequisites and accept the software license.
9. Select **Standard Install**, which gives you more flexibility in the components you can choose to install.

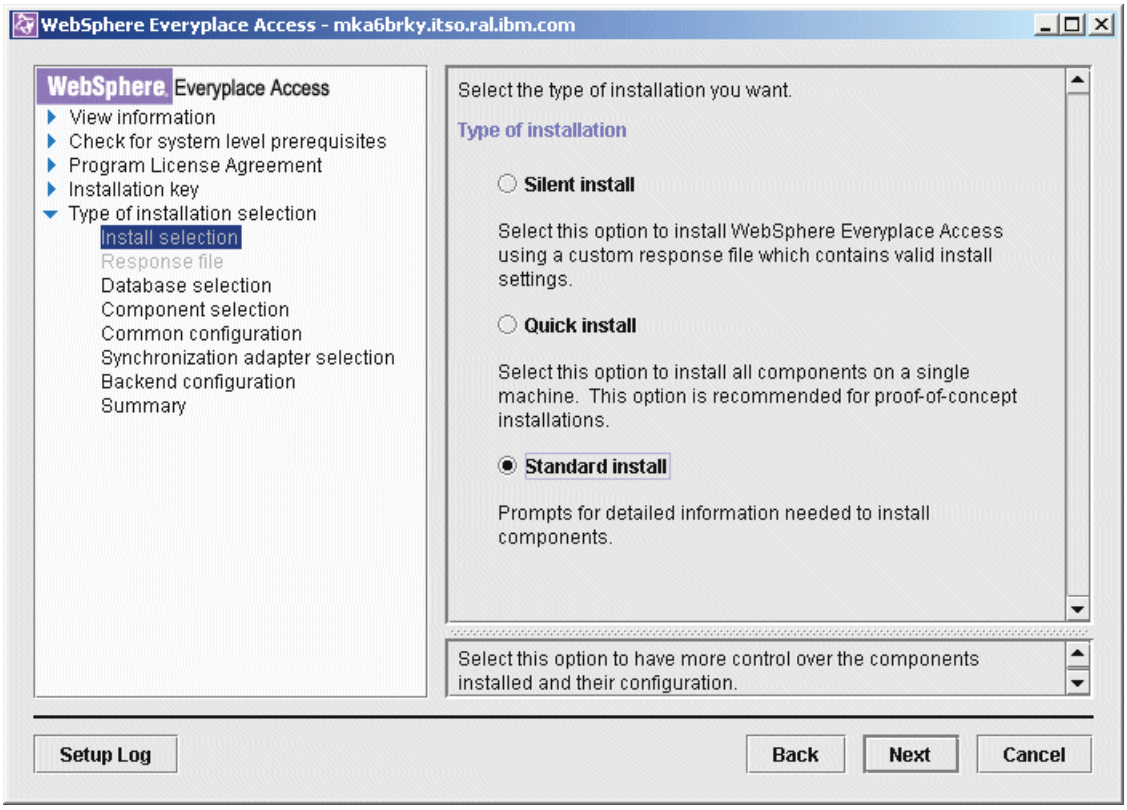


Figure A-8 Standard Install

10. On the next window, you are prompted for a response file. The Setup Manager provides the ability to store a configuration created during an installation in a file called a response file. Since this is the first time we are installing WebSphere Everyplace Access, we don't have a response file. Just leave the response file entry blank and click **Next**.
11. Next, you need to select the components you want to install. Determine the components you need to be installed.

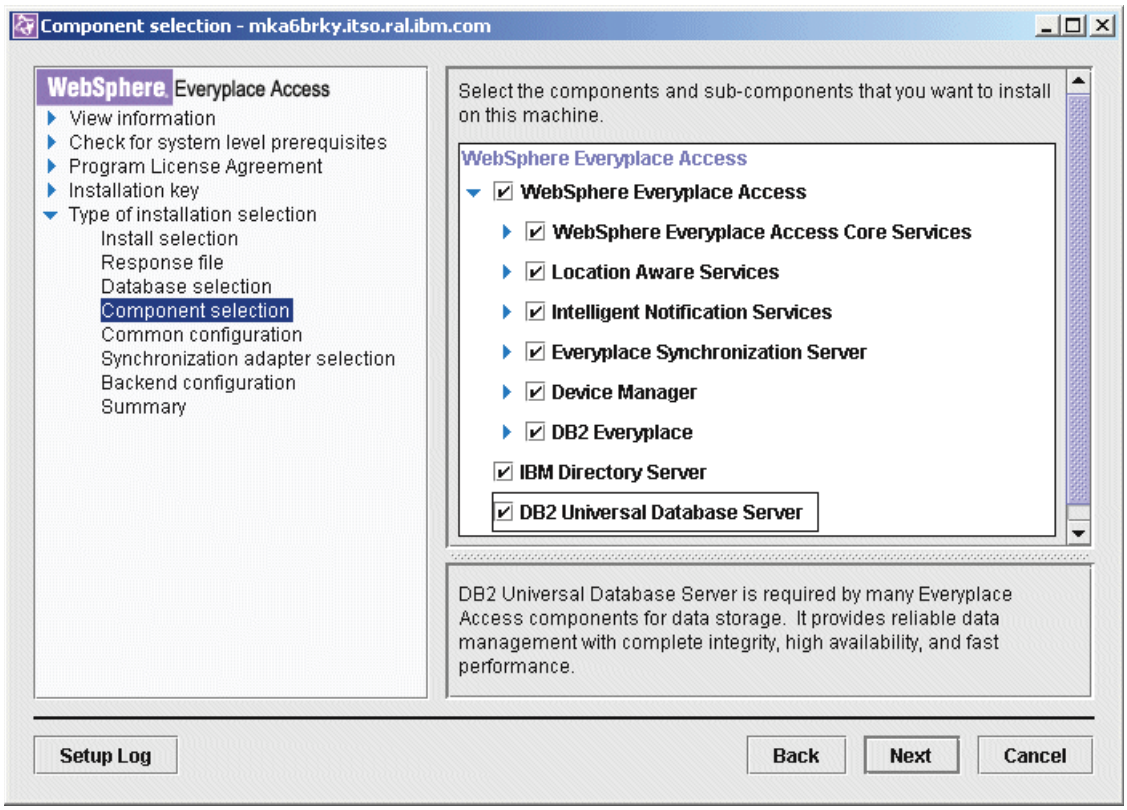


Figure A-9 Select WebSphere Everyplace Access components

Note: If you select a product that is already installed on the machine, that part of the install will not be performed. Also, if you don't select a product that is a prerequisite to installing another selected component, the prerequisite component will automatically be installed.

Configuring the IBM HTTP Server

IBM HTTP Server is an IBM enhanced version of the Apache Web server supports both the Secure Sockets Layer (SSL) Version 2 and 3 protocols for secure connections.

1. The first component that requires some setup is the HTTP Server. Accept the default installation directory (c:\Program Files\IBM HTTP Server) for the HTTP.

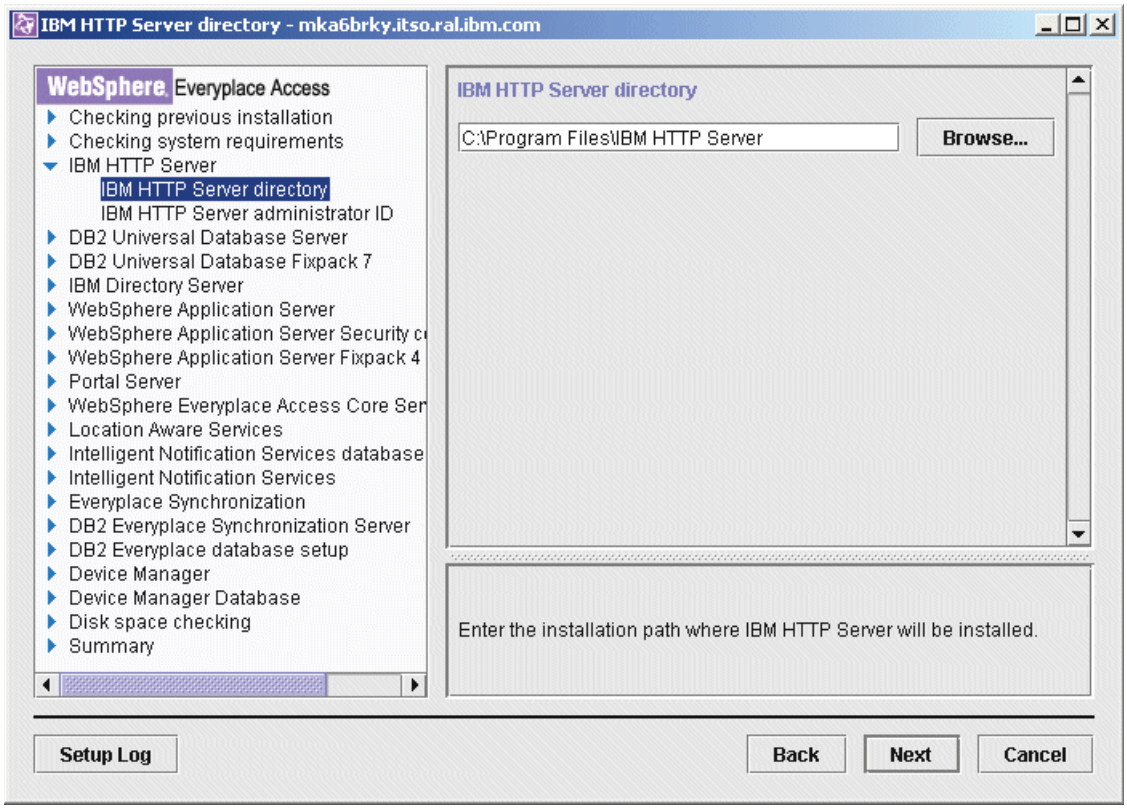


Figure A-10 IBM HTTP Server

2. Enter a user name and password of `httpadmin`.

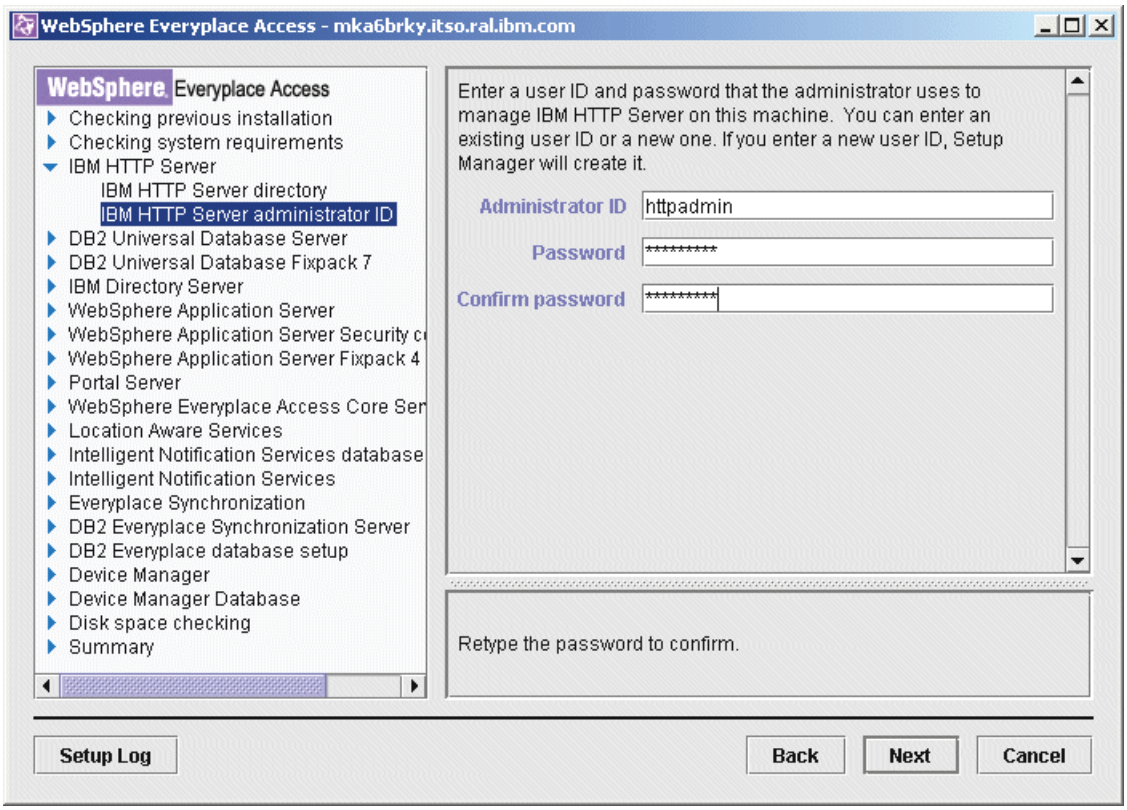


Figure A-11 HTTP admin

Configuring the DB2 Universal Database server

DB2 Universal Database is a Web-enabled relational database management system, used by WebSphere Portal to store Portal-specific data.

1. Accept the default installation directory for DB2 (c:\Program Files\SQLLIB).

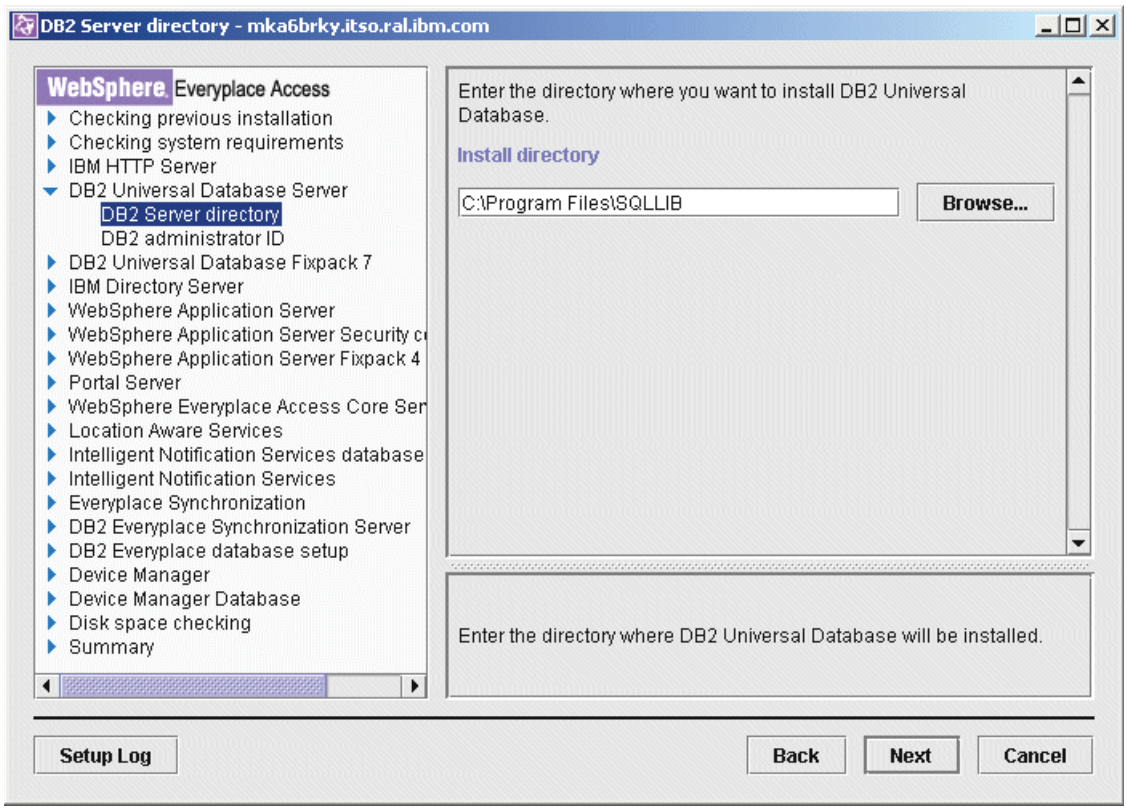


Figure A-12 DB2 directory

2. Enter the user name and password of `db2admin`.

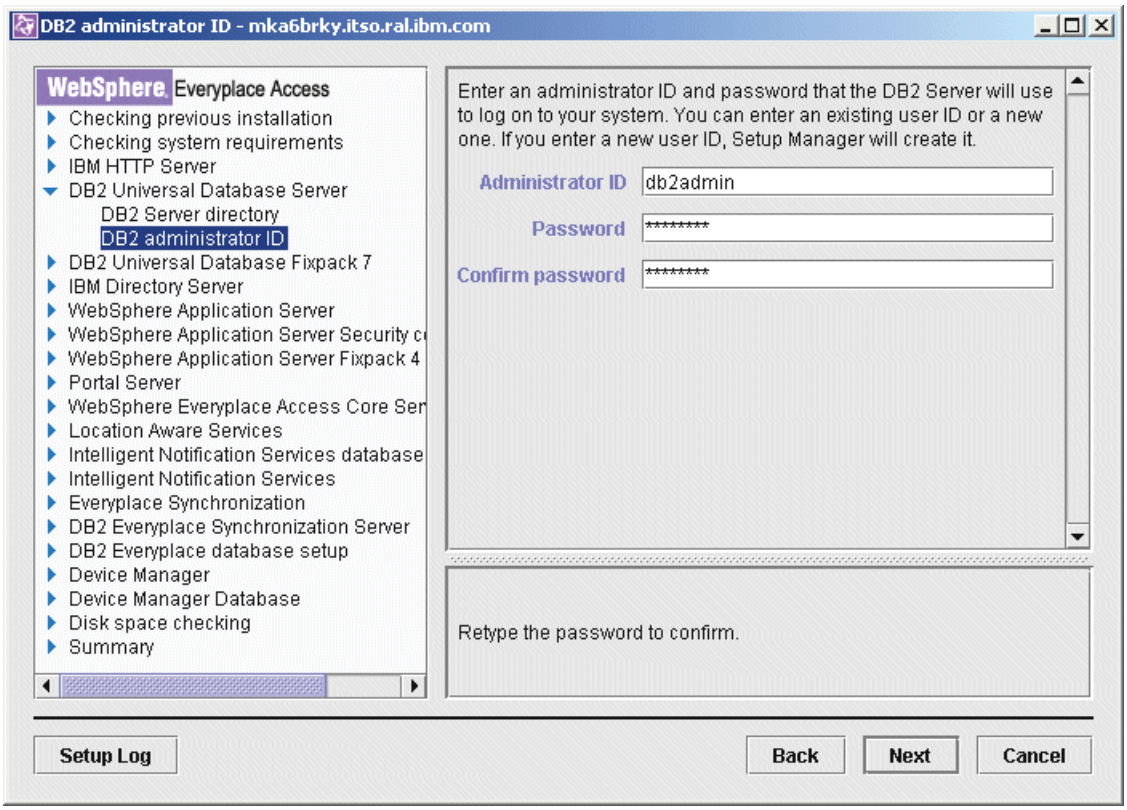


Figure A-13 DB2 admin

Configuring the SecureWay Directory Server

You will notice that you have skipped the DB Universal Database Fix Pack 7 section, since there is no user configuration required at this stage during a standard install.

Next, we will install the IBM Directory Server. This is a Lightweight Directory Access Protocol (LDAP) directory that runs as a stand-alone daemon.

IBM Directory Server provides an easy way to maintain directory information in a central location for storage, updating, retrieval, and exchange of information. WebSphere Portal uses LDAP to store user-specific information.

1. Accept the default installation directory (c:\Program Files\IBM\LDAP).
2. Enter the following information to configure IBM Directory Server:

   Admin ID `cn=root`
   Admin PW `root` (you can change this)

Suffix `dc=yourdomain` (for example, `dc=itso,dc=ral,dc=ibm,dc=com`)
TCP/IP Port `389`

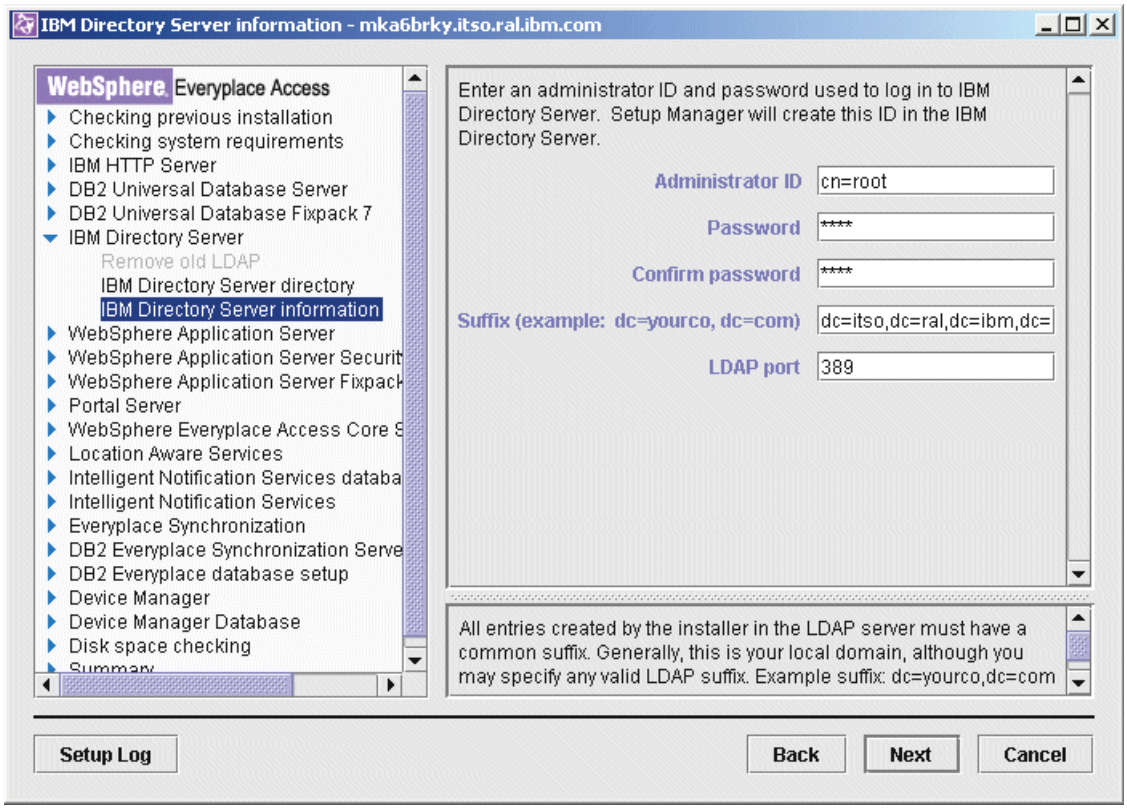


Figure A-14 LDAP information

WebSphere Application Server

Enables Web transactions and interactions with a robust deployment environment for e-business applications. It provides a portable, Java-based Web application deployment platform focused on supporting and executing servlets, JavaBeans, JavaServer Pages (JSP) files, and enterprise beans.

1. Accept the default installation directory (c:\WebSphere\AppServer).

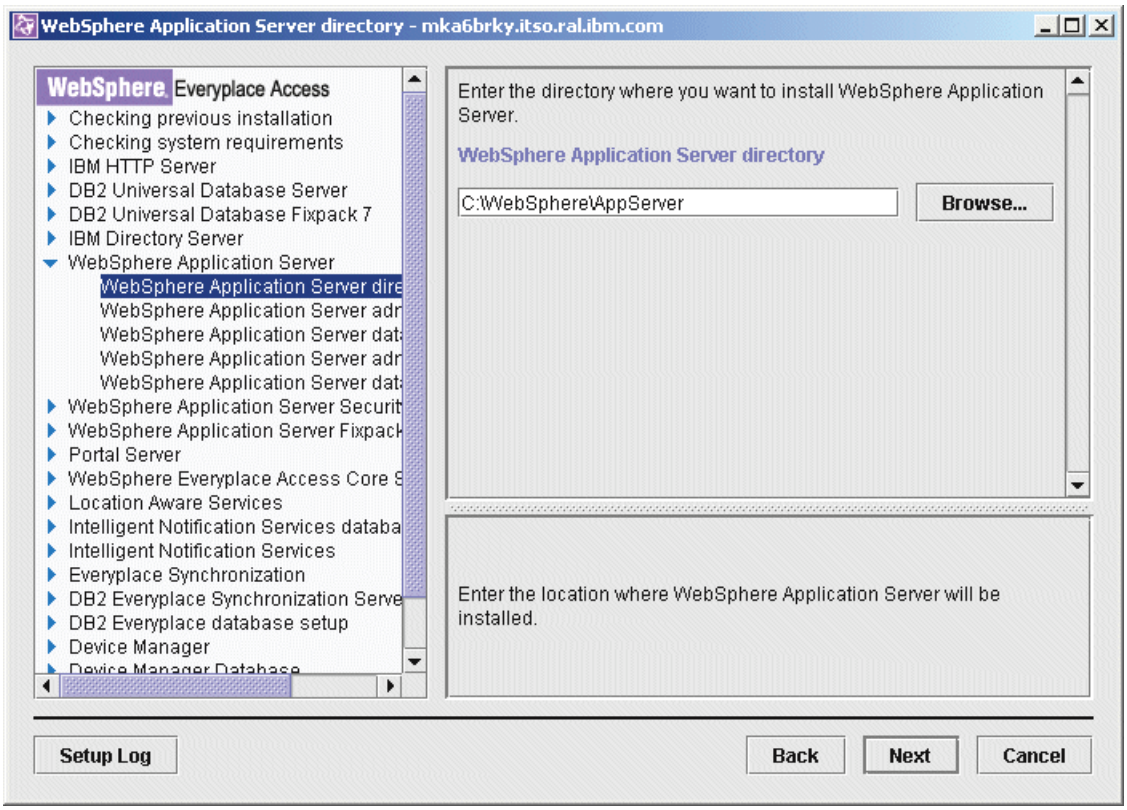


Figure A-15 WebSphere Application Server Directory

2. Enter WebSphere Application Server administrator ID and password of `wasadmin`.

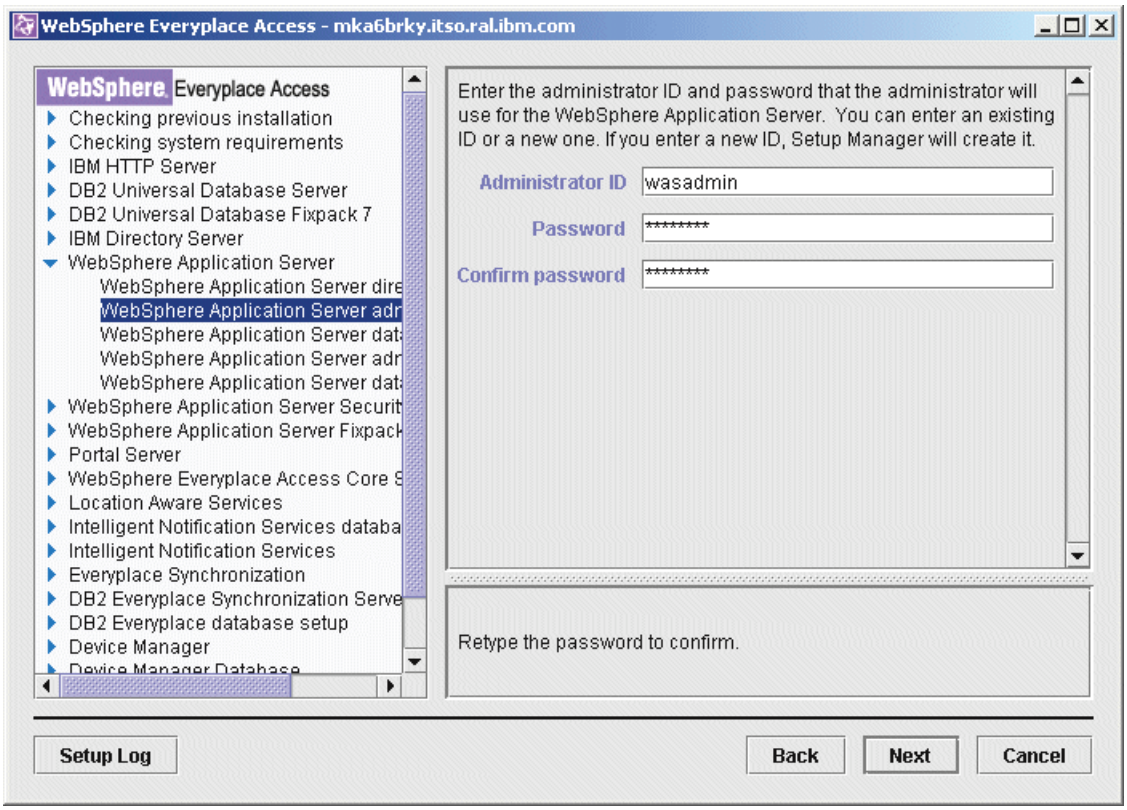


Figure A-16 WebSphere Application Server administration

3. Because we are installing WebSphere Everyplace Access on a single system for this exercise, leave the default No and click **Next**.

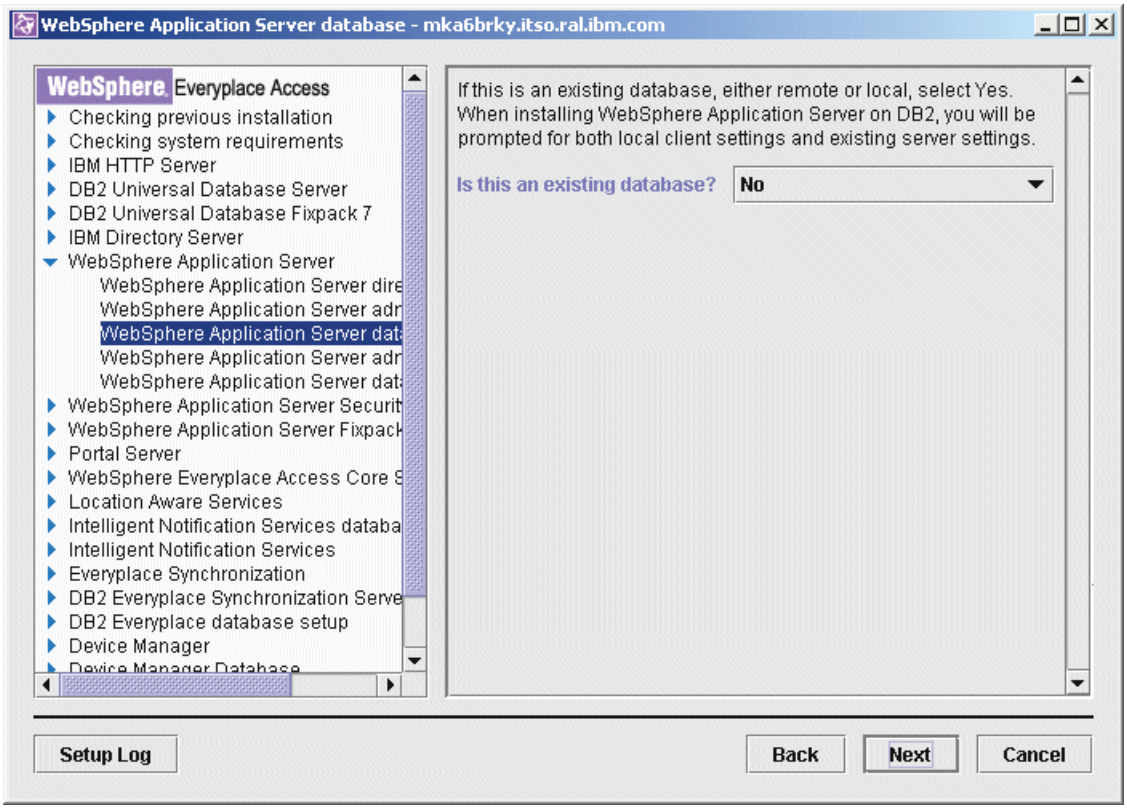


Figure A-17 WebSphere Application Server no existing database

4. Enter `db2admin` as the user ID and password.

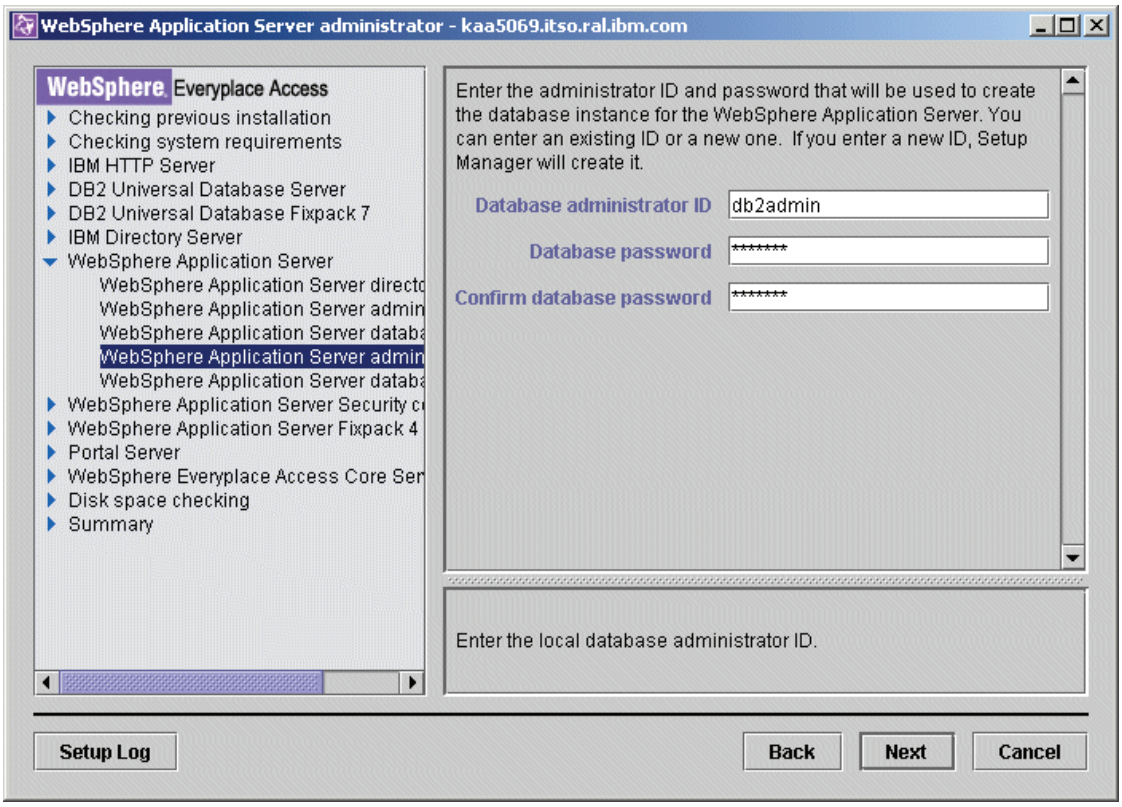


Figure A-18 WebSphere Application Server DB2 admin

5. Verify the database setttings are:

Local Database Name	`wasdb1`
Local Database Alias Name	`wasdb`
Node Name	`LOOPBACK`
Database Server Port	`55555`

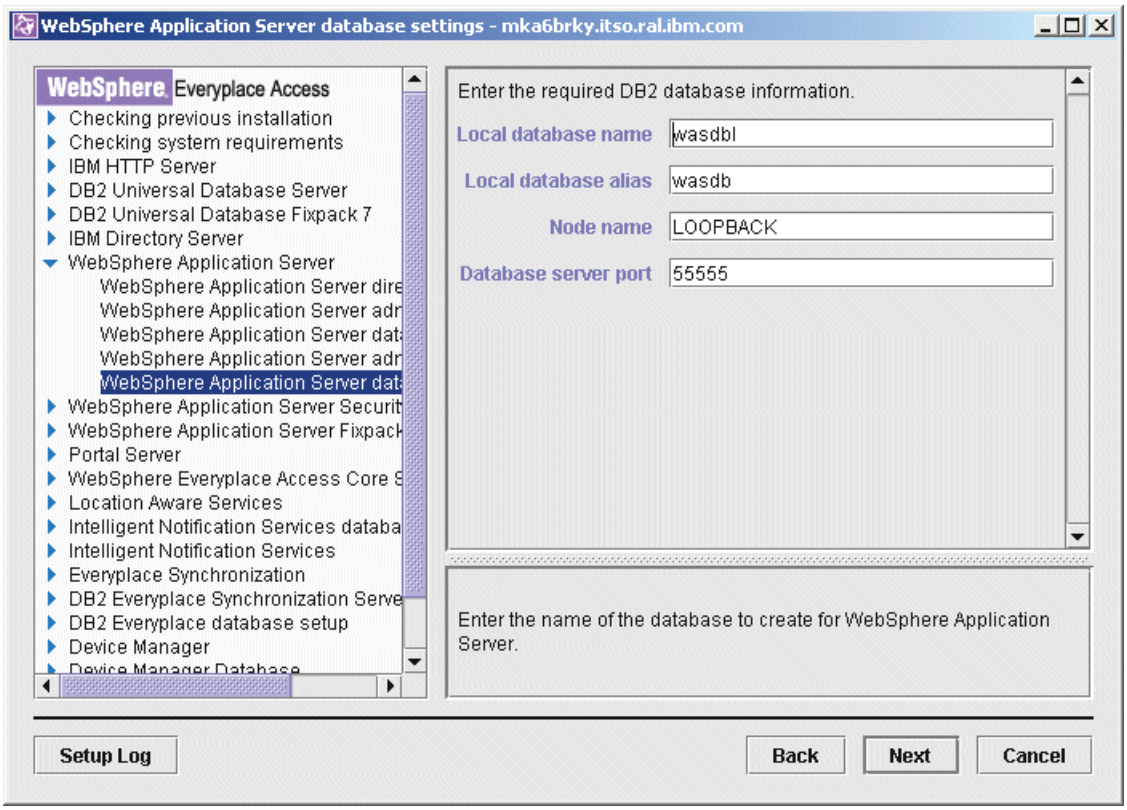


Figure A-19 WebSphere Application Server database settings

6. Next you will be asked whether WebSphere Application Server security has been enabled. Security has not yet been enabled. Make sure that **No** is checked and click **Next**.

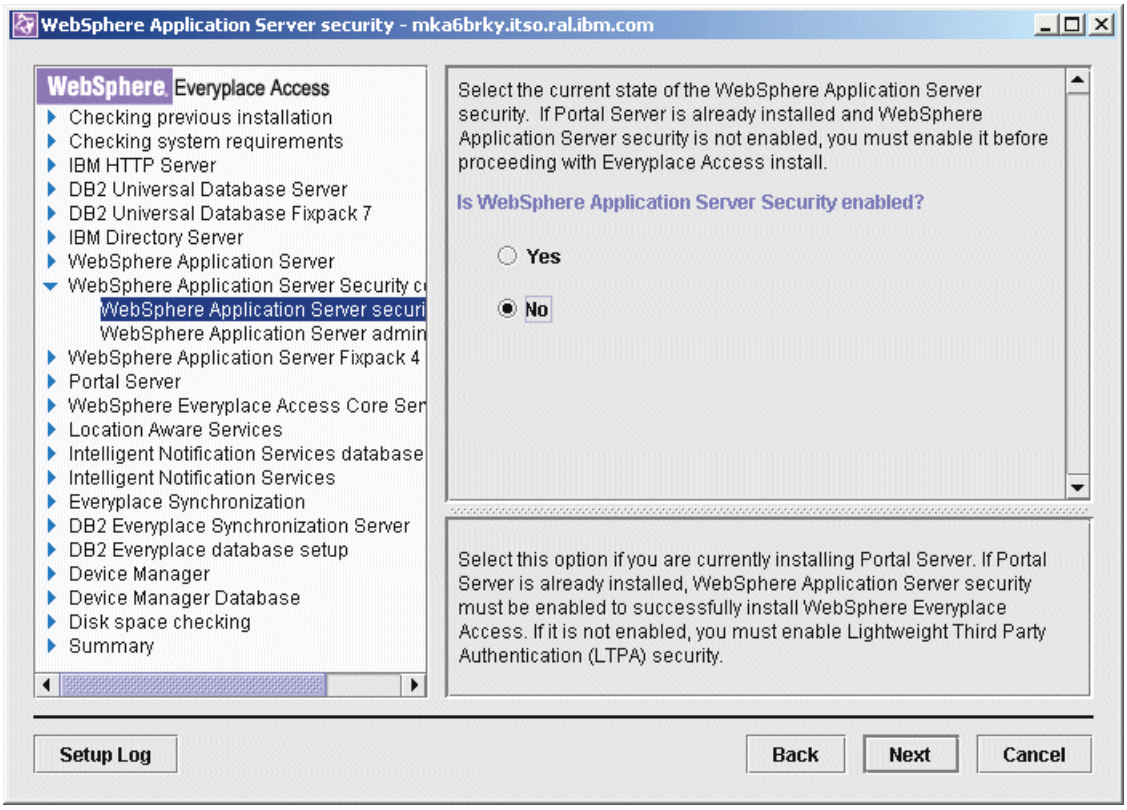


Figure A-20 WebSphere Application Server Security not enabled

Portal Server

WebSphere Portal allows companies to build their own custom Portal Web site.

1. Enter an LTPA password of `root`.

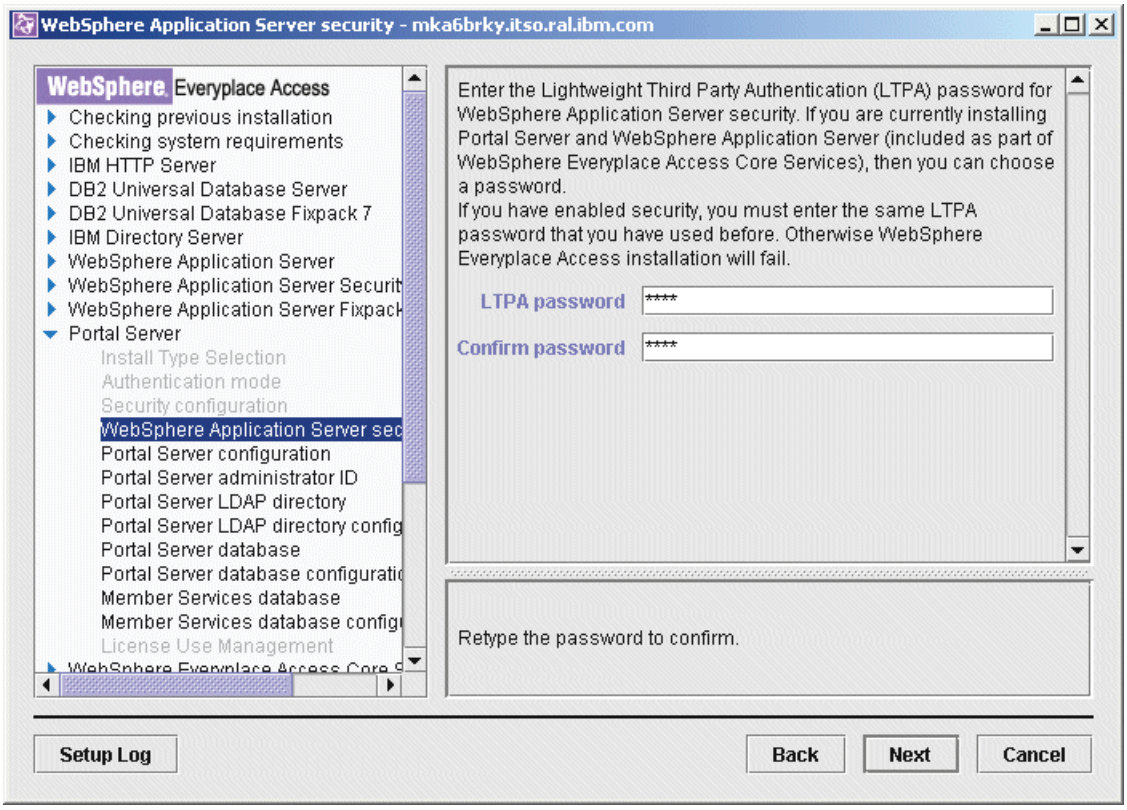


Figure A-21 WebSphere Application Server LTPA

2. Verify the following information to configure the Portal Server.

Install Directory	`C:\WebSphere\PortalServer`
Host name	*`yourmachinename`*`.yourdomain`
Base URI	`/wps`
Home page	/portal
Customized page	`/myportal`
Proxy host	(not used)
Proxy port	(not used)

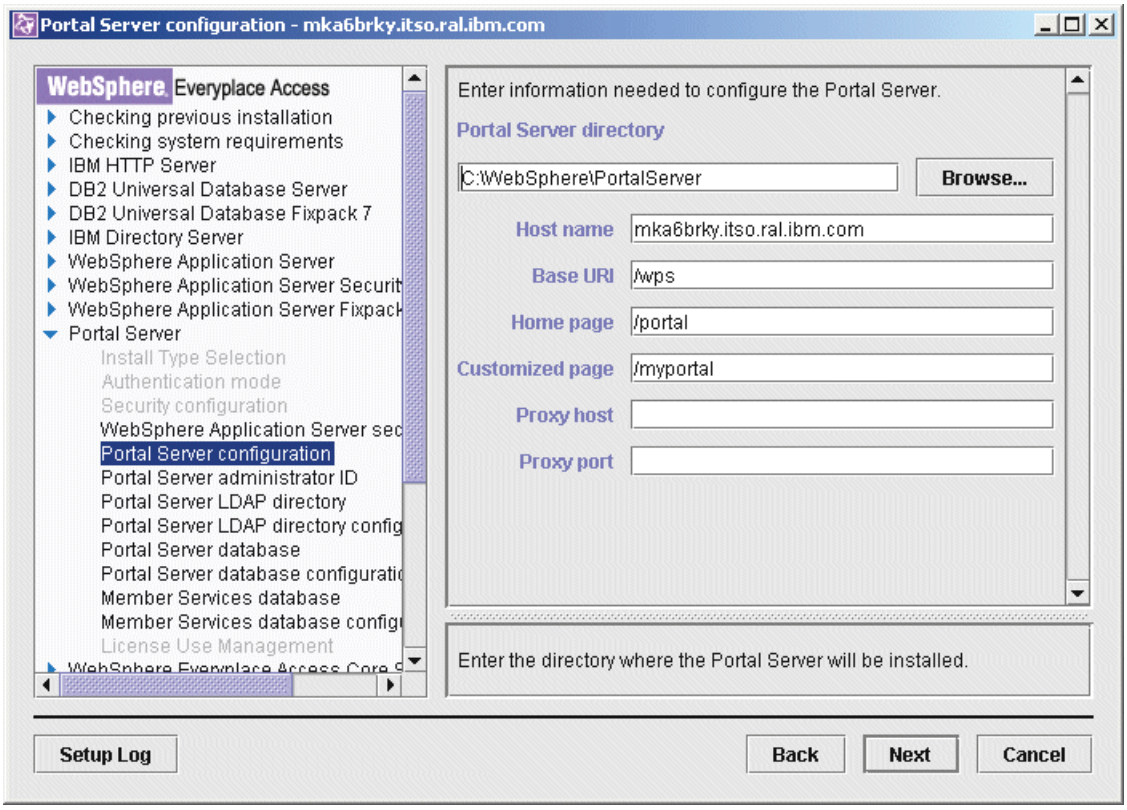


Figure A-22 WebSphere Portal Server configuration

3. Enter the WebSphere Portal Server administrator ID and password; use `wpsadmin` for both.

Important: If you use a user name and password different from `wpsadmin`, the uninstall program will not work.

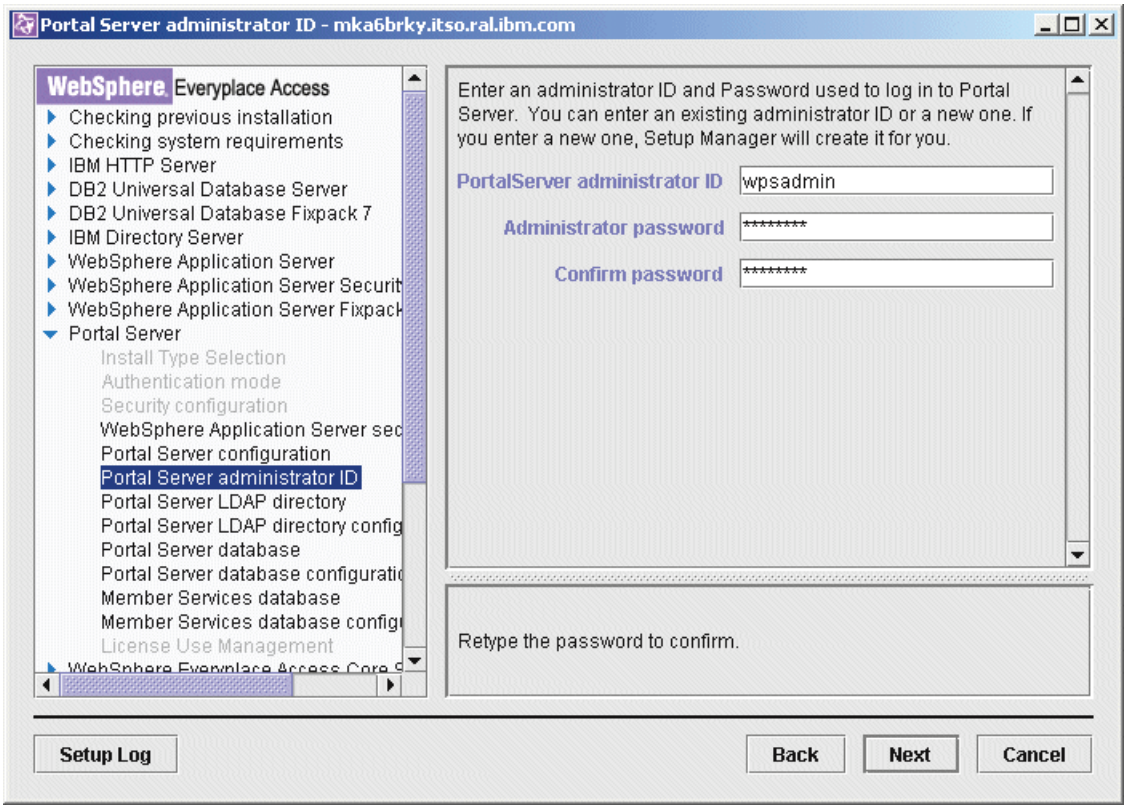


Figure A-23 WebSphere Portal Server admin

4. Configure the LDAP directory to use. You will see there are four supported selections. In our installation we will use the IBM SecureWay Directory as the LDAP directory, with the following entries:

LDAP Server	*yourmachinename*.yourdomain
User DN	cn=root
User Password	root
Suffix	dc=yourdomain (for example, dc=itso,dc=ral,dc=ibm,dc=com)
Customized Page	/myportal
LDAP Port Number	389

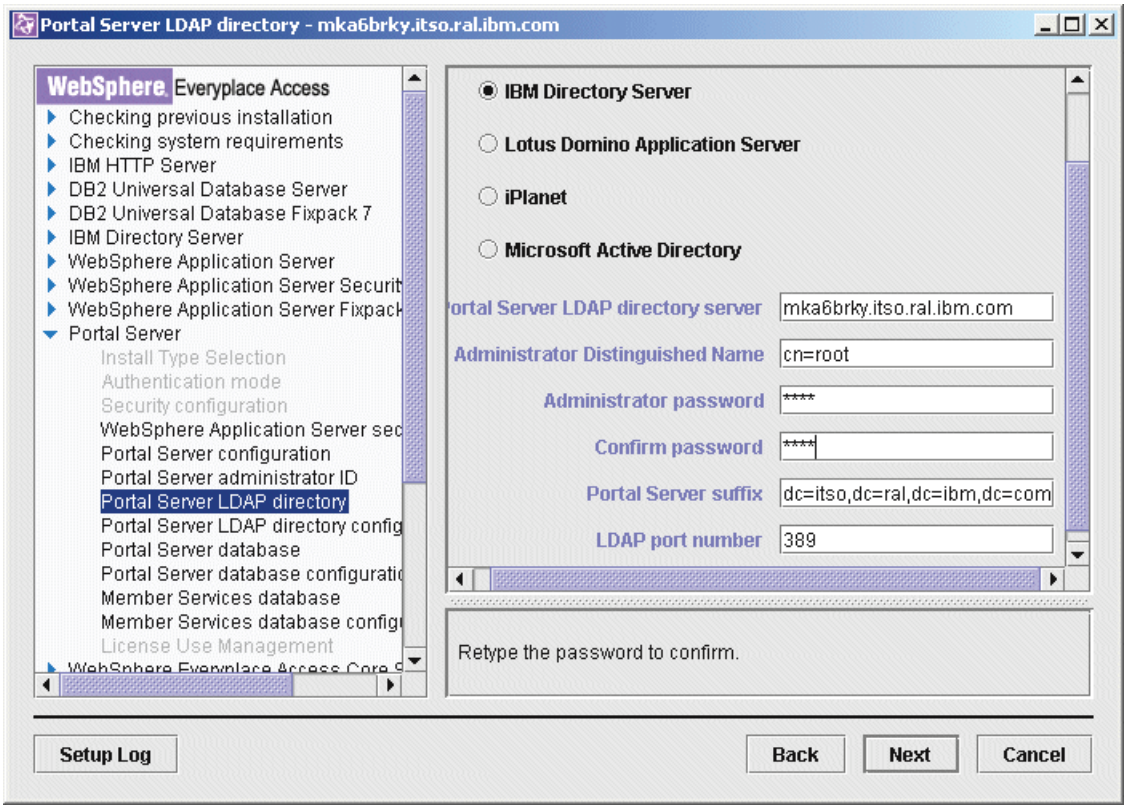


Figure A-24 WebSphere Portal Server LDAP

5. Use the existing parameters for the LDAP server:

User Object Class	`InetOrgPerson`
User DN prefix	`uid`
User DN suffix	`cn=users,dc=yourdomain,dc=com`
Group Object Class	`groupOfUniqueNames`
Group Member	`uniqueMember`
Group DN prefix	`cn`
Group DN suffix	`cn=groups,dc=yourdomain`
Administrative DN	`uid=wpsadmin,cn=users,dc=yourdomain`
Administrative group DN	`cn=wpsadmins,cn=groups,dc=yourdomain`

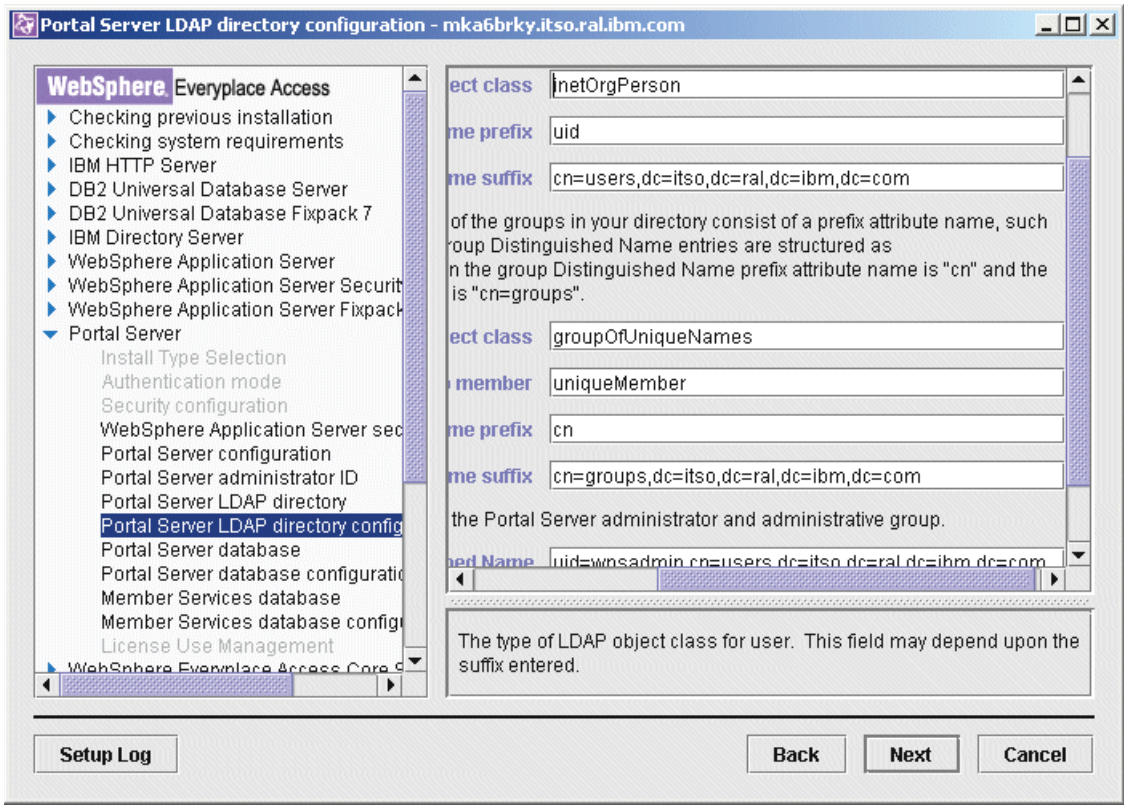


Figure A-25 WebSphere Portal Server LDAP configuration

6. Select **Create and Initialize a New Database** for the Portal Server.

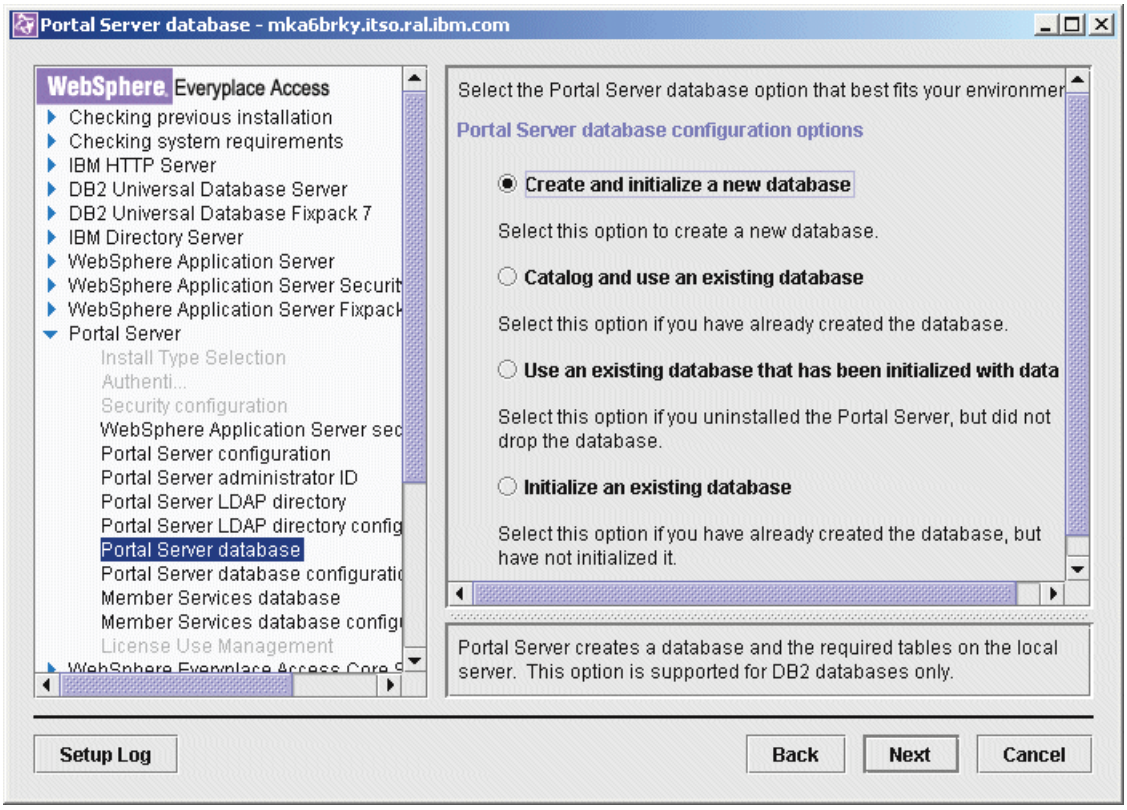


Figure A-26 WebSphere Portal Server database create

7. For the additional database configuration, enter a database user ID and password of `db2admin`.

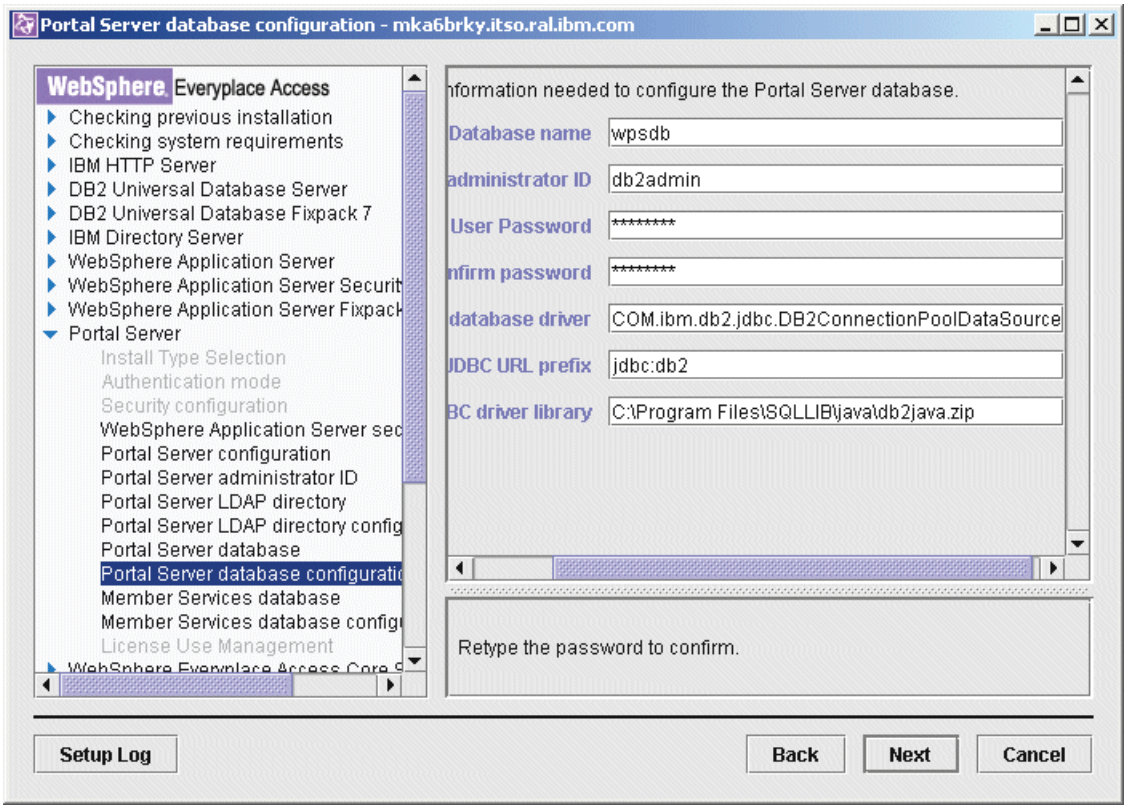


Figure A-27 WebSphere Portal Server database configuration

8. Select **Create and Initialize a New Database** for use with the member services, and select the default option.

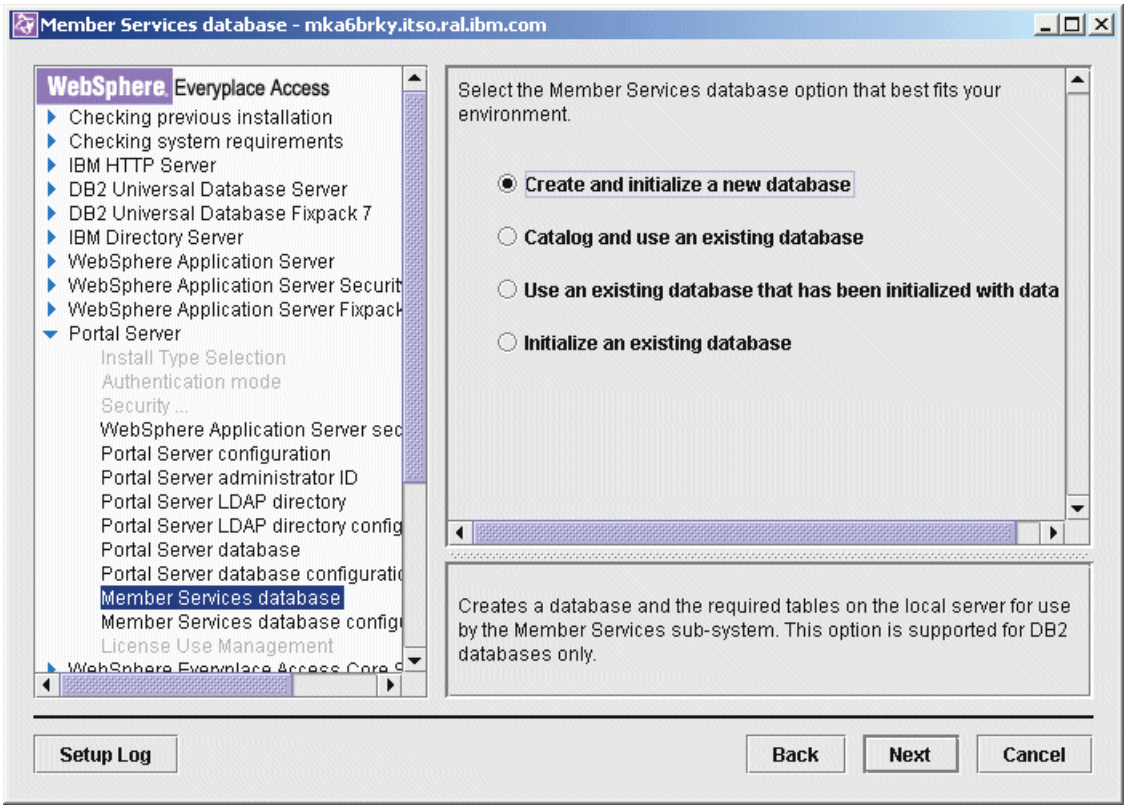


Figure A-28 WebSphere Portal Server WMS database create

9. Configure access to the database with a user ID and password of db2admin.

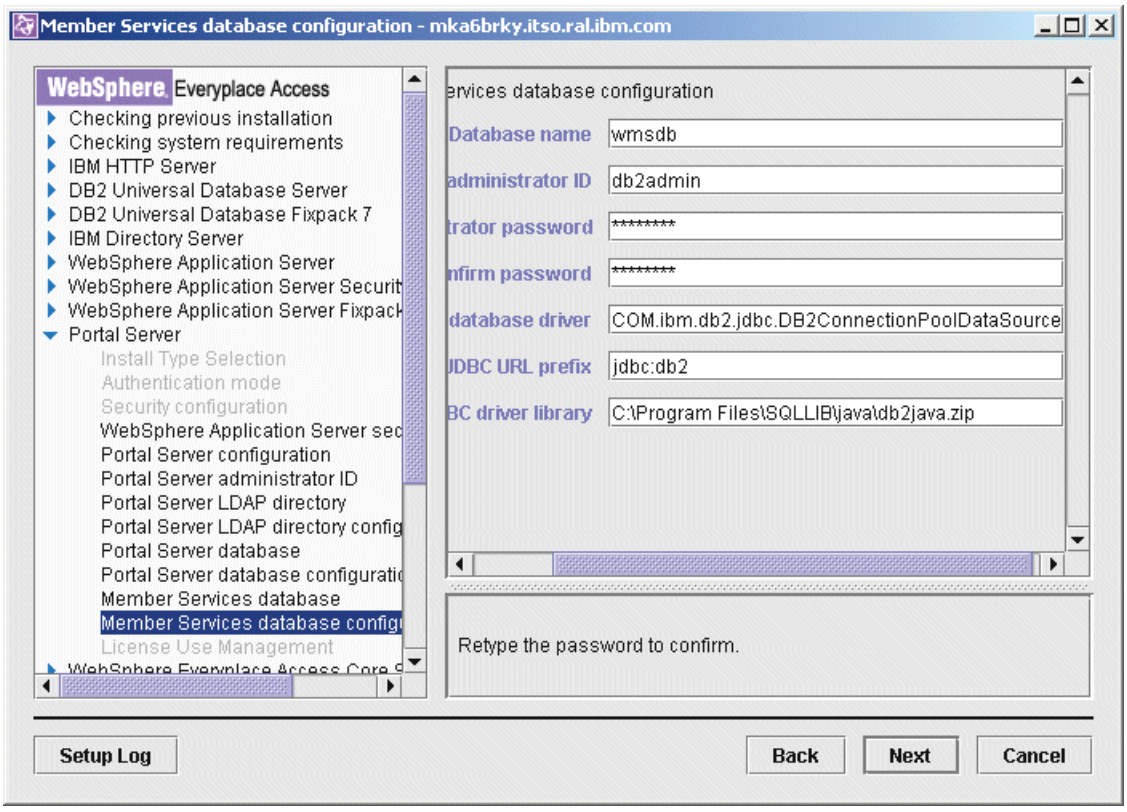


Figure A-29 WebSphere Portal Server member services database configuration

WebSphere Everyplace Access Core Services

1. Resolve the directory where the WebSphere Everyplace Access Core Services is to be installed.

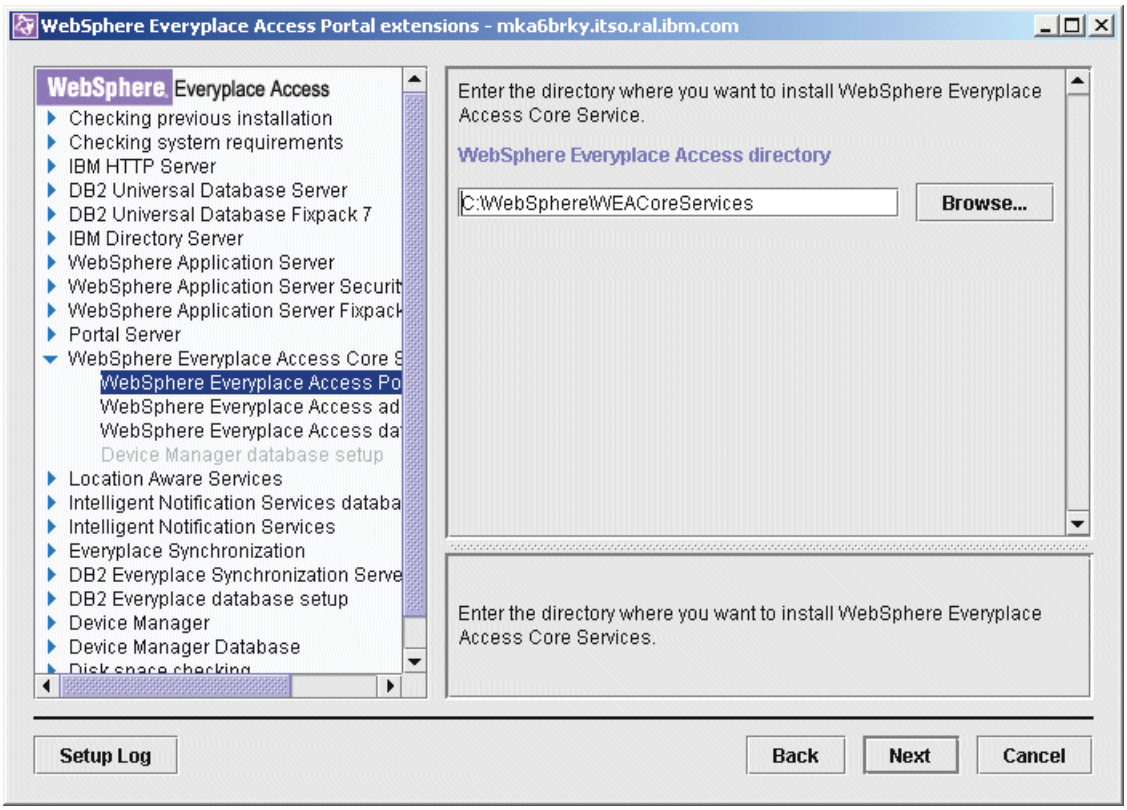


Figure A-30 WebSphere Everyplace Access core services

2. Select to install all WebSphere Everyplace Access portlets.

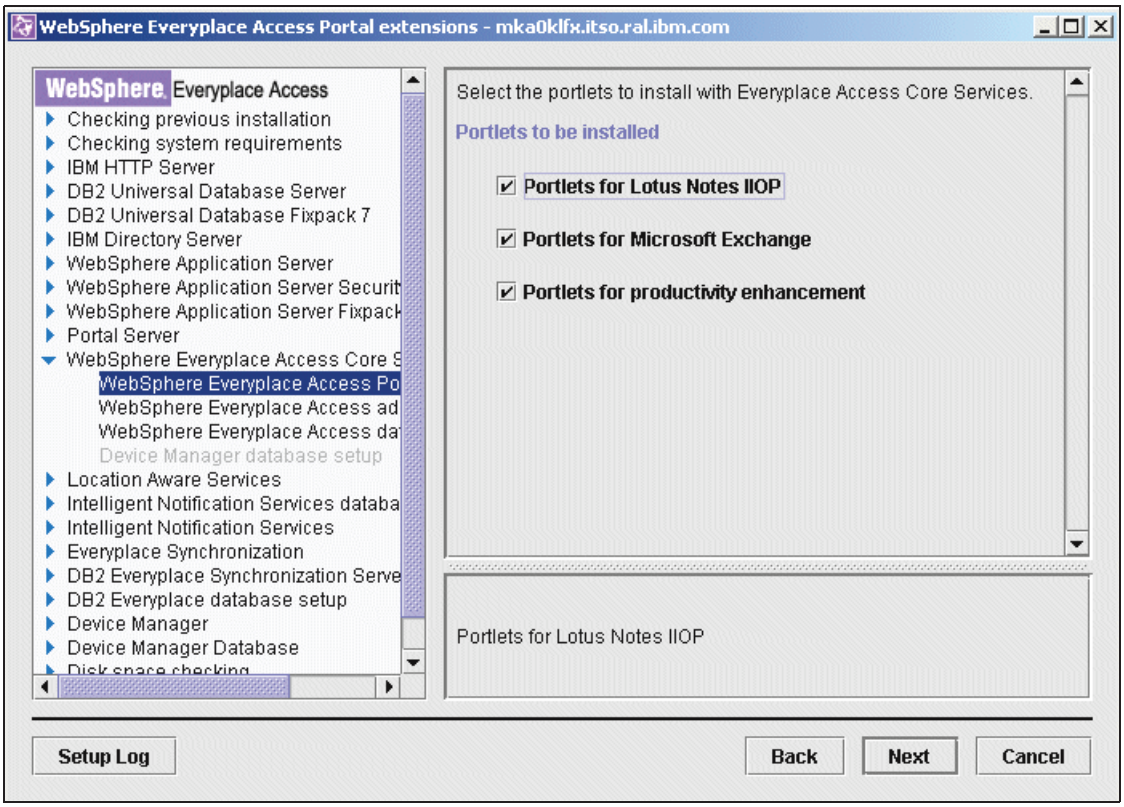


Figure A-31 WebSphere Everyplace Access core extensions

3. Enter admin user IDs and passwords for the portal:

Portal Server host name	***yourmachinename***`.yourdomain`
Portal base URI	`/wps`
Home page	`/portal`
Portal customized page	`/myportal`
Portal administrator ID	`wpsadmin`
Portal Admin Password	`wpsadmin`
Administration Group	`wpsadmins`

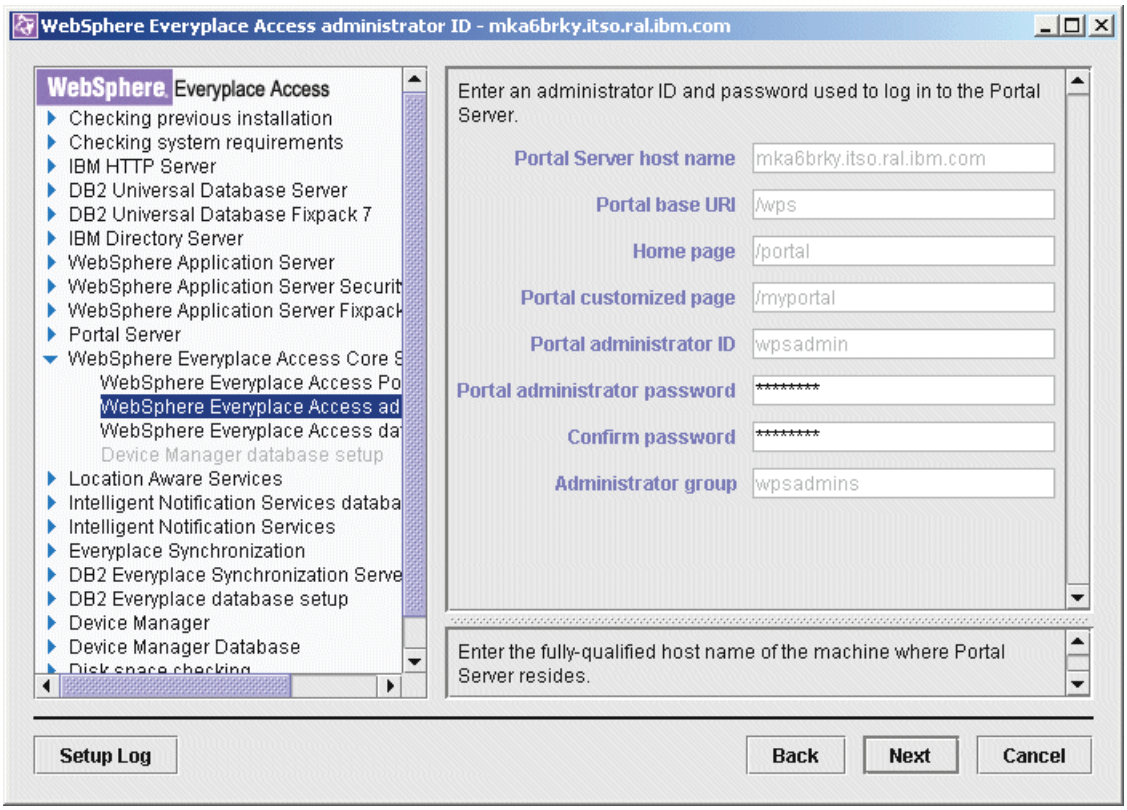


Figure A-32 WebSphere Everyplace Access core admin ID

4. Enter the DB2 admin user ID and password of `db2admin`.

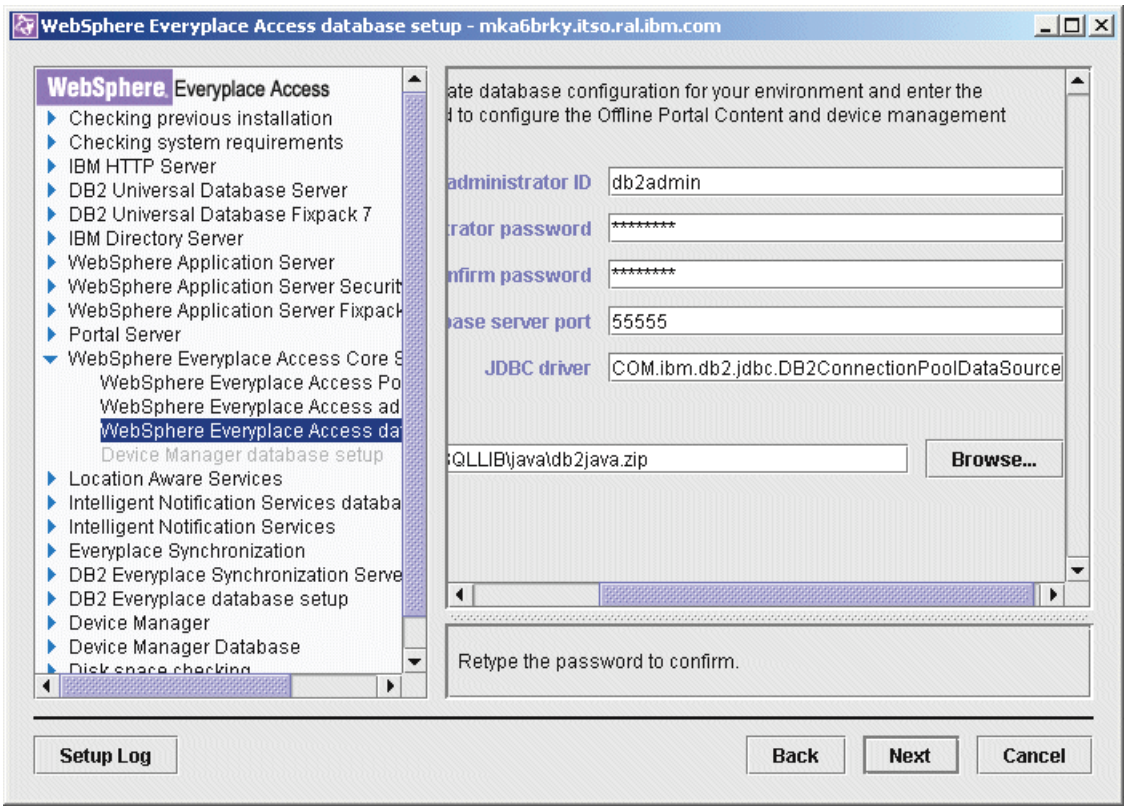


Figure A-33 WebSphere Everyplace Access core database admin

Location Aware Services

1. Choose the install directory for Location Aware Services and leave the default install directory and click **Next**.

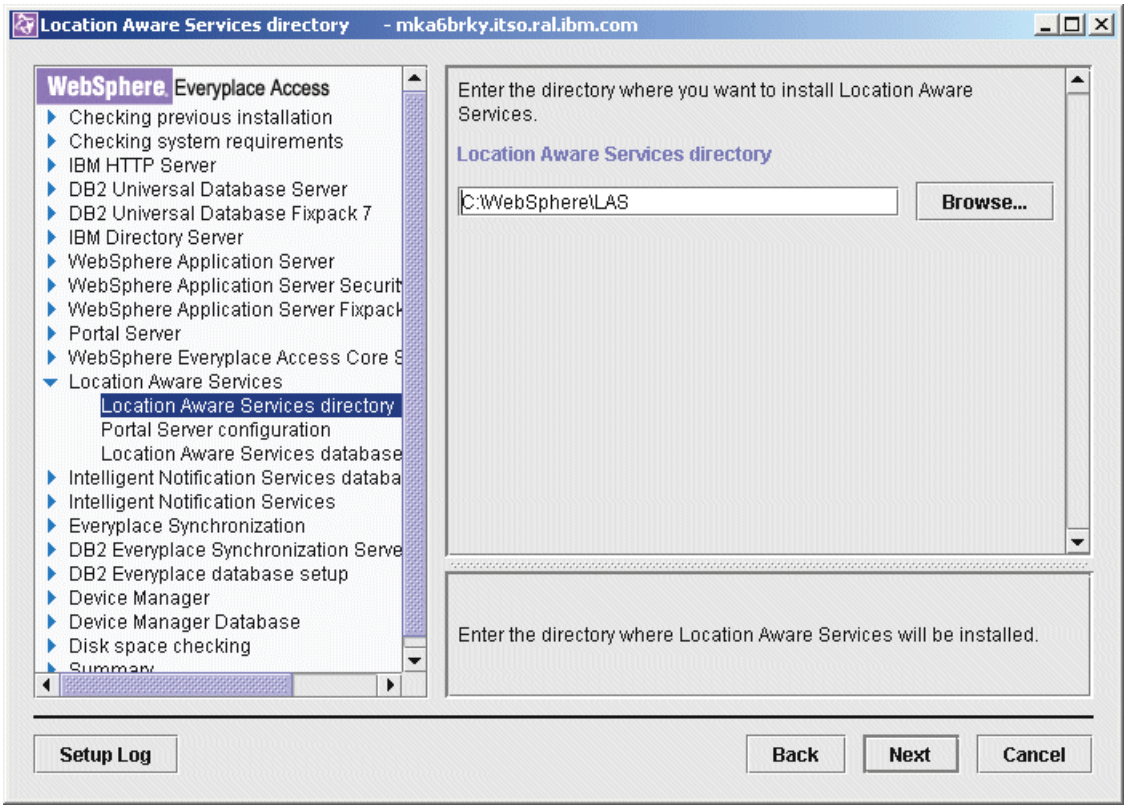


Figure A-34 Location Aware Services directory

2. Enter the information for the Portal Server (use the existing information).

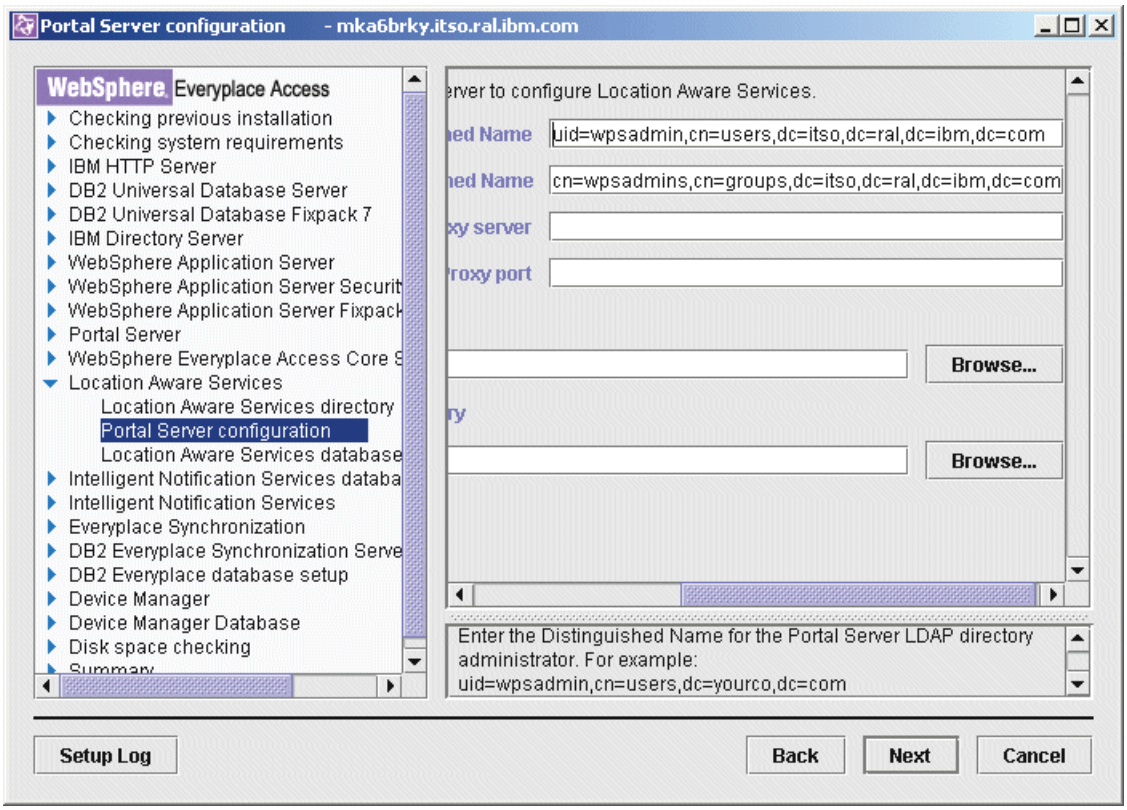


Figure A-35 Location Aware Services WebSphere Portal Server configuration

3. For the location of the Location Aware Services database, use the default of c:\Program Files\SQLLIB and set the database administrator ID and password as `db2admin`.

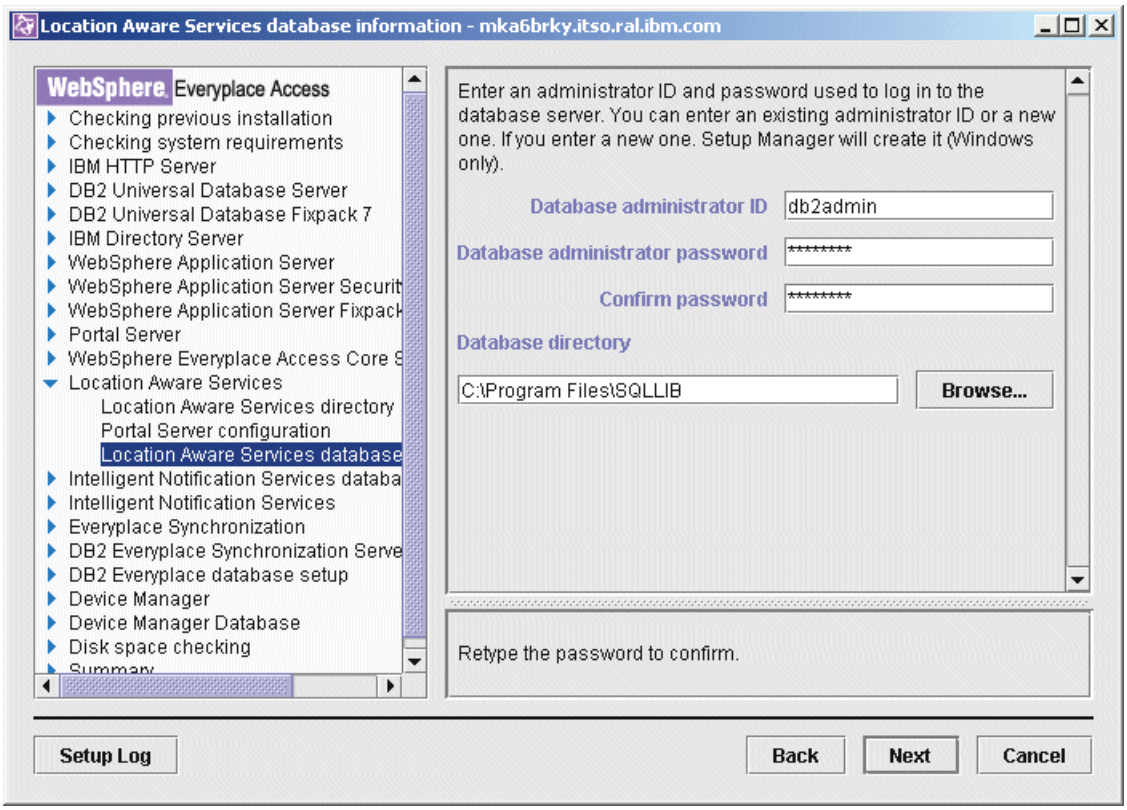


Figure A-36 Location Aware Services database admin

Intelligent Notification

1. Choose a directory for INS to install. Use the default c:\WebSphere\INS.

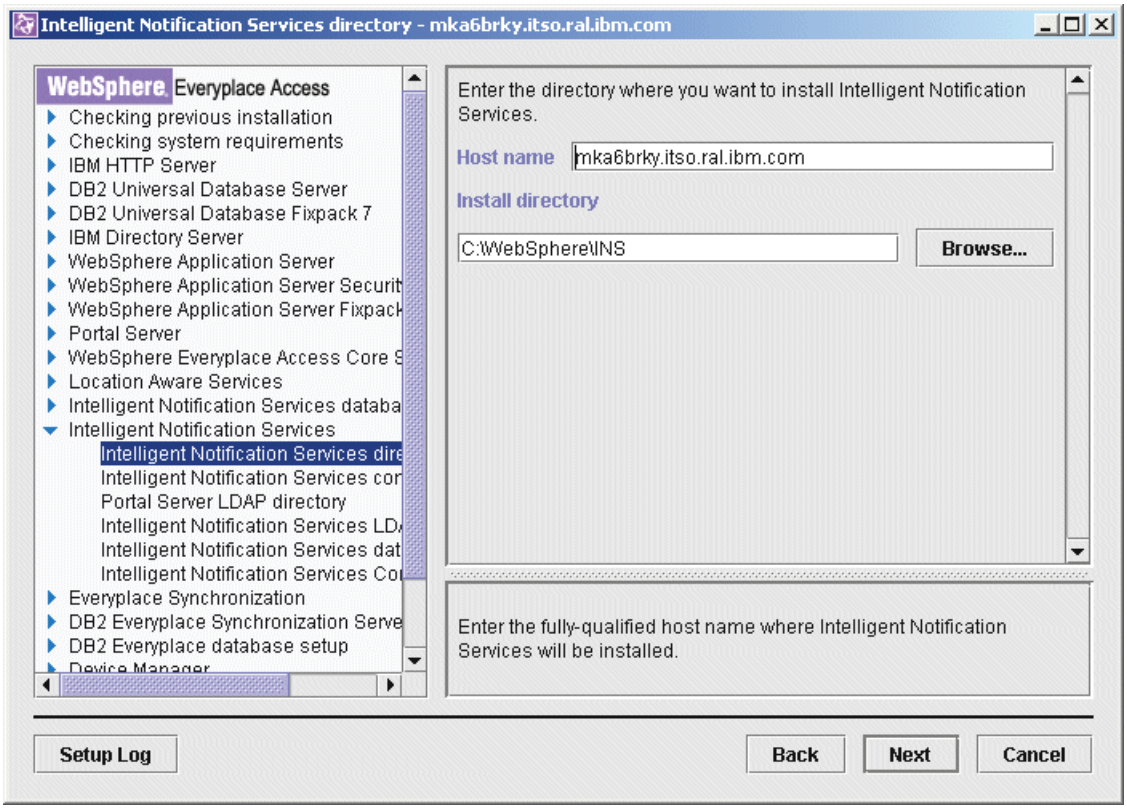


Figure A-37 INS directory

2. Choose a directory for the configuration parameters to be stored. Use the default and click **Next**.

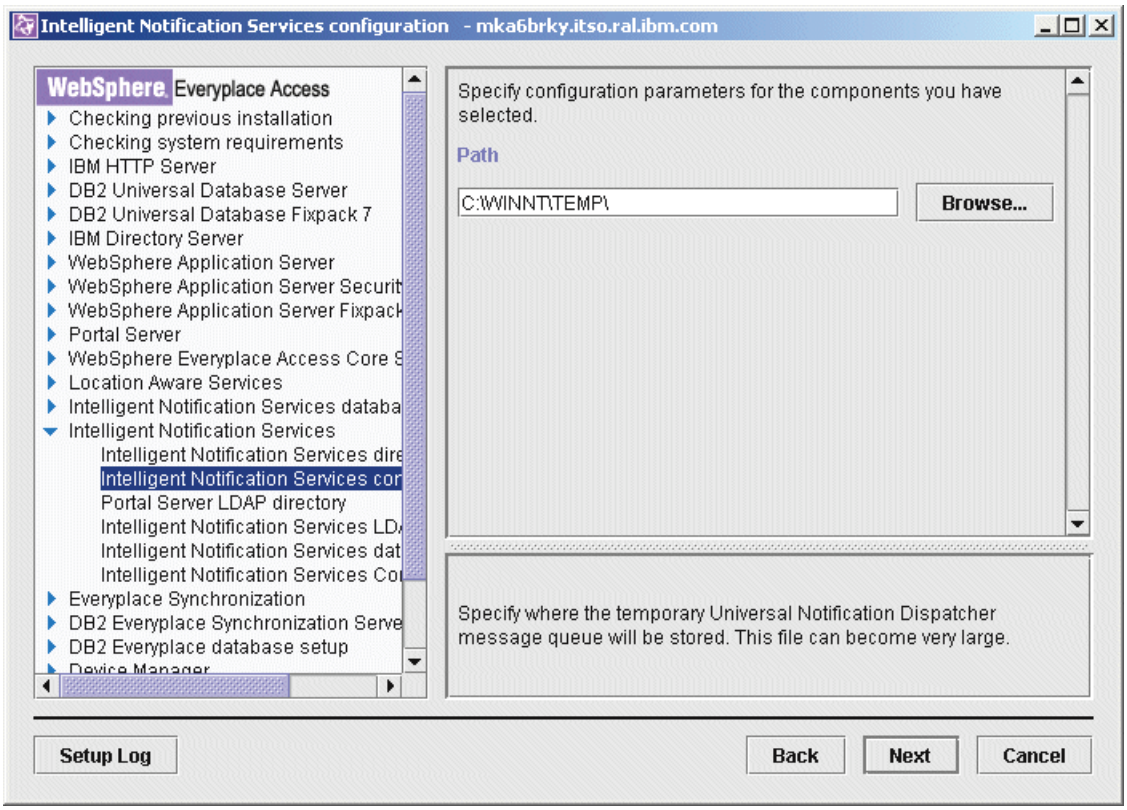


Figure A-38 INS config

3. Set up the database for INS to use. Use the default database name and for the database administrator use `db2admin` for both the user ID and password.

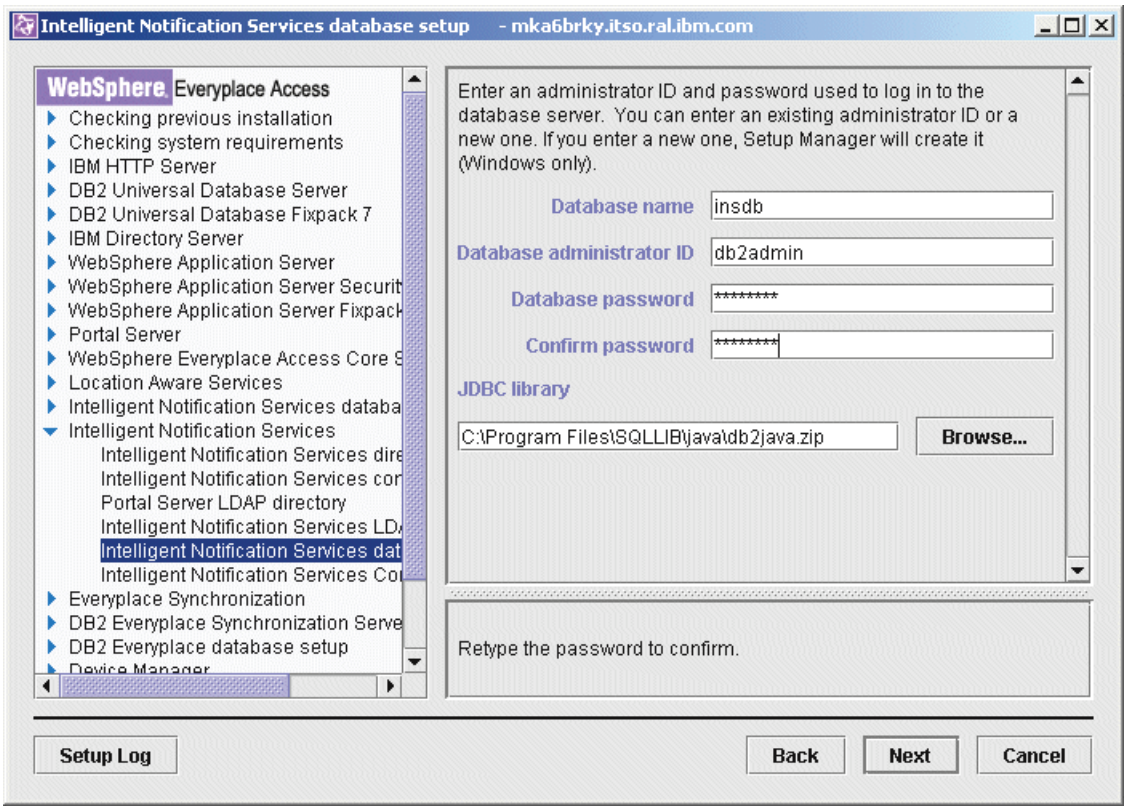


Figure A-39 INS database admin

Everyplace Synchronization Server (ESS) adapters

1. To configure ESS, enter the ESS administrator ID and password of `essadmin`.

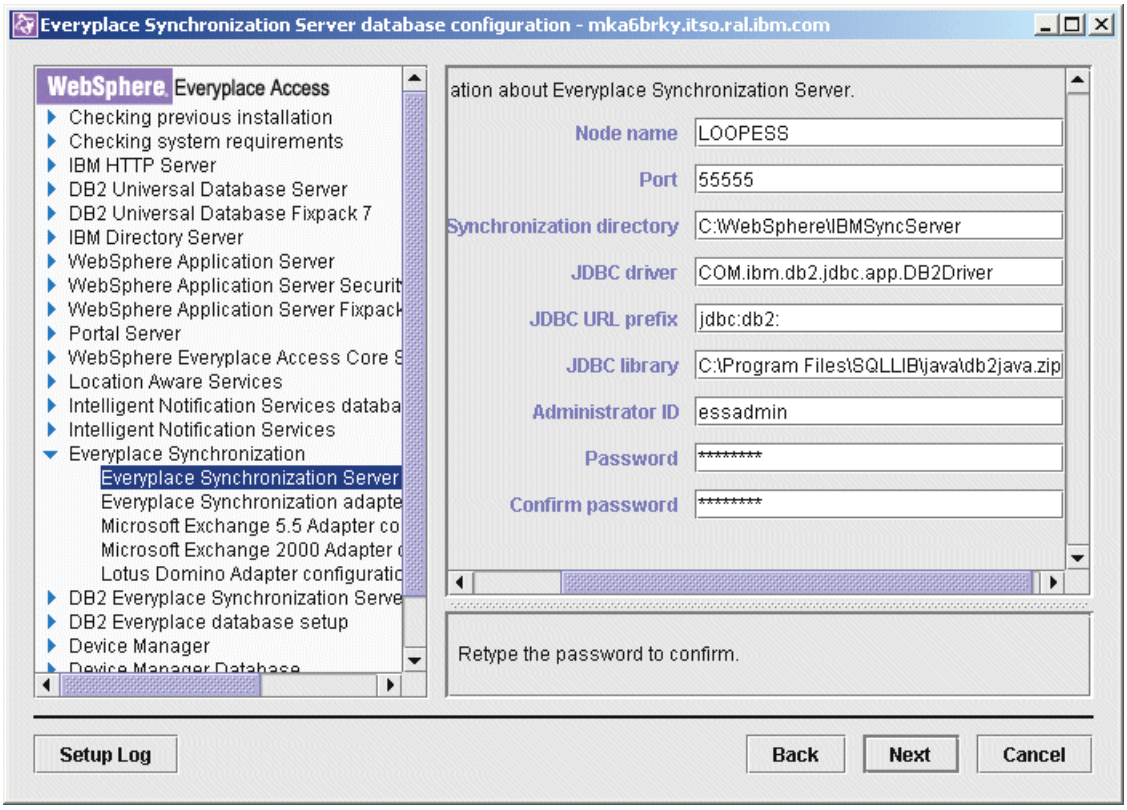


Figure A-40 Configuring Everyplace Synchronization Server

2. Select the adapters you need. Note that you cannot add any adapter later.

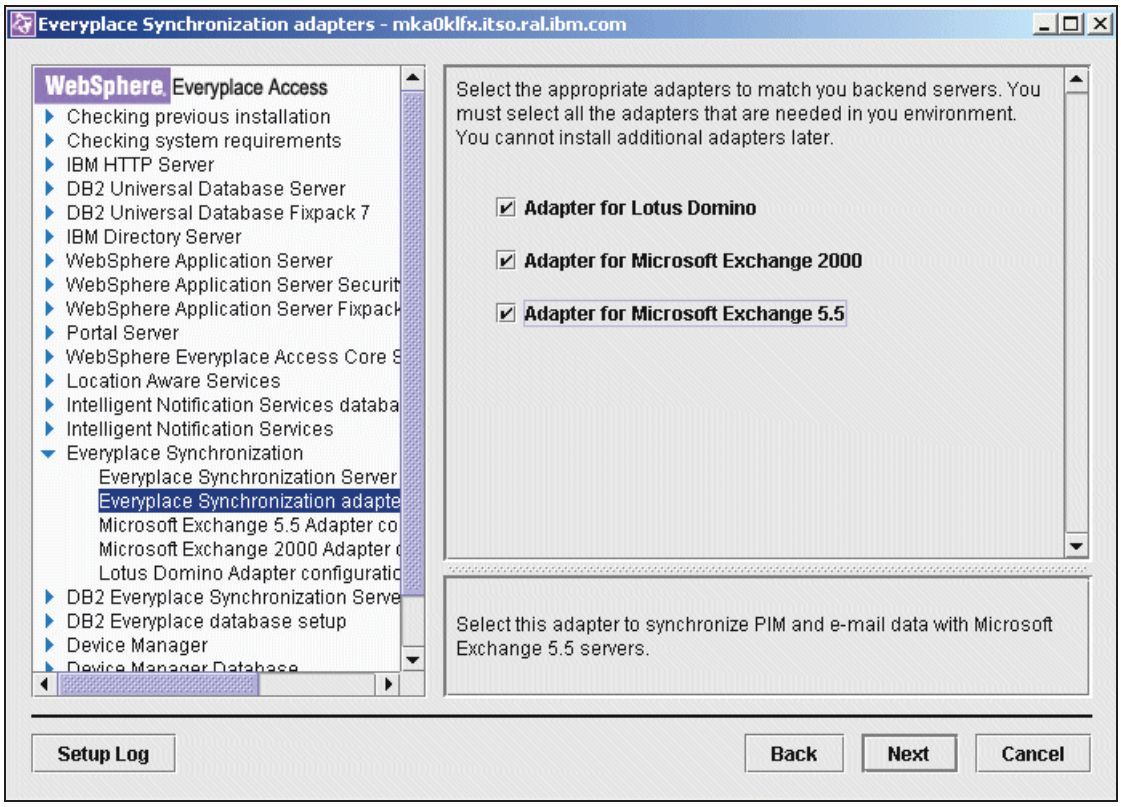


Figure A-41 Selecting the ESS adapters

3. If your are using a Microsoft Exchange server 5.5, enter these details for the server:

Host name	`<yourexchangeserver_hostname>`
Username	`<exchange_username>`
Password	*<exchange_password>*
Domain name	`<windows_domain_for_exchange_server>`

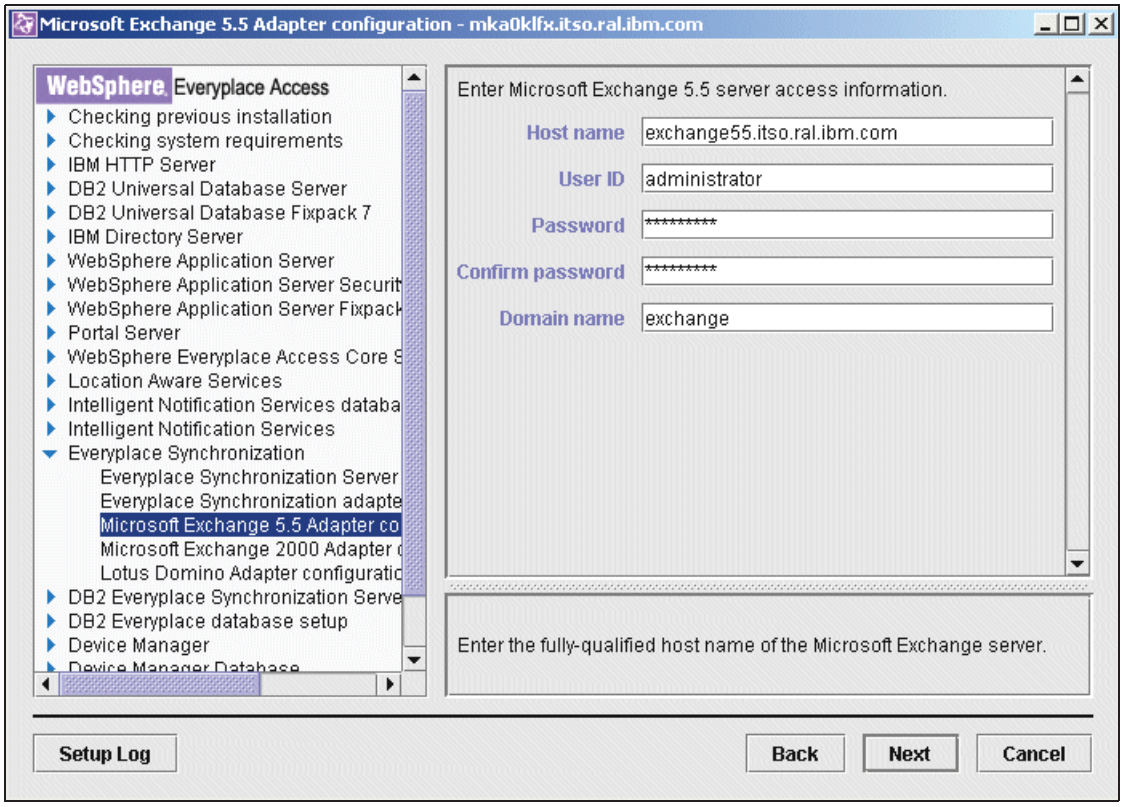


Figure A-42 Entering the Exchange 5.5 server information

4. Enter the following to connect to Microsoft Exchange server 2000:

   Host name `<yourexchange2kserver_hostname>`

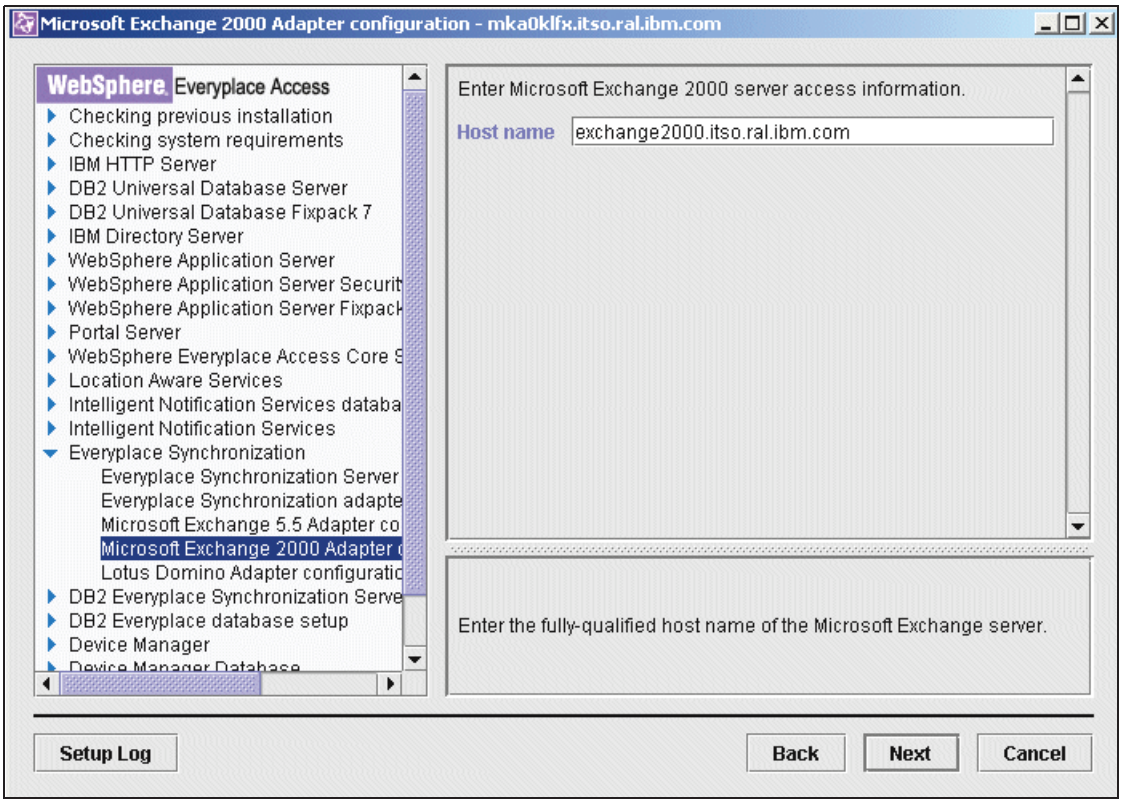


Figure A-43 Entering Exchange 2000 Server information

5. For the Lotus Domino Server, enter:

Host name	`<yourdominoserver_hostname>`
Username	`<domino_username>`
Notes Password	`<domino_password>`

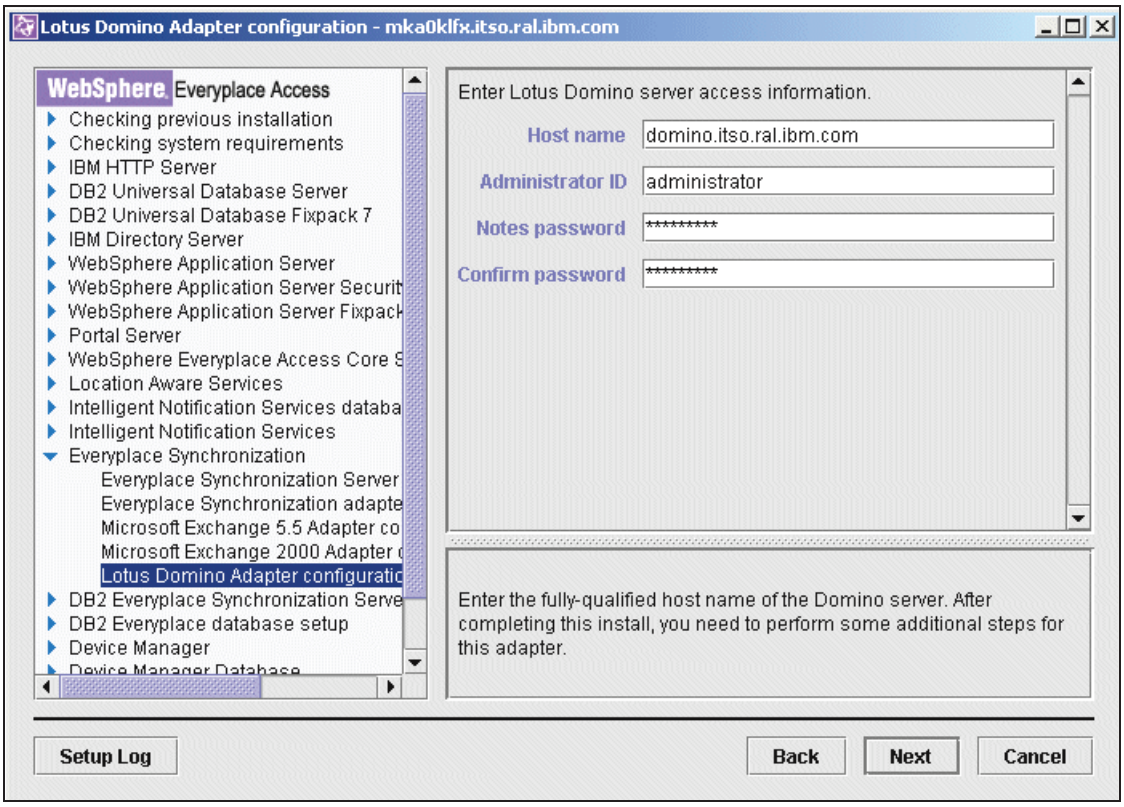


Figure A-44 Entering the Lotus Domino Server information

DB2 Everyplace Config

Configure access to DB2 by entering the following:

WebSphere Application Path	`c:\WebSphere\AppServer`
DB2 Everyplace Path	`c:\WebSphere\IBMSyncServer\db2e`
User Name	`db2admin`
Password	`db2admin`

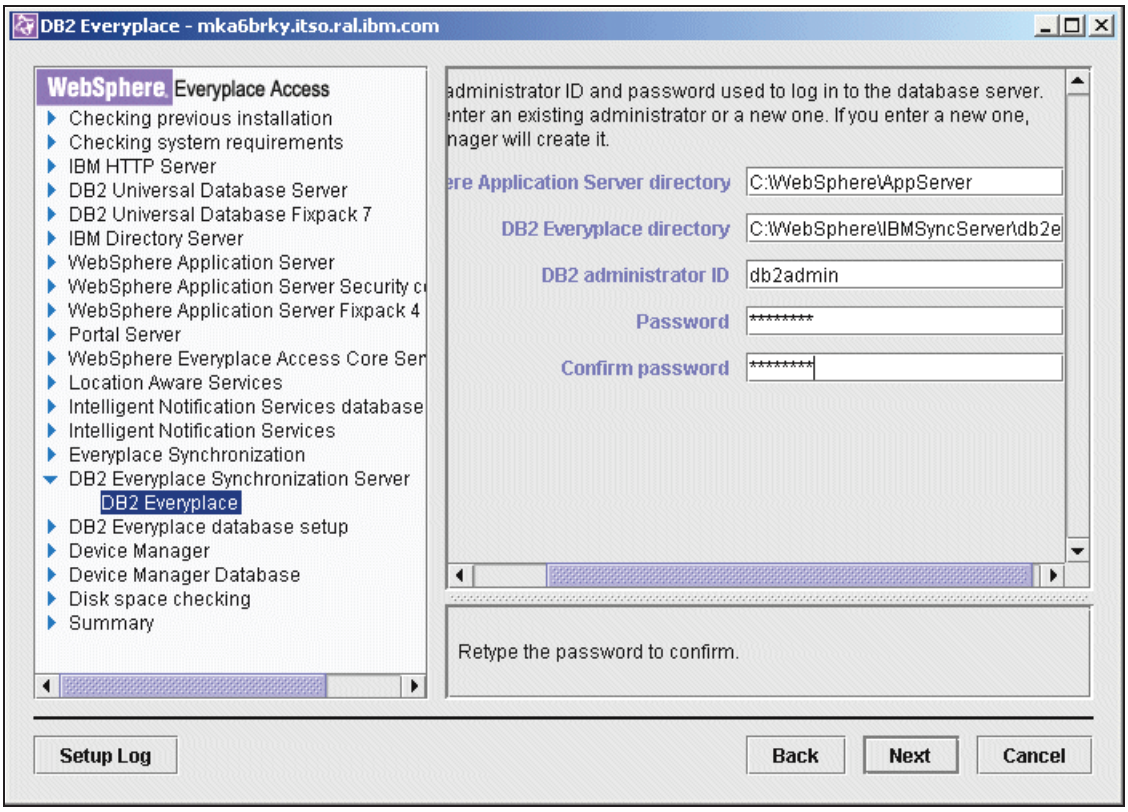


Figure A-45 DB2 Everyplace

Device Manager - Configure DMS

Enter the values for DMS to be able to connect to the database that it will populate:

Database user	`db2admin`
User password	`db2admin`

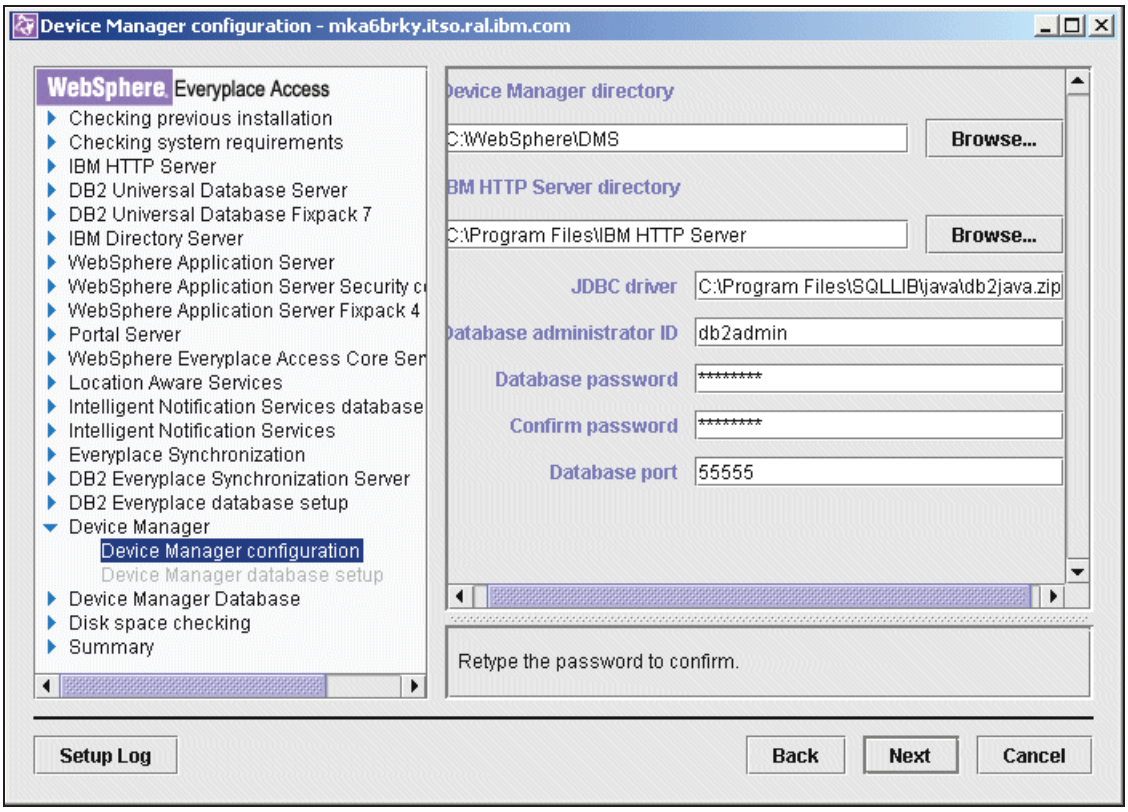


Figure A-46 DMS config

Installation begins

Once the initialization has completed, you are ready to begin the actual install process. The Display Summary window lists all the components that will be installed when the Next button is clicked.

1. From the Display Summary window:
   a. Click the **Save Configuration** button.
   b. Click **Next** to start the installation.

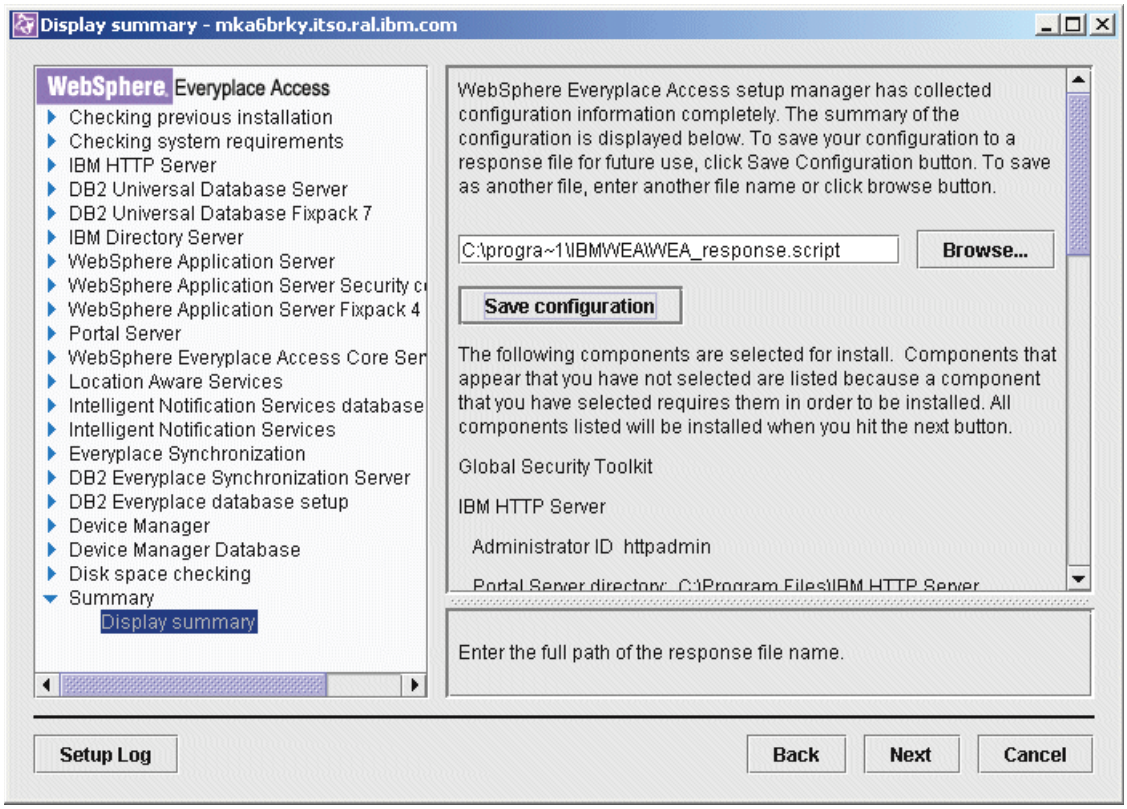


Figure A-47 Response file

2. As the components are installed the status bar displays the progress of the installation. You can click the **View log** button to check whether a specific component was successfully installed.

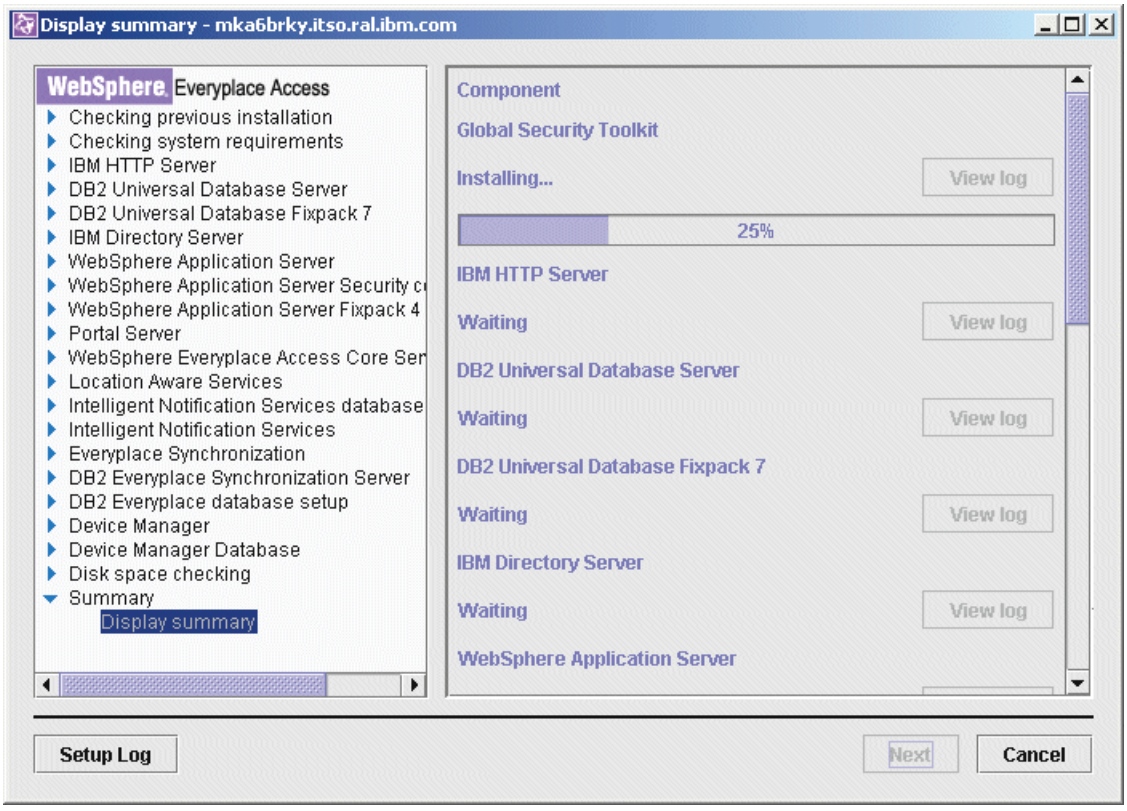


Figure A-48 Install Global Security

3. During the installation process, you will be prompted to insert CDs. A window will be displayed asking for the CD by number.

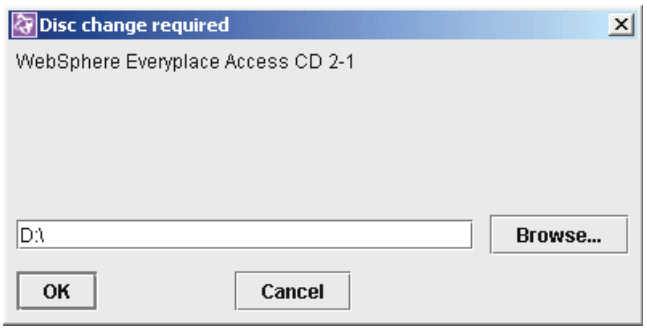


Figure A-49 Change CD

4. During the installation of DB2, you will be prompted to reboot. Go ahead and reboot the machine. The installation will resume where you left off.



Figure A-50 Install DB2 reboot

5. Continue with the steps in the next section, since these steps are required about half way through the installation in order to complete the installation successfully.

Configuring the admin role

Important: Toward the end of the install, the Configuring the Admin Role window displays. You must complete all the steps listed on the window prior to clicking OK. Step-by-step instructions are provided in this section to walk you through this process.

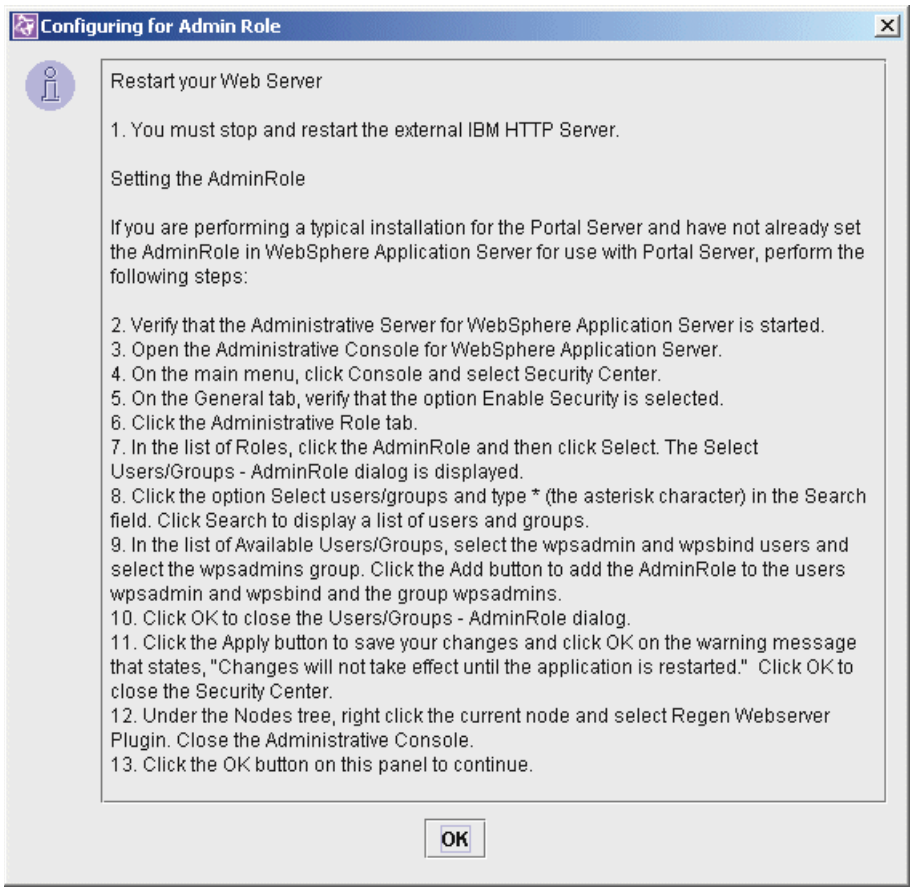


Figure A-51 Configuring for admin role

1. Select **Start -> Programs -> IBM HTTP Server -> Stop HTTP Server**. A window will show you the progress of the server stopping. Once the server has stopped, the window will close.

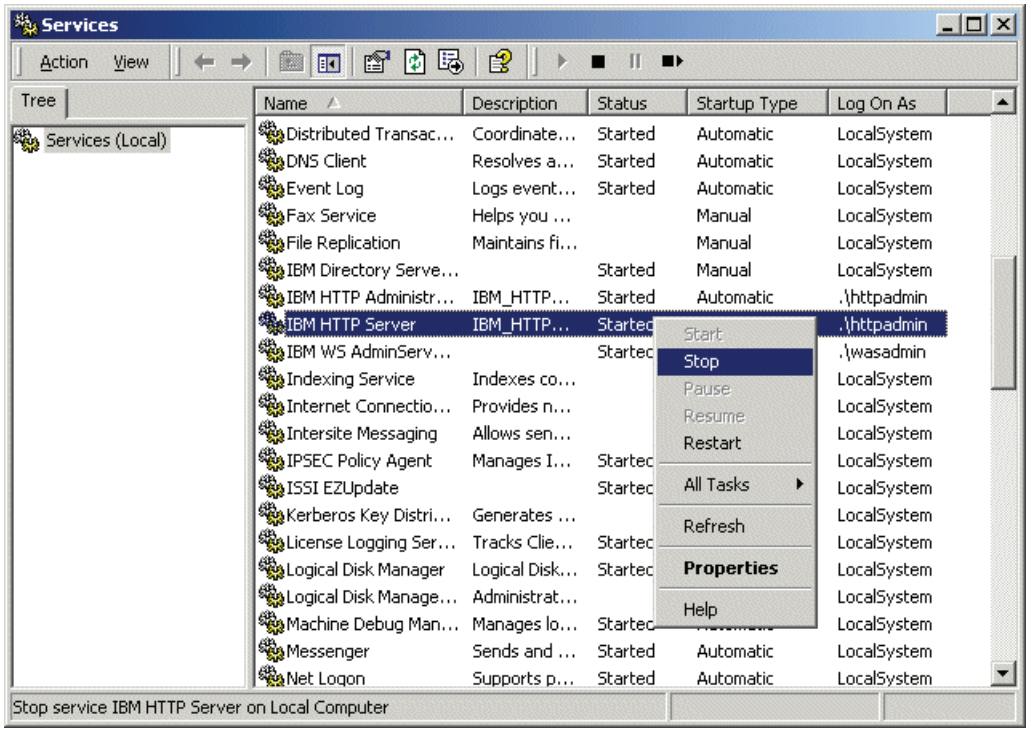


Figure A-52 Stop IBM HTTP Server

2. Select **Start -> Programs -> IBM HTTP Server -> Start HTTP Server**. A window will show you the progress of the server starting. Once the server has started, the window will close.
3. Verify that the WebSphere Application Server admin server is started.

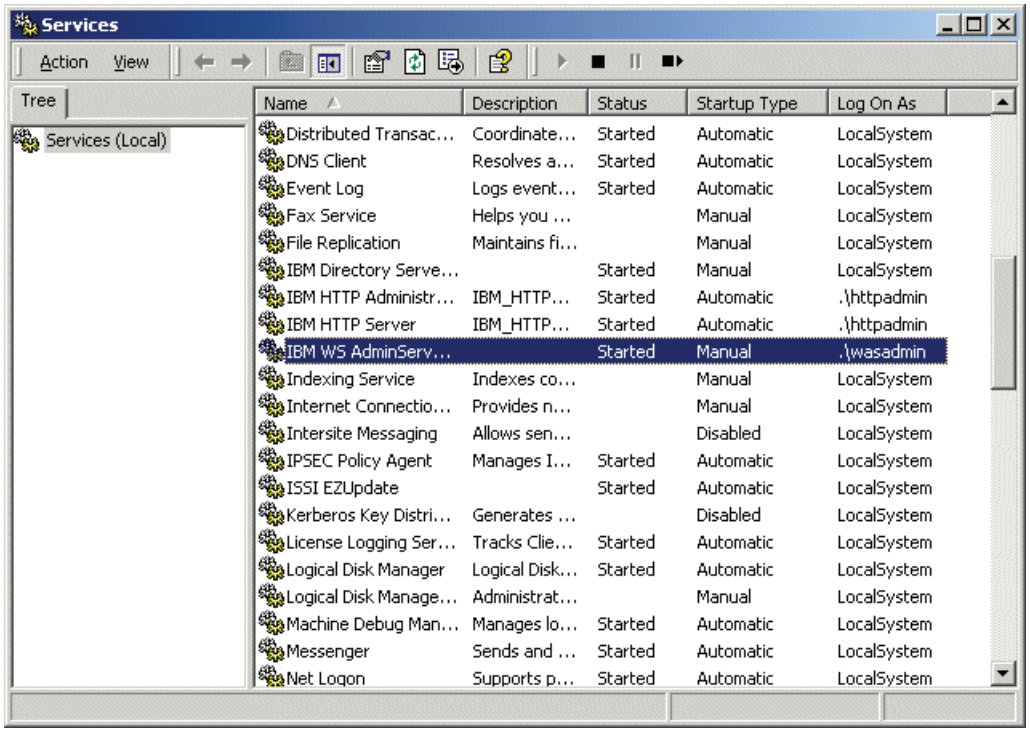


Figure A-53 WebSphere Application Server started

4. Select **Start -> Programs ->IBM WebSphere -> Application Server 4.0 -> Administrators Console**. The following window will appear.

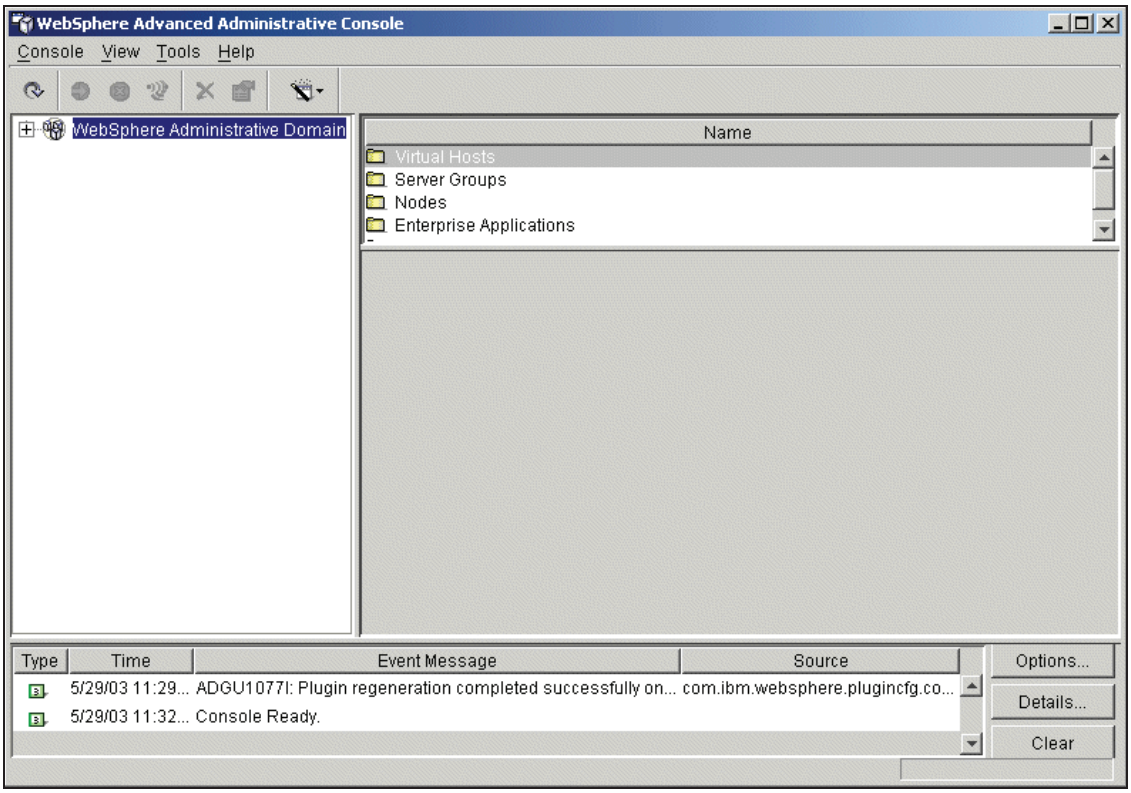


Figure A-54 WebSphere Application Server Administration Console

5. From the main menu, select **Console -> Security Center**.

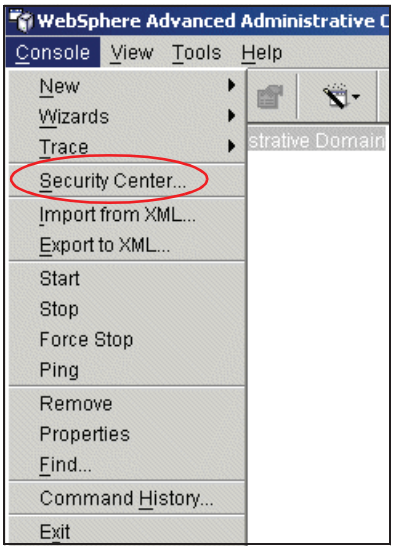


Figure A-55 Install WebSphere Application Server Security Center

6. Ensure the **Enable Security** box is checked on the General tab.

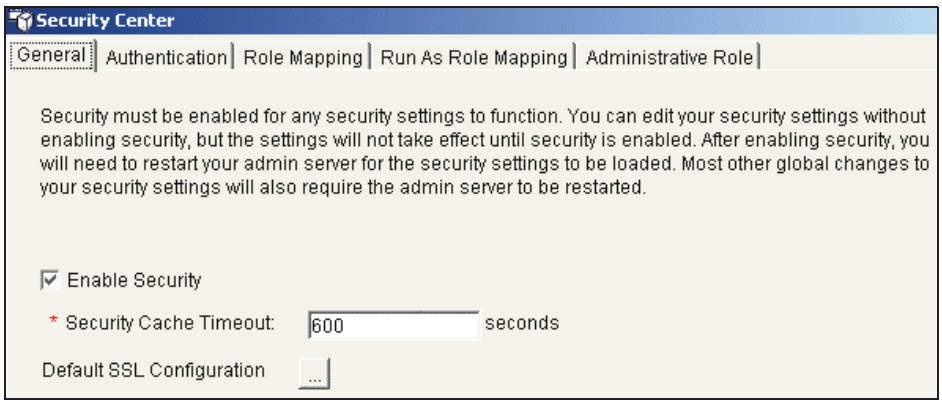


Figure A-56 Install WebSphere Application Server Enable Security

7. On the Administrative Role tab, highlight **AdminRole** and click the **Select** button.

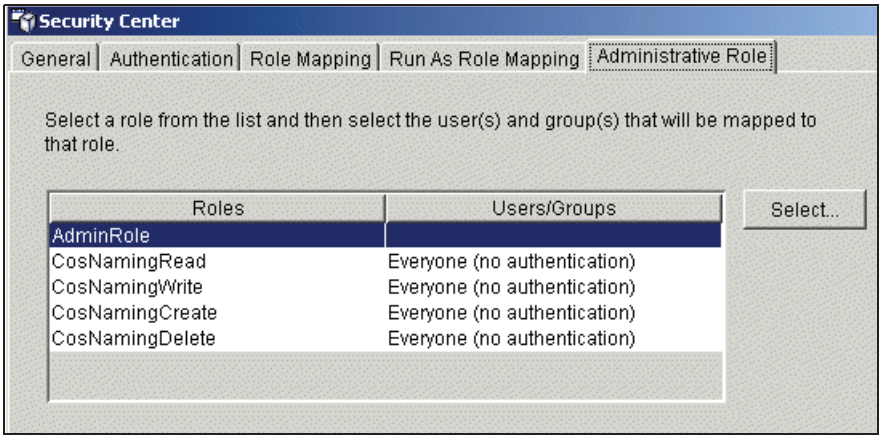


Figure A-57 Install WebSphere Application Server Admin Role

8. Select the **Select users/groups** check box, enter an * (asterisk) in the Search field and click **Search**.
9. In the Users folder, highlight the **wpsadmin** and the **wpsbind** users and click **Add.**

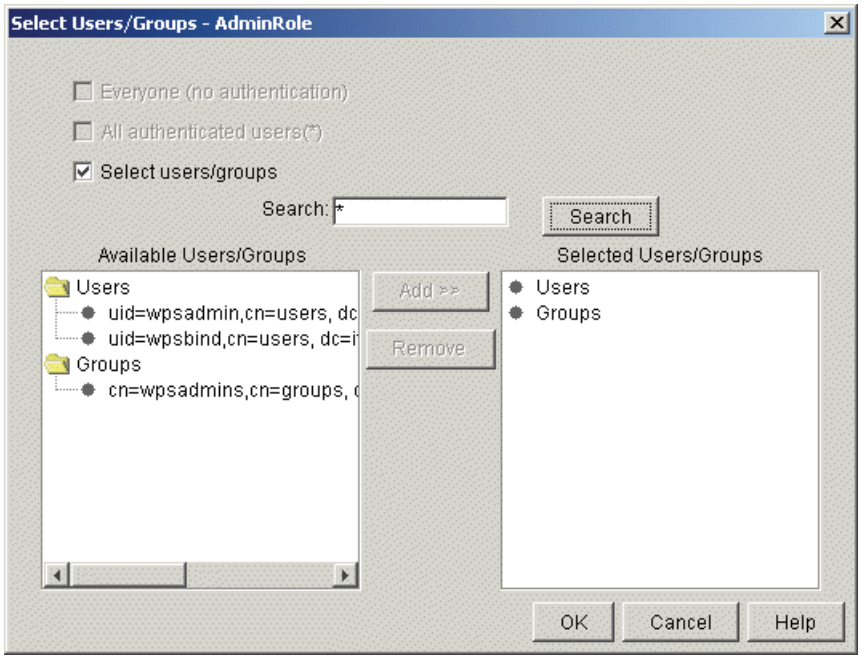


Figure A-58 Install WebSphere Application Server user groups

10. In the Group folder, highlight the **wpsadmins** group and click **Add**.

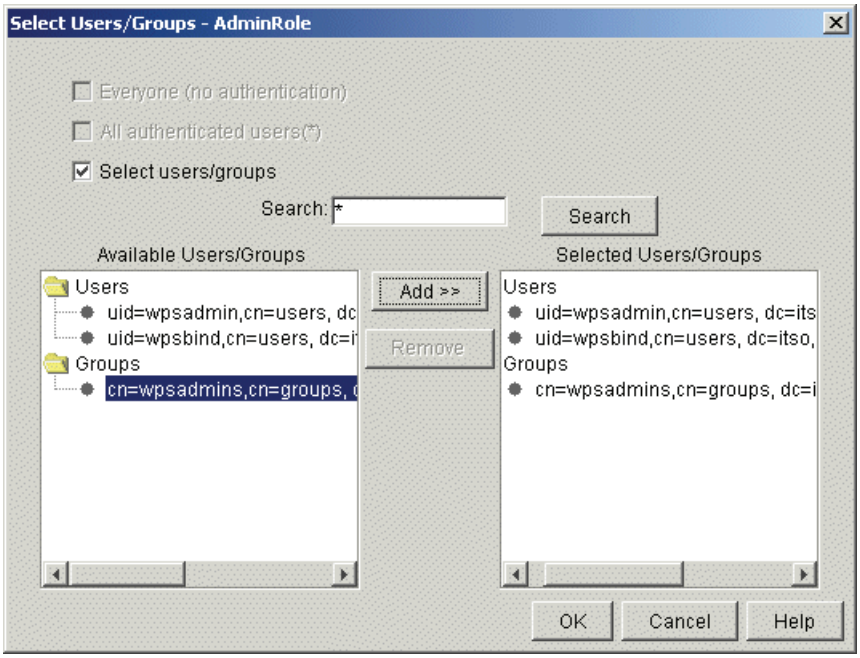


Figure A-59 Install WebSphere Application Server Setup Group

11. Perform the following actions:
    a. Click **OK** on the Select User/Groups window
    b. Click **Apply** on the Security Center window
    c. Click **OK** on the message box
    d. Click **OK**

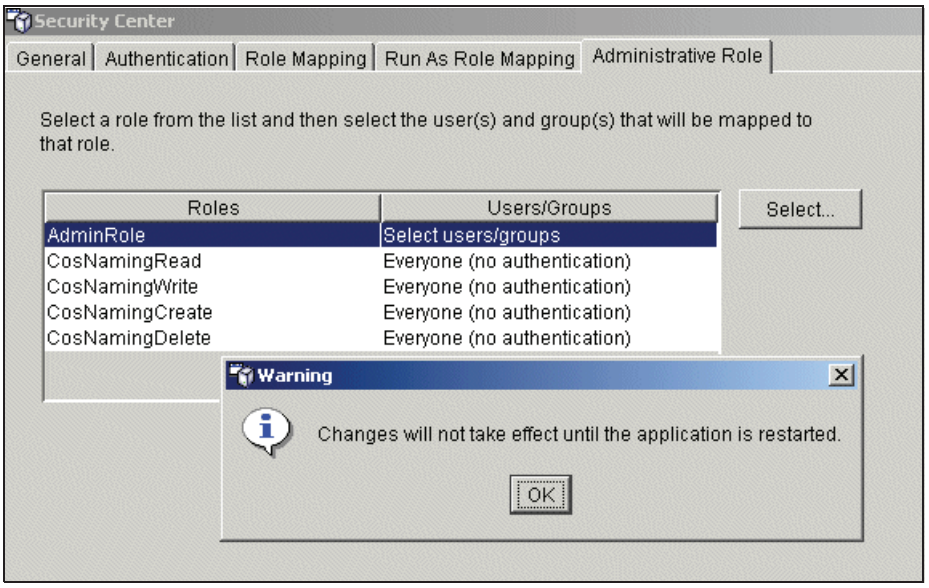


Figure A-60 Install WebSphere Application Server Restart

12. Under the Node tree, right-click the current node and select **RegenWebserver Plugin**.

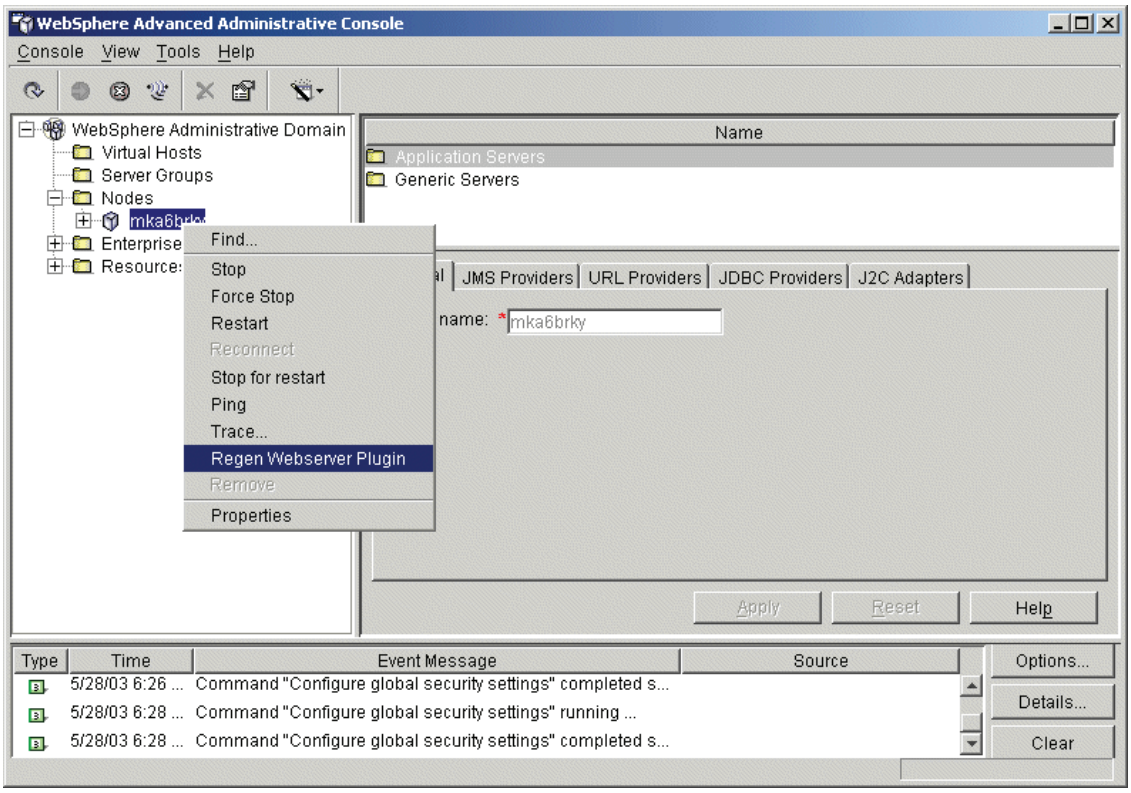


Figure A-61 Install WebSphere Application Server Regen

13. Close the Administration Console.
14. Click **Yes** to confirm the message.
15. Click **OK** on the Configure for Admin Role window.
16. Confirm the previous steps have all been completed. Then click **No**.

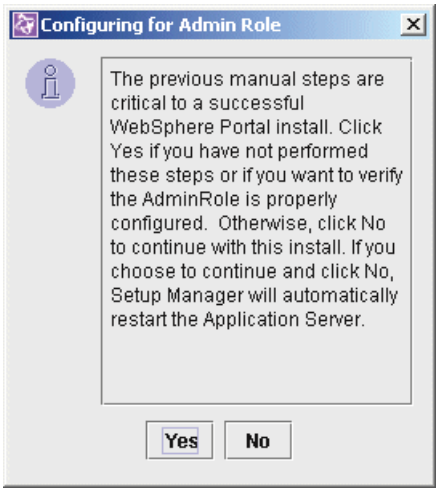


Figure A-62 Install Config complete

17. The installation will now continue.
18. When the installation is complete, you will see the screen below. Click **OK**.

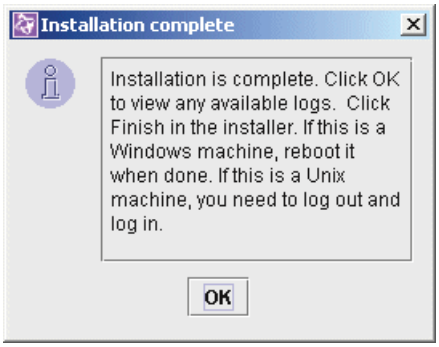


Figure A-63 Successful install

19. Click **Finish** then reboot the machine to complete the process. You can now log on with your user ID and password and switch to being on the network.

Verifying the install

Once the installation is complete, there are a few steps you need to perform to verify the install was successful:

1. Start the IBM Directory Server V4.1 which brings up the LDAPDB2.

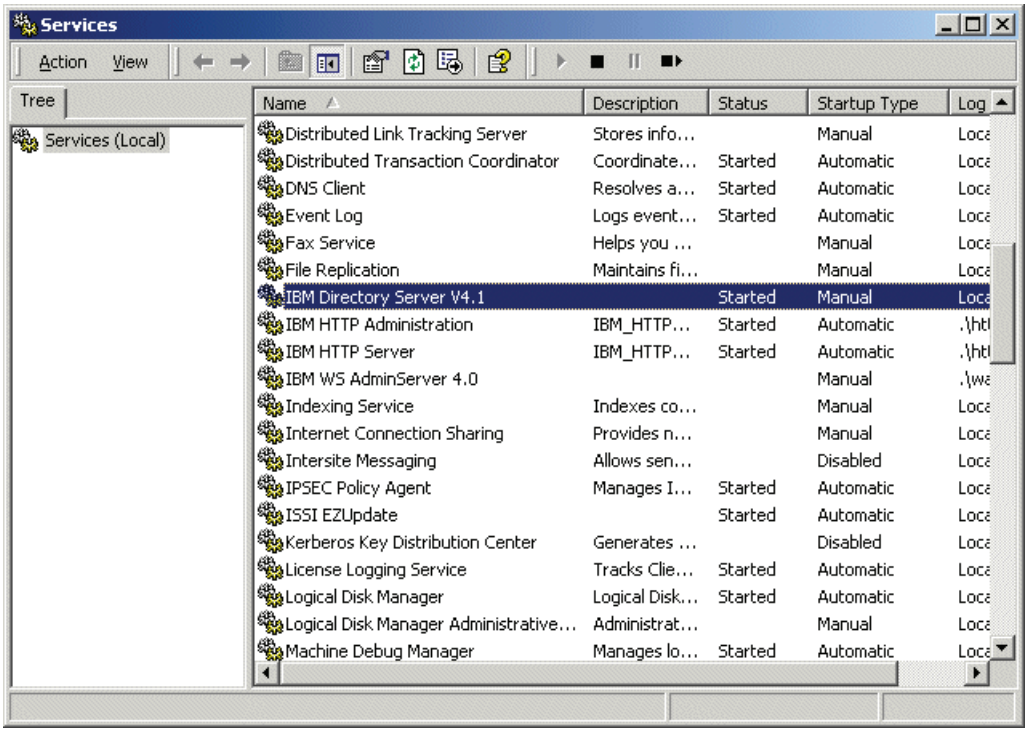


Figure A-64 Start IBM Directory

2. Start the IBM WebSphere AdminServer 4.0 services from the Services window.
3. Wait to ensure it starts completely. You can check the task manager to see when the system quiets down.
4. Start the IBM HTTP Server in the Service window, if it is not already started.
5. If you installed ESS, start the IBM Everyplace Synchronization Server Service.
6. Once the services are started, select **Programs -> IBM WebSphere -> Application Server v4.0 -> Administrator's Console**. Once the WebSphere Advanced Administration Console is started, log on with a user ID and password of `wpsbind`.
7. Within the WebSphere Advanced Administration Console, select **WebSphere Administrative Domain -> Nodes -> yourNode -> Application Servers**.
   a. Ensure the Portal Server, IBM ESS, DMS, DM console_Appserver, WebSphere Everyplace Access server and Everyplace Recent Alert

Server are running (indicated by a green arrow). If any of these components are not running, highlight them and click **Start**.

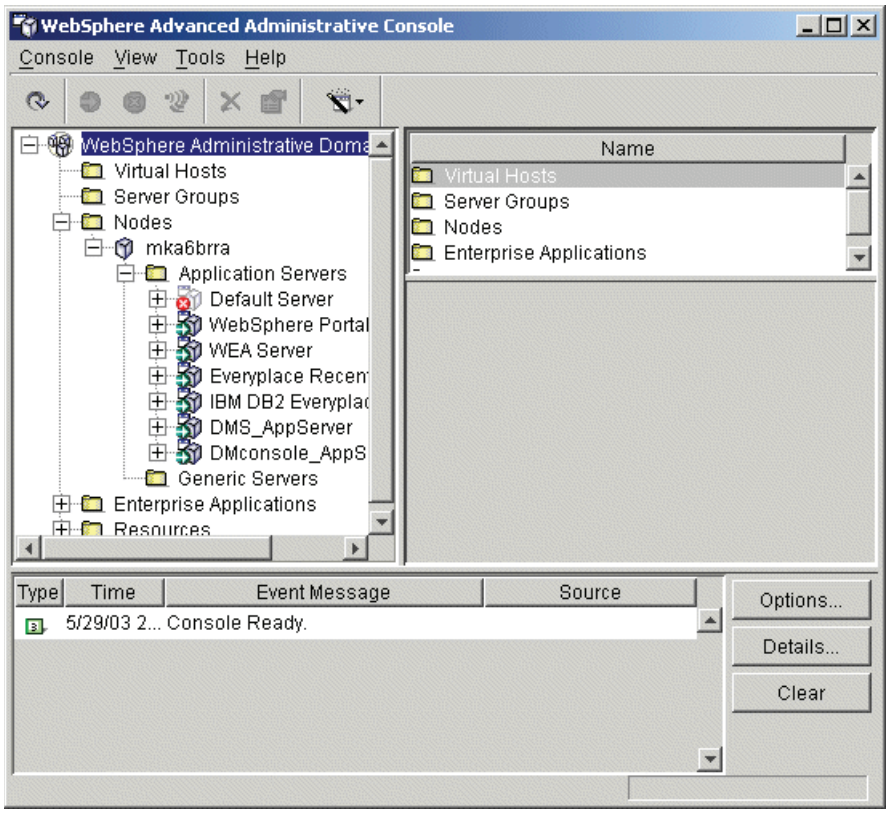


Figure A-65 Verify Services Started

8. The next step to verify the installation process is to start a Web browser and enter the following URL: `http://<server_name>/wps/portal` where the `<server name>` is the fully qualified address of your WebSphere Everyplace Access server (for example `YourMachineName.itso.ral.ibm.com`). If the installation was successful, you should see the WebSphere Everyplace Access welcome window.

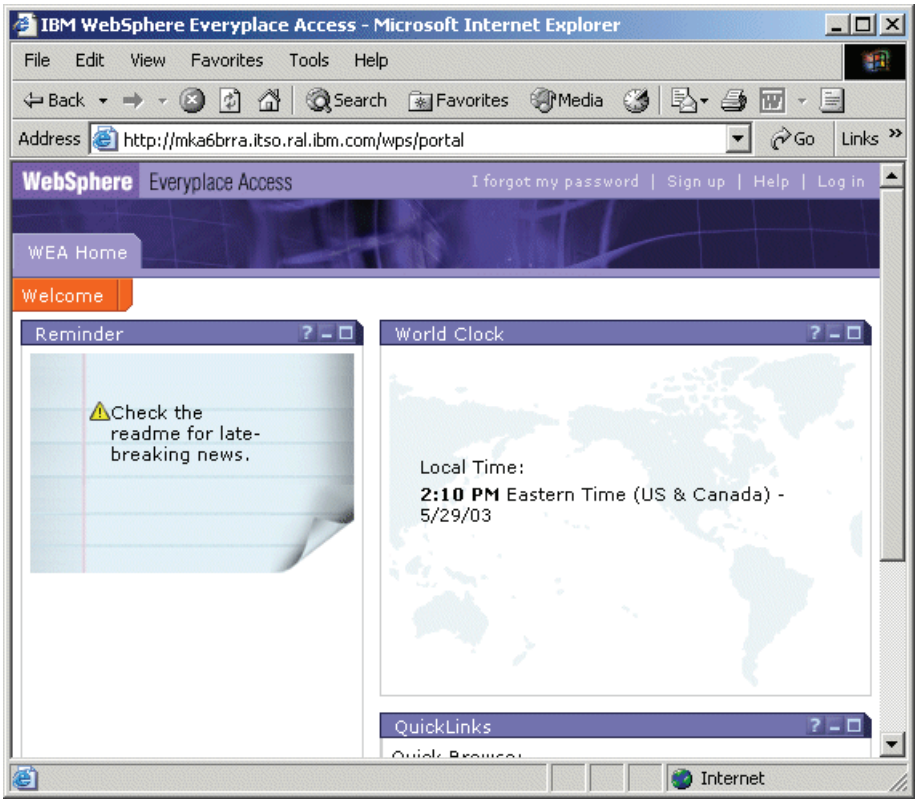


Figure A-66 WebSphere Everyplace Access welcome window

9. The last step is to verify the availability of all the WebSphere Everyplace Access features. This last step is time consuming and should therefore be done only if you have doubts about the installation. The following features are verified:

   – DMS portlets are available
   – Productivity portlets are available
   – PIM Notes/Exchange portlets are available
   – ESS portlets are available
   – ESS server starts
   – INS server starts
   – INS portlets are available
   – SIA portlets are available
   – LAS portlets are available
   – DMS server starts
   – DM Console starts, can connect to server
   – DB2e Server starts, can connect to MDAC

Troubleshooting

The installation log files are located in C:\Program Files\IBMWEA.

In this directory, the main log file is named setup_mm.dd.hh.mn.log (where mm=month, dd=day, hh=hour, and mn=minute of the installation, a 24-hour format). You may find valid information in that file.

Most of the stages of the installation are performed via Run commands. The log files of these stages are located in the directory C:\Program Files\IBMWEA\RunCommand_mm.dd.hh.mn.

Whenever a problem occurs, you should first look for information in the main log file. This file should indicate in which secondary log file, depending on the stage, the information about this stage is logged.

For each indicated file in the directory C:\Program Files\IBMWEA\RunCommand_mm.dd.hh.mn, two files are actually present: thisStage.log and thisStage.stdErr.log.

For instance, let's say the symptom you have is that the LDAP server did not get updated with the WebSphere Everyplace Access scheme. This may mean that the LDIF file importing this scheme onto the LDAP server was not correctly processed. Then you look for the string `ldif` in the main log file, as shown in Figure A-67.

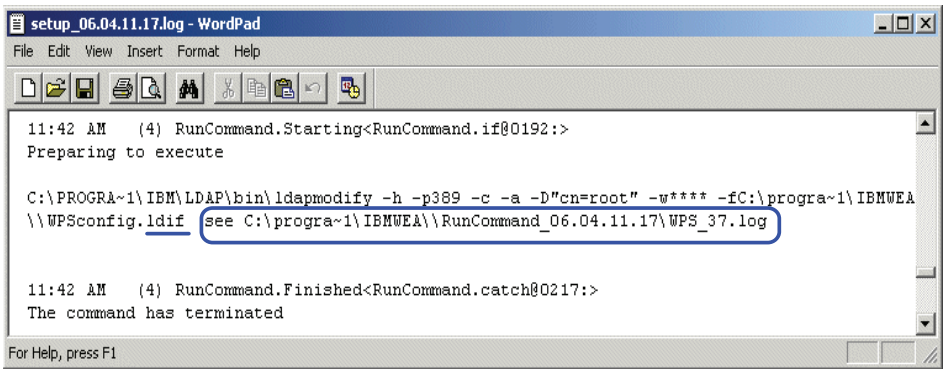


Figure A-67 Main installation logging file

No failure information is logged in the main file, so let's look at the secondary one. In our example, if an error occurs, it should then be logged into the file C:\Program Files\IBMWEA\RunCommand_06.04.11.17\WPS_37.stdErr.log.

Opening this file shows the message Figure A-68 on page 1143.

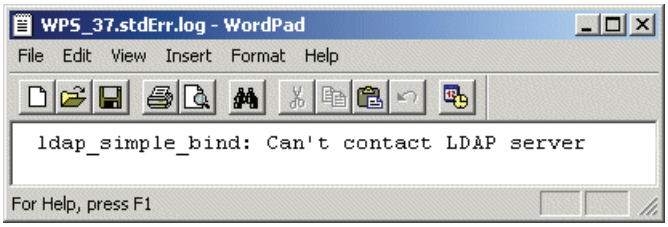


Figure A-68 Installation secondary log file

So the LDAP server could not be updated because of communication problems between itself and the installation machine. Then further investigation can be done with this information.

B

Sample Oracle Enterprise Edition installation

This appendix describes a sample installation of Oracle. It also gives an example of how to create a simple database using wizards. This appendix is intended for those who have no experience with Oracle databases.

© Copyright IBM Corp. 2003. All rights reserved.

Oracle installation

In this sample installation, Oracle9i Enterprise Edition, Release 2 (9.2.01) for Microsoft Windows 2000 and Windows NT is used.

1. Insert the CD titled Oracle9i Server into the CD-ROM drive.
2. The CD runs automatically and displays a selection window.

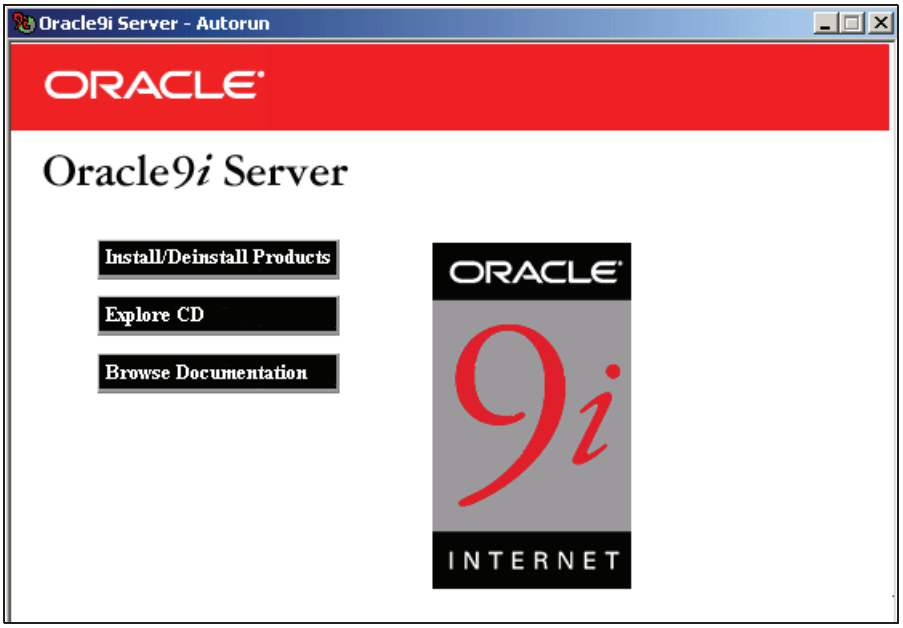


Figure B-1 Selection window

3. Click the **Install/Deinstall Products** button.The Welcome window is displayed.

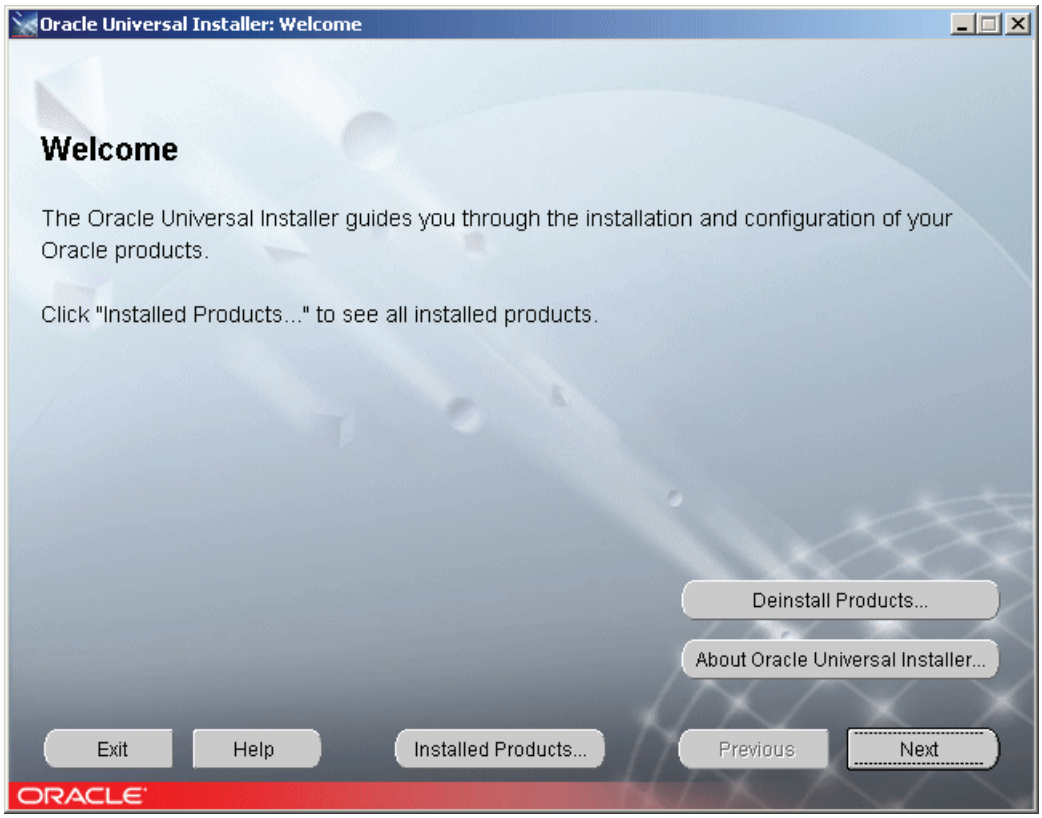


Figure B-2 Welcome window

Click **Next**.

4. In the File Locations window, accept the defaults or enter file location information. Click **Next**.

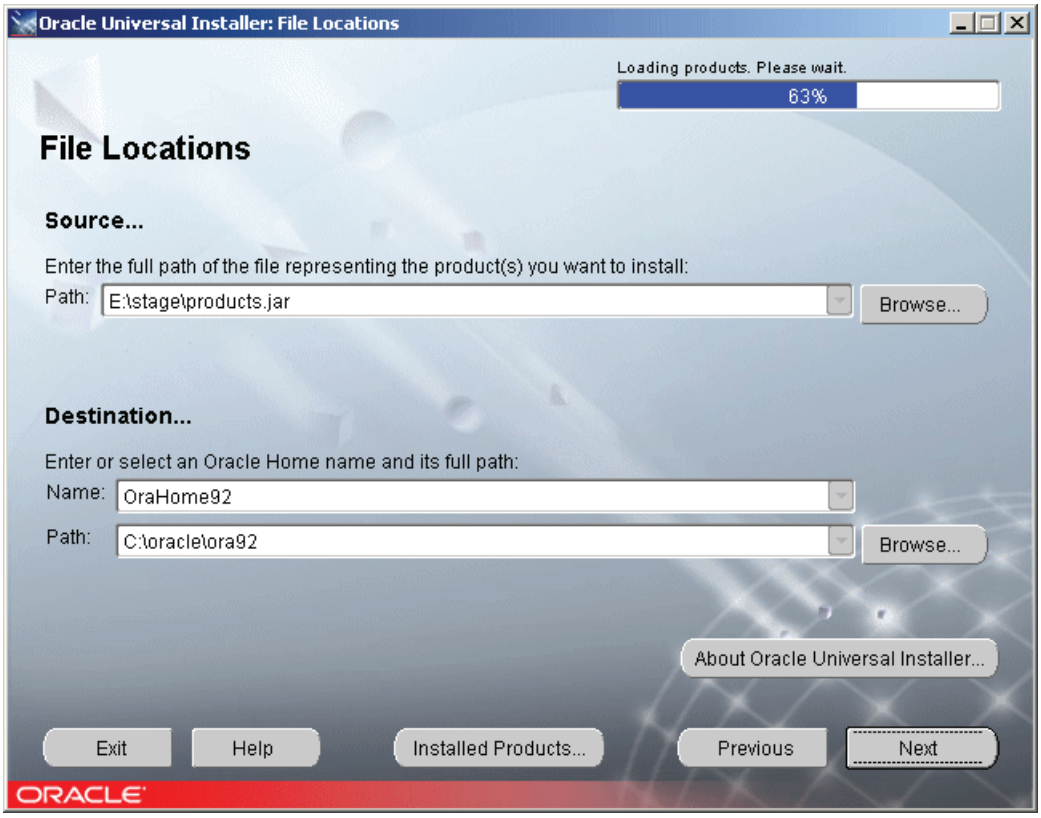


Figure B-3 Specify file locations

5. When the files are loaded, a list of available products are displayed. Select **Oracle9i Database 9.2.0.1.0** and click **Next**.



Figure B-4 Select product to install

6. Choose **Enterprise Edition** or **Standard Edition** as the installation type. Click **Next**.

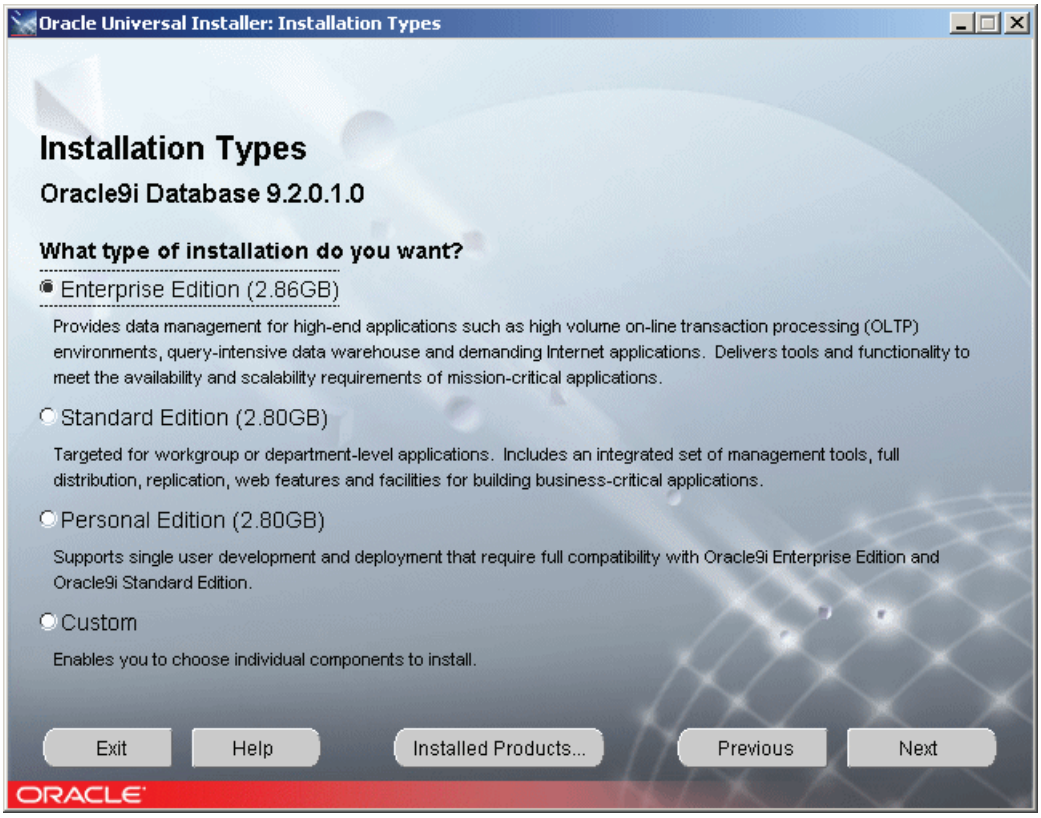


Figure B-5 Choose installation type

7. Select the database configuration adjusted to your needs. In this case, select **General Purpose.** Click **Next**.

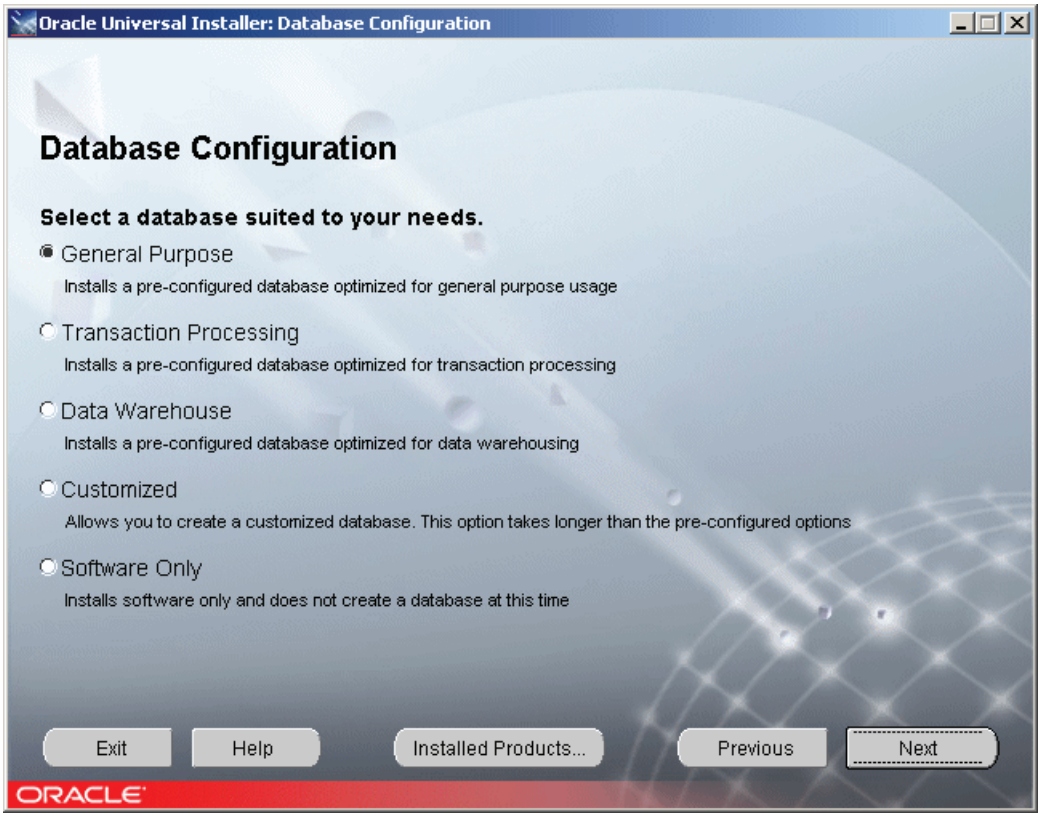


Figure B-6 Loading installation files

8. Enter a port number that is free in your machine or accept the default. Click **Next**.

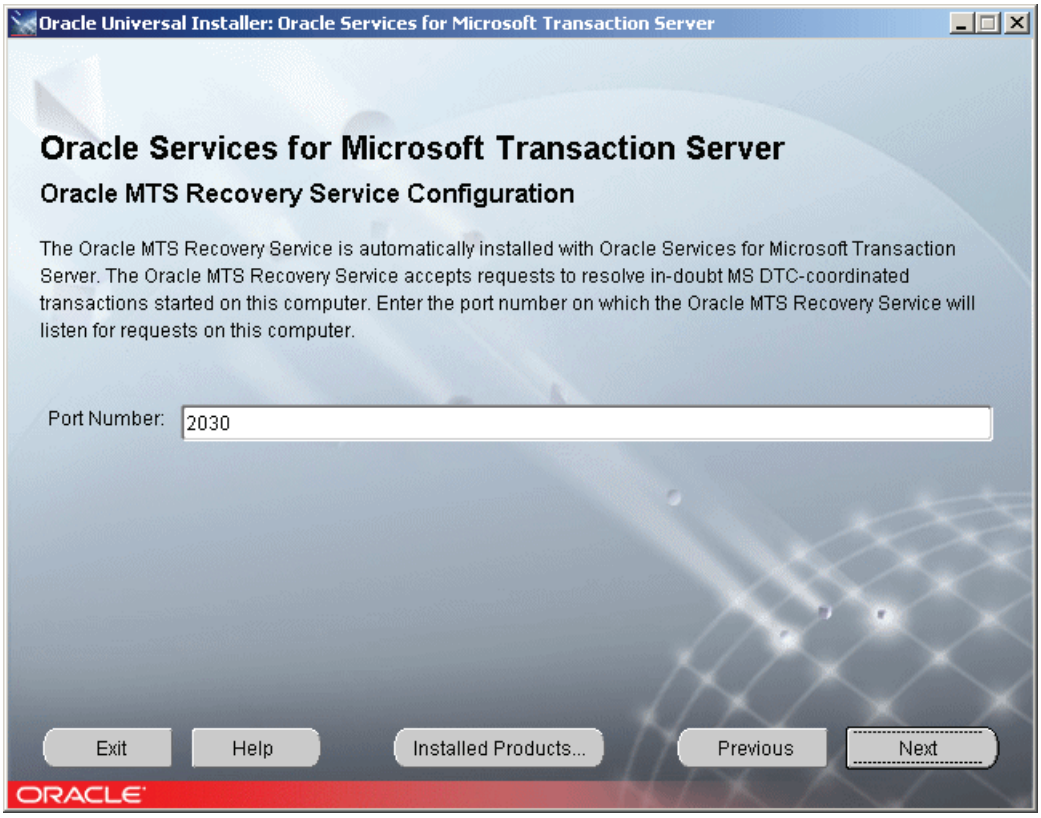


Figure B-7 Choose the port number for Oracle MTS Recovery Service

9. In the next window, fill in the Global Database Name and System Identifier (SID) fields.

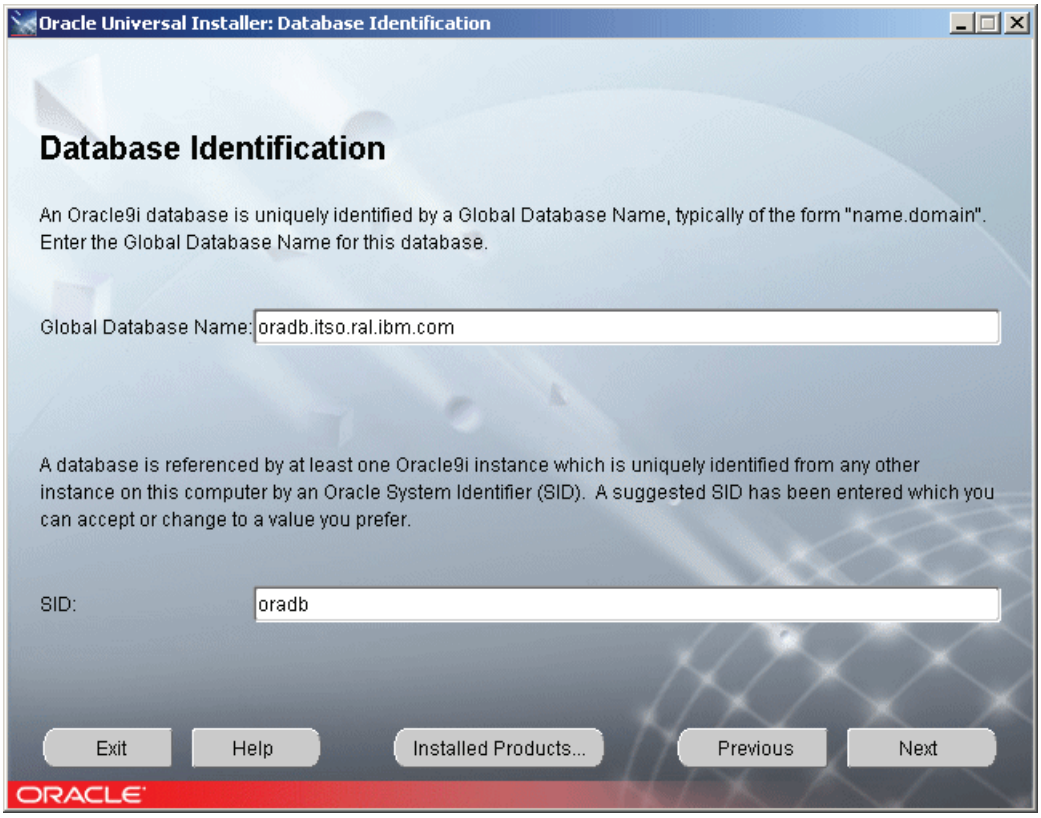


Figure B-8 Enter Global Database Name and SID

10. Choose the directory for your database files. Click **Next**.

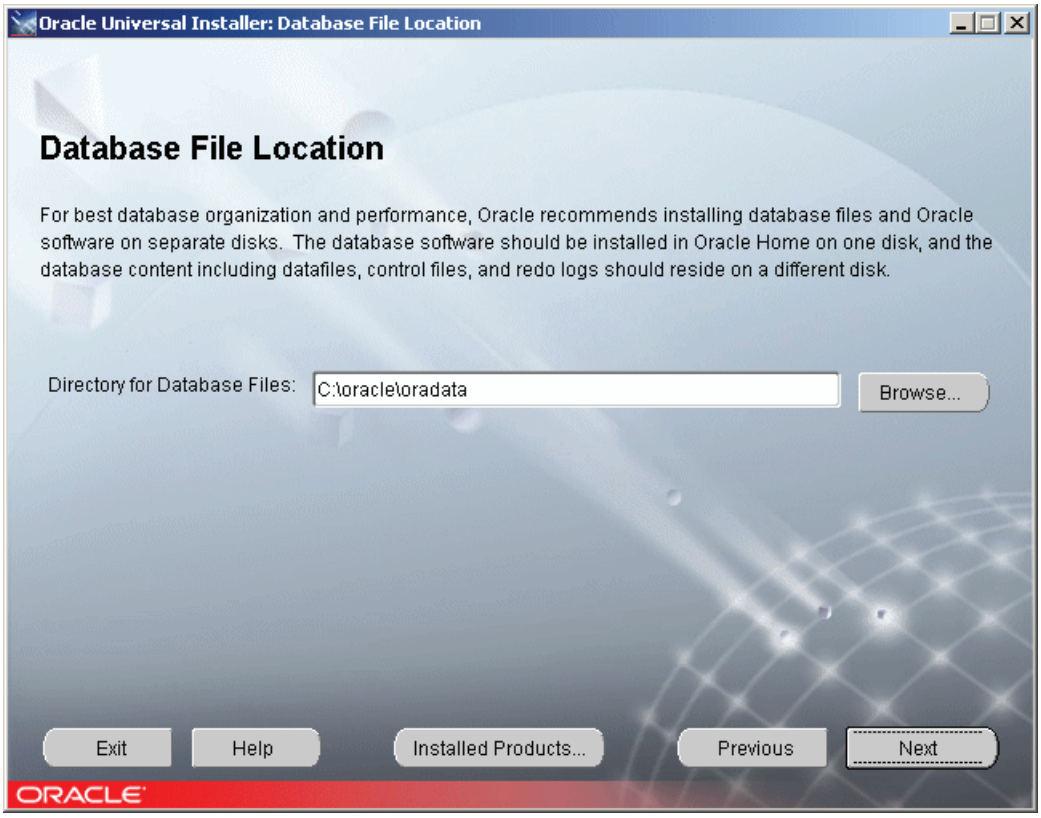


Figure B-9 Choosing the database file location

11. Choose the database character set or use the machine default. Click **Next**.

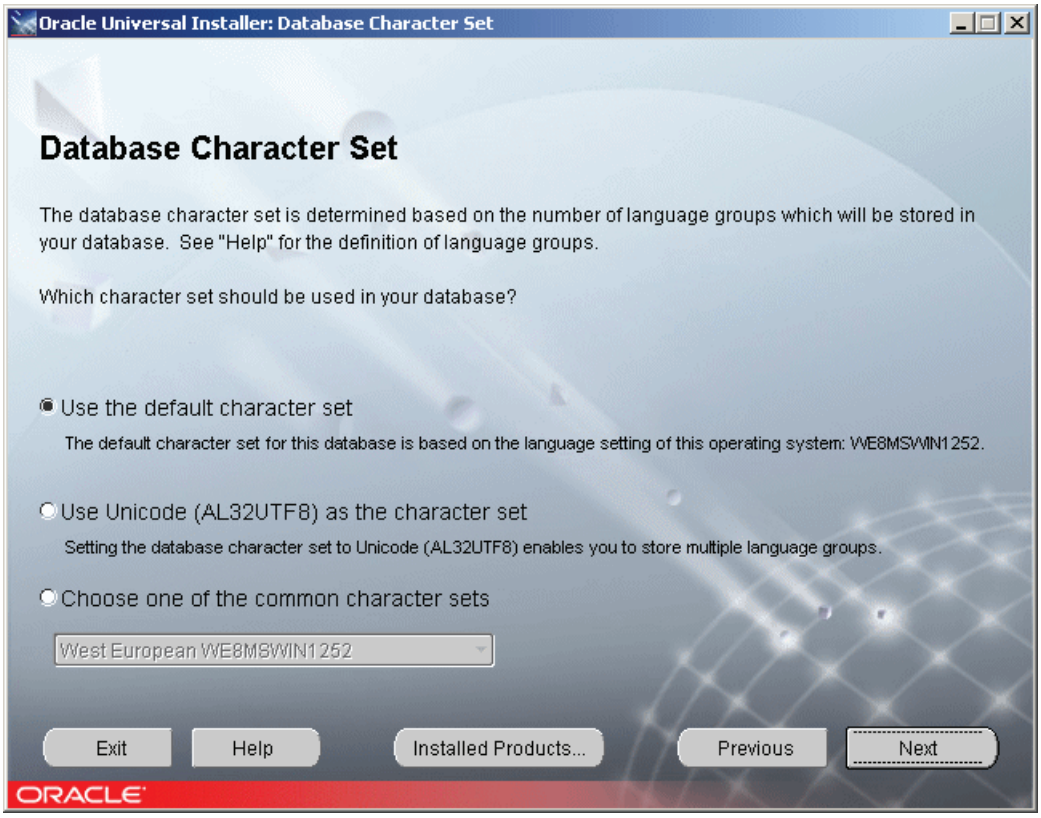


Figure B-10 Choosing the database character set

12. A summary of this installation is displayed. Review it and click **Install**.

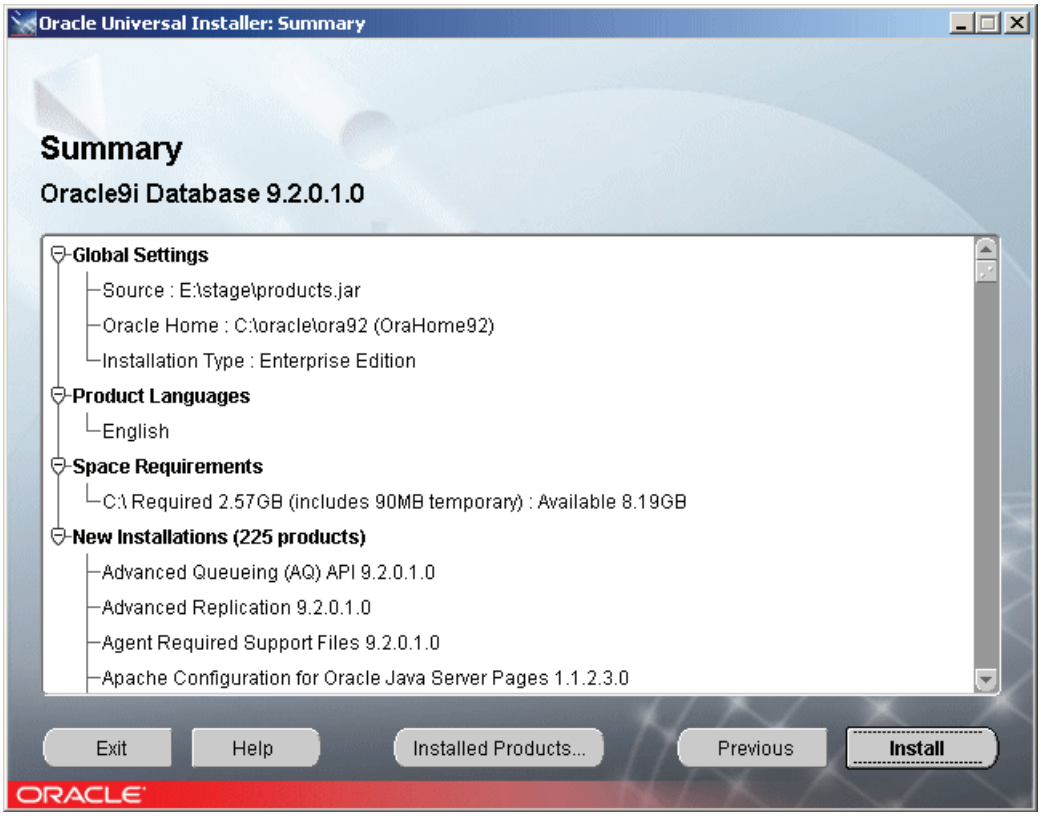


Figure B-11 Summary window

13. The installer starts installing.



Figure B-12 Installing Oracle Database program files

14. During the installation, the installer ask you to insert to the next disk. When you are done, click **OK**.



Figure B-13 Changing installer disk

15. Specify the passwords for the SYSTEM and SYS users. Click **OK**.

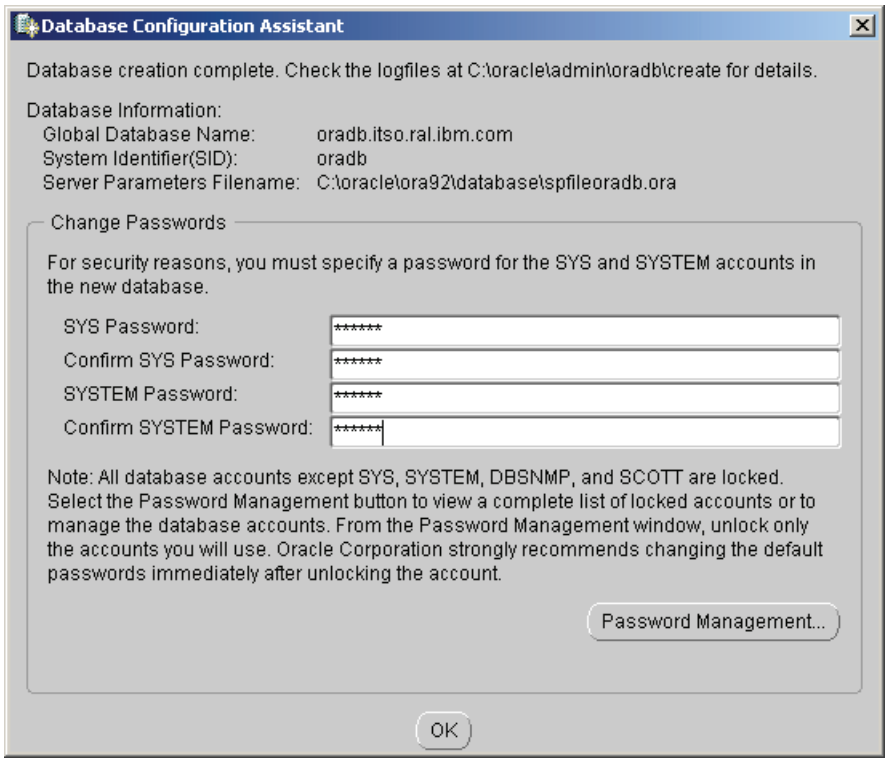


Figure B-14 Specifying the passwords for the SYS and SYSTEM users

16. Wait while the installer configures the tools for the database.

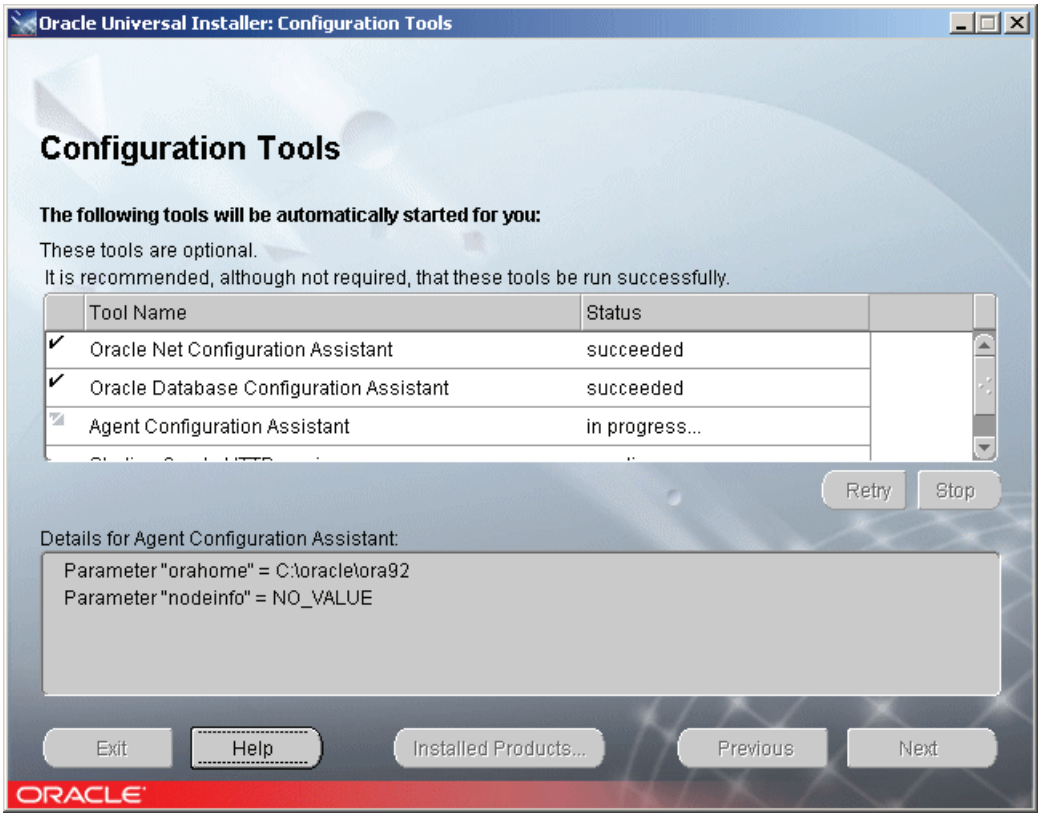


Figure B-15 Configuring tools

17. At the end of the installation, a success message is shown. Click **Exit**.

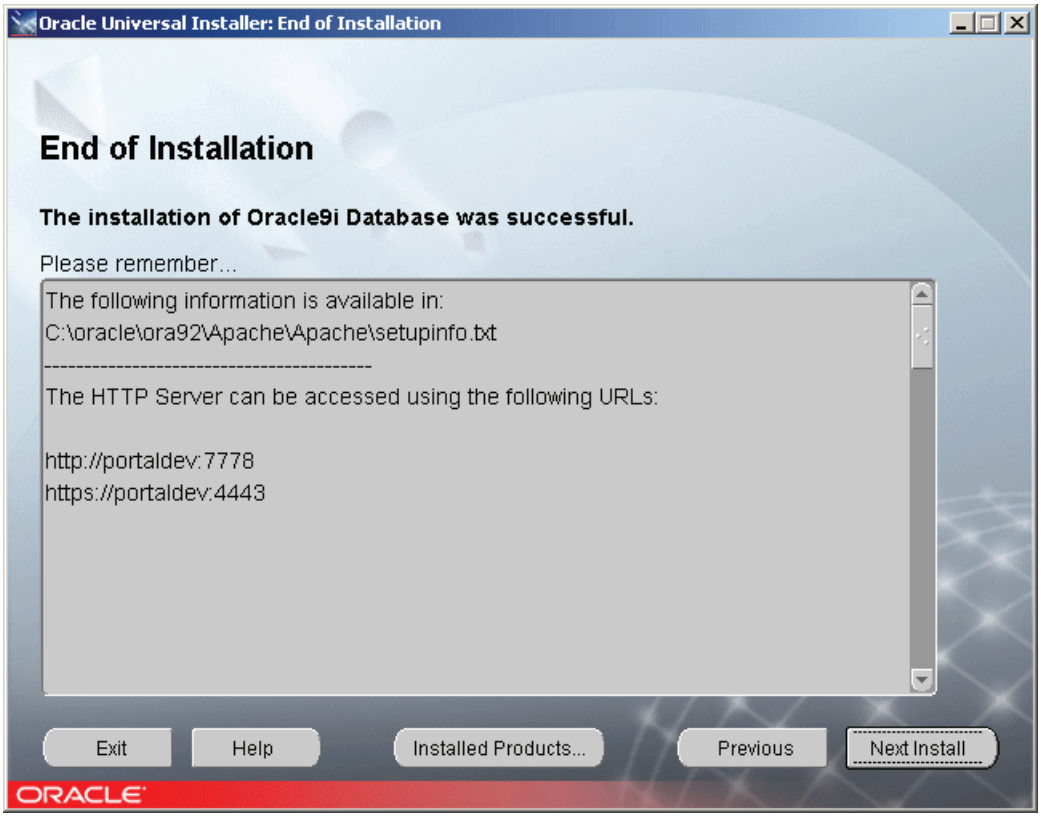


Figure B-16 Installation is successful

The Oracle Enterprise Manager is launched. You can close it. The Oracle database has been successfully installed.

Create a simple database using wizards

To create a simple sample database called SALES, do the following:

1. Click **Start -> Programs -> Oracle - OraHome92 -> Configuration and Migration Tools -> Database Configuration Assistant**.
2. The Welcome window appears. Click **Next**.



Figure B-17 Welcome to the Database Configuration wizard

3. Select **Create a database**. Click **Next**.



Figure B-18 Database Configuration Assistant

4. Select **General Purpose**. Click **Next**.

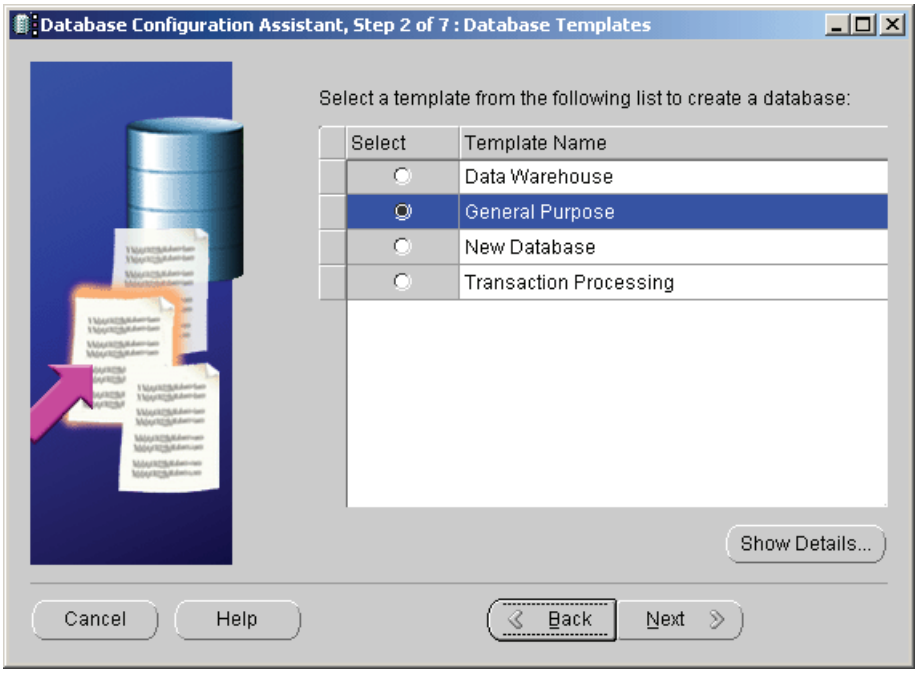


Figure B-19 Database type

5. Specify the name and System Identifier (SID) of the database to be created, for example `sales.itso.ral.ibm.com` and `sales`. Click **Finish**.

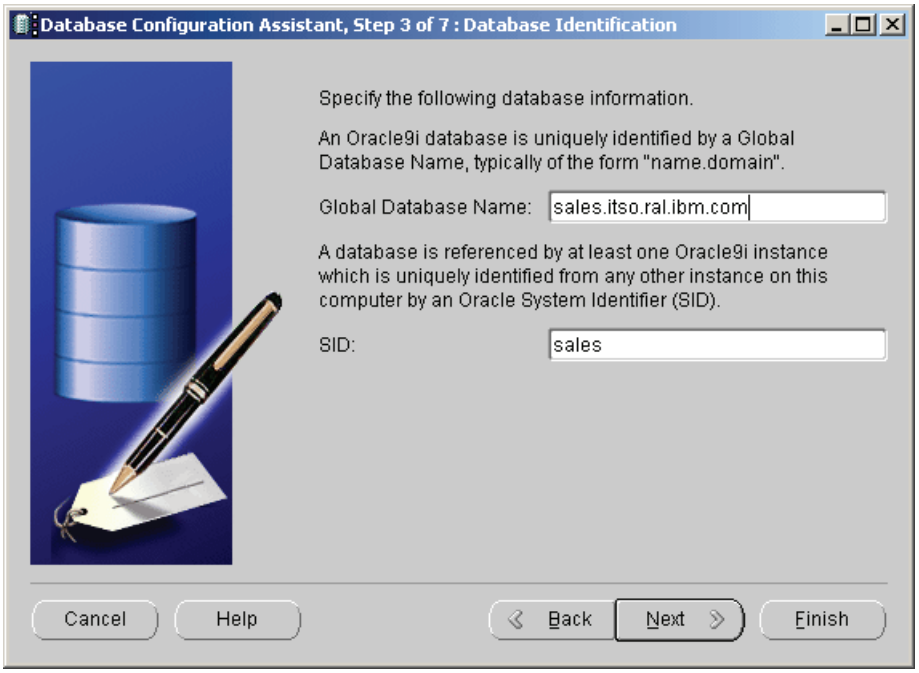


Figure B-20 Set database identification

6. A summary window with the configuration of the new database appears. Click **OK**.
7. The database creation process starts. Depending on the options selected earlier, this may take quite some time.

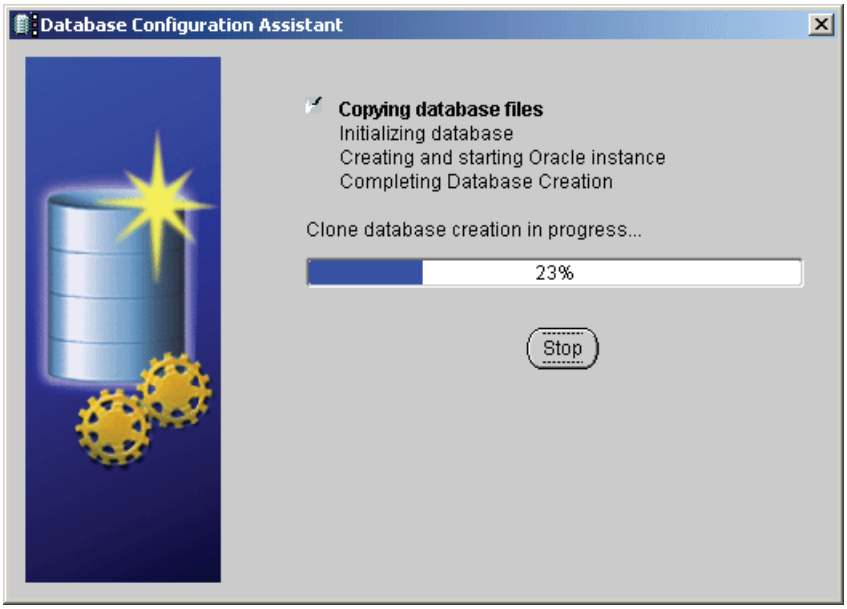


Figure B-21 Creating database

8. Specify the passwords for the SYS and SYSTEM users. Click **Exit** to finish.

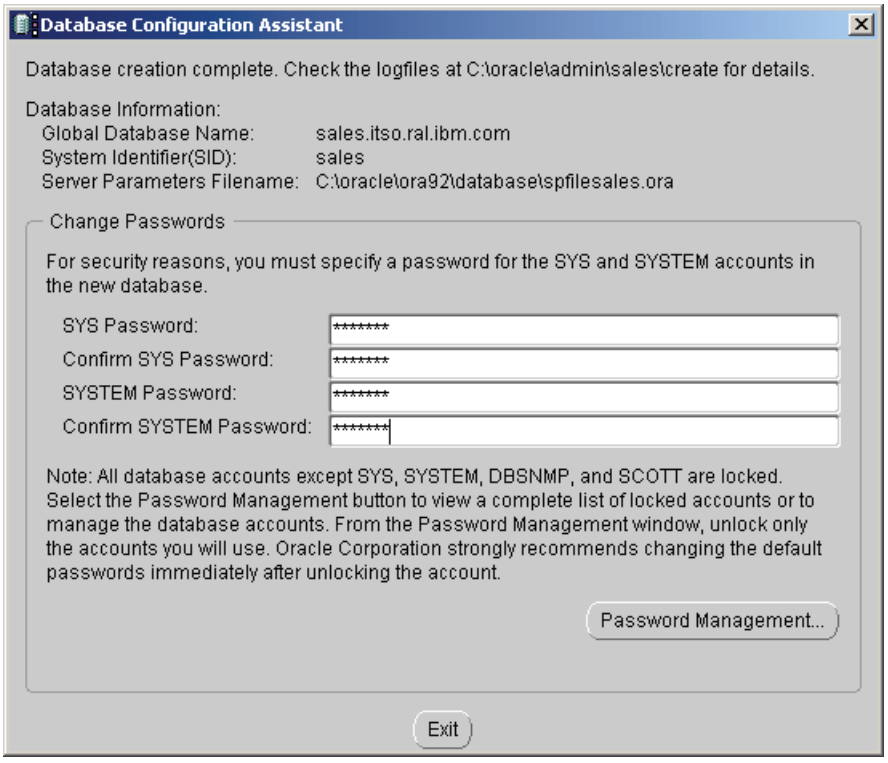


Figure B-22 Database created

The simple database has been created successfully.

Create simple table using wizards

To create a table called HISTORY in our simple sample database SALES, do the following:

1. To start the Enterprise Manager Console, click **Start -> Programs -> Oracle - OraHome92 -> Enterprise Manager Console**. On the Oracle Enterprise Manager login window choose **Launch standalone** and click **OK**.



Figure B-23 Running Enterprise Manager Console in standalone mode

2. On the tree view, double-click **Network -> Databases -> SALES.ITSO.RAL.IBM.COM** to open the SALES database administrative console.

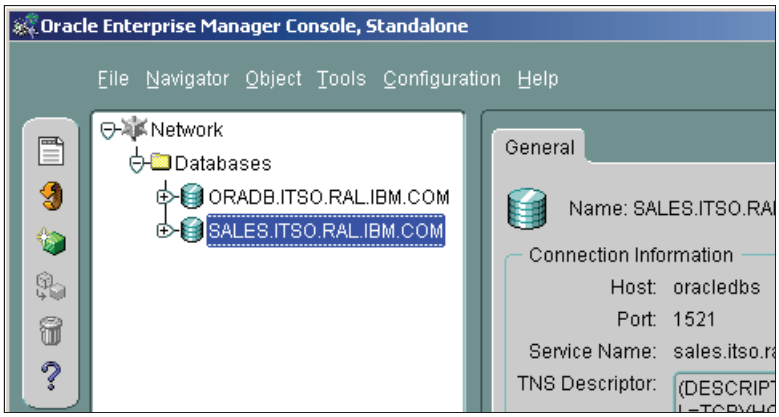


Figure B-24 Opening the SALES database administrative console

3. Enter the administrator's user name and password. Click **OK**.

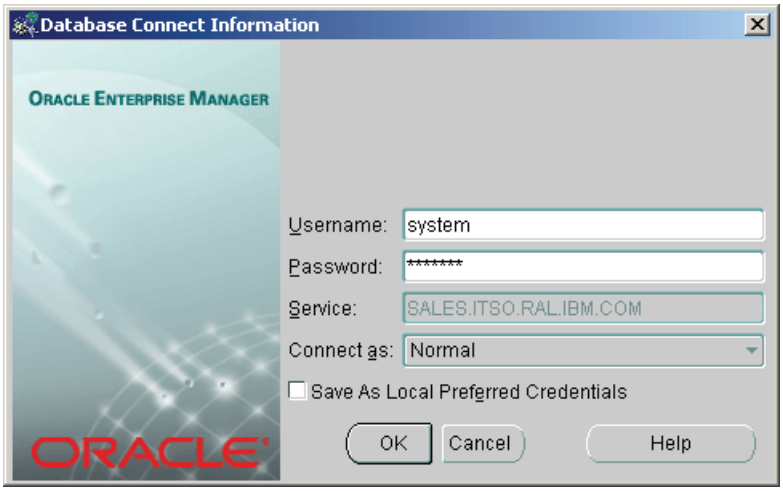


Figure B-25 Enter administrator's user name and password

4. Expand the Schema object for database SALES. Expand the SYSTEM schema. Right-click **Tables -> Create Using Wizard**.

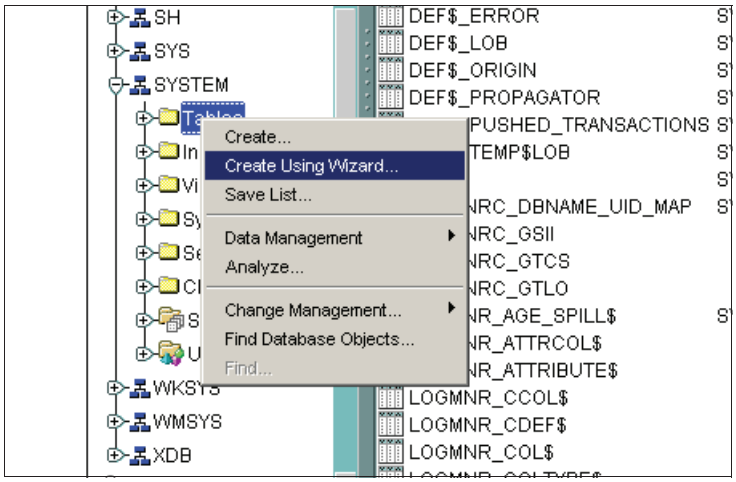


Figure B-26 Create table using wizard

5. Enter a name for the new table to be created, for example `HISTORY`. Click **Next.**

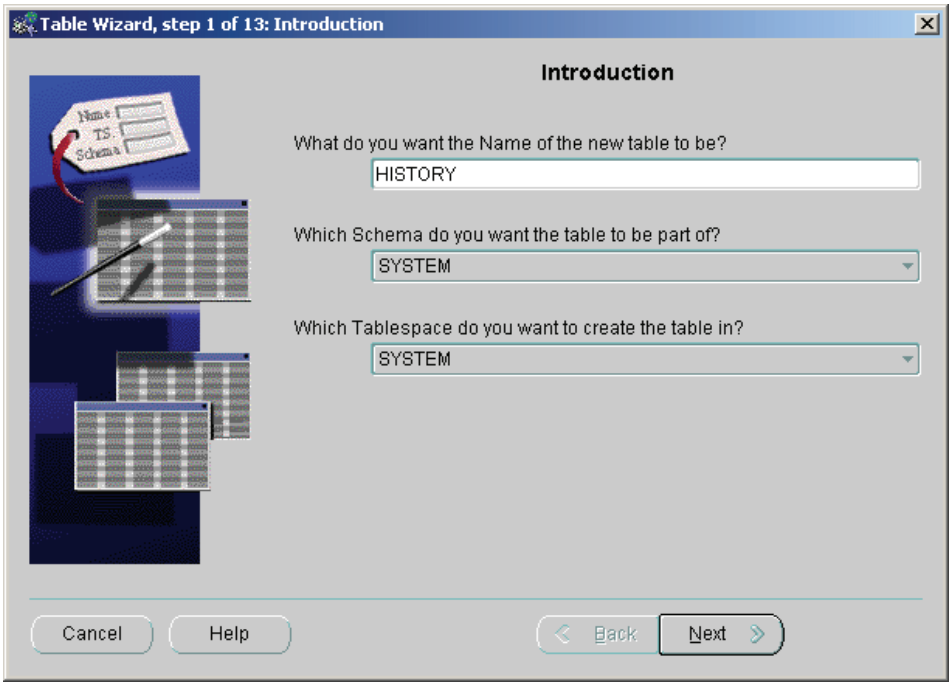


Figure B-27 Specify table name

6. Specify all the columns in the table. Click **Add** for each column to bring it to the Columns defined list. When you finish, click **Next**.

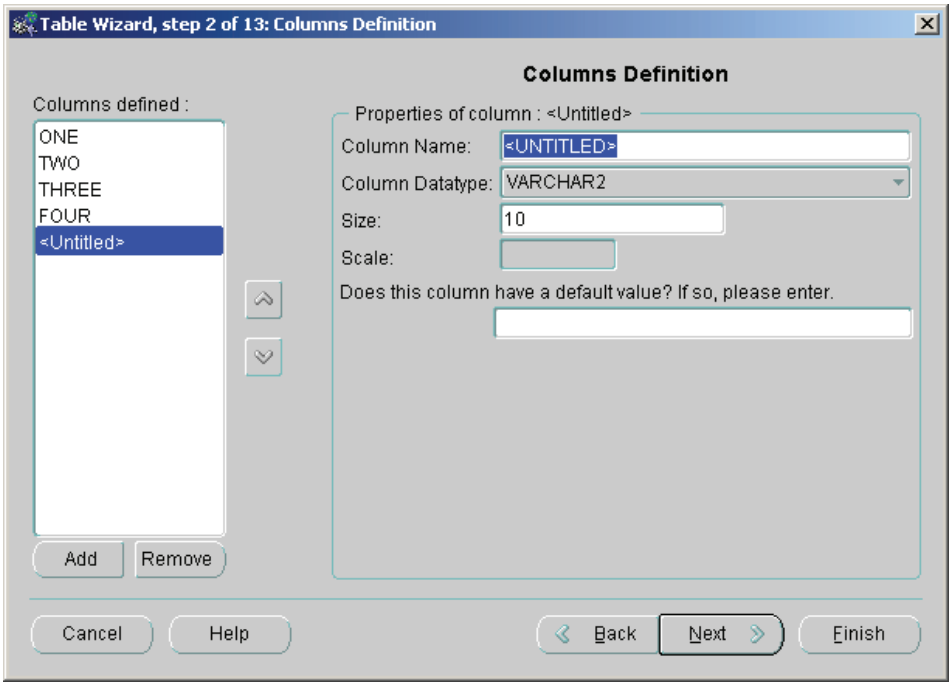


Figure B-28 Add columns

7. Define primary key(s) by clicking in the Order column. Click **Next**.

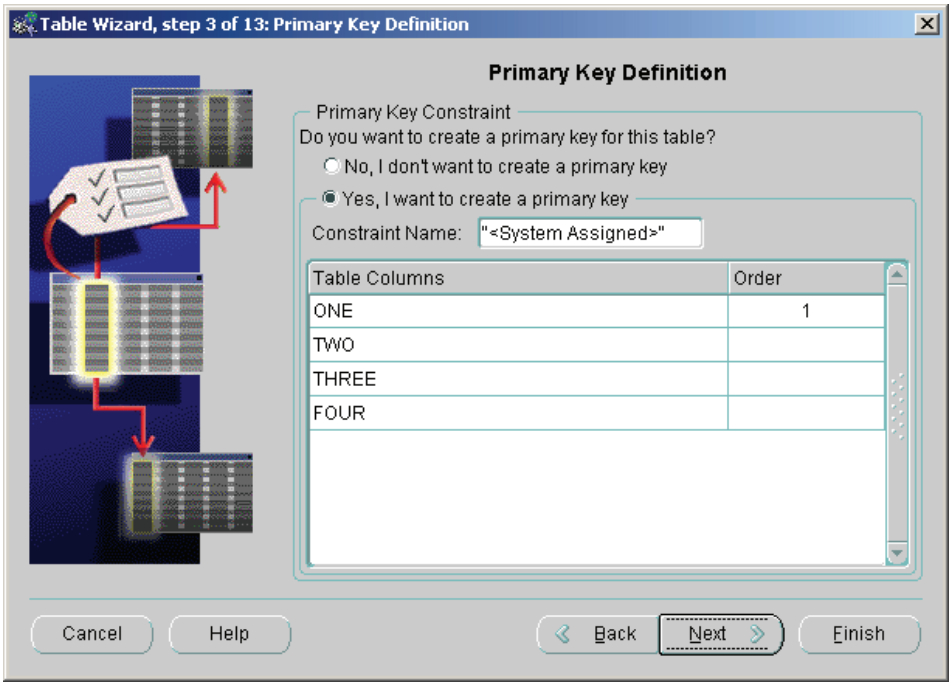


Figure B-29 Define primary key(s)

8. Specify for each column whether it is nullable and whether it must have unique values. In our example, all columns are not nullable and they do not have to have unique values. Click **Next**.

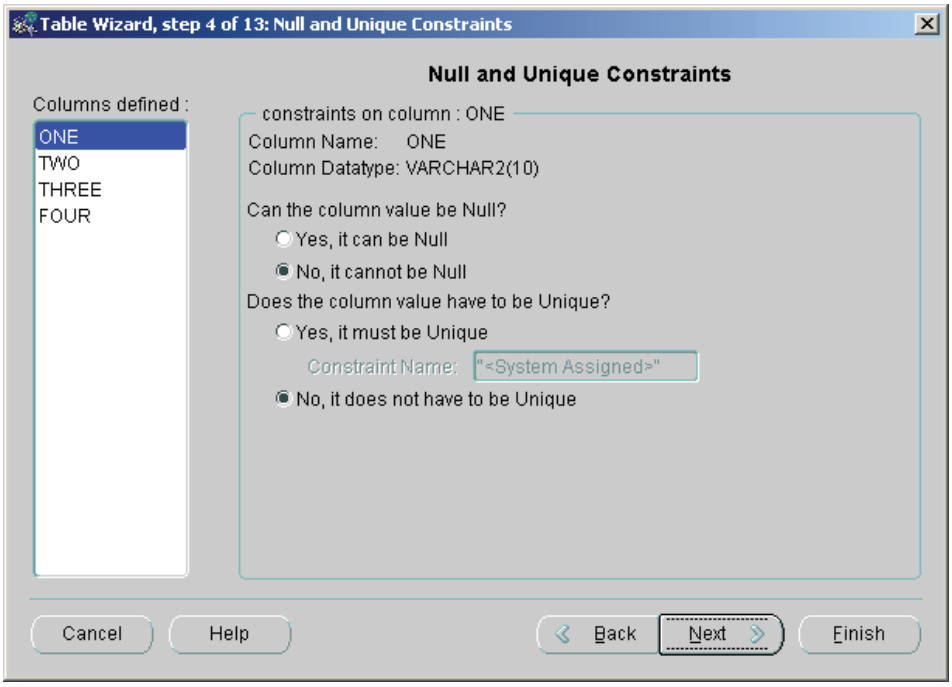


Figure B-30 Null and unique constraints

9. Specify for each column whether it is a foreign key. In our example, they are not. Click **Next**.

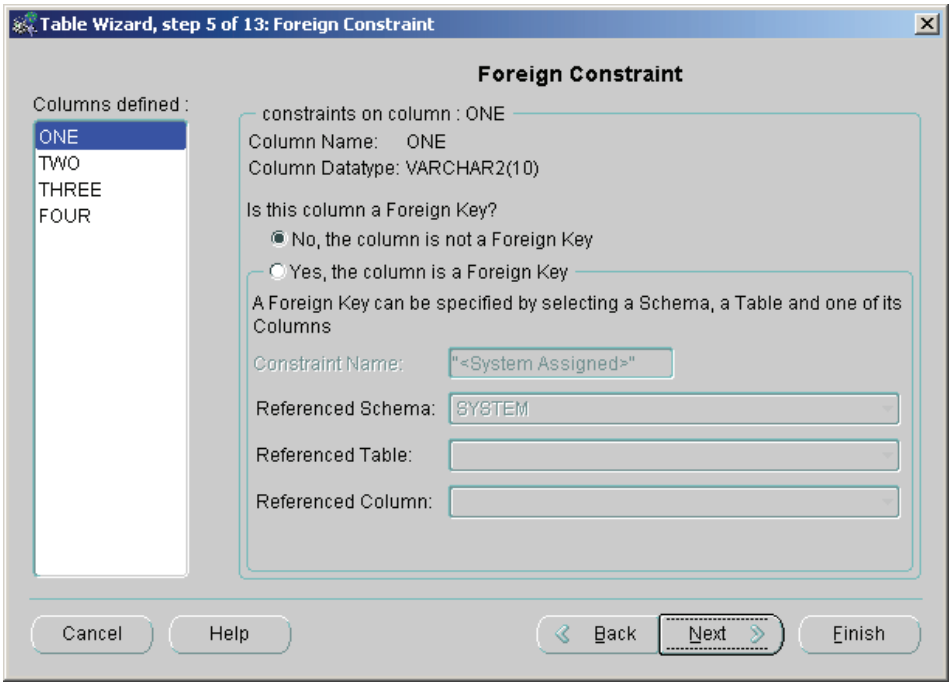


Figure B-31 Foreign constraint

10. Specify for each column whether it has check constraints. In our example, they do not. Click **Next**.

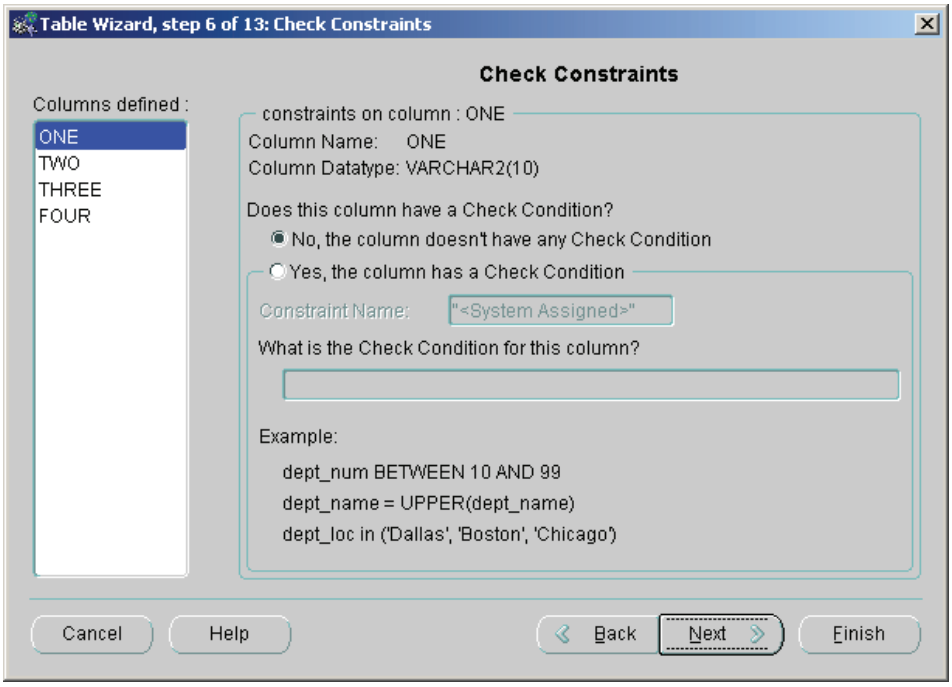


Figure B-32 Check constraints

11. Click **Next** in the Storage Information window.
12. All the necessary information has been collected, and a summary is shown. Click **Finish**.

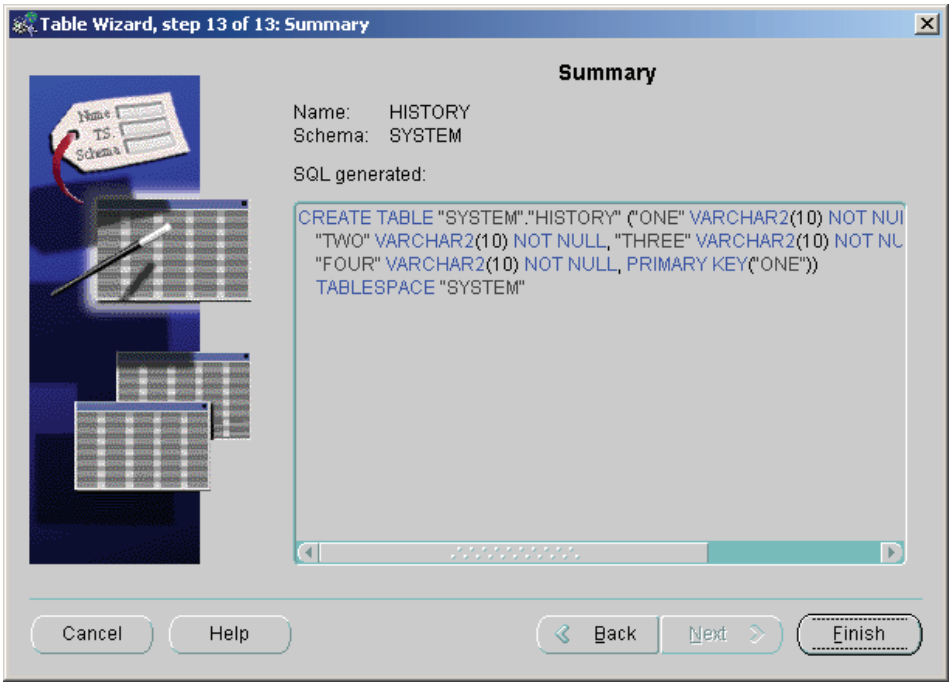


Figure B-33 Summary

13. The table is created. A success message is shown. Click **OK** to close it.

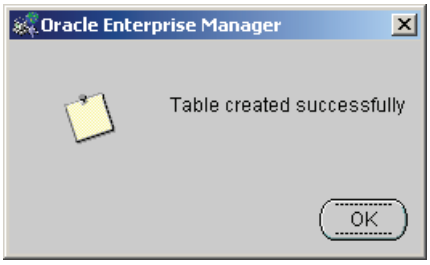


Figure B-34 Table created successfully

14. The newly created table should appear in Enterprise Manager now.

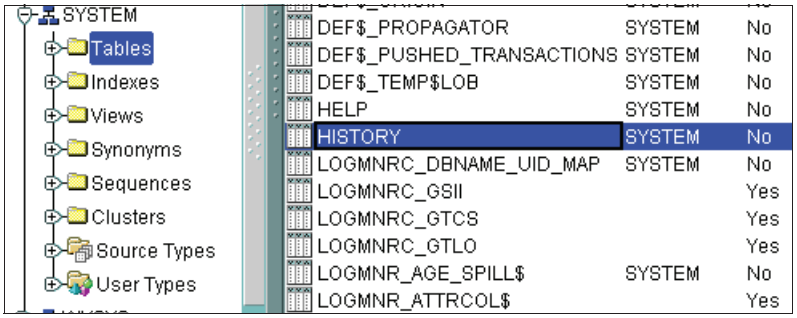


Figure B-35 Newly created table shown in Enterprise Manager

Populate table with data

There are various ways to populate the newly created table with data. You can use the SQLPlus Worksheet tool, or the more traditional SQLPlus. In addition, you can also use a Java program using the JDBC API to populate database tables.

For more information about the tools and the database itself, visit the Oracle Technology Network site at `http://otn.oracle.com`.

C

Palm Simulator installation

This appendix describes the following:

- Palm Simulator installation and configuration
- Usage to access WebSphere Everyplace Access

Palm Simulator is used to simulate Palm devices with the Palm OS 5.X operating system.

© Copyright IBM Corp. 2003. All rights reserved.

Overview

Palm Simulator simulates a Palm handheld device on a desktop computer, and lets you test and debug your Palm OS application within the simulated environment. Simulator provides a graphical representation of a Palm handheld device on the desktop screen, and supports user interactions that mimic actual stylus actions on a handheld device.

For more details about the Palm Simulator software, visit PalmSource at http://www.palmos.com/dev/tools/simulator/

Installation

To install the Palm Simulator software, follow these steps:

1. Download the Palm Simulator software for Palm OS 5.2 from http://www.palmos.com/dev/tools/simulator/. You can download the release version of the debug version that comes with the documentation.
2. Once you have downloaded the file, you must unzip it to an appropriate directory.
3. To start the Simulator, run the PalmSim.exe file located in the installation directory.
4. The program shows you a window with the license agreement. If you want use the product, you must click **I agree**.
5. Select the appropriate ROM file, in this case **NTFull_enUS.rom**, and click **Open**.

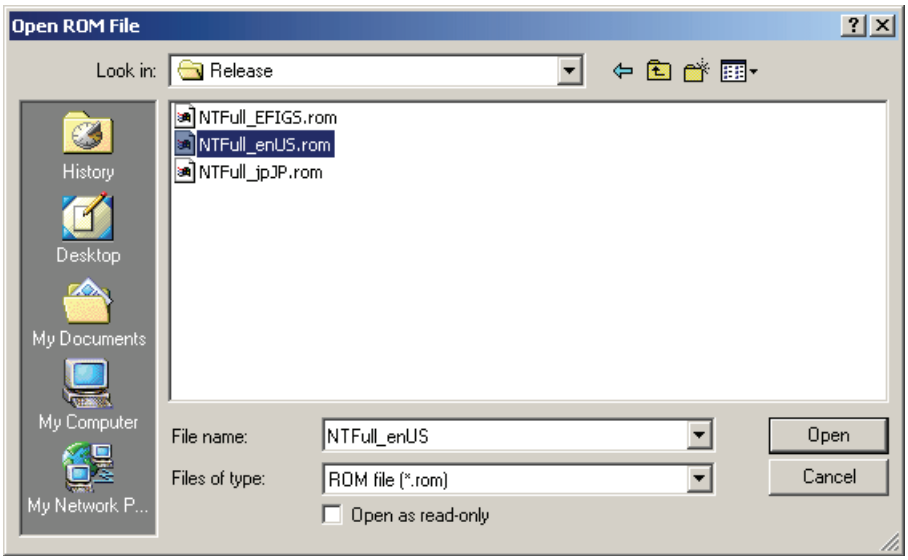


Figure C-1 Select the appropriate ROM

The Simulator screen looks like the window shown in Figure C-2.



Figure C-2 Palm Simulator for Palm OS 5

Configuration

You need to perform some configuration tasks before using the Simulator software. Follow these steps:

1. Right-click the Emulator window and select **Settings -> Communication -> Redirect NetLibs Calls to Host TCP/IP**. This will redirect the device network calls to the host PC.
2. Now install the Web Browser 2.0 for testing the online features of WebSphere Everyplace Access. You can download the Web Browser SDK from `http://www.palmos.com/dev/tech/palmos5/webbrowser.html`. On this page select the **SDK 5.0** link, accept the license agreement, and select the **WebBrowser 2.0 SDK** to download. Once downloaded, extract the WebBrowser20.dll and WebBrowser20.prc files to the Simulator installation directory.
3. Right-click the **Simulator** window and select **Install -> Database** from the pop-up menu. Move to the Simulator installation directory, select the **WebBrowser20.prc** file, and click **Open**, as shown in Figure C-3.

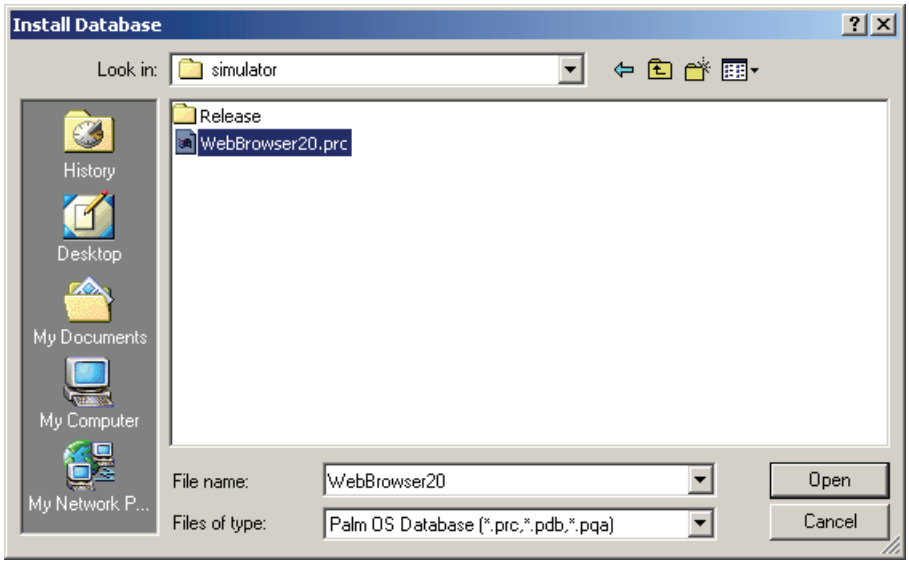


Figure C-3 Select the WebBrowser 2.0 application file

4. You will not see the WebBrowser icon yet. To see it, simply switch to any application (Calculator for example) and return to the Applications window.



Figure C-4 Palm Simulator with WebBrowser installed

Usage

So far, the configuration is sufficient to access online conten. To test the online access, follow these steps:

1. Click the **Web** icon on the Simulator and then click **Connect**.



Figure C-5 Connect to the Network

2. The default page opens. Click the title bar to display the menu and select the **Open URL** option.

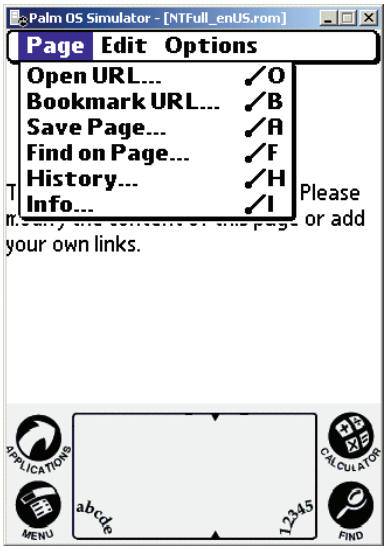


Figure C-6 Open a URL

3. Type the URL of the WebSphere Everyplace Access default page: `http://<wea_host>/wps/myportal`, where `<wea_host>` is the DNS name of the

machine where WebSphere Enterprise Addition has been installed, and click **Go**.

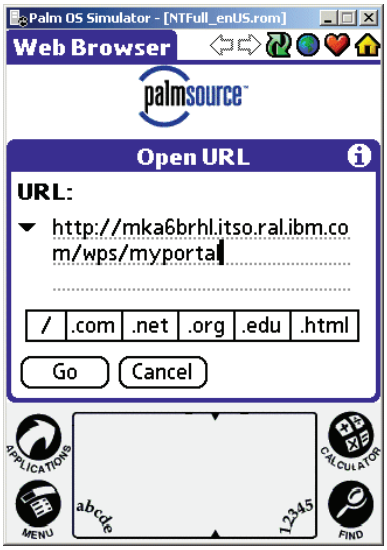


Figure C-7 Type the WebSphere Everyplace Access URL default page

4. Type the user ID and password for access to the WebSphere Everyplace Access server, for example `wpsadmin`. Click **Log in**.

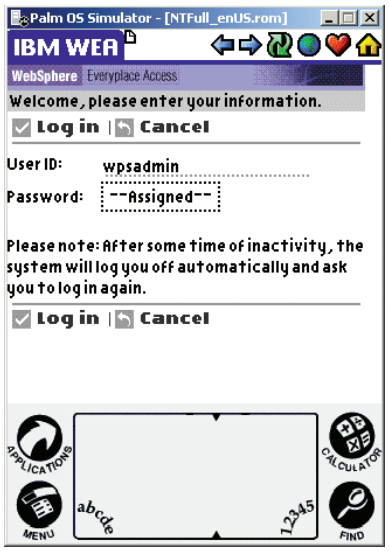


Figure C-8 Type your user ID and password

5. You will see the WebSphere Everyplace Access Welcome page.



Figure C-9 WebSphere Everyplace Access Welcome page

D

Palm Emulator installation

This appendix describes the installation and configuration of the Palm Emulator software. We use this software to emulate Palm devices that have Palm OS 3.X and 4.X installed.

© Copyright IBM Corp. 2003. All rights reserved.

Overview

The Palm OS Emulator is software that emulates the hardware of the Palm device. It uses the ROM image downloaded from the device to simulate the Palm OS running on a Palm device. In a sense, the Emulator is like a computer without an operating system and the ROM is like the operating system used to drive the computer.

Installation

To install the Palm OS Emulator, perform the following steps:

1. Download the Palm OS Emulator software from:

   `http://www.palmos.com/dev/tools/emulator/`

2. Unzip the downloaded file into a directory on your hard drive, for example C:\Palm Emulator.
3. Launch the Emulator application file Emulator.exe in the Palm Emulator directory. A message may appear saying the skins directory was not found. Click **OK** for now. We will install skins later. Follow the on-screen instructions to set up a short cut in the Start menu.
4. The Emulator is now installed. Click **Exit**.

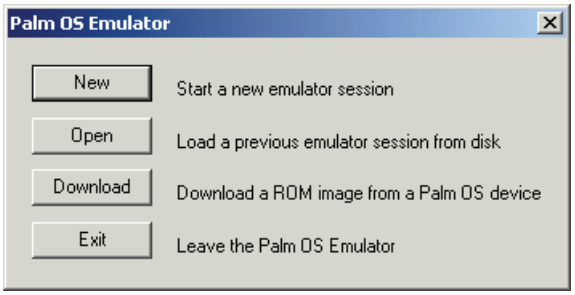


Figure D-1 Palm OS Emulator

Downloading ROM from Palm device

Before you can begin using the Emulator, you need a ROM image. A ROM image can be downloaded from the Palm Web site. Alternatively, it can also be downloaded from the Palm device:

1. There is a file in the Emulator directory called Rom Transfer.prc. This is a Palm OS application and needs to be installed on the Palm OS device, as follows:

   a. Run Palm Quick Install: Select **Start -> Programs -> Palm Desktop -> Palm Quick Install**.

   b. Click **Add** or **File -> Add Files** and select the **Rom Transfer.prc** file from the Palm Emulator directory.

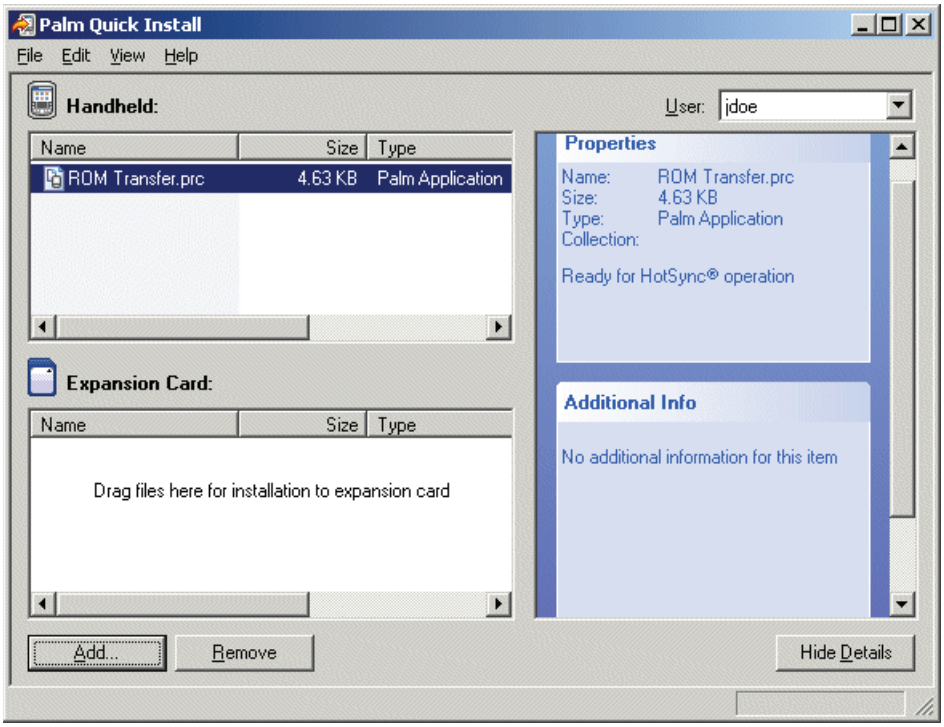


Figure D-2 Palm Quick Install

   c. Exit Palm Quick Install and click **OK** to any message.

   d. If the HotSync Manager is not already running (check the task bar for the HotSync Manager icon), launch it by selecting **Start -> Programs -> Palm Desktop -> HotSync Manager**.

   e. Right-click the **HotSync Manager** icon in the task bar and make sure **Local USB** is selected.

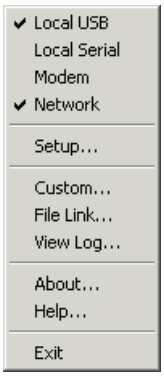


Figure D-3 Hot Sync - Select Local USB

f. Make sure the Palm is sitting in its cradle. Click the **HotSync** button on the cradle.

g. When the Users window appears, select the user name you entered when you installed Palm desktop software.

h. Wait for a message on your handheld indicating that the process is complete.

2. Copy the USBPort.dll file from the Palm Desktop directory into the Palm Emulator directory. This will allow you to select the USB option when transferring the ROM. If this step is not done, you will only be able to select COM 1 or COM 2.
3. Start Palm Emulator (select **Start -> Palm OS Emulator**) and click **Download**.
4. Under Port, select **USB**.

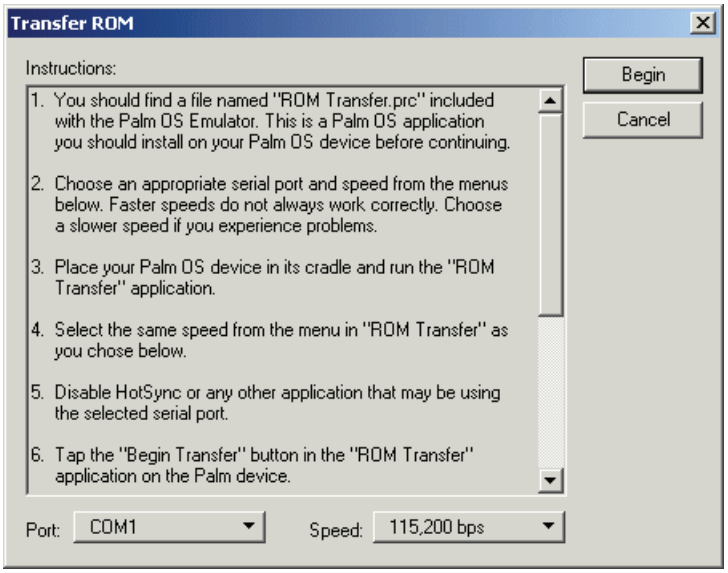


Figure D-4 Transfer ROM

5. Select the **Begin Transfer** button in the ROM Transfer application on the Palm device.
6. Click **Begin** on the Transfer ROM window of the Emulator software.
7. After the ROM has been transferred to your desktop computer, you will be asked to save it to a file. Save it with an appropriate name (for example, Palm m515.rom).
8. The ROM transfer is now complete.

Usage and skins

To begin using the Emulator with a ROM image, perform the following steps:

1. Start Palm Emulator (**Start -> Palm OS Emulator**) and click **New** to begin a new session.
2. Under ROM file, navigate to the saved ROM image and select it.
3. Under Device, select your Palm device. For this installation, we are using the Palm m515.
4. Click **OK**. You will now see an emulated Palm device.

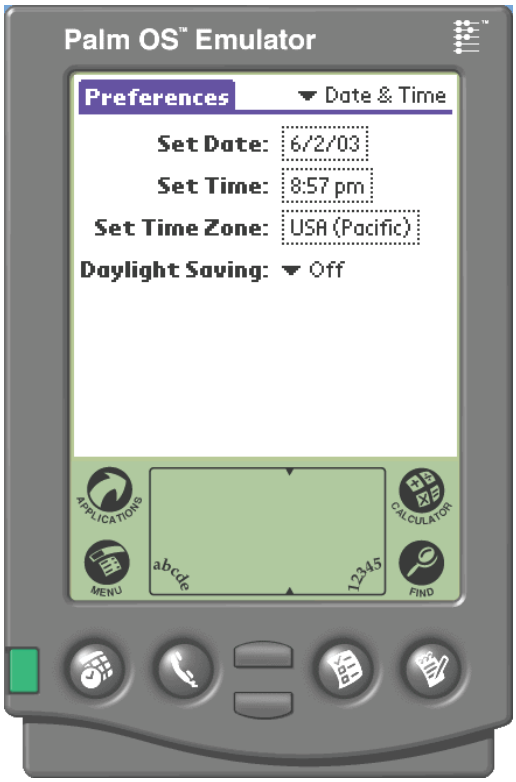


Figure D-5 Palm OS Emulator session with generic skin

5. You will notice that the Palm device in the Emulator session does not look like the actual device. To emulate the look of the actual device, we need to install the Emulator skins:

   a. Close the current Emulator session. Right-click the **Emulator** window and select **Close -> No -> Exit**.

   b. Download the Emulator skins from:

      `http://www.palmos.com/dev/tools/emulator/`

   c. Unzip the downloaded file to the Palm Emulator directory. The files will be extracted to a directory called Skins_v1.9 under the Palm Emulator directory. We need to rename that directory to skins. If this is not done, the Emulator will not be able to find the skins.

   d. Start Palm Emulator (select **Start -> Palm OS Emulator**). If it brings up the previous Emulator session, right-click the **Emulator** window and select **Close -> No**. Click **New** to begin a new session.

e. It should remember the previous settings for ROM file and device. In the Skin field, select **Standard-English**.

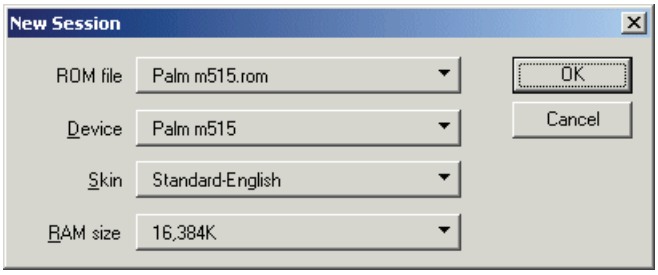


Figure D-6 Change Emulator skin

f. Click **OK**. You should now see an emulated Palm device that looks like the selected device. In our case, this is the Palm m515.



Figure D-7 Palm OS Emulator with Palm m515 skin

g. We now need to save the session. Right-click the **Emulator** window, select **Save** and enter an appropriate session name, such as Palm m515.psf.

h. The next time you start up the Palm Emulator, open this saved session instead of creating a new one.

E

Palm Desktop installation

This appendix describes the installation of the Palm Desktop software. This software provides syncrhonization capabilities with the desktop and it is required for the installation of the Everyplace Client.

© Copyright IBM Corp. 2003. All rights reserved.

Overview

Palm Desktop is a desktop companion to the Palm device. It includes the same applications as your handheld, allowing you to quickly add or update contacts, appointments, to-dos and memos at your desktop. Using the HotSync Manager, you can then synchronize the data in your Palm device with the data in Palm Desktop.

Palm Desktop is a component required for the Everyplace Client installation on Palm devices.

Installation

To install Palm Desktop, perform the following steps:

1. Ensure the Palm device is not connected to the desktop PC. If you experience problems later in the installation, it might help to perform a hard reset of your Palm device and start from here again. However, this should only be done if absolutely necessary.
2. Download Palm Desktop 4.1 from:

   `http://www.palm.com/support/downloads/win_desktop.html`
3. Run the downloaded file. It will prompt for a setup directory to unpack the setup files into. For the purposes of this installation, we will just accept the default and click **Next**.

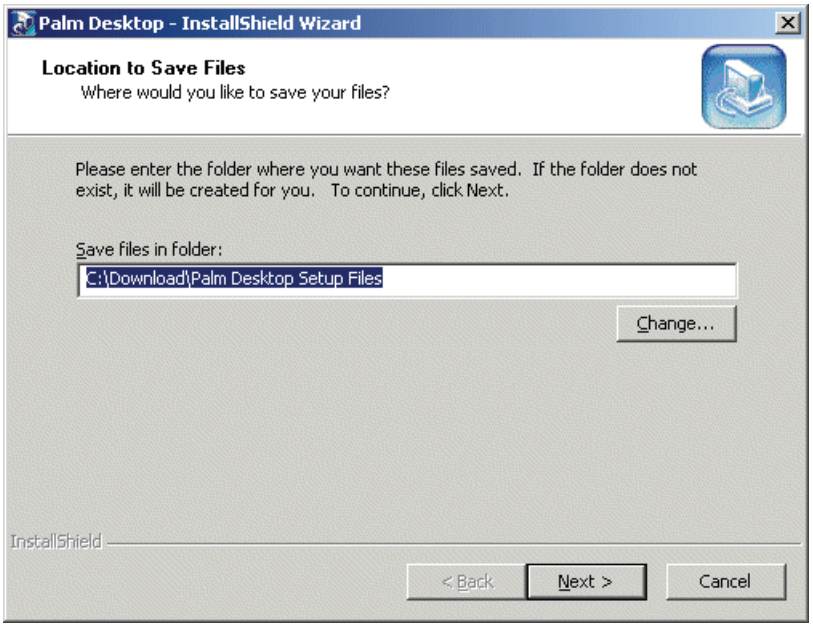


Figure E-1 Select the setup directory

4. Click **Next** on the welcoming window of the installation.
5. Accept the license agreement and click **Next**.
6. Select **Custom** to perform a customized installation, allowing you to select which features you want to install.

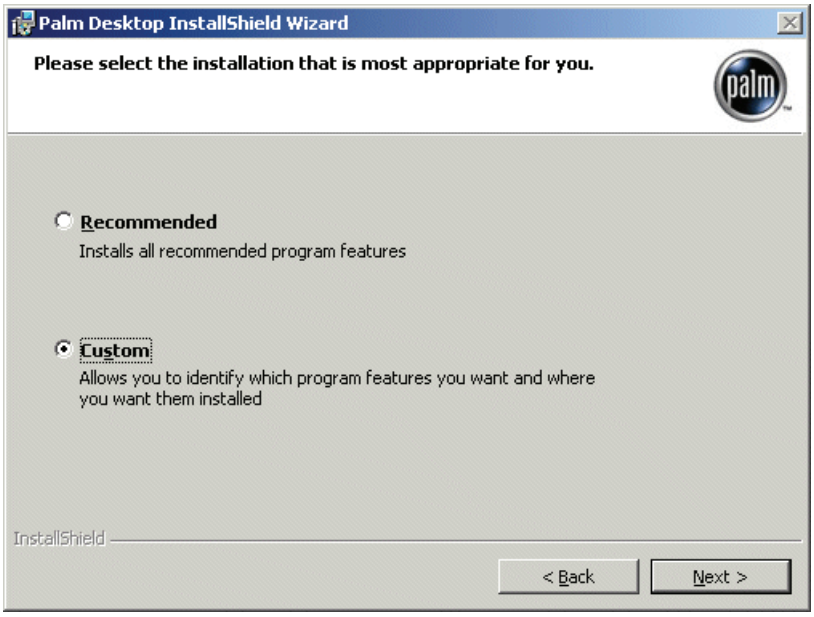


Figure E-2 Custom setup

7. On the Custom Setup window, select the features you want to install. By default, all features are installed. For the purposes of this installation, we will go with the default selection.

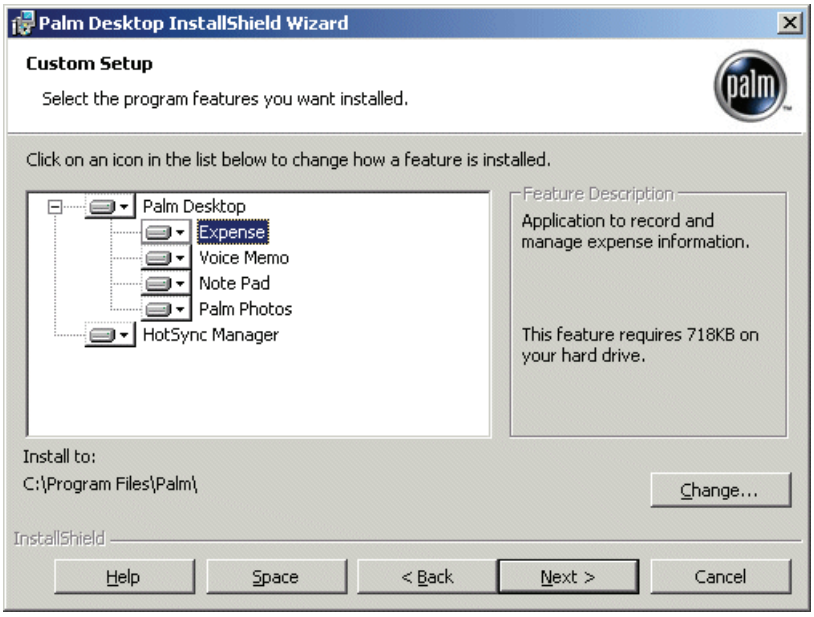


Figure E-3 Select features to install

8. You will need to enter a user name for your handheld. If you plan on sharing the Palm Desktop software with another handheld user, it is recommended that the user name for each handheld user have a unique user name. This will prevent complications and potential data loss during synchronization.

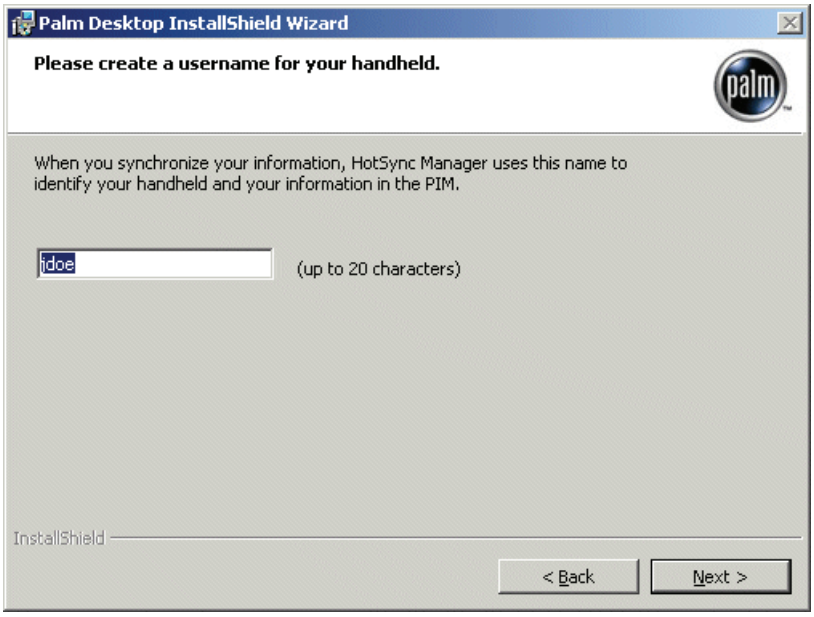


Figure E-4 Create user name for handheld

9. The next window will ask you to connect your cradle/cable to the USB or serial port on your desktop PC. For the purpose sof this installation, we will select **USB**. Connect your device cradle to the PC's USB port and click **Next**.

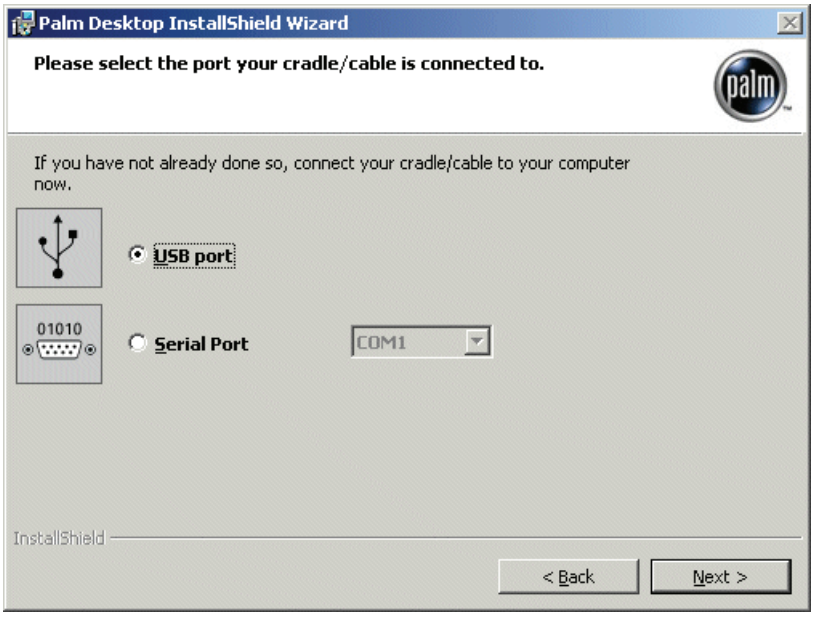


Figure E-5 Select cable port

10. Click **Install** to begin the installation.
11. Click **Finish** when the installation completes.

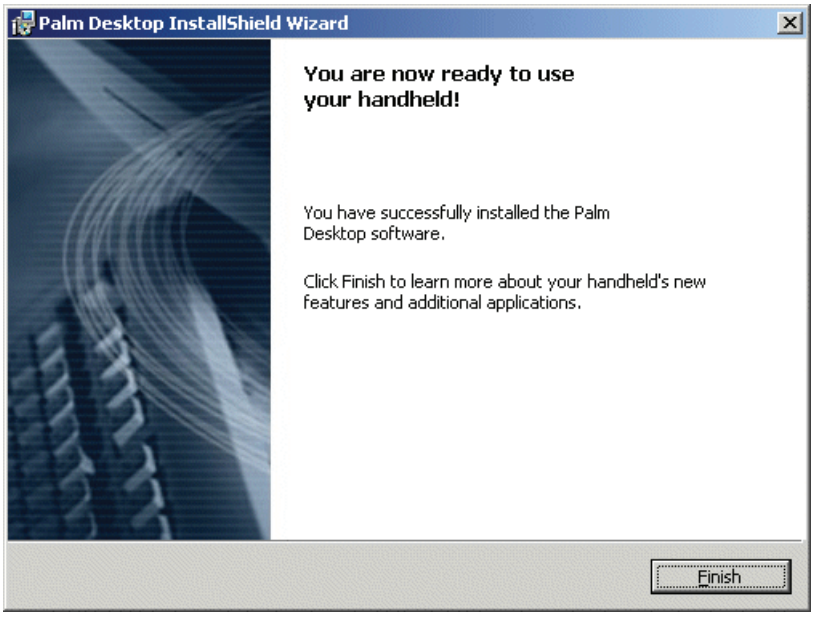


Figure E-6 Installation complete

12. It will now ask if you want to restart the system for the changes to take effect. Click **Yes** and let the system restart.

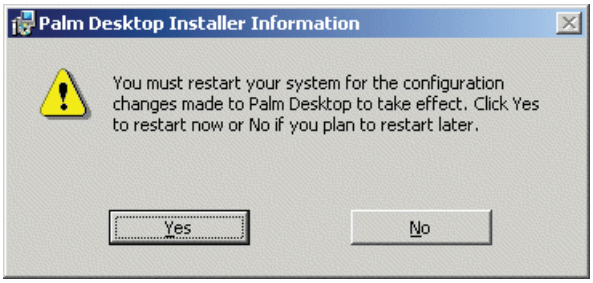


Figure E-7 Restart computer

Now you can use Palm Desktop and HotSync to synchronize your desktop with your device.

F

Pocket PC development tools

This appendix describes the following tools for Pocket PC development:

- Microsoft ActiveSync
- Microsoft Remote Display Control

© Copyright IBM Corp. 2003. All rights reserved.

Overview

For development on Pocket PC devices, we have used several tools. In this appendix we describe two of them. ActiveSync is required to connect the Pocket PC to the desktop and, using the pass-through feature, to the network. Also you may find the Remote Display Control useful for capturing screens and using your desktop keyboard with your Pocket PC.

Microsoft ActiveSync 3.7

Microsoft ActiveSync is a synchronization tool for the Pocket PC that performs automatic and continuous updates without the need to explicitly invoke the synchronization. It also provides direct and seamless connectivity to the Internet via your desktop PC, allowing you to perform such tasks as browsing the Web, checking e-mail, and sending instant messages from your Pocket PC.

Installation

To install ActiveSync, perform the following steps:

1. Ensure the Pocket PC is not connected to the desktop PC. If you experience problems later in the installation, it might help to perform a hard reset of your Palm device and start from here again. However, this should only be done if absolutely necessary.
2. Download Microsoft ActiveSync 3.7 from:

   `http://www.microsoft.com/mobile/pocketpc/downloads/activesync37.asp`
3. Run the downloaded file MSASYNC.exe.
4. Click **Next**.
5. Accept the default installation folder by clicking **Next**.

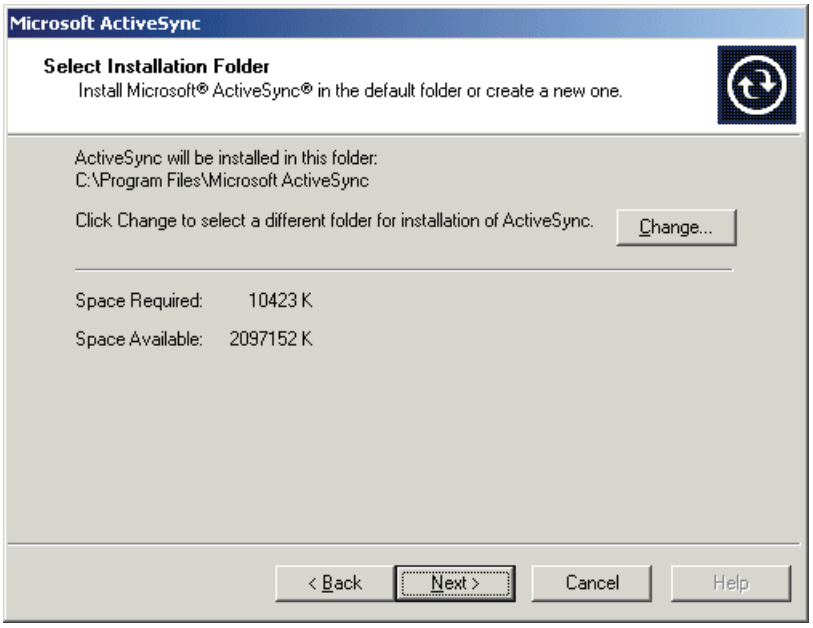


Figure F-1 Select installation folder

6. Once the setup program finishes copying files to the ActiveSync install directory, a window will be displayed prompting you to connect your Pocket PC to the desktop PC. Place your Pocket PC in the cradle and connect the USB cable from the cradle into the USB port on the desktop PC.

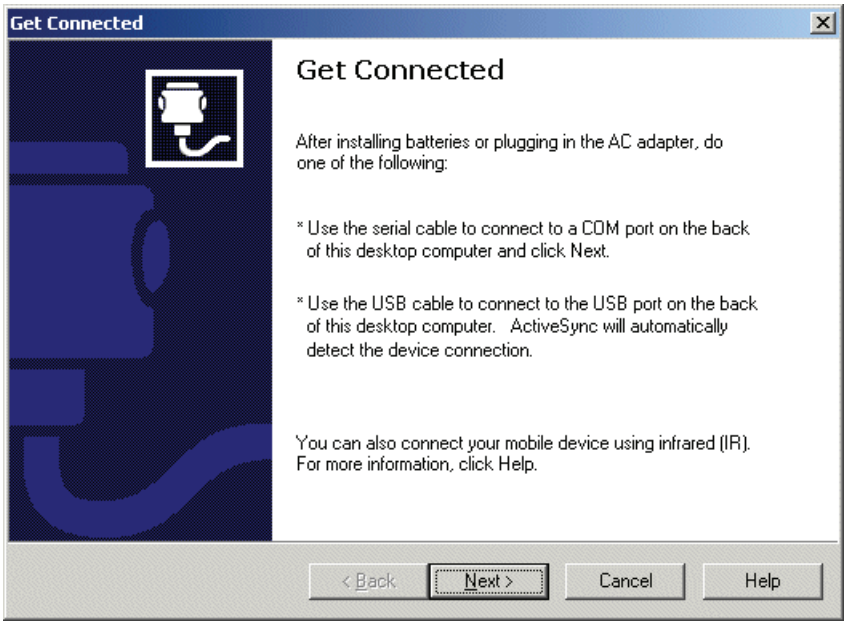


Figure F-2 Connect Pocket PC to desktop

7. Once client device is connected to the desktop, Windows 2000 should automatically detect the device. A window will be displayed asking you to set up a partnership. Select **Yes** if you would like to synchronize information between your device and the desktop. Select **No** if you only want to use the device as a guest to copy and move information between your device and the desktop. For the purpose of this installation, we will select **Yes**.

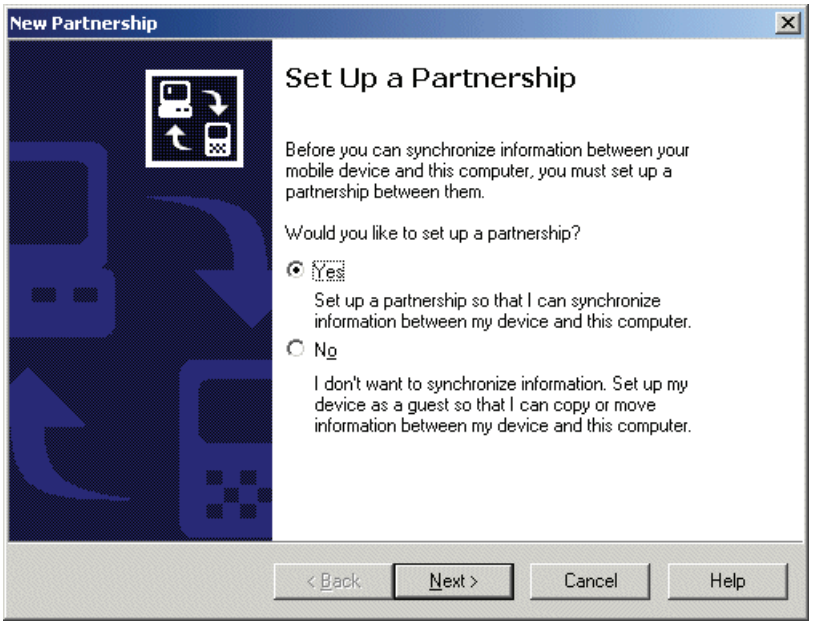


Figure F-3 Set up partnership

8. On the Select Synchronization Settings window, select the type of information you wish to synchronize. For the purposes of this installation, we will only select **Favorites**.

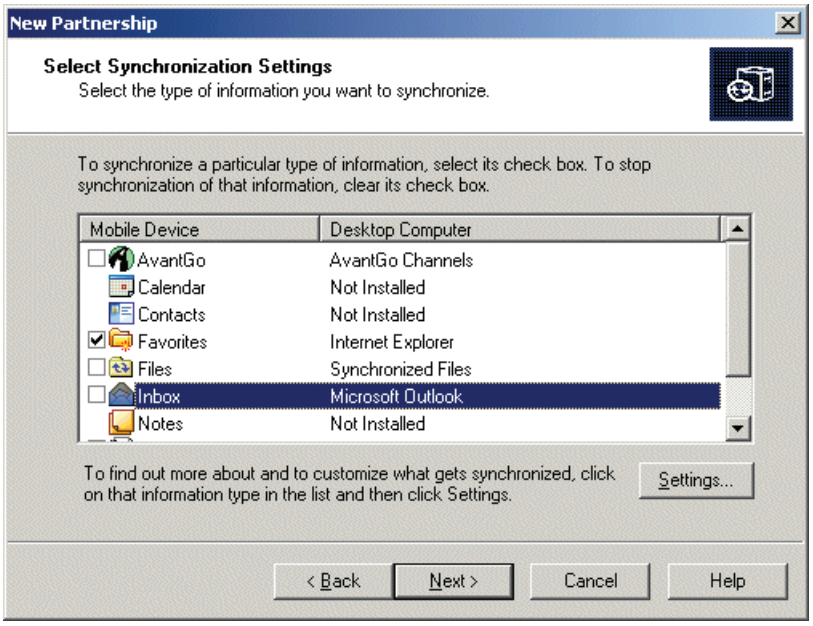


Figure F-4 Select components to synchronize

9. You should now arrive at the Setup Complete window. Click **Finish**.
10. Microsoft ActiveSync will run and synchronize the information on your device with the information on your desktop. Once the synchronization is complete, you can close the window. ActiveSync will not shut down, but will remain minimized in the Windows task bar.

Microsoft Remote Display Control

Microsoft Remote Display Control is a tool that allows you to display and control your Pocket PC remotely on your desktop or laptop. This can be a useful tool for developers wishing to test or demonstrate their Pocket PC applications on a larger screen.

Installation

To install Remote Display Control, perform the following steps:

1. Ensure your Pocket PC is placed on the cradle and connected to the desktop via the USB cable.
2. Download Remote Display Control from:

http://www.microsoft.com/mobile/pocketpc/downloads/powertoys.asp

3. Run the downloaded file RemoteDSP.exe.
4. Click **Next**.
5. Accept the default destination folder by clicking **Next**.

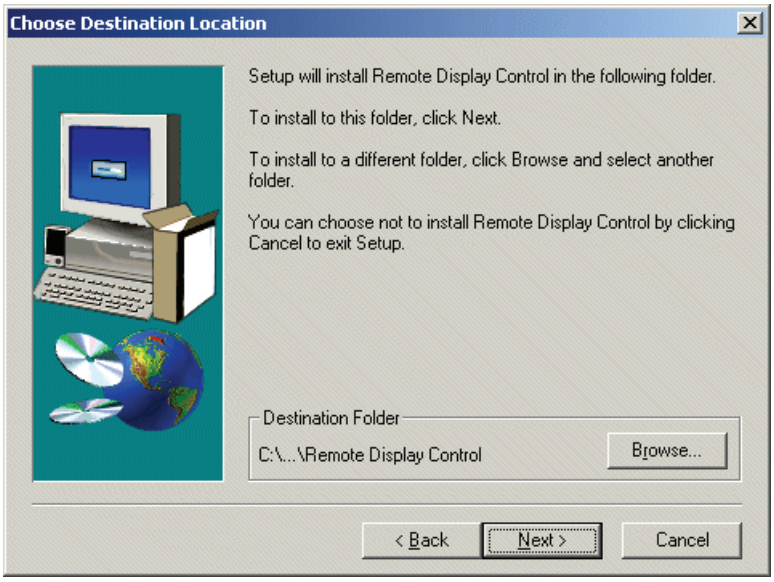


Figure F-5 Select destination folder for Remote Display Control

6. Click **Yes** on the next window.

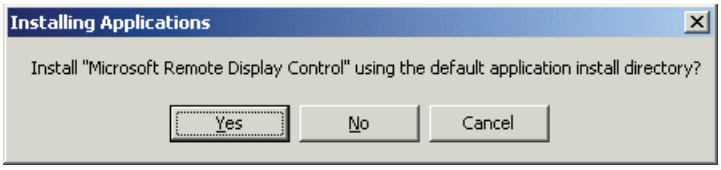


Figure F-6 Install using default application install directory prompt

7. Click **OK**.



Figure F-7 Check mobile device for additional steps

8. The Setup Complete window will appear. Click **Finish**.

Usage

To use Remote Display Control:

1. On the desktop PC, launch the Remote Display Control application by selecting **Start -> Programs -> Remote Display Control -> Remote Display Control Host**.
2. A window will be displayed with nothing inside.

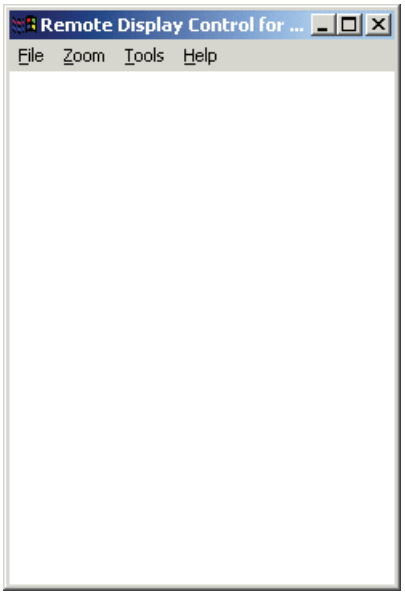


Figure F-8 Empty Remote Display Control window

3. On the Pocket PC, select **Start -> Programs** and launch **cerdisp**. Click **OK** on the About CERDisp window.
4. Click **Connect** and enter the host name of your desktop PC.
5. The screen on your Pocket PC will now be remotely displayed on the window that appeared in step 2. Using your desktop mouse and keyboard, you can also control your Pocket PC through this window.

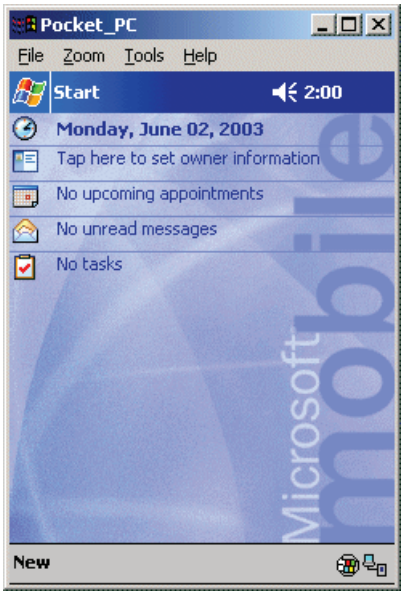


Figure F-9 Remote Display Control window

G

Portlet development platform installation

This appendix describes a sample installation and the steps necessary to prepare a single machine running Windows 2000 Professional or Server to be used for WebSphere application development and testing, including:

- Installation of DB2 V7.2 and FP7
- Installation of the WebSphere Studio Site Developer V5.0
- Installation of WebSphere Application Server Single Server V4.0 FP4
- Installation of the Everyplace Toolkit for WebSphere Studio Site Developer

© Copyright IBM Corp. 2003. All rights reserved.

G.1 Prerequisites

In this section, we list the prerequisites for installing the Everyplace toolkit.

Minimum machine requirements

- 1 GHz Pentium III or higher
- 768 MB RAM/1 GB recommended
- 10 GB hard drive, LAN attached or Loopback support
- TCP/IP stack must be installed

Software requirements

- Windows 2000 Professional or Server and Service Pack 3
- IBM DB2 V7.2 FP 7
- WebSphere Studio Site Developer V5.0
- WebSphere Application Server Single Server V4.0 FP4
- WebSphere Everyplace Access/Portal Toolkit V4.3 for WebSphere Studio Site Developer
- WebSphere Portal V4.2

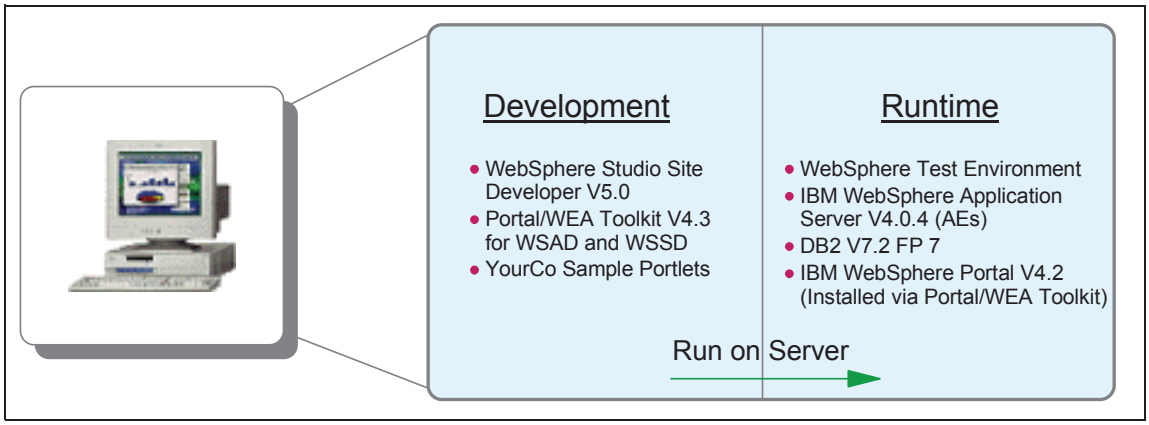


Figure G-1 Portlet development workstation

G.2 Windows configuration (before installation)

It is necessary to grant several essential permissions to the administrator or other user ID, for example db2admin, who will perform the installation. These are:

- Act as part of the operating system
- Create a token object
- Increase quotas
- Replace a process level token

1. Open the Control Panel by clicking **Start -> Settings -> Control Panel -> Administrative Tools -> Local Security Policy**.
2. Select **Local Policies -> User Rights Assignment.** You should see a window like that in Figure G-2.

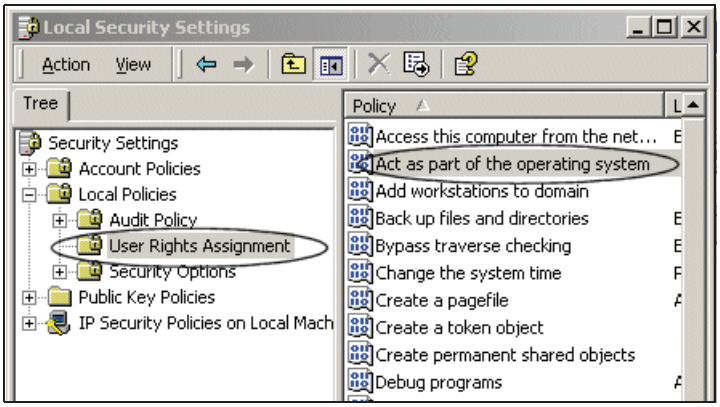


Figure G-2 Selection of installation user rights

3. Double-click **Act as part of the operating system**. You will see a window like Figure G-3 on page 1214.

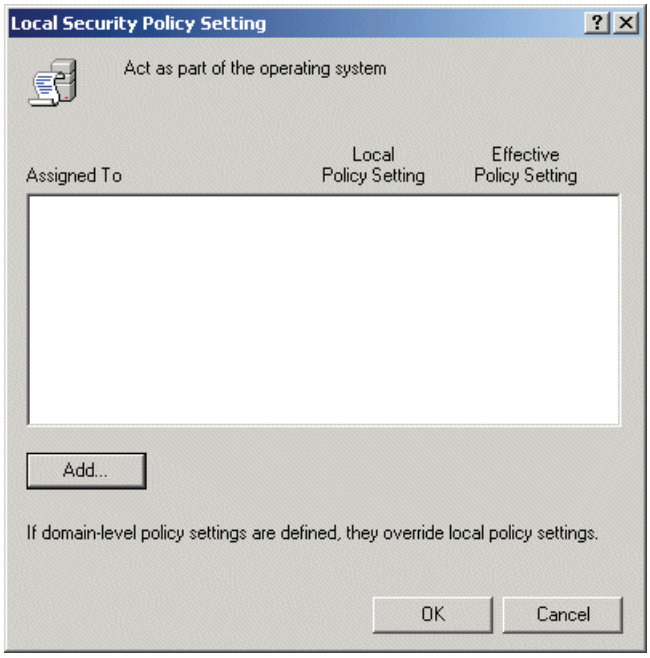


Figure G-3 Administration window for setting local security policy

4. Click the **Add** button and you will see a window like that in Figure G-4 on page 1215.

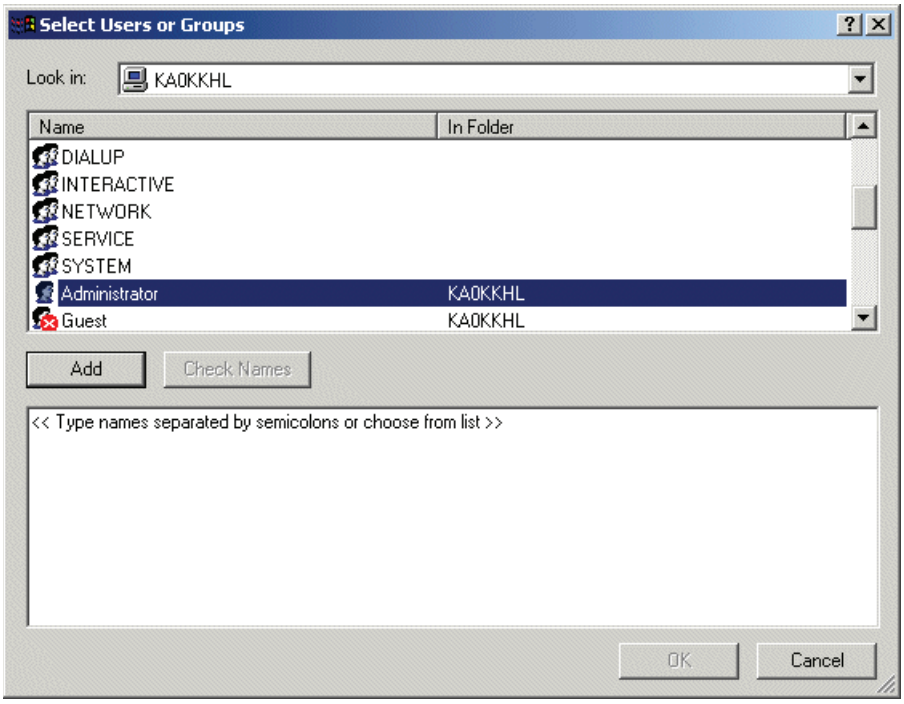


Figure G-4 Local security policy selection window for users and groups

5. Select the name of the user whose ID will be used to perform the installation. In this example, we are using Administrator. Click the **Add** button. You should now see something like Figure G-5 on page 1216.

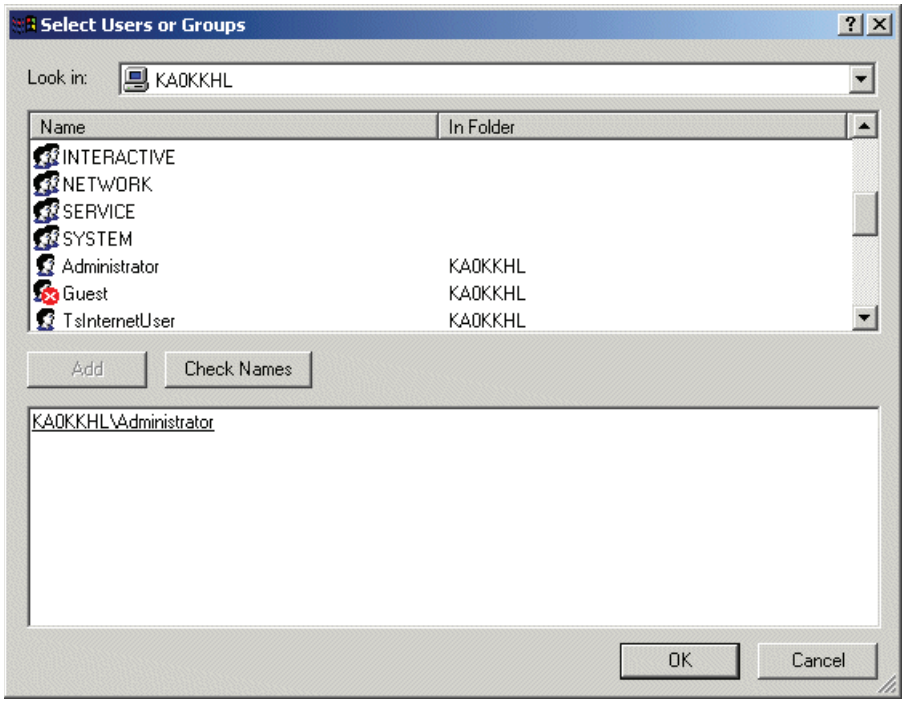


Figure G-5 Selected user shown in window

6. Click **OK**. You should now see a window like Figure G-6 on page 1217.

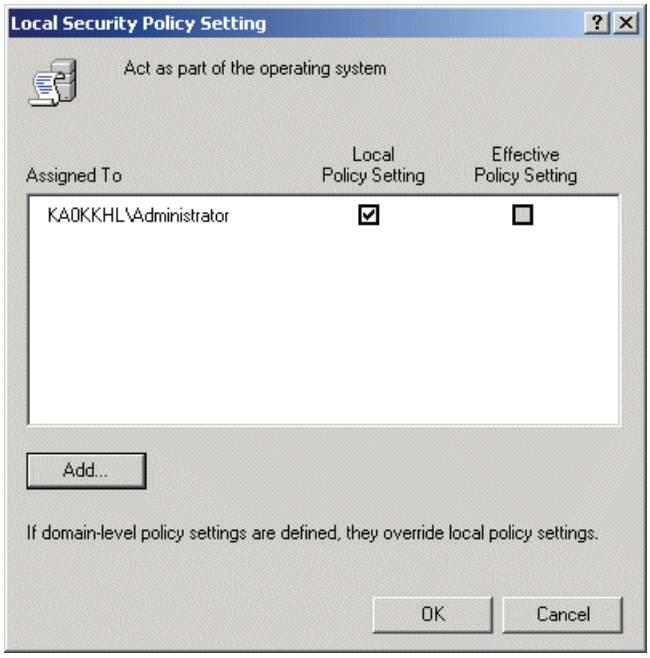


Figure G-6 Local security policy window after assigning Administrator rights

7. Click **OK** to close the window.
8. Repeat steps 1 through 8 for permissions to:
   - Create a token object
   - Increase quotas
   - Replace a process level token

   All these permissions are circled in the Local Security Settings window shown in Figure G-7 on page 1218.

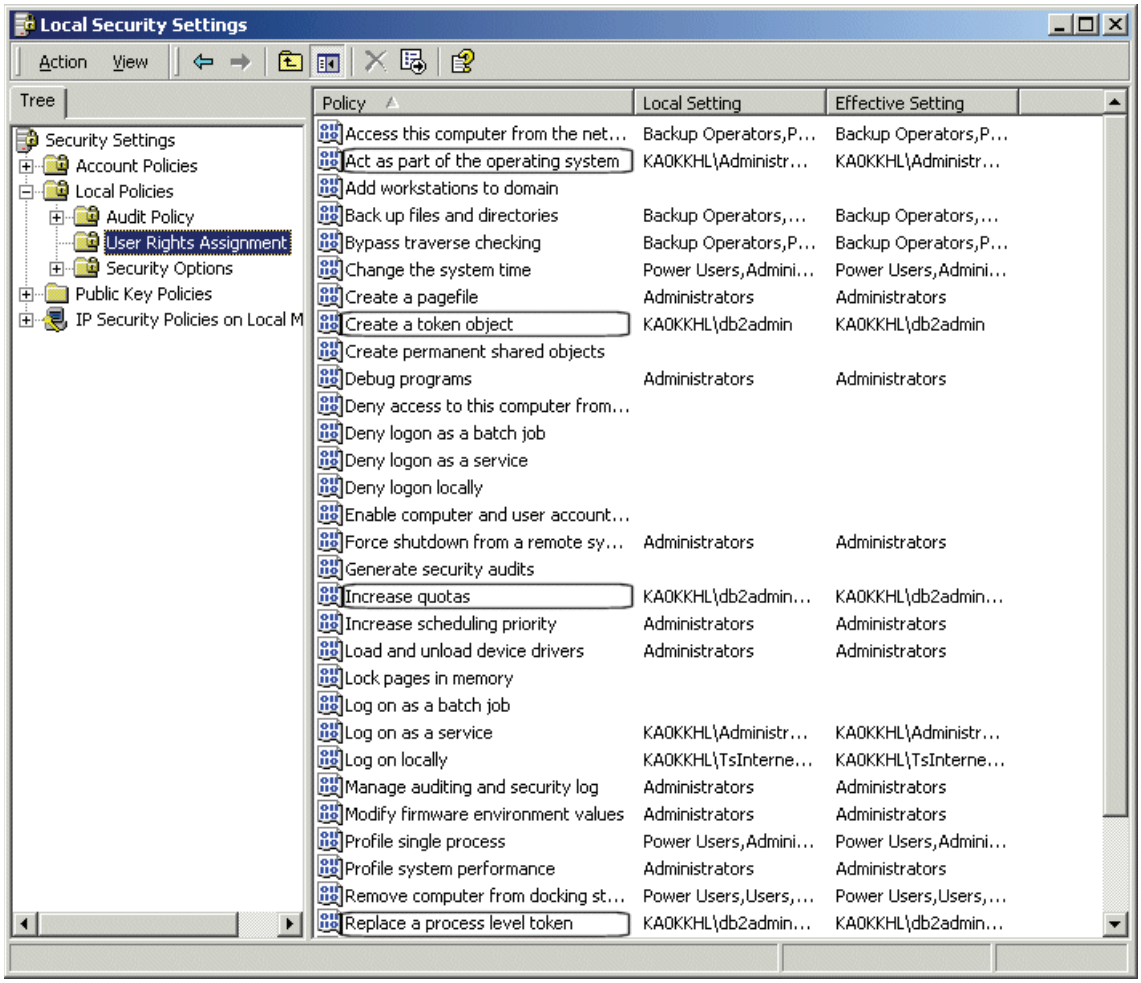


Figure G-7 User rights assignments in Local Policies

9. Close the window when you are done.

G.3 Configuring TCP/IP

Each development machine should have a static IP address assigned to it.

Prior to software installation, verify that you do have a static IP address by doing the following:

i. Click **Start -> Settings -> Control Panel**.

j. Double-click **Network and Dial-up Connections**.

k. Right-click your Ethernet connection and click **Properties**.

l. Double-click **Internet Protocol (TCP/IP)**.

m. Verify that **Use the following IP address** is selected and configured properly.

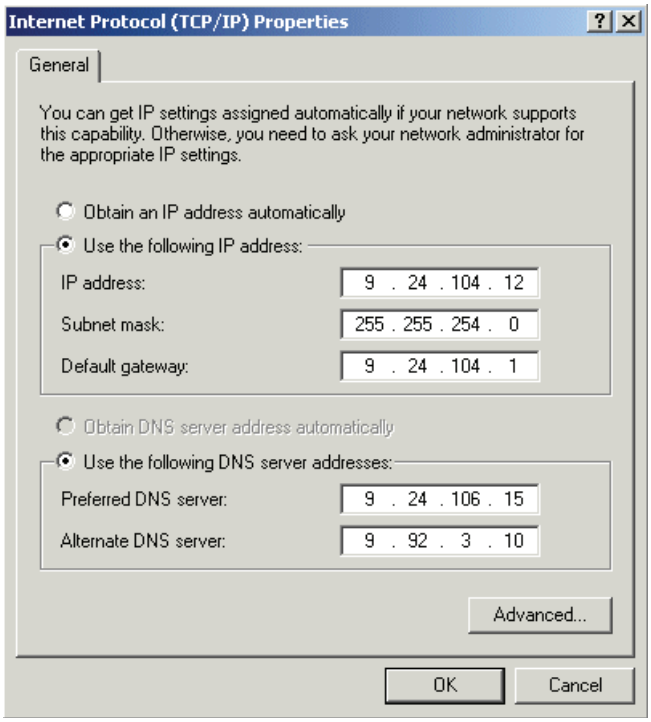


Figure G-8 TCP/IP configuration

G.4 DB2 installation

This section details the necessary steps to install DB2 V7.2 and Fix Pack 7.

G.4.1 DB2 installation

The following steps are necessary to install DB2.

1. Insert WebSphere Everyplace Access CD 2-1. Navigate to the \db2\win directory and double-click **setup.exe**. You will see the Welcome window as illustrated in Figure G-9. Click **Install** to continue.

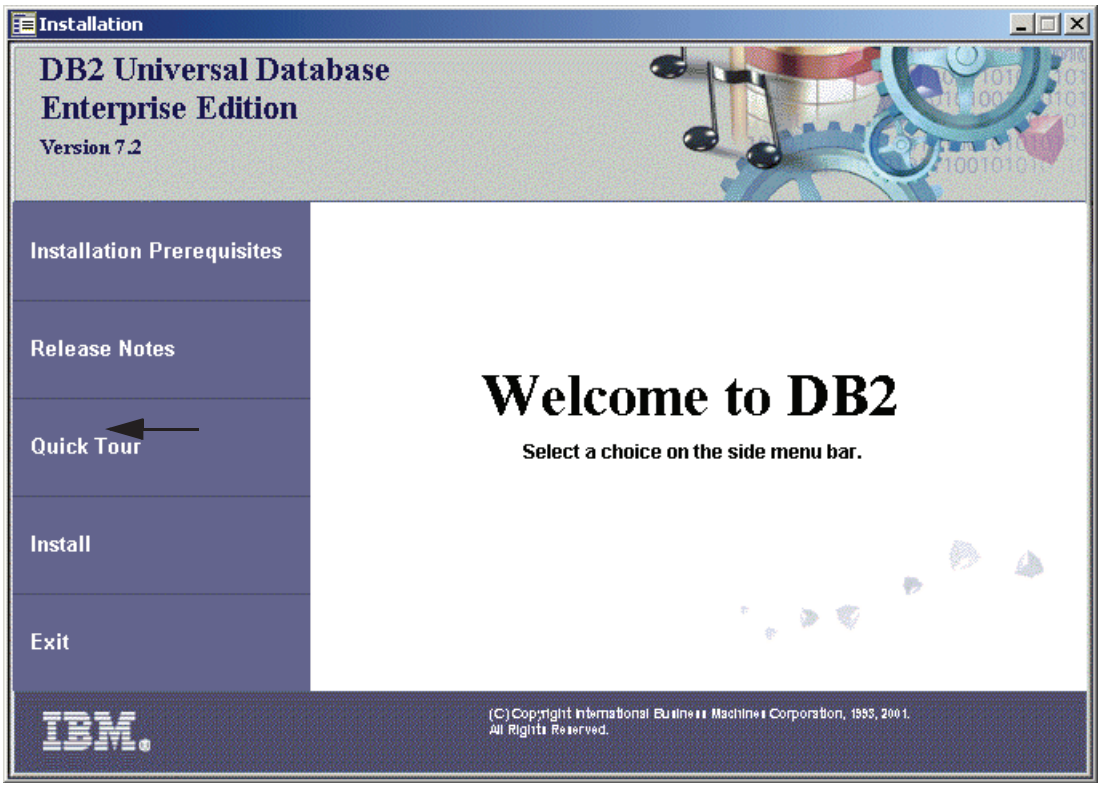


Figure G-9 DB2 Installation Welcome window

2. Accept the default product for installation, DB2 Enterprise Edition, as shown in Figure G-10 on page 1221. Click **Next**.

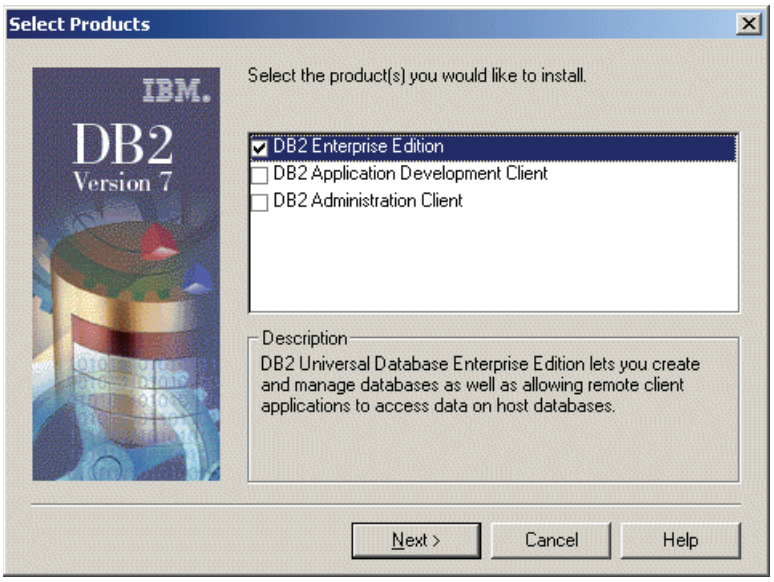


Figure G-10 DB2 product selection window

3. Accept the default installation type of Typical as shown in Figure G-11. Click **Next** to continue.

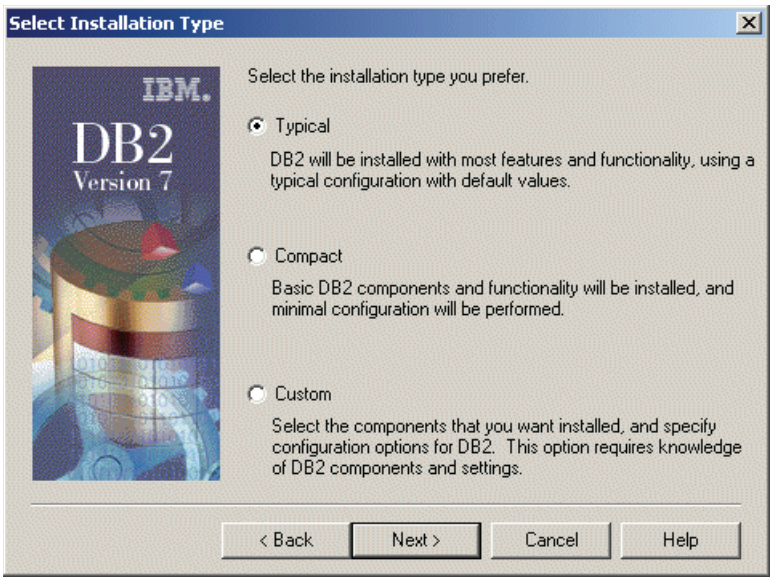


Figure G-11 DB2 Installation Type selection window

4. Accept the default installation folder of C:\Program Files\SQLLIB as shown in Figure G-12. Click **Next**.

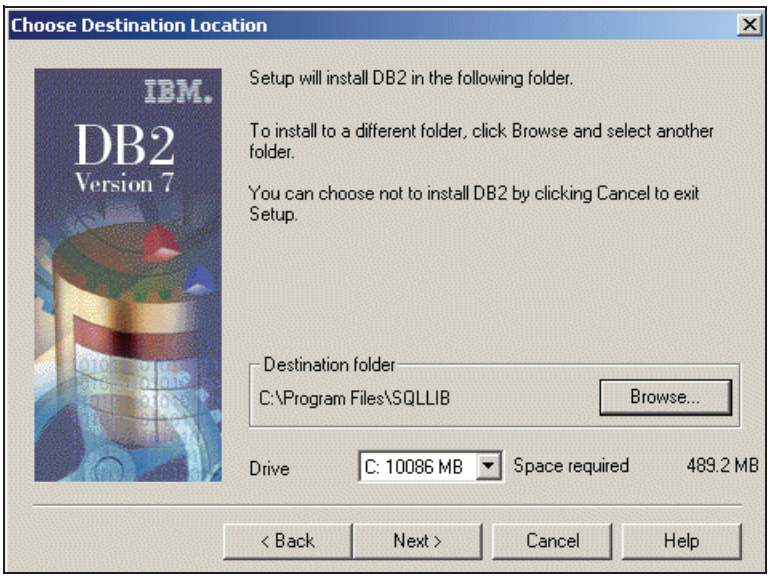


Figure G-12 DB2 installation folder selection

5. Enter `db2admin` for both the user name and password. Check **Use the same values for the remaining DB2 Username and Password settings**. Click **Next**.

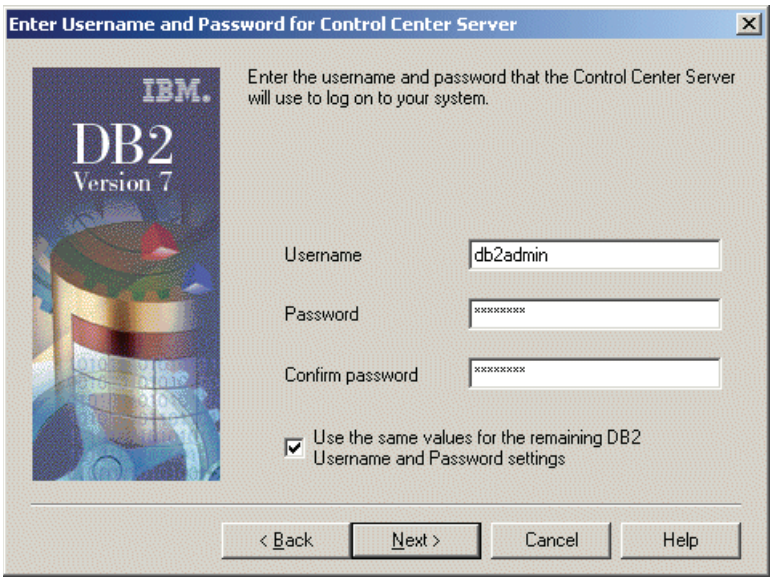


Figure G-13 Administrator information entry

6. If the window in Figure G-14 appears, click **Yes** to create a db2admin user ID.

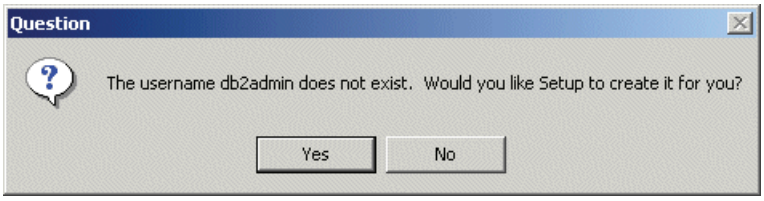


Figure G-14 Administrator creation query

7. Accept the final configuration as displayed in the window shown in Figure G-15 on page 1224. Click **Next**.

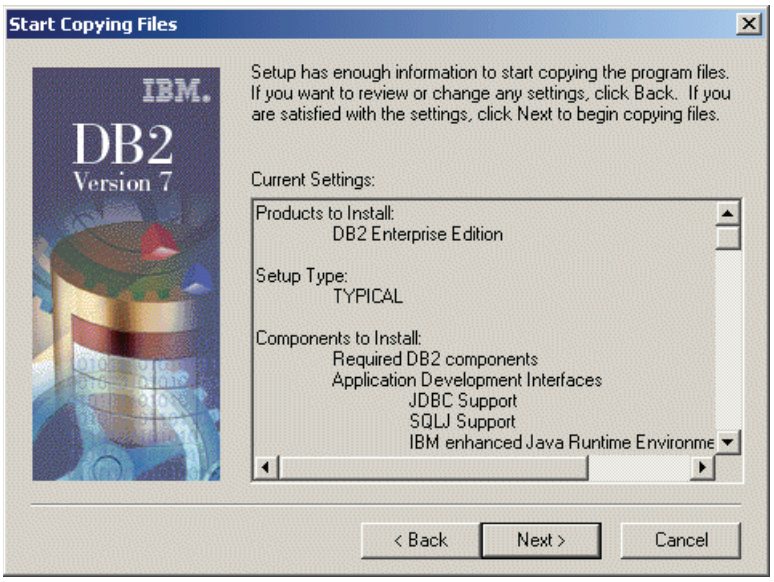


Figure G-15 Final configuration display window

While configuration is in progress, the window in Figure G-16 will be displayed.

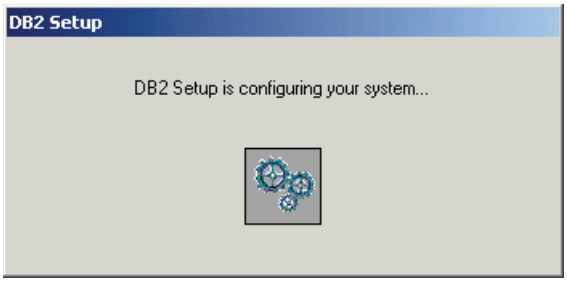


Figure G-16 Configuration display message

8. When the OLAP Starter Kit option is offered, as shown in Figure G-17 on page 1225, select **Do not install the OLAP Starter Kit**. Click the **Continue** button.

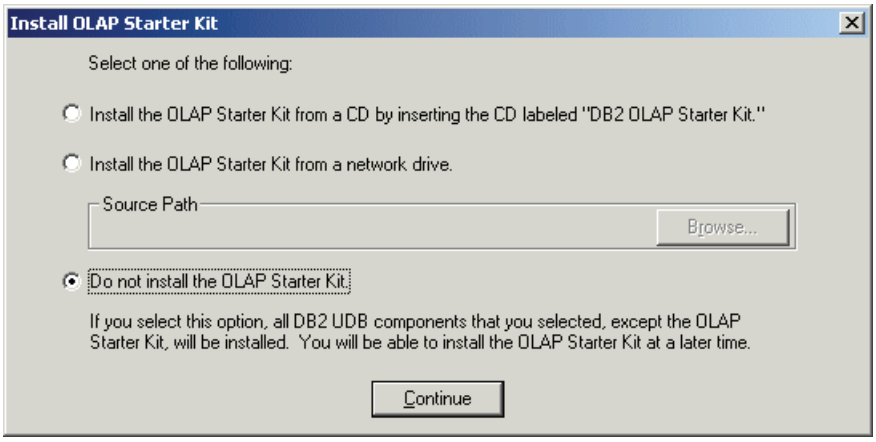


Figure G-17 OLAP Starter Kit query

9. The window shown in Figure G-18 will be displayed indicating the necessary files have been copied for the DB2 installation. Shut down any other software that is running on the system, then click **Finish**.

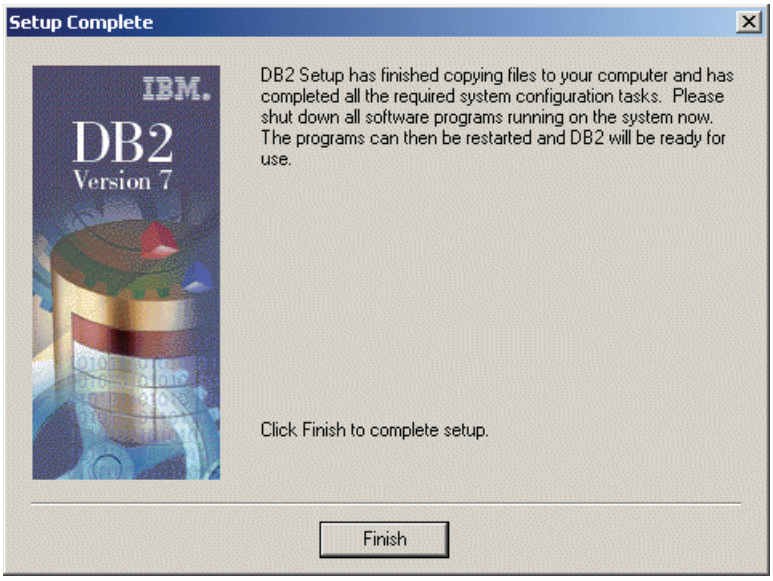


Figure G-18 File copying complete

10. The First Steps window shown in Figure G-19 on page 1226 will appear. Click **Exit**.

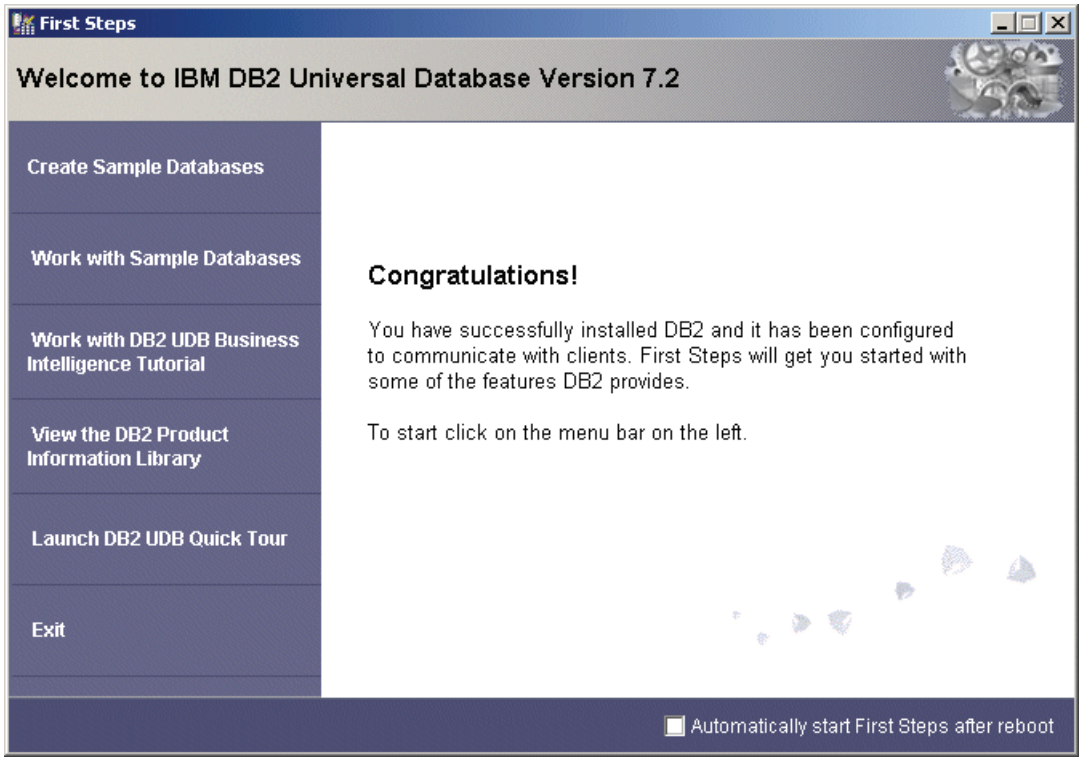


Figure G-19 DB2 First Steps window

11. The product registration window will be displayed. Click **Exit**.
12. Click **Yes** to confirm your exit when the window shown in Figure G-20 pops up.

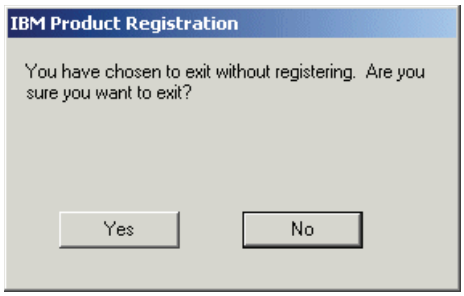


Figure G-20 Registration query window

The installation of DB2 is complete.

G.4.2 DB2 fix pack installation

This section describes the steps necessary to install the DB2 fix pack.

1. Before beginning the installation check to make sure that the db2admin user (or the user you will be using to install the fix pack) has been provided the following rights under Local Security Settings:
   - Act as part of the operating system
   - Create a token object
   - Increase quotas
   - Replace a process level token

   a. Open the Control Panel by clicking **Start -> Settings -> Control Panel -> Administrative Tools -> Local Security Policy**.

   b. Select **Local Policies -> User Rights Assignment.** You should see a panel like that in Figure G-21 on page 1228. Double-click each of the four rights to make sure that db2admin has been assigned the right. If db2admin has not been assigned the right, follow the steps detailed in G.2, “Windows configuration (before installation)” on page 1213 to assign the right to db2admin.

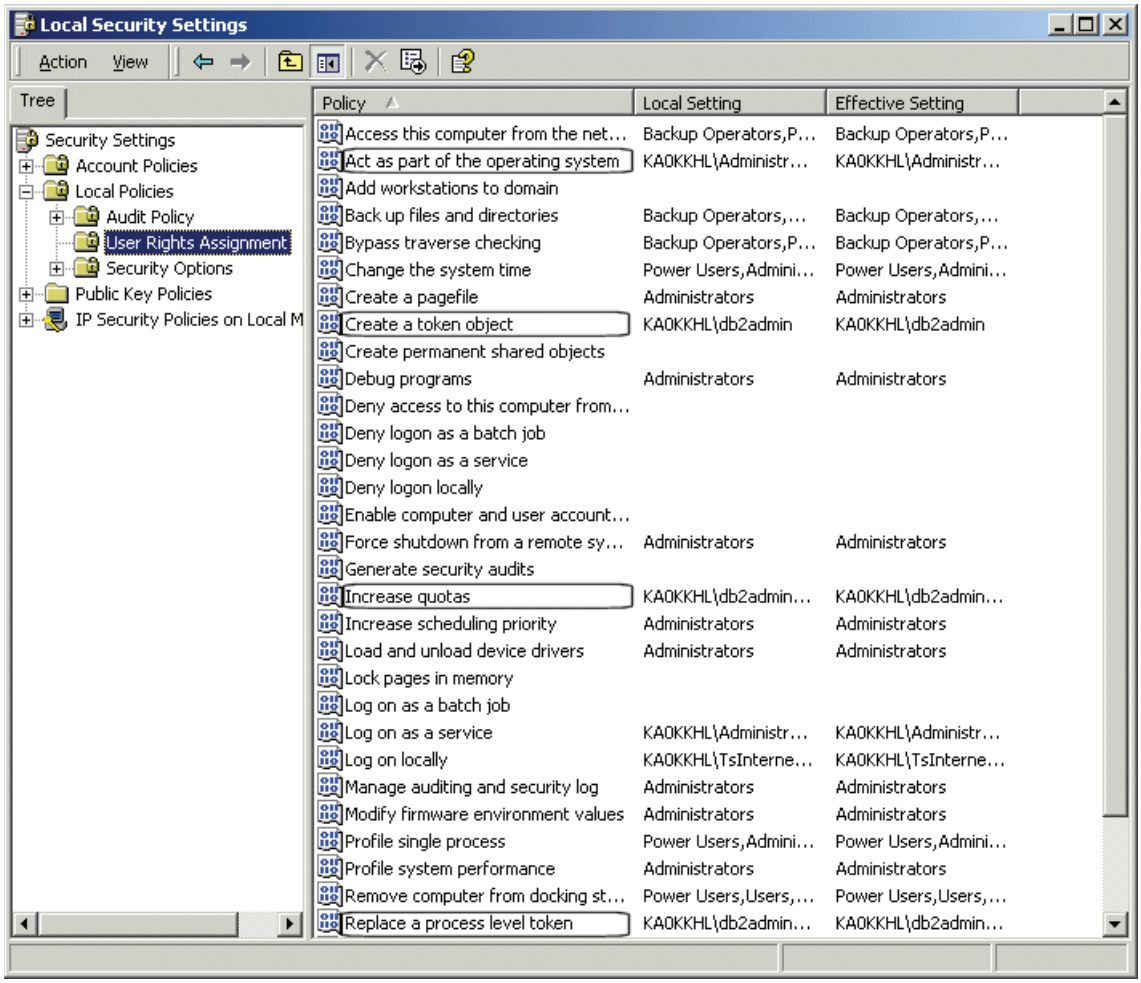


Figure G-21 db2admin local security settings

2. Insert WebSphere Everyplace Access CD 2-3 and navigate to the \db2fp\win directory. Double-click **setup.exe**. You will probably see a warning window like that shown in Figure G-22 on page 1229. If so, click **Yes**.

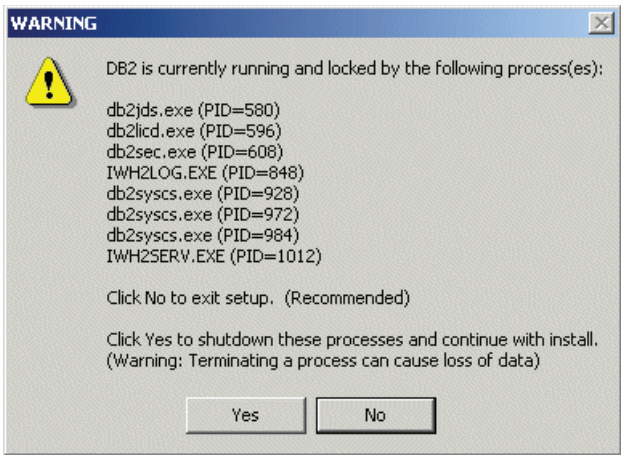


Figure G-22 DB2 process warning

3. An installation folder selection window will be displayed as shown in Figure G-23. Accept the default and click **Next**.



Figure G-23 Destination folder selection

4. You will be presented with a window to enter information for the control database. Accept the defaults, and enter `db2admin` for the user name and password. Click **Next** to continue.

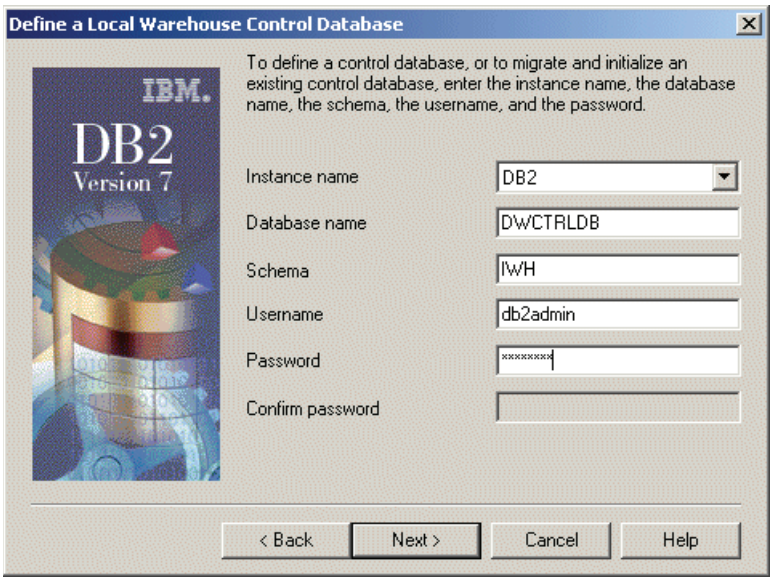


Figure G-24 Control database information

5. If you see the window shown in Figure G-25, then insufficient rights were granted to the database administrator ID. Click **OK**, then **Cancel**, and return to step 1 in G.2, “Windows configuration (before installation)” on page 1213.

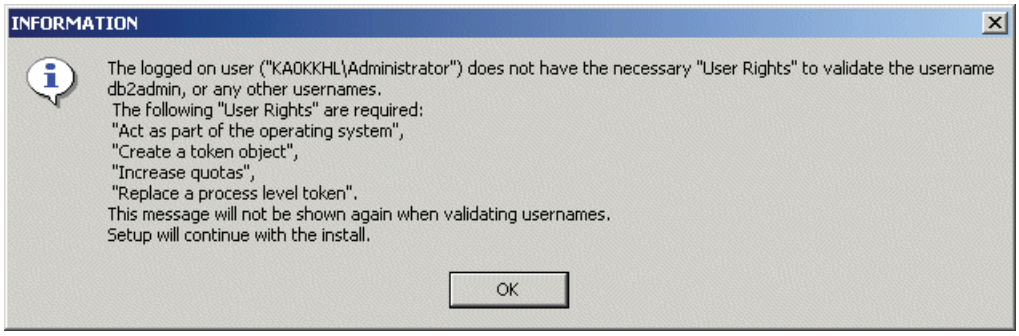


Figure G-25 db2admin user rights warning

6. The final configuration settings window will be displayed. Click **Next** to continue.

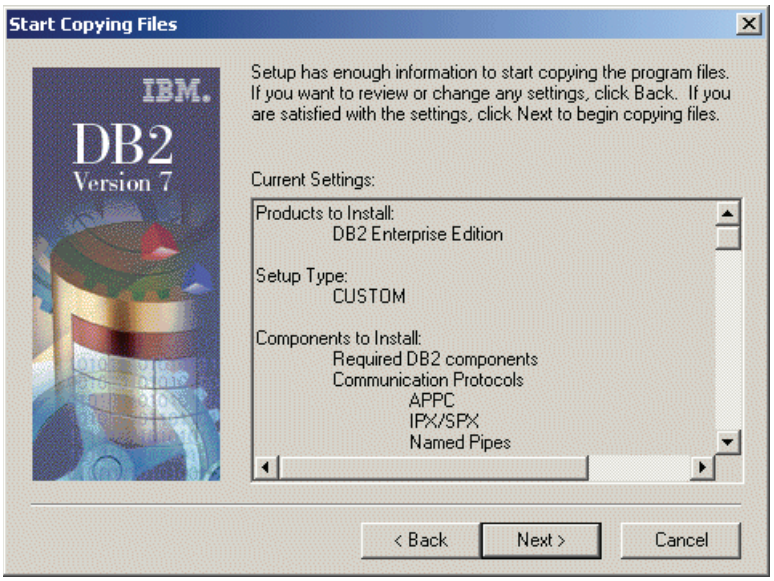


Figure G-26 Final configuration settings window

7. After the necessary files have been copied, you will be prompted to restart the machine. Be sure you have selected **Yes, I want to restart my computer now** (the default) and click the **Finish** button.

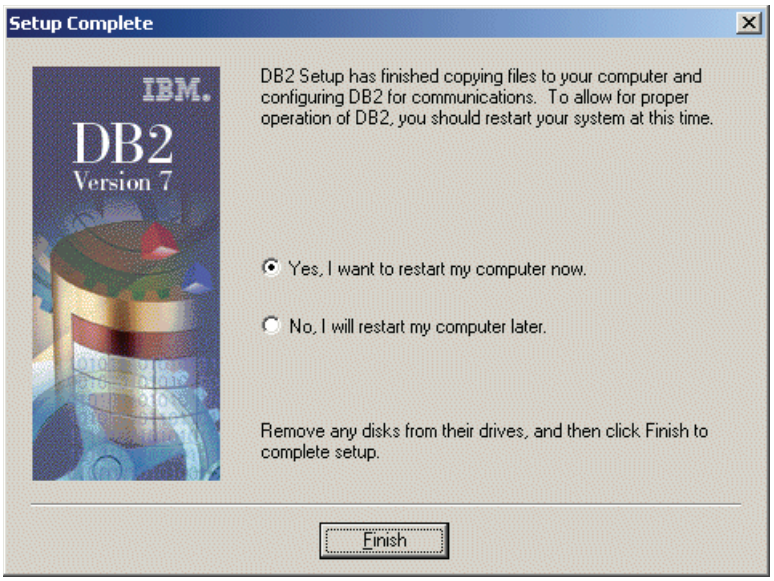


Figure G-27 DB2 fixpack Finish window

8. After the computer has been restarted you will need to select JDBC Version 2.0.

   Important: JDBC 2.0 enables Java applications to access database management systems such as IBM DB2. The usejdbc2.bat file gives this ability to this installation.

   a. You must stop all DB2 services before to switching to JDBC Version 2.0:
      i. Right-click **My Computer**, and select **Manage**.
      ii. Expand **Services and Applications** from the left tree and select **Services**.
      iii. Locate all service names that begin with DB2.
      iv. For each DB2 entry that has a status of Started, right-click the service and select **Stop**.
      v. Once you have all the DB2 processes stopped, you may close the Computer Management window.
   b. Log in and open a DOS window.
   c. Navigate to \SQLLIB\java12.
   d. Enter `usejdbc2`. This will execute the usejdbc2.bat file.
   e. Test the selection by entering `type inuse`. This will dump the contents of the inuse file, which should be displayed as JDBC 2.0 as illustrated in Figure G-28.

```
C:\>cd Program Files\SQLLIB\java12

C:\Program Files\SQLLIB\java12>usejdbc2
Backing up java\db2java.zip to java11 directory
        1 file(s) copied.
Copying db2java.zip to usejdbc2\..\..\java
        1 file(s) copied.
Usejdbc2.bat successful

C:\Program Files\SQLLIB\java12>type inuse
JDBC 2.0

C:\Program Files\SQLLIB\java12>_
```

Figure G-28 JDBC selection

Installation of the DB2 fix pack is now complete.

G.5 WebSphere Studio Site Developer V5.0

To install the WebSphere Studio Site Developer, do the following:

1. Insert WebSphere Everyplace Access CD 8-2. Navigate to the \wssd directory and double-click **setup.exe**. You will see the Welcome window as illustrated in Figure G-29. Click **Install IBM WebSphere Studio Site Developer** to continue.

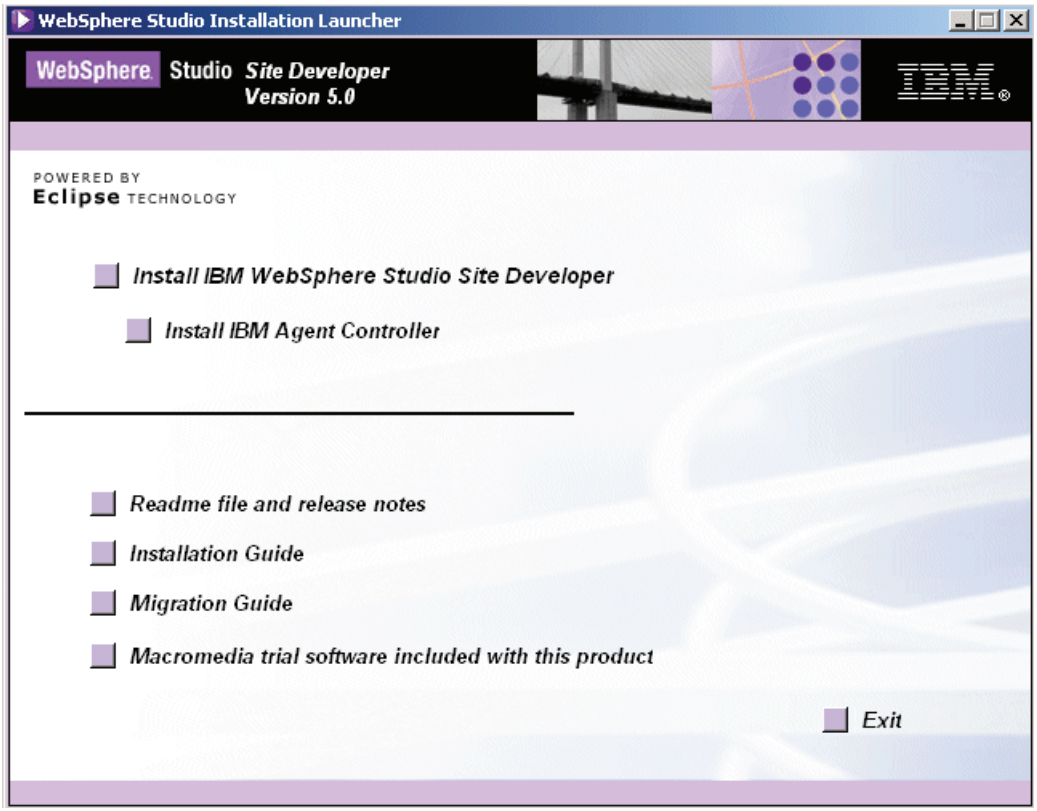


Figure G-29 WebSphere Site Developer installation Welcome window

2. Click **Next** to continue past the copyright notice window.
3. Select **I accept the terms of this license agreement**. Then click **Next** to continue.
4. Accept the default settings for directory locations. Click **Next** to continue.

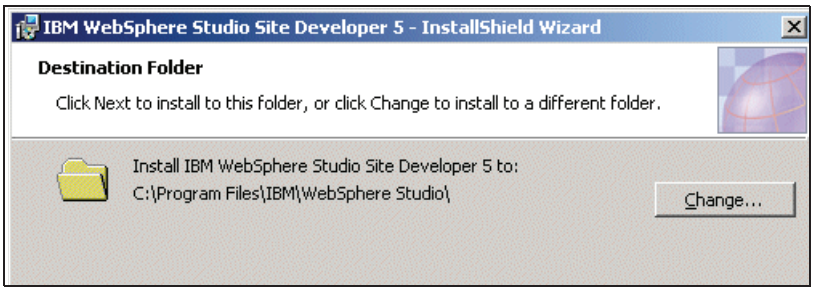


Figure G-30 WebSphere Studio Site Developer features to install

5. In addition to the default features, select **WebSphere Application Server v4.0** from the Run-time environments category and **Language Pack** from the Optional Features category to be installed to the local hard drive as shown in Figure G-31. Click **Next** to continue.

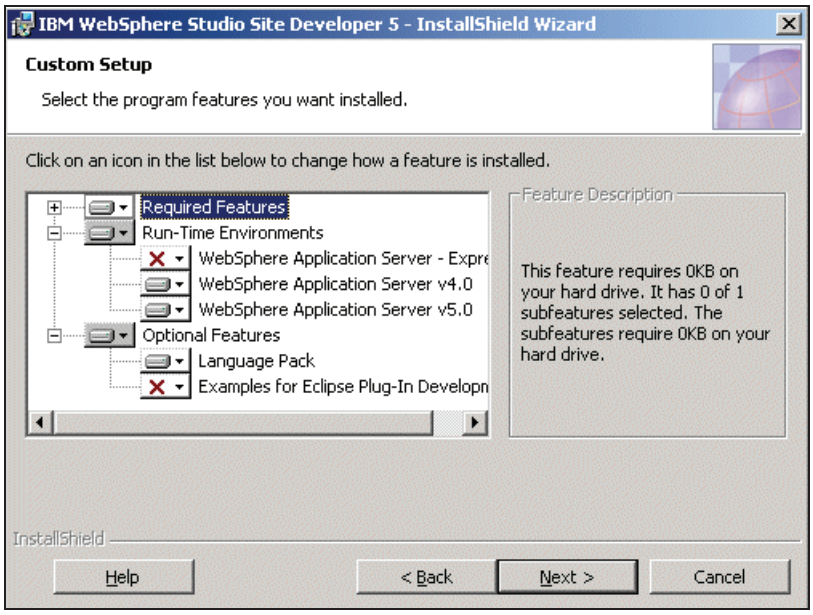


Figure G-31 WebSphere Studio Site Developer features to install

6. Click **Install** to begin the installation.
7. Midway through the installation, you will be prompted to insert the disk IBM WebSphere Studio Site Developer - DISK2 (WebSphere Everyplace Access CD 8-3). Insert this disk and click **OK** to continue the installation.

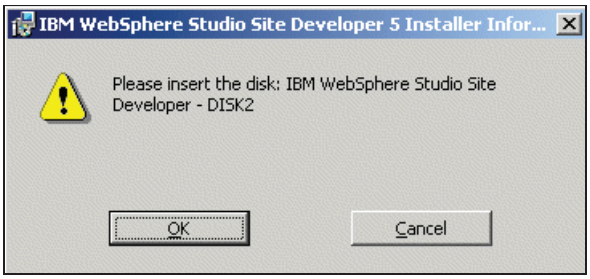


Figure G-32 Insert the disk and click OK to continue the installation

8. At the completion of the installation process, click **Finish** to exit.
9. If you are prompted to restart the machine, click **Yes** to do so.

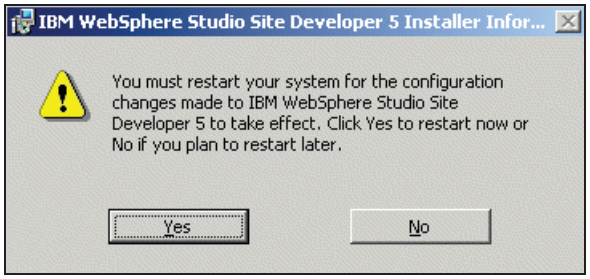


Figure G-33 Click Yes to restart the machine

G.6 Application Server Single Server V4.0 FP4

To install the WebSphere Application Server Single Server V4.0, do the following:

1. Insert CD 8-1. Navigate to the \was_ss\win directory and double-click **setup.exe.** You will be prompted with a window as shown in Figure G-34 on page 1236. Select **English (United States)** and click **OK** to continue the installation.



Figure G-34 Language window

2. Shut down all Web servers and close all Windows applications as recommended by the installer. Then click **Next** to continue.

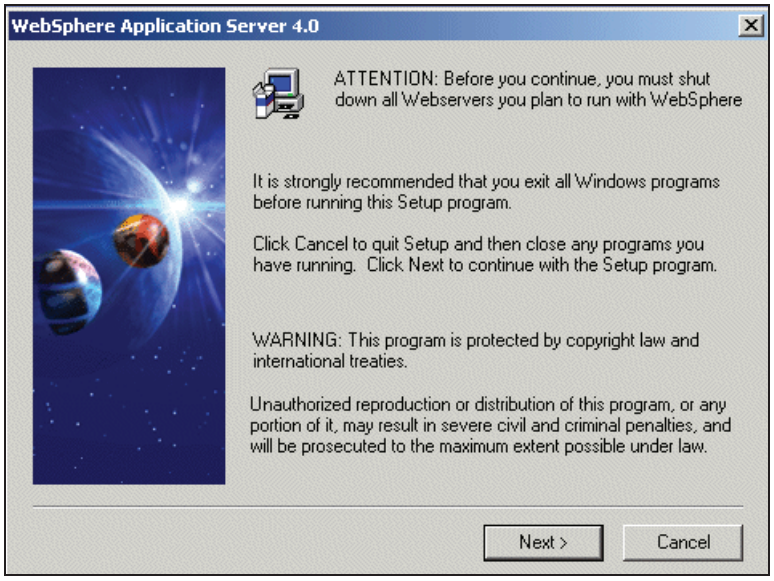


Figure G-35 WebSphere Application Server installation window

3. Choose **Custom Installation**. Then click **Next** to continue the installation.

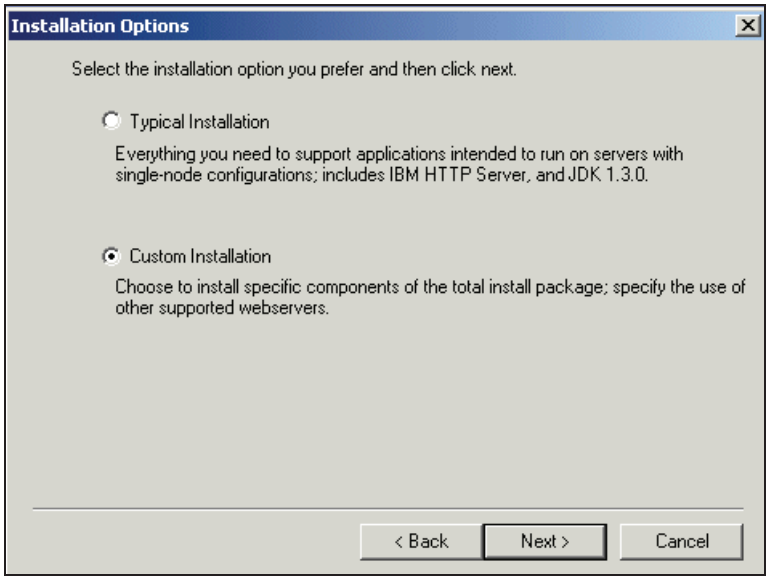


Figure G-36 Installation options window

4. Select **Application Server, Application and Development Tools**, and **IBM JDK 1.3.0**. Make sure that no other options are selected. Click **Next** to continue.

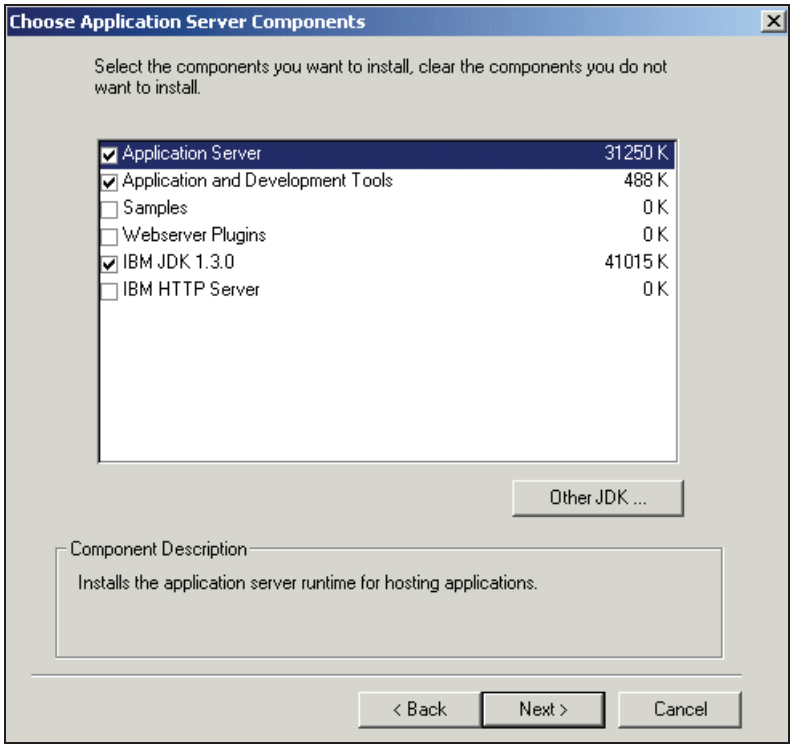


Figure G-37 Component selection window

5. Accept the default installation directory, C:\WebSphere\AppServer. Click **Next** to continue the installation process.

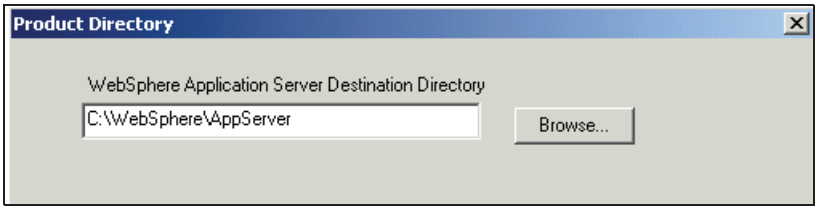


Figure G-38 Selecting the destination directory

6. Accept the default for the program folder name, and click **Next** to continue.

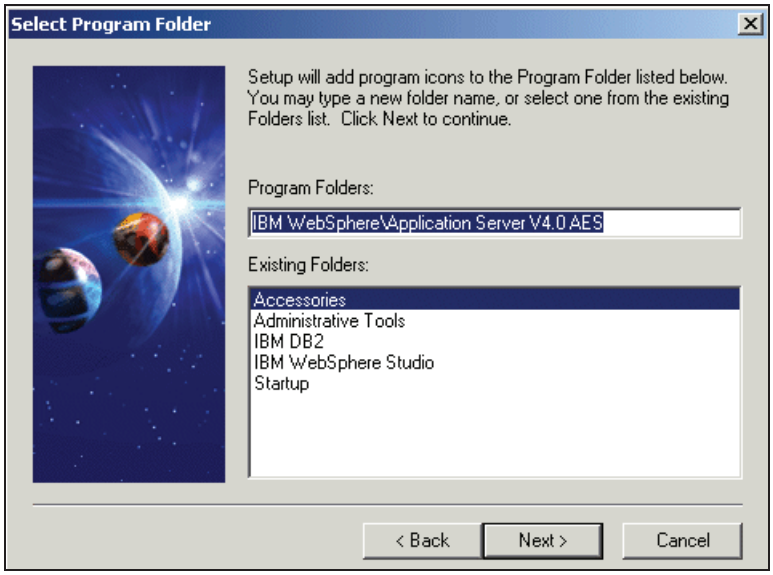


Figure G-39 Programs folder name

7. Review the installation options as shown in Figure G-40. Click **Next** to begin the installation of the program.

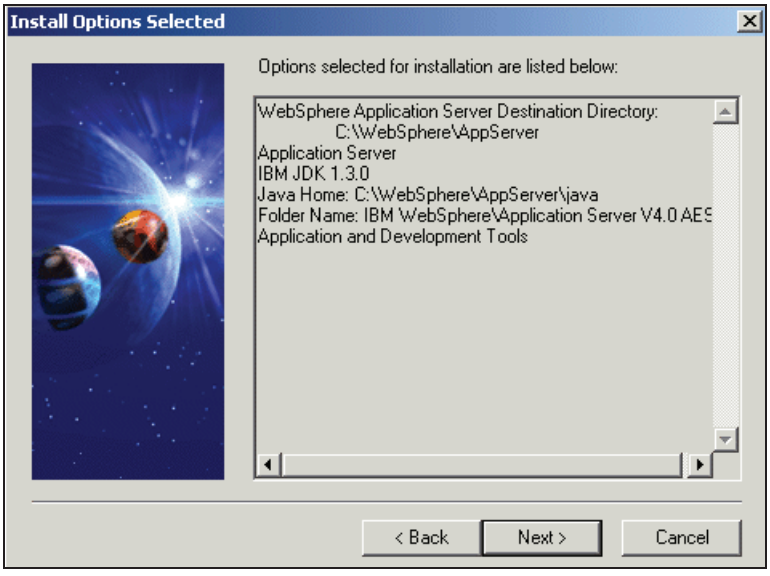


Figure G-40 Installation options

8. When installation is complete, you may check **Yes, I want to view the ReadMe file now**. Click **Finish** to end the installation process.

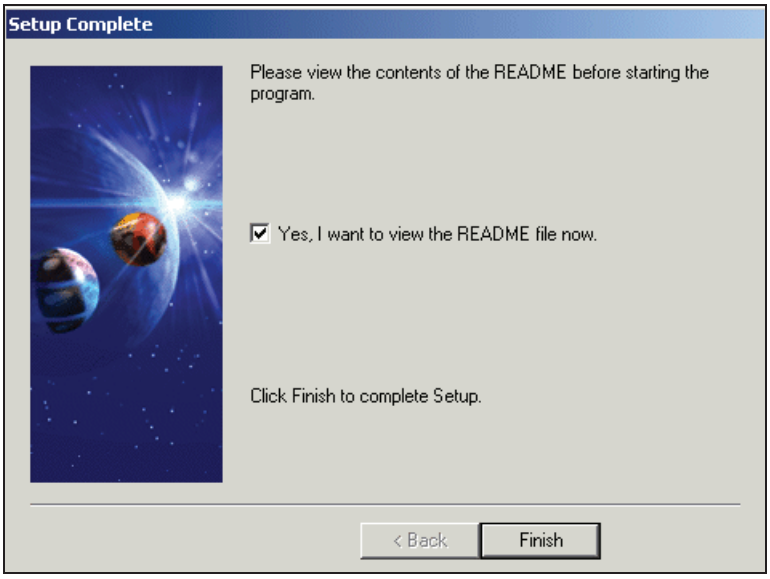


Figure G-41 WebSphere Application Server final installation step

9. Select **Yes, I want to restart my computer now** (the default) and click **OK**.

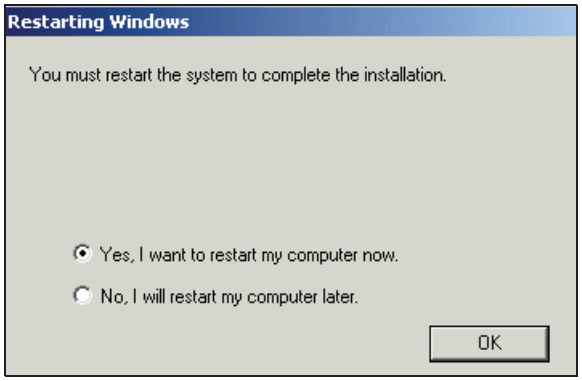


Figure G-42 Restart the machine

10. Once the machine is restarted, you will be see the window shown in Figure G-43 on page 1241. Close this window. Installation is complete.



Figure G-43 WebSphere Application Server First Steps window

G.6.1 Manually starting WebSphere Application Server

> **Note:** This section is only included for reference. Under normal testing conditions, it will never be necessary to do this.

Although the WebSphere Studio Site Developer test environment is suitable for development work and testing, at times it may be necessary to access the Administration Console to WebSphere Application Server. This requires manually starting the WebSphere Application Server. To do this, click **Start -> Programs -> IBM WebSphere -> Application Server V4.0 AES -> Start Application Server**.

The Administration Console is the accessible by clicking **Start -> Programs -> IBM WebSphere -> Application Server V4.0 AES -> Administrator's Console**. The default user ID used to access the console is `wpsadmin`.

G.7 WebSphere Everyplace Access/Portal Toolkit V4.3

This section provides information on the installation of the WebSphere Everyplace Access/Portal Toolkit.

1. Insert WebSphere Everyplace Access CD 8-1. Navigate to the \EveryplaceToolkit directory and double-click **Install.bat.** Wait for the window shown in Figure G-44 to load. Then click **Next** to continue.

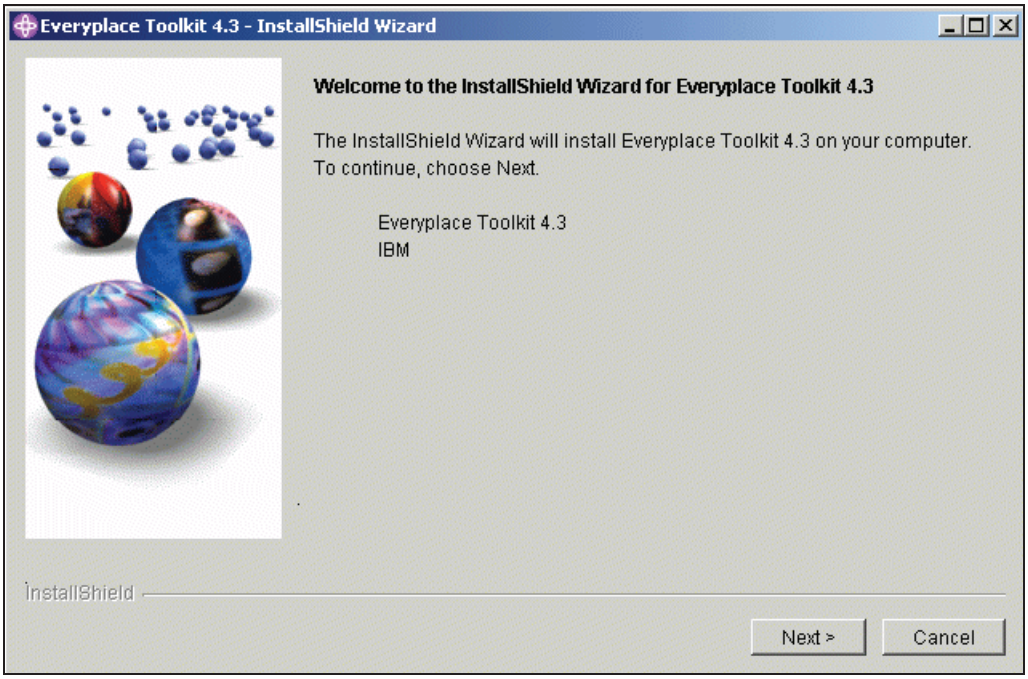


Figure G-44 Toolkit installation

2. Click **Next** to continue.

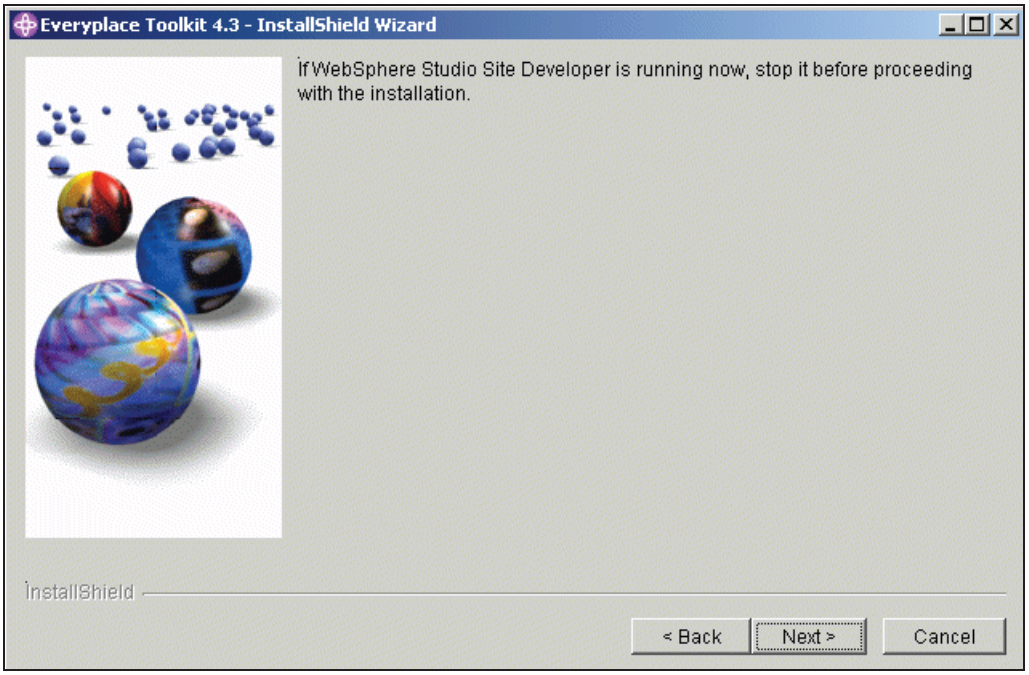


Figure G-45 Close Site Developer if it is running

3. Accept the Software License Agreement by selecting **I accept the terms** and click **Next**.



Figure G-46 Software License Agreement window

4. Choose both the **Everyplace Toolkit** and **WebSphere Portal Test Environment** to install. Then click **Next** to continue the installation process.

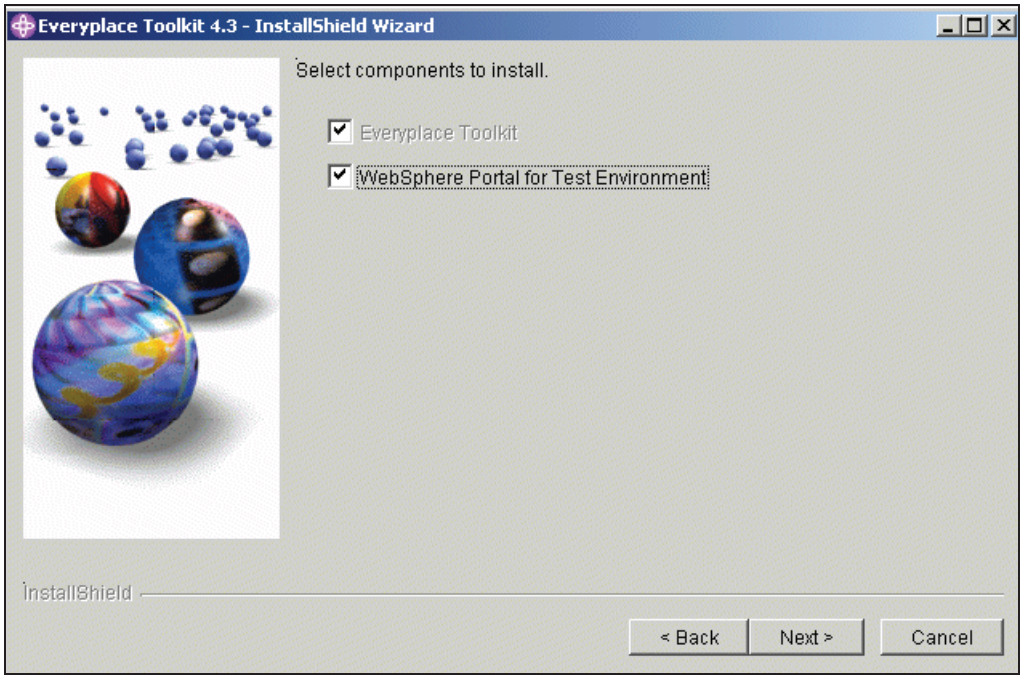


Figure G-47 Component selection window

5. Accept the default installation location for the Toolkit. Then click **Next** to continue.

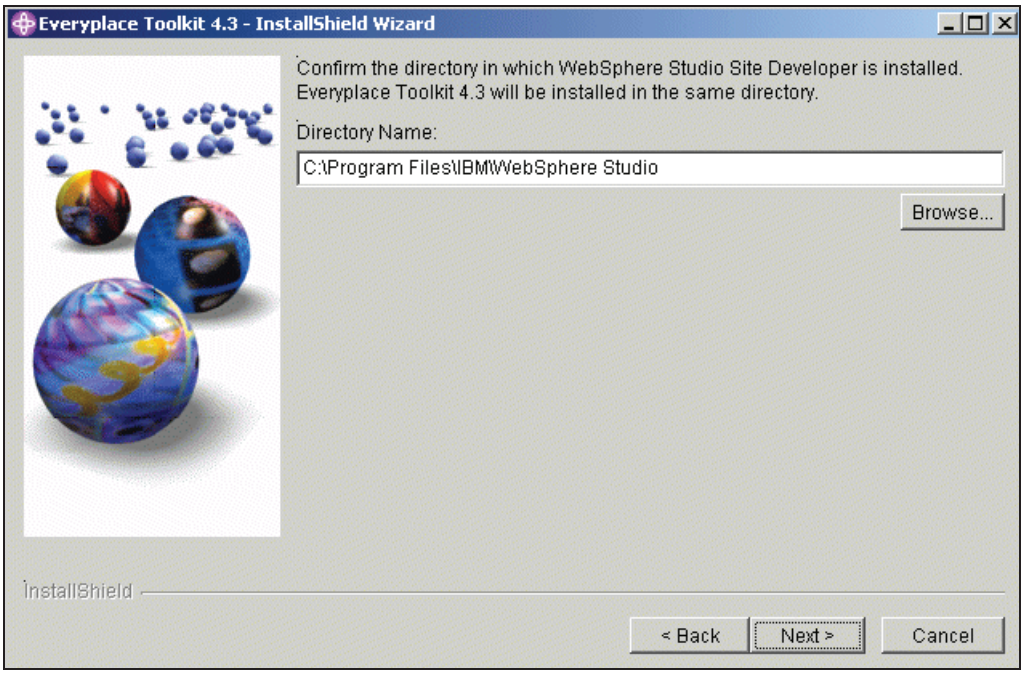


Figure G-48 Installation directory location window

6. Review the installation information and click **Next** to continue.

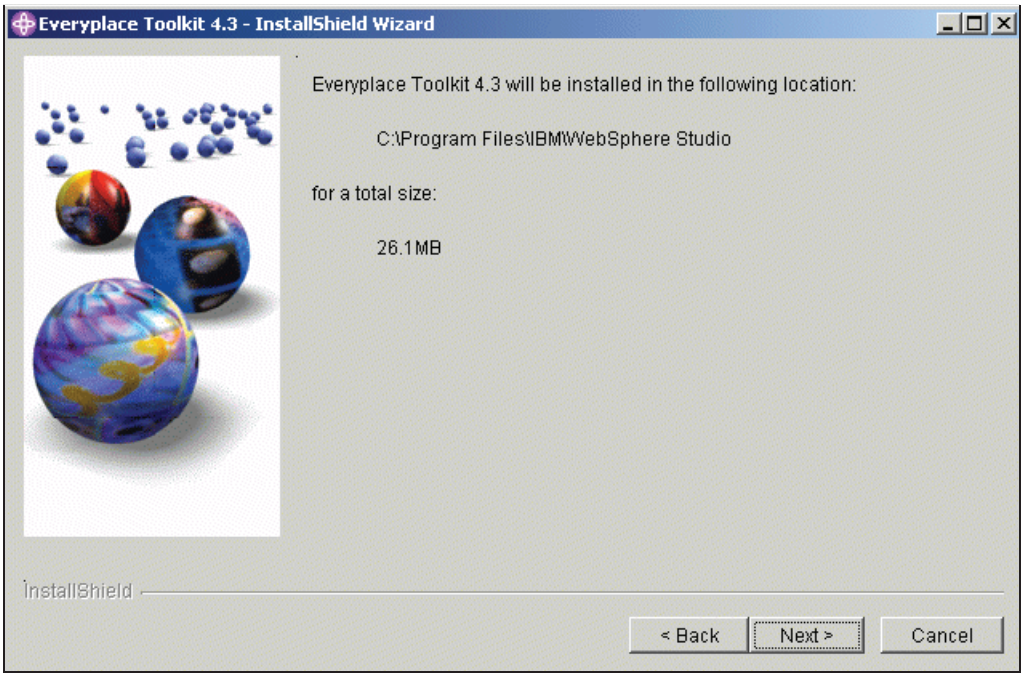


Figure G-49 Installation information

7. Click **Next** to begin installation.
8. The next step is to configure the WebSphere Portal environment with which the Toolkit will be used. You should have the WebSphere Portal V4.2 disk (WebSphere Everyplace Access CD 5) ready to complete this installation. WebSphere Portal will also be installed as past of the process of installing the toolkit. Enter `portaldev.itso.ral.ibm.com` as the host name, `db2admin` as the DB2 Database user, and `db2admin` as the DB2 Database password as shown in Figure G-50 on page 1248. Click **Next** to continue.

Note: When configuring your machine for use, substitute `myserver.mycompany.com` for `portaldev.itso.ral.ibm.com` and enter a DB2 user and password that are correct for your configuration.

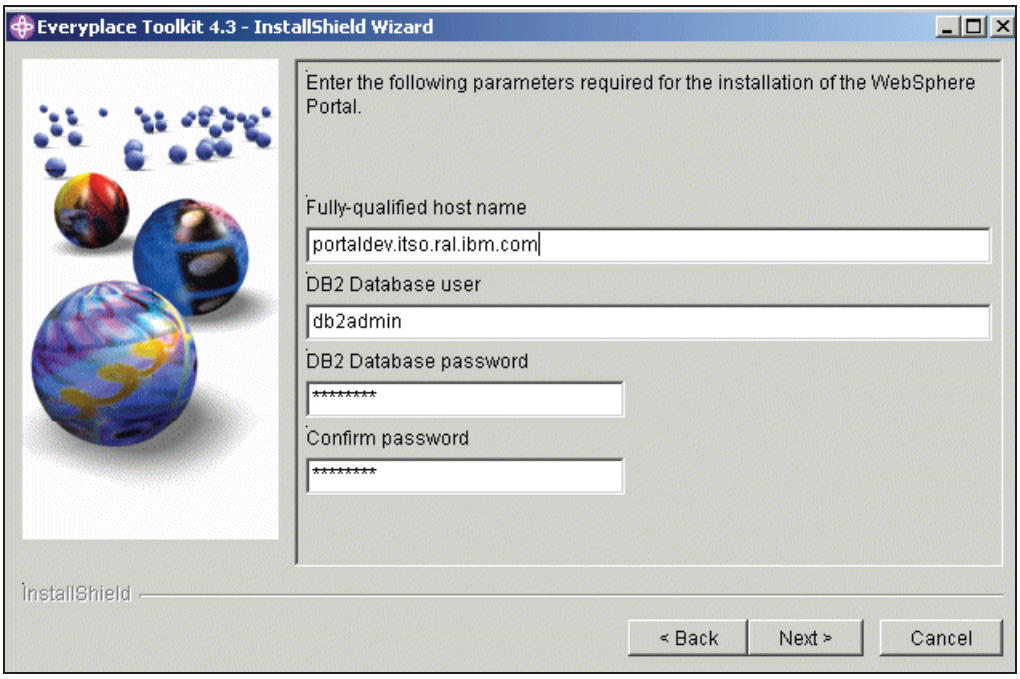


Figure G-50 WebSphere Portal configuration

9. Insert the disk containing WebSphere Portal V4.2 into the machine's CD-ROM drive. Browse to the \wps directory on the disk for the location of the WebSphere Portal installer. Browse to the \pzn directory on the disk for the location of the Personalization JARs. When the directory locations are accurate, as shown in Figure G-51 on page 1249, click **Next** to continue.

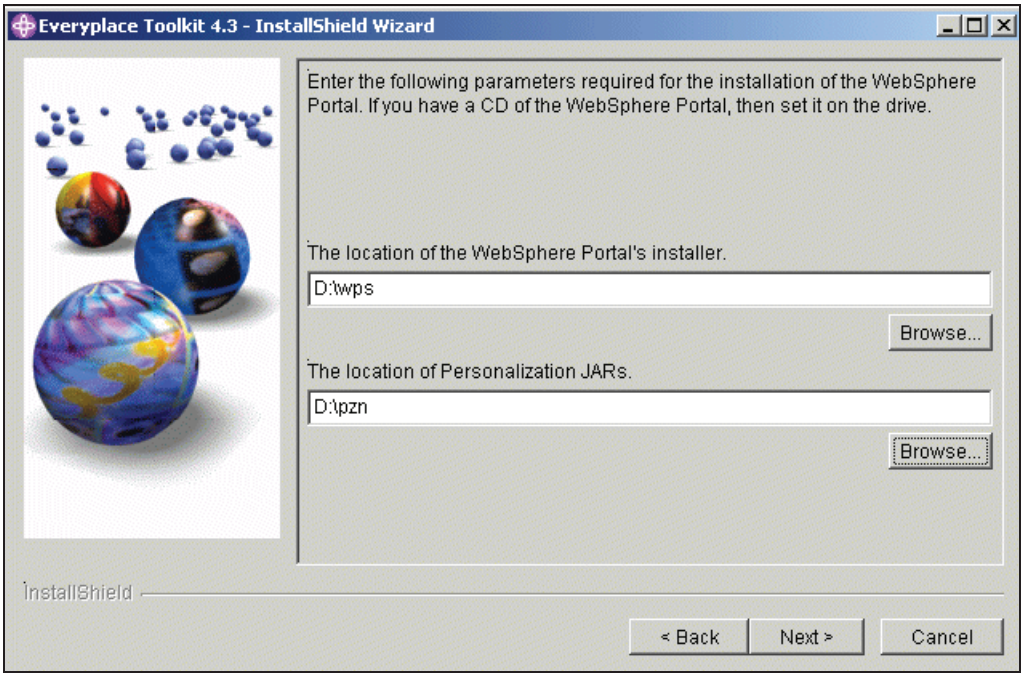


Figure G-51 WebSphere Portal installation file locations

10. Click **Next** to install WebSphere Portal.
11. When the installation is complete, click **Next** to continue.
12. Finally, click **Finish** to complete the installation.



Figure G-52 Completion of the Everyplace/Portal Toolkit installation

You are now ready to use WebSphere Studio Site Developer and the WebSphere Everyplace Access/Portal Toolkit to create portlets and Portal applications.

G.8 Configure WebSphere Studio Site Developer and the Toolkit

To use the WebSphere Studio Site Developer and WebSphere Everyplace Access/Portal Toolkit, you must first configure a test environment in which to run your code. It is not necessary for a separate instance of WebSphere Application Server or WebSphere Portal to be running in order to test code. All testing can be done from within the IDE.

1. Start the WebSphere Studio Application Server. Select **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.0**.
2. The first time that WebSphere Studio Site Developer is run, it will ask you to specify a directory to be used as your development workspace. Accept the default directory, check the box next to **Use this workspace as the default**

and do not show this dialog box again, and click **OK** to open Site Developer.

Note: Alternatively, you can specify your own directory to store your development workspace.

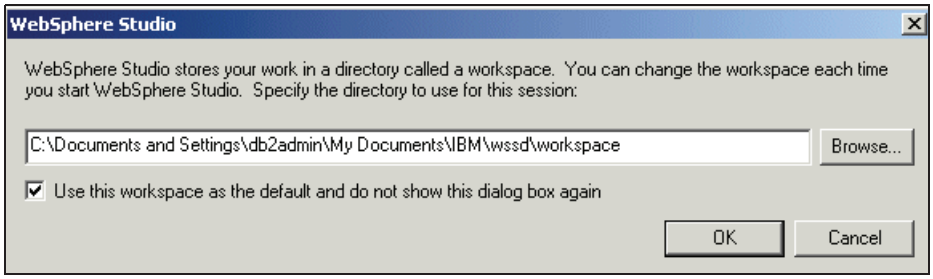


Figure G-53 Workspace directory

3. It may take a while for Site Developer to load for the first time.

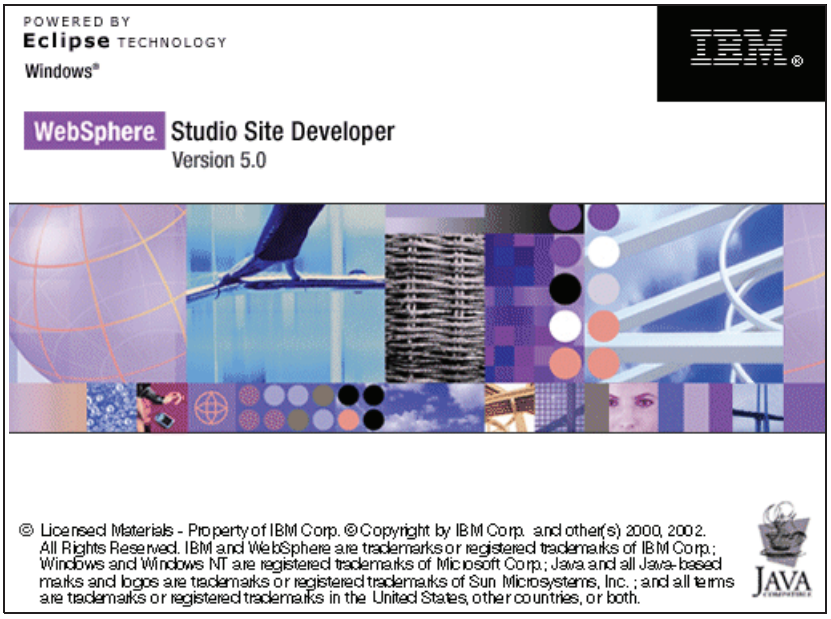


Figure G-54 Loading WebSphere Studio Site Developer

4. Once WebSphere Studio Site Developer is loaded, click the **Server Configuration** tab as shown in Figure G-55 on page 1252.

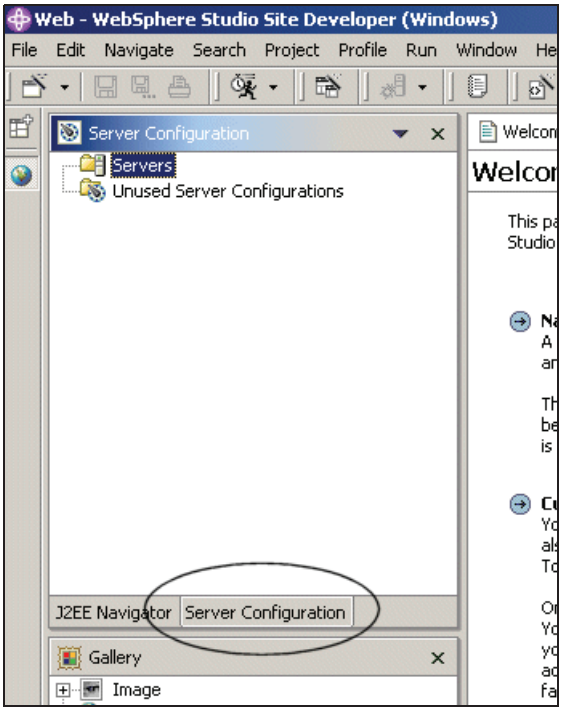


Figure G-55 Server Configuration tab

5. Right-click **Servers**, then click **New**, then **Server and Server Configuration** as shown in Figure G-56 on page 1253.

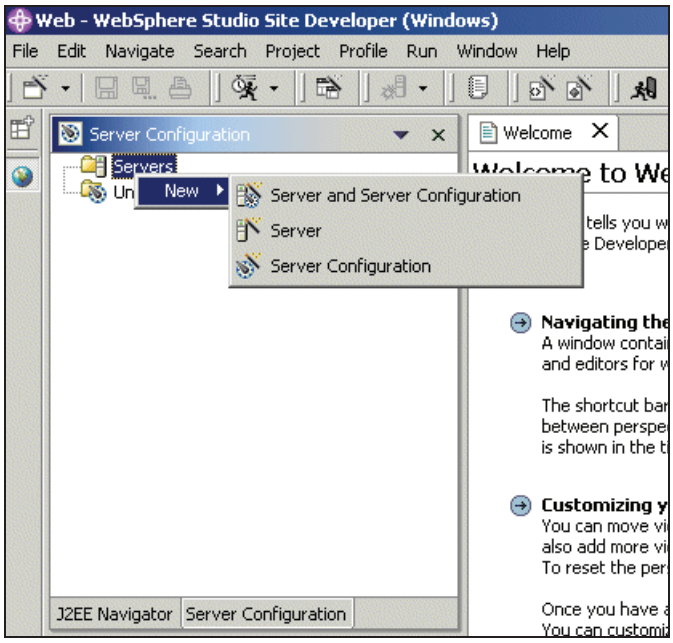


Figure G-56 Creating a new server and configuration

6. A new window will be displayed to allow you to configure the new server and server configuration. Enter `Test Environment` as the server name. Select **Test Environment** under **WebSphere Portal Version 4.2** as the server type. The result should look like that shown in Figure G-57 on page 1254. Click **Next** to continue.

Important: You must enter a server name to continue. Also, be sure to select **Test Environment**.

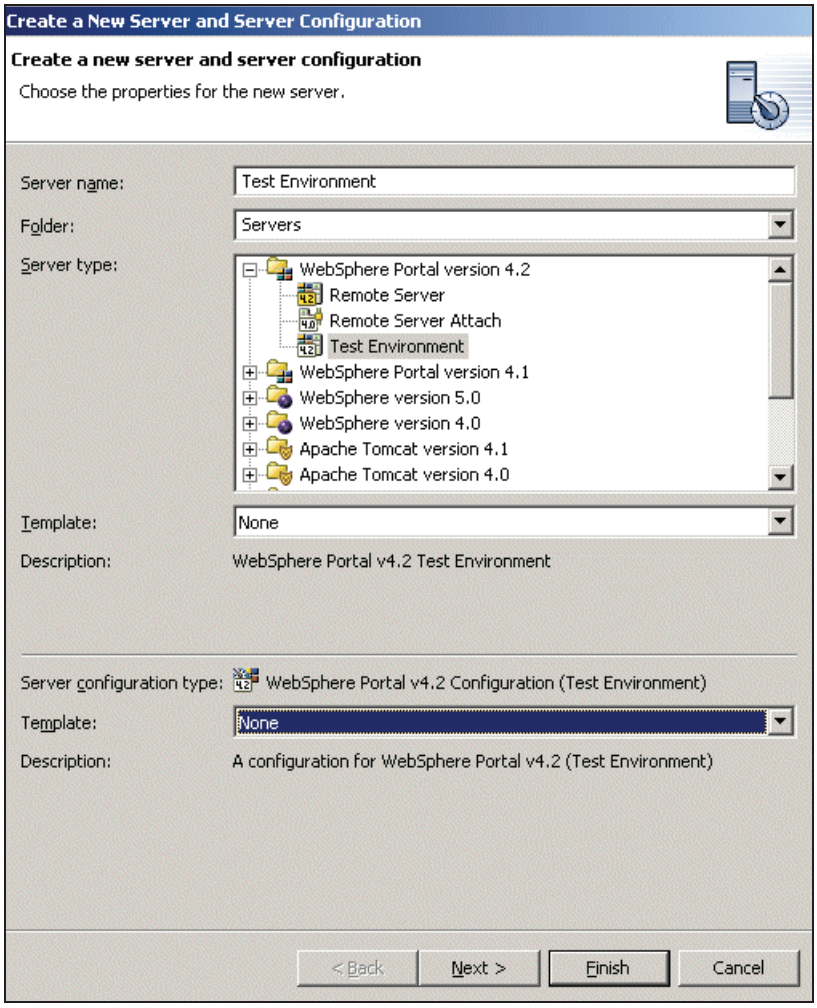


Figure G-57 Server configuration settings window

7. Click **Yes** when asked **Do you want to create a new server project with the name Servers?**

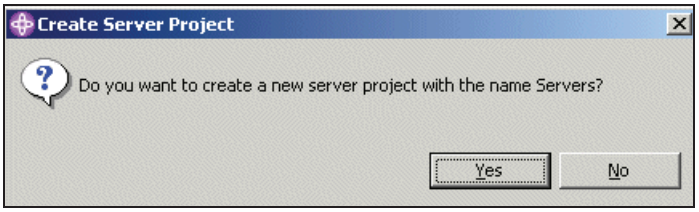


Figure G-58 Do you want to create a new server project?

8. This will bring up a window to select the HTTP port number to be used by Site Developer. Enter 9080 as shown in Figure G-59, and click **Finish** to add the test environment.

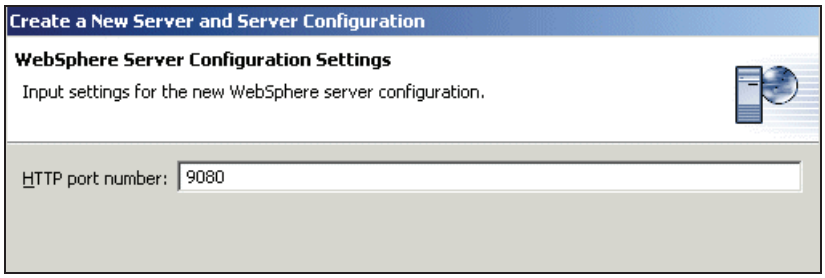


Figure G-59 Port selection

9. The server has been successfully added and can now be seen in the Server Configuration tab.

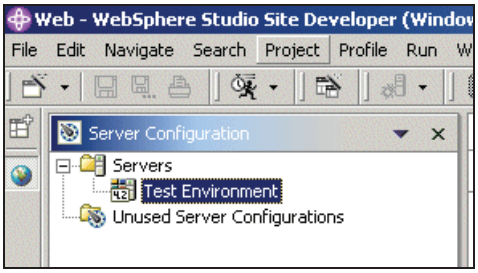


Figure G-60 Test environment has been added successfully

10. To add a project to the test environment, right-click **Test Environment**, click **Add**, and select the project you would like to add.

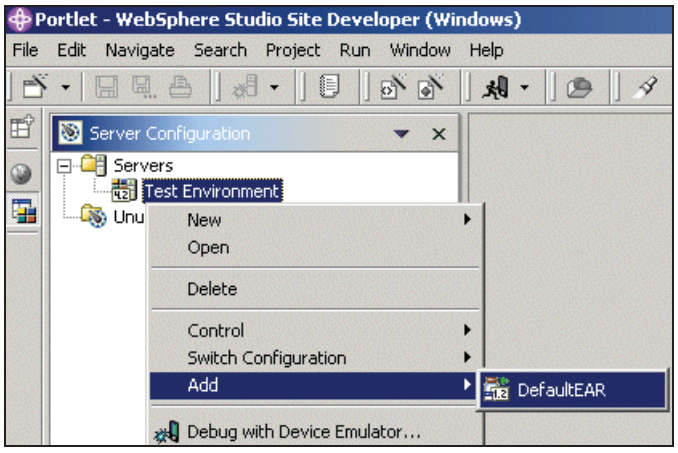


Figure G-61 Adding a project to the test environment

11. Now, WebSphere Studio Site Developer and the Everyplace/Portal Toolkit are configured so that whenever you test a portlet by right-clicking it and selecting **Run on Server**, it will execute in this test environment.

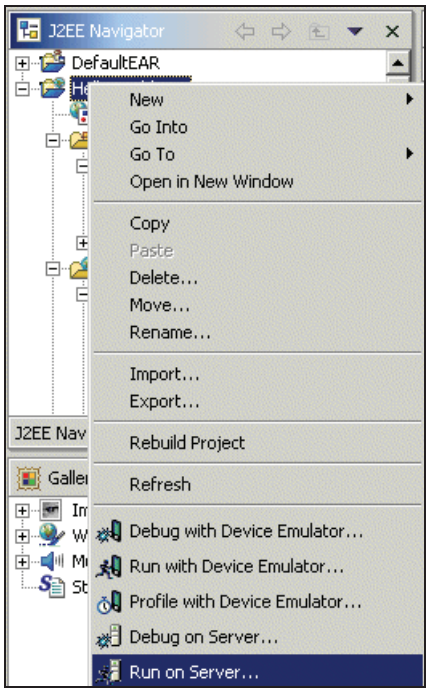


Figure G-62 Running a portlet in the test environment

H

Lotus Domino Server and Sametime installation

In this appendix we provide step-by-step instructions for installing Lotus Domino Server V5.0.10 and Sametime Server V3.0.

© Copyright IBM Corp. 2003. All rights reserved.

Installing Lotus Domino Server

The installation steps shown in this section were carried out on Windows 2000 Server. The following steps are required to install Lotus Domino Server V5.0.10.

1. Run **`Setup.exe`** from the Domino Server Installation CD. At the license agreement window, click **Yes**.
2. Enter the name and company field and click **Next**.

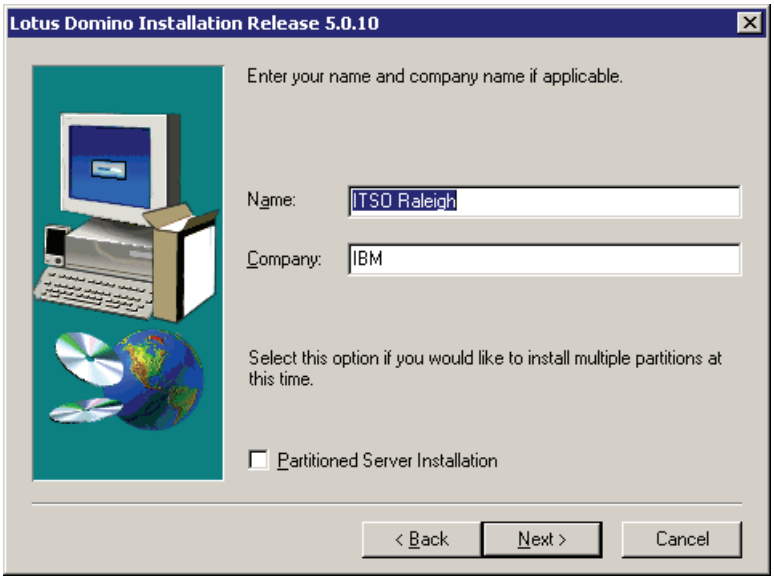


Figure H-1 Name and company details

3. Select the folder where you want to install the server and click **Next**.

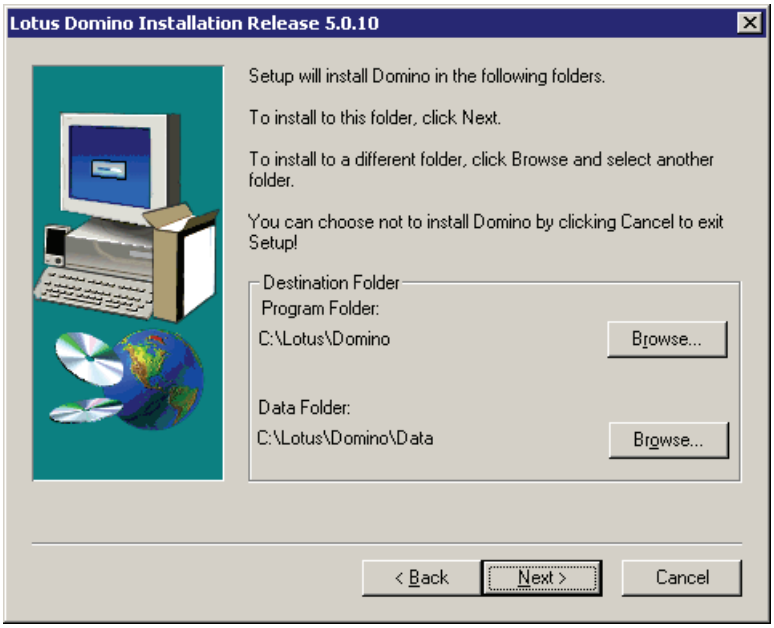


Figure H-2 Installation path

4. Select the setup type accordingly.



Figure H-3 Setup type

5. Click **Next** on the next window. This will start the installation.
6. Click **Finish** to complete the installation.
7. Select **Start -> Programs -> Lotus -> Domino Server**. This will bring up the window in Figure H-4 on page 1261. The next few steps are for configuring the Domino Server. This was the first Domino Server we were installing, so we choose the first option.

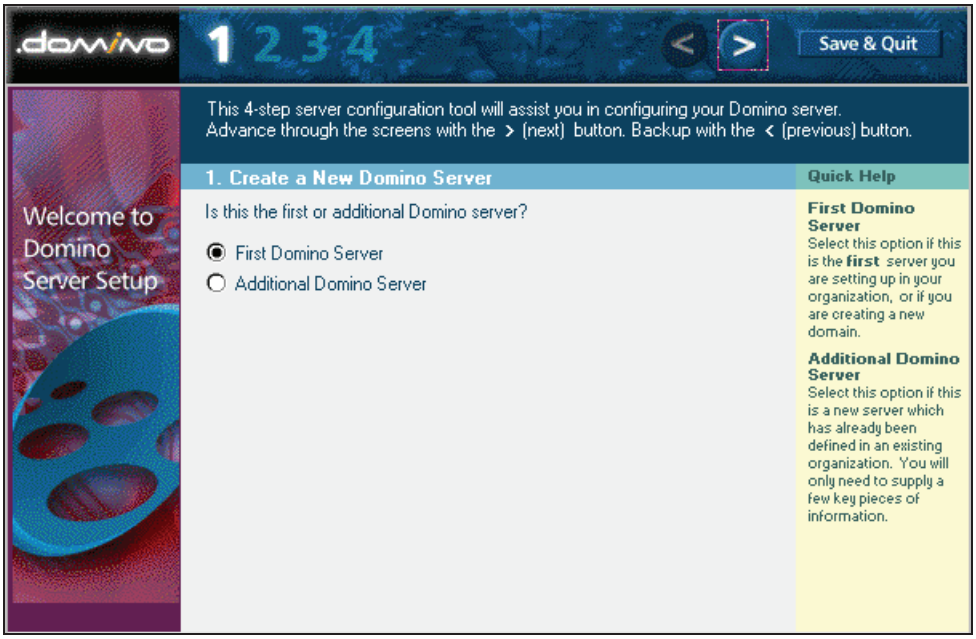


Figure H-4 Domino configuration -step 1

8. Select **Quick and Easy Install**.

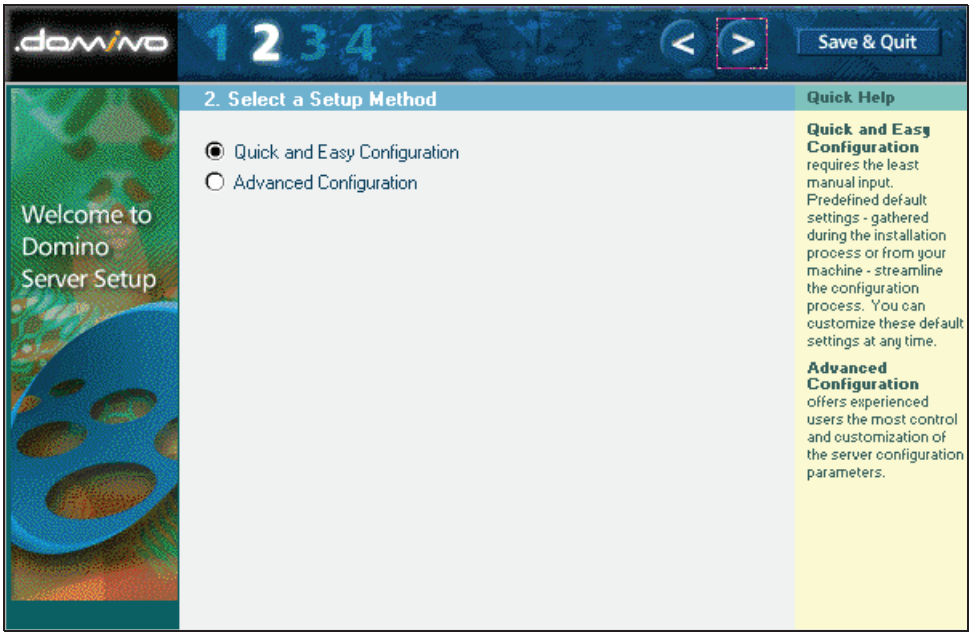


Figure H-5 Domino configuration - step 2

9. Select the appropriate options based on how the users connect to the Domino Server.

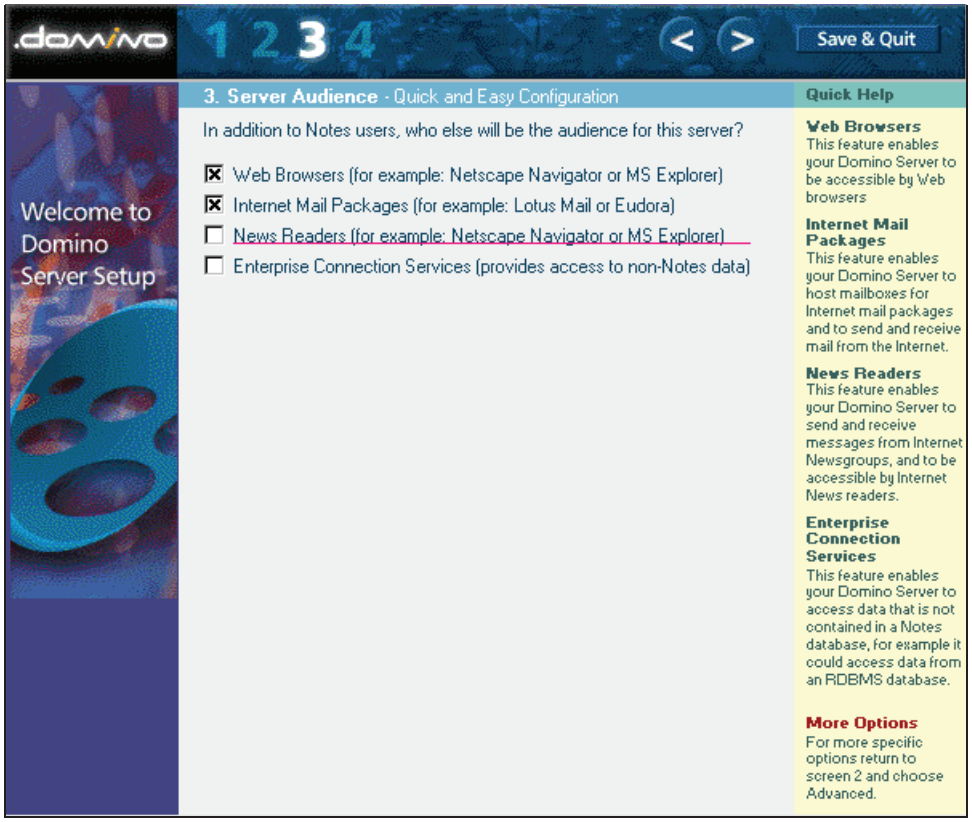


Figure H-6 Domino configuration - step 3

10. Enter the server's domain name. If you already have a certifier's ID, you can enter it or you can let the tool create a new certifier ID. A certifier ID is required for creating new users. The tool generates a random password, you can change it. Enter the server name and host name. You can choose to create a new administrator ID or if you have an existing ID you can use it.



Figure H-7 Domino configuration - step 4

11. Click **Exit Configuration**.

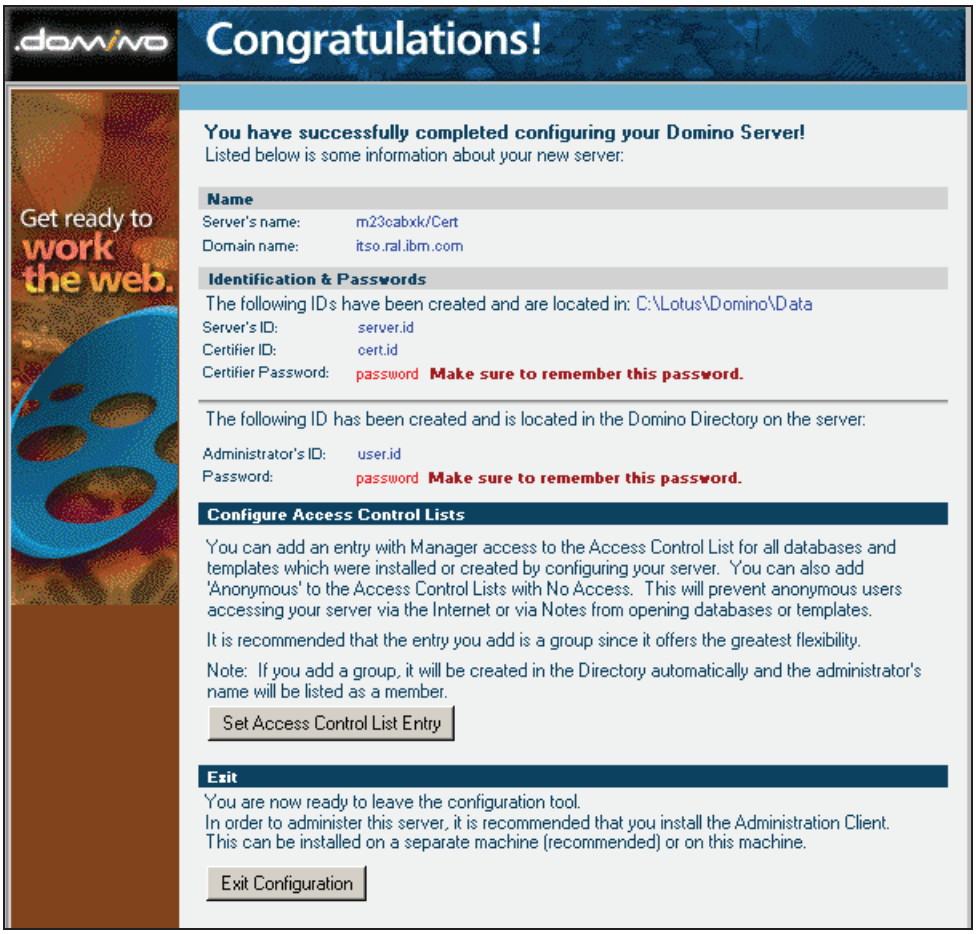


Figure H-8 Domino configuration - finished

You can check if the Domino Server was installed correctly by starting the server. To start the server, click **Start -> Programs -> Lotus -> Domino**.

Installing the Domino Administration Client

To administer the Domino Server, Lotus Domino Administration Client has to be installed. For example:

1. Click **Setup** on the Lotus Client. This will bring up the window shown in Figure H-9 on page 1266. Select **Domino Administrator** and click **Next**.

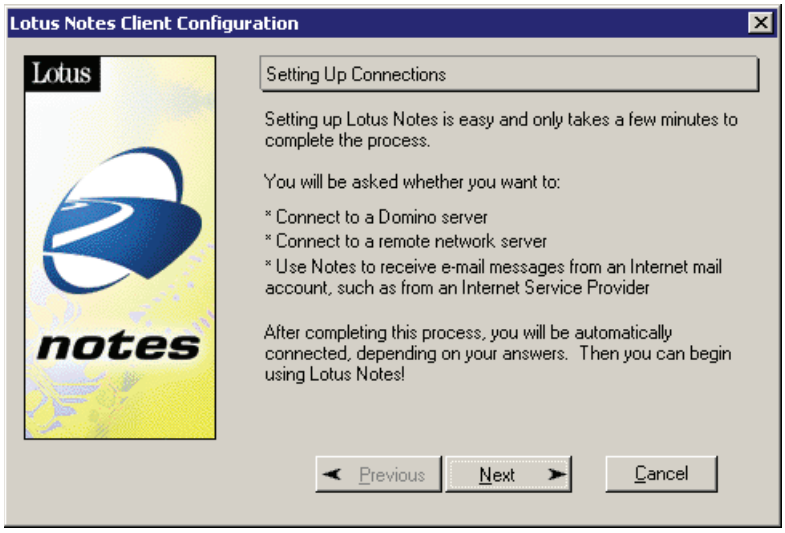


Figure H-9 Lotus Notes client configuration

2. On the window that follows, select **I want to connect to Domino Server** and click **Next**.
3. Select **Setup a connection to Local area network (LAN)** and click **Next**.
4. Enter the Domino Server name and click **Next**.
5. If you have a user ID, select the first option. Select your Notes ID file. If you do not have a user ID, enter the user name. Enter the Domino Server administrator name.

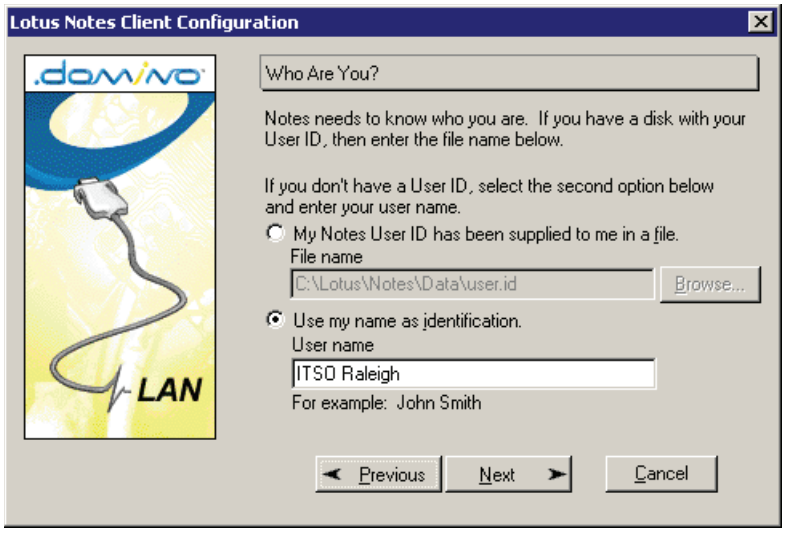


Figure H-10 Client Configuration: Identification

6. If you are selecting the second option, you will be prompted to configure the following:
   - Mail account
   - News server
   - Internet directory server
   - Proxy Server
7. Once the setup is complete you will be prompted for the password. After you log in, successfully the window shown in Figure H-11 on page 1268 is displayed.

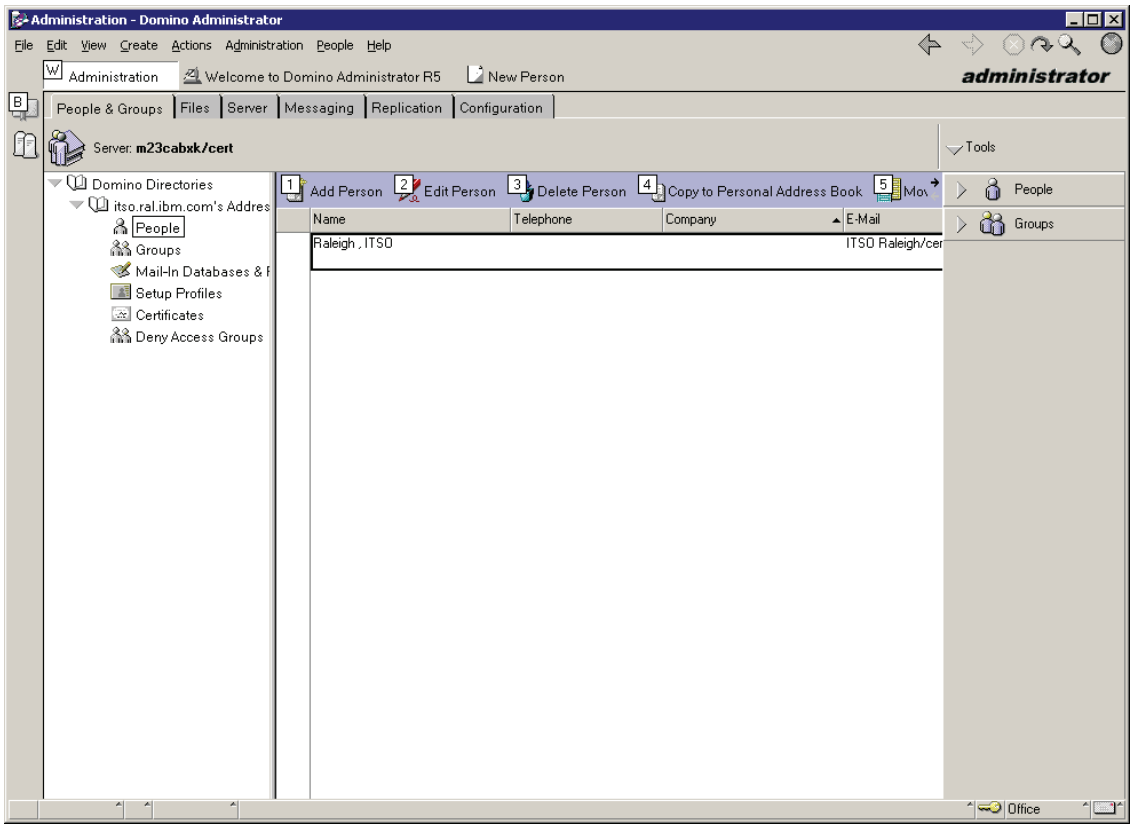


Figure H-11 Domino Client Administrator

Installing Lotus Sametime Server

The installation of Lotus Sametime Server involves installing:

- The Sametime Server V3.0
- The fix pack
- Sametime Server Mobile extension

Note: Sametime Server can be installed only if the Lotus Domino Server has been installed. Sametime requires that the Domino Server be installed on the same system as the Sametime Server.

Installing Sametime V3.0 Server

1. Run **setup.exe** in the Sametime Server installation CD.
2. This will bring up the welcome window. Click **Next**.This will bring up the following window. Click **Yes**.

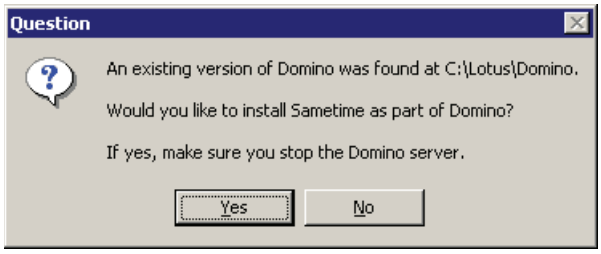


Figure H-12 Question

3. This will bring up the window asking about your path settings. Click **Next**. At the completion of the installation, you will be shown an Installation Complete window. Click **Finish** to end the installation.
4. Once the installation is complete, you will have to configure the Sametime Server. You will be prompted to enter the Domino Server ID file. This file will be created during the Domino Server installation. Open the folder C:\lotus\domino\data and select the **server.id** file.

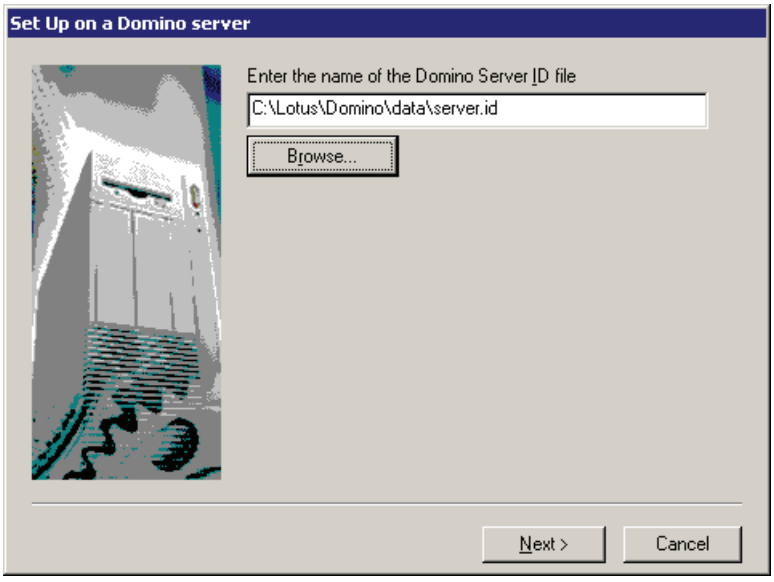


Figure H-13 Domino Setup

5. Select the type of directory server you want to use. You can let Domino manage the users or could configure any other LDAP server. In our installation we let Domino manage users. Click **Next**.
6. Choose if you want HTTP tunnelling. In our Domino installation we did not enable HTTP tunnelling. Click **Next**.
7. This will begin the installation. A window is displayed when the installation is complete. Click **Finish**.

Note: To start the Sametime Server, start the Domino Server. Sametime Services will start automatically along with Domino Server.

Installing Sametime V3.0 Service Pack 1

After the installation of the Sametime Server, Sametime 3.0 Service Pack 1 has to be installed. There are no configuration steps in this installation.

Installing Sametime Mobile Extension V3.0

If you want mobile users to connect to the Sametime Server, Sametime Mobile Extension has to be installed. There is no configuration during the installation of the Sametime Mobile Extension. Sametime Mobile Extension has to be installed on the same system as the Sametime Server.

I

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG247015

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG247015.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
sg247017.zip	Zipped Code Samples

© Copyright IBM Corp. 2003. All rights reserved.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space: 13 MB minimum
Operating System: Windows
Processor: 1 GB or higher
Memory: 512 MB

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 1274. Note that some of the documents referenced here may be available in softcopy only.

- *RDB Synchronization, Transcoding and LDAP Directory Services in IBM WebSphere Everyplace Access Version 4.1.1,* SG24-6856
- *IBM WebSphere Portal V4 Developer’s Handbook*, SG24-6897
- *IBM Redbook IBM Replication Solutions for Pervasive Computing with DB2 Everyplace and DB2 Satellite Edition*, SG24-6217
- *IBM WebSphere Everyplace Server Service Provider and Enable Offerings: Enterprise Wireless Applications*, SG24-6519
- *Getting the Most From Your Domino Directory*, SG24-5986

Other publications

These publications are also relevant as further information sources:

- *IBM DB2 Everyplace Sync Server Administration Guide 7.2*, SC27-0845
- *DB2e Application Development Guide Version 8 Release 1*, SC18-7185
- *IBM DB2 Everyplace Sync Server Administration Guide Version 8 Release 1,* SC18-7186

Online resources

These Web sites and URLs are also relevant as further information sources:

- WebSphere Portal:
  `http://www.ibm.com/software/websphere/portal`

© Copyright IBM Corp. 2003. All rights reserved.

- WebSphere Application Server:
  http://www.ibm.com/software/webservers/appserv
- Lightweight Directory Access Protocol standard:
  http://www.ietf.org/rfc/rfc1777.txt
- WebSphere Studio:
  http://www.ibm.com/software/websphere/studio/
- WebSphere Transcoding Publisher:
  http://www.ibm.com/software/webservers/transcoding
- Lotus product information:
  http://www.lotus.com/home.nsf/welcome/products
- WebSphere Everyplace Suite:
  http://www.ibm.com/pvc/products/wes_provider/index.shtml
- Java Community Process, JSR 168:
  http://jcp.org/jsr/detail/168.jsp

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

access control 7, 47
Access Control List (ACL) 726
Access Control List details 56
Access Privileges 658
account 818
action event handling 229
ActionEvent 230
ActionEvent portlet 230, 232, 247
ActionListener 241, 246
actionPerformed 231
active credential vault 287
ActiveSync 1201–1202
ActiveSync 3.7 1202
add additional Domino 738
Add to group 652
Add users to group 30, 652
Additional Services 732
Address book 776
Administration 5, 7, 25
administrator 818
annotation 411, 427
  external 404, 416
  file 423
  instructions 416, 428
  internal 418
  language 418
  processing 405
Apache Tomcat 180
API 275
Application Developer 4
application development 1211
Application Server 2
application.xml 146
attachment filter 743
attachments 7, 743
ATTW 16
Audiovox Maestro 16
Audiovox Thera 16
Au-Systems 17

B

Blue Action 273
Bundle Base Name 276
business applications 130

C

C 643
Casio E200 16
cell phones 2
certificate 118
chat session 836
chatting 834
Compaq iPAQ 16
Concurrent Versioning System 185
Connected browsing 15
Contacts 816
Contacts portlet 767
container model 6
CORBA 288
Create DataPropagator Subscription wizard 677
Create filter 747
Create JDBC Subscription wizard 655, 659
Create new user 28
Create Subscription Set wizard 661
Create Upload Subscription wizard 686, 706
Credential Vault 287
  Active credential objects 289
  Administrative slots 289
  Credentials objects 289
  CredVaultBasicAuthACTIVE 292
  organization 289
  Passive credential objects 289
  Portlet private slots 289
  PortletService 288
  resource 289
  Shared slots 289
  System slots 289
  vault implementation 289
  Vault Segments 289
  Vault Slots 289
Criteria 10

D

data directory 732
data synchronization 8–9

© Copyright IBM Corp. 2003. All rights reserved.

database servers 94
database URL 698
Date received 745
DB2 Control Center 653
DB2 data source 694
DB2 Everyplace 9, 603, 647, 693
DB2 Everyplace database 15
DB2 Everyplace Sync Server 647
DB2 Everyplace Synchronization 622, 645
  Everyplace Client 645
  JDBC subscription 646
  MDAC 646
  Mobile Devices Administration Center 646
  overview 646
  relational database synchronization 646
  remote databases 645
DB2 V7.2 1211
db2admin 1222
DB2e applications
  capabilities 529
  Configurations 531
  Connected Device Configuration 531
  Connected Limited Device Configuration 531
  data synchronization 529
  Development tools 530
  Eclipse Workbench 530
  Foundation Profile 531
  frameworks 530
  Introduction 530
  J2ME 530
  J2ME architecture 530
  Java 2 Micro Edition 530
  Mobile Information Device Profile 531–532
  Overview 530
  Personal Profile 531
  Profiles 531
  runtime 610
  standalone mode 532
  WebSphere Micro Environment 532
  WebSphere Studio Application Developer 532
  WebSphere Studio Device Developer 529–530
DBA Studio 1175
deck 13
default device profile 741
default network profile 755
DefaultPortletMessage 254
Define Replication Subscription wizard 681
define subscription set 660
Define Upload Subscription wizard 688, 706

Delivery 10
Dell AXIM X5 16
deployment descriptor editor 240
descriptive name 748
desktop PC 1202
Device Management 1015, 1019, 1025
Device Manager 16, 1017
Device Manager client 127
Device Manager console 1018
Device Plug-ins 1017
Device position 11
device profile 751, 753, 816–817
Device Services 6, 12
  Administration portlets 1018
  Architecture 1016
  Client Agent 1017
  DM Console 1018
  Manage permission 1019
  overview 1016
  Palm 1016
  Palm plug-in 1017
  PocketPC 1016
  PocketPC plug-in 1017
  portlets 1017
  SyncML plug-in 1017
  Tivoli Device Manager 1016
  User Group 1019
  View permission 1019
Directory 11
Directory Manager 31
DM servlet 1017
Document Object Model 513
doEdit 231
Domino Adapter 729
Domino Administrator 733
Domino Mapping databases 736
Domino Server 711, 723, 1257–1258
doView 231
DSYLDAP.properties 650
dsysetjavahome.bat file 709
DTD editor 187, 456–457

E

EAR file 146
Eclipse 180
Edge Server 2
Edit icon 246
edit mode 246

Edit my profile 283
E-mail filter 742
enterprise applications 6
Enterprise JavaBeans 181
Enterprise Resources 2
ESS 8, 733, 740
ESS Admin portlets 742
Eudora 17
event-processing phase 253
Events 816
Everyplace Access 1
Everyplace Client 2, 4, 14, 81, 648
  Active network profile 104
  application 1058
  CD 92
  configuration 101
  database information 92
  Database synchronization 82
  E-mail Synchronization 101
  external device 91
  Home page 103
  HotSync 88
  installation 82
  IPK 107
  Launch 102
  main window 93
  My Settings 104
  Network profiles 94
  offline portal pages 82
  Palm Desktop software 82
  Palm OS 82
  Palm OS 5.2 93
  Password 101
  PDAs 82
  Personal Digital Assistants 82
  PIM 92
  Pocket PC 100
  Pocket PC 2002 95
  Portal Server 100
  Settings page 94
  SSL support 94
  synchronization 93
  USB connection 105
  User name 101
  Wizard 87
  Zaurus devices 105
  Zaurus File Transfer 107
  Zaurus Install Package 107
Everyplace Synchronization Server 711, 754, 810

Everyplace Toolkit 4, 19, 179, 182–183, 1211
ExportResources 386
Extensible Markup Language 384, 441
Extensible Stylesheet Language 441

F

Filtering 7
filtering options 816
Filters settings 819
first synchronization 752
form creation wizard 629
Form submission 16
forms 12
forms data 634
Fragmentation 13
    common problems 516
    example 516
    fragmentation transcoder 513
    how it works 513
    i-mode (cHTML) 514
    WML 513
fragmentor 513

G

Geocoding 11
geographical coordinates 11
Global Database Name 1152
Global filters 743
Group membership 7
groups 7, 647

H

Hints and tips 372, 645, 691, 710, 806, 828, 1075
host variables 634
HotSync 88
How portlets work 137
HP Jornado 56x 16
HTML 13
    Document Object Model 513
HTML Annotation Editor 420
HTML document 414
HTML form 230
HTTP 648
HTTP Basic Authentication 291
http://localhost/wps/myportal 244, 271

I
IBM J9 JVM 609
IBM SecureWay Firewall 24
IBM Tivoli Access Manager 14
IKEYMAN 118, 124
i-mode 511, 514
ImportCocoon 386
ImportResources 386
ImportResources command 414
information source 9–10
INS application programming 844
Instant messaging
alert preferences 830
components 829
Desktop user 835
Login information 833
Lotus Domino Server 830
Lotus Sametime Mobile Extension 830
Overview 830
Palm 831
PocketPC 835
Sametime 3.0 client 830
Sametime Client 829
Sametime Messaging 831
Sametime Server 832
Sample scenario 830
STMobile 830
Tools 835
users 834
Intelligent Notification Services 6, 9
851
adapters configuration 910
Administration 841
administration servers 846
Components 841
Concepts 840
Configure 845
Content 840
Content Adapter 842
Context drivers 843
context information 844
Custom gateway adapters 844
Custom Transcoder 862
delivery channel 840
Deployment 865
Development 854
Directory Services Engine 841
Dispatcher 840
e-mail 840

Exporting the portlet 875
gateway adapter 842, 853
implementation 852
instant messaging transcoder 842
Manage subscriptions 844
Overview 840
portlet configuration 846
portlets 840
Privacy Manager 844
Publish content 844
Sametime 840
sample code 844, 854
sample scenarios 839
Scenario 852
Secure Context Server 843
Send notifications 844
server configuration 908
Short Messaging Service 840
Simple Message Transfer Protocol 842
SMS 840
SMTP 840
Subscribe to content 844
subscription configuration 908
Subscription Manager 840
subscription portlet 892
transcoders 842
trigger 844
trigger handler 883
Trigger Manager 842
Universal Notification Dispatcher 842
WAP 840
Wireless Application Protocol 842
internal annotation 405, 412, 414
Internet directory server 1267
Internet Explorer 414
Internet Explorer - External annotation 434
Internet Protocol 1219
Invitations 8
iPAQ 5400 16

J

J2EE 6, 181
J2ME development 1058
J9 on WinCE 643
JAAS 14
Java 3
Java APIs 11
Java debugging 185

JavaDoc 181
JavaServer Pages 180
javax.servlet.http.HttpServlet class 137
JDBC database 5
JDBC Subscription 655, 659
JDK 1.3 181
JSP 1.2 181
JSP tags 275
JVM 643

K

keep elements 410
key database 117

L

LDAP 31
Load Balancing 2
Local debug 155, 183
Location Aware Service 11
Location Aware Services 6, 11, 971
  Add 977
  Add new Location 987
  administration portlets 977, 979
  API 971
  application providers 971
  Architecture 974
  authentication 972
  Configuration 977
  configuration profile 977
  coverage region 976
  development 996, 1008
  Device Position 972
  directions 989
  Directory 974
  Disable 977
  Dynamic Service Binding 973
  Enable 977
  error handling 975
  Errors 1013
  Geocoding 974
  geographical coordinates 974
  Go2Map 975
  IBM Location Server 975
  implementations 973
  Information 1013
  Installation 977
  Interface translation 975
  Introduction 972

LIF 975
location services 976
location-based services 971
locations 974
MapInfo 975
Mapping 974
persistence 987
Portlet Service 975
properties 977, 983
Provider Differentiation 973
Provider Transparency 972
quality of service 973
Remove 977
Reverse-geocoding 974
Routing 974
service 984
service adapters 972
service availability 977
Service Binder 976
service binder object 976
service configuration profile 983
service factory 976
service failure 976
Service invocation 975
Service Monitor 976
Service portlet 985
service provider 983
service providers 975
Service Registry 976
Tracing 1013
transparency 973
Troubleshooting 1013
Verifying 977
Warning 1013
Webraska 975
log in 26
Lotus 840
Lotus Domino Server 711
LTPA cookie 288

M

Mail portlet 762
mailbox name 818
Manage Markups 444
Managing Users and Groups 27
MANIFEST.MF 149
Mapping 11
markup language 13, 449

Markups 145
cHTML 145
HTML 145
PDA markup 145
VoiceXML 145
WML 145
MDAC 650, 662
MDAC screen 704
members 815
Membership 29, 652, 814
Memos 816
Message Center portlet 10
Message events 230, 253
MessageListener 253–254
MessageReceiver 254, 262, 268
MessageReceiverBean 281
messaging 829
messaging portlets 134
Messaging Services 6
Microsoft ActiveSync 3.7 1202
Microsoft Exchange 816
Microsoft Exchange 2000 Adapter 810
Microsoft Exchange 2000 Server 817
Microsoft Exchange servers 810
Microsoft Exchange V5.5 Adapter 810
Microsoft Exchange V5.5 servers 809
mirror database 647, 699
Mobile Application Builder 604, 608, 614
Mobile client 12
Mobile client devices 2
Mobile computing 3
mobile computing solutions 4
Mobile device 12
mobile devices 7, 12–13, 441, 647
Mobile Devices Administration Center 650, 654
Mobile Extension 1270
Mobile ITSO Users 752
mobile users 3
Model-View-Controller 137, 194
Multiple Device Authoring Technology 184
multiple network profiles 104
MVC methodology 194

N

native mail client 823
native mail program 823
NEC MobilePro 300 16
network 3

Network (connectivity) 2
Network and Dial-up Connections 1219
Network Profiles 104
News server 1267
NLS 275
  Accessing NLS bundles from JSPs 280
  Bundle Base Name 276
  bundles 276
  Key 276
  readystatus 279
  receivedstatus 279
  resource bundles 277
  viewmode 279
NLSLab.properties 277
Nokia Mobile Internet toolkit 415
Notebook Entry 769
Notes 816
Notes Contacts Portlet 764
Notes Mail portlet 763
Notes Notebook 769
notification criteria 9
NTT DoCoMo 504i 16

O

offline Domino applications 648
Offline Portal Browsing and Forms 12
offline Portal content 648
online content 1181
Oracle
  back-end database 695
  data source 698
  Database URL 698
  DataPropagator subscription 695
  Event Message 696
  filter 702, 707
  hints and tips 710
  IBM DB2 Everyplace Server 710
  JDBC 695
  JDBC driver 695
  mirror database 699
  Mobile Devices Administration Center 695
  one-way data transfer 705
  replication interval 704
  source database 704
  subscription 704
  tables 703, 707
  test connection 710
  upload changes 705

Upload synchronization 695
Oracle data source 694
Oracle database
Hints and tips 710
JDBC driver 695
JDBC subscription 696
subscription 695
Test Connection 699
upload subscription 705
Oracle Enterprise Edition 1145
Oracle8i Enterprise Edition 1146
Order column 1170
org.apache.jetspeed.portlet 240
Outline view 421

P

Page Aggregation 4
Page Designer 186, 406, 414
Palm applications 627
Palm client 823
Palm Desktop 606
Palm device 124, 779, 823, 830, 1178
Palm OS 3.5 16
Palm OS 4.0 16
Palm OS 5.2 device 756, 761
Palm OS 5.X 1177
Palm OS V5.2 Simulator 271
Palm SDK 606
Palm SDK V5.0 606
Palm Simulator 1177
Configuration 1180
Overview 1178
software 1178
Usage 1181
Palm Tungsten C 17
Palm Tungsten T 17
Palm user 836
Palm Web Pro Browser 17
PalmSim.exe 1178
PalmSource Web Browser 2.0 17
parent synchronization group 651
Password 741
PDAs 2, 4, 14, 82, 604, 648
Personal Information Management 711
Phone Edition 16
Pilrc 606
PIM 5, 711
PIM and E-mail Synchronization 7–8

Address Book 715
Admin ID 713
Administrator 717
Administrator client 726
Administrator group 725
Basics 717
client device 714
Configuration 714
Create new databases 719
Database Folder Path 763
databases 713
Domino Server 712–713, 715
E-mail data 713
ESS 712
Everyplace Access 712
Everyplace Synchronization 712
Exchange 2000 809
Get locales 753
How It Works 712
iNotes Web Access 723
mail database 714
Manager 734
New Contact 767
Notes Client 729
Overview 712
Palm 811
PocketPC 811
restart 715
synchronization demonstration 761
Synchronization feature 736
updates 714
user 721
WEA Server 714
Welcome screen 730
wpsadmin 812
PIM components 780
PIM portlets 761, 765–766
PIM Server 758
platform installation 1211
Pocket PC 779, 830, 1203
development 1201
Everyplace Client API 1057
External annotation 437
Status command 1059
Stop command 1059
stylesheet 453
Sync command 1058
synchronization 1057
Pocket PC 2002 16, 1057–1058

Pocket PC API
Commands available 1058
Examples 1059
Policy Directory 14
port number 832
Portal 6
Portal Administration 444, 650
Portal Administrator 737
Portal application 132
portal browsing 12, 16
portal session 413
portal settings 7
Portal technology 4
Portal Toolkit 183
portlet 414
Portlet API 230
Portlet application name 144
portlet application wizard 140
Context root 142
Enterprise project 142
Portlet location 141
Project name 141
Types of portlet projects 143
portlet applications 132
Portlet Container 4
Portlet projects 182
Portlet terminology 132
Action events 133
Configure 134
default portlet message object 134
Edit 134
Help 134
Maximized 135
Message events 133
Minimized 135
Normal 135
Portal Application 132
Portal Page 132
Portlet 132
Portlet API 132
Portlet Events 132
Portlet Messaging 133
Portlet Modes 134
Portlet States 135
View 134
Window events 133
portlet WAR file 442
Portlet.xml 151
portlet.xml 151, 238

PortletAction 230
PortletApplicationSettingsAttributeEvent 230
PortletException 276
portlet-level transcoding 377
PortletMessage 268
portlets 6–7, 130
  ActionListener interface 240
  API 132
  application programming interface 132
  application projects 129
  Basic type portlet 232
  custom messages 134
  databases 130
  debugging 129
  HTML-based 131
  messaging 133
  modes 129
  Palm 131
  PIM 131
  Pocket PC 131
  Portal Application 132
  Portal Page 132
  portlet application 270
  portlet code 237
  projects 129
  sequence flow 231
  states 129
  terminology 132
  Testing 129
  view 134
  View mode 270
  WAR file 237
  Web sites 130
  wizard 129
  WML 131
PortletSettingsAttributeEvent 230
PPP connection 1019
preferences 9, 758
Preview view 422
primary key 1170
Priority 748
Privacy Manager 844
private keys 288
process level token 1217
profile database 10
profiles 816, 818
proxy installation 106
Proxy Server 1267

Q

QuickLinks 58

R

random password 1263
receiver portlet 254
Receiving a message 268
Red Action 272
Redbooks Web site 1274
  Contact us xx
Refresh 760
Registration queue 726
relational database adapter 652
relational database synchronization 653
Reminder 58
Remote Attach 182
Remote Display Control 1201
remote Oracle database 709
Remote Single Server 182
remove an existing Domino 739
removing columns 410
replication frequency 659
Replication Subscriptio 681
Request Viewer 387, 390–391, 401
Reverse geocoding 11
Routing 11
RSS Portlet 58
Run on Server 281

S

Sametime 10
Sametime Connect 16
Sametime installation 1257
Sametime Server 1268
sample database 1166
sample scenarios 406
  YourRedbookNews 419
Sample stylesheets 451
schema 658
Schema object 1168
scripting 643
Search Results 653
Secure Sockets Layer
  Binary DER Data 125
  certificate authority 118
  configuration 120
  DB2 Everyplace 126
  Device Manager client 127

Enabling SSL 122
Everyplace Client 115
HTTP server 116
IBM Agent 127
IKEYMAN 116
Key database 116
key-pairs 116
mobile devices 115
New Self-Signed 118
Palm 122, 124
Personal Certificates 125
PocketPC devices 122
self-signed key file 118
Stash the password to a file 117
SyncML client 127
Virtual Hosts 121
WEA Client 116
Web Cache Client 128
WebSphere Application Server 121–122
security 7, 14, 121, 290
selected users 815
selection 10
Selection Criteria 635
send a message 255
Send Invitations 778
Server Configuration dialog 235
Server Initiated Actions 11
administrative portlet 958
administrator 960
All synchronizations 958
Client processing 961
data flow 960
DB2e synchronization 958
delivery channels 957
DMS synchronization 958
e-mail synchronization 958
gateway adapter 958, 960
installation 962
log files 969
Notes content adapter 958
notification 960
Offline synchronization 958
Overview 958
Palm support 959
PIM synchronization 958
Pocket PC 959, 968
Pocket PC 2002 Phone Edition 959
providers 958
Sametime notifications 959

settings 962
SMS 959, 961
SMS notifications 959
sources 958
synchronization request 961
user delivery channel 958
Server IP field 759
Servlet 2.3 181
Servlet Container 6
Servlets 180
Sharp Zaurus 17, 615
Siemens SX56 16
Siemens SX56(HTC) 18
simple database 1160, 1166
Simple notifications 9
Simple Object Access Protocol 186
Site Developer Advanced 4
Siteminder 6
Skip 760
Source view 421
SSL 115, 647
SSL client certificates 288
Standard Mail 823
Struts 180
Studio Device Developer 608
Studio Site Developer 1211
stylesheets
configure a portlet 442
Configuring 442
Internet Explorer stylesheet 451
ITSO News XML Feed 449
multiple stylesheets 443
overview 442
PortletFilterService.properties file 443
processing 442–443
RSS data stream 446
sample scenario 449
wtp-condition statements 449
XML documents 441
XML Feed 446
XMLFeed application 446
XMLFeed portlet 448
subscribers 11
subscription 647, 653, 697, 706
Subscription based notifications 9
Subscription Manager 1018
subscription set 647, 653, 700, 706
subscription settings 9
subscription wizard 659

Symbian 615
Sync 760
Sync Server 9
sync servlet 648
Synchronization 647, 779, 824
  Configuration 711, 715
  Create Adapter 721
  Hints and Tips 711
  How It Works 711
  options 824
  Overview 711
  scenarios 766
  Troubleshooting 711
  Usage 711
synchronization group 652–653
Synchronization Server 648, 739
SyncML 8, 18
SyncML 1.1.1 811
SyncML client 127, 758, 823–824
SyncML mailbox 784

T

taglib 242
tags 406
target devices 449
target portlet 257
Tasks 747
technologies 3
Test Environment 255
Tested devices 16
Testing and debugging portlets 154
  Local debug 155
  Remote Attach 154
  Remote Single Server 155
text clipping
  annotation 404
  annotation editor 404
  annotation instructions 404
  annotation language 418
  condition 418
  external annotations 404
  HTML document 403
  HTTP header 403
  internal annotations 404
  markup action 418
  modified annotation 414
  sample scenario 414
  target node 418

text replacement 408
Theme default 58
Themes and Skins 5, 56
Tivoli Access Manager 6
To Do List portlet 770
token object 1217
Toshiba e740 16
Tracing files 396
transcoding 5, 414
Transcoding Technologies 13, 377–378, 403
  a simple portlet 396
  aggregator 380
  annotation transcoding plug-in 385
  annotations 403
  annotators 378–379, 384
  architecture 379
  column 404
  device 383
  Disable resources 386
  document clipping 403
  Document Object Model (DOM) 385
  Enable resources 386
  enable transcoding 396
  Export resources 386
  Extensible Markup Language 378
  external annotation 385, 403–404
  field 404
  fragmentation transcoding plug-in 385
  HTML pages 378
  HTTP header 384
  image transcoding plug-in 385
  Import resources 386
  insertattribute 404
  inserthtml 405
  internal annotation 384, 403–404
  keep 404
  logging 394
  markup language conversions 379
  message files 395
  monitor the process 401
  option 404
  plug-in 385
  Portal page level transcoding 379
  portlet filter 380
  Portlet-level transcoding 379
  preference profiles 378, 381
  profile 383
  remove 404
  replace 405

replacewithhtml 405
Request Viewer 387
row 404
sample application 407
setpreference 405
splitpoint 405
stylesheets 379
text transcoding plug-in 385
tracing 394
tracing records 396
transcoding plug-in 378, 385
troubleshooting data 396
user 383
user agent 380
XML config tools 386
XML stylesheets 378, 384
Transcoding Technology 13
TrueSync Plus 14

U

Universal Description Discovery and Integration 181, 186
upload subscription 705
USB port 1203
User 647
user customization 7
user ID 7
User name 741
users 7
users and groups 653
users's name 7

V

Vault Slots 289
Verizon 16
VersaMail 756
VersaMail 2.0 823
Visual Editor for Java 185
Voice Application Access Toolkit 145
VoiceXML 183

W

WAP browser 400
WAP/IP connection 519
WAR modules 147
WEA Client 93
Web Clipping 7, 13
Web content 12

Web deployment descriptor 153
Web development tools 185
Web page 410
Web Pro 17
Web Service Inspection Language 181
Web services 180
Web Services Description Language 186
web.xml 153
WEB-INF 150
WebSphere Application Server 3, 1211
WebSphere Edge Server 23
WebSphere Everyplace Access 1–2, 4, 404
WebSphere Everyplace Connection Manager 23
WebSphere MQ Everyplace 23
WebSphere Portal 3, 24, 650, 810
WebSphere Studio 4
WebSphere Studio Application Developer 181
WebSphere Studio Enterprise Developer 181
WebSphere Studio Page Designer 408
WebSphere Studio Site Developer 179–180, 1211
  editors 188
  help 190
  Mobile application development 129
  perspectives 188
  tasks area 189
  views 188
  Web perspective 188
  Workbench 187
  Workbench user interface 188
Win32 emulation 618
WindowEvents 230
Windows CE 615
  applications 617, 628
  devices 608
  JVM 608
wizards, using to create simple databases 1160
wizards, using to create simple tables 1166
WML 1.3 183
WML browser 414
  external annotation 435
WML document 415
World Clock 58
WP portlets for ESS 811
wpsadmin 244, 271, 282

X

XHTML 181
XML 3, 13

XML content 5
XML dialect 441
XML editor 186, 456, 470
XML schema editor 187
XML to XML Mapping Editor 456, 497
XML Tools 441, 456
XPath Expression Wizard 456
XSL 13
XSL Debug and Transformation Tool 456
XSL editor 187, 456, 479
XSL stylesheets 384

Z

Zaurus 105
Zaurus 5600 17
Zaurus Manager 105
Zaurus USB driver 105



(2.5” spine)
2.5”<->nnn.n”
1315<-> nnnn pages



WebSphere Everyplace Access Version 4.3 Handbook for Developers



Redbooks

Extend your e-business applications to PDAs, phones, and other device types

Develop mobile applications with data synchronization

Adapt content to multiple markup languages

This IBM Redbook helps you plan and develop mobile applications to run in a WebSphere Everyplace Access Version 4.3 environment. The information provided in this redbook targets business-to-employee (B2E) enterprise applications, but most of the scenarios presented will apply to business-to-consumer (B2C) applications as well. In this redbook, you will find step-by-step examples and scenarios showing ways to integrate your enterprise applications into a WebSphere Everyplace Access environment using the WebSphere Studio Site Developer and the Everyplace Toolkit as well as extending your portlet capabilities to use other advanced functions such as Transcoding Technology. You will also find numerous scenarios describing recommended ways to develop and implement server-initiated actions, Everyplace Client security and others. This redbook includes sample scenarios describing ways to implement Intelligent Notification Services to notify Lotus Sametime, SMTP e-mail, WAP devices and others. It also includes scenarios illustrating the new Location Aware Services (LAS) function to provide access to location-based services from multiple vendors using the available APIs. Sample scenarios will also include offline applications using Offline Portal Content and Forms, DB2e applications and Relational Database Synchronization with JDBC back-end databases, PIM and e-mail synchronization with Domino and Microsoft Exchange servers.

A basic knowledge of Java technologies such as servlets, JavaBeans, EJBs, JavaServer Pages (JSPs), as well as XML applications and the terminology used in Web publishing, is assumed.

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

SG24-7015-01 ISBN 0738498750