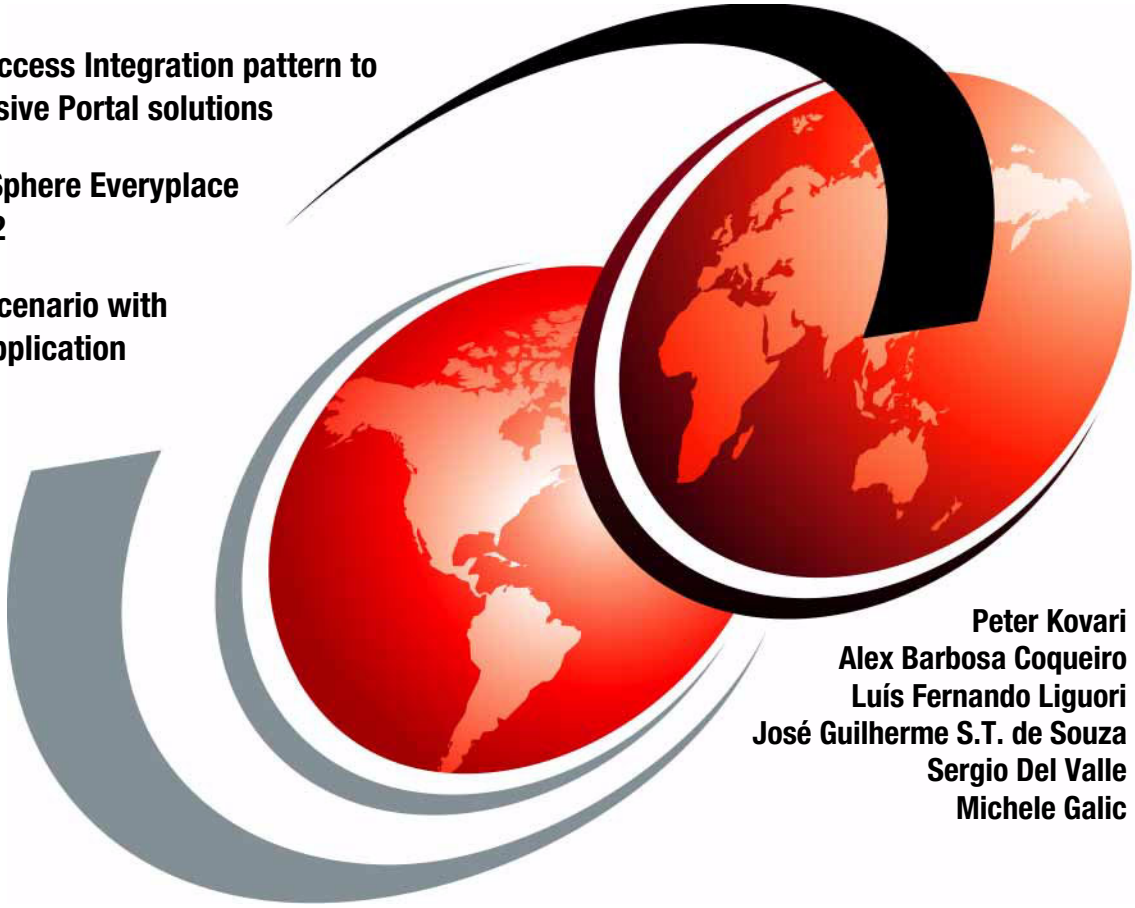
WebSphere® software

IBM

Patterns: Pervasive Portals
Patterns for e-business Series

- Using the Access Integration pattern to build Pervasive Portal solutions
- Using WebSphere Everyplace Access V4.2
- Technical scenario with a sample application



Peter Kovari
Alex Barbosa Coqueiro
Luís Fernando Liguori
José Guilherme S.T. de Souza
Sergio Del Valle
Michele Galic

Redbooks

ibm.com/redbooks

International Technical Support Organization

Patterns: Pervasive Portals
Patterns for e-business Series

September 2003

SG24-6876-00

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

First Edition (September 2003)

This edition applies to WebSphere Everyplace Access V4.2 for use with Windows 2000 Server and some of the components on AIX 4.3.3 and Sun Solaris; WebSphere Portal Server 4.1 Enable on Windows 2000 Server.

© 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

© Copyright IBM Corp. 2003. All rights reserved.

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.

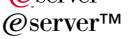
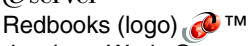
This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used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

Redbooks (logo) ™	DB2®	Redbooks™
developerWorks®	Everyplace®	Sametime®
ibm.com®	Informix®	SecureWay®
iNotes™	IBM®	Tivoli Enterprise™
pSeries™	Lotus Notes®	Tivoli Enterprise Console®
z/OS®	Lotus®	Tivoli®
AIX®	Mobile Notes®	WebSphere®
CICS®	MQSeries®	Word Pro®
Domino™	NetView®	
Domino Designer®	Notes®	
DB2 Universal Database™	PAL®	

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 discusses the Access Integration pattern. The book is a valuable source for IT architects, IT specialists, application designers, application developers and consultants who wish to know more about pervasive solutions. The application framework for this book includes WebSphere® Portal and WebSphere Everyplace® Access.

Part 1 of the redbook guides you through the process of choosing the Business and Integration patterns of the Pervasive solution and then drills down to the Application and Runtime patterns and Product mapping to deliver the desired functionality for a pervasive solution.

Part 2 provides guidelines for pervasive applications, including application design and development, and some of the non-functional requirements for such applications, including security, system management and performance.

Part 3 demonstrates how to set up and configure a system for the sample application presented in this book.

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.

© Copyright IBM Corp. 2003. All rights reserved.

The Team (José, Luís, Alex, Sergio, Peter)

Peter Kovari is a WebSphere Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively about all areas of WebSphere. His areas of expertise include e-business, e-commerce, security, Internet technologies and mobile computing. Before joining the ITSO, he worked as an IT Specialist for IBM in Hungary.

Alex Barbosa Coqueiro is a Software Engineer in IBM Business Consulting Service in Brazil. He received a degree in Computer Technology in Fatec, Sao Paulo and has been working at IBM for two years. He has over eight years of experience in object-oriented development and is certified by Sun as an Architect (SCEA), Web Developer (SCWD) and Programmer (SCJP) for Java™. His areas of expertise include WebSphere Application Server, WebSphere Portal Server and WebSphere Content Publisher. Currently, he is working toward a Master's Degree in Software Engineer at USP.

Luís Fernando Liguori is a certified Consulting IT architect in the e-business Technical Sales group in Brazil and has been working at IBM for nine years. He has worked in many customer engagements and his areas of expertise include e-business infrastructure, Internet solutions, patterns, wireless and security. Luís worked as an IT Architect in the last three IT waves in the Client-Server, Networking Computing and e-business departments.

José Guilherme S.T. de Souza is a Senior IT Architect responsible for Wireless Solutions in Latin America. He joined IBM two years ago to develop solutions and advise enterprises and services providers on wireless technologies. His current areas of expertise include patterns for e-business, telecommunications systems and technologies, mobile e-business solutions and ROI studies. He has

more than 18 years of working experience in IT. Before joining IBM, he worked with ExxonMobil for ten years as an IT Infrastructure Project Manager. He holds degrees in Computer Engineering and MBAs in Finance and Telecommunication Management.

Sergio Del Valle is an IT specialist at Synerg-e Consulting in Mexico; his areas of expertise include pervasive applications using IBM/lotus Wireless products.

Michele Galic is a WebSphere Specialist at the International Technical Support Organization, Raleigh Center. Her focus is on the WebSphere family of products and Patterns for e-business. She has 13 years of experience in the IT field. She holds a degree in Information Systems. Before joining the ITSO, she was a Senior IT Specialist in IBM Global Services in the Northeast, specializing in the WebSphere field.

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

International Technical Support Organization, Raleigh Center

Cecilia Bardy
Gail Christensen
Carla Sadtler
Margaret Ticknor
Jeanne Tucker

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

Jonathan Adams, IBM UK, Software Group Technical Strategy
Marshall Lamb, IBM US, WebSphere Portal Business Portlet Development
Jennifer Lanier, IBM US, WebSphere Transcoding Publisher Development

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, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an 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

Part 1

Patterns for e-business

© Copyright IBM Corp. 2003. All rights reserved.

1

Introduction

This IBM Redbook is part of the Patterns for e-business series. In this introductory chapter, we provide an overview of how IT architects can work effectively with the Patterns for e-business.

The role of the IT architect is to evaluate business problems and to build solutions to solve them. To do this, the architect begins by gathering input on the problem, an outline of the desired solution, keeping in mind any special considerations or requirements that need to be factored into that solution. The architect then takes this input and designs the solution. This solution can include one or more computer applications that address the business problems by supplying the necessary business functions.

To enable the architect to do this better each time, we need to capture and reuse the experience of these IT architects in such a way that future engagements can be made simpler and faster. We do this by taking these experiences and using them to build a repository of assets that provides a source from which architects can reuse this experience to build future solutions, using proven assets. This reuse saves time, money and effort and in the process helps ensure delivery of a solid, properly architected solution.

The IBM Patterns for e-business help facilitate this reuse of assets. Their purpose is to capture and publish e-business artifacts that have been used, tested, and proven. The information captured by them is assumed to fit the majority, or 80/20, situation.

© Copyright IBM Corp. 2003. All rights reserved.

The IBM Patterns for e-business are further augmented with guidelines and related links for better use.

The layers of patterns plus their associated links and guidelines allow the architect to start with a problem and a vision for the solution, and then find a pattern that fits that vision. Then, by drilling down using the patterns process, the architect can further define the additional functional pieces that the application will need to succeed. Finally, he can build the application using coding techniques outlined in the associated guidelines.

1.1 The Patterns for e-business layered asset model

The Patterns for e-business approach enables architects to implement successful e-business solutions through the re-use of components and solution elements from proven successful experiences. The patterns approach is based on a set of layered assets that can be exploited by any existing development methodology. These layered assets are structured in a way that each level of detail builds on the last. These assets include:

- Business patterns that identify the interaction between users, businesses, and data.
- Integration patterns that tie multiple Business patterns together when a solution cannot be provided based on a single Business pattern.
- Composite patterns that represent commonly occurring combinations of Business patterns and Integration patterns.
- Application patterns that provide a conceptual layout describing how the application components and data within a Business pattern or Integration pattern interact.
- Runtime patterns that define the logical middleware structure supporting an Application pattern. Runtime patterns depict the major middleware nodes, their roles, and the interfaces between these nodes.
- Product mappings that identify proven and tested software implementations for each Runtime pattern.
- Best-practice guidelines for design, development, deployment, and management of e-business applications.

These assets and their relation to each other are shown in Figure 1-1 on page 6.

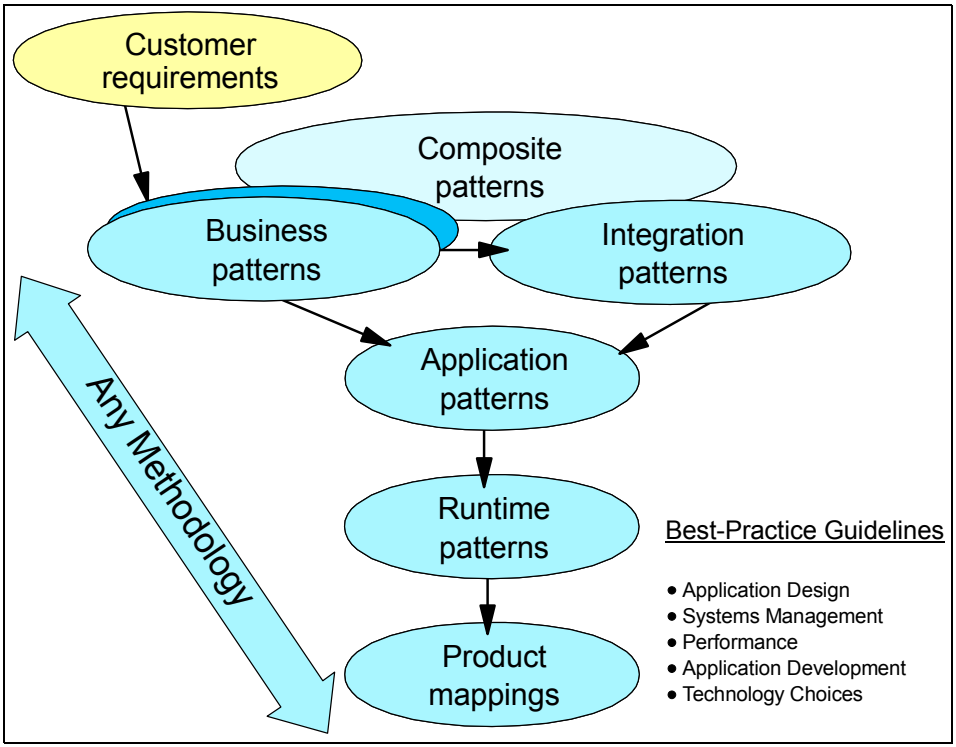


Figure 1-1 The Patterns for e-business layered asset model

Patterns for e-business Web site

The patterns Web site provides an easy way of navigating top down through the layered patterns' assets in order to determine the preferred reusable assets for an engagement.

For easy reference to Patterns for e-business, refer to the Patterns for e-business Web site at:

`http://www.ibm.com/developerWorks/patterns/`

1.2 How to use the Patterns for e-business

As described in the last section, the Patterns for e-business are a layered structure where each layer builds detail on the last. At the highest layer are Business patterns. These describe the entities involved in the e-business solution.

Composite patterns appear in the hierarchy shown in Figure 1-1 on page 6 above the Business patterns. However, Composite patterns are made up of a number of individual Business patterns, and at least one Integration pattern. In this section, we discuss how to use the layered structure of Patterns for e-business assets.

1.2.1 Selecting a Business, Integration, or Composite pattern, or a Custom design

When faced with the challenge of designing a solution for a business problem, the first step is to take a high-level view of the goals you are trying to achieve. A proposed business scenario should be described and each element should be matched to an appropriate IBM Pattern for e-business. You may find, for example, that the total solution requires multiple Business and Integration patterns, or that it fits into a Composite pattern or Custom design.

For example, suppose an insurance company wants to reduce the amount of time and money spent on call centers that handle customer inquiries. By allowing customers to view their policy information and to request changes online, the company will be able to cut back significantly on the resources spent handling this by phone. The objective is to allow policyholders to view their policy information stored in legacy databases.

The Self-Service business pattern fits this scenario perfectly. It is meant to be used in situations where users need direct access to business applications and data. Let's take a look at the available Business patterns.

Business patterns

A Business pattern describes the relationship between the users, the business organizations or applications, and the data to be accessed.

There are four primary Business patterns, as shown in Figure 1-2:

Business Patterns	Description	Examples
Self-Service (User-to-Business)	Applications where users interact with a business via the Internet or intranet	Simple Web site applications
Information Aggregation (User-to-Data)	Applications where users can extract useful information from large volumes of data, text, images, etc.	Business intelligence, knowledge management, Web crawlers
Collaboration (User-to-User)	Applications where the Internet supports collaborative work between users	E-mail, community, chat, video conferencing, etc.
Extended Enterprise (Business-to-Business)	Applications that link two or more business processes across separate enterprises	EDI, supply chain management, etc.

Figure 1-2 The four primary Business patterns

It would be very convenient if all problems fit nicely into these four slots, but reality says that things will often be more complicated. The patterns assume that most problems, when broken down into their most basic components, will fit more than one of these patterns. When a problem requires multiple Business patterns, the Patterns for e-business provide additional patterns in the form of Integration patterns.

Integration patterns

Integration patterns allow us to tie together multiple Business patterns to solve a business problem. The Integration patterns are outlined in Figure 1-3 on page 9.

Integration Patterns	Description	Examples
Access Integration	Integration of a number of services through a common entry point	Portals
Application Integration	Integration of multiple applications and data sources without the user directly invoking them	Message brokers, workflow managers

Figure 1-3 Integration patterns

These Business and Integration patterns can be combined to implement installation-specific business solutions. We call this a Custom design.

Custom design

We can represent the use of a Custom design to address a business problem through an iconic representation as shown in Figure 1-4.

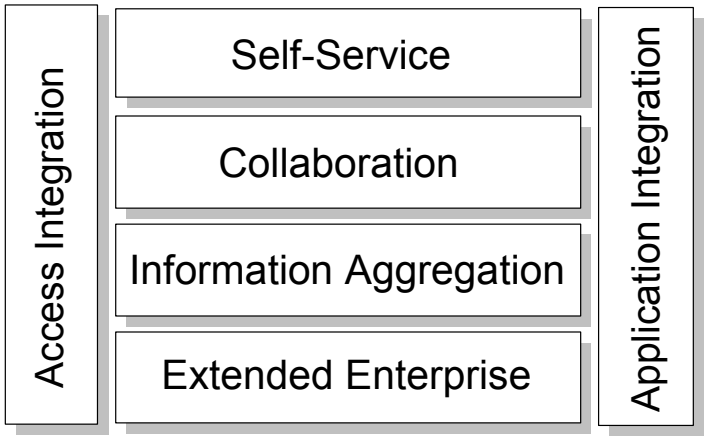


Figure 1-4 Patterns representing a Custom design

If any of the Business or Integration patterns are not used in a Custom design, we can show that with blocks lighter than the others. For example, Figure 1-5 on page 10 shows a Custom design that does not have a Collaboration business pattern or an Extended Enterprise business pattern for a business problem.

Figure 1-5 Custom design with Self-Service, Information Aggregation, Access Integration and Application Integration

A Custom design may also be a Composite pattern if it recurs many times across domains with similar business problems. For example, the iconic view of a Custom design in Figure 1-5 can also describe a Sell-Side Hub composite pattern.

Composite patterns

Several common uses of Business and Integration patterns have been identified and formalized into Composite patterns. The identified Composite patterns are shown in Figure 1-6 on page 11.

Composite Patterns	Description	Examples
Electronic Commerce	User-to-Online-Buying	• www.macys.com • www.amazon.com
Portal	Typically designed to aggregate multiple information sources and applications to provide uniform, seamless, and personalized access for its users.	• Enterprise Intranet portal providing self-service functions such as payroll, benefits, and travel expenses. • Collaboration providers who provide services such as e-mail or instant messaging.
Account Access	Provide customers with around-the-clock account access to their account information.	• Online brokerage trading apps. • Telephone company account manager functions. • Bank, credit card and insurance company online apps.
Trading Exchange	Allows buyers and sellers to trade goods and services on a public site.	• Buyer's side - interaction between buyer's procurement system and commerce functions of e-Marketplace. • Seller's side - interaction between the procurement functions of the e-Marketplace and its suppliers.
Sell-Side Hub (Supplier)	The seller owns the e-Marketplace and uses it as a vehicle to sell goods and services on the Web.	www.carmax.com (car purchase)
Buy-Side Hub (Purchaser)	The buyer of the goods owns the e-Marketplace and uses it as a vehicle to leverage the buying or procurement budget in soliciting the best deals for goods and services from prospective sellers across the Web.	www.wre.org (WorldWide Retail Exchange)

Figure 1-6 Composite patterns

The makeup of these patterns is variable in that there will be basic patterns present for each type, but the Composite can easily be extended to meet additional criteria. For more information on Composite patterns, refer to *Patterns for e-business: A Strategy for Reuse* by Jonathan Adams, Srinivas Koushik, Guru Vasudeva, and George Galambos.

1.2.2 Selecting Application patterns

Once the Business pattern is identified, the next step is to define the high-level logical components that make up the solution and how these components interact. This is known as the Application pattern. A Business pattern will usually have multiple possible Application patterns. An Application pattern may have logical components that describe a presentation tier for interacting with users, an application tier, and a back-end application tier.

Application patterns break the application down into the most basic conceptual components, identifying the goal of the application. In our example, the application falls into the Self-Service business pattern and the goal is to build a simple application that allows users to access back-end information. The Application pattern shown in Figure 1-7 fulfills this requirement.

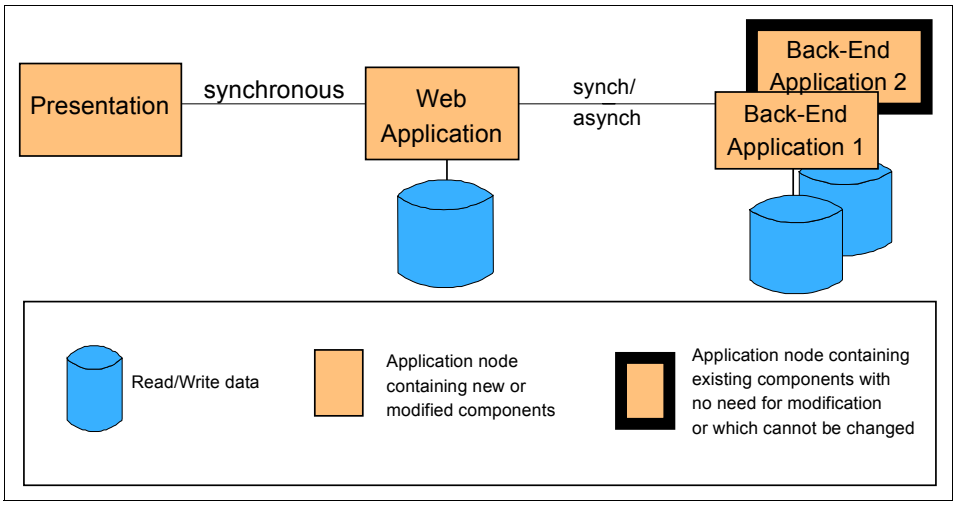


Figure 1-7 Self -Service::Directly Integrated Single Channel

The Application pattern shown consists of a presentation tier that handles the request/response to the user. The application tier represents the component that handles access to the back-end applications and data. The multiple application boxes on the right represent the back-end applications that contain the business data. The type of communication is specified as synchronous (one request/one response, then next request/response) or asynchronous (multiple requests and responses intermixed).

Suppose that the situation is a little more complicated than that. Let's say that the automobile policies and the homeowner policies are kept in two separate and dissimilar databases. The user request would actually need data from multiple, disparate back-end systems. In this case there is a need to break the request

down into multiple requests (decompose the request) to be sent to the two different back-end databases, then to gather the information sent back from the requests, and then put this information into the form of a response (recompose). In this case the Application pattern shown in Figure 1-8 would be more appropriate.

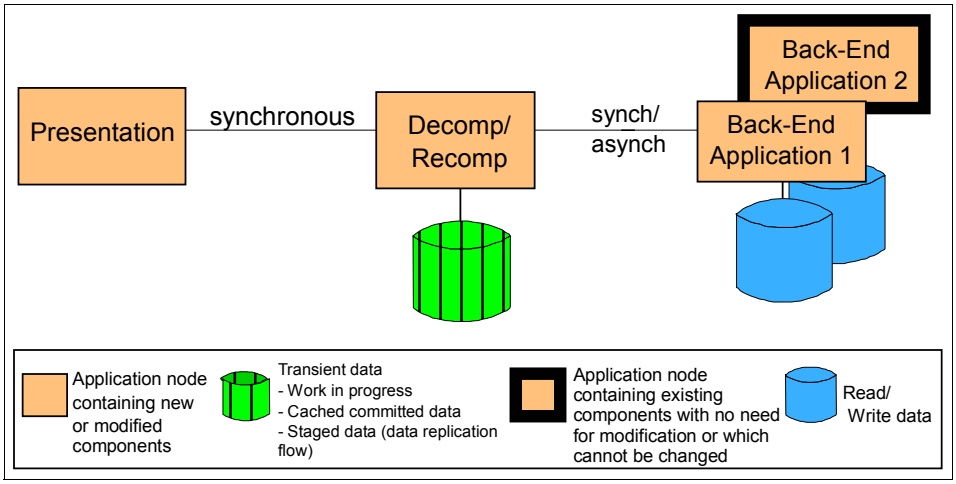


Figure 1-8 Self-Service::Decomposition

This Application pattern extends the idea of the application tier that accesses the back-end data by adding decomposition and recomposition capabilities.

1.2.3 Review Runtime patterns

The Application pattern can be further refined with more explicit functions to be performed. Each function is associated with a runtime node. In reality these functions, or nodes, can exist on separate physical machines or may co-exist on the same machine. In the Runtime pattern this is not relevant. The focus is on the logical nodes required and their placement in the overall network structure.

As an example, let's assume that our customer has determined that his solution fits into the Self-Service business pattern and that the Directly Integrated Single Channel pattern is the most descriptive of the situation. The next step is to determine the Runtime pattern that is most appropriate for his situation.

He knows that he will have users on the Internet accessing his business data and he will therefore require a measure of security. Security can be implemented at various layers of the application, but the first line of defense is almost always one or more firewalls that define who and what can cross the physical network boundaries into his company network.

He also needs to determine the functional nodes required to implement the application and security measures. The Runtime pattern shown in Figure 1-9 is one of his options.

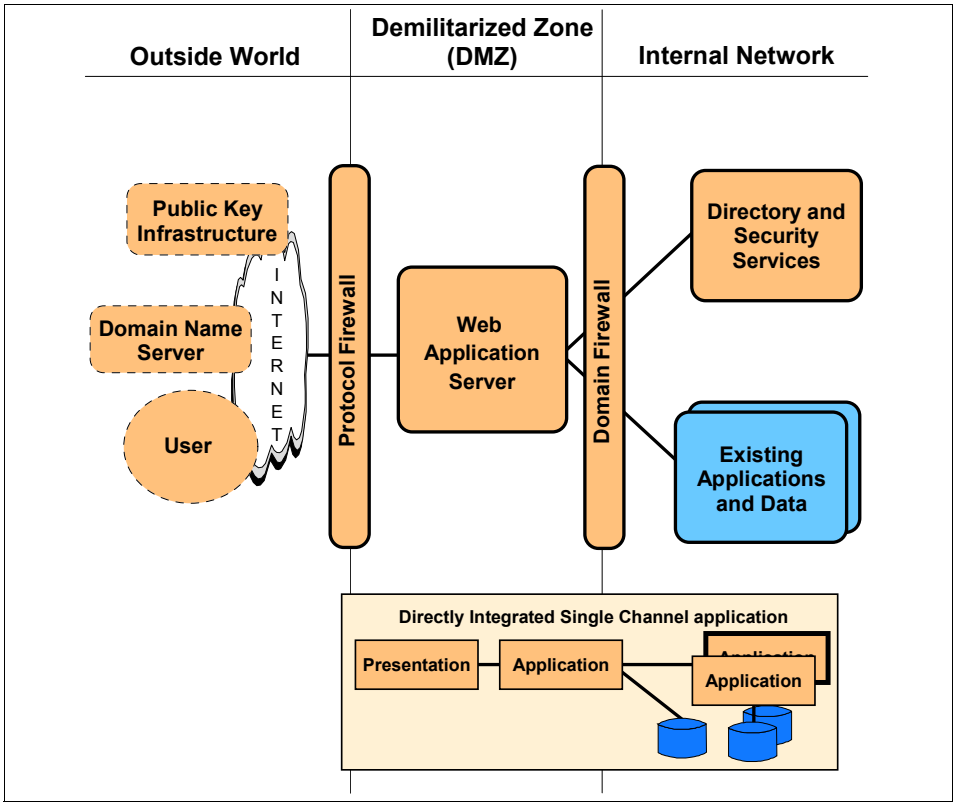


Figure 1-9 Directly Integrated Single Channel application pattern::Runtime pattern

By overlaying the Application pattern on the Runtime pattern, you can see the roles that each functional node will fulfill in the application. The presentation and application tiers will be implemented with a Web application server, which combines the functions of an HTTP server and an application server. It handles both static and dynamic Web pages.

Application security is handled by the Web application server through the use of a common central directory and security services node.

A characteristic that makes this Runtime pattern different from others is the placement of the Web application server between the two firewalls. The Runtime pattern shown in Figure 1-10 is a variation on this. It splits the Web application server into two functional nodes by separating the HTTP server function from the application server. The HTTP server (Web server redirector) will serve static Web pages and redirect other requests to the application server. It moves the application server function behind the second firewall, adding further security.

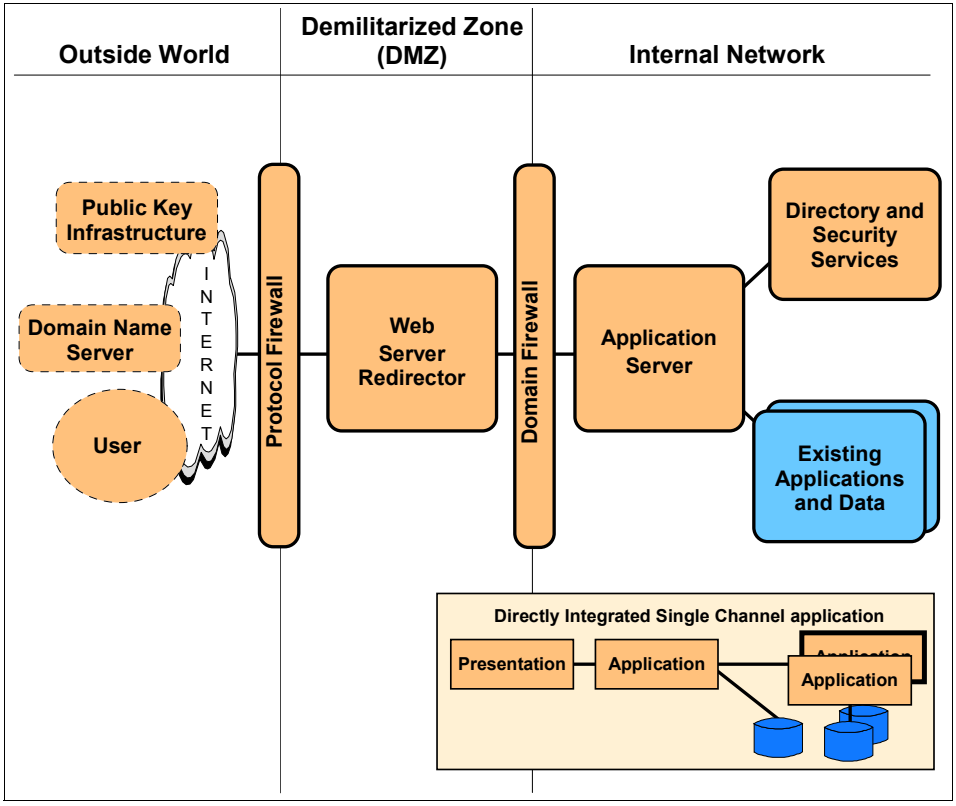


Figure 1-10 Directly Integrated Single Channel application pattern::Runtime pattern: Variation 1

These are just two examples of the possible Runtime patterns available. Each Application pattern will have one or more Runtime patterns defined. These can be modified to suit the customer's needs. For example, he/she may want to add a load-balancing function and multiple application servers.

1.2.4 Review Product mappings

The last step in defining the network structure for the application is to correlate real products with one or more runtime nodes. The patterns Web site shows each Runtime pattern with products that have been tested in that capacity. The Product mappings are oriented toward a particular platform, though more likely the customer will have a variety of platforms involved in the network. In this case, it is simply a matter of mix and match.

For example, the runtime variation in Figure 1-10 on page 15 could be implemented using the product set depicted in Figure 1-11.

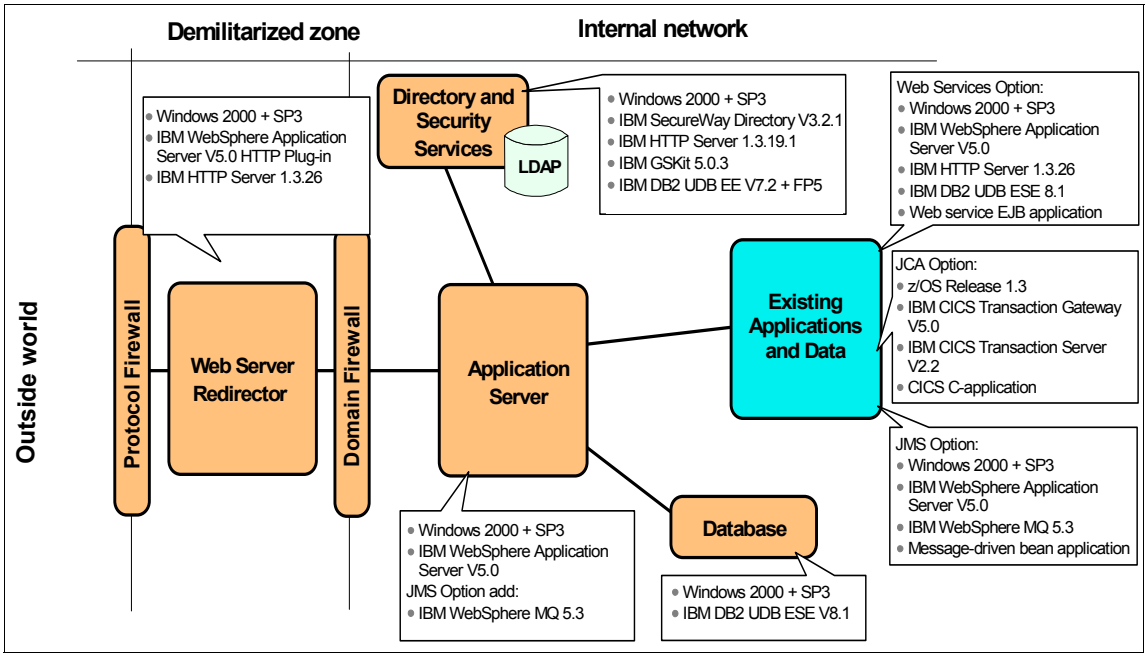


Figure 1-11 Directly Integrated Single Channel application pattern: Windows® 2000 product mapping

1.2.5 Review guidelines and related links

The Application patterns, Runtime patterns, and Product mappings are intended to guide you in defining the application requirements and the network layout. The actual application development has not been addressed yet. The patterns Web site provides guidelines for each Application pattern, including techniques for developing, implementing, and managing the application based on the following:

- Design guidelines instruct you on tips and techniques for designing the applications.

- Development guidelines take you through the process of building the application, from the requirements phase all the way through the testing and rollout phases.
- System management guidelines address the day-to-day operational concerns, including security, back-up and recovery, application management, etc.
- Performance guidelines give information on how to improve the application and system performance.

1.3 Summary

The IBM Patterns for e-business are a collective set of proven architectures. This repository of assets can be used by companies to facilitate the development of Web-based applications. They help an organization understand and analyze complex business problems and break them down into smaller, more manageable functions that can then be implemented.

2

The Access Integration pattern

So far in this redbook, we have introduced the key concepts behind the Patterns for e-business. In this chapter, we will introduce the Access Integration pattern and Portal Composite pattern. The goal of these patterns is to provide a solution for the pervasive requirements that are currently being identified as organizations deploy portal solutions into their environments and integrate them into their business processes.

Later on in this book, we will use the WebSphere Everyplace Access 4.2 product to implement pervasive solutions.

In this chapter, you can read about the Portal Composite pattern and the WebSphere Portal Server. You will also find that the WebSphere Everyplace Access product ships with WebSphere Portal Server. The relation between WebSphere Everyplace Access and WebSphere Portal Server requires some explanation. The portal server in WebSphere Everyplace Access serves two purposes:

- To provide a portal application server for pervasive solutions; it also includes the Transcoding Technology
- To provide management interface for Pervasive services such as Intelligent Notification.

© Copyright IBM Corp. 2003. All rights reserved.

2.1 Access integration patterns

Access Integration patterns describe the services and components commonly required to provide users with consistent, seamless, secure, device-independent access to relevant applications and information. Access Integration patterns are useful when:

- Users need access to multiple applications and information sources via a Single Sign-On and application-independent security context.
- Applications need to be accessible via multiple device types, including fat clients, browsers, voice response units, mobile devices, and PDAs.
- There is a requirement to provide a common look and feel to a collection of applications or to aggregate result sets from discrete applications in a business process.
- The user wishes to customize the choice of applications and how they are presented.
- The business wishes to target information and applications to a specific user or group.

The Access Integration pattern, however, can be used to enable more complex e-business solutions composed of multiple Business patterns. For example, a browser-based, personalized portal can be developed by combining applications that automate the Self-Service business pattern and the Collaboration business pattern. Additionally, this personalized portal might add accessibility to mobile devices.

A study of several e-business solutions that have successfully met these challenging requirements reveals the use of four recurring, common services, including:

1. Device Support service
2. Presentation service
3. Personalization service
4. Security and Administration service

The objective of the Access Integration pattern is to externalize these services and make them selectable by developers of integrated solutions. If you want to learn in more detail these patterns, you can read the IBM Redbook entitle *Access Integration Pattern Using IBM WebSphere Portal Server*, SG24-6267.

2.1.1 Access Integration services

Application patterns for a Business pattern are designed to solve a specific business problem. Application patterns for Access Integration are different in that they are designed to externalize common, user-to-system interaction services from the applications within a Business pattern. These services do not provide end-to-end business value if deployed in isolation from the application patterns that they support.

Access Integration patterns observed in practice are composed of the following services:

- Presentation
- Personalization
- Security and Administration
- Pervasive Device Support
  - Notification
  - Synchronization
  - Device management

We often realize the benefits of Access Integration patterns best when these services are combined. For example, the Personalization service and the Pervasive Device Support service require the use of the Presentation service to create the user interface. The Pervasive Device Support, Personalization, and Presentation services require the Security and Administration service to be effective. System designers can mix and match these services to facilitate consistent and seamless access to multiple applications.

2.2 The Portal composite pattern

Composite patterns represent commonly occurring combinations of Business patterns and Integration patterns to form higher level solutions. Composite patterns typically solve major portions of functionality within a solution.

The Business and Integration patterns that are identified as the building blocks or the more common patterns of the Portal composite pattern are as follows:

- Access Integration pattern
- Self-Service business pattern
- Collaboration business pattern
- Information Aggregation business pattern

Please note that based on your specific requirements, your building blocks of the Business and Integration patterns for your portal may vary from the Portal composite pattern. For example, you may find that you have use for the Extended Enterprise business pattern in addition to the ones we defined, or you may find that you only need the Access Integration, and Self-Services business patterns for your portal. Based on your specific requirements, this would then be defined as a Portal custom design.

The visual representation of the Portal composite pattern is shown in Figure 2-1.

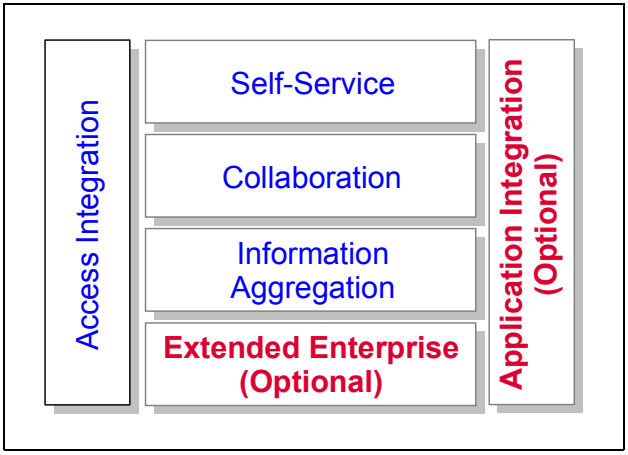


Figure 2-1 Portal composite pattern showing mandatory patterns

For more details on the Portal composite pattern refer to the redbook, *A Portal composite pattern Using WebSphere Portal V4.1*, SG24-6869.

2.2.1 Benefits

The Portal composite pattern is a combination of patterns, technologies and products. It allows for an understanding of the business and IT drivers that help an organization answer these questions:

- Do I need a portal?
- What can I achieve with a portal?

Once an organization has determined that it needs to aggregate information, target that information to specific users, analyze the usage of information, and collect and manage information, it can use a portal to handle these requirements. Consequently, using the Portal composite pattern will eventually lead to a choice of Application patterns and the subsequent combined Runtime pattern. This, in

turn, will drive the creation of a portal architecture. Some specific benefits include:

- A single aggregated view of content targeted to specific user types
- Ability to analyze usage patterns to make marketing efforts more efficient
- Ability to tailor the user interface to specific groups enabling a focus on cultural, language, and nationality-based differences
- Single sign-on, allowing the user to “save time” and have access to information while lessening the requirements for direct interaction with the organization (saves money)

2.2.2 Limitations

The creation of a portal is a complex undertaking. It requires linking together various normally incompatible systems to provide a single view of the information in an enterprise. Although this of value to most organizations, it will require the following of an organization:

- Organizational changes
- Process changes
- Restructuring of existing data sources
- Rebuilding some existing applications to support available connectivity options
- The detailed analysis of the various user groups that need to be supported (usually in much more detail than what currently exists)

A portal implementation assumes that there will be impacts in all of these areas.

2.3 Pervasive solution business strategies

The following table shows a list of Business drivers and strategies to execute the initiatives.

Table 2-1 Strategic goals and initiatives

Business drivers	Strategy	Initiatives
Grow the data-subscriber customer base. Increase the number and length of subscriber content sessions.	Customer strategy	Strengthen content-partnering programs. Invest in user interface design.

Business drivers	Strategy	Initiatives
Gain revenue by providing value-added service to partners. Increase the availability of content.	Content provider strategy	Develop solutions to help partners make money. Increase the percentage of revenue shared with partners or agree to higher licensing fees. Adopt an open-access content policy. Increase the number of content provider relationships
Adopt new technologies and services that create value. Learn more about customer needs.	Technology strategy	Adopt open standard, interoperable platforms. Increase R&D investment in experimental technologies. Invest in resources for capturing customer metrics. Develop a preliminary CRM strategy.
Increase the number and quality of relationships with key vendors and service providers. Make it easy to create unique value by leveraging the partners in the ecosystem.	Network strategy	Develop tightly integrated products and services. Standardize middleware, security platform and management solutions among the partners in the ecosystem.

Business drivers for implementing a pervasive solution are as follows.

- Mobile portal customer strategy for the wireless carrier includes providing open access and open availability, improving user-interface and customer-experience design, and offering services that are convenient and easy to use.
- Content provider strategy determines not only what content and applications will be made available to end users, but how content and application contributors will be compensated and how brands will be represented.
- Although wireless businesses tend to be technologically savvy, a wireless company must also be able to discover and assess the impact of global trends, both in technology and urban culture.

Network strategy is the way that a wireless carrier creates the business partnerships and seamless interworking between partners in the mobile portal to create value for end users.

2.4 Summary

In summary, when designing your solution, evaluate the chosen patterns to ensure that they contain the characteristics that are important for the Pervasive solution you are creating. Remember, this it is ultimately based on the business drivers leading to the pattern and subsequent architecture that supports those drivers.

3

Selecting the Application patterns

After identifying the Composite, Business and Integration patterns that comprise the Pervasive solution, the next step in planning an e-business application is to choose the Application pattern(s) that apply to the business drivers and objectives. An Application pattern shows the principal layout of the application, focusing on the shape of the application, the application logic, and the associated data.

The Application patterns use logical tiers to illustrate various ways to configure the interaction between users, applications, and data. Based on the Application patterns, a set of Runtime patterns is identified. These Runtime patterns are discussed in Chapter 4, “Selecting the Runtime patterns” on page 39.

© Copyright IBM Corp. 2003. All rights reserved.

3.1 Application patterns described

Each Application pattern has associated business and IT drivers. The architect should review each of the business and IT drivers with the associated Application pattern to determine the best fit for the requirements. In this section, the Application patterns that apply to the Access Integration pattern are described. Please note that you may have different Application patterns based on your business requirements.

We did not need to introduce any new Application pattern for our solutions to implement Pervasive Portal applications.

3.1.1 Access Integration application patterns

Application patterns for Access Integration are composed of four services described on the patterns Web site found at:

http://www-106.ibm.com/developerworks/patterns/access/index.html

Based on the specific installation needs of the application being built, developers can mix and match these services to facilitate consistent and seamless access to multiple applications. Three commonly observed Application patterns for Access Integration are presented below. It is highly probable that a solution will utilize more than one of these Application patterns.

Pervasive Device Access application pattern

The Access Integration pattern is used to provide consistent access to various applications using multiple device types. In order to provide pervasive device access to an existing Business pattern, we therefore need to use an Access Integration application pattern as shown in Figure 3-1 on page 29. The Pervasive Device Access application pattern brings a new tier into the architecture. This tier is responsible for the pervasive extensions to the original application. The function of this tier is to convert the source (for example HTML) issued by the application presentation logic into a format appropriate for the pervasive device. In this way, the Pervasive Device Access application pattern provides a structure for extending the reach of individual applications from browsers and fat clients to pervasive devices such as PDAs and mobile phones.

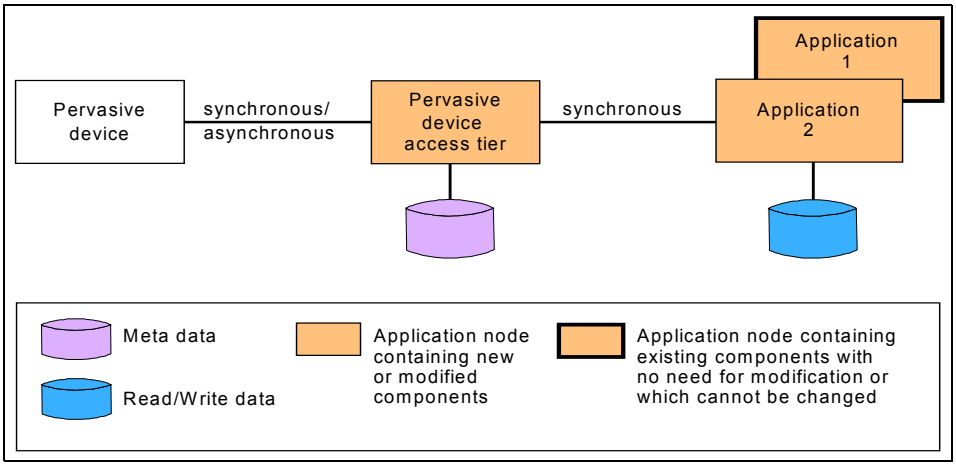


Figure 3-1 Access Integration::Pervasive Device Access

Business and IT drivers

- Provide universal access to information and services
- Reduce time to market
- Reduce Total Cost of Ownership (TCO)

Striving to provide universal access to information and applications is often the primary business driver for choosing this Application pattern.

The primary IT driver for choosing this Application pattern is to quickly extend the reach of applications to new device types without having to modify every individual application to enable its use by additional device types.

Note: WebSphere Everyplace Access V4.2 currently supports different types of wireless phones and PDAs.

The Portal composite pattern supports the use of pervasive device access. In fact, “any type of device” access is supported through the use of templates in the Pervasive device access tier. At this tier, the session data containing the type of device is known and the properly formatted content can be delivered. This formatted content can be transcoded in content management or datasource nodes, or it can be transcoded “dynamically” when requested by a specific type of client. This will depend on the frequency of updates to the data.

Web Single Sign-On application pattern

The Web Single Sign-On application pattern provides a framework for seamless application access through unified authentication services. Figure 3-2 shows an example of this pattern.

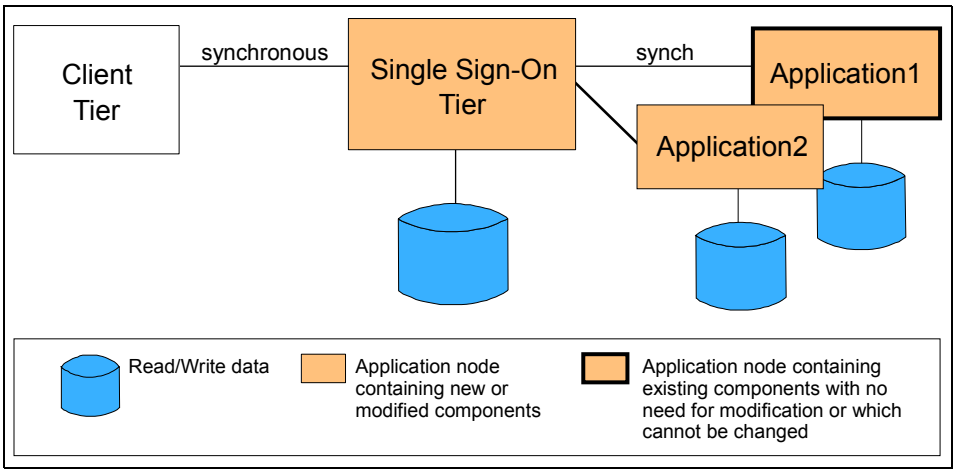


Figure 3-2 Access Integration::Web Single Sign-On

Business and IT drivers

- Provide Single Sign-On across multiple applications
- Reduce Total Cost of Ownership (TCO)
- Reduce user administration cost

The primary business driver for choosing this Application pattern is to provide seamless access to multiple applications with a Single Sign-On while continuing to protect the security of enterprise information and applications.

Simplification and increased efficiency of user profile management are the main IT driver for Single Sign-On.

Benefits

- Users can access their application portfolios easily and securely.
- User profile information is centralized in a common directory, simplifying profile management and reducing costs.
- Application development cost is reduced by providing a standard security solution.

Limitations

Many existing applications are not capable of accepting a standard set of user credentials as a substitute for local authentication. Integration with such systems can be difficult or even impossible.

Personalized Delivery application pattern

The Personalized Delivery application pattern provides a framework for giving access to applications and information tailored to the interests and roles of a specific user or group. This Application pattern extends basic user management by collecting rich profile data that can be kept current up to the user's current session. Data collected can be related to application, business, personal, interaction, or access device-specific preferences. An example of the Personalized Deliver application pattern is shown in Figure 3-3.

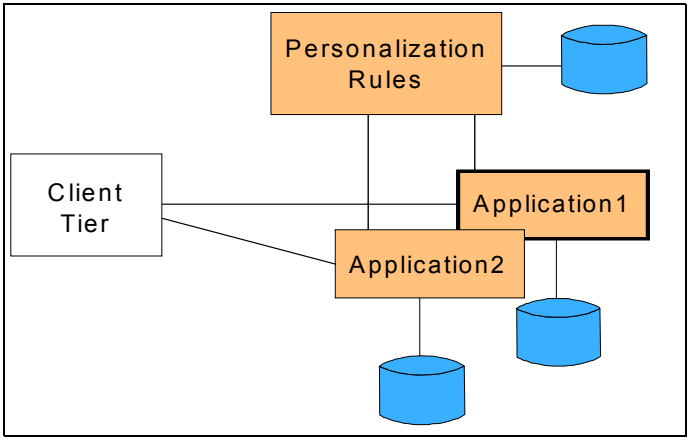


Figure 3-3 Access Integration::Personalized Delivery

Business and IT drivers

The primary business driver for choosing this Application pattern is to increase usability and improve the efficiency of Web applications by tailoring their presentation to the user's role, interests, habits and/or preferences.

Benefits

- Users' interaction with the site is benefited because of increased perception of control and efficiency.
- Fine-grained control of users' access to applications is enabled according to role and preferences by the enterprise.
- Improved user effectiveness is enabled by adapting the complexity and detail of content to a user's skill level.

Limitations

Personalized Delivery can be very complex and expensive to fully implement.

3.1.2 Self-Service application patterns

As you can see in Figure 3-4, the Self-Service business pattern covers a wide range of uses. Applications of this pattern can range from the very simple function of allowing users to view data built explicitly for one purpose, to taking requests from users, decomposing them into multiple requests to be sent to multiple, disparate data sources, personalizing the information, and recomposing it into a response for the user. For this reason, there are currently seven defined Application patterns that fit this range of function. We summarize these for you here. More detailed information can be found in *Patterns for e-business: A Strategy for Reuse*, by Jonathan Adams, Srinivas Koushik, Guru Vasudeva, and George Galambos.

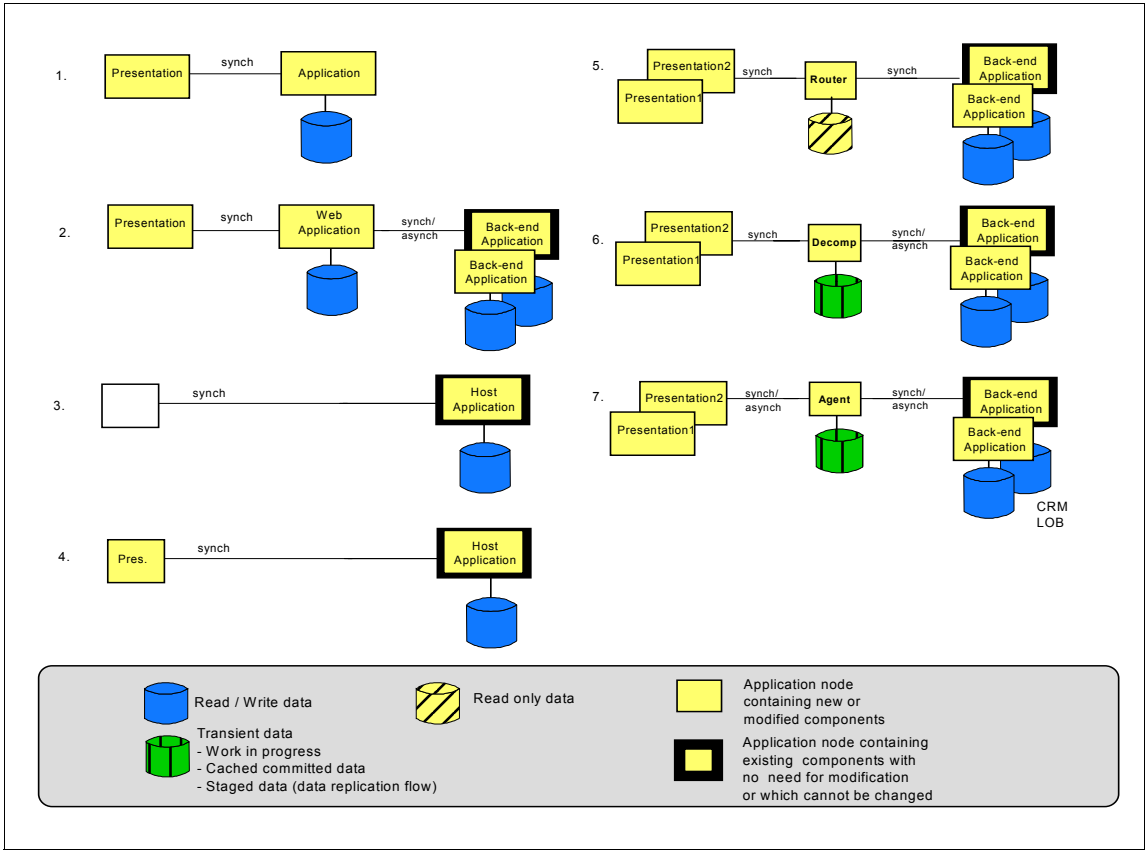


Figure 3-4 Self-Service application patterns

1. **Stand-alone Single Channel application pattern:** provides for stand-alone applications that have no need for integration with existing applications or data. It assumes one delivery channel, most likely a Web client, although it could be something else. It consists of a presentation tier that handles all aspects of the user interface, and an application tier that contains the business logic to access data from a local database. The communication between the two tiers is synchronous. The presentation tier passes a request from the user to the business logic in the Web application tier. The request is handled and a response sent back to the presentation tier for delivery to the user.
2. **Directly Integrated Single Channel application pattern:** provides point-to-point connectivity between the user and existing back-end applications. As with the Stand-alone Single Channel application pattern, it assumes one delivery channel; the user interface is handled by the presentation tier. The business logic can reside in the Web application tier and in the back-end application. The Web application tier has access to local data that exists primarily as a result of this application, for example, customer profile information or cached data. It is also responsible for accessing one or more back-end applications. The back-end applications contain business logic and are responsible for accessing the existing back-end data. The communication between the presentation tier and Web application tier is synchronous. The communication between the Web application tier and the back-end can be either synchronous or asynchronous, depending on the characteristics and capabilities of the back-end application.
3. **As-is Host application pattern:** provides simple direct access to existing host applications. The application is unchanged, but the user access is translated from green-screen type access to Web browser-based access. This is very quickly implemented but does nothing to change the appearance of the application to the user. The business logic and presentation are both handled by the back-end host. Because the interface is still host driven, this is more suited to an intranet solution where employees are familiar with the application.
4. **Customized Presentation to Host application pattern:** this is one step up from the As-is Host application pattern. The back-end host application remains unchanged, but a Web application now translates the presentation from the back-end host application into a more user-friendly, graphical view. The back-end host application is not aware of this translation.
5. **Router application pattern:** the Router application pattern provides intelligent routing from multiple channels to multiple back-end applications using a hub-and-spoke architecture. The interaction between the user and the back-end application is a one-to-one relation, meaning the user interacts with applications one at a time. The router maintains the connections to the back-end applications and pools connections when appropriate, but there is

no true integration of the applications themselves. The router can use a read-only database, most probably to look up routing information. The primary business logic still resides in the back-end application tier.

This pattern assumes that the users are accessing the applications from a variety of client types such as Web browsers, voice response units (VRUs) or kiosks. The Router application pattern provides a common interface for accessing multiple back-end applications and acts as an intermediary between them and the delivery channels. In doing this, the Router application pattern may use elements of the Integration patterns.

6. **Decomposition application pattern:** the Decomposition application pattern expands on the Router application pattern, providing all the features and functions of that pattern and adding recomposition/decomposition capability. It provides the ability to take a user request and decompose it into multiple requests to be routed to multiple back-end applications. The responses are recomposed into a single response for the user. This moves some of the business logic into the decomposition tier, but the primary business logic still resides in the back-end application tier.
7. **Agent application pattern:** the Agent pattern includes the functions of the decomposition tier, plus it incorporates personalization into the application to provide a customer-centric view. The agent tier collects information about the user, either from monitoring their habits or from information stored in a CRM. It uses this information to customize the view presented to the user.

Stand-Alone Single Channel application pattern

The application in this book is based on the simplest pattern, the Stand-Alone Single Channel application pattern. This pattern provides for stand-alone applications that have no need for integration with existing applications or data. It assumes one delivery channel, most likely a Web client, although it could be something else. It consists of a presentation tier that handles all aspects of the user interface, and an application tier that contains the business logic to access data from a local database. The communication between the two tiers is synchronous. The presentation tier passes a request from the user to the business logic in the Web application tier. The request is handled and a response is sent back to the presentation tier for delivery to the user.

There are other IBM Redbooks on the Self-Service Business pattern which you can reference:

- *Self-Service Patterns using WebSphere Application Server V4.0*, SG24-6175
- *Patterns: Connecting Self-Service Applications to the Enterprise*, SG24-6572
- *Self-Service Applications using IBM WebSphere V4.0 and IBM MQSeries Integrator*, SG24-6160

- *Patterns on z/OS: Connecting Self-Service Applications to the Enterprise*, SG24-6827
- *Patterns: Building Messaging-based and Transactional Applications*, SG24-6875

3.1.3 Identified Application patterns for the Portal composite pattern

The Business and Integration patterns that we have identified as the building blocks to the Portal composite pattern are:

- Access Integration
- Self-Service
- Collaboration
- Information Aggregation

Note: The discussions involving the identified Application patterns use this naming convention:

```
<business/integration pattern name>::<application pattern name>
```

For example, in discussing the Access Integration pattern and the Web Single Sign-On application pattern, the format is:

```
Access Integration::Web Single Sign-On
```

Figure 3-5 shows the identified Application patterns that make up a Portal composite pattern.

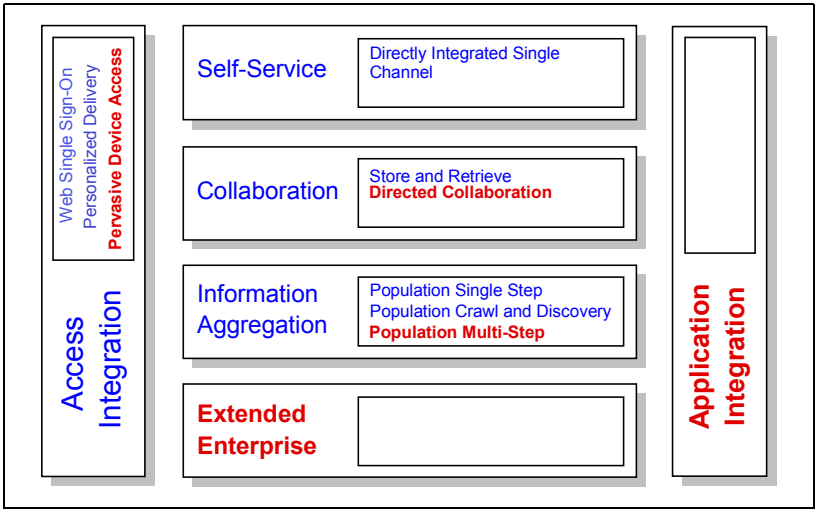


Figure 3-5 Application patterns in a Portal composite pattern

Over time in the marketplace, the definitions of the Composite patterns are becoming more defined. The mandatory patterns represent the occurring patterns that are being implemented by companies in a portal solution. The optional patterns are not yet implemented with each portal solution but are optional for a specific company's requirements and would be implemented as a Portal custom design. Over time, we may see that all the patterns in Figure 3-5 on page 35 are shown as mandatory. For now, the mandatory and optional patterns for the Portal composite pattern are as follows:

- Mandatory Business and Integration::Application patterns
  - Access Integration::Single Sign-On, Personalized Delivery
  - Self-Service::Directly Integrated Single-Channel
  - Collaboration::Store and Retrieve
  - Information Aggregation::Population Single Step, Population Crawling and Discovery
- Optional Business and Integration::Application patterns
  - Access Integration::Pervasive Device Access
  - Collaboration::Directed Collaboration
  - Information Aggregation::Population Multi-Step

These identified Application patterns are based on the requirements of a typical portal implementation.

3.2 Where to find more information

For more information on the Pervasive Access device application pattern, refer to *Mobile Applications with IBM WebSphere Everyplace Access Design and Development*, SG24-6259.

For those wanting a full description of the functionality required in a pervasively enabled e-business application, and the design issues accompanying these requirements, we recommend that you review *IBM WebSphere Everyplace Server Service Provider and Enable Offerings: Enterprise Wireless Applications*, SG24-6519. This redbook describes a full-featured implementation of the WebSphere Everyplace Suite, used to enable pervasive applications.

If you would like additional information on the use of IBM WebSphere Everyplace Server V2.1.1 to build successful mobile e-business solutions, review *IBM WebSphere Everyplace Server: A Guide for Architects and Systems Integrators,* SG24-6189. Page 55 of this redbook features a table that lists the features, functions, and benefits of the WebSphere Everyplace Server Service Provider Offering, featured in the product mappings documented on this Web site.

Access Integration pattern Redbooks:

- *Access Integration Pattern using IBM WebSphere Portal Server*, SG24-6267
- *Mobile Applications with IBM WebSphere Everyplace Access Design and Development*, SG24-6259

Self-Service business pattern Redbooks:

- *Self-Service Patterns using WebSphere Application Server V4.0*, SG24-6175
- *Patterns: Connecting Self-Service Applications to the Enterprise*, SG24-6572
- *Self-Service Applications using IBM WebSphere V4.0 and IBM MQSeries Integrator*, SG24-6160
- *Patterns on z/OS: Connecting Self-Service Applications to the Enterprise*, SG24-6827
- *Patterns: Building Messaging-based and Transactional Applications*, SG24-6875

4

Selecting the Runtime patterns

After choosing the appropriate Business pattern and Application pattern, it is time to define the Runtime pattern and map the products used to implement it.

Runtime patterns define functional nodes (logical) that underpin an Application pattern. The Application pattern exists as an abstract representation of application functions, whereas the Runtime pattern is a middleware representation of the functions that must be performed, the network structure to be used, and the systems management features, such as load balancing and security. In reality, these functions, or nodes, can exist on separate physical machines or may co-exist on the same machine. In the Runtime pattern, this is not relevant. The focus is on the logical nodes required and their placement in the overall network structure.

© Copyright IBM Corp. 2003. All rights reserved.

4.1 Runtime pattern nodes description

A Runtime pattern is represented by logical nodes, where each node has a specific role in the architecture. It defines the topology of the architecture and node placement. Most patterns will consist of a core set of common nodes, with the addition of one or more nodes unique to that pattern. To understand the Runtime patterns presented in this book, you will need to review the following node definitions.

User node

The user node is most frequently a personal computing (PC) device supporting a commercial browser, for example, Netscape Navigator or Internet Explorer. The browser is expected to support SSL and some level of DHTML. Increasingly, designers need to also consider that this node might be a pervasive computing device, such as a Personal Digital Assistant (PDA).

Pervasive user node

A pervasive user node is a catch-all category of portal users that includes all mobile (non-desktop) connected end-user devices other than a Web browser. In most current scenarios, this includes devices such as mobile phones, personal digital assistants, and text pagers.

Domain Name Server (DNS) node

The DNS node assists in determining the physical network address associated with the symbolic address (URL) of the requested information. The Domain Name Server node provides the technology platform to provide host to IP address mapping, that is, to allow for the translation of names (referred to as URLs) into IP addresses and vice versa.

Public key infrastructure (PKI) node

PKI is a system for verifying the authenticity of each party involved in an Internet transaction, protecting against fraud or sabotage, and for non repudiation purposes to help consumers and retailers protect themselves against denial of transactions. Trusted third-party organizations called certificate authorities issue digital certificates - attachments to electronic messages - that specify key components of the user's identity. During an Internet transaction, signed, encrypted messages are automatically routed to the certificate authority, where the certificates are verified before the transaction can proceed. PKI can be embedded in software applications, or offered as a service or a product. e-business leaders agree that PKIs are critical for transaction security and integrity, and the software industry is moving to adopt open standards for their use. In the context of the topologies defined in this IBM Redbook, PKI supports the authentication of the server to the browser client and the confidentiality, using the SSL protocol.

Gateway node

Gateway nodes switch between the different networks to establish communication between pervasive devices and the Web applications. This only means that the two parties can communicate with each other. It does not mean that they will understand each other. Communicating and passing data between the two parties is one thing, but adapting the content and translating between different protocols is another. The content translation is done by the Transcoding Proxy node.

Firewall node

A firewall is a hardware/software system that manages the flow of information between the Internet and an organization's private network. Firewalls can prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets, and can block some virus attacks -- as long as those viruses are coming from the Internet. A firewall can separate two or more parts of a local network to control data exchange between departments. Firewalls provide the first line of defense for protecting private information, but comprehensive security systems combine firewalls with encryption and other complementary services, such as content filtering and intrusion detection.

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

- Screening routers (the Protocol firewall)
- Application gateways (the Domain firewall)

Two levels of firewall nodes provide increasing protection at the expense of increasing computing resource requirements. They have different levels of security implementation:

- **The Protocol firewall** is typically implemented as an IP Router and is basically configured with filters. It protects from access to unauthorized services in the DMZ and also can avoid inadequate LAN bandwidth usage.
- **The Domain firewall** prevents unauthorized access to servers on the internal network by limiting incoming requests to a tightly controlled list of trusted servers in the DMZ. In an n-tier architecture, it prevents the user from accessing any critical data or application directly.

Load Balancer node

The Load Balancer node provides horizontal scalability by dispatching HTTP connections among several, identically configured Web servers. The Load Balancer component distributes interactive traffic across a number of hosts using dynamically updated rules for load balancing, while providing a single system image to the client system. It is used to achieve scalability through the use of multiple servers, and high availability through being able to dynamically vary the algorithms by which a host is selected if one host fails or becomes overloaded.

Transcoding Proxy node

The Transcoding Proxy node is a legacy component from earlier Pervasive solutions. This node is responsible for the content transformation between the content provider (server) and the pervasive devices (client).

This node appears on the diagram in a lighter shade, indicating that it is a legacy component.

Web presentation server node

The Web presentation server node provides services to enable a unified user interface. It is responsible for all presentation-related activity. In its simplest form, it serves HTML pages and runs servlets and JSPs. For more advanced patterns, it acts as a portal and provides the access integration services (Single Sign-On, for example). It interacts with the personalization server node to customize the presentation based on the individual user preferences or on the user role. The Web presentation server allows organizations and their users to standardize and configure the presentation of applications and data in the most efficient way, while enabling fine-grained access control.

Application server node

The application server node provides the execution and communication runtime environment for the business logic of the application. The business logic may be self-contained on the application server node. If not, the application server node is responsible for interacting with back-end applications and retrieving data from back-end data sources. The application server node typically enables infrastructure services such as persistence, resource connection pooling, scalability, failover, administration, and support for Java.

Web application server node

A Web application server node is an application server that includes an HTTP server (also known as a Web server) and is typically designed for access by HTTP clients and to host both presentation and business logic (it includes the Web presentation server and the Application Server node).

Web server redirector node

In order to separate the Web server from the application server, a so-called Web server redirector node (or just redirector for short) is introduced. The Web server redirector is used in conjunction with a Web server. The Web server serves HTTP pages and the redirector forwards servlet and JSP requests to the application servers. The advantage of using a redirector is that you can move the application server behind the domain firewall into the secure network, where it is more protected than within the DMZ.

Personalization server node

The personalization server node works with the Web presentation server node to customize the presentation with data that matches a user's interest. The personalization server identifies the type or class of the user based on information available about the user. Based on this classification, data taken from a content datastore either in the Personalization tier or from back-end sources is selected for presentation to the user. It provides the mapping function of user classification to content data.

Collaboration node

The collaboration node provides synchronous and asynchronous modes of communicating between organizations. We call this a *community*. A community is empowered by collaborative work between users. The collaboration node provides interactive discussions (interactive messaging and chat functionality) and the sharing of documents/ideas (team room environment).

Content management node

The content management node provides for the management of digital assets (for example images, documents, and "pieces" of text) and applies a workflow and security rules (for example access control) to each discrete asset. Note that assets can also be referred to as resources (as they are in WebSphere Content Publisher). The content management node will commonly include and/or leverage the following functions:

- Content type/category identification
- Workflow (based on a user's role and/or the type of content)
- Versioning (including rollback to previous versions)
- Handling of static or dynamic content
- Transcoding/reformatting of content (more recently added to handle multiple end-user channel device types)
- Storage of content to multiple data source types (for example DBMS, file systems)

Search and indexing node

A search and indexing node provides a function to catalog and/or index the content data sources. This will provide the capabilities to locate specific content (for example product or catalog information) and to update this search capability when updates are added (via indexing). In addition, this information can be indexed in a manner that provides the Presentation and Personalization server an ability to find information that is associated with the actions taken by the end user. For example, this could provide for cross-selling or up-selling on a commerce site, which is a specific form of Implicit Personalization. For more details, refer to the Predictive Personalization runtime pattern at:

`http://www-106.ibm.com/developerworks/patterns/access/at3-runtime.html`

Pervasive devices services node

This node includes three services. In some cases, not all three services are implemented. The services can also be separated onto separate nodes.

- Notification

  The notification node provides message interchange between users and their applications. It allows users to subscribe to services, define the delivery method and specify rules for how and when the information will be delivered.

- Synchronization

  The synchronization node enables handheld computing devices to link remotely to desktop applications and synchronize data with several applications like mail servers, relational databases, etc. The mobile device can synchronize using several channels such as a modem, cellular phone, the Internet, wireless, an intranet, a local area network (LAN) or a wide area network (WAN).

- Device manager

  The device manager node provides identification, configuration, inventory management and software distribution to devices such as personal digital assistants (PDAs), handheld PCs, smartphones, wireless access protocol (WAP) devices, or other emerging devices for pervasive computing.

Directory and security services node

The directory and security services node supplies information on the location, capabilities and attributes (including user ID/password pairs and certificates) of resources and users known to this Web application system. This node can supply information for various security services (authentication and authorization) and can also perform the actual security processing, for example, to verify certificates. The authentication in most current designs validates the access to the Web application server part of the Web server, but this node also authenticates for access to the database server.

To provide Single Sign-On services, a Lightweight Directory Access Protocol (LDAP) directory is used.

Shared file server node

The timely synchronization of several Web servers is achieved by using a shared file system as the content storage and capitalizing on the replication capability of this technology. In a Web environment with several Web application servers, this component can be a centralized repository for HTML, Java and JSPs files, facilitating their management and update process to serve all the application servers. Any changes or updates to the content of the application server can be done on the file server.

Database server node
This node's function is to provide persistent data storage and retrieval service in support of transactional interactions.

Existing applications and data node
The existing application and data node represents the legacy systems, which are running on the internal network. These elements provide business logic and also persistence of the data.

4.2 Runtime pattern for the Self-Service application

In order to understand the complete concept, it is best to start with the basic elements. We will start with the simplest Self-Service Runtime pattern, then later enhance it with nodes relevant to the Access Integration Runtime pattern.

4.2.1 Basic Runtime pattern

This Runtime pattern, shown in Figure 4-1 on page 46, provides an initial implementation with an entry-level footprint. It is a simple yet effective way to make the solution available. The basic pattern uses a minimum of runtime nodes, yet provides a measure of security by putting all sensitive persistent data behind a firewall. While it does not provide scalability or failover capabilities, it is a good starting point from which you can easily progress to Runtime patterns that do provide these functions.

The Runtime pattern does not differentiate between intranet and Internet implementations. However, you should be aware of certain issues:

- Bandwidth is usually greater in an intranet, allowing the use of more network-intensive solutions, such as thick clients.
- Security may be less of an issue in an intranet. However, protecting the network is still important and firewalls protecting resources from unauthorized access are still advisable.
- Due to corporate rules, you may expect certain browsers to be used, allowing you to exploit all available features and not code to the least common denominator.

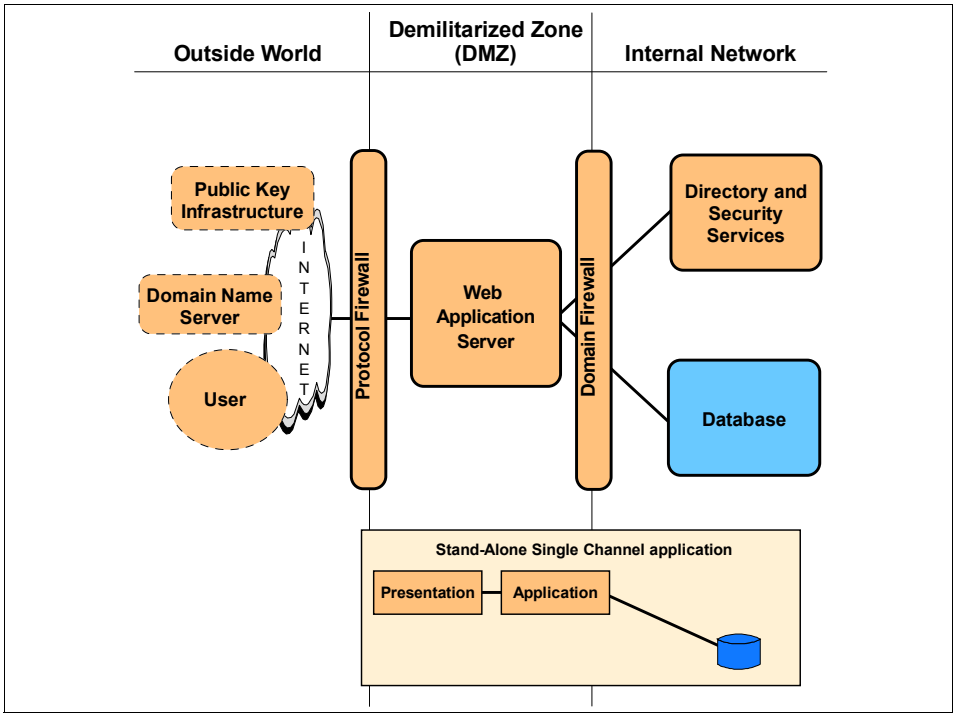


Figure 4-1 Stand-Alone Single Channel application pattern::Runtime pattern

The presentation logic and business logic of the application are provided by a single Web application server node in a demilitarized zone (DMZ). The data to be accessed from the business logic is behind the domain firewall in the internal network.

In addition to the network security provided by the firewalls, application-level security is provided by the Web application server node. The user information required for authentication and authorization is stored in the directory and security services node behind the domain firewall in the internal network.

4.2.2 Runtime pattern: Variation 1

As shown in Figure 4-2 on page 47, this variation to the basic Runtime pattern uses one Web server redirector containing the Web server and one application server, effectively splitting the function of a Web application server across two machines. In this case, the application server resides in the internal network to

provide it with more security. The application server node will run both presentation and business logic. The Web server remains in the DMZ and serves static pages. A Web server redirector is used to forward the requests from the Web server to the application server.

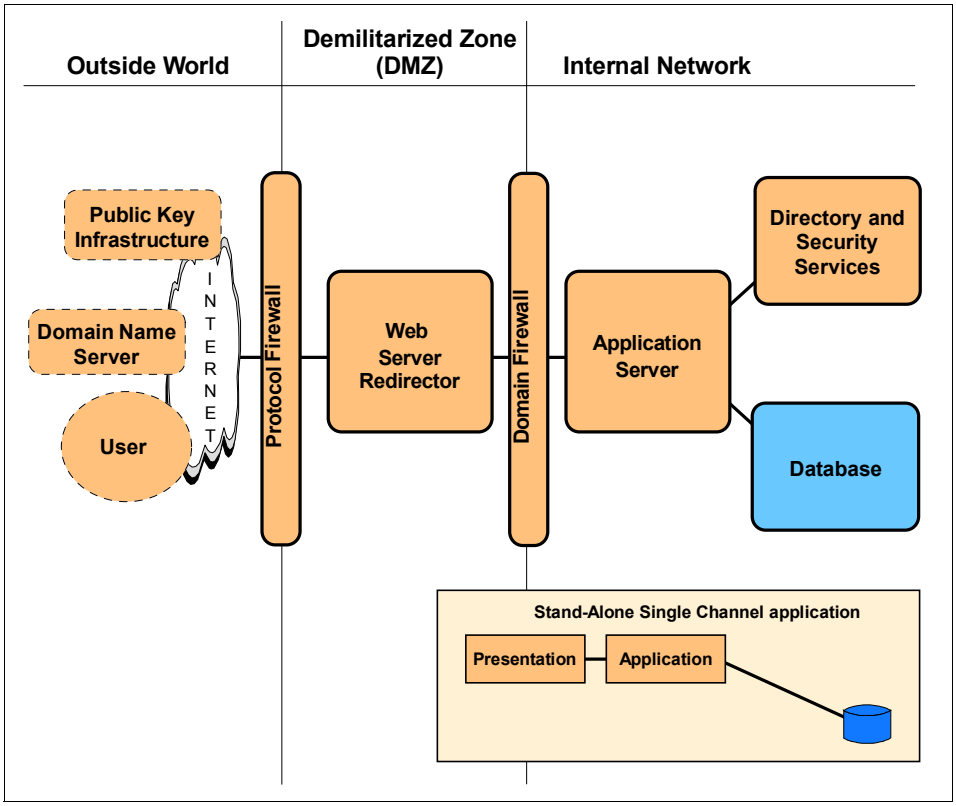


Figure 4-2 Stand-Alone Single Channel application pattern::Runtime pattern: Variation 1

We focus on this variation of the Stand-Alone Single Channel application pattern with the Pervasive Device Access application pattern.

4.3 Runtime pattern for the Pervasive Device Access application

Self-Service runtime patterns provide the foundation for the Pervasive Device Access Runtime pattern in this book. The Access Integration::Pervasive Device Access application pattern provides the support for pervasive solutions.

In the last Runtime pattern, we extend the Portal composite pattern with the Pervasive Device Access pattern to provide a so called “Pervasive Portal” type of solution, where all the Access Integration functions come together in one place.

4.3.1 Access Integration pattern

The Access Integration pattern provides users with a seamless and consistent user experience that combines access to multiple applications, databases, services and presentation through multiple devices.

There are multiple runtime patterns defined for this integration pattern and you can see each of them in detail by referring to the Patterns for e-Business site at:

http://www-106.ibm.com/developerworks/patterns/

Old Pervasive Device Access Runtime pattern

The following Runtime pattern is from the redbook *Mobile Applications with IBM WebSphere Everyplace Access - Design and Development*, SG24-6259. It was based on the technologies available in WebSphere Everyplace Access V1.1. This Runtime pattern serves as a quick introduction to see where the pervasive technologies started in the WebSphere Everyplace Access product line.

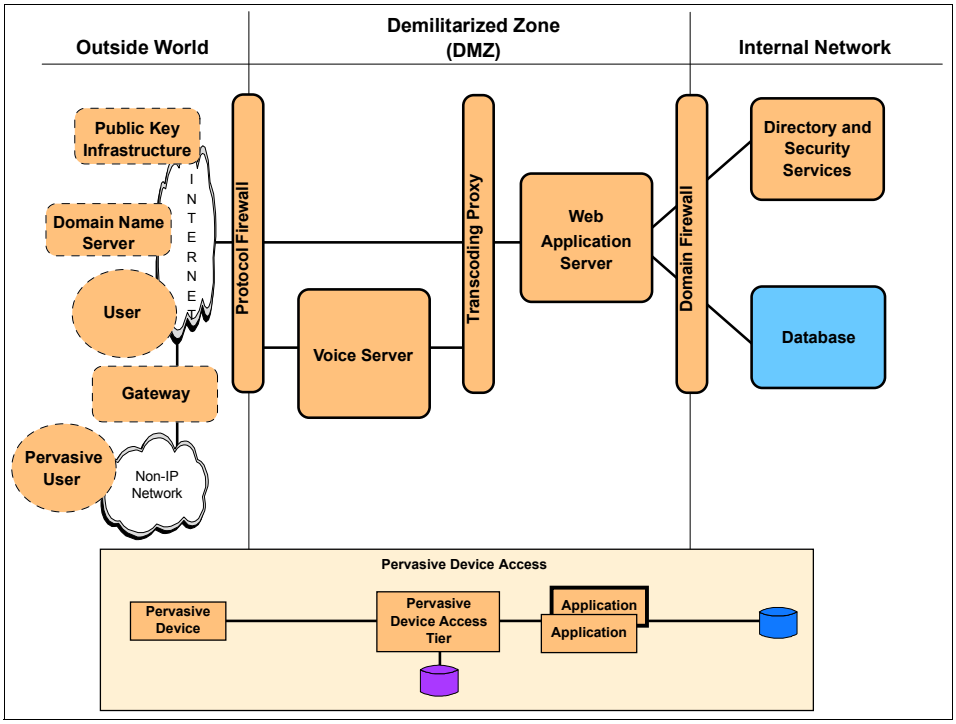


Figure 4-3 Old Pervasive Device Access Runtime pattern

The main technology that drove the solution was transcoding. The transcoding proxy took care of the requests and responses from and to the different pervasive devices. The transcoding proxy had three runtime modes:

- Reverse-proxy
- Forward-proxy
- Servlet filter for WebSphere

The voice server provided voice recognition and voice synthesis for the solution; enabling the applications for clients using traditional phone to access the application.

The Web application was a classic J2EE application with minor modifications to support pervasive device access.

Pervasive Device Access Runtime pattern: Variation 1

This Runtime pattern extends the Self-Service Basic Variation 1 Runtime pattern to allow a wide variety of pervasive devices to participate in the e-business solution.

There are two Pervasive Access runtime patterns defined on the Patterns for e-Business site at:

http://www-106.ibm.com/developerworks/patterns/

The Pervasive Device Access runtime pattern Variation 1 is a variation to these two Runtime patterns.

This pattern assumes that there are incoming requests for the application from both IP and non-IP networks. The non-IP network consists of wireless networks (GSM, CDPD, etc.) and phone networks (PTSN). In order to access the IP-based networks (Internet or intranet), a special gateway is required. Phone networks require Voice over IP gateways to connect the analog or digital lines connection to the IP-based, packet-switched networks. Wireless networks also have to use gateways to handle the connections, translate the protocols, and connect to IP networks (for example WAP or i-mode).

The functionalities that are part of a pervasive solution and are included in this Runtime pattern are:

- Connectivity - it is implemented by the Gateway node providing communication between pervasive devices and the Web applications.
- Application server - in this scenario, the application server hosts the application that is designed to handle different type of devices. It is more like a programmatic approach and it is most common for well designed existing Web applications with minor modifications or applications that only need to support a few (one or two) different device presentation logics.
- Notification - provides message interchange between users and applications.
- Synchronization - provides data synchronization with several applications like mail servers and relational databases through several communication channels.
- Device Management - provides identification, configuration, inventory management and software distribution to devices such as personal digital assistants (PDAs), handheld PCs, smartphones, wireless access protocol (WAP) devices, or other emerging devices for pervasive computing.

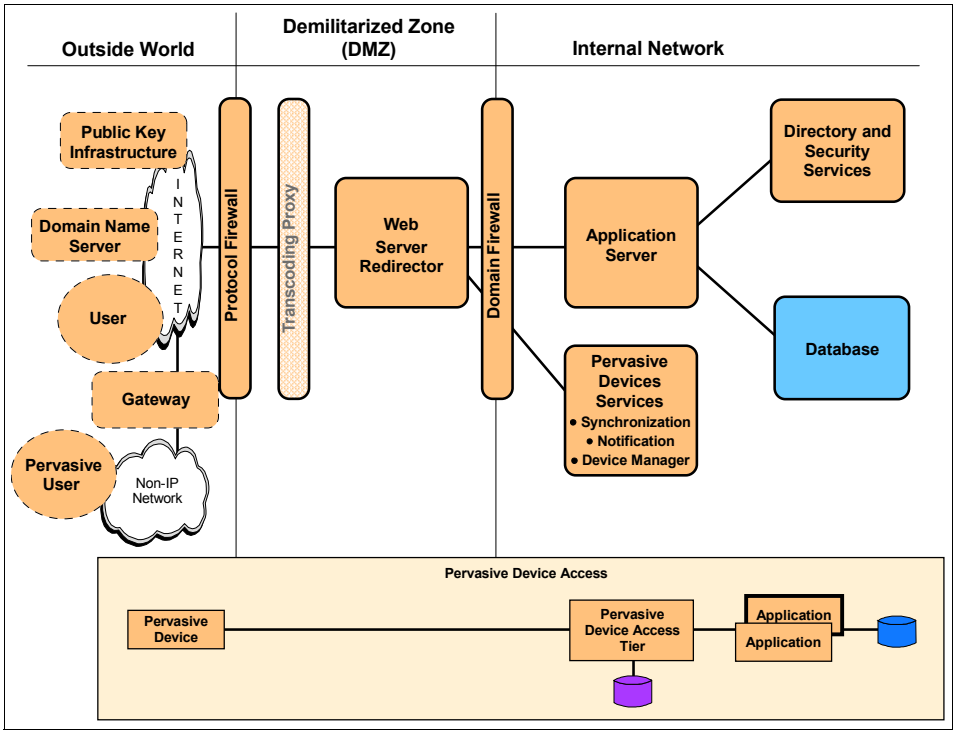


Figure 4-4 Pervasive Device Access Runtime pattern: Variation 1

This Runtime pattern is based on the n-tier model and supports browser users (client node) and mobile users (pervasive user node) accessing back-end applications. The mobile users will have to pass through the gateway to access the IP network and through the transcoding proxy to adapt the content, based on the type and capabilities of the mobile user device.

The presentation tier is split into two, represented by the Web Server Redirector and the Application Server nodes. In this case, the application server has to take care of the presentation for the different devices. It can be done programmatically, or using the WebSphere Transcoding Publisher product as an intermediary node. For more information about the WebSphere Transcoding Publisher solution, refer to the redbook *Mobile Applications with IBM WebSphere Everyplace Access - Design and Development*, SG24-6259.

The Self-Service application runs on the application server. The notification, synchronization and device manager services also run in the application tier on a separate node.

The relevant data that should be stored in the database for each component is as follows:

- For the notification - privacy policies, context data and subscriptions.
- For the synchronization - configuration information, enterprise database access information, user IDs and passwords, user device preferences and data replicas. In order to improve performance and security in the synchronization process, the synchronization node can implement a cache by replicating the data from the back-end database.
- For the device manager - device management information.

The authentication information and user profiles will be held by the directory and security node and is protected by the domain firewall in the internal network.

4.4 Portal composite pattern variation for Pervasive solutions

This Runtime pattern is a combination of the Portal composite pattern and the Pervasive Device Access Variation 1 runtime pattern.

The personalization server node is part of the application tier and interacts with the Web application server node and the database node.

The personalization services can be implemented to support participatory, predictive and prescriptive personalization. This book will not focus on the Personalization pattern; to get detailed information, refer to the redbook *A Portal Composite Pattern Using WebSphere V4.1*, SG24-6869. *User-to-Business Pattern using WebSphere Personalization Patterns for e-business Series,* SG24-6213-00 or the Patterns for e-Business site at:

http://www-106.ibm.com/developerworks/patterns/

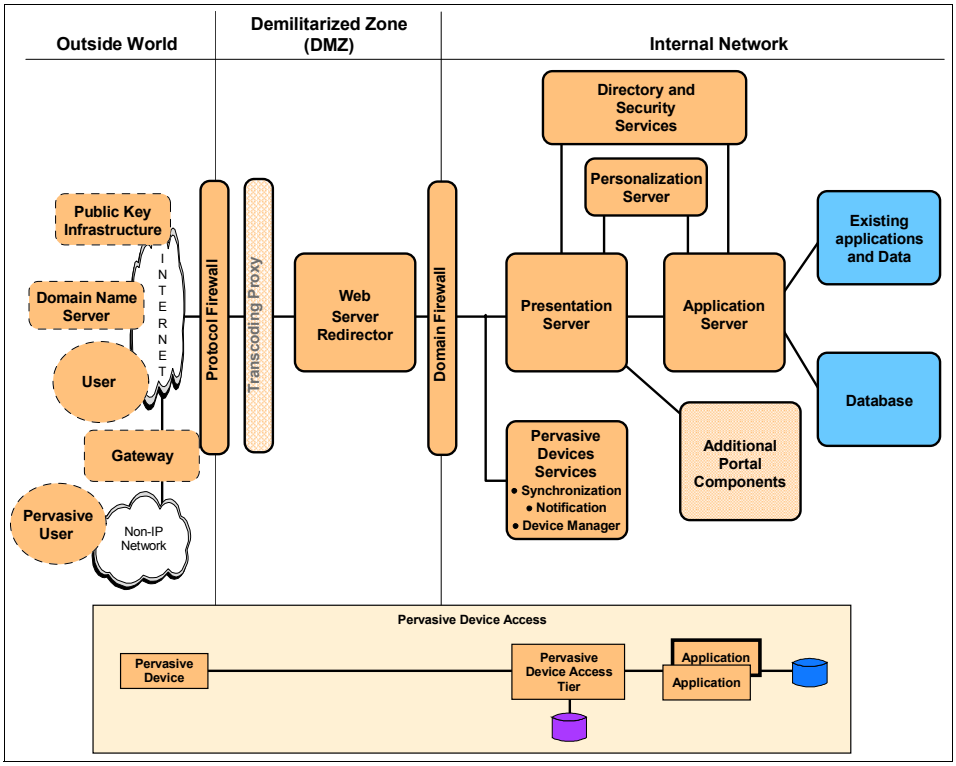


Figure 4-5 Portal composite pattern variation for Pervasive solution

The Portal composite pattern is a major part of this Runtime pattern. Portal is the main content provider for most of the pervasive devices. Portal also has the ability to transform the content as necessary.

Note that the Additional Portal Components are less important from the pervasive device access point of view, so components such as Content Management, Collaboration, Search and Indexing are not discussed here. For more information on these components, refer to the redbook *A Portal Composite Pattern Using WebSphere V4.1*, SG24-6869.

For details on the rest of the nodes, refer to “Pervasive Device Access Runtime pattern: Variation 1” on page 49.

5

Selecting the product mapping

After choosing the appropriate runtime pattern, it is time to map the Runtime pattern and map the products used to implement it.

A product mapping maps the logical nodes defined in the Runtime pattern to specific products which implement the Runtime solution design on a selected platform. The Product mapping identifies the platform, software product name, and often version numbers as well.

© Copyright IBM Corp. 2003. All rights reserved.

5.1 Product mappings

The next step after choosing a Runtime pattern is to determine the actual products and platforms to be used. Selecting products means selecting hardware and software elements as well. Choosing the right elements is always difficult because of various considerations which must be taken into account. In some cases, decisions are made considering not only the technical and logical points but other factors such as expenses, existing knowledge, or time. The platform chosen should fit into the customer's environment and ensure quality of service, such as scalability and reliability, so that the solution can grow along with the e-business.

5.1.1 Pervasive Portal solution framework

The products used to implement a Pervasive Portal solution in this redbook includes the WebSphere product family running in a heterogeneous Microsoft® Windows 2000 and IBM AIX® environment.

5.1.2 Product mapping for Pervasive solutions

Product mappings are shown for:

- Pervasive Device Access Runtime pattern: Variation 1
- Portal composite pattern runtime variation for Pervasive solutions

Product mapping for Pervasive Device Access Runtime pattern: Variation 1

The product mapping for this runtime supports the basic runtime and the variation for security and addressing the Windows and AIX platforms.

This product mapping represents the Pervasive Device Access pattern with the separation of the Web presentation server and the application server, providing a more secure, scalable and reliable architecture as described in the Access Integration pattern section, “Pervasive Device Access Runtime pattern: Variation 1” on page 49.

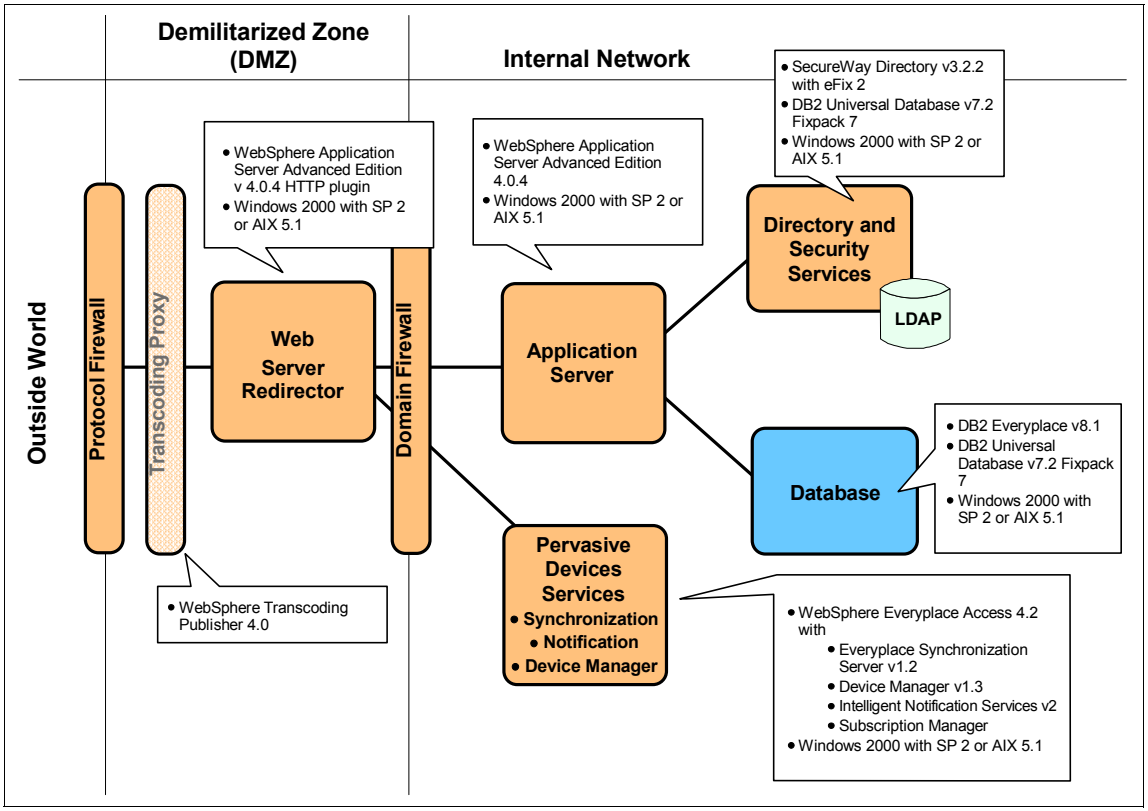


Figure 5-1 Pervasive Device Access Runtime pattern Variation 1 product mapping

Although this product mapping is focusing on enabling an existing application for Pervasive devices and does not include portal components, it still requires the portal server to run system management services, such as Intelligent Notification.

Portal composite pattern runtime variation for Pervasive solutions

The product mapping for this runtime is based on the Portal composite runtime pattern, additionally it supports the pervasive device access and addresses the Windows and AIX platforms.

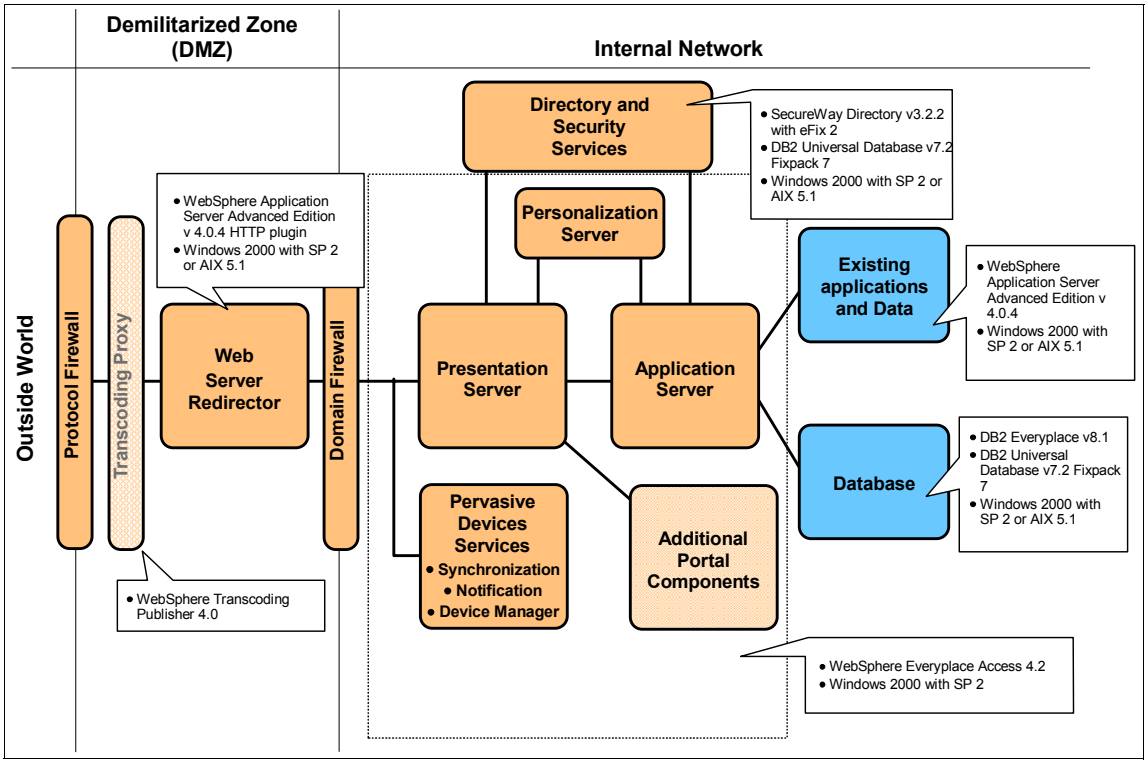


Figure 5-2 Portal composite pattern runtime pattern variation for Pervasive

Protocol mapping

The protocol mapping in Figure 5-3 on page 59 represents the protocols used to communicate between the nodes and the applications.

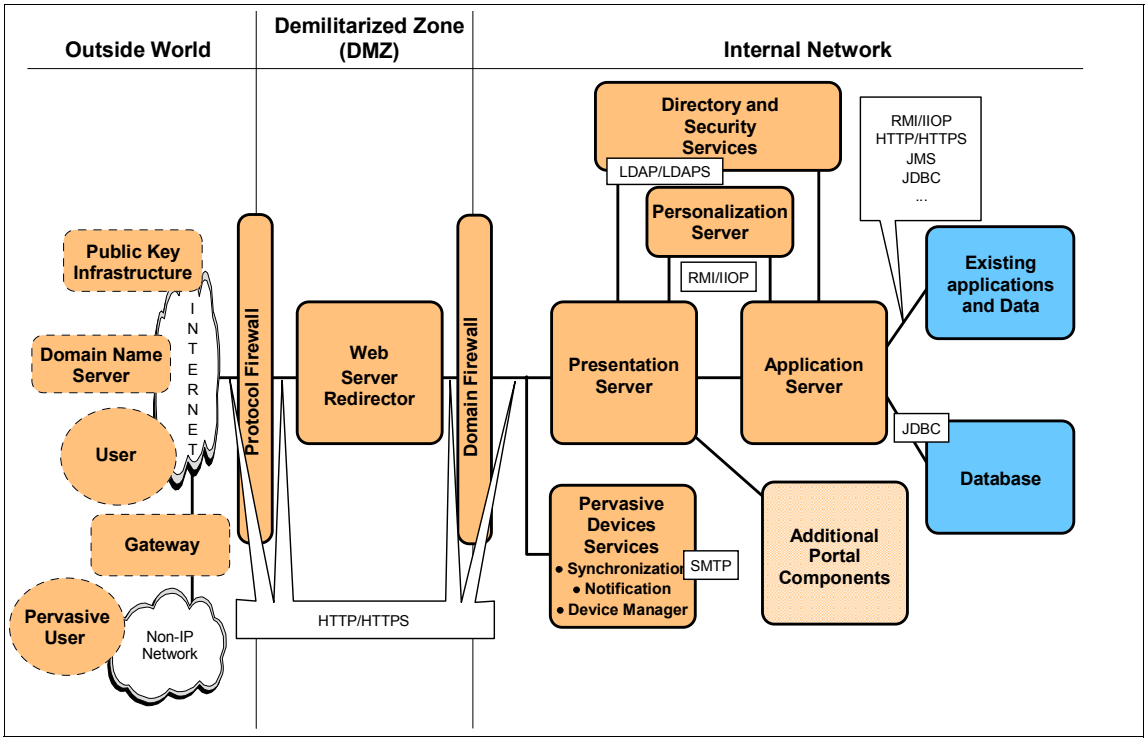


Figure 5-3 Protocol mapping

As shown in Figure 5-3, the network protocols used for the Windows implementation are as follows:

- HTTP/HTTPS: Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) are used from the user's Web browser to the HTTP server in the Web server redirector node.

  HTTP or HTTPS are also used from the WebSphere Web server plug-in in the Web server redirector node to the Web container in the presentation server node, as well as from the collaboration and content management node to the presentation server node.

- LDAP/LDAPS: the presentation and application server uses Lightweight Directory Access protocol (LDAP) to access the LDAP server in the Directory and Security Services node. LDAPS is the secure LDAP connection to a directory server using SSL. Since LDAP directories store essential and sensitive applications and business information, the communication can use LDAPS to be secure.

- SMTP: the notification server can use the SMTP protocol to send messages (mail) to the clients. Using SMTP gateways, the mail or message can be

transmitted through different protocols and transports, for example SMS (GSM network).

- JDBC: the Application Server node and the Directory and Security Services node uses a Java database connectivity (JDBC) driver to access the Database Server node.
- RMI/IIOP: the Personalization Server node uses Remote Method Invocation (RMI) over Internet Inter-Orb Protocol (IIOP) to access the EJB container in the Presentation Server node and the EJB container in the Application Server node.

  RMI/IIOP is also used from the Presentation Server node to the EJB container in the Application Server node.

5.2 Products

In order to have a better understanding of the entire solution and its components, each product will be briefly described.

The product used in this redbook is WebSphere Everyplace Access V4.2. This offering includes the following integrated products:

- WebSphere Portal Server
- WebSphere Application Server
- WebSphere Personalization Server
- Everyplace Intelligent Notification Services
- Everyplace Synchronization Server
- Domino™ Everyplace Enterprise Server (we did not use this product in our scenario in this redbook)
- Device Manager
- DB2®
- DB2 Everyplace
- SecureWay® Directory
- Everyplace Client
- Everyplace toolkit
- WebSphere Studio Site Developer Advanced

WebSphere Everyplace Access V4.2

WebSphere Everyplace Access delivers the technology needed to give mobile access to productivity data and enterprise applications from virtually anywhere, at any time. It supports multiple pervasive devices such as PDAs and smartphones from a single platform. It helps move business-critical information throughout the organization more efficiently and without boundaries, whether to deliver customer information to field sales professionals, inventory information to warehouse pickers, or personal information management (PIM) data to managers on the move.

Everyplace Access allows customers to start small, then expand over time to a richer set of wireless application functions. It provides an entry level solution as a starting point, and lays the foundation of a scalable infrastructure.

WebSphere Everyplace Access is a platform for mobile applications that provides:

- PIM and e-mail for Lotus® Notes® and Microsoft Exchange PDA clients (we did not use this function in our scenario in this redbook)
- All of the components required to extend access to business processes and back-office data to mobile devices
- Notification, synchronization and device management services
- Offline content access services
- A reliable, scalable infrastructure, based on open standards, that easily integrates into your existing IT infrastructure
- The proven technologies of WebSphere Application Server and WebSphere Portal Server
- An application development toolkit that includes samples and a set of plug-ins

The key functionality of Everyplace Access is the support to different devices accessing online and offline applications and content.

Online features

Online Portal content is available to users who are connected to a network, using either a wired or wireless network card. This gives users access to the most current information possible. Everyplace Access provides device-viewing enhancements that improve the mobile Portal experience by enhancing how the Portal looks on selected mobile devices.

Offline features

Since a wireless connection is not always possible or desirable, Everyplace Access makes a selected set of Portal content available for viewing offline as

static content. You can also submit forms while offline. Each time you synchronize, offline content is refreshed and form data is submitted.

Everyplace Access also provides synchronization which utilizes industry standard SyncML technology. The synchronization services, provided with Everyplace Access, allow users to remain productive while offline. Disconnected and connected users can work with the same information and synchronize the information when all users are connected again.

For example, an employee can synchronize their calendar before leaving the office on a sales trip. While the employee is gone, someone else can schedule a meeting on the traveling employee's calendar. At the same time, the traveling employee can schedule a follow-up meeting with a customer on their calendar. When the traveling employee returns to the office and synchronizes again, both new calendar entries are updated on the server and the device.

The following applications and data can be synchronized:

- Lotus Notes PIM and e-mail
- Microsoft Exchange PIM and e-mail
- Domino applications and data
- Any JDBC compliant relational database data

Everyplace Access also provides a client, IBM Everyplace Client, that installs on a mobile device which provides a common interface that initiates synchronization requests.

The WebSphere Everyplace Access is composed of Server, Client and Application Development components.

Server components

1. **WebSphere Everyplace Access Basic Services**

   Basic Services includes:

   – **WebSphere Portal V4.1.4**

     The IBM WebSphere Portal allows you to build your own custom portal Web site. Users can sign on to the portal and receive personalized Web pages providing access to the information, people and applications they need. This personalized single point of access to all necessary resources reduces information overload, accelerates productivity, and increases Web site usage. WebSphere Portal Server allows you to:

     - Build multiple types of portals on a single integrated infrastructure based on the WebSphere Portal Architecture.
     - Provide a scalable, single point of access for data, people, and applications.

- Deliver an easy to use graphical interface suitable for both occasional and expert users.
- Crawl and categorize intranet and Internet repositories.
- Execute a federated search against all forms of data, structured and unstructured.
- Aggregate and summarize content for users.
- Customize the look and content of home page displays by user.
- Build rules-based and collaborative filtering personalization using WebSphere Personalization server.
- Integrate applications and workflow systems into the portal.
- Add collaborative services such as e-mail, shared places, and instant messaging.
- Add pervasive wireless device support for remote and mobile users.
- Provide multiple levels of security and authentication services.
- Leverage syndicated information from over 50,000 databases for news and research.
- Add modules from Independent Software Vendors or custom developed modules.
- Leverage Web site tools for JSP page building, performance monitoring, caching, etc.
- Build next generation Web sites with standards such as XML, SOAP, CORBA, and LDAP.
- Manage users as individuals or within groups.
- Access control at the portlet level.
- Access Lotus and Microsoft Office applications via portlets.
- Implement a distributed, heterogeneous search across disparate data sources.
- Use a flexible architecture that enables integration with your current directory, Database, and security infrastructure.

– **WebSphere Application Server Advanced Edition V4.0.4**

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) and enterprise beans. Some of the Synchronization Server logic is deployed as enterprise applications in WebSphere Application

Server. WebSphere Application Server is the foundation component and is required for WebSphere Portal, Everyplace Synchronization Server, Device Manager and Everyplace Intelligent Notifications Services

– **DB2 V7.2**

  DB2 Universal Database™ is a Web-enabled relational database management system supporting many levels of complexity in database environments.The DB2 is used to support the persistence requirements of all the components of the solution. The Synchronization Server stores user information such as user authentication, device profile preferences, adapter information such as adapter authentication, server monitoring lists and configuration information. The notification stores privacy policies, context data and subscriptions. The device manager stores device management information. The personalization server stores users profile informations and content information.

– **SecureWay Directory V3.2.2**

  Secureway directory is a Lightweight Directory Access Protocol (LDAP) directory that runs as a stand-alone daemon. It is based on a client/server model that provides client access to an LDAP server. SecureWay Directory provides an easy way to maintain directory information in a central location for storage, updating, retrieval, and exchange. WebSphere Portal uses LDAP to store user-specific information. It is an important component when providing a Single Sign-On functionality, working as a centralized authentication information directory.

2. **Everyplace Synchronization Server V1.2**

   Everyplace Synchronization Server is a scalable solution for synchronizing personal information management (PIM) data to back-end databases. The Synchronization Server also provides a powerful relational database synchronization solution using DB2 Everyplace. The Synchronization Server provides a single synchronization point. Clients synchronize to the central Synchronization Server which then synchronizes with back-end databases. This enables clients to share data through the Synchronization Server and get concurrent updates to the same data. Clients can also simultaneously synchronize with multiple databases.

   In an enterprise environment, the Synchronization Server provides adapters to support synchronization with databases such as Lotus Notes, Microsoft Exchange and DB2 (or other JDBC compliant databases). One adapter serves many back-end databases, for a scalable solution.

   To support multiple platforms, Synchronization Server uses the Java 2 platform (J2EE) and SyncML 1.0 protocol between the server and client devices. 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) using TCP/IP.

It includes an optional, high-performance cache for systems with heavy workloads. When caching is enabled, the Synchronization Server replicates the back-end data in order to respond to client synchronization requests quickly.

The Everyplace Synchronization Server adapters provide a neutral interface for accessing different enterprise databases like Lotus Domino, Microsoft Exchange Server or DB2. The adapters convert data to a common format in order to synchronize the data between clients and back-end databases. The Synchronization Server provides the following adapters.

- Lotus Domino Adapter - provides an interface for the Synchronization Server to access enterprise Lotus Domino databases. The Lotus Domino Adapter supports e-mail, journal, to-dos, address book and calendar data.
- Microsoft Exchange Adapter - provides an interface for the Synchronization Server to access Microsoft Exchange databases. The Microsoft Exchange Adapter supports e-mail, tasks, notes, calendar and contacts.
- Relational database adapter using DB2 Everyplace - uses DB2 Everyplace to provide an interface for the Synchronization Server to access DB2 databases or other JDBC compliant relational databases.

The Synchronization Server uses the WebSphere Portal architecture to provide an administration interface using portlets for the Synchronization Server and adapters. The Synchronization Server user and group management is handled through the WebSphere Portal Users and Groups portlet. Setting synchronization preferences for individual users is handled through the WebSphere Everyplace Access PIM portlets.

Synchronization Server also includes:

- **DB2 Everyplace V8.1**

  DB2 Everyplace is a relational database and enterprise synchronization system for mobile and embedded devices. DB2 Everyplace enables enterprise application functionality and enterprise data to be extended to mobile devices such as personal digital assistants (PDAs).

3. **Everyplace Intelligent Notification Services V2.0**

   Intelligent Notification Services allows users to subscribe to services, define the delivery method and specify rules for how and when the information will be delivered. Intelligent Notification Services delivers messages to users based on the users' preferences and subscriptions. For example, users can tell Intelligent Notification Services to send them the URL of any Web-based news article published with "computing technology" in the headline. Users

can also specify message-sending behaviors based on the urgency of the message. For example, if the message is marked FYI, send it to e-mail. If the message is marked urgent, send it to Sametime® instant message.

In order to understand better how the Intelligent Notification Services works, some concepts and components knowledge are required.

– **Notification**

  A notification is a message sent to the subscriber by Intelligent Notification Services. Intelligent Notification Services supports the following types of notifications.

  - Simple notifications - messages, such as personal messages or reminders, that originate from other users or applications. Intelligent Notification users can send simple notifications to one another using the Message Center portlet.
  - Subscription-based notifications - messages that are triggered by events to which the user subscribes. For instance, the subscriber is monitoring stock prices for a certain company and has specified that Intelligent Notification Services notify her when the stock for company XYZ has gone above 90. Another subscriber is watching the activity of company ABC in producing wireless widgets and has specified that Intelligent Notification Services notify him when any articles are published to the Associated Press that relate to company ABC and wireless widgets. Intelligent Notification users can set up subscriptions using the My Subscriptions portlet.

  Intelligent Notification Services supports context-aware delivery of these messages. Context-aware delivery enables the delivery method to be based on a user's context, such as a user's online availability or location. Context parameters that possibly affect the delivery method include a user's location, online presence, and availability.

– **Delivery channels**

  Delivery channels are the mechanisms via which users of Intelligent Notification receive messages. Intelligent Notification Services supports the following types of delivery channels.

  - Message Center portlet - an Intelligent Notification user portlet with which users view messages, delete messages, and send simple notifications to other Intelligent Notification users.
  - Lotus Sametime - an instant messaging server and client pair.
  - Simple Mail Transfer Protocol (SMTP) - a protocol for sending e-mail messages between servers.
  - Short Message Service (SMS) - a service for sending short text messages to mobile devices. SMS requires a Wireless Gateway

(WebSphere Everyplace Connection Manager), which is not provided as part of the WebSphere Everyplace Access package.

– **Components**

  - **Administrative portlets** - the administrator of Intelligent Notification Services uses administrative portlets to manage servers, configure gateway adapters, remove preferences for deleted users, and configure e-mail subscriptions.
  - **User portlets** - a user of Intelligent Notification Services uses portlets to manage delivery channels, manage his or her notification groups, specify rules for message delivery, manage notifications, and subscribe to content sources.
  - **Content adapters** - a content adapter is an application that captures data from an information source and converts that data into a format that the Trigger Manager can read. After converting the data to the correct format, the content adapter publishes the data to the 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.
  - **Gateway adapters** - connect the Universal Notification Dispatcher to supported delivery channels. Each gateway adapter is responsible for delivering messages to the delivery channel that it supports. Each gateway adapter includes transcoding support for converting a notification into a format appropriate for the recipient's device. Intelligent Notification Services provides gateway adapters for the following types of delivery channels.
    – Message Center portlet
    – Lotus Sametime
    – Simple Mail Transfer Protocol (SMTP)
    – Short Message Service (SMS)
  - **Trigger Manager** - a component of Intelligent Notification Services that accepts content from content adapters and subscriptions from Intelligent Notification users. The subscriptions contain criteria for matching content from the content adapters. The Trigger Manager matches the content against subscription criteria. When a match occurs, the Trigger Manager uses a trigger handler to determine how to handle the match and notify the appropriate user via the Universal Notification Dispatcher.
  - **Universal Notification Dispatcher** - a component of Intelligent Notification Services that delivers notifications to users based on their preferences and context. Messages may originate as simple

notifications from other users or applications, or they may result from subscriptions. Messages resulting from subscriptions are generated and sent to the Universal Notification Dispatcher by trigger handlers. Simple notifications are sent directly to the Universal Notification Dispatcher from an application using the sendMessage() method of the NotificationService class.

- **Secure Context Server** - a component of Intelligent Notification Services that enables context-aware notifications by providing context information such as a user's location or Sametime availability. This information may be used to determine where and when to send notifications, based on the user's context preferences. Context information is provided by context drivers. There is a context driver for each specific category of context information. Intelligent Notification Services currently supports a Sametime context driver that provides information about whether a user is online in a Sametime environment.

4. **Device Manager V1.3**

   Device Manager is a device management technology that helps to manage personal digital assistants (PDAs), handheld PCs, smartphones, wireless access protocol (WAP) devices, in-vehicle information systems, or other, emerging devices for pervasive computing. Integrated with a service provider enrollment application or enterprise user database, Device Manager can be used to identify, configure, inventory, and distribute software to any device supported by the service provider or enterprise.

5. **Lotus Domino Everyplace Enterprise Server V2.7**

   Lotus Domino Everyplace Enterprise Server (DEES) is packaged with Everyplace Access to provide the access to existing Domino databases.

Client components

1. **Everyplace Client V4.2**

   Everyplace Client provides a common interface that supports synchronization, security, device management, offline Portal browsing, offline Domino applications, and DB2 Everyplace database replication. Everyplace Client provides a single user control interface for the following client agents and applications:

   – Lotus Mobile Notes®
   – e-mail and PIM function (SyncML Client)
   – Database (DB2 Everyplace)
   – Offline Portal Browsing
   – Offline Form submission
   – IBM device agent

2. **TrueSync Plus V3.1**

This is a multi-point synchronization engine that provides synchronization of calendar, address book, to do list, and memo between multiple information sources including mobile devices, organizer applications, and servers. This single-step process overcomes the fundamental inefficiency of traditional point-to-point synchronization technology that requires users to perform a sequential device-to-desktop, desktop-to-server, and desktop-to-device synchronization.

Application development components

1. **Everyplace Toolkit V4.2**

   The Everyplace Toolkit provides a comprehensive toolkit for the development of portlet applications. The toolkit is implemented as a plug-in to WebSphere Studio Site Developer Advanced or WebSphere Studio Application Developer. In this release, Everyplace Toolkit provides:

   – Portlet Projects, in which you can create Abstract portlets, JSP portlets, Servlet Invoker portlets, XML/XSL portlets, and Multi-Device/View portlets.

   – Portlet application samples for enterprise applications.

2. **WebSphere Studio Site Developer Advanced V4.0.3**

   Studio Site Developer is an easy-to-use tool set that minimizes the time and effort required to create, manage, and debug multi-platform Web sites. It is designed to the J2EE specifications and delivers integrated support for open Web standards, including Java, JSP, servlets, XML, full HTML, DHTML, JavaScript, rich media and Web services tools. It includes an advanced Java IDE and tools for developing images and animated GIFs. It also allows you to use your favorite content creation tools in conjunction with its built-in local and remote publishing capabilities.

5.3 Considerations

It is suggested that you make the final platform recommendation based on the following considerations:

- **Platform** - Choosing the right platform depends on several other elements. The platform in this context means the operating system and the underlying hardware.
- **Homogenous/heterogeneous solution** - This is closely related to the platform, whether the solution deals with different platforms (heterogeneous) or just one (homogenous).
- **Availability** - Availability is very complex from an architecture standpoint. It is related to the number of systems. The higher the redundancy, the higher the availability; this also means higher costs. It is very important to ensure just the

right level of availability; less than required can cause loss of business, more than required can raise the costs.

- **Performance** - Performance has influence on the end user experience and has a significant impact on the business. Users want fast responses, and the system has to perform well, no matter what kind of application is involved.
- **Security** - Security has always been an issue. It depends on the business, and becomes very important for mission-critical applications. Just as with availability, the right level of security has to be found. Security can be very strict or stay at a basic level; this depends on the requirements.

System management

Solutions cannot stand alone without any system management. e-business solutions are moving towards centralized system management, but users still have to deal with multiple tools to manage the system within their solution.

Figure 5-4 on page 71 shows the framework of a complete Pervasive Portal solution and the relationship between the components. The solution supports internal users (intranet) and external users (Internet). For external users, the only component different from those for internal users that have to be addressed is the security represented in the diagram by the Security Layer for external access block. The black captions represent the logical nodes and the white captions represent the product. The blocks below the Notification, Synchronization and Device Manager represent their components.

The diagram shows how the connectivity is provided to the Portal for each type of device. Wireless devices need a wireless gateway, voice devices also need a Voice Server; all the others access the portal directly.

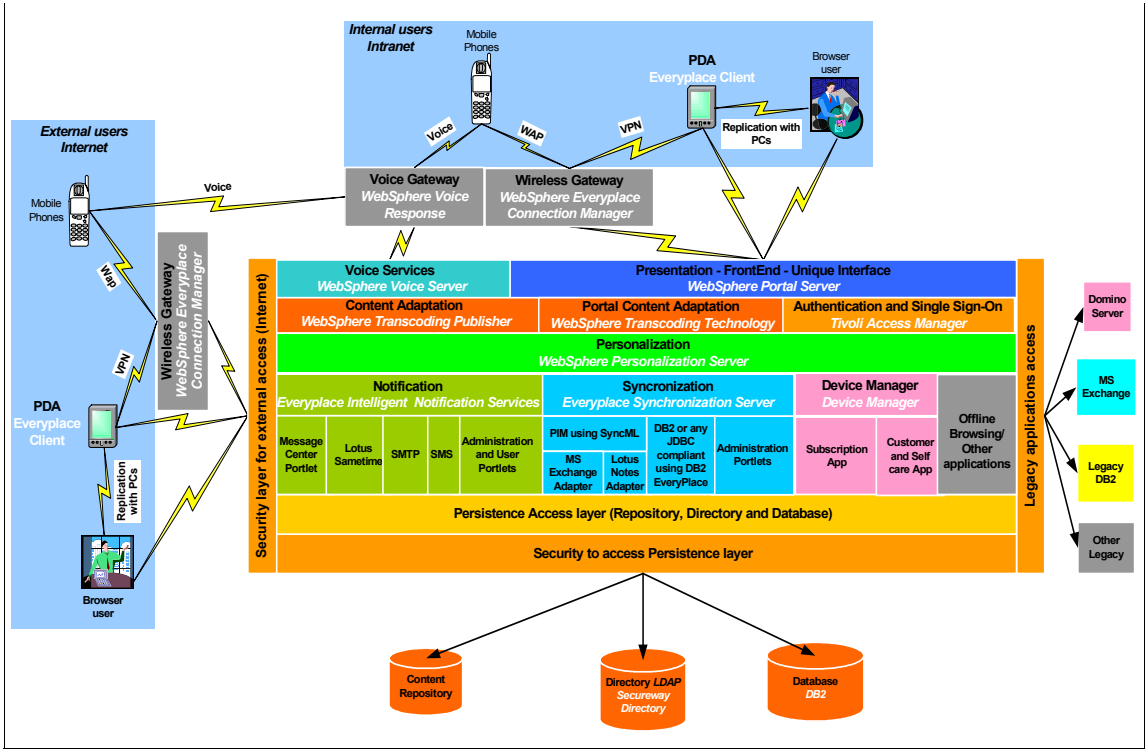


Figure 5-4 Pervasive Portal solution framework

5.4 Where to find more information

- For more information about Patterns for e-Business, go to:
  http://www-106.ibm.com/developerworks/patterns/
- For more information about WebSphere Everyplace Access, go to:
  http://www.ibm.com/software/pervasive/products/mobile_apps/ws_everyplace_access.shtml
- For more information about WebSphere Application Server, go to:
  http://www.ibm.com/software/webservers/appserv/
- For more information about WebSphere Portal Server, go to:
  http://www.ibm.com/software/webservers/portal/
- For more information about DB2, go to:
  http://www.ibm.com/software/data/db2/udb/

- For more information about WebSphere Everyplace Connection Manager, go to: http://www.ibm.com/software/pervasive/products/mobile_sols/wireless_gateway.shtml
- For more information about Tivoli® Access Manager, go to: http://www.tivoli.com/products/index/access-mgr-e-bus/
- For more information about WebSphere Voice Server, go to: http://www.ibm.com/software/pervasive/products/voice/voice_server.shtml
- For more information about WebSphere Transcoding Publisher, go to: http://www.ibm.com/software/pervasive/products/mobile_sols/transcoding_publisher.shtml
- For more information about WebSphere Edge Server, go to: http://www.ibm.com/software/webservers/edgeserver/
- For more information about WebSphere Studio Site Developer, go to: http://www-3.ibm.com/software/ad/studiositedev/

Part 2

Pervasive Portal solution guidelines

© Copyright IBM Corp. 2003. All rights reserved.

6

Technology options

We now take a look at the Web application technology options you should consider in this chapter. The recommendations are guided by the demands of reuse, flexibility, and interoperability, and subsequently are based on the open industry standards outlined by Java 2 Platform, Enterprise Edition (J2EE). Many of the choices continue to evolve and expand as the J2EE specification matures to include a broader view of the enterprise architecture. These recommendations are based on the J2EE1.3 specification. In this chapter, we are going to show some of the current technologies to implement wireless and pervasive solutions.

To read more about mobile technologies, refer to the IBM Redbooks entitled *Mobile Commerce Solutions Guide using WebSphere Commerce Suite V5.1*, SG24-6171, and *Mobile Applications with IBM WebSphere Everyplace Access Design and Development*, SG24-6259-00

© Copyright IBM Corp. 2003. All rights reserved.

6.1 Web client

Figure 6-1 shows the recommended technologies for Web clients.

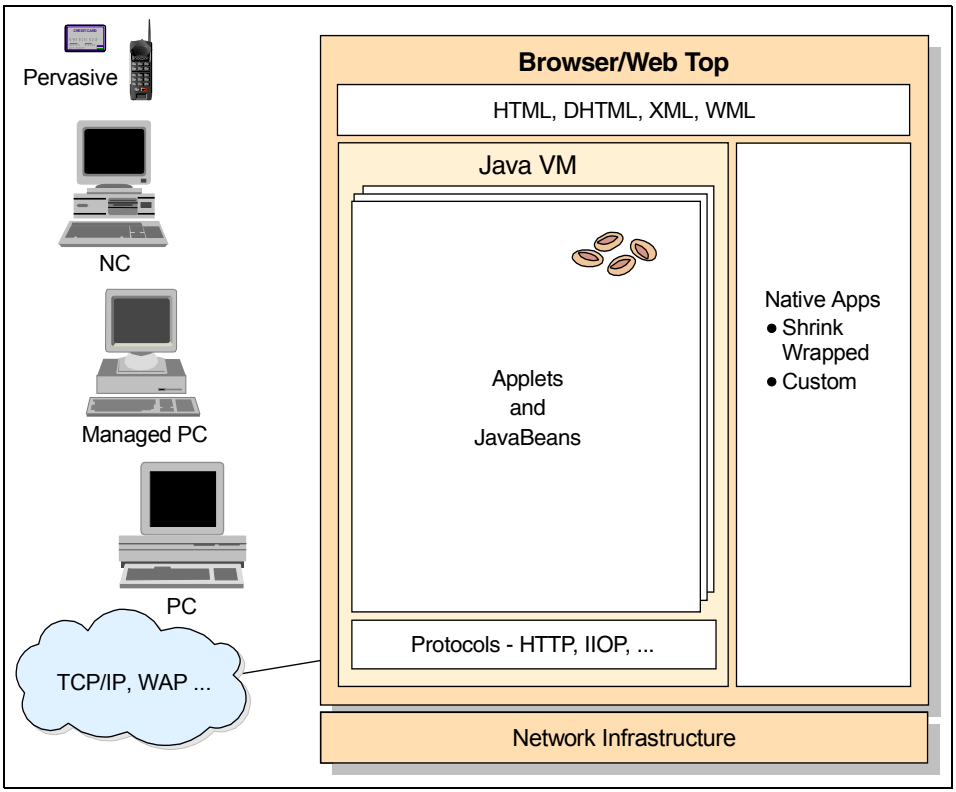


Figure 6-1 Web client technology model

The clients are “thin clients” with little or no application logic. Applications are managed on the server and downloaded to the requesting clients. The client portions of the applications should be implemented in HTML, dynamic HTML (DHTML), XML, and Java applets.

The selection of client-side technologies used in your design will require consideration for the server side, such as whether to store, or dynamically create, elements for the client side.

The following sections outline some of the possible technologies that you should consider, but remember that your choices may be constrained by the policy of your customer or sponsor. For example, for security reasons, only HTML is allowed in the Web client at some government agencies.

We also touch on some of the current technology choices in the wireless area.

6.1.1 Web browser

A Web browser is a fundamental component of the Web client. For PC-based clients, the browser typically incorporates support for HTML, DHTML, JavaScript, and Java. Some browsers are beginning to add support for XML as well. Under user control, there is a whole range of additional technologies that can be configured as “plug-ins”, such as RealPlayer from RealNetworks or Macromedia Flash.

As an application designer, you must consider the level of technology you can assume will be available in the user’s browser, or you can add logic to your application to enable slight modifications based upon the browser level. For Internet users, this is especially true. With intranet users, you can assume support for a standard browser. Regarding plug-ins, you need to consider what portion of your intended user community will have that capability.

Cross-browser strategies are required to ensure robust application development. Although many of these technology choices are maturing, they continue to be inconsistently supported by the full range of browser vendors. Developers must know browser compatibility for all features being exploited by the application. In general, developers will need to code to a lowest denominator or at least be able to distinguish among browser types using programmatic techniques. The key decision here is to determine the application requirements and behavior when handled by old browsers, other platforms such as Linux and Mac, and even the latest browsers.

In the J2EE model, the Web browser plays the role of client container. The model requires that the container provide a Java Runtime Environment as defined by the Java 2 Platform, Standard Edition (J2SE). However, for an e-business application that is to be accessed by the broadest set of users with varying browser capabilities, the client is often written in HTML with no other technologies. On an exception basis, limited use of other technologies, such as using JavaScript for simple edit checks, can then be considered based on the value to the user and the policy of the organization for whom the project is being developed.

The emergence of pervasive devices introduces new considerations to your design with regard to the content streams that the device can render and the more limited capabilities of the browser. For example, WAP (Wireless Application Protocol) enabled devices render content sent in WML (Wireless Markup Language).

6.1.2 HTML

HTML (HyperText Markup Language) is a document markup language with support for hyperlinks that is rendered by the browser. It includes tags for simple form controls. Many e-business applications are assembled strictly using HTML. This has the advantage that the client-side Web application can be a simple HTML browser, enabling a less capable client to execute an e-business application.

The HTML specification defines user interface (UI) elements for text with various fonts and colors, lists, tables, images, and forms (text fields, buttons, checkboxes, and radio buttons). These elements are adequate to display the user interface for most applications. The disadvantage, however, is that these elements have a generic look and feel, and lack customization. As a result, some e-business application developers augment HTML with other user-interface technologies to enhance the visual experience, subject to maintaining access by the intended user base and compliance with company policy on Web client technologies.

Because most Web browsers can display HTML V3.2, this is the lowest common denominator for building the client side of an application. To ensure compatibility, developers should be unit testing pages against a validator tool. Free tools, such as the W3C HTML Validation Service, are available at:

`http://validator.w3.org/`

6.1.3 Dynamic HTML

DHTML allows a high degree of flexibility in designing and displaying a user interface. In particular, DHTML includes Cascading Style Sheets (CSS) that enable different fonts, margins, and line spacing for various parts of the display to be created. These elements can be accurately positioned using absolute coordinates. See 6.1.4, “CSS” on page 79 for details on Cascading Style Sheets.

Another advantage of DHTML is that it increases the level of functionality of an HTML page through a document object model and event model. The document object enables scripting languages such as JavaScript to control parts of the HTML page. For example, text and images can be moved about the window, and hidden or shown, under the command of a script. Also, scripting can be used to change the color or image of a link when the mouse is moved over it, or to validate a text input field of a form without having to send it to the server.

Unfortunately, there are several disadvantages when using DHTML. The greatest of these is that two different implementations (Netscape and Microsoft) exist and are found only on the more recent browser versions. A small, basic set of functionality is common to both, but differences appear in most areas. The significant difference is that Microsoft allows the content of the HTML page to be modified by using either JScript or VBScript, while Netscape allows the content to be manipulated (moved, hidden, shown) using JavaScript only.

Due to varying levels of browser support, cross-browser design strategies must be used to ensure appropriate presentation and behavior of DHTML elements. In general, this technology is not recommended unless its features are needed to meet usability requirements.

6.1.4 CSS

Cascading Style Sheets (CSS) allow you to define a common look and feel for HTML documents. This specification describes how Web documents are to be presented in print and online.

CSS is defined as a set of rules that are identified by selectors. When processed by the client browser, the selectors are matched to specific HTML tags and then are applied against the properties of the tag. This allows for global control over colors, fonts, margins, and borders. More advanced commands allow for control over pixel coordinates. Related stylesheet commands can be grouped and then externalized as a separate template file to be referenced by a multitude of Web pages.

CSS is defined as level 1 and level 2 specifications. Level 1 was written with HTML in mind, while level 2 was expanded to include general markup styles for XML documents. Developers using CSS should unit test against a validator tool, such as the W3C CSS Validation Service at:

```
http://jigsaw.w3.org/css-validator/
```

Due to varying levels of browser support, cross-browser design strategies must be used to ensure appropriate presentation and behavior of CSS elements. In general, this technology should be used with great attention to support of specification elements.

6.1.5 JavaScript

JavaScript is a cross-platform object-oriented scripting language. It has great utility in Web applications because of the browser and document objects that the language supports. Client-side JavaScript provides the capability to interact with HTML forms. You can use JavaScript to validate user input on the client and help improve the performance of your Web application by reducing the number of requests that flow over the network to the server.

ECMA, a European standards body, has published a standard (ECMA-262) that is based on JavaScript (from Netscape) and JScript (from Microsoft), called ECMAScript. The ECMAScript standard defines a core set of objects for scripting in Web browsers. JavaScript and JScript implement a superset of ECMAScript.

To address various client-side requirements, Netscape and Microsoft have extended their implementations of JavaScript in version 1.2 by adding new browser objects. Because Netscape's and Microsoft's extensions are different from each other, any script that uses JavaScript 1.2 extensions must detect the browser being used, and select the correct statements to run.

One caveat is that users can disable JavaScript on the client browser, but this can be programmatically detected.

The use of JavaScript on the server side of a Web application is not recommended, given the alternatives available with Java. Where your design indicates the value of using JavaScript, for example for simple edit checking, use JavaScript 1.1, which contains the core elements of the ECMAScript standard.

6.1.6 Java applets

The most flexibility of the user interface (UI) technologies that can be run in a Web browser is offered by the Java applet. Java provides a rich set of UI elements that include an equivalent for each of the HTML UI elements. In addition, because Java is a programming language, an infinite set of UI elements can be built and used. There are many widget libraries available that offer common UI elements, such as tables, scrolling text, spreadsheets, editors, graphs, charts, and so on.

You can use either the AWT or the Swing classes to build a Java applet. But while designing your applet, you should keep in mind that Swing is supported only by later browser versions.

A Java applet is a program written in Java that is downloaded from the Web server and run on the Web browser. The applet to be run is specified in the HTML page using an APPLET tag:

```
<APPLET CODEBASE="/mydir" CODE="myapplet.class" width=400 height=100>
  <PARAM NAME="myParameter" VALUE="myValue">
</APPLET>
```

For this example, a Java applet called “myapplet” will run. An effective way to send data to an applet is by using the PARAM tag. The applet has access to this parameter data and can easily use it as input to the display logic.

Java can also request a new HTML page from the Web application server. This provides an equivalent function to the HTML FORM submit function. The advantage is that an applet can load a new HTML page based upon the obvious (a button being clicked) or the unique (the editing of a cell in a spreadsheet).

A characteristic of Java applets is that they seldom consist of just one class file. On the contrary, a large applet may reference hundreds of class files. Making a request for each of these class files individually can tax any server and also tax the network capacity. However, packaging all of these class files into one file reduces the number of requests from hundreds to just one. This optimization is available in many Web browsers in the form of either a JAR file or a CAB file. Netscape and HotJava support JAR files simply by adding an ARCHIVE="myjarfile.jar" variable within the APPLET tag. Internet Explorer uses CAB files specified as an applet parameter within the APPLET tag. In all cases, executing an applet contained within a JAR/CAB file exhibits faster load times than individual class files. While Netscape and Internet Explorer use different APPLET tags to identify the packaged class files, a single HTML page containing both tags can be created to support both browsers. Each browser simply ignores the other's tag.

JavaScript can be used to invoke methods on an applet using the SCRIPT tag in the applet’s HTML page.

A disadvantage of using Java applets for UI generation is that the required version of Java must be supported by the Web browser. Thus, when using Java, the UI part of the application will dictate which browsers can be used for the client-side application. Note that the leading browsers support variants of the JDK 1.1 level of Java and have different security models for signed applets.

Using Java plug-ins, you can extend the functionality of your browser to support a particular version of Java. Java plug-ins are part of the Java Runtime Environment (JRE) and they are installed when the JRE is installed on the computer. You can specify certain tags in your Web page, to use a particular JRE. This will download the particular JRE if it is not found on the local computer. This can be done in HTML either through:

- The conventional APPLET tag, or
- The OBJECT tag instead of the APPLET tag for Internet Explorer or the EMBED tag with the APPLET tag for Netscape.

A second disadvantage of Java applets is that any classes such as widgets and business logic that are not included as part of the Java support in the browser must be loaded from the Web server as they are needed. If these additional classes are large, the initialization of the applet may take from seconds to minutes, depending upon the speed of the connection to the Internet.

Using HTTP tunneling, an applet can call back on the server without reloading the HTML page. For users who are behind a restrictive firewall, HTTP tunneling offers a bidirectional data connection to connect to a system outside the firewall.

Because of the above shortcomings, the use of Java applets is not recommended in environments where mixed levels and brands of browsers are present. Small applets may be used in rare cases where HTML UI elements are insufficient to express the semantics of the client-side Web application user interface. If it is absolutely necessary to use an applet, care should be taken to include UI elements that are core Java classes whenever possible.

6.1.7 XML (client side)

XML allows you to specify your own markup language with tags specified in a Document Type Definition (DTD) or XML Schema. Actual content streams are then produced that use this markup. The content streams can be transformed to other content streams by using XSL (Extensible Stylesheet Language), which is based on CSS.

For PC-based browsers, HTML is well established for both document content and formatting. The leading browsers have significant investments in rendering engines based on HTML and a Document Object Model (DOM) based on HTML for manipulation by JavaScript.

XML seems to be evolving to a complementary role for active content within HTML documents for the PC browser environment.

For new devices, such as WAP-enabled phones and voice clients, the data content and formatting is being defined by new XML schema, WML for WAP phone and VoiceXML for voice interfaces.

For most Web application designs, you should focus your attention on the use of XML on the server side.

6.1.8 XHTML 1.1 (HTML 4.01)

XHTML (Extended HyperText Markup Language) is an extension to HTML 4, which supports document types that are XML-based. It is intended to be used as a language for XML-conforming content as well as for HTML 4-conforming user agents.

The advantages of XHTML are as follows:

- Since XHTML documents are XML conforming, they can be viewed, edited, and validated with standard XML tools.
- XHTML documents can be used to traverse either the HTML Document Object Model or the XML Document Object Model.

Some issues with XHTML are:

- XHTML documents are not as easy to create as HTML documents because XHTML is validated more strictly than HTML.
- HTML is already used so widely that it is difficult for XHTML to attract the attention of most Web developers.
- Browser support is not usually an issue since documents can be created using HTML-compatible XHTML that is understood by most browsers. There are also utilities that can be used to convert HTML documents to HTML-compatible XHTML.
- Development tool support for XHTML is also improving. The Page Designer tool in IBM WebSphere Studio Application Developer V5.0, for example, allows visual authoring of XHTML pages.

XHTML Basic is designed for Web clients that do not support the full set of XHTML features. It is meant to serve as a common language and share basic content across mobile phones, pagers, car navigation systems, vending machines, etc.

Some of the common features found in Wireless Markup Language (WML) and other subsets of HTML have been used as the basis for developing XHTML Basic:

- Basic text
- Basic forms and tables
- Hyperlinks

Some HTML 4 features have been found inappropriate for non-desktop devices, so extending and building on XHTML Basic will help to bridge that gap.

6.1.9 XForms

XForms is W3C's specification for Web forms that can be used with desktop computers, hand-held devices, etc. The disadvantage of the HTML Web forms is that there is no separation of purpose from presentation. XForms separates the data and logic of a form from its presentation. Also, XForms are device-independent.

XForms uses XML for transporting the data that is displayed on the form and the data that is submitted from the form. HTML is used for the data display.

Currently, the main issue with XForms is that it is still an emerging technology, so browser and server support is not yet standard.

6.2 Pervasive clients

Pervasive, mobile, wireless and many other terms constitute a confusing terminology which makes it difficult to have a common understanding of our topic. The problem always starts with classifying sophisticated systems, when it is discovered that the terms mean different things to different people.

Mobile in this context means that the information is not only accessible from a desktop browser but also from different types of devices, using different connections, from different locations. Wireless means you are not connected to the network using a cable. Mobile clients include wireless devices such as phones, pagers, and PDAs. Pervasive computing is computing power freed from the desktop. It's a step ahead in the mobile and wireless arena. It could be embedded in wireless handheld devices, automobile telematics systems, home appliances, commercial tools-of-the-trade or a smart card that you carry with you. Pervasive computing is convenient access to relevant information with the ability to easily take action on it, when and where you need to.

6.2.1 Architecture

Support for mobile clients impacts the runtime topology and therefore must be designed and implemented using best practices for system architecture. The good news is that any past investment in Web architecture to support Internet-based applications can be extended to support mobile clients.

A Wireless Application Protocol (WAP) gateway is used between the mobile client device and the Web server. The gateway translates requests from the wireless protocol into HTTP requests and, conversely, converts HTTP requests into the appropriate device format.

6.2.2 WAP

WAP is the Wireless Application Protocol. This is the standard for presentation and delivery of information to wireless devices, which are platform, device and network neutral. The goal of this protocol is to provide a platform for global, secure access through mobile phones, pagers, and other wireless devices.

6.2.3 Microbrowser

WAP microbrowsers run on mobile clients. They are responsible for the display of Web pages written in WML and can execute WMLScripts. These play the same role as HTML browsers that run on a PC.

6.2.4 WML

The Wireless Markup Language (WML) is based on XML and HTML 4.0 to fit small hand-held devices. It is a tag-based language that handles formatting static text and images, can accept data input, and can follow hyperlinks. WML also uses WMLScript, a compact JavaScript-like language that runs in limited memory. WML is the markup language of WAP.

The WML specification is maintained by The Open Mobile Alliance. The Open Mobile Alliance has been established by the consolidation of the WAP Forum and the Open Mobile Architecture Initiative, two industry-wide consortiums concerned about the development of an open standard for the wireless industry.

For more information, you can visit The Open Mobile Alliance Web site at:

`http://www.openmobilealliance.org` or
`http://www.wapforum.org`

6.2.5 WMLScript

This is the companion language to WML, in the same way that JavaScript is a companion language to HTML. WMLScript allows for procedural programming such as loops, conditional and event handling. It has been optimized for a small memory footprint and small devices. This language is derived from JavaScript.

6.2.6 cHTML

cHTML stands for Compact HTML and is a subset of the HTML specifications targeting small appliances such as smartphones and mobile PDAs. The cHTML tries to bypass several hardware restrictions by providing a standard markup language and small browser that could be executed in a constrained environment with a small memory, low power CPU, a small display, etc.

Since the Compact HTML is based on standard HTML recommendations from W3C, we can develop and apply software tools to adapt pure HTML to cHTML, making Internet information available and adequately formatted to new classes of devices and appliances. Basically, cHTML excludes JPEG images, tables, image map, multiple character fonts and styles, background color or images, frames, and cascading style sheets from the HTML specification.

cHTML is the markup language of i-Mode. i-Mode is a wireless service developed by NTT DoCoMo in Japan. It is designed to provide mobile phone voice service, Internet and e-mail access.

For more information about Compact HTML, you can read the document submitted to W3C, World Wide Web Consortium, at:

http://www.w3.org/TR/1998/NOTE-compactHTML-19980209.

6.2.7 VoiceXML

The Voice eXtensible Markup Language (VoiceXML) is an XML-based industry standard language for creating voice applications, much as HTML is a language for developing visual applications.

VoiceXML is defined and promoted by an industry forum, the VoiceXML Forum, founded by AT&T, IBM, Lucent and Motorola, and currently supported by more than 570 member companies.

VoiceXML was designed to create audio dialogs that feature text-to-speech, digitized as well as prerecorded audio, recognition of both spoken and dual-tone multi-frequency (DTMF) key input, recording of spoken input, telephony, and mixed-initiative conversations. Its goal is to provide voice access to Web-based content and applications. It enables the development of voice applications via the

use of a familiar markup style and Web server-side logic to deliver applications over telephone lines. The resulting applications allow conversational access to Web-based data, and can also interact with existing back-end business data and logic.

A VoiceXML application is capable of retrieving information from a Web server and, by making use of scripts and appropriate grammars, the application can interact with the customer through spoken words.

For more information, you can visit the VoiceXML Forum Official Web site at:

http://www.voicexml.org

X+V

X+V is an abbreviation of XHTML + VoiceXML and it is a markup language specification submitted to the World Wide Web Consortium (W3C) by IBM, Motorola and Opera Software to simplify the development of multimodal applications.

Multi-modal access will give users of pervasive devices (smartphones, PDAs, kiosks, set-top-boxes, etc.) a range of options for interacting with an application. To input information, they might use some combination of voice, keypad, stylus, touchscreen, and the application would deliver information using a combination of speech synthesis, text, graphics, A/V, etc.

X+V allows you to operate in a voice-only environment; in a visual-only environment, and if you want, in a multimodal environment.

6.2.8 SyncML

SyncML stands for Synchronization Markup Language and it is a an open standard protocol for data synchronization optimized for wireless networks. The SyncML consortium is sponsored by IBM, Nokia, Symbian, Ericsson, Matsushita, Motorola, Nokia, Openwave, Starfish Software and Symbian. It is being supported by the leading wireless companies including Apple Computer, Siemens AG, Vodafone Group, France Telecom and America Online.

The goal of SyncML is to enable synchronization of any type of data, from any application, on any device, and over any network. It has been designed to cope well with the specificities of mobile phones such as low bandwidth, unreliable connections and high network latency.

For more information, you can visit the SyncML Official Web Site at:

http://www.syncml.org

6.2.9 Mobile devices

Today, we have thousands of mobile devices to choose from. Mobile clients include wireless devices such as phones, smartphones, PDAs and even laptops. When deciding which one is more suitable for your specific application, you should analyze several factors such as the mobile platform, the mobile application platform availability device and network connectivity options. The goal is to overcome some device limitations to provide access to information and application services from anywhere.

Mobile devices include wireless desktop PCs, WAP devices, i-mode devices, PDAs, and Phone w/Voice. PDA devices cannot run the major operating systems that run on desktop PCs and consequently there are various mobile device-specific platforms. Palm devices use Palm OS. WinCE/PocketPC devices use a version of Microsoft Windows called Windows CE.

Voice-enabled applications allow for a hands-free user experience unencumbered by the limitations of computer interface controls.

Voice technology fall into two categories: those that recognize speech and those that generate speech. The ability to recognize human voice by computers is called Automatic Speech Recognition (ASR). The ability to generate speech from written text is called speech synthesis or Text-to-Speech (TTS).

Mobile devices categories

We have placed the mobile devices into three categories to better describe the specific features that they offer: mobile phones, smartphones and PDAs.

Mobile phones

When we refer to a mobile phone in this redbook, we are talking about a cellular phone that has a microbrowser to access Internet content. A standard cellular phone includes voice capabilities, messaging features (SMS or WAP push) and data functionality (can access Internet content through a microbrowser).

When selecting a device to a new application, you should also consider some industry-specific ruggedized devices. They have similar PDA functionality but they are more robust to work in hazard environments. Symbol and Intermec are two manufacturers specialized in these devices.

Smart phones

This is a generic name for voice-centric mobile phones with information capability. It's a hybrid device that combines the PDA and cellular phone functionalities. Several models have been launched recently and we expect to see more new models.

PDA (Personal Digital Assistant)

A Personal Digital Assistant (PDA) is a handheld computer with a wireless interface that serves as an organizer for personal information. PDAs often have a pen-based stylus to tap selections on menus and to enter printed characters. The unit may also include a small on-screen keyboard that is tapped with the pen. Data is synchronized between the PDA and desktop computer via cable or wireless transmission. We are interested in PDAs that have wireless transmission capability and include a Web browser. The major operating systems for PDAs are Palm OS, Epoc, and Windows CE.

Wireless laptops

This category includes laptops, notebooks, or portable PC browser clients that have a wireless interface to the network for Internet access. These clients use standard TCP/IP protocols and a standard browser, such as Netscape Navigator or Microsoft Internet Explorer. The wireless connection is usually much slower than wireline-based network clients.

6.2.10 Mobile client platforms

There are numerous mobile client platforms on the market today. They all target the same objectives: small footprint, optimized user interface, simple operation. This section gives a quick overview of the most common mobile client platforms.

Microsoft Pocket PC

The PocketPC operational system is based on the Microsoft Windows CE 3.0 (WinCE) operating system. It's is similar to the well-known Windows operating system, but optimized and developed especially for PocketPC mobile devices.

For more information, you can visit the Microsoft Web site at:

http://www.microsoft.com/mobile/pocketpc

Microsoft Windows Smartphone platform

Like PocketPC, Microsoft a smartphone platform also based on Windows CE V3.0. It combines voice and text communication and data applications with a similar look and feel. Like the J2ME and BREW platforms, it can run online and disconnected applications.

For more information, you can visit the Microsoft Web site at:

http://www.microsoft.com/mobile/smartphone

Palm OS

The base for this solution is not the device, but the operating system (OS). The OS called PALM OS is used with a wide range of devices, including the Palm, Sony and HandSpring devices. This device has built-in mobile capabilities, and works as a mobile phone with a data connection.

Like many other operating systems on the market, the Palm OS comes in different editions. The most widely used editions are V3.x and the latest, V4.0. All the versions are improvements on the previous versions. It is important to note that some of the software relies on a specific version of the OS.

For more information, you can visit the Palm OS official Web site at:

http://www.palmsource.com.

Symbian OS

Symbian is an open standard operating system for data-enabled mobile phones. It includes a multi-tasking multithreaded core, a user interface framework, data services enablers, application engines and integrated PIM functionality and wireless communications. It is present in several smartphones including Ericsson R380 Smartphone, Nokia 3650, Nokia 9290 and 9210 Communicator, Nokia 3650.

For more information, you can visit the Symbian Web site at:

http://www.symbian.com.

J2ME

J2ME stands for Java 2 Platform Micro Edition; it is a very small Java application environment suitable for several segments such as a TV set-top box, mobile phones and PDAs. The J2ME platform is composed of standard Java APIs and runtime environment to handle the user interface, security and network protocols. It also support online and disconnected applications; for example, handset users can download and run applications in the devices themselves, without the need for a continuous connection. The Motorola i85s and Nokia 8910i are examples of handsets with J2ME support.

For more information, you can visit the SUN Microsystems J2ME official Web site at:

http://java.sun.com/j2me.

Java Midlet

Midlet stands for Mobile Information Device Application. The latest generation of mobile phones uses a reduced version of J2ME. This version of Java has been

specifically adapted for mobile information devices (MIDs), including mobile phones under the MID Profile.

BREW

BREW stands for Binary Runtime Environment for Wireless and it is an application platform execution environment for wireless devices developed by QUALCOMM. The BREW platform is part of an end-to-end solution for wireless applications development, device configuration, application distribution, and billing which will enable services providers and carriers' customers, for example, to download new applications over the air and pay for them. The target devices are cellular phones like Motorola T720 and Samsung SPH-X2700.

For more information, you can visit the Qualcomm BREW official Web site at:

http://www.qualcomm.com/brew.

6.3 Wireless networks

Wireless networks are used to transmit data between mobile devices or personal computers using wireless adapters without the use of a physical cable or wire. Depending on the coverage range, these networks could be classified as:

- PAN (Personal Area Network)
- WLAN (Wireless Local Area Network)
- WWAN (Wireless Wide Area Network)

6.3.1 PAN (Personal Area Network)

A PAN has a very short range, usually up to 10-20 feet. The main objective of this network is to interconnect personal devices, for example a cell phone to a PDA or keyboard to a desktop computer. It is also used to connect these with other available devices in the environment, for example a cell phone with vending machines. Several technologies could be used to implement a PAN, such as Infrared and Bluetooth.

Infrared technology

Infrared is an invisible band of radiation at the lower end of the electromagnetic spectrum; it starts in the middle of the microwave spectrum and goes up to the beginning of visible light. Infrared transmission requires an unobstructed line of sight between transmitter and receiver. It is used for short range point-to-point wireless transmission between computer devices, as well as many handheld remotes for TVs and video and stereo equipment.

The Infrared technology standards are controlled by IrDA, Infrared Technology Data Association, to ensure interoperability between devices of all types. For more information, you can visit the official IrDA Web site at:

http://www.irda.org

Bluetooth

Bluetooth is an open technology specification created for short-range (up to 10 meters) wireless connection using low-cost transceiver chips to be embedded in mobile PCs, smart phones, and other portable devices. It provides three voice and data channels via a one-to-one connection with built-in encryption and verification. The objective is to connect devices and establish ad-hoc connections in a peer-to-peer mode, using the unlicensed 2.4GHz frequency. This technology can transfer data at up to 1 Mb per second. The Bluetooth technology signals are omni-directional, thus eliminating the need for line-of-sight.

The Bluetooth technology is governed by the Bluetooth Special Interest Group (SIG) which includes promoter group companies 3Com, Agere, Ericsson, IBM, Intel®, Microsoft, Motorola, Nokia and Toshiba, and more than 2000 Associate and Adopter member companies representing a broad spread of industry interest.

The Bluetooth Special Interest Group is working together with the IEEE, Institute of Electrical and Electronics Engineers, focused on the development of consensus standards for PAN. It is also working on recommended practices and guidelines to handle, among other issues, manufacturer interoperability. The IEEE 802.15.1 standard is an additional resource for those who implement Bluetooth devices.

For more information, you can visit the official Bluetooth Web site at:

http://www.bluetooth.com

or the IEEE Web site at:

http://www.ieee.org

If you want to understand the IBM's perspective regarding the synergies between the Bluetooth technology and the pervasive computing, you can read our white paper at:

http://www.ibm.com/industries/telecom/doc/content/ bin/Bluetooth.pdf.

6.3.2 WLAN (Wireless Local Area Network)

A Wireless LAN is a local area network that uses wireless technologies to connect devices and transmits data. To work properly, a WLAN requires an

antenna connected to the network and a client with a wireless card. The antenna is known as an *access point* and the wireless card is usually a PCMCIA card. The widespread technology is the IEEE 802.11b which uses an unlicensed 2.4GHz frequency and can transfer data at up to 11MB per second. The current speed is dependent on the distance between the access point and the client. The closer the client is to the access point, the higher the speed it will achieve.

The IEEE (Institute of Electrical and Electronics Engineers) is responsible for the IEEE 802.11b standard. There are some other WLAN standards that are starting to have commercial products like IEEE 802.11a, which uses the 5GHz frequency and can transfer data at up to 54MB per second. The IEEE is continuously developing other standards like 802.11g (for WLANS operating in the 2.4 GHz frequency but with a bandwidth of 54 MBps), 802.11i (enhanced security), 802.11h (spectrum and power control management) and 802.11e (quality of service).

A wireless LAN does not require lining up devices for line-of-sight transmission such as IrDA. The wireless access points (base stations) are connected to an Ethernet network or server and transmit a radio frequency over an area of several hundred to a thousand feet. This frequency can penetrate walls and other non-metal barriers. Roaming users can be handed off from one access point to another as in a cellular phone system. Laptops usually use PCMCIA cards to connect to a wireless access point and desktops and servers use plug-in cards.

Wi-Fi Alliance

The Wi-Fi Alliance is an important member of the IEEE 802.11 ecosystem. They are a nonprofit trade organization that promotes the use of standardized 802.11 technologies and certify Wi-Fi product interoperability.

For more information, you can visit the official IEEE Web site at:

`http://www.ieee.org`

or the Wi-Fi Alliance Web site at:

`http://www.weca.net`

6.3.3 WWAN (Wireless Wide Area Network)

Wireless WANs use the cellular networks to provide access. The cellular technology is in constant evolution. Each major advance is often referred to as a *generation* with its acronyms: 1G (first generation), 2G (second generation) and so on.

1G

The First-generation (1G) was the first analogic circuit-switching cellular technology appropriated to mobile voice communication.

2G

The Second-generation (2G) systems represented the digital evolution and were introduced in the 1990s. Some of the technologies implemented included TDMA, CDMA, and GSM. 2G systems were used primarily for voice. They have basic data capabilities but with a slow data transfer speed, between 9.6KBps and 14.4KBps. These protocols support high bit rate voice and limited data communications. They offer auxiliary services such as data, fax and SMS.

2.5G

The 2.5G represents a major step towards the convergence between telecommunication and the Internet. It brings packet-switching technology to data transfer. Some 2.5G systems have been implemented recently, such GPRS and CDMA 1xRTT. Both can be considered a packet-based extension of GSM and CDMA networks, respectively, and provide higher data throughput and always-on connectivity. They also provide new business models such as the pay per packet model.

Under the wireless solution perspective, we should focus on the 2.5G which brings higher packet data transfer speed, allowing the development of a huge spectrum of new applications.

3G

Third-generation (3G) systems are starting to be implemented and will be called IMT-2000 (International Mobile Telecommunications-2000). IMT-2000 is the ITU (International Telecommunication Union) globally coordinated definition of 3G, covering key issues such as frequency spectrum use and technical standards. 3G networks provide higher-speed transmission to support high-quality audio and video, as well as global roaming capability. 3G will support bandwidth-hungry applications such as full-motion video, video-conferencing and full Internet access. For more information, you can visit the ITU Official Web site at:

http://www.itu.int

UMTS

The Universal Mobile Telecommunications System (UMTS) is the European implementation of the 3G wireless phone system supported by the European Telecommunications Standard Institute (ETSI). UMTS, which is part of IMT-2000, offers global roaming and personalized features. UMTS was designed as an evolutionary system for GSM network operators. UMTS uses the W-CDMA

technology. GPRS and EDGE are interim steps that will speed up wireless data for GSM. For more information you can visit the UMTS Forum Web site at:

http://www.umts-forum.org

or the ETSI Web site at:

http://www.etsi.org.

WCDMA

The Wideband Code Division Multiple Access (WCDMA) radio access technology is part of UMTS and is supported by several manufacturers such as Ericsson (Sweden) and Nokia (Finland). This technology will be used mainly in Europe and Japan. The data translation rate is 64 kbps for upstream and 384 kbps for downstream.

CDMA2000

CDMA2000 is a specification developed by the Third Generation Partnership Project 2 (3GPP2). It is an evolution from CDMA technology. The first phase of CDMA2000 is known as CDMA2000 1X and will provide bandwidth of 144 kbps. The second phase is labeled CDMA2000 1xEV and will provide for bandwidth in the 2Mbps range. For more information you can visit the 3GPP Web site at:

http://www.3gpp2.org

6.4 Web application server

Figure 6-2 on page 96 shows the recommended technology model for a Web application server.

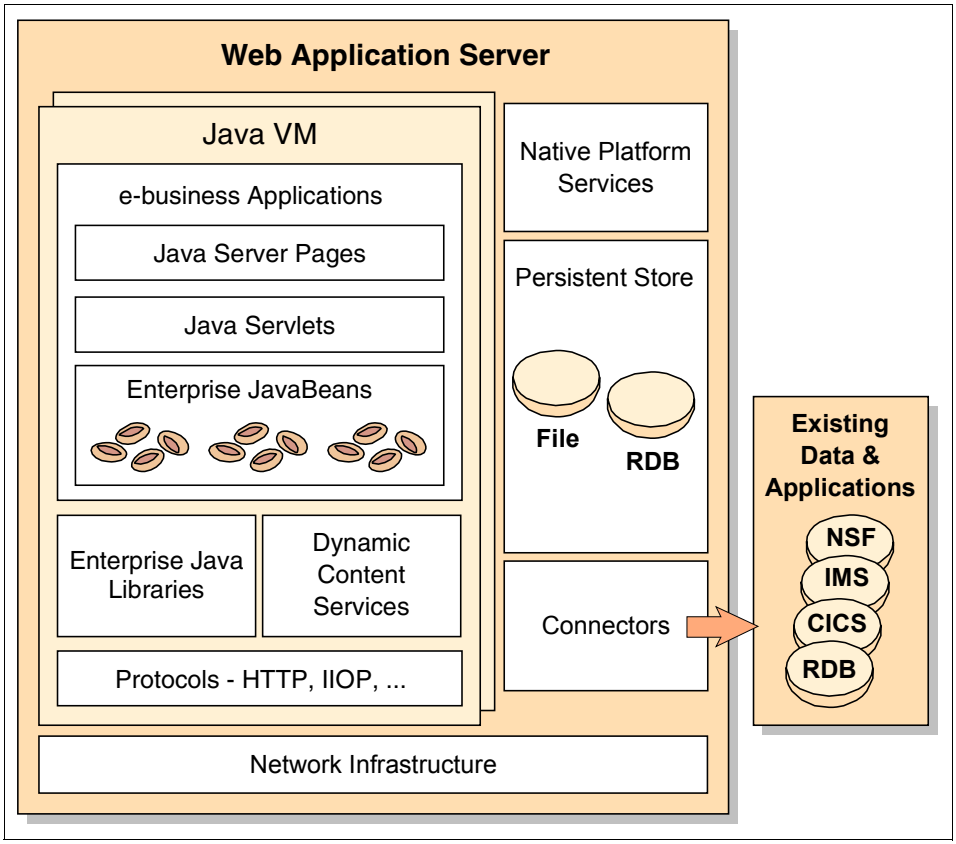


Figure 6-2 Web application server technology model

We assume in this book that you are using a Web application server and server-side Java. While there have been many other models for a Web application server, this is the one that is experiencing widespread industry adoption.

Before looking at the technologies and APIs available in the Web application programming environment, first let's have a word about two fundamental operational components on this node, the HTTP server and the application server. For production applications, they should be chosen for their operational characteristics in areas such as robustness, performance, and availability.

We follow the well-known Model-View-Controller (MVC) design structure so often used in user interfaces. For the Web application programming model, the following applies.

- The Model represents the data of the application, and the business rules and logic that govern the processing of the data. In a J2EE application, the Model is usually represented to the View and the Controller via a set of JavaBeans components.
- The View is a visual representation of the Model. Multiple Views can exist simultaneously for the same model and each View is responsible for making sure that it is presenting the most current data by either subscribing to state change events or by making periodic queries to the Model. With J2EE, the view is generally implemented using JavaServer Pages (JSP).
- The Interaction Controller decouples the visual presentation from the underlying business data and logic by handling user interactions and controlling access to the Model. It processes the incoming HTTP requests and invokes the appropriate business or UI logic. Using J2EE, the Controller is often implemented as a servlet.

6.4.1 Java servlets

Servlets are Java-based software components that can respond to HTTP requests with dynamically generated HTML. Servlets are more efficient than CGI for Web request processing, since they do not create a new process for each request.

Servlets run within a Web container as defined by the J2EE Model and therefore have access to the rich set of Java-based APIs and services. In this model, the HTTP request is invoked by a client such as a Web browser using the servlet URL. Parameters associated with the request are passed into the servlet via the HttpServletRequest, which maintains the data in the form of name/value pairs. Servlets maintain state across multiple requests by accessing the current HttpSession object, which is unique per client and remains available throughout the life of the client session.

Acting as an MVC Controller component, a servlet delegates the requested tasks to beans that coordinate the execution of business logic. The results of the tasks are then forwarded to a View component, such as a JSP to produce formatted output.

One of the attractions of using servlets is that the API is a very accessible one for a Java programmer to master. The specification of the J2EE 1.3 platform requires Servlet API 2.3 for support of packaging and installation of Web applications.

Servlets are a core technology in the Web application programming model. They are the recommended choice for implementing the Interaction Controller classes that handle HTTP requests received from the Web client.

6.4.2 Java portlet

Portlets are reusable components that provide access to Web-based contents, applications, and other resources. Web pages, Web Services, applications, and syndicated content feeds can be accessed through portlets.

Companies can create their own portlets or select portlets from a catalog of third-party portlets. Portlets are intended to be assembled into a larger portal page, with multiple instances of the same portlet displaying different data for each user.

From a user's perspective, a portlet is a window on a portal site that provides a specific service or information, for example, a calendar or news feed. From an application development perspective, portlets are pluggable modules that are designed to run inside a portlet container of a portal server.

From the development perspective, the portlets are components coded against the portlet API. The portlet components are part of the portal application. The portal application is deployed on the portal server.

For more information about portlets and portlets programming, you can refer to IBM Redbook entitled *A Portal composite pattern Using WebSphere Portal V4.1,* SG24-6869.

6.4.3 JavaServer Pages (JSPs)

JSPs were designed to simplify the process of creating Web pages by separating the Web presentation from Web content. In the page construction logic of a Web application, the response sent to the client is often a combination of template data and dynamically generated data. In this situation, it is much easier to work with JSPs than to do everything with servlets. The JSP acts as the View component in the MVC model.

The chief advantage JSPs have over standard Java servlets is that they are closer to the presentation medium. A JavaServer Page is developed as an HTML page. Once compiled, it runs as a servlet. JSPs can contain all the HTML tags that Web authors are familiar with. A JSP may contain fragments of Java code that encapsulate the logic that generates the content for the page. These code fragments may call out to beans to access reusable components and enterprise data.

JSP technology uses XML-like tags and scriptlets written in Java programming language to encapsulate the conditional logic that generates dynamic content for an HTML page. In the runtime environment, JSPs are compiled into servlets before being executed on the Web application. Output is not limited to HTML but also includes WML, XML, cHTML, DHTML, and VoiceXML. The JSP API for J2EE 1.3 is JSP 1.2.

JSPs are the recommended choice for implementing the View that is sent back to the Web client. For those cases where the code required on the page is to be a large percentage of the page, and the HTML minimal, writing a Java servlet will make the Java code much easier to read and therefore maintain.

6.4.4 JavaBeans

JavaBeans are an architecture developed by Sun Microsystems, Inc. describing an API and a set of conventions for reusable, Java-based components. Code written to Sun's JavaBeans architecture is called JavaBeans or just beans. One of the design criteria for the JavaBeans API was support for builder tools that can compose solutions that incorporate beans. Beans may be visual or non-visual.

Beans are recommended for use in conjunction with servlets and JSPs in the following ways:

- As the client interface to the Model layer. An Interaction Controller servlet will use this bean interface.
- As the client interface to other resources. In some cases this may be generated for you by a tool.
- As a component that incorporates a number of property-value pairs for use by other components or classes. For example, the JavaServer Pages specification includes a set of tags for accessing JavaBeans properties.

6.4.5 XML

XML (Extensible Markup Language) and XSL stylesheets can be used on the server side to encode content streams and parse them for different clients, thus enabling you to develop applications for a range of PC browsers and for the emerging pervasive devices. The content is in XML and an XML parser is used to transform it to output streams based on XSL stylesheets that use CSS.

This general capability is known as transcoding and is not limited to XML-based technology. The appropriate design decision here is how much control over the content transforms you need in your application. You will want to consider when it is appropriate to use this dynamic content generation and when there are advantages to having servlets or JSPs specific to certain device types.

XML is also used as a means to specify the content of messages between servers, whether the two servers are within an enterprise or represent a business-to-business connection. The critical factor here is the agreement between parties on the message schema, which is specified as an XML DTD or Schema. An XML parser is used to extract specific content from the message stream. Your design will need to consider whether to use an event-based approach, for which the SAX API is appropriate, or to navigate the tree structure of the document using the DOM API.

IBM's XML4J XML parser was made available through the Apache open source organization under the Xerces name. For open source XML frameworks, see:

http://xml.apache.org/

Defining XML documents

XML documents are defined using DTDs or XML Schemas.

DTDs are a basic XML definition language, inherited from the SGML specification. The DTD specifies what markup tags can be used in the document along with their structure.

DTDs have two major problems:

- Poor data typing: in DTDs, elements can only be specified as EMPTY, ANY, element content, or mixed element-and-text content, and there is no standard way to specify null values for elements.

  Date formats, numbers, or other common data types cannot be specified in the DTD, so an XML document may comply with the DTD but still have data type errors that can only be detected by the application.
- Not defined in XML: DTD uses its own language to define XML syntax, and it is not compliant with the XML specification. This makes it difficult to manipulate a DTD.

To solve these problems, the World Wide Web Consortium (W3C) defined a new standard to define XML documents called XML Schema. XML Schema provides the following advantages over DTDs:

- Strong typing for elements and attributes
- Standardized way to represent null values for elements
- Key mechanism that is directly analogous to relational database foreign keys
- Defined as XML documents, making them programmatically accessible

Even though the XML Schema is a more powerful technology to define XML documents, it is also a lot harder to work with, so DTDs are still widely used to define XML documents. Additionally, simple, not hard-typified documents can be easily defined using DTDs with similar results to using XML Schema.

Whether to use one or the other will depend on the complexity of the messages and the validation requirements of the application. Actually, in many cases, both (a DTD and a XML Schema) are provided, so they can be used by the application depending on its requirements.

Note: We have to remember that the validation process of an XML document using XML Schemas is an *expensive process*. Validation should be performed only when it is necessary.

XSLT

Extensible Stylesheet Language Transformations (XSLT) is a W3C specification for transforming XML documents into other XML documents. The XSLT is built on top of the Extensible Stylesheet Language (XSL) which, like CSS2 seen in 6.1.4, "CSS" on page 79, is a stylesheet language for XML. Unlike CSS2, XSL is also a transformation language.

A transformation expressed in the XSLT language defines a set of rules for transforming a source tree to a result tree; it is expressed in the form of a stylesheet.

An XSLT processor is used for transforming a source document to a result document. There are currently a number of XSLT processors available on the market. DataPower has introduced an XSL just-in-time (JIT) compiler, which speeds up the time taken for the XSL transformation.

The XSLT processor has a performance overhead, so online processing of larger documents can be slow.

XML security

XML security is an important issue, particularly where XML is being used by organizations to interchange data across the Internet. Several new XML security specifications are working their way through three standards bodies - the W3C (World Wide Web Consortium), IETF (Internet Engineering Task Force), and OASIS (Organization for the Advancement of Structured Information Standards). We highlight a few of them here:

- XML Signature Syntax and Processing is a specification for digitally signing electronic documents using XML syntax. According to the W3C, "*XML Signatures provide integrity, message authentication, and/or signer*

authentication services for data of any type, whether located within the XML that includes the signature or elsewhere."

A key feature of the protocol is the ability to sign parts of an XML document rather than the document in its entirety. This is necessary because an XML document might contain elements that will change as the document is passed along or various elements that will be signed by different parties.

WebSphere Studio provides you with the ability to create (using a wizard) and verify XML digital signatures.

- XML encryption will allow encryption of digital content, such as Graphical Interchange Format (GIF) images or XML fragments. XML encryption allows the parts of an XML document to be encrypted while leaving other parts open, the encryption of the XML itself, or the super-encryption of data (that is, encrypting an XML document when some elements have already been encrypted).
- XKMS (XML Key Management Specification) establishes a standard for XML-based applications to use Public Key Infrastructure (PKI) when handling digitally signed or encrypted XML documents. An XML signature addresses message and user integrity, but not issues of trust that key cryptography deals with.
- SAML (Security Assertion Markup Language) is the first industry standard for secure e-commerce transactions using XML. It aims to standardize the exchange of user identities and authorizations by defining how this information is to be presented in XML documents, regardless of the underlying security systems in place.

For further discussion, see the Sun ONE article *Riddle Me This: Is Your XML Data Safe?* by Brett Mendel, which can be found at:

`http://dcb.sun.com/practices/websecurity/overviews/xmldata.jsp`

Advantages of XML

There are many advantages to XML in a broad range of areas. Some of the factors that influenced the wide acceptance of XML are as follows:

- Acceptability of use for data transfer

  XML is a standard way of putting information in a format that can be processed and exchanged across different hardware devices, operating systems, software applications and the Web.

- Uniformity and conformity

  XML gives you an common format that could be developed upon and is accepted industry-wide.

- Simplicity and openness

  Information coded in XML is human readable.

- Separation of data and display

  The representation of the data is separated from the presentation and formatting of the data for display in a browser or other device.

- Industry acceptance

  XML has been accepted widely by the information technology and computing industry. Numerous tools and utilities are available, along with new products for parsing and transforming XML data to other data, or for display.

Disadvantages of XML

Some XML issues to consider are as follows:

- Complexity

  While XML tags can allow software to recognize meaningful content within documents, this is only useful to the extent that the software reading the document knows what the tagged content means in human terms, and knows what to do with it.

- Standardization

  When multiple applications use XML to communicate with each other, they need to agree on the tag names they are using. While industry-specific standard tag definitions often do exist, you can still declare your own non-standard tags.

- Large size

  XML documents tend to be larger in size than other forms of data representation.

6.4.6 Enterprise JavaBeans

“Enterprise JavaBeans” is Sun's trademarked term for its EJB architecture (or “component model”). When writing to the EJB specification, you are developing “enterprise beans” (or, if you prefer, “EJBs”).

Enterprise beans are distinguished from JavaBeans in that they are designed to be installed on a server, and accessed remotely by a client. The EJB framework provides a standard for server-side components with transactional characteristics.

The EJB framework specifies clearly the responsibilities of the EJB developer and the EJB container provider. The intent is that the “plumbing” required to implement transactions or database access can be implemented by the EJB container. The EJB developer specifies the required transactional and security characteristics of an EJB in a deployment descriptor (this is sometimes referred to as *declarative programming*). In a separate step, the EJB is then deployed to the EJB container provided by the application server vendor of your choice.

There are three types of Enterprise JavaBeans:

- Session beans
- Entity beans
- Message-driven beans

A typical session bean has the following characteristics:

- Executes on behalf of a single client.
- Can be transactional.
- Can update data in an underlying database.
- Is relatively short lived.
- Is destroyed when the EJB server is stopped. The client has to establish a new session bean to continue computation.
- Does not represent persistent data that should be stored in a database.
- Provides a scalable runtime environment to execute a large number of session beans concurrently.

A typical entity bean has the following characteristics:

- Represents data in a database.
- Can be transactional.
- Has shared access by multiple users.
- Can be long lived (lives as long as the data in the database).
- Survives restarts of the EJB server. A restart is transparent to the client.
- Provides a scalable runtime environment for a large number of concurrently active entity objects.

A typical Message-Driven Bean has the following characteristics:

- Consumes messages sent to a specific queue.
- Is asynchronously invoked.
- Is stateless.

- Can be transaction aware.
- May update shared data in an underlying client message.
- Executes upon receipt of a single client message.
- Has no component or home interface.
- Is removed when the EJB container crashes. The container has to re-establish a new message-driven object to continue computation.

Typically, an entity bean is used for information that has to survive system restarts. In session beans, on the other hand, the data is transient and does not survive when the client's browser is closed. For example, a shopping cart containing information that may be discarded uses a session bean, and an invoice issued after the purchase of the items is an entity bean.

An important design choice when implementing entity beans is whether to use Bean Managed Persistence (BMP), in which case you must code the JDBC logic, or Container Managed Persistence (CMP), where the database access logic is handled by the EJB container.

The business logic of a Web application often accesses data in a database. EJB entity beans are a convenient way to wrap the relational database layer in an object layer, hiding the complexity of database access. Because a single business task may involve accessing several tables in a database, modeling rows in those tables with entity beans makes it easier for your application logic to manipulate the data.

An important change to the specification in EJB 2.0 is the addition of a new enterprise bean type, the message-driven bean (MDB). The message-driven bean is designed specifically to handle incoming JMS messages. The EJB container uses message properties and bean deployment descriptor to select the bean to invoke when a message arrives, so your application logic only needs to process the message contents.

The J2EE 1.3 platform requires support for EJB 2.0. As a tool provider, WebSphere Application Server V5.0 supports J2EE 1.3 and therefore supports EJB 2.0. EJBs are packaged into EJB modules (JAR files) and then combined with Web modules (WAR files) to form an enterprise application (EAR file). EJB deployment requires generating EJB deployment code specific to the target application server.

6.4.7 Additional enterprise Java APIs

The J2EE specification defines a set of related APIs that work together. Here are the remainder not discussed so far:

- JNDI: Java Naming and Directory Interface. This package provides a common API to a directory service independent of any directory access protocol. This allows for easy migration to new directory services. Through this interface, component providers can store and retrieve Java object instances by name. Service provider implementations include those for JDBC data sources, LDAP directories, RMI and CORBA object registries. Sample uses of JNDI include:
  - Accessing a user profile from an LDAP directory
  - Locating and accessing an EJB home
  - Locating a driver-specific data source
- RMI-IIOP: Remote Method Invocation (RMI) and RMI over IIOP are part of the EJB specification as the access method for clients to access EJB services. From the component provider point of view, these calls are local. The EJB container takes care of calling the remote methods and receiving the response. To use this API, component providers create an IDL description of the EJB interface and then compile it to generate the client-side and server-side stubs. The stubs connect the object implementations with the Object Request Broker (ORB). ORBs communicate with each other through the Internet Inter-ORB Protocol (IIOP). RMI can also be used to implement limited-function Java servers.
- JTA: Java Transaction API. This Java API for working with transaction services is based on the XA standard. With the availability of EJB servers, you are less likely to use this API directly.
- JAF: JavaBeans Activation Framework. This API is not intended for typical application use, but it is required by the JavaMail API.
- JavaMail: This is a set of classes for supporting e-mail. Functionally, it provides APIs for reading, sending, and composing Internet mail. This API models a mail delivery system and requires the SMTP for sending mail and POP3 or IMAP for receiving mail. Special data wrapper classes are provided to view and edit data in the mail content. Support for MIME data is delegated to the JAF-aware beans.
- JAXP: API for parsing and transforming XML documents.
- JAAS: Java Authentication and Authorization Service.

6.5 Transcoding technology

Transcoding Technology is delivered as an embedded transcoding engine in the WebSphere Portal Server as part of the WebSphere Everyplace Access product. It takes care of transcoding the content for different devices using the original HTML source. The transcoding technology is transparent for application designers and developers. It is configured in the portal server and for each portlet that wishes to use content transcoding.

The Transcoding technology that is shipped as part of WebSphere Everyplace Access is an extracted part of the Transcoding Publisher product.

7

Application design

This chapter provides design guidelines for solution designers and application developers. It addresses the design stage of the solution development process.

The objective of this chapter is to highlight those key design issues that are specific to a pervasive and portal solution.

The chapter starts with basic e-business and Self-Service application design, then goes into details about the sample application that was developed for this book.

The majority of the chapter goes into details about certain design guidelines, such as:

- Model-View-Controller design
- Object-oriented design
- Portal solution design guidelines
- Mobile application design guidelines

© Copyright IBM Corp. 2003. All rights reserved.

7.1 e-business application design

e-business application design is changing the way that the products are created. The designer needs to take care of the functional and non-functional requirements and also be clear the business issues in considering the solution implementation.

In the traditional way to develop applications, the designer should take care of the business requirements but basically the interface does not have variations, considering that the basic capabilities are the same.

In the new e-business environment, the application must be available for many devices and the users have different ways to access the same information. These considerations make the design more ample and the designer must take into further consideration the business requirements and the non-functional requirements such as availability, performance (this depends on the device the user has been using), and other facilities that the e-business application needs.

There are many differences between the traditional applications and e-business applications; some of them are listed below:

- The domain of e-business applications users is typically much more diverse than that of the user group for traditional applications. Users can be known to the systems or anonymous, and can come from inside or outside the enterprise. Web applications must be developed to meet the varied needs of those end users.
- The diversity in user types and exposure to the outside world significantly increases security risks to internal systems. e-business applications security infrastructure and applications must be designed accordingly, and will likely require dedicated security components.
- The user experience, look and feel, and features of the site need to be enhanced frequently to leverage emerging technologies, attract and retain site users.
- Changes and enhancements must be delivered more quickly and flawlessly than ever to avoid damage to the corporation's reputation and lost customers.
- Content for e-business applications can come from many sources: technical and non-technical, from inside or outside the enterprise. Such diversity in location, skill levels, and access creates significant challenges for content creation and management.
- It is hard to predict the runtime load of e-business applications. Based on the many variables, system load can increase dramatically over time. e-business applications and infrastructure must be designed with rapid scalability and availability in mind.

- Connection device, bandwidth, and reliability have an enormous impact on the success of a site. Access devices can range from small-footprint mobile devices such as wireless phones to PCs with near-LAN speed bandwidth. Infrastructure and interfaces for e-business applications must take this into account.

To meet these challenges, flexibility is a critical aspect in the design of Web applications. The following sections provide some suggestions for meeting the diverse challenges of e-business, especially in regard to a Pervasive Portal solution.

7.2 Self-Service application guidelines

The design guidelines outlined here primarily focus on Self-Service Web applications. Before exploring these guidelines, you should be familiar with the Self-Service runtime patterns and the various technology options available for implementing a Self-Service Web application, including server-centric Java-based technologies such as servlets, JSPs, JavaBeans, and EJBs for such implementations. The self-service basic concepts can be found in more detail in Chapter 3, “Selecting the Application patterns” on page 27.

Clients are responsible for accepting and validating the user input, communicating the user inputs to the Web application server, and presenting the results received from the Web browser, PDAs or WAP (for example cell phones).

Clients may use HTTP, IIOP, WAP, VoIP (Voice over IP), TCP/IP, or other Internet standard protocols to communicate with the Web application server. These clients can be broadly classified into the following categories:

- HTML clients - These use HTTP protocol to communicate with the Web application server. These clients display HTML Web pages. In addition, they are capable of processing client-side JavaScript for enhancing navigation to perform simple input validation and to handle simple errors. Furthermore, the HTML clients can display small Java applets to enhance the GUI.
- Application clients - These are primarily large Java applets or Java applications. These clients provide rich graphical user interfaces compared to HTML clients. They may communicate with the Web application server over a number of protocols including HTTP, IIOP, MQ, etc. Application clients communicate with the Web application server primarily to receive data rather than preformatted HTML pages. These clients use the data received to format and render the user interface. All of the user interface processing is performed on the client side. In addition, under this model, some parts of the business logic can also be processed on the client side.

- Application clients that use Java Application for PDA using J2ME (Java 2 Micro Edition) -This application can be online or offline and can use the synchronization techniques using SyncML to update the data. In this case, the PDA should have a Java Runtime Environment (CVM, KVM, J9) and the appropriate configuration (CLDC - Connected Limited Device Configuration or CDC - Connected Device Configuration) and profile (MIDP, PDAP, etc.). Also in this case, the application will have some capabilities for processing.
- WML clients - These use WAP protocols to communicate with the Web application server. These clients display WML in the client device. The communication between the gateway and Web application can be HTML using transcoding technologies to translate from HTML to WML; the application throws the data into XML format and certain software links it with a Stylesheet; WML can be directly provided by the application.

7.3 Sample scenario

The sample application is a Pervasive Portal for a computer maintenance company. It can be used by its customers, technicians and help desk attendants. This Pervasive Portal can be accessed through a regular Internet browser like Netscape or Internet Explorer, and also through a cell phone with a WAP browser or a PDA with PalmOS. It can be expanded to other platforms supported by the tools used, such as WebSphere Everyplace Access.

The sample application implements a help desk portal with three perspectives: customer, technician and customer service attendant.

The customer perspective shows the “problem entry” and the “problem follow-up” applications. These applications are available through a Web browser, on PDAs and for WAP phones. In the technician perspective, it also has the “close defect” and “invoice customer” applications and access to the knowledge system. The call center attendants can have the same perspective as the customer but can search the knowledge database to find help for the customer at the first call.

This section gives a detailed description of the sample scenario we have used for this book. This scenario is based on a sample application that handles service requests from the customers, manages the service fulfilment, and provides help for the agents in the field.

There are three different types of users for the application:

- The Client can submit a request to the system. There is an option to request help using a computer with a simple Web browser. Since the application handles service requests for computer parts, the computer is probably going

to be broken, so we have to provide another channel for the customer: a phone contact.

- The Customer Service will help the customer submit the request, in case the customer has a broken system or is not willing to use a Web browser on a computer.
- The Technicians are the technical specialists who can fix the problem at the customer site. They need to know about the problem, the details of the problem, the address where the problem has to be solved. They have to be able to place orders remotely in case the service requires new parts or replacement. The technician needs to have access to a knowledge or supporting database to solve unusual situations and problems.

  With today's technology, we can arm the technician with several different accessories on the field, for example a laptop, PDA, or intelligent wireless phone. These instruments all have some sort of network connection to the Internet, either wireless or wireline connection, permanent (online most of the time) or temporary (offline most of the time).

7.3.1 Business flow

The business flow explains how the solution works in an everyday situation, without too many technical details. There are three different flows, according to the roles identified for the business.

1. Customer's perspective
2. Customer representative's perspective
3. Technician's perspective

Customer flow

1. The customer runs into a hardware problem with the machine in the office. The hardware is supported by the company from our sample scenario.
2. The user goes to the company's Web site to submit a service request.
3. If the user is not registered yet, then he/she can register via the registration page and process.
4. Once the user is registered or logged in, a service request needs to be submitted. The request is a simple Web-based form; it needs to be filled out and sent to the system.
5. The user gets a reply from the system that the request was successfully processed.
6. Based on the severity of the problem, the technician will visit the location, sooner or later, to fix the problem.

Customer representative flow

1. The customer calls the customer representative to get help and schedule a service for the defective hardware.
2. The customer representative uses an intranet version of the same application that the customer uses to submit a new service request.
3. The request is a simple Web-based form; it needs to be filled out then sent to the system.
4. The customer representative gets a reply from the system that the request was successfully processed. The customer representative can tell the customer about the details of the request or can send a notification to the customer.

Technician flow

1. The technician receives the service request. If the request has a high priority, the technician gets a notification on a pervasive device. Otherwise, the technician can get the details by synchronizing a PDA, accessing the company's Web site or accessing the Pervasive Portal application with a smartphone. The technician receives a to-do list with the service request items.
2. The technician goes to the customer to repair the defective hardware.
3. The technician can access the company's knowledge database to find further information about the defect discovered at the customer site. It can be accessed with a browser on a laptop or a PDA, with a smartphone or using an interactive voice response system.
4. If it is necessary to order parts to repair the hardware, the technician must fill out the order request form using a browser on a laptop, a PDA or a smartphone.
5. The technician can also close the defect and put it in the closed state. In this case, the defect will disappear from the to-do list.

7.3.2 Component diagram

Figure 7-1 on page 115 depicts the components in the sample application.

Note: The Voice Server was not implemented in this book.

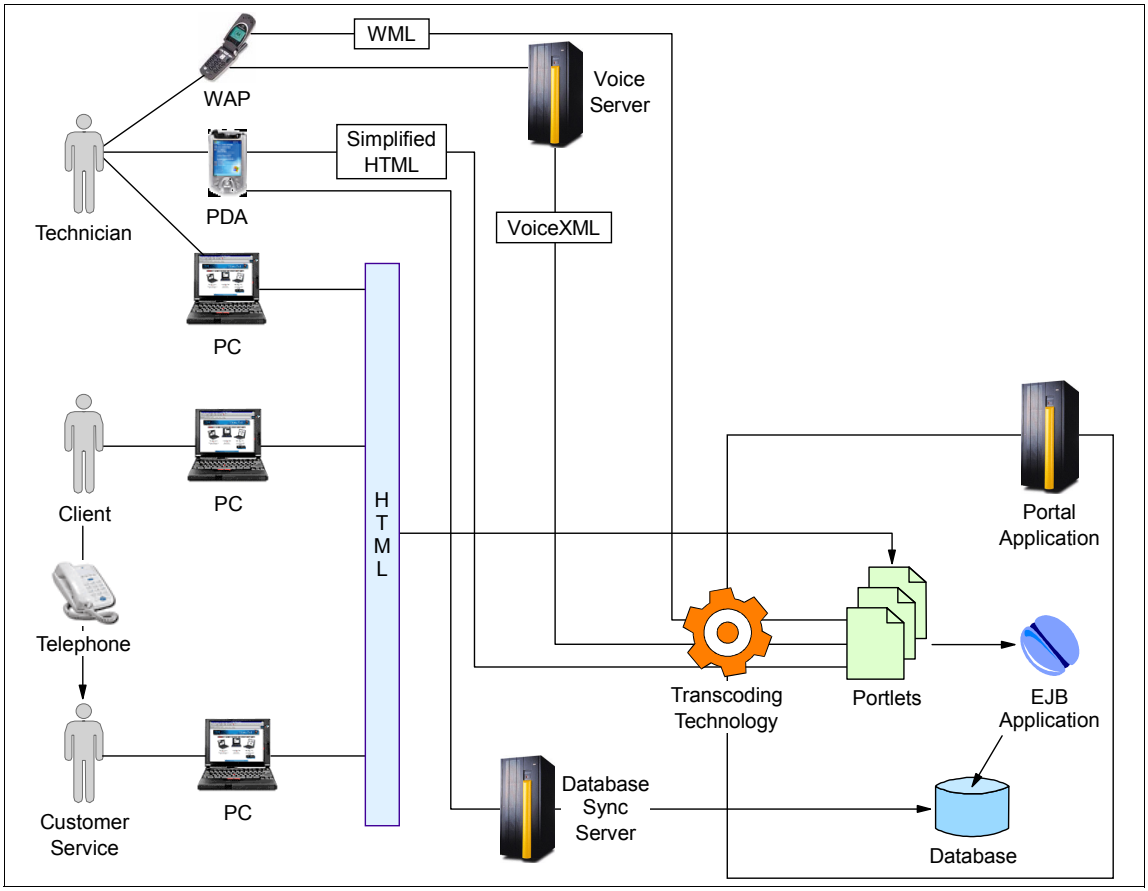


Figure 7-1 Component diagram

The client and the customer service users are using a desktop machine with a Web browser to access the portal application in order to submit a service request.

The interesting part is when the technician comes into the picture. There are different scenarios for this user, depending on the field device.

1. Portable computer (laptop) with remote network connection. This option is quite simple; the technician needs a remote connection to the network to access the application using a Web browser.

2. Smaller, more portable PDA for field agents. There are options again using this device:

   a. If the PDA has an online network connection and is powerful enough, then it can run a Web browser that is capable of handling simplified Web content. This option is very similar to the first one.

   b. The other option is to use the synchronization capabilities of the PDA and download (synchronize) the data between the server and the PDA every once in a while, when the technician has access to a network, to get the service request and the customer information. The synchronization can happen at the base (service provider station) a couple of times a day, or at home in the morning and evening.

      The knowledge management supporting database can be too “heavy” for a PDA to store everything, so the technician needs an alternative way to access that data. In this case, a simple telephone application (VoiceXML) can help out the technician to access the necessary information.

3. An intelligent wireless phone (for example, a WAP phone) has limited online browsing capabilities. The technician can access the most essential information in order to get the service request and the customer information.

   The phone device is definitely not capable of storing or showing support information for the service. The knowledge database in this case can be accessed through a voice (VoiceXML) application.

7.3.3 Use case diagram

We have two actors in the system; one is the client that can register one defect. The client need to log in to the Portal because only authenticated users can log in to the system.

You can create new users with the self-register option in the Portal. The user will be saved in the LDAP Server (IBM SecureWay Directory).

The other actor is the technician. This user cannot register using the application; technicians are added to the system internally. There is a group created in the user registry (LDAP) and all technicians can join the group.

Important: In the application database, there is one table that states which technician ID is responsible for receiving the defect to work on. If you would like to create a new technician ID, you should create the categories and put this ID in a RESPONSIBLE_TECHNICIANID field in the CATEGORY table. The system uses this table to redirect the defect.

The technician can get information from the knowledge database; this database works as an intellectual capital.

Another function is checking the status of a reported defect. The technician can see who the client is and what the problem is. When the problem is resolved, the technician can close the defect and submit an invoice to the client.

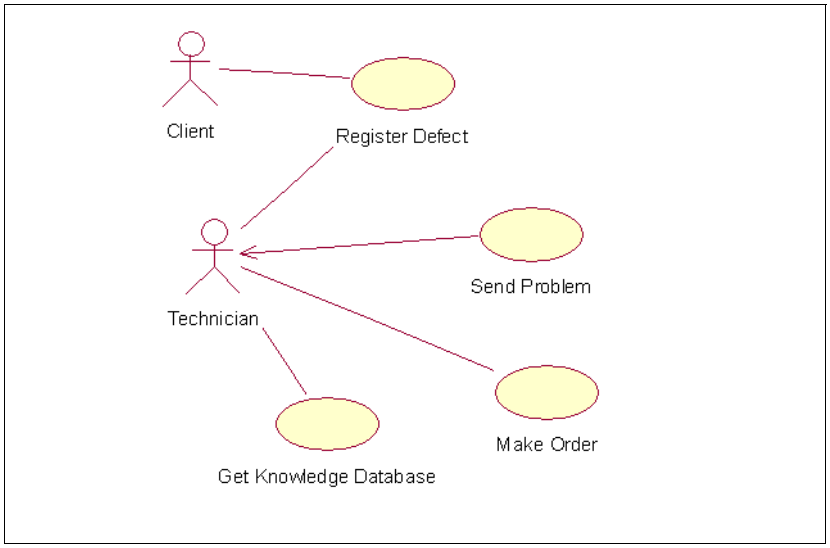


Figure 7-2 Help desk use case diagram

7.3.4 Class diagram

This next diagram helps to understand the classes created for the sample scenario. It is not the goal of this book to explain in detail the decisions involved in the business structure.

Each entity bean maps one table in the database. These beans do not contain business process logic; they only model the data.

The value that entity beans provide is an object-oriented in-memory view of data in an underlying data store. The traditional method for applications in dealing with data is to work with relational tables in a database, reading and writing the data as needed. Entity beans, on the other hand, are object representations of this underlying data. You can treat data in a relational store as real objects. You can read an entire set of data out of a database at once into an in-memory entity bean component. You can then manipulate this entity bean in memory by calling methods on it.

Because EJBs model permanent data, entity beans are long lasting. They can survive critical failures, such as application servers crashing, because entity beans are just representations of data in a permanent, fault-tolerant underlying storage.

If a machine crashes, the entity bean can be reconstructed in-memory again by simply reading the data back in from the permanent database. Because the database survives crashes, the components that represent them do as well.

Figure 7-3 shows the relationship between the defect, order, category and knowledge management EJBs.

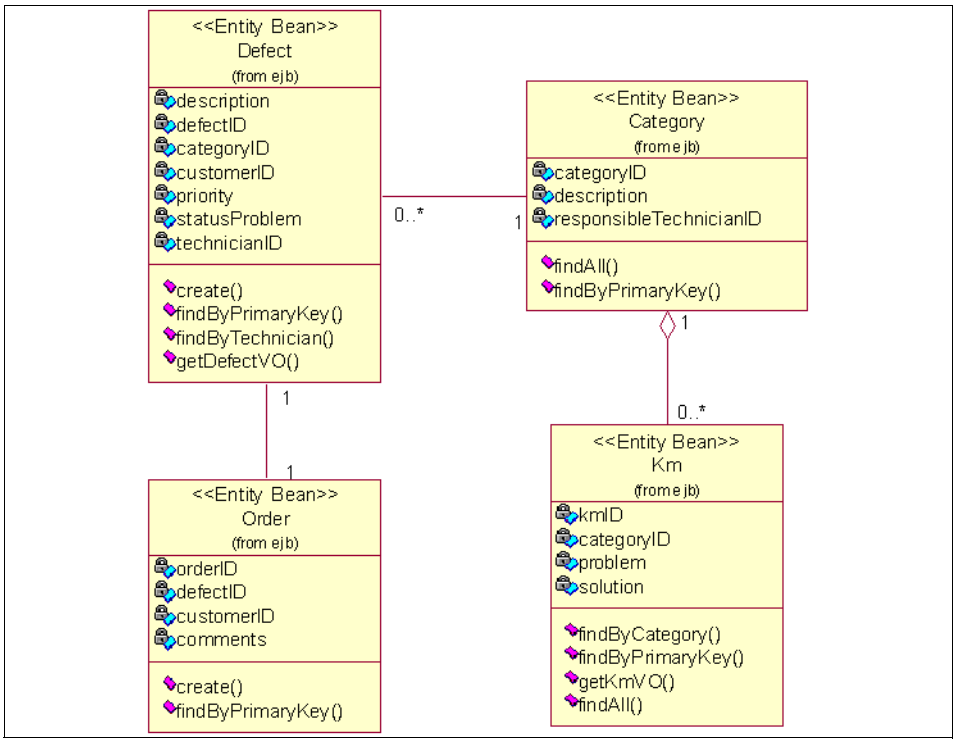


Figure 7-3 Class diagram - persistent class

In our sample, the stateless session bean is working in a session facade pattern; it has to interact with entity beans and the portlets have to interact only with the stateless session bean.

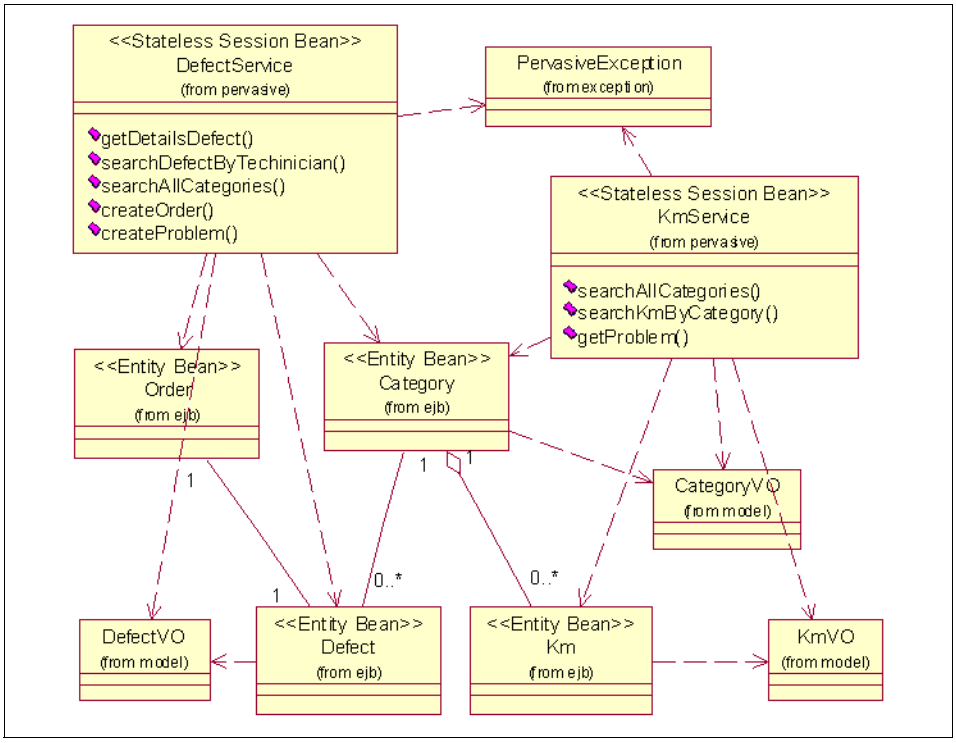


Figure 7-4 Class diagram - business class interaction with the entity bean

7.3.5 Sequence diagram

In Figure 7-5 on page 120, you can see the interaction between the portlets and the business components.

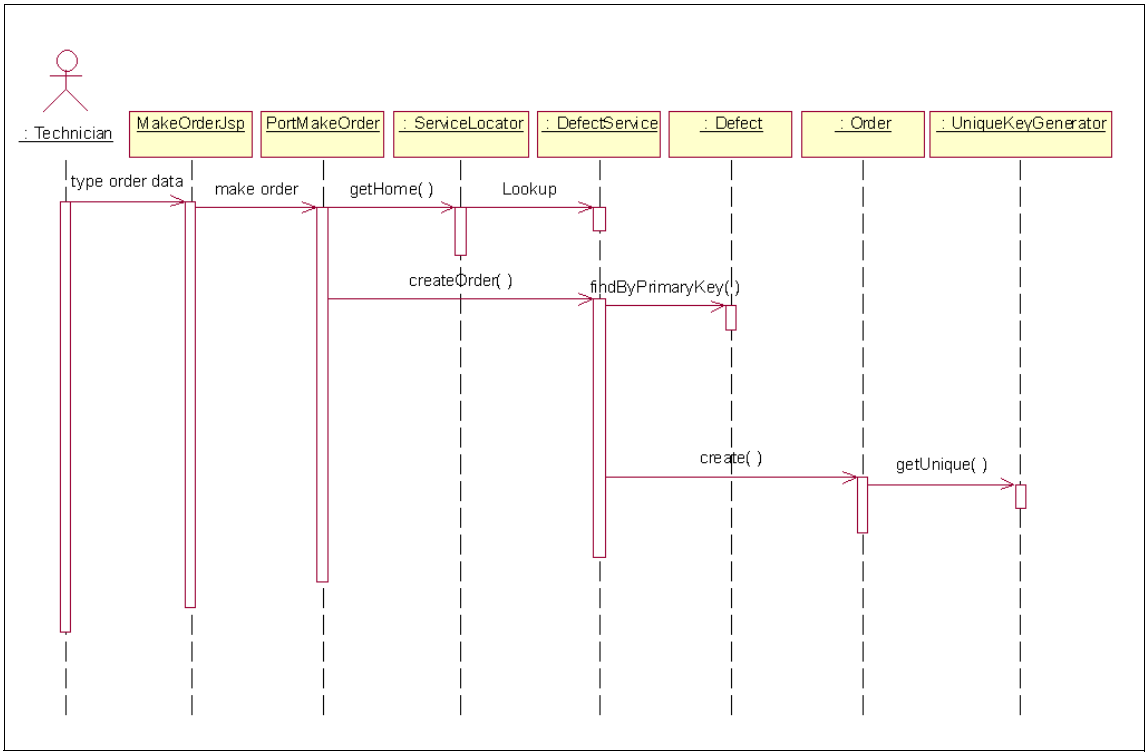


Figure 7-5 Sequence diagram - make order

In the figure above, the portlet is *PortMakeOrder*. The business component is an EJB Stateless Session Bean. Its name is *DefectService*. The portlet has no access to the persistent component (entity bean) because the portlet knows only the business methods.

When the business method needs to send the results to the portlet, the portlet calls a method on the session bean to create a Value Object component. If you want to know more about this pattern, refer to “Data communication between the portlets and business tier” on page 135.

7.4 Application structure

In this section, we explain the Design patterns to develop good object-oriented projects. The topics cover theory and practical samples. We do not discuss the samples in detail; if you need more information, refer to Chapter 8, “Application development” on page 157.

7.4.1 Device-specific content

Before we get into the Model-View-Controller (MVC) design concept, we need to take a look at the options and how the device-specific content can be processed.

There are two fundamental methods to providing device-specific content:

- Generating the content in the code in a programmatic way. It is a direct solution using either different presentation pages or stylesheets.
- Using an intermediary component, like a transcoding engine, that takes care of the content transformation. It is a configurable, intelligent engine that is capable of recognizing the device and transform the content as required on the fly.

For more information about the options described above, refer to the redbook *Mobile Applications with IBM WebSphere Everyplace Access Design and Development*, SG24-6259.

7.4.2 Model View Controller (MVC)

In the MVC paradigm, the user input, the modeling of the external components, and the visual feedback to the user are explicitly separated and handled by three types of object, each specialized for its task.

The View manages the graphical and/or textual output to the portion of the “bitmapped” display that is available for the application.

The Controller interprets the user inputs, commanding the model and/or the view to change as appropriate.

Finally, the Model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the View), and responds to instructions to change state (usually from the Controller).

The MVC behavior is then inherited, added to, and modified as necessary to provide a flexible and powerful system.

To use the MVC paradigm effectively, you must understand the division of labor within the MVC tier. You also must understand how the three parts of the tier communicate with each other and with other active views and controllers; the sharing of a single mouse, keyboard and display screen among several applications demands communication and cooperation. To make the best use of the MVC paradigm, you also need to learn about the available subclasses of *View* and *Controller* which provide ready-made starting points for your applications.

In a general application, the input can be a specific hardware device or some specific data that identifies a client. For example, in the mobile realm, a piece of hardware can be a wireless phone or a PDA. These are different hardware devices that need differently supported presentations. The same output has a different presentation on a wireless phone and on a PDA. The important fact to remember is that an application should be client-independent.

An application that is only designed to support a specific type of client can be considered a “bad” application. For example, consider information that is requested by a mobile phone and by a PDA. The same information is being requested by the two different hardware devices. Why should applications be differently designed and implemented if they use the same business logic? Having two applications serving the same information results in duplicate components, duplicate maintenance and duplicate costs. How to avoid this problem will be explained later.

The best way to learn MVC is to understand the concept very well and practice in architecture design. In this project, the portlet development uses the MVC pattern.

It will be presented one component example that was applied the MVC pattern.

The first step is to design the components and put them in the right tier. To do this, some name convention were created. Take a look at the sequence diagram shown in Figure 7-6.

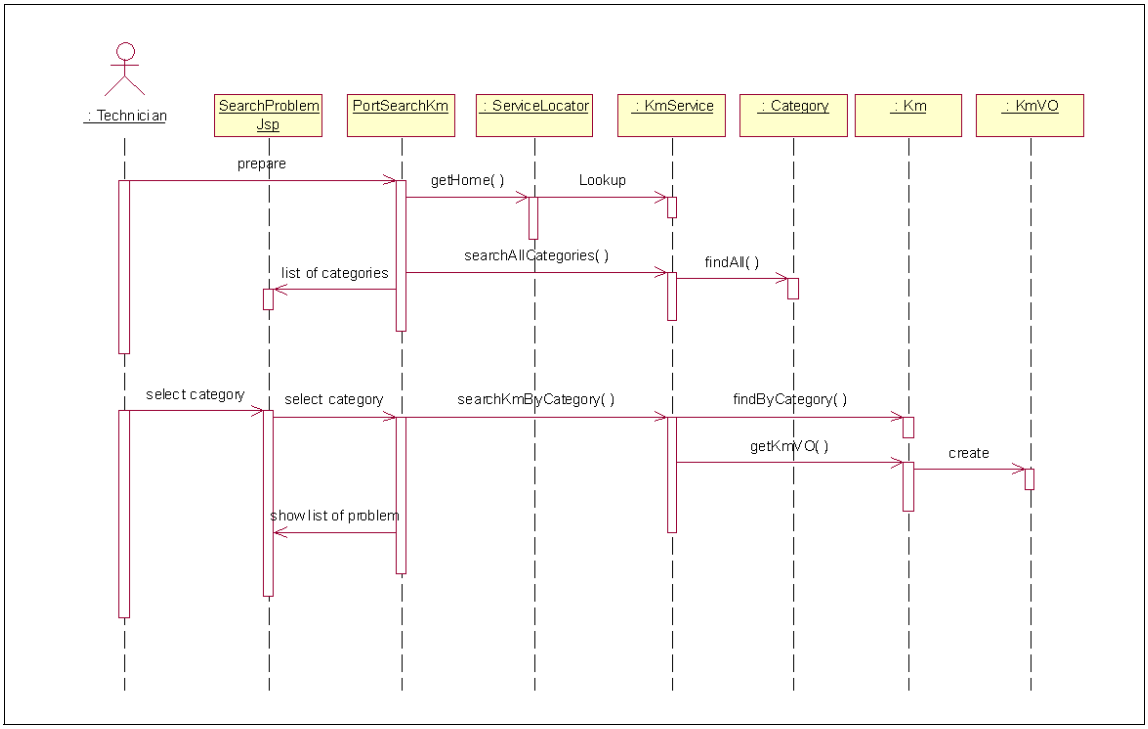


Figure 7-6 Sequence diagram with MVC design

SearchProblem.jsp is one View component; this is a JavaServer Page and is responsible for viewing the data. The Controller in this diagram is the *PortSearchKm*, which is one portlet. This class has no business process and its responsibility is to interact with the Service Controller Singleton to get the references from EJB and call the business methods.

KmService is a stateless session bean that has the business methods. This bean works as a Session Facade and it is recommended that in a pervasive project, the portlets have no access to entity beans. Category and KM class are entity beans and they are responsible for interacting with the database records. There are other classes and applied patterns in this diagram. Refer to 7.4.4, “Applying the Design patterns” on page 135 for more information about the patterns that were applied.

In order to understand the physical implementation of this diagram, part of the code for each tier is shown.

The first sample code is a JSP. This code is responsible for viewing the data and sending it to the user.

Example 7-1 View tier - JSP

```
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ page import="com.ibm.itso.pervasive.model.CategoryVO" %>
<%@ page import="com.ibm.itso.pervasive.model.KmVO" %>
<%@ page session="false" %>
<jsp:useBean id="categoryList" class="java.util.ArrayList" scope="request"/>
<jsp:useBean id="solutionList" class="java.util.ArrayList" scope="request"/>
<form name="<portletAPI:encodeNamespace value='frmSearchProblemJsp'/>"
method="post" action="<portletAPI:createURI><portletAPI:URIAction
name='select'/></portletAPI:createURI>">
<TABLE border="0">
<TR>
<TD>Select Category Of Problem</TD>
<TD><SELECT name="selCategoryID">
<%
for(int x=0; x < categoryList.size(); x++) {
CategoryVO category = (CategoryVO)categoryList.get(x);
out.println("<OPTION value=");
out.println(">" + category.getDescription() + "</OPTION>");
}
%>
</SELECT></TD>
</TR>
</TABLE>
<INPUT type="submit" name="ok" value="OK">
<HR>
<TABLE border="1">
<TR><TD>Problem</TD>
<TD>Solution</TD>
</TR>
<% for(int y=0; y < solutionList.size(); y++) {
KmVO solution = (KmVO)solutionList.get(y);
%>
<TR>
<TD><%= solution.getProblem() %></TD>
<TD><%= solution.getSolution() %></TD>
</TR>
<% } %>
</TABLE>
```

In this code, you can see the data from other tiers and also HTML. This mix of Java and HTML code makes it unstructured and difficult to maintain. This code does not contain any transaction or business process and represents the View in the MVC pattern.

The next class is the Controller. This tier has no business process because it is responsible for calling the business methods. This code is more structured than the JSP code.

Example 7-2 Controller tier - Java class

```
import com.ibm.itso.pervasive.ejb.*;
import com.ibm.itso.pervasive.exception.*;
import com.ibm.itso.pervasive.model.*;
import com.ibm.itso.pervasive.foundation.*;
public class PortSearchKm extends AbstractPortlet implements ActionListener {
   //* Constants with pages to redirect
   private final static String ERROR_PAGE = "GeneralErrorJsp.jsp";
   private final static String FORM_PAGE = "SearchProblemJsp.jsp";
   public void doView(PortletRequest req, PortletResponse res) throws
PortletException, IOException {
      ArrayList listCategories = new ArrayList();
      ArrayList listSolution = new ArrayList();
      //* get all categories
      try {
         KmServiceHome kmServiceHome =
(KmServiceHome)ServiceLocator.getInstance().getHome("ejb/KmService");
         KmService kmService = kmServiceHome.create();
         listCategories = kmService.searchAllCategories();
         req.setAttribute("categoryList", listCategories);
      } catch (Exception e) {
         e.printStackTrace(System.out);
         req.setAttribute("errorMessage", e.getMessage());
          getPortletConfig().getContext().include(ERROR_PAGE, req, res);
      }
      if (req.getAttribute("errorMessage") == null)
         getConfig().getContext().include(FORM_PAGE, req, res);
      else
         getConfig().getContext().include(ERROR_PAGE, req, res);
   }

}
```

Example 7-2 shows a Controller in the MVC pattern; it is responsible for handling the requests.

Example 7-3 on page 126 shows a Model class which contains the business logic. The code is well-structured and shows the method *createOrder*.

Example 7-3 Model Tier - Java Class

```
import com.ibm.itso.pervasive.model.*;
import com.ibm.itso.pervasive.exception.*;
import com.ibm.itso.pervasive.foundation.*;
public class DefectServiceBean implements javax.ejb.SessionBean {
   public void createOrder(Long defectID, String comments) throws
PervasiveException {
      try {
         OrderHome orderHome =
(OrderHome)ServiceLocator.getInstance().getHome("ejb/Order");
         DefectHome defectHome =
(DefectHome)ServiceLocator.getInstance().getHome("ejb/Defect");
         DefectKey defectKey = new DefectKey(defectID);
         Defect defect = defectHome.findByPrimaryKey(defectKey);
         DefectVO defectVO = defect.getDefectVO();
         //* create an service order
         orderHome.create(defectVO.getCustomerID(), defectVO.getDefectID(),
comments);
         //* change the status of defect
         defect.setStatusProblem(DefectVO.PROBLEM_CLOSE);
      } catch (ObjectNotFoundException onf) {
         throw new DefectNotFoundException();
      } catch (Exception ex) {
         throw new PervasiveException("This is not possible create one order
for client", ex.getMessage());
      }
   }
}
```

In Chapter 8, “Application development” on page 157, this code is explained in detail.

The goal is to understand the tiers and have one practical view of what each class needs to do.

MVC and portlets

Portlets include both visual elements and processing logic. A typical portal can include class files, Web pages, images and some deployment files. All of these files are packaged together into a .jar file, called a Web archive file (WAR). When writing custom portlets, a Model-View-Controller design is recommended, as shown in Figure 7-7 on page 127.

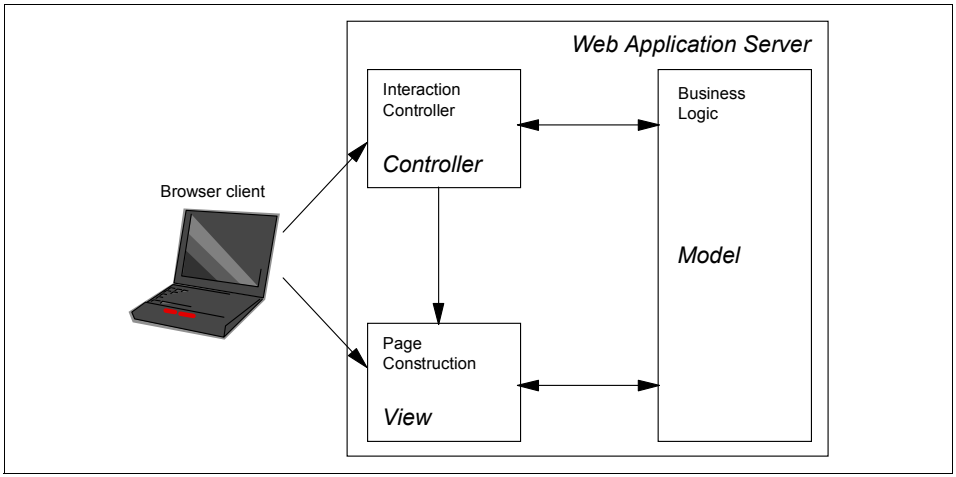


Figure 7-7 Model-View-Controller tiers

- The Controller is the class responsible for interfacing with the Model and for rendering the portlet by calling upon the appropriate View. The portlet class is responsible for executing this function.
- Views are usually implemented as JavaServer Pages. Portlets may have several different Views, including their standard View (which renders the portlet on the home page), a maximized View (which renders the portlet in its maximized state), and an edit View (which displays a page for changing the portlet settings).
- Models are usually implemented outside of portlet applications. The best approach is to implement the business logic using EJBs and just call the methods from the portlets.

The key is to separate the user interface from the business model so the user interface can be used for different Views. Portlets, JSPs, servlets and desktop applications can use the same business Model.

Pervasive portlets

Now that we have a background for a modular approach to application development, let's apply it to the pervasive development.

It is easy to see how the Model remains the same regardless of the actual display that is rendered. The Controller will essentially remain the same between the Views, but may need to perform some special customization of data based on the intended View.

Example 7-4 and Example 7-5 show a base Controller that contains common function for both the HTML and WML Views. In *SampleBaseController,* the methods to call the business operations and to pass the values in the request to an specific Controller are implemented.

Example 7-4 Controller for WML implementation

```
public class SampleWMLController extends SampleBaseController {
  public void doView(PortletRequest request, PortletResponse response)
    throws PortletException, IOException {
      getPortletConfig().getContext().include("jspWML.jsp", request, response);
  }
}
```

When you compare the sample codes, the difference that you can see is the page that the Controller will use to forward the response.

Example 7-5 Controller for HTML implementation

```
public class WeatherHTMLController extends WeatherBaseController {
  public void doView(PortletRequest request, PortletResponse response)
    throws PortletException, IOException {
      getPortletConfig().getContext().include("jspHTML.jsp", request,
response);
  }
}
```

The View component has the flexibility to render the data for a best fit on the destination display. For example, SmartPhones display content in WML (Wireless Markup Language) or another markup language developed for phones.

We can define a separate view JSP for each intended device: an HTML JSP for browsers and a WML JSP for cell phones. Look at the sample portlet in Example 7-6 and Example 7-7 on page 129, which uses this design scheme and the WebSphere Portal pervasive APIs.

Example 7-6 JSP for HTML

```
<%@ page contentType="text/html" errorPage="" %>
<jsp:useBean id="valueVO"
     class="com.ibm.itso.model.ValueVO"
     scope="request"/>
<p>
Value is  <%= valueVO.showData()  %>
```

Since the HTML view is intended for a rich content client, it can show appropriate design for one device (the browser for example), but omit it from the WML view since this is a more constrained client.

Example 7-7 JSP for WML

```
<%@ page contentType="text/wml" errorPage="" %>
<jsp:useBean id="valueVO"
     class="com.ibm.itso.model.ValueVO"
     scope="request"/>
<p>
Value is  <%= valueVO.showData()  %>
```

Up to now, we have only described how to provide views for a large categorization of devices, assuming that a broad range of devices will have the same capabilities. However, this is not usually appropriate, particularly when referring to mobile devices. For example, some cell phones support WML V1.1, others support a subset of WML 1.2 (for example: no table support), and still others support the entire WML 1.2 specification. How is it possible to provide a view that is intelligent enough to take into account all these device variations without resorting to one that supports only the least common denominator of capabilities?

The IBM WebSphere Portal API provides a mechanism for a portlet to query the capabilities of the device for which the portlet view is intended. Instead of requiring every portlet to understand the capabilities of all User-Agent types, the Portal provides an abstraction between the User-Agent and device capabilities.

How does the Portal determine device capabilities? When an HTTP request is made to the Portal, the requesting device sends a field called User-Agent in the HTTP header that contains information about the requesting device.

For example, a Nokia Communicator 9110 sends the following User-Agent field:

```
Nokia-Communicator-WWW-browser/3.0 (Geos 3.0 Nokia-9110)
```

The Portal internally maintains mapping from the User-Agent field to the expected capabilities of the device. Thus, the Portal can know the capabilities of a device and provide that information to a portlet. This provides scalability considering that new User-Agent mappings can be added to the Portal as new devices become available, but portlets does not need to change their behavior. Rather, they use the same device capabilities abstraction to know how a particular view should be rendered.

This capabilities abstraction class, appropriately called *Capability*, is a part of the *org.apache.jetspeed.portlet* package. It provides generic capability attributes, such as what level of markup is supported (for example WML 1.1 versus WML

1.2), as well as more specific capabilities, such as what specific type of function is supported (for example: JavaScript versus no JavaScript).

It can be modified in the WML JSP to take into account the capabilities of the WML client. If the client does not support WML tables, then we will output straight text.

Example 7-8 Selecting the capabilities for the device

```
<%@ page contentType="text/wml" errorPage="" %>
<%@ page import="org.apache.jetspeed.portlet.*" %>
<jsp:useBean id="weatherBean"
     class="com.ibm.wps.samples.weather.WeatherBean"
     scope="request"/>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init />
  <% if (portletRequest.getClient().isCapableOf(Capability.WML_TABLE)) {
   //show the tale with data
     } else {
   //show data without table
  } %>
```

Instead of using the previously described approach, Transcoding Technology can be used to transform the content. In this case, nothing in the code needs to be changed, because the Transcoding Technology will do it for you. The advantage of this approach is that you can publish your portlet for more pervasive devices.

The sample in this book was done with this approach using Transcoding Technology. For more information about Transcoding Technology, refer to 7.6.1, “Transcoding guidelines” on page 146.

7.4.3 Object-oriented Design patterns

Design patterns are common strategies for developing reusable object-oriented components. Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. This definition is by Christopher Alexander and show perfectly the idea of the patterns.

We will now be talking about patterns for components design. The patterns that we will show in this chapter can be used in other solutions and in other programming languages. Design patterns can exist at many levels from a very low level to specific solutions to broadly generalized system issues. There are now hundreds of patterns in the literature; they have been discussed in articles and in conferences at all levels of granularity.

It has become apparent that you do not just write a Design pattern off the top of your head. In fact, most such patterns are discovered rather than written. The process of looking for these patterns is called “pattern mining”.

Design patterns began to be recognized more formally in the early 1990s by Helm (1990) and Erich Gamma (1992), who described patterns incorporated in the GUI application framework and published the book *Design Patterns - Elements of Reusable Software*, by Gamma, Helm, Johnson and Vlissides (1995). This book, commonly referred to as the Gang of Four or “GoF” book, has had a powerful impact on those seeking to understand how to use Design patterns and has become an all-time best seller.

The authors divided these patterns into three categories: creational, structural and behavioral.

- Creational patterns are ones that create objects for you, rather than having you instantiate objects directly. This gives your program more flexibility in deciding which objects need to be created for a given case.
- Structural patterns help you compose groups of objects into larger structures, such as complex user interfaces or accounting data.
- Behavioral patterns help you define the communication between objects in your system and how the flow is controlled in a complex program.

It is not the intention of this book to explain in detail all twenty-three patterns from the GoF, but to explain the main ideas around the most common patterns that we can use in portlet development. The patterns described are:

- Singleton
- Factory
- Abstract Factory
- Proxy
- Decorator
- Command
- Facade

Singleton

This is a creational pattern that is used to ensure that a class has only one instance, and provide a global point of access to it. This pattern is interesting when you want to keep track of a sole instance. You can use this in many ways, for example, when you want to load application variables from a file or control the access to components.

The easiest way to make a class that can have only one instance is to embed a static variable inside the class that we set on the first instance and check each

time we enter the constructor. A static variable is one for which there is only one instance, no matter how many instances there are of the class.

```
static boolean instance_flag = false;
```

The problem is how to find out whether or not creating an instance was successful, since constructors do not return values. One way would be to call a method that checks for the success of creation, and which simply returns some value derived from the static variable.

Another approach, suggested by Design Patterns, is to create Singletons using a static method to issue and keep track of instances. To prevent instantiating the class more than once, we make the constructor private so an instance can only be created from within the static method of the class.

This approach was used in the sample application.

Example 7-9 Singleton pattern

```
public static ServiceLocator getInstance() throws ServiceLocatorException{
     if (myServiceLocator == null) {
        myServiceLocator = new ServiceLocator();
     }
     return myServiceLocator;
}
```

Figure 7-8 represents the class diagram for the Singleton pattern.

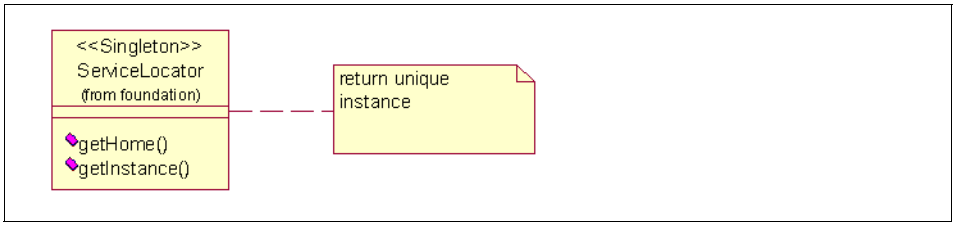


Figure 7-8 Class diagram for Singleton pattern

Factory

This is a creational pattern that is used to define an interface for creating an object, but lets subclasses decide which class to instantiate. The Factory method lets a class defer instantiation to subclasses. This approach can be found in EJB technology for home and remote classes.

A Factory pattern is one that returns an instance of one of several possible classes depending on the data provided to it. Usually, all of the classes it returns

have a common parent class and common methods, but each of them performs a task differently and is optimized for different kinds of data.

Abstract Factory

This is a creational pattern that provides an interface for creating families of related or dependent objects without specifying their concrete classes. This approach can be found in EJB technology for home and remote classes.

The Abstract Factory pattern is one level of abstraction higher than the Factory pattern. You can use this pattern when you want to return one of several related classes of objects, each of which can return several different objects upon request. In other words, the Abstract Factory is a factory object that returns one of several Factories.

Proxy

This is a structural pattern that provides a surrogate or placeholder for another object to control access to it.

The Proxy pattern is used when you need to represent a complex object with a simpler one. If creating an object is expensive in terms of time or computer resources, Proxy allows you to postpone this creation until you need the actual object. A Proxy usually has the same methods as the object it represents, and once the object is loaded, it passes on the method calls from the Proxy to the actual object. This approach can be found in remote implementation of EJB technology.

Decorator

This is a structural pattern that attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.

The Decorator pattern provides a way to modify the behavior of individual objects without having to create a new derived class. Suppose we have a program that uses eight objects, but three of them need an additional feature. You could create a derived class for each of these objects, and in many cases this would be a perfectly acceptable solution. However, if each of these three objects requires different modifications, this would mean creating three derived classes. Further, if one of the classes has features of both of the other classes, you begin to create a complexity that is both confusing and unnecessary.

We can see this applicability in the portlet API. We have one skin for each portlet and the skin has some functionality such as resizing the portlet, call edit mode for personalization issues and so on.

In a portlet development, you extend the AbstractPortlet and do not care how the engine implements this class. In other words, it is totally transparent for portlet developers.

Command

This is a behavioral pattern that encapsulates a request as an object, thereby letting you parameterize clients with different requests, queue or log requests.

The Command pattern forwards a request to a specific module. It encloses a request for a specific action inside an object and gives it a known public interface. It lets you give the client the ability to make requests without knowing anything about the actual action that will be performed, and allows you to change that action without affecting the client program in any way.

Facade

This is a structural pattern that provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.

Usually, as programs are developed, they grow in complexity. In fact, for all the excitement about using Design patterns, these patterns sometimes generate so many classes that it is difficult to understand the program's flow. Furthermore, there may be a number of complicated subsystems, each of which has its own complex interface.

The Facade pattern allows you to reduce this complexity by providing a simplified interface to these subsystems. This simplification may in some cases reduce the flexibility of the underlying classes.

The class diagram for the Facade pattern is shown in Figure 7-9 on page 135.

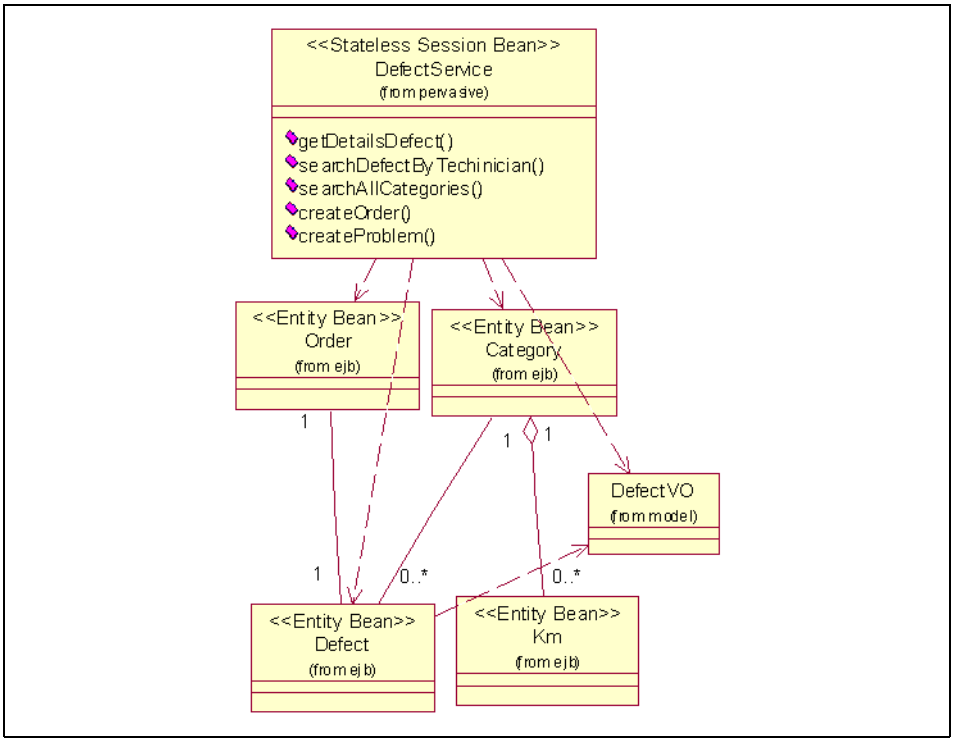


Figure 7-9 Class diagram for Facade

In this diagram, the *DefectService* works as a Facade. This class is responsible for interacting with many classes and the client. In this case, the portlet needs to talk only with *DefectService* to execute the operations.

7.4.4 Applying the Design patterns

The MVC can help you structure the tier in the application, but when you are developing components, there are many design problems that other patterns can help you deal with.

There are patterns for different solutions, but in this book we choose the patterns that you should use when you are developing solutions to Pervasive with WebSphere Portal Solution.

Data communication between the portlets and business tier

The business tier has server-side business components (session beans and entity beans). Session beans represent the business services and maintain a

one-to-one relationship with the client. Entity beans are multi-user, transactional objects representing persistent data with a data store.

Some of the service methods may return data to the client that invoked the methods. In such cases, the client must invoke the session bean methods multiple times until the client obtains values for all the attributes. Every method call made is remote and takes time to complete the operation.

Efficient transfer of remote, fine-grained data by sending a coarse-grained view of the data is the key to the Value Object Pattern. Instead of transferring fine-grained data in multiple remote calls, transferring a value object reduces network traffic and simplifies entity beans and remote interfaces.

The Value Object pattern is used to encapsulate the business data. A single method call is used to send and retrieve the object. When the client requests the enterprise bean for the business data, the enterprise bean can construct the Value Object, populate it with its attribute values, and pass it by value to the client.

Clients usually require a lot of value from an enterprise bean and to reduce the number of remote calls and avoid the associated overhead, it is best to use value objects to transport the data from the enterprise bean to the client.

When an enterprise bean uses a value object, the client makes a single remote method invocation to the enterprise bean to request the value object instead of numerous remote method calls to get individual attribute values. The enterprise bean then constructs a new value object instance and copies the attribute values from its attributes into the newly created value object. It then returns the value object to the client. The client receives the value object and can then invoke its accessor (or getter) methods to get the individual attribute values from the value object. The implementation of the value object may be such that all attributes are made public. Because the value object has been passed by value to the client, all calls to the value object instance are local calls instead of remote method invocations. We have one value object example, as shown in Example 7-10.

Example 7-10 Value object example

```
public class CategoryVO implements Serializable {
   private int categoryID;
   public int getCategoryID() {
      return categoryID;
   }
   public void setCategoryID(int categoryID) {
      this.categoryID = categoryID;
   }
}
```

Getting the references to the business component

The portlet requires a way to look up the service objects that provide access to distributed components. The business component infrastructure (EJB) uses Java Naming and Directory Interface (JNDI) to look up enterprise bean home interfaces, data sources, connections, and connection factories. The lookup code is used many times by the project.

Unnecessary JNDI initial context creation and service object lookups can cause performance problems. The Service Locator pattern can help you resolve this problem. It centralizes distributed service object lookups, provides a centralized point of control, and may act as a cache that eliminates redundant lookups. It also encapsulates any vendor-specific features of the lookup process.

Service Locator abstracts the complexity of the network operation and lookup of various services. It can also cache the service object so that the expensive lookup operation does not have to be performed every time.

Here is one example of getting the reference in the Service Locator class.

Example 7-11 Service Locator class

```
public EJBHome getHome(String name)throws ServiceLocatorException {
   Object objref = context.lookup(name);
   EJBHome home = (EJBHome)PortableRemoteObject.narrow(objref, EJBHome.class);
   return (EJBHome)PortableRemoteObject.narrow(home,
home.getEJBMetaData().getHomeInterfaceClass());
}
```

The context of this code involves the ServiceLocator class (using Singleton pattern) and *getHome* method returning the *EJBHome* object from the bean we want to use. This method works between the portlets, stateless beans and entity beans. You can use this method to get all home lookups.

Structuring the requests

Portlet Applications is a Controller in the MVC Model. The action in the form goes to a method that understands the request and redirects to the appropriate business component. Portlets can have multiple actions to perform, unlike the servlets.

In this case, you have some options; one idea is to implement the command pattern (GoF) to encapsulate a request as an object, thereby letting you parameterize the client with different requests.

Another option is to use Struts. Struts is an Apache Jakarta open source project that provides a framework based on the Model-View-Controller (MVC) that

allows developers to efficiently implement their Web applications, keeping the business logic and presentation aspects separate.

In the Portal Server environment, the ability to use Struts is a logical extension. The portlets on a Portal page can be essentially thought of as servlet applications in their own right. For many of the same reasons one would use Struts to implement a Web application, using Struts in a portlet would be desirable.

7.5 WebSphere Portal Solution guidelines

When you implement one portal solution, it is possible to integrate information from multiple sources in your company in a single point of access. In this case, you have one infrastructure to support different types of devices and data sources.

- Create components with object-oriented design principles. Design Portlet applications according to good object-oriented design principles with loosely coupled, encapsulated components to maximize the opportunity for reuse.
- Avoid writing Java code in JSP: use Java beans to encapsulate the interface operation.
- Understand the customer's problem: take the time to fully understand the requirements of a comprehensive user profile to support all related applications before starting the application design process.
- Understand Portal Solution; use the Portal in the way it was intended: as a framework for aggregating application components. Design JSPs, servlets, beans, and so forth, to handle all of the business processing and plug into the portal framework rather than developing portlets. This will maximize flexibility of the solution and reduce platform independence.
- To avoid making default portlets user-customizable, create a default page using custom portlets and persistence based on a key comprising the page, portlet, and user ID to allow for a default page. Do not use the default persistence model for shared portlets; changes made by one user will be seen by all.
- A clear definition of success and measurement criteria at the beginning of a portal effort is critical. Clear definitions of the goals, functions, and measurable success criteria will help lead to a successful effort.
- Existing business rules must be evaluated and potentially changed to fit into the user experience.
- Consumer trend data must be constantly monitored and acquired, internally and externally, to be useful. Applications must be developed with flexibility.

The vast amounts of data that can be collected must be managed and transformed to be worthwhile.

This solution should cover many functionalities and provide flexibility for the user. The Portal Solution provides some concepts to help you achieve this level of quality. These concepts, such as Single Sign-On, personalization and internationalization, are ones that the developer should take into consideration when developing the solution.

7.5.1 Internationalization

Today's international high-tech marketplace has created a need for global applications and Web-delivered content. As more and more companies attempt to accommodate their customers' needs and internationalize their offerings, the challenges of managing the technology and resources are emerging.

Included in the Java 1.1 API are the java.text package and the *ResourceBundle* classes contained in the *java.util* package, which together with the Locale classes make up the backbone of localization on the Java platform. Resource bundles use a cascading mechanism to provide the best translation support with the minimal amount of work for both the developer and resources of the computer. In this scheme, the developer creates files that define a lookup table for translation and adds encoding to the names of these files to specify the locale of the file.

Portal aggregation allows you to package JSPs to support multiple markups, clients, and locales. JSPs that contain mostly text, such as help JSPs, can be translated directly rather than storing strings in a resource bundle. You need to store JSPs that do not use resource bundles in the appropriate localized location. When a portlet uses a JSP for rendering the portlet's content, the portal searches for and selects the proper JSP based on the client type (including browser), markup language, and locale indicated in the request. To include a JSP in a portlet, use the function *PortletContext.include()*:

```
getPortletConfig().getContext().include(jsp_path/jspname.jsp,
portletRequest, portletResponse);
```

To support multiple markup types and locales, the portlet's JSPs must be packaged in the WAR. The elements in this structure are as follows.

- jsp_path: a path defined by the developer. For example, JSPs can be located in the root of the WAR file or in a JSP directory. However, this path must not include mime-type/language_country_variant. The *include()* method already locates the correct JSP also in those directories.
- markup_type: either html, wml, or chtml.
- language: the language for this JSP, for example: en, ja, or pt .

- country: the country of the JSP, for example: US, UK, or BR.
- client: the type of device. For example, this could indicate a JSP with browser-specific markup, such as IE or NS4. Manage Clients Help describes how clients are identified by the portal server.

The directory structure is:

```
jsp_path/markup_type/language _country/client/jspname.jsp
```

You can find more information about internationalization at the following site:

http://java.sun.com/docs/books/tutorial/i18n/

7.5.2 Session

The PortletSession holds user-specific data for the virtual instance of the portlet. Virtual instances differ from each other only by the data stored in their respective PortletSession or objects. The PortletSession contains transient data for the portlet virtual instance. Any persistent data must be stored using PortletData. Information stored in a portlet's class variables is shared between all virtual instances, with read and write access. Make sure you do not use class attributes for user-specific data. On the other hand, you have to be cautious about what the portlet adds to the session, especially if the portlet ever runs in a cluster environment where the session is being serialized to a shared database. Everything being stored in the session must be serializable too.

Like an HttpSession, a PortletSession is not available on an anonymous page.

During login, a PortletSession is automatically created for each portlet on a page. To get a PortletSession, the getSession() method (available from the PortletRequest) has to be used. The method returns the current session or, if there is no current session and the given parameter "create" is true, it creates one and returns it.

When you are working with Session, the Portal Application Server creates a session cookie to maintain the data from PortletSession.

We have three different ways to manage this cookie.

- The browser should accept cookie from the server. This cookie will still be on the client and when the client closes the browser, this cookie will be removed from the client machine. In another words, this cookie will exist while the session is alive.
- With WAP, the cell phone should support the cookie. The older phones do not support the cookie; in this case, it is necessary to install software to manage this cookie for you, such as WebSeal.

- With PDA the device should accept the cookie. The behavior of the PDA is similar to the Web browser.

7.5.3 Personalization

Personalization is about tailoring Web content and applications to specific users. This is done by gathering and storing information about site visitors, analyzing the information, and based on the analysis, delivering the right information to each visitor at the right time. A number of techniques can enable your site to personalize news feeds, recommend documents, provide advice, target e-mail, target advertising, and promote products.

When personalizing a Web site, there are two fundamental reasons for targeting individual users:

- To match content (information), application access, application interests, roles, and needs of each visitor to a Web site.
- To deliver the information and applications that you want your visitors to see and alter the processing of the applications based on the services you want to provide.

Elements of a personalization solution include the following:

- User profile
  - Users of the Web site
  - Attributes about the user
  - Groups and hierarchies
- Content model
  - Products, articles, programs, and so on
  - Defines attributes about the content
  - Groups and hierarchies
- Matching technology
  - User profile, rules, collaborative filtering, and collaborative filtering/market basket analysis or a combination
  - Matches the user to the right content
- Populating the user content repositories
  - Explicit and implicit
- Feedback on personalization effectiveness

There are several types of personalization:

- User profile-based: User profile-based personalization or simple filtering is used to display content based on predefined groups or user profiles. For example, when a user registers on a portal site, selects which news feeds he or she would like to see information on.
- Rules-based: Rules-based personalization is used to display content on predefined business rules.
- Collaborative filtering: Collaborative filtering is used to display information based on a combination of your preferences and those of like-minded users.

If you want more information about personalization, please refer to the redbook *WebSphere Personalization Solutions Guide*, SG24-6214.

7.5.4 Single sign-on

The users in a corporate system are commonly required to use a separate password to authenticate themselves to each server they need to access in the course of their work. Multiple passwords are an ongoing headache for both users and system administrators. Users have difficulty keeping track of different passwords, tend to choose poor ones, and tend to write them down in obvious places. Administrators must keep track of a separate password database on each server and deal with potential security problems related to the fact that passwords are sent over the network routinely and frequently.

Solving this problem requires some way for a user to log in once, using a single password, and get authenticated access to all servers that the user is authorized to use without sending any passwords over the network. This capability is known as *Single Sign-On*.

This process includes two different processes that need to be clearly understood. These are *authentication* and *authorization*.

Authentication

Authentication is the mechanism through which callers and service providers prove to one another that they are acting on behalf of specific users or systems. When the proof is bidirectional, it is referred to as *mutual authentication*. Authentication establishes the call identities and proves that the participants are authentic instances of these identities. An entity that participates in a call without establishing or proving an identity (anonymous authentication) is called *unauthenticated*.

Authentication is carried out in two phases:

- Service-independent authentication that requires knowledge of some secret is performed to establish an authentication context that encapsulates the identity and is able to create proofs of identity, called authenticators.
- The authentication context is then used to authenticate with other called or calling entities.

Authorization

Authorization mechanisms limit interactions with resources to collections of users or systems for the purpose of enforcing integrity, confidentiality, or availability constraints. Such mechanisms allow only authentic caller identities to access components. Mechanisms provided by the J2SE platform can be used to control access to code based on identity properties, such as the location and signer of the calling code. In the J2EE distributed component programming model, additional authorization mechanisms are required to limit access to called components based on who is using the calling code. As mentioned in “Authentication” on page 142, caller identity can be established by selecting from the set of authentication contexts available to the calling code. Alternatively, the caller may propagate the identity of its caller, select an arbitrary identity, or make the call anonymously.

In all cases, a credential is made available to the called component. The credential contains information describing the caller through its identity attributes. In the case of anonymous callers, a special credential is used. These attributes uniquely identify the caller in the context of the authority that issued the credential. Depending on the type of credential, it may also contain other attributes which define shared authorization properties (for example, group memberships) that distinguish collections of related credentials. The identity attributes and shared authorization attributes appearing in the credential are referred to together as the caller's security attributes. In the J2SE platform, the identity attributes of the code used by the caller may also be included in the caller's security attributes. Access to the called component is determined by comparing the caller's security attributes with those required to access the called component.

In the J2EE architecture, a container serves as an authorization boundary between callers and the components it hosts. The authorization boundary exists inside the container's authentication boundary, so that authorization is considered in the context of successful authentication. For inbound calls, the container compares security attributes from the caller's credential with the access control rules for the target component. If the rules are satisfied, the call is allowed. Otherwise, the call is rejected.

There are two fundamental approaches to defining access control rules: *capabilities* and *permissions*. The capabilities approach focuses on what a caller can do. The permissions approach focuses on who can do something. The J2EE application programming model focuses on permissions. In the J2EE architecture, the Deployer maps the permission model of the application to the capabilities of users in the operational environment.

When a user requests a resource from a server, the server collects the access-control lists (ACLs) associated with that resource and evaluates them. If the server's evaluation of the ACLs requires identification of the user, the server requests client authentication, in the form of either a name and password or a digital certificate presented according to the Secure Sockets Layer (SSL) protocol.

After the server has established the user's identity, optionally including user/group information stored in a Lightweight Directory Access Protocol (LDAP) directory, it continues its evaluation of the ACLs and authorizes or denies access to the requested information according to the user's access privileges.

Single Sign-On WebSphere Portal Server

Single sign-on support in Portal Server provides a mechanism that assists a portlet in retrieving one of several representations of a user's authenticated identity, which the portlet can then pass to a back-end application. This is much like Portal Server and the portlet acting as an authentication proxy to the back-end application. Using Single Sign-On, a user can authenticate once when logging into the Portal Server, and the user's identity is passed on to applications without requiring additional identity verification from the user. Portal Server supports Single Sign-On through Application Server as well as other authentication proxies, such as IBM Tivoli Access Manager and Netegrity SiteMinder, and leverages the Single Sign-On capabilities between Application Server and Domino.

Single Sign-On in Portal Server has two levels:

- The first is a Credential Service, which encapsulates the functionality of Single Sign-On for the portlet writer in an object provided by the Service and for which sample code exists so as to make the use of these objects easy for the portlet writer.
- The second level is more flexible but requires portlet writers to directly utilize the Single Sign-On functions of the Portal Server and manage their own connections and authentication to back-end applications.

JAAS

The Single Sign-On functions of Portal Server utilize a subset of Java Authentication and Authorization Services (JAAS). The subset is the

authentication portion; Portal Server does not support true JAAS authorization. Portal Server builds a JAAS Subject for each logged on user. The Subject consists of Principals and Credentials.

A Principal is a piece of data, such as the user ID or user's DN, that gives the identity of the Subject.

A Credential is a piece of data, such as a password or a CORBA credential, that can be used to authenticate a subject. The Subject carries around the Principals and Credentials that can be used by the portlet directly or via the Credential Service. See Advanced Topics in Developing Portlets for details on working with Single Sign-On.

JAAS is one Java specification from Sun. JAAS can be used for the authentication of users, to reliably and securely determine who is currently executing Java code, and for authorization of users to ensure they have the access control rights (permissions) required to perform security-sensitive operations.

JAAS authentication is performed in a pluggable fashion. This permits Java applications to remain independent from underlying authentication technologies. New or updated technologies can be plugged in without requiring modifications to the application itself. An implementation for a particular authentication technology to be used is determined at runtime. The implementation is specified in a login configuration file.

JAAS authorization (not supported by WebSphere Portal Server) extends the existing Java security architecture that uses a security policy to specify what access rights are granted to executing code. This architecture, introduced in the Java 2 platform, is code-centric, that is, the permissions are granted based on code characteristics: where the code is coming from, whether it is digitally signed, and if so, by whom.

Credential Service

Credential Service objects exist to handle Basic Authentication, LTPA Token authentication, and simple form-based user ID/password login challenges. Credentials can take their input identity from the JAAS Subject Principals, from the portlet configuration, or from the Credential Vault service. Portlet writers can use the Credential Service to retrieve Credentials from the Credential Vault or the JAAS Subject. Credential Service objects can also be used to pass IBM Tivoli Access Manager or Netegrity SiteMinder Single Sign-On tokens from the JAAS subject to the back-end application in the appropriate headers.

Credential Vault

The Credential Vault is a portal service to assist portlets and portal users in managing multiple identities. The Credential Vault stores credentials that allow portlets to log in to applications outside the portal's realm on behalf of the user. Portal Server provides one simple database vault implementation for mappings to secrets for other enterprise applications. The Default Vault comes pre-configured with an administrator-managed vault segment and a user-managed vault segment. The user managed vault allows users to add application definitions, such as a POP 3 mail account, under the user vault and store a mapping there. Administrator-managed vaults allow users to update mappings; however, users may not add new applications to this vault. By default, the default vault loads an encryption exit for which Base 64 encodes the passwords.

7.6 Designing the mobile applications

In the following sections, we discuss the options provided by WebSphere Everyplace Access to extend applications to mobile and wireless devices. These options cover the two ways an application can be accessible from pervasive devices, either by using wireless devices with WAP, HTML or cHTML devices or by using mobile devices that store data for further manipulation.

When you want to extend existing applications to pervasive devices, the design tasks are basically the following:

- Define the application processes that are going to be accessible from the devices.
- Define the kind of access the users must have to these applications.

The application itself may help you decide what kind of access must be provided to the users; for example, a Web application is, in general terms, easier to extend by using a browser-based device than by developing a whole new application for the device in order to manipulate the synchronized data.

The target device platform is also a decision point; if you want your application to be enabled for several device platforms, it is better to develop a Web application and make it accessible from the browsers included on each platform.

7.6.1 Transcoding guidelines

Once you have developed a portal, you may want it to be accessible by using pervasive devices such as WAP-phones and PDAs. The easiest way to do this is by enabling the Transcoding Technologies provided with WebSphere Everyplace Access.

With Transcoding Technologies, the portal content and presentation can be modified in order to fit specific device characteristics; for example, the content can be optimized to show only the most relevant information and the images can be reduced in size and color depth. By providing a single access point for multiple device types, the cost of re-development and maintenance of applications is significantly reduced.

The content is transformed according to the information associated with each device and user profile. Transcoding Technologies also transforms the markup language to make it comprehensible by the device.

In Transcoding Technologies, there are three types of *resources* that can be used to transform contents:

- XML Stylesheets
- Annotators
- Transcoding plug-ins

There is another type of resource called *preference profiles*; this resource is used by Transcoding Technologies to determine the device and user characteristics, as well as the stylesheet, annotator or plug-in to use.

Transcoding Technologies acts as a filter for a single portlet, so you need to enable this filter for every portlet you want to be accessible from wireless devices.

Next, we discuss the differences between the transformation options, as well as scenarios showing their use.

Preference profiles

This represents a particular type of device, or a particular user or group of users. Transcoding Technologies uses these files to decide how to treat documents depending on the device and the user making the request.

The device preference profiles are represented by .prop files located in:

<TTRoot>\etc\preferences\device

The user preference profiles are represented by .prop files located in:

<TTRoot>\etc\preferences\user

If the X-IBM-PVC-Device-Type field is present in the HTTP header, Transcoding Technologies uses the device profile whose file name matches the value specified for that field. If this field is not present in the HTTP header, Transcoding Technologies 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 Technologies uses the default device profile. Example 7-12 shows an example of a device preference profile.

Example 7-12 Device preference profile

```
#version = 1.0
#Wed Jul 10 12:55:23 CDT 2002
framesSupported=true
deviceRule=(User_Agent%e*MSIE 6.0*)
javaAppletSupported=true
portalOrdinal=360
portalMarkupVersion=ie
createCHTML=false
portalClient=true
desiredContentTypes=[text/html]
parent=NT.InternetExplorer
javaScriptSupported=true
```

If more than one device profile matches the incoming request, it is impossible to predict which of the matching profiles would be selected.

The user profile is selected for a request following these steps:

1. If a value is specified for UserAndSessionExtractor in <TTRoot>\etc\localConfig.prop, Transcoding Technologies tries to execute the referenced implementation of the UserAndSessionExtractor interface to obtain the user and session names and select the user profile that matches the user name.
2. You can specify a field in the HTTP header to be used to select a user profile by setting the httpUserIdField value in <TTRoot>\etc\localConfig.prop; Transcoding Technologies would then find the value of the defined field in the HTTP header and select the user profile that matches that value.
3. If the X-IBM-PVC-User field is present in the HTTP header, Transcoding Technologies uses the user profile that matches the value specified for that field.
4. If none of these methods identifies a user profile, Transcoding Technologies does not use a specific user profile. If one of these methods is used to specify the user profile and the specified user profile file is not found, Transcoding Technologies does not try the other methods.

Each profile contains values for the preferences that are important to the device represented by the profile. If the preference is not important for the device, it can be omitted so that a value can be chosen from a different profile. Transcoding Technologies will check profiles for a value in this order:

1. Specific user

2. Specific device
3. Default user
4. Default device

XML stylesheets

If your existing Web applications generate XML documents, the easiest way to extend them to wireless devices is to use XSL stylesheets; this approach also provides the company a single way of sharing data among different kind of users, while maintaining one representation of the data as XML documents.

In order to transform the XML documents, the stylesheets must be registered in the transcoding server. You can do so by:

- Adding the stylesheet resources to the configuration.
- Configuring the portlet to use a stylesheet. This can be done by using one of the following alternatives:
  - 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 portlet is not within the portlet .war file, you can specify the StylesheetFile parameter with the installed portlet.
- Specifying the stylesheets within the XML document using the wtp-condition. This approach supports multiple stylesheets for a single document.

Annotators

Annotation files are used to modify an HTML document to meet specific device characteristics. This type of modification is known as *clipping*.

This option is useful when you are trying to reach devices with limited screen size, since it only allows the showing of critical information and avoids showing irrelevant content, such as graphics. There are two types of annotators:

- Internal annotators can be created by using the tools available in WebSphere Studio Application Developer and WebSphere Studio Site Developer. They are special tags embedded in the original HTML document. When the transcoding server filters the document, these tags are processed.
- External annotators are separate and independent annotation files used by Transcoding Technologies to operate on the original HTML document. They can be created with simple text editors or with tools supplied in the Pervasive Toolkit.

The external annotators can be registered by using the XML Configuration tool, and they can be visible to all portlets, or can be set for specific portlets using Portal Administration.

Transcoding plug-ins

A transcoding plug-in is a piece of software that modifies the content of a document. These plug-ins are selected to process a document based on conditions specified by the program when the plug-in is created.

Several transcoding plug-ins are provided with Transcoding Technologies and you can obtain or develop others. Some of the plug-ins included are as follows.

- The *image transcoding plug-in* modifies images regarding color support and size to better support the display capabilities 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 smaller pieces that can be managed by the target device.
- The *HTML DOM generator* creates a Document Object Model (DOM) version of the incoming HTML documents.
- The *annotation transcoding plug-in*, also known as the annotation engine, interprets the contents of the annotation files to perform the 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* transforms HTML documents to Compact HTML documents for devices with cHTML browsers.

New transcoding plug-ins can be developed by using the API provided.

For more information about transcoding, refer to the following IBM Redbook and Redpaper:

- *Mobile Applications with IBM WebSphere Everyplace Access Design and Development*, SG24-6259
- *Transcoding Technologies in IBM WebSphere Everyplace Access V4.1.1*, REDP3592

7.7 Embedded mobile client applications

This section is a short introduction to the embedded mobile client applications for pervasive solutions.

Embedded mobile applications run on the pervasive device itself and they are specific to the solution, unlike a multi-purpose browser.

Several technologies and platforms exist today to build and run mobile applications. J2ME is only one of them, but it is one of the most promising, and it is also in line with IBM's Java strategy. In this book, you will only find the J2ME technology documented.

Note: For the sample application, we have used the Mobile Application Builder to provide a working prototype for our scenario. Using the Mobile Application Builder, we could build a client application quickly, although the functionalities and programming capabilities of the tool are very limited.

The result is platform-dependent; in our case, it is a PalmOS native application compiled from a C source.

7.7.1 J2ME

J2ME technology is a highly optimized Java runtime environment; it specifically addresses the vast consumer space, which covers the range of extremely tiny commodities such as smart cards or a pager all the way up to the set-top box, an appliance almost as powerful as a computer.

The following is a summary of key goals for this architecture:

- Provide support to a variety of devices with different capabilities. These devices often vary in the user interface, data storage, network connectivity and bandwidth, memory, power consumption, security and deployment requirements.
- Provide one architecture that can be optimized for small spaces.
- Focus on devices that can be highly personalized, often used by a single person.
- Maximize cross-platform capabilities.
- Maximize the flexibility and provide a means to support a rapidly changing market-place.

- Support multiple devices.
- Support device-specific functionality.
- Maintain a common architecture.

Keep in mind that the network connectivity is different according to capabilities (low bandwidth, wireless, and intermittent connection to high-fidelity, high bandwidth) and services.

One of the key problems that J2ME architecture tries to resolve is how to support a wide range of devices with different constraints, capabilities, features and intended uses without introducing limitations on any specific device.

One way to do it is to create a large, monolithic architecture that includes everything any application would ever need on any given device. In this case, the architecture would be too large in terms of resources.

Another way is to identify the common denominator of functionality that applies to all devices in the J2ME space.

The J2ME decided to use the second solution and introduced two concepts: configuration and profiles.

- **Configuration**: make up the set of low-level APIs that define the runtime characteristics of a particular J2ME environment.
- **Profile**: address the more device-specific and usage-specific API. The profile provides characteristics such as user interface widgets, event handling and data storage.

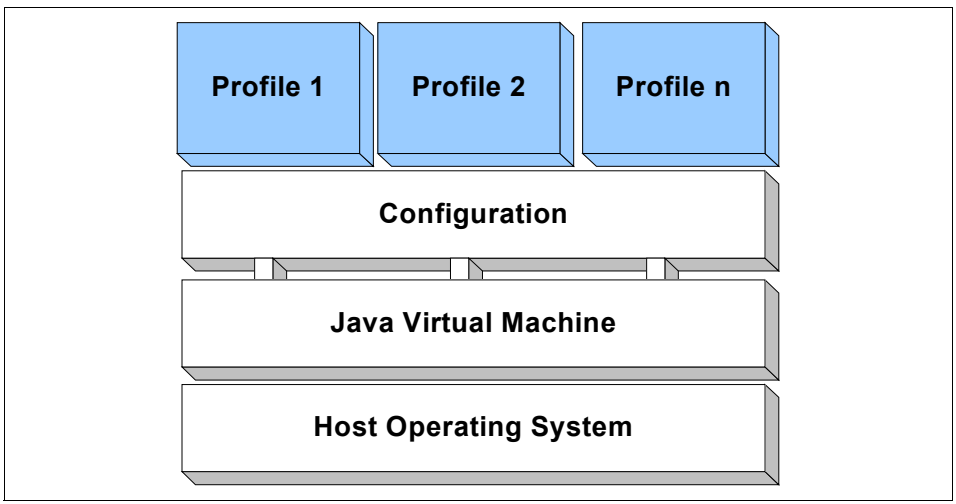


Figure 7-10 J2ME architecture tiers

Configuration defines the contract between a profile and the Java Virtual Machine; it basically has two different realizations. These configurations are CLDC (Connected Limited Device Configuration) and CDC (Connected Device Configuration).

You can see the difference between both technologies in Table 7-1.

Table 7-1 Configuration, Java Virtual Machine and devices

Configuration	Virtual Machine	Example Device
CDC	CVM J9	Pocket PC Communicator device TV set-top boxes
CLDC	KVM J9	Cellular phones PDAs Two-way pagers

The common question is: if you have two different configurations, how is portability maintained between them? We can see the relationship between the different configurations in Figure 7-11; this gives you an idea of portability in this case.

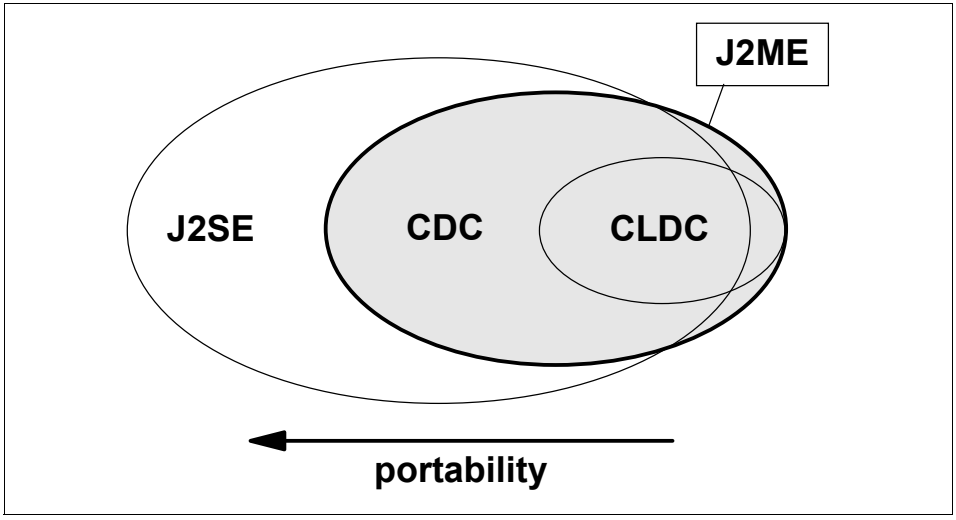


Figure 7-11 Portability between the specifications

The profile provides an API that focuses on a single device, such as a PDA or a group of related devices such as cell phones and pagers. The devices supported by a particular profile tend to have much in common in terms of how the device is used, what the user capabilities are, how the device connects to the network,

and how the device works with persistent data. Profiles are vertical and created to address different kinds of devices.

Choosing a profile is a very important decision that is made when creating applications using J2ME. The number of providers in development is increasing day by day.

Table 7-2 shows a summary of the different J2ME profiles.

Table 7-2 Profiles for J2ME

Profile	Configuration/VM	Virtual Machine	Target Device Examples
MIDP	CLDC	KVM	Cellular phones and two way pagers
PDAP	CLDC	KVM	PDA
Foundation	CDC	CVM	foundation for personal profile
Personal	CDC	CVM	Pocket PC, tablets, communicator and device
RMI	CDC	CVM	Any
Multimedia	CLDC/CDC	KVM/CVM	Any
Gaming	CLDC/CDC	KVM/CVM	Any
Telephony	CLDC/CDC	KVM/CVM	Cellular phones

The most famous profile is MIDP. This is first official profile released by Sun and originally released for cellular phones and two-way pagers. This profile now has also been implemented to run on the PalmOS platform. These devices have very constrained resources, such as a small screen for user interface, limited data entry capability and limited data storage.

The next version of MIDP will address features such as security, using HTTPS, push interface, small XML parser and sound API.

7.7.2 What has changed in J2ME for J2SE programmers

The J2ME specification had to remove some features that were available in the J2SE environment to meet the requirements in developing for pervasive devices.

Following is a list of features that have changed from J2SE:

- Java Native Interface
- User-defined class loaders
- Java Reflection
- Thread groups and daemon threads
- Finalization (Garbage Collector thread)
- Weak references
- Floating point data type (float and double)
- Some security feature and APIs
- Class file verification (modified for efficiency)
- Some error handling limitations (not all exceptions are included)

These features have been changed because obtaining the necessary memory or processing power is quite expensive.

8

Application development

This chapter will give practical advice on how to use certain development tools to build a pervasive solution for the WebSphere Portal Server.

Some of the tools introduced here are also used to build the sample application for the book. You will find detailed descriptions of these tools and even some step-by-step instructions on how to use them to build the sample application.

© Copyright IBM Corp. 2003. All rights reserved.

8.1 Application development methodology

Today, it is quite common in the industry to develop object-oriented software via an iterative and incremental process. This approach has different roots. For more information, refer to *Object-Oriented Analysis and Design with Applications* by Grady Booch, *Object-Oriented Software Engineering* by Ivar Jacobson, and *Object-Oriented Modeling and Design* by James Rumbaugh.

There is no defined standard process for development that everyone uses. Different teams typically adopt a recognized process using a vendor methodology or their services team methodology. IBM Global Services has its own methodology used in customer engagements that covers the development process. Rational® Software Corporation®, for example, uses its Rational Unified Process®.

These methodologies generally divide the development process into different phases. Each phase is done in a sequential manner and is subdivided into further smaller phases. Some phases are only run through once. Others are done over and over again, forming the iterative and incremental part of the development process. The actual process and which phases you use might differ slightly depending on the development team or organization that uses the process. We can divide the whole process into the following phases:

- Solution outline
- Macro design
- Micro design
- Build cycle
- Deployment

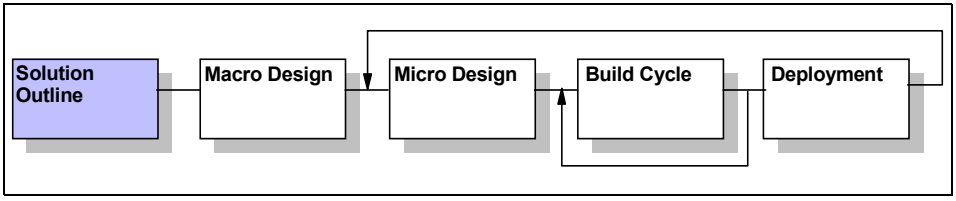


Figure 8-1 Development process overview

In the Solution Outline phase, you decide the scope of the project, explore what the essential business needs are, come up with an idea of the base architecture, and get the commitment from the project sponsor to start.

Then you start with the macro design, which concentrates on the detailed requirements gathering, business process modeling, high-level analysis and design, base architecture, and a plan for the subsequent development phases, including a development release plan. These two phases are usually done once in a project.

Now the iterative and incremental part of the development starts. For each release of the developed e-business application, the micro design, build cycle, and deployment phases are completed. Usually, a subset of use cases that has to be developed to meet a part of the system requirements make up a release. Use cases are grouped according to relevance and timeliness of functionality. The releases are defined in the project plan produced in the previous phase. A release can be an internal one that is not deployed to any users. This is quite common for early stages of big projects. Others, such as alpha or beta releases, might be deployed to a certain number of test users. It might take several iterations until a first official release of the application is deployed to the users. In turn, there are often several releases to the users until all requirements are met, plus maintenance releases to fix errors and other defects.

For more information on the application development process in the context of IBM Patterns for e-business, refer to the publication *Self-Service Patterns using WebSphere Application Server V4.0*, SG24-6175.

8.2 Pervasive solutions tools

In this section, we will explain the tools environment for developing a solution for Pervasive.

8.2.1 WebSphere Studio Application Developer

IBM WebSphere Studio Application Developer is the core development environment from IBM. It helps you to optimize and simplify Java 2 Enterprise Edition (J2EE) and Web services development by offering best practices, templates, code generation, and the most comprehensive development environment in its class.

You can create J2EE and Web services applications with integrated support for Java components, EJB, servlet, JSP, HTML, XML, and Web services all in one development environment.

You can also increase productivity with intuitive user interface, Visual Editor for Java, code assist, re-factoring, configurable runtime, incremental compilation, scrapbook, dynamic debugging, etc.

You can build new applications or enable existing assets quickly with Web services using open standards such as UDDI, SOAP, and WSDL via Web services Client Wizard.

It is also possible to transform data using a comprehensive XML development environment that offers wizards and mapping tools for creating DTDs, XML schemas, XSL style sheets and other data transformation.

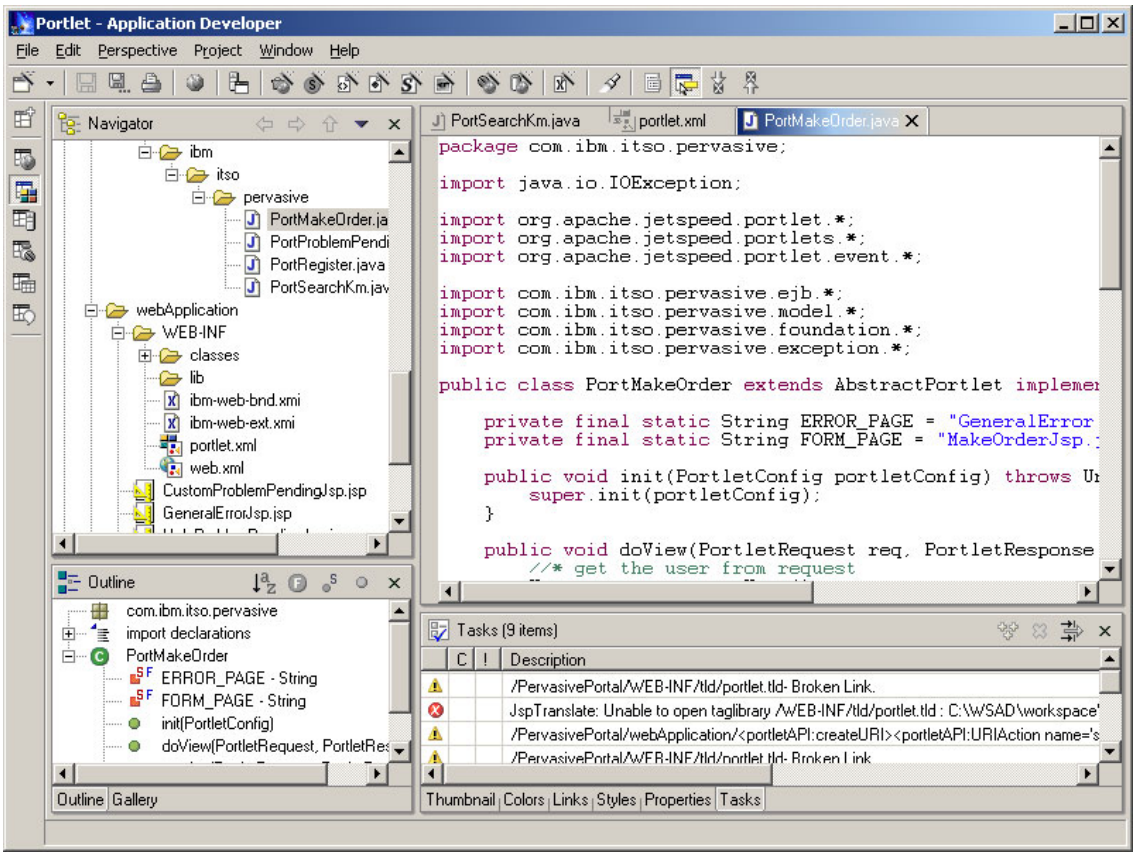


Figure 8-2 IDE for WebSphere Studio Application Developer

If you want more information about this product, go to:

http://www-3.ibm.com/software/ad/studioappdev/

8.2.2 Portal Server Toolkit

The Portal Server Toolkit is a comprehensive toolkit for developing portlet applications. It is provided in the form of plug-ins to WebSphere Studio Site Developer Advanced or WebSphere Studio Application Developer.

The Portal Toolkit provides the following features:

- Portlet projects, with which you can create basic portlets, JSP portlets, servlet invoker portlets, XSL portlets and MVC portlets.
- Portal server configuration, with which you can publish your portlet application onto your target WebSphere Portal server. Your portlet will appear on the debug page of your WebSphere Portal server.
- A capability for debugging a portlet at the source level.
- Portlet application samples for Enterprise Application.

This tool has a separate perspective inside WebSphere Studio Application Developer, as you can see in Figure 8-2 on page 160.

8.2.3 Development for pervasive devices

In this section, we discuss the configuration and development steps you need to follow in order to extend the portal application to pervasive devices such as WAP-enabled phones and PDAs. For the WAP-enabled devices, we show how to enable the Transcoding Technology bundled with WebSphere Portal Server to use it with the portlets shown in 8.3.3, “Using Transcoding Technology” on page 178. For PDAs, we show how to develop the application for Palm OS devices by using the Mobile Application builder discussed in 8.4, “Building a client application” on page 181.

Thin client

The thin client refers to those devices that can access the application online, by using either a WAP or HTML browser. When you want these clients to connect to the portal application, you may want to modify the contents of the portlets in order to fit the information into the device screen.

The Portal Server included in WebSphere Everyplace Access has embedded Transcoding Technology which allows you to configure what to show to pervasive clients depending on the device platform and/or capabilities. Is important to note that this component is *not* WebSphere Transcoding Publisher (WTP), so if you are used to working with WebSphere Transcoding Publisher you will find some differences between it and the Transcoding Technology.

In the sample scenario, the thin client can access the portal application and the user can perform the following tasks:

- Customer registration
- Submit the service request by the customer or customer representative
- Technician can check the to-do list
- Technician can browse the knowledge base
- Technician can submit an order request
- Technician can close a defect

Fat client

The fat client development can be done with several different development tools, based on different platforms, technologies and languages.

Within the WebSphere Everyplace Access there are three developer tools you can use to develop mobile applications for PDAs; a comparison of these tools is shown in Table 8-1 on page 167.

In the sample scenario, the fat client can perform the following tasks:

- Technician can check the to-do list
- Technician can browse the knowledge base
- Technician can submit an order request
- Technician can close a defect

Mobile Application Builder

This tool is intended to rapidly create applications for PDAs, to access information stored in a DB2 Everyplace database on the device. Palm OS, PocketPC, SymbianOS and any other device supporting the PersonalJava 3.0.2 specification API at JDK 1.1.7.* are supported. The MAB does not allow you to edit the generated code; however, you can add a barcode scanner and printing capabilities to the Palm OS target applications.

The code generated by MAB can be edited and recompiled with other development tools in order to add other functionalities to your applications; for Palm OS, MAB generates C code, and for all other devices it generates Java code.

To get the DB2 Everyplace Mobile Application Builder software, go to the following URL:

`http://www-3.ibm.com/software/data/db2/everyplace/.`

For more information about Mobile Application Builder, please refer to:

http://www-3.ibm.com/software/data/db2/everyplace/library.html.

WebSphere Studio Device Developer

IBM WebSphere Studio Device Developer is an integrated development environment (IDE) for the creation and testing of applications that will be deployed on handsets and other small devices.

It is the newest member of the WebSphere Studio family of application development products. It helps developers create applications that enable “devices” (cellular phones, PDAs and handheld computers) to become part of an end-to-end e-business solution.

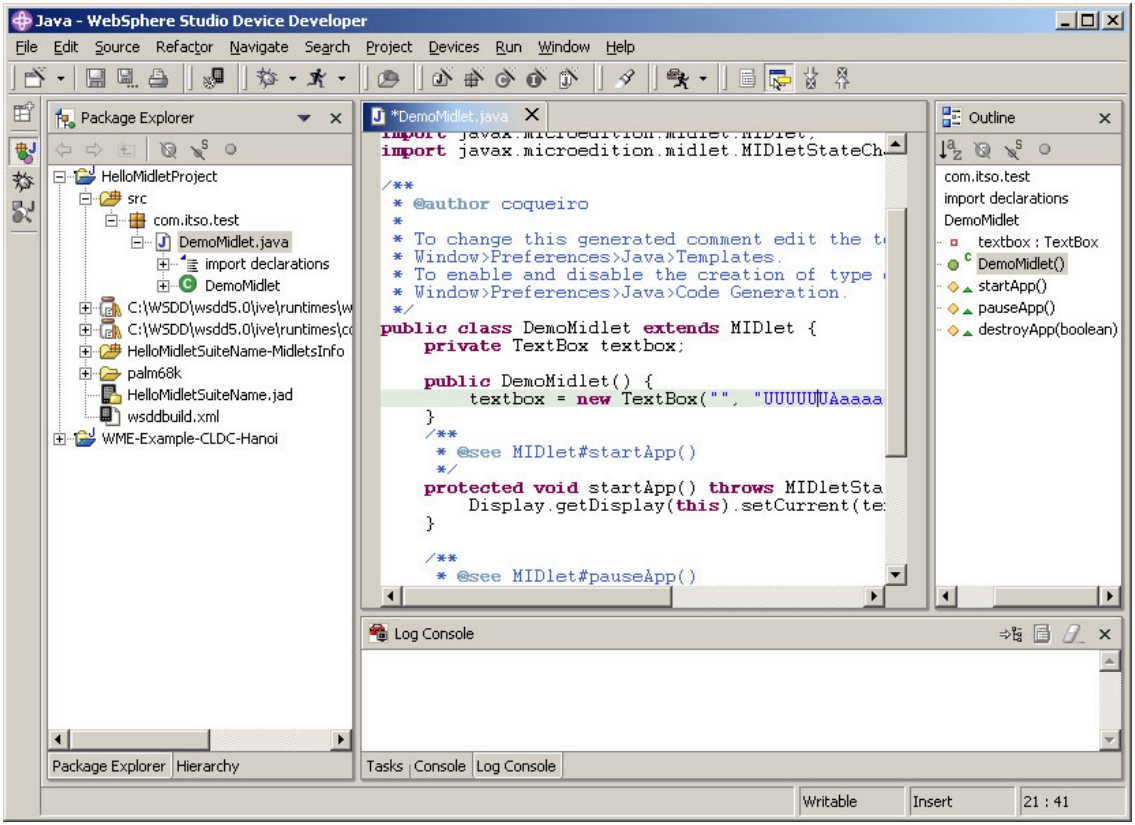


Figure 8-3 IDE for WebSphere Studio Device Developer

With this tool, you can develop applications based on C and the J2ME platform; it also contains the J9 JVM for the different device types. The applications can be developed based either on the CDC, CLDC, MIDP, or the Foundation classes. As

in the regular Java applications, the most important issue you can address with this kind of development is the portability of the code. You always use the standard configurations and profiles libraries. Another issue to keep in mind is the GUI; each device manufacturer has defined its own standard for the screen size and depth, although standards like MIDP allow you to leave the UI rendering to the JVM. This requires processing time, therefore the performance is impacted. For Palm OS devices, you can create the GUI resources using other tool, compile it with the PiLRC tools, and import the binary file into the linking process; this feature will leverage the performance of the application on a Palm OS, but you will not be able to port the application to other devices.

For more information about WebSphere Studio Device Developer, please refer to:

http://www-3.ibm.com/software/pervasive/products/wsdd/index.shtml

Domino Everyplace Enterprise Server (DEES)

Domino Everyplace Enterprise Server is a set of tools that allows you to develop, administer, run the devices and synchronize Domino mobile applications. The Domino Everyplace Enterprise Server components are as follows.

- Domino Everyplace Enterprise Mobile Application Designer: this is the Domino Everyplace Enterprise Server development tool used to create the applications for handheld devices. It enables the developer to specify Domino forms and views for the mobile application, to test the application on a device emulator, and to publish the mobile application's profile to a DEES Administration database on a Domino Everyplace Enterprise Server. Figure 8-4 on page 165 shows the Mobile Application Compiler, which is used to define the properties, forms and views the developer want to add to the mobile application.

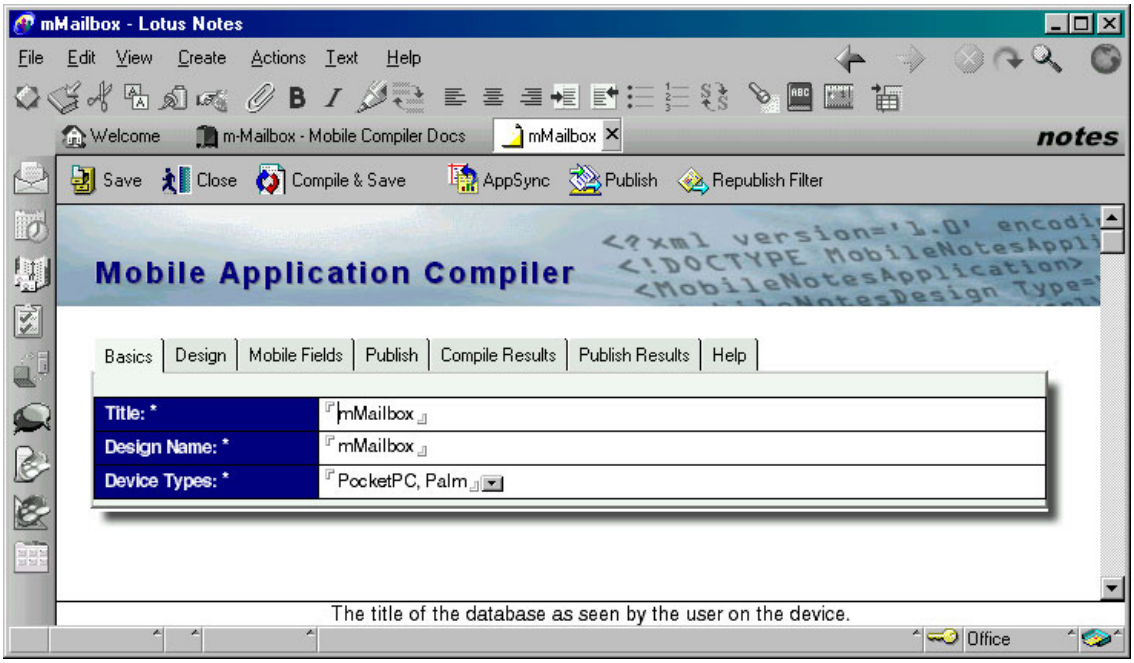


Figure 8-4 The Mobile Application Compiler form

- Domino Everyplace Enterprise Server: this is a servlet that runs on top of a Domino server and provides the synchronization services for Domino servers. It includes the DEES Administration database for administering mobile applications to users; Figure 8-5 on page 166 shows the published applications on a DEES Administration database. The Mobile Notes users connect to the Domino Everyplace Enterprise Server to synchronize both the design and data of a mobile application.

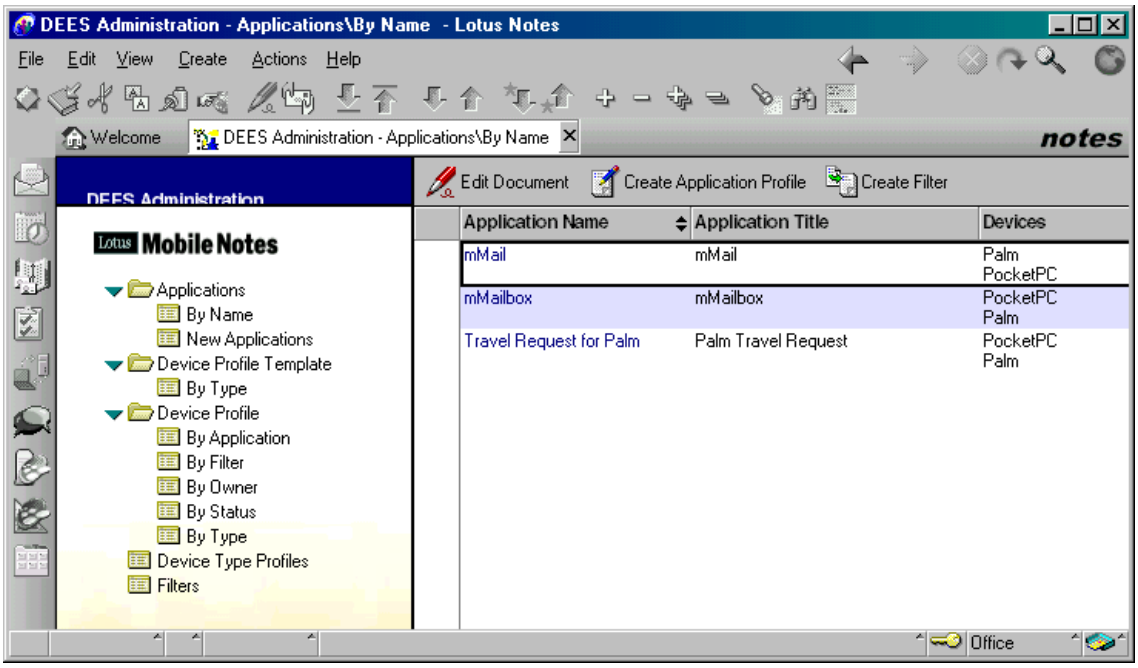


Figure 8-5 The DEES Administration Database

- Mobile Notes: this provides the Notes-like functionality on the handheld devices. From Mobile Notes, you can create, modify and delete documents from your mobile applications. The applications maintain the security features implemented on a Domino application. Figure 8-6 on page 167 shows the Mobile Notes client on a Palm OS device.

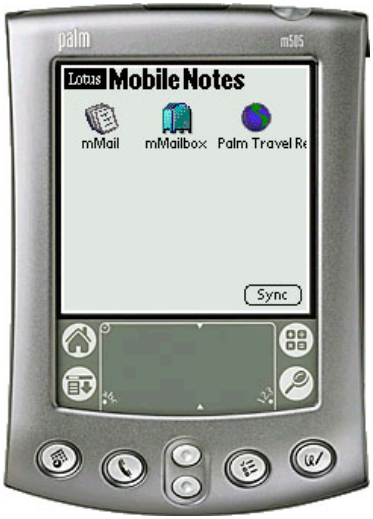


Figure 8-6 The Mobile Notes client on a Palm OS device

This is a very useful tool when you want to extend Notes applications to mobile users, maintaining the security and collaboration capabilities from the regular Notes applications.

For more information about Lotus Domino Everyplace Enterprise Server, please refer to:

http://www.lotus.com/products/domeveryplace.nsf

Table 8-1 Development Tools Included in WebSphere Everyplace Access V4.2

	Lotus Domino Everyplace Enterprise	**WebSphere Studio Device Developer**	**DB2 Everyplace Mobile Application Builder**
Version	2.7	4.0	8.1
Device Platform Supported	PalmOS, PocketPC, EPOC	PalmOS, PocketPC, Linux, Windows	PalmOS, SymbianOS, WinCE/PocketPC
Allows coding	No	Yes	No
Programming Language	Domino Designer®	J2ME, C/C++	Generates C (PalmOS), J2ME (Others)

	Lotus Domino Everyplace Enterprise	WebSphere Studio Device Developer	DB2 Everyplace Mobile Application Builder
Mobile Application Type	Lotus Notes		DB2 Everyplace
Portable Applications	Yes	Yes	No
Application Performance	Low	Low	High
Synchronization Tool	Domino Everyplace Enterprise Server		

8.3 Portlet development

In this section, we discuss the steps that you need to develop a portlet.

For the sample application, the portlets provide the content for every task.

8.3.1 Developing a portlet

This section provides the steps for developing a portlet. Basically, there are three things we must do. The first step is to create the .java class; the second step is to create the .jsp for the user to view the data and the last step is to create the deployment descriptor.

We are not covering the steps for installing the portlet in the WebSphere Portal Server; the assumption is that you already are familiar these steps. If you need more information, read the *IBM WebSphere Portal 4.1 Handbook - Volume One*, SG24-6883.

Portlet class

At this point, we can start developing the portlets. The portlet we develop here is the same that the sequence diagram represents in the previous section.

The method doView is activated when View Mode is active in the framework. This is a default mode for a portlet. When a portlet is in this mode, it is rendering data in its default display mode. This method will receive the request and generate a response.

The framework has the following other modes.

- **Edit Mode**: this mode is active when the user clicks the edit icon displayed in the portlet's title bar. When a portlet is in edit mode, a dialog is displayed that lets the user customize the settings of the portlet.
- **Configure Mode**: a portal administrator, during the manage portlets process, can invoke configuration of the portlet by selecting the **Modify Parameters** function. In this mode, the portlet displays a dialog that lets the administrator configure the portlet for the portal. These settings are global in nature, and typical settings for the portlet (the target back-end server URL, for example); they are not settings for a particular user.
- **Help Mode**: this mode is made active when the user clicks the help icon displayed in the portlet's title bar. When a portlet is in help mode, help information for the user should be displayed. This help information should explain how the user customizes the portlet settings and how the portlet display is interpreted.

Example 8-1 Getting the request from the browser

```
public void doView(PortletRequest req, PortletResponse res)throws
PortletException, IOException {
   //* get the user from request
   User user = req.getUser();
   req.setAttribute("userID", user.getUserID());
   //* forward to page error if find some problem
   if (req.getAttribute("errorMessage") == null)
      getConfig().getContext().include(FORM_PAGE, req, res);
   else
      getConfig().getContext().include(ERROR_PAGE, req, res);
}
```

In this code, the method doView gets the user information from the request and saves it in the user object. The user object is one instance of the user interface that contains methods for accessing attributes from the user, such as the user's full name or user ID.

In this case, the user ID is saved in the request. The method forwards to a JSP because we are using the MVC model and have no HTML code in the portlet class.

When the user posts the data, the container calls one event. Portlet events contain information about an event to which a portlet might need to respond. For example, when a user clicks a link or a button, it generates an action event. To receive notification of the event, the portlet must have an event listener implemented in the portlet itself. In the Portlet API, there are three types of events:

- **ActionEvents**: generated when a HTTP request is received by the portlet container that is associated with an action, such as when the user clicks a link.
- **MessageEvents**: generated when a portlet sends a message to another portlet.
- **WindowEvents**: generated when the user changes the state of the portlet window.

The portlet container delivers all events to the respective event listeners (and thereby to the portlets) before generating the new content that the event requires. Should a listener, while processing the event, find that another event needs to be generated, that event will be queued by the portlet container and delivered at a point of time that is at the discretion of the portlet container. All that is guaranteed is that it will be delivered and that this will happen before the content generation phase.

The most common event is the ActionEvent. An ActionEvent is sent to the related portlet when an HTTP request is received that is associated with a PortletAction. PortletActions can be linked to URLs by using the PortletAPI. Normally, PortletActions are linked with HTTP references or buttons in HTML forms, enabling the portlet programmer to implement different processing paths for different user selections. The ActionEvent of the clicked link then carries the associated PortletAction back to the portlet which in turn is able to process different paths on behalf of the PortletAction.

We have one sample for the ActionEvent, below.

Example 8-2 Getting the ActionEvent

```
public void actionPerformed(ActionEvent event) {
   DefaultPortletAction action = (DefaultPortletAction)event.getAction();
   PortletRequest req = event.getRequest();
   if (action.getName().equals("save")) {
      //* This action will call when the user want to save defect.
      try {
         //* Get the data from request
         Long defectID = new Long(req.getParameter("txtDefectID"));
         String comments = req.getParameter("txtComments");
         //* Call one stateless bean to create one order to client
         DefectServiceHome defectServiceHome =
         (DefectServiceHome)ServiceLocator.getInstance().getHome("ejb/DefectSe
         rvice");
         DefectService defectService = defectServiceHome.create();
         defectService.createOrder(defectID, comments);
         //* Send one message jsp
         req.setAttribute("appMessage", "Order Registered");
```

```
      } catch (DefectNotFoundException dnfe) {
         req.setAttribute("appMessage", dnfe.getMessage());
      } catch (Exception e) {
         e.printStackTrace(System.out);
         req.setAttribute("errorMessage", e.getMessage());
      }
   }
}
```

In the actionPerformed method, you will receive the event as an object. This object has one method to get the action. The input form sends this action name for portlet. When the technician fills the form and clicks the **Submit** button, the HTML sends one action (save) and starts the process.

This method will get the data from the request and will call one singleton Service Locator. This component centralizes distributed service object lookups and provides a centralized point of control. This class will return one EJBHome class. After that, the portlet will get one remote reference to start to call the business methods. If the business methods have an exception, the component will get the exception and write one message in the request.

JavaServer Pages for portlets

The next step in development is to create the component to show the data coming from the portlet class. The portal aggregator calls the portlet to render the data and the portlet responds with the stream of the response object.

The job to implement the interface becomes more difficult when the interface becomes more complex. It is necessary that one method for a portlet delegate rendering to another entity; the JSP is the entity which commonly does this job.

The WebSphere Portal Server has a portlet API to provide the class to support the use of JSP components in order to render portlet markup. Refer to the JSP code shown in Example 8-3.

Example 8-3 JSP interface code for the user type defect

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>

<jsp:useBean id="userID" class="java.lang.String" scope="request"/>

<form name="<portletAPI:encodeNamespace value='frmMakeOrderJsp'/>"
method="post"
action="<portletAPI:createURI><portletAPI:URIAction name='save'/>
</portletAPI:createURI>">

Technician ID:     <%= userID %> <BR>
```

```
Select the Defect: <INPUT size="20" type="text" name="txtDefectID"><BR>
Comments:          <TEXTAREA rows="2" cols="20"
name="txtComments"></TEXTAREA><BR>

<INPUT type="submit" name="ok" value="OK">
<INPUT type="reset" name="Clear" value="Clear"></P>

</FORM>
```

In this code, the JSP receives one JavaBean that was created by the portlet. The portlet places the JavaBean in the portlet request scope. In this code, the name of the bean is userID.

The portlet call the portlet context method include() to invoke the JSP. You can see this call in Example 8-1 on page 169.

The JSP component uses the <jsp:useBean> tag to establish the reference to the bean and extract the data from the request. You can write the value from the bean in a traditional JSP way. This is a basic step necessary to get the data from portlet.

Another consideration is how to send the data to the portlet and call an action. This JSP has a custom tag know as portletAPI:URIAction name='save'. In this tag, you can define the name of the action. The URIAction works only if you define another tag, CreateURI.

The CreateURI is used to create an URI that points to the current portlet with the given parameters. You can pass a parameter or action in the URI by including URIParameter or URIAction between the createURI start and end tags.

The URIAction is used to add a default action to the URI of the createURI and createReturnURI tags. The portlet must provide an action listener to handle the event.

This action will activate and you can code one method to execute the operations compatible with this action.

Deployment descriptor for the portlet

The last step is to create the deployment descriptor file.

A collection of related portlets make up a portlet application and are packaged together in the Web archive (WAR) file. Portlets packaged together in a single .war file may share images, stylesheets, JSP components, and other kinds of resources. A .war file is a specially structured JAR file that includes a special XML descriptor called the *Web application deployment descriptor*.

The file name for the XML descriptor file is web.xml. This file is always stored in the WEB-INF directory of the WAR file.

Example 8-4 The web.xml file

```
<?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>PervasivePortal</display-name>
    <servlet id="Servlet_1">
        <servlet-name>PortSearchKm</servlet-name>
        <display-name>PortSearchKm</display-name>
        <servlet-class>com.ibm.itso.pervasive.PortSearchKm</servlet-class>
        <init-param>
           <param-name>listener.action</param-name>
           <param-value>com.ibm.itso.pervasive.PortSearchKm</param-value>
        </init-param>
    </servlet>
    <servlet-mapping id="ServletMapping_1">
        <servlet-name>PortSearchKm</servlet-name>
        <url-pattern>/PortSearchKm/*</url-pattern>
    </servlet-mapping>
</web-app>
```

In this example, we have the reference and mapping for only one portlet. If you have more portlets, you will have more entries. In a sample application, we have the complete version for the project. The portlet here is PortSearchKM (see this on the servlet-name tag) and the class name is com.ibm.itso.pervasive.PortSearchKm (see this on the servlet-class tag).

The Web application descriptor describes the portlets in the portlet application .war file to the application server as servlets. Each portlet is defined as a servlet, and the portlet application is synonymous with the definition of the Web application.

One important point is that when you have the action, you should define the listener action. You can have one class for the portlet and another class to receive the actions. In this case, the two classes are the same but if you want to change the action, you should change the class in the param-value class.

Look at the tags below:

```
<param-name>listener.action</param-name>
<param-value>com.ibm.itso.pervasive.PortSearchKm</param-value>
```

In addition to the web.xml file in the portlet application's .war file, a .war file that contains a portlet application and associated portlets must also contain a portlet

application descriptor. The portlet application descriptor file name is portlet.xml, and this file is also always stored in the WEB-INF directory of the .war file.

This document describes the portlet application, each portlet in the application and each portlet's title and capabilities. Refer to Example 8-5.

Example 8-5 The portlet.xml file

```
<?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="DCE:f3f5b310-f757-1201-0000-005d15cf6940:1"
major-version="1" minor-version="0">
   <portlet-app-name>PervasivePortal application</portlet-app-name>
   <portlet id="Portlet_1" href="WEB-INF/web.xml#Servlet_1" major-version="1"
minor-version="0">
      <portlet-name>Search KM</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="DCE:f3f5b310-f757-1201-0000-005d15cf6940:1.1">

<concrete-portlet href="#Portlet_1">
    <portlet-name>ITSO Search Knowledge Managment portlet</portlet-name>
    <default-locale>en</default-locale>
    <language locale="en">
        <title>Search Knowledge Management</title>
        <title-short></title-short>
        <description></description>
        <keywords></keywords>
    </language>
</concrete-portlet>
```

```
</concrete-portlet-app>
</portlet-app-def>
```

In this file, you can find the link between the portlet (portlet.xml) and the servlet (web.xml). In this file, we can find a lot of interesting information such as whether the portlet can be maximized, or the name that the user can see to identify this portlet on the list of portlets, or what kind of device this portlet can be used.

The information about the device can be set in the markup tag.

Example 8-6 Tags to set portlet to HTML, WML and cHTML devices

```
<supports>
  <markup name="html">
    <view/>
  </markup>
  <markup name="wml">
    <view/>
  </markup>
  <markup name="chtml">
    <view/>
  </markup>
</supports>
```

In Portal Administration, this information will be available when you install the portlet application.

Generating the portal deployable code

This step is really easy if you have some tool such as WebSphere Studio Application Developer with the Portal Toolkit.

The XML files, portlets and JSP files are inside the \WEB-INF folder. The portlet follows the same conventions as a J2EE specification. You should create one .war file to deploy portlets inside the WebSphere Portal Server.

Select **Export WAR** in WebSphere Studio Application Developer. You should see a window as shown in Figure 8-7 on page 176.

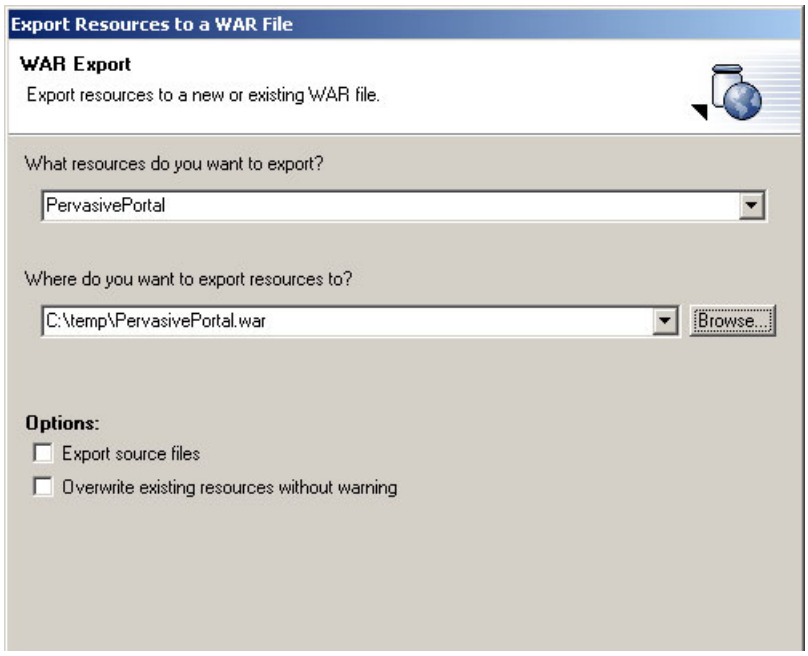


Figure 8-7 Generating the portlet .war file

You can choose the place that you want to generate the deploy file. For example, this could be C:\temp\PervasivePortal.war.

Now your portlet has finished delivering to WebSphere Portal Server. The Portal Administrator should import this in Portal Page Administration to finalize the process.

For more information about deploying the sample application in a runtime environment, refer to Chapter 12, “Technical scenario” on page 271.

8.3.2 User registry

The application exhibits a different behavior if the user logged in is a technician or a customer. The information about these two actors is stored in an LDAP directory. The product responsible for managing this data is IBM SecureWay Directory.

IBM SecureWay Directory is a Lightweight Directory Access Protocol (LDAP) directory that runs as a stand-alone daemon. It is based on a client/server model that provides client access to an LDAP server. IBM SecureWay Directory provides an easy way to maintain directory information in a central location for storage, updating, retrieval, and exchange.

IBM SecureWay Directory provides Secure Sockets Layer (SSL) V3 support, both for the directory server and client. SSL provides encryption of data and authentication using X.509v3 public-key certificates. The directory may be configured to run with or without SSL support. IBM SecureWay Directory also supports LDAP referrals, allowing directory operations to be redirected to another LDAP Directory server. Replication of the LDAP Directory is supported and allows for additional copies of the directory to be available for directory read operations, which increases performance and reliability when accessing directory information.

The suggested mapping for the structure to use the LDAP can be seen in Table 8-2 below.

Table 8-2 Structure of LDAP

Business Name	Class	Attribute
User Identification	inetOrgPerson	uid
Password	person	userPassword
First Name	inetOrgPerson	givenName
Last Name	person	sn

The next step is for the users to populate this structure. Of course, you can create your own users, but if you want to see an example, take a look at Example 8-7.

Example 8-7 Example for Technician

```
dn: uid=tech1, cn=users, dc=itso, dc=ral, dc=ibm, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: tech1 Technician
givenName: tech1
sn: Technician
uid: tech1
userPassword: password
```

You can customize this LDAP structure to save the information about the employee (Technician). We have one more example for another kind of user, the Customer. In this case, both structures are the same but in real conditions, the intranet and Internet profiles would be completely different.

Example 8-8 Example for Customer

```
dn: uid=client1, cn=users, dc=itso, dc=ral, dc=ibm, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: client1 Customer
givenName: client1
sn: Customer
uid: client1
userPassword: password
```

Now we have the users for the portal, but the next step is to separate the Technician and Customer. We can separate using groups in LDAP. We have been using the default definition group that is configured when you installed IBM SecureWay. This group has the information about the users who work with this application.

Example 8-9 Example for a group of users

```
dn: cn=technician, cn=groups, dc=itso, dc=ral, dc=ibm, dc=com
objectclass: groupOfUniqueNames
description: Technician from ITSO Help Desk
cn: technician
uniquemember: uid:tech1, cn=users, dc=itso, dc=ral, dc=ibm, dc=com
uniquemember: uid:tech2, cn=users, dc=itso, dc=ral, dc=ibm, dc=com
uniquemember: uid:tech3, cn=users, dc=itso, dc=ral, dc=ibm, dc=com
```

If you want to test all scenarios with this application, you should have the Customer and Technician users as shown in Table 8-3.

Table 8-3 List of users to use the application

User Identification	Profile
client1	Customer
client2	Customer
tech1	Technician
tech2	Technician

8.3.3 Using Transcoding Technology

The Transcoding Technology comes as a feature of WebSphere Portal in WebSphere Everyplace Access.

If you log on as an administrator to WebSphere Portal on a WebSphere Everyplace Access installed machine, select the **Portal Administration** option from the menu, then select the **Portal Settings** tab and **Global settings** under it.

At the bottom of the page, you will find a check box, by which you can enable transcoding for the portlet content, as shown in Figure 8-8.

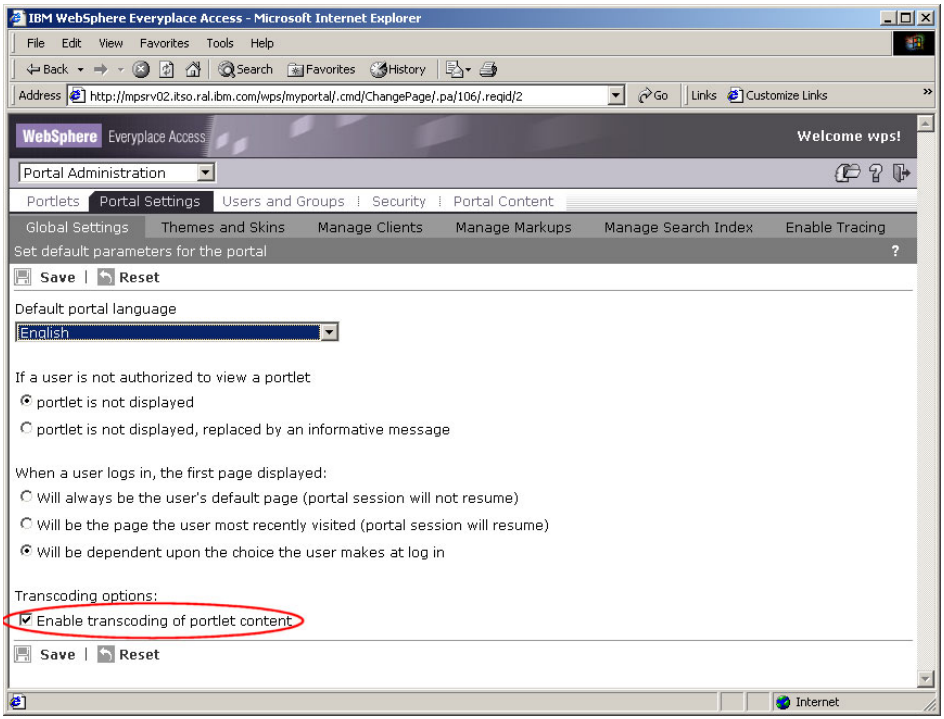


Figure 8-8 Enabling Transcoding Technology

This is a global setting for WebSphere Portal, and it is set to be enabled by default.

If you want to transcode your portlets, you have to configure the Transcoding Technology for the portlet. In order to do so, you have to select the **Portlets** tab, then **Manage Portlets** under it.

Select your portlet from the list of portlets, then select the **Modify Parameters** link. For example, select **Welcome Portlet** from the list.

A page will appear with the details and settings for the portlet; add a new parameter, type in `FilterChain` for the parameter, and `Transcoding` as the value, then click **Add**.

You should get the resulting window as shown in Figure 8-9.

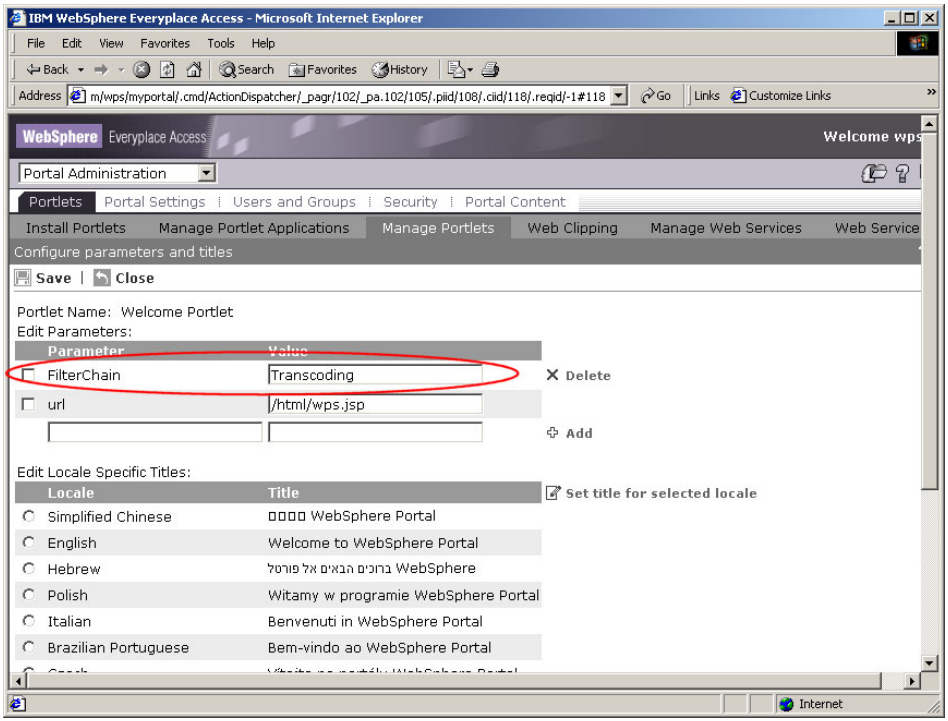


Figure 8-9 Configuring Transcoding for a portlet

Once you have transcoding enabled using the FilterChain in WebSphere, you can access the portal application using a WAP phone or, in our case, an emulator. In this case, we accessed the `http://<servername>/wps//portal` site; first, we got the list of items from the Welcome portlet, then we looked at the World time to get the result as shown in Figure 8-10 on page 181.

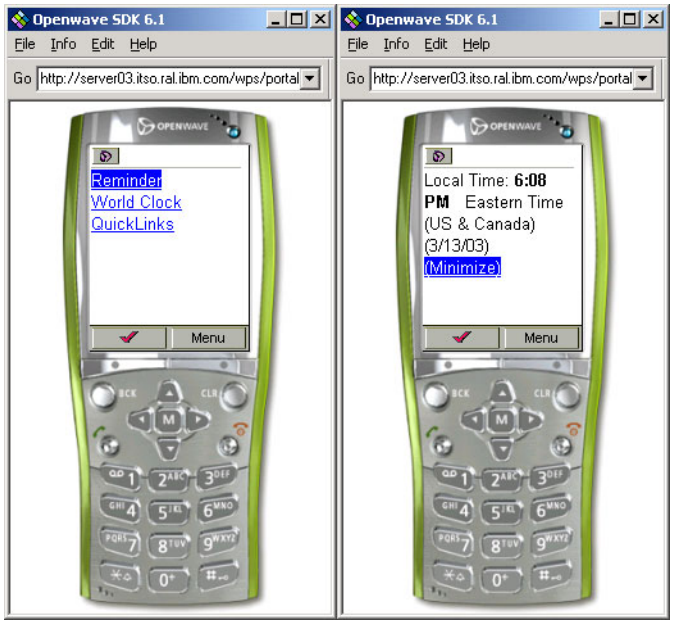


Figure 8-10 Portlet results on a WAP phone

In the sample application, the Transcoding Technology enables the users to access Web content on pervasive devices, for example a wireless laptop, PDA, smartphone, interactive voice response system.

8.4 Building a client application

The objective of this section is to quickly show a sample application built using the Mobile Application Builder.

In the sample application, this fat client application handles the to-do list, the knowledge base, and the order request functions for the technician.

The recommended development environment for pervasive applications is the WebSphere Studio Device Developer. We have used the Mobile Application Builder for our project to generate a simple application quickly and easily for the sample scenario. The capabilities of the Mobile Application Builder are very limited; it does not allow any programming, only drag and drop, rapid application development. This product is good for quick prototype development.

This section will not go into details as to how to use the Mobile Application Developer; we only provide some information about the developed sample client application.

1. Open the project ITSO HelpDesk.mab under the ITSOHelpDesk directory.

   You should get the window shown in Figure 8-11.

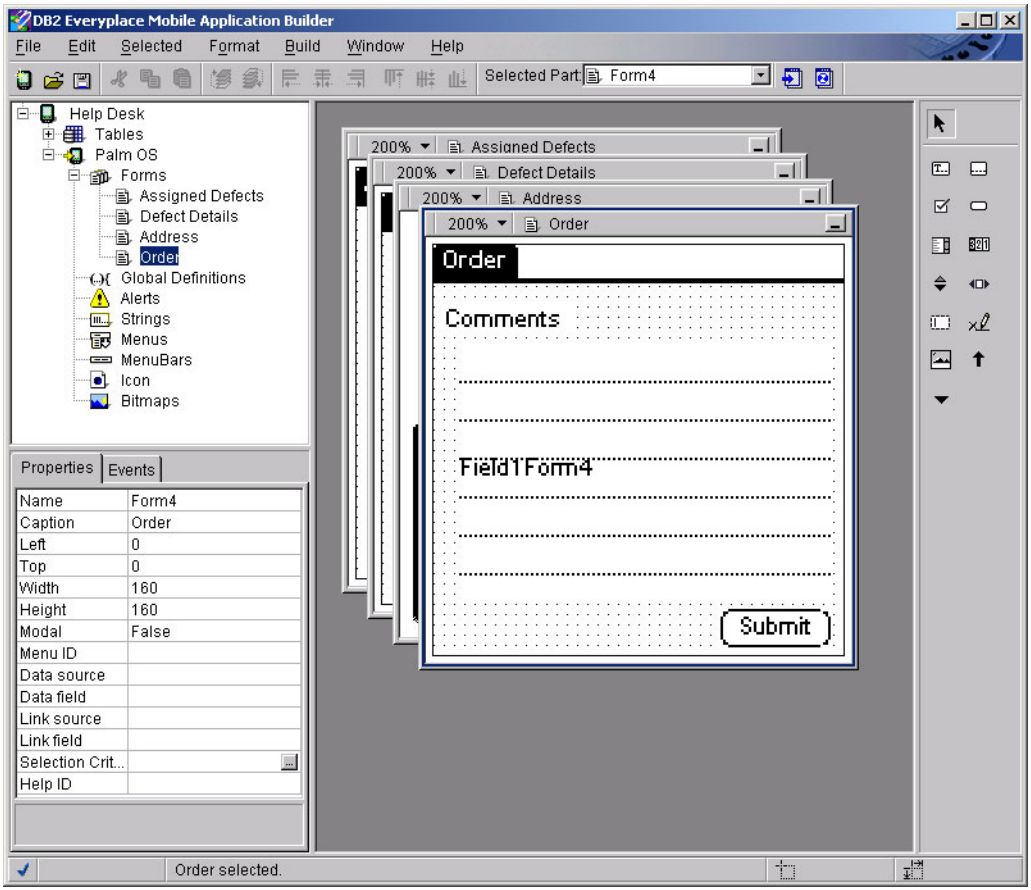


Figure 8-11 Mobile Application Builder

2. The application for mobile devices is already developed; you only have to compile and deploy it on a PalmOS device. For more information about the building process, refer to the product documentation. The building process requires different applications and libraries to be installed on the system in order to compile the code.

3. Once the code is built and you have the resulting .prc file for the Palm device, you have to install the following libraries on the PalmOS device in order to use

DB2 Everyplace and synchronization working on PalmOS for the DB2 Everyplace Server:

- Clients\PalmOS\sync\imsaconfig.prc
- Clients\PalmOS\sync\imsadb2e.prc
- Clients\PalmOS\sync\imsafile.prc
- Clients\PalmOS\sync\isyncl.prc
- Clients\PalmOS\sync\isynconf.prc
- Clients\PalmOS\sync\isyncore.prc
- Clients\PalmOS\sync\isynce.prc
- Clients\PalmOS\sync\isyncui.prc
- Clients\PalmOS\sync\wbxmllib.prc
- Clients\PalmOS\database\DB2eCat.prc
- Clients\PalmOS\database\DB2eCLI.prc
- Clients\PalmOS\database\DB2eComp.prc
- Clients\PalmOS\database\DB2eRunTime.prc
- Clients\PalmOS\database\DB2eDMS.prc
- Clients\PalmOS\database\PBSPkcs11.prc*
- Clients\PalmOS\QBE\lang\QBE.prc

4. Install the application ITSO HelpDesk.prc file, then reset the device.

Important: The following step assumes that you have a DB2 Everyplace server up and running and configured for the application. For configuration details, refer to 8.5.2, “Configuring the DB2 Everyplace Server” on page 186.

5. Configure the DB2 Synchronization on the PalmOS device.

   You will need the server name and port. The server name is that of the server where your synchronization server is running, the port number is that of the port where the server is listening. You can get the port number from the WebSphere Administrative Console if you look up the port number for the Web container transport, for example: 9082.

   You also have to specify the user name and password, for example: `tech1` and `password`.

6. Synchronize the database for the application, then start the Help Desk Palm application.

8.5 Everyplace Synchronization Server

Everyplace Synchronization Server is the component provided by WebSphere Everyplace Access for synchronizing data between several types of data repositories and the pervasive devices. It is intended to allow the mobile devices

to synchronize personal information data (PIM) and relational database management systems (RDMS) with the back-end repositories.

The Synchronization Server includes adapters for synchronizing with databases such as Lotus Notes, Microsoft Exchange and DB2 or other JDBC-compliant databases.

Synchronization Server uses the Java 2 (J2EE) platform and SyncML V1.0 protocol between the server and the client devices. The devices can use any connection type that can support TCP/IP.

Synchronization Server is a set of servlets that are installed as enterprise applications in WebSphere Application Server. These servlets are as follows.

- The SyncML servlet handles SyncML requests from pervasive devices.
- The Relay servlet handles requests from desktops using TrueSync Plus.
- The DB2 Everyplace Sync Server is the DB2 Everyplace synchronization engine and allows the synchronization with JDBC-compliant databases. The Mobile Devices Administration Center, shown in Figure 8-12, is used to administer users, filters and subscriptions.

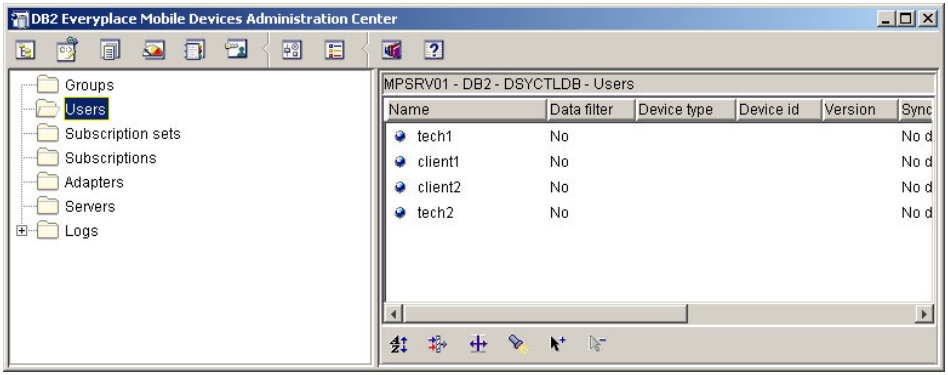


Figure 8-12 The Mobile Devices Administration Center

The adapters provided by Synchronization Server convert data into a common format in order to synchronize it between the back-end repositories and the devices. The adapters provided by Synchronization Server are as follows.

- Lotus Domino Adapter: provides an interface for the Synchronization Server to access enterprise Lotus Domino databases. Supports e-mail, address book, calendar, to dos and journal data using the standard templates Std50Mail, iNotes5, Std4PersonalAddressBook, Std50Journal, Std6Journal, and custom templates defined in the Mapping Database directory.

- Microsoft Exchange Adapter: provides an interface for the Synchronization Server to access Microsoft Exchange databases. Supports e-mail, contacts, calendar, tasks and notes synchronization.
- Relational Database Adapter: uses DB2 Everyplace Sync Server to provide an interface for the Synchronization Server to access DB2 or other JDBC-compliant databases.

8.5.1 Using DB2 Everyplace

Figure 8-13 depicts the relational database synchronization environment for JDBC subscription types.

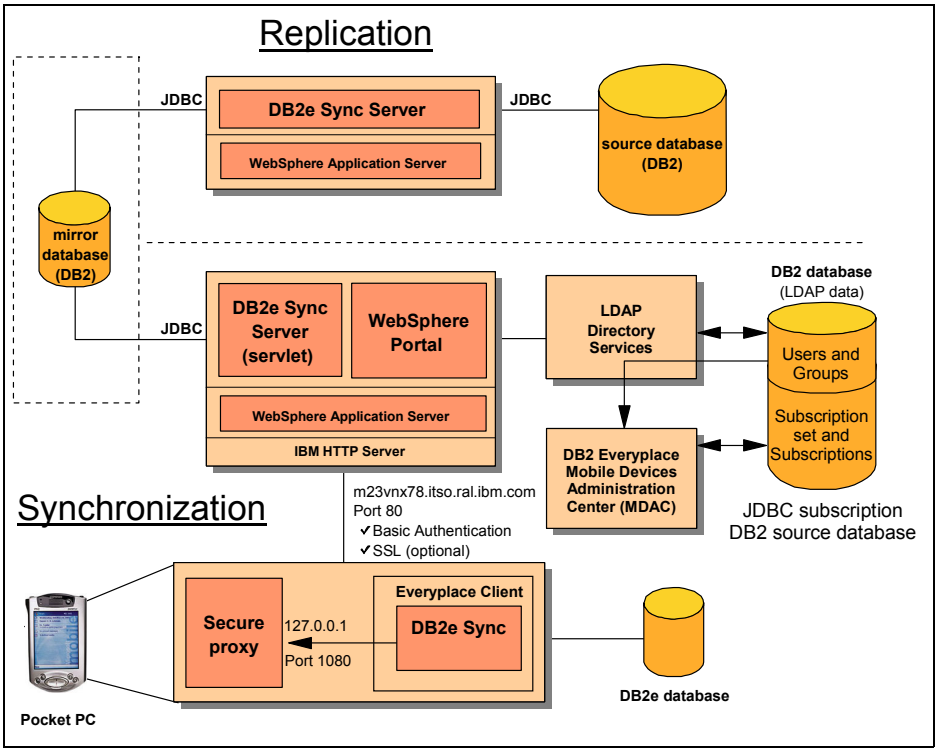


Figure 8-13 DB2 Everyplace synchronization for JDBC subscriptions

The hand-held device sits on the IBM Everyplace Client, which is the unified client. The unified client has a component called Secure Proxy, which is transparent to the user. Secure Proxy handles user authentication and data encryption between the client and the Web Server.

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. DB2 Everyplace Mobile Devices Administration Center or MDAC, together with WebSphere Portal, provides the complete administration functionalities for 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 more information about DB2 Everyplace and database synchronization, refer to the IBM Redpaper *Relational Database Synchronization in WebSphere Everyplace Access V4.1.1*, REDP3590.

8.5.2 Configuring the DB2 Everyplace Server

The following steps show you how to set up the sample application that is using DB2 Everyplace to interact with the application database in order to collect information on the PDA for the Technician in the field.

1. Create a mirror database for the application, run the script mirror.bat from the database directory.
2. Create the SyncGroup with the portal interface. Add the wpsadmin user to the SyncGroup group. Also add all the Technician (tech1, tech2) and Client (client1, client2) users to the group.
3. Create the DB2e_Technicians group and add the users: tech1, tech2 to the group.
4. Create the DB2e_Clients group and add the users: client1, client2 to the group.
5. Start the DB2 Everyplace Mobile Devices Administration Center.
6. Create a new subscription set, right-click the **Subscription Set**, then select **Create**. Set the Name field to `HelpDesk`, then add the DB2e_technicians group to the subscription set.

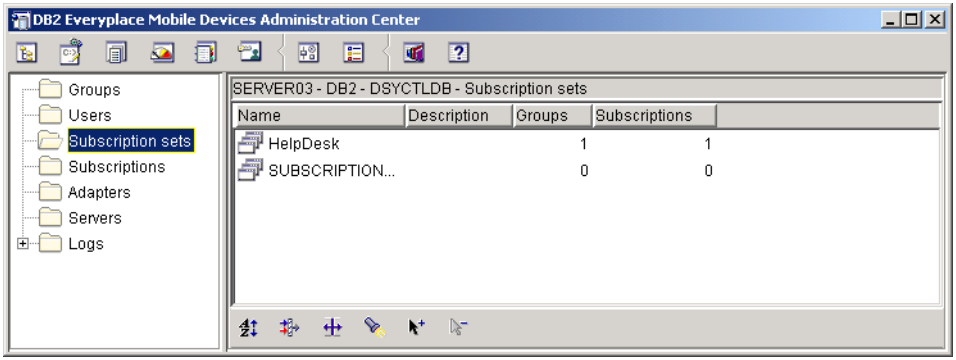


Figure 8-14 Subscription Set

7. Create a new subscription; right-click **Subscriptions**, then select: **Create -> Table Subscription -> JDBC subscription...**
8. Specify the following field for the subscription:

   Name: `Customer Service`
9. Switch to the Source tab, the set the following fields:

   Database URL: `jdbc:db2:PERPORDB`

   User ID: `db2admin`

   Password and Verify password: `password`
10. Switch to the Mirror tab and set the following fields:

    Database URL: `jdbc:db2:PERPORMI`

    User ID: `db2admin`

    Password and Verify password: `password`
11. Switch to the Subscription sets tab and add the HelpDesk subscription set to this subscription.
12. Switch back to the Identification tab and click the **Define subscription...** button.
13. On the Define Replication Subscription panel, click **Add** then add the following tables to the subscription:
    - DBUSER.CATEGORY
    - DBUSER.DEFECT
    - DBUSER.ORDER
14. Select the **DBUSER.DEFECT** item, then click **Advanced**.

15. Switch to the Rows tab, then specify the following condition for the Subset of rows for individual users:

```
TECHNICIANID=:TECHID
```

16. Close the Replication panel.

17. Click **OK** to create the subscription.

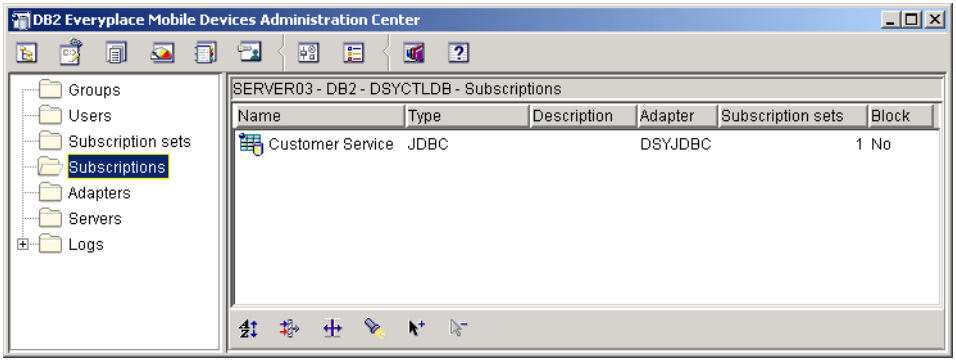


Figure 8-15 Subscription

18. Select the **Groups** node, right-click the **DB2e-Technicians** item and select **Edit...**.

19. Select the **Data filter** tab, then click **Add**. Set the Parameter name to `TECHID`, then click **OK**. Click **OK** on the Edit group panel.

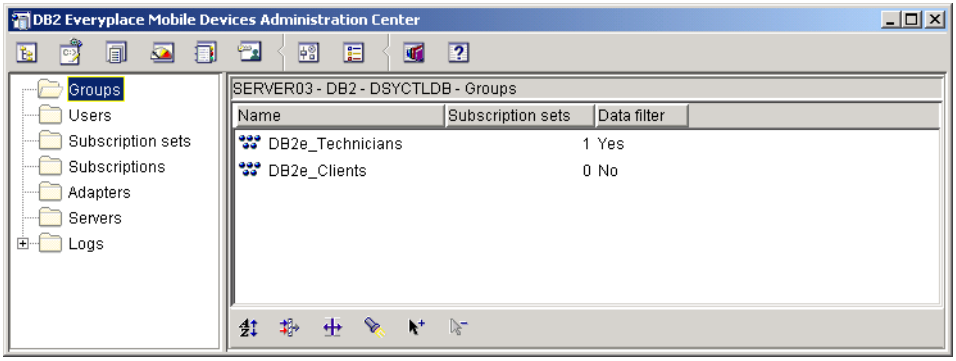


Figure 8-16 Groups

20. Select the **Users** node, then right-click **tech1** from the list and select **Edit...**.

21. Select **TECHID** under the Data filter tab, then click **Change**. Set the User override field to `tech1`. Click **OK**.

22. Repeat the last two steps for the user tech2.

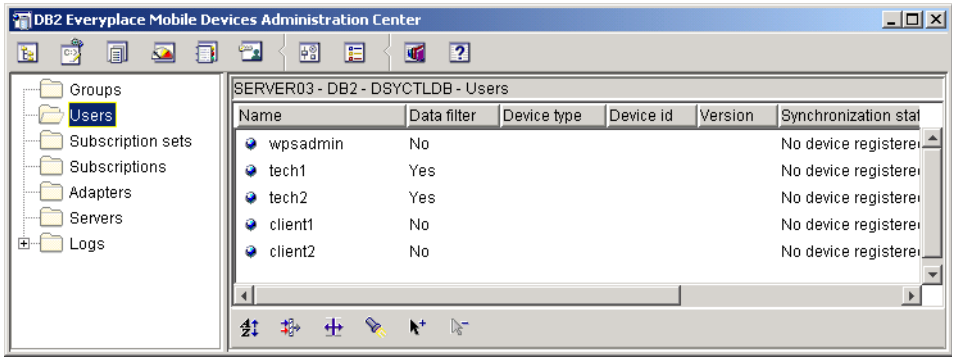


Figure 8-17 Users

The database is configured for remote synchronization for pervasive devices.

8.6 Developing Java Application for J2ME

This section shows the concepts, architecture and development of a J2ME pervasive application. For design considerations and details on this technology, refer to 7.7, “Embedded mobile client applications” on page 151.

J2ME applications are not exercised in this book’s sample application, but they can provide the same functionality as the previous fat client, although in this case, it is also available for smartphones besides the PDAs.

8.6.1 Developing a Midlet

In this part, we explore CDLC and MIDP API in more detail. We choose this profile because it has became more popular than others.

A Midlet is an abstract class that is subclassed to form the basis of the application. The Midlet class resides in the package javax.microedition.midlet.

Example 8-10 is quite simple; it shows the Midlet and does nothing other than putting a short text message on the screen.

Example 8-10 Midlet class

```
package com.itso.test;

import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
```

```
public class DemoMidlet extends MIDlet {
    private TextBox textbox;

    public DemoMidlet() {
        textbox = new TextBox("", "UUUUUUAaaaa", 20, 0);
    }

    protected void startApp() throws MIDletStateChangeException {
        Display.getDisplay(this).setCurrent(textbox);
    }

    protected void pauseApp() {
    }

    protected void destroyApp(boolean flag) throws MIDletStateChangeException {
    }
}
```

When a device receives a message to start a Midlet , the Midlet is instantiated and the application management service on the device calls the startApp() method. At this point, our application takes over control and performs any initialization that may be required.

The startApp() method can be called a number of times during the lifecycle of a Midlet. It can be placed in a pause state as a result of a call to the pauseApp when the user, or the device needs to perform some action such as writing content.

The user can choose to close the application, or for some reason the system can also require the application to close. The method destroyApp is called.

The next step is to compile the code and create the deployment files. This task can be automated by ANT or you can use WebSphere Studio Device Developer to do it for you.

Example 8-11 shows an ANT script for generating the deployable code.

Example 8-11 Steps to compile the Midlet

```
<?xml version="1.0" encoding="UTF-8"?>
<project default="all" name="HelloMidletProject">
    <target depends="updatejad palm68k/HelloMidletSuiteName" name="jxe2prc
palm68k/HelloMidletSuiteName">
        <jxe2prc applicationName="DemoMidlet" apptype="midp"
            compress="true" creatorId="0001" midp="true"
            outputFile="palm68k/HelloMidletSuiteName.prc"
            platform="palm68k" scale="" translate="true">
            <efileset dir="palm68k">
```

```
                <include name="HelloMidletSuiteName.jxe"/>
            </efileset>
        </jxe2prc>
    </target>
    <target depends="includemidlets palm68k/HelloMidletSuiteName"
name="smartlink palm68k/HelloMidletSuiteName">
        <jxelink optionsFile="palm68k/HelloMidletSuiteName.jxeLinkOptions"/>
    </target>
    <target name="includemidlets palm68k/HelloMidletSuiteName">
        <includemidlets jadFile="HelloMidletSuiteName.jad"/>
    </target>
    <target depends="smartlink palm68k/HelloMidletSuiteName" name="updatejad
palm68k/HelloMidletSuiteName">
        <updatejad inputJadFile="HelloMidletSuiteName.jad"
            jarFile="palm68k/HelloMidletSuiteName.jxe"
outputJadFile="palm68k/HelloMidletSuiteName.jad"/>
    </target>
</project>
```

This script compiles the code for PalmOS.

The WebSphere Application Device Developer has tools to emulate this program, or you can download a simulator for the device to test this program.

8.7 Testing your pervasive application

This section provides information about testing the sample application. We can do this job in two different ways.

In the first case, we are using the cell phone emulator with Java support. After you finish with the code and deployment descriptor, you can test your application with WebSphere Studio Device Developer Debug. To do this, press **F11** inside the IDE environment and this tool will show the emulator.

If you follow all the steps properly, you will see a window as shown in Figure 8-18 on page 192.

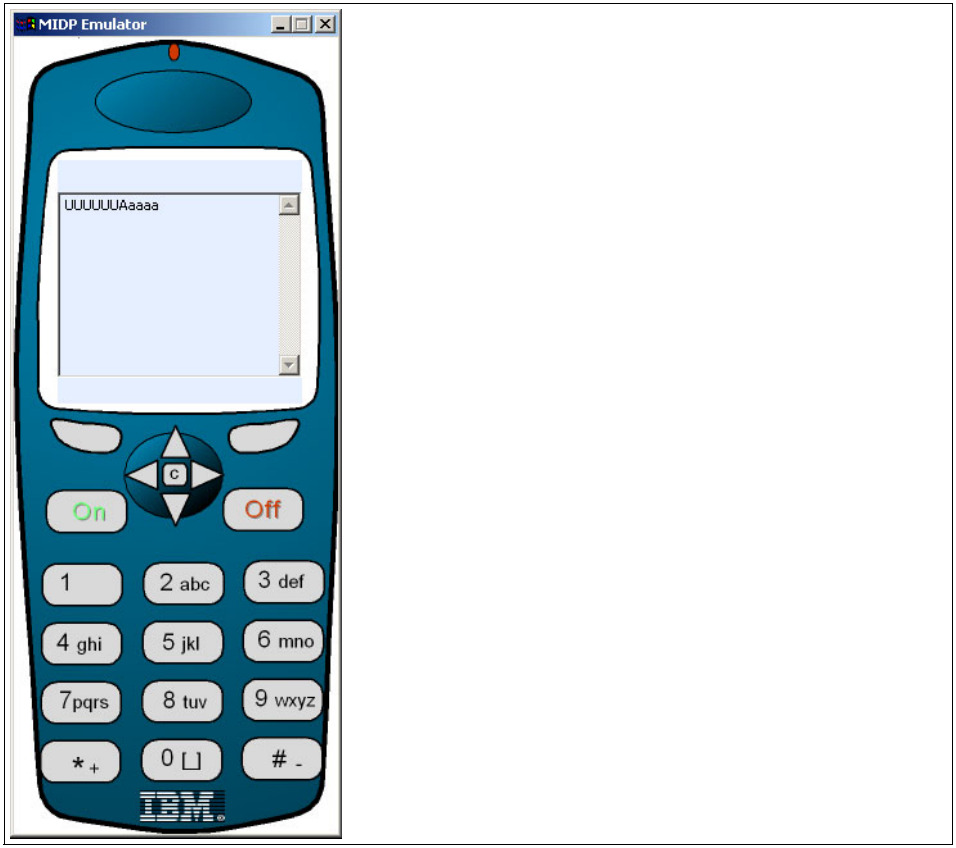


Figure 8-18 WAP device simulator

You can also run the same application on a Palm device. This is a test outside of the device developer. The deployment descriptor generates the file to import in Palm OS.

You can download a simulator from the PalmOS Web site (`http://www.palmos.com`) then you have to install the Java Runtime Environment for J2ME to run this application.

In order to install the Java Runtime Environment, you should open the Palm simulator; right-click the Palm screen, choose the **Install application/database** option and select the following files:

- <WSDD_home>\wsdd5.0\ive\runtimes\palmos\68k\ive\bin\cldc20.prc
- <WSDD_Home>\wsdd5.0\ive\runtimes\palmos\68k\ive\bin\j9_vm_bundle.prc

- <WSDD_home>\wsdd5.0\ive\runtimes\palmos\68k\ive\bin\j9pref.prc
- <WSDD_home>\wsdd5.0\ive\runtimes\palmos\68k\ive\bin\midp20.prc

Important: WSDD_home is the folder where you have WebSphere Studio Device Developer installed.

After this, you can install the sample application in the same way that you have installed the JVM. For this device, you should install the .prc file. The deployment descriptor in this chapter generates the .prc file in <WSDD_home>\workspace\HelloMidletProject\palm68k\HelloMidletSuiteName.prc.

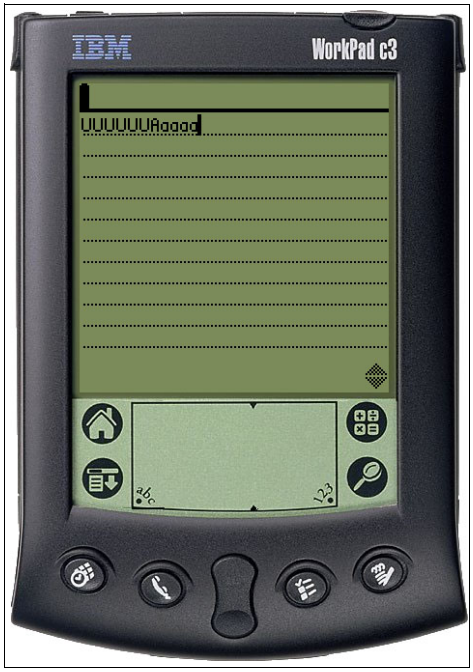


Figure 8-19 Palm device simulator

8.8 Everyplace Client

Everyplace Client is a client application for Personal Digital Assistants (PDAs) which provides a common interface that supports information updating, security, device management, offline Portal pages, offline Domino Applications and database synchronization. The Everyplace Client features the following products:

- Domino applications, which provide Lotus Notes functionality for the handheld device.
- e-mail and PIMs, which provide the capability to synchronize with e-mail and PIM data with Lotus Domino and Microsoft Exchange servers.
- Database synchronization, which supports DB2 Everyplace as the database synchronization engine. It enables the synchronization with DB2 and JDBC compliant database servers.
- Offline Portal pages, which provide the ability to view Portal content and to handle Form operation offline, using the capabilities of Web-based services. This feature is available only in PocketPC devices.
- Device Manager, which can be used to identify, configure, inventory, and distribute software to supported PDA devices.
- Everyplace Client: the Everyplace Client interface provides a simple and unified operation and launching interface for packaged applications. This interface is available only for PocketPC devices.
- Secure HTTP proxy: this supports SSL connection and basic authentication, and contributes to support the offline browsing of the static portal server, the unified user interface and configuration for all Everyplace client applications.
- Sametime Connect: Sametime Connect is an easy-to-use application that provides complete awareness and conversation features. It lets people find other members online and create personalized lists of team members and colleagues. Features for managing user privacy are also included so a member can control their own online presence. Users can also list their presence as "active", "away", or "do not disturb". Once a team member is aware of who is online, immediate communication is easy. A single mouse click lets them send an instant message to anyone. The Sametime Connect client for Palm OS is shown in Figure 8-20 on page 195.

Figure 8-20 The Sametime Connect for Palm OS

8.9 Notification Services

Everyplace Intelligent Notification Services delivers messages to users of pervasive devices based on the users' preferences and subscriptions. For example, subscribers can tell Everyplace Intelligent Notification Services to notify them when news articles with "pervasive computing" in the headline are published by a content provider. Subscribers can also specify which devices to send a notification to based on the urgency of the message. For example, if the message is marked FYI, send it via e-mail, if the message is marked urgent, send it via Sametime instant messaging.

In the sample application, we did not implement notification. In the sample scenario, the notification takes care of the urgent customer requests, so the technician gets the request immediately on the pervasive device while he/she is working in the field.

The supported notification mechanisms by Everyplace Intelligent Notification Services are:

- Lotus Sametime
- Message Center portlet
- Short Message Service (SMS); this requires WebSphere Everyplace Wireless Access
- Simple Mail Transfer Protocol (SMTP)

In order to send notification through another type of messaging service, you must implement a custom gateway adapter, which can be developed by using the API provided.

8.9.1 Configuring Notification Services

Notification Services in WebSphere Everyplace Access can be configured through the Portal administration.

1. Open a Web browser, go to the location `http://<WEA_server>/wps/portal`.
2. Log in as the portal administrator (wpsadmin/wpsadmin).
3. Select **Notification Services** from the menu. You will get the following page for administering the notification services.

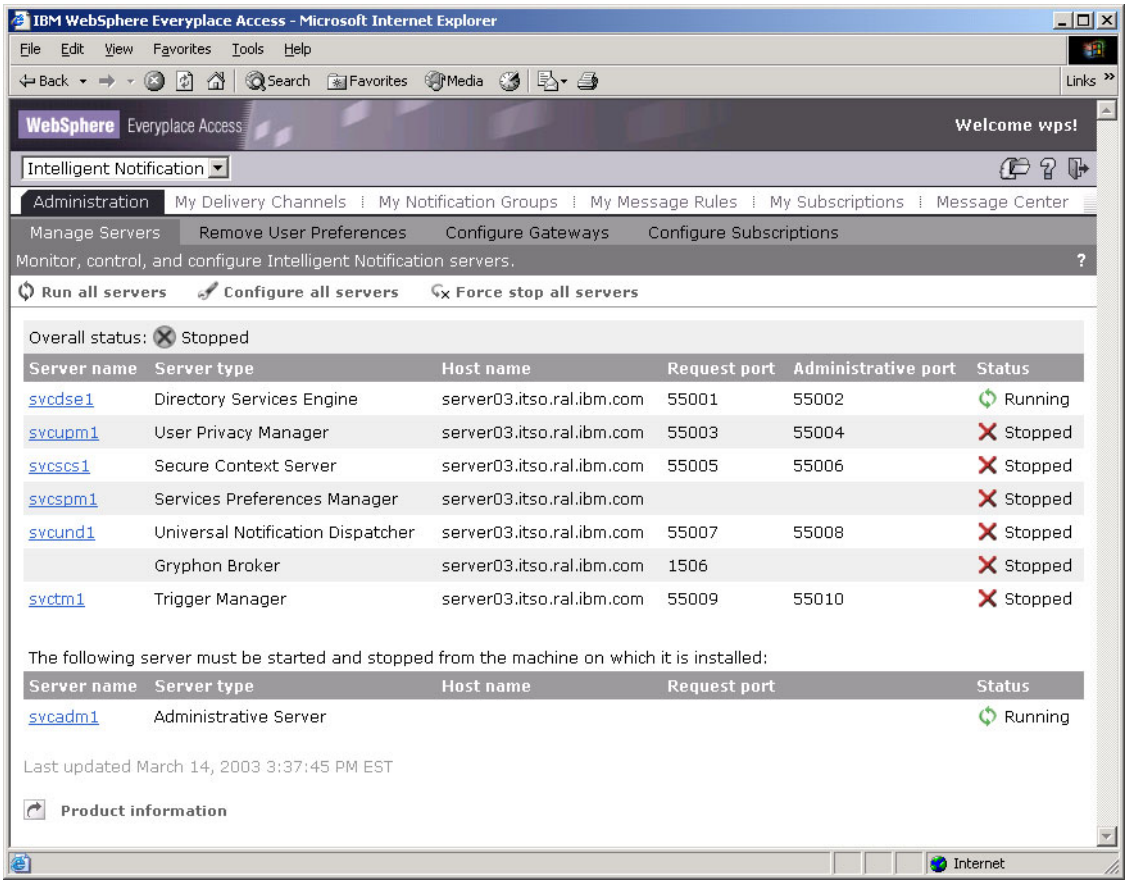


Figure 8-21 Notification Services in WebSphere Portal

In our sample scenario, the Notification Service sends messages to the Technician on the field on the PDA. Messages can be filtered on the application level, so the Technician only gets notification when the reported defect reaches a certain severity level.

Notification can use the following technologies:

- Instant messaging; this requires a special client application for messaging.
- e-mail, message sending via SMTP. This also requires a client application to receive messages.
- SMS is usually a built-in function in GSM devices and the phone takes care of the message handling.
- WAP push is part of the WAP specification, and it is implemented in the mobile device.

For more information about notification, refer to the product documentation.

9

Security

The pervasive world was envisioned to make the user's experience easier and extend the capabilities and reach of e-business applications to everybody, everywhere and at every time. As an unknown and unexplored solution, pervasiveness brings with all these announced advantages new issues for critical areas such as security.

This chapter discusses the common security issues to a Pervasive Portal solution using WebSphere Everyplace Access.

This chapter will not present basic security concepts; some of this information can be found in the IBM Redbook *Mobile Applications with IBM WebSphere Everyplace Access Design and Development*, SG24-6259-00.

© Copyright IBM Corp. 2003. All rights reserved.

9.1 Security for a Pervasive Portal solution

In order to better understand the security issues on a Pervasive Portal solution, the same operational model diagram will be used and the security issues for some of the components of the solution will be detailed.

The components explored are:

- Boundary components - this includes routers, firewalls and architecture recommendations for security layers.
- WebSphere Everyplace Connection Manager
- WebSphere Edge Server
- WebSphere Everyplace Access and its components, such as:
  - Everyplace Client
  - Device Manager
  - Everyplace Synchronization Server
- Tivoli Access Manager and Single Sign-On

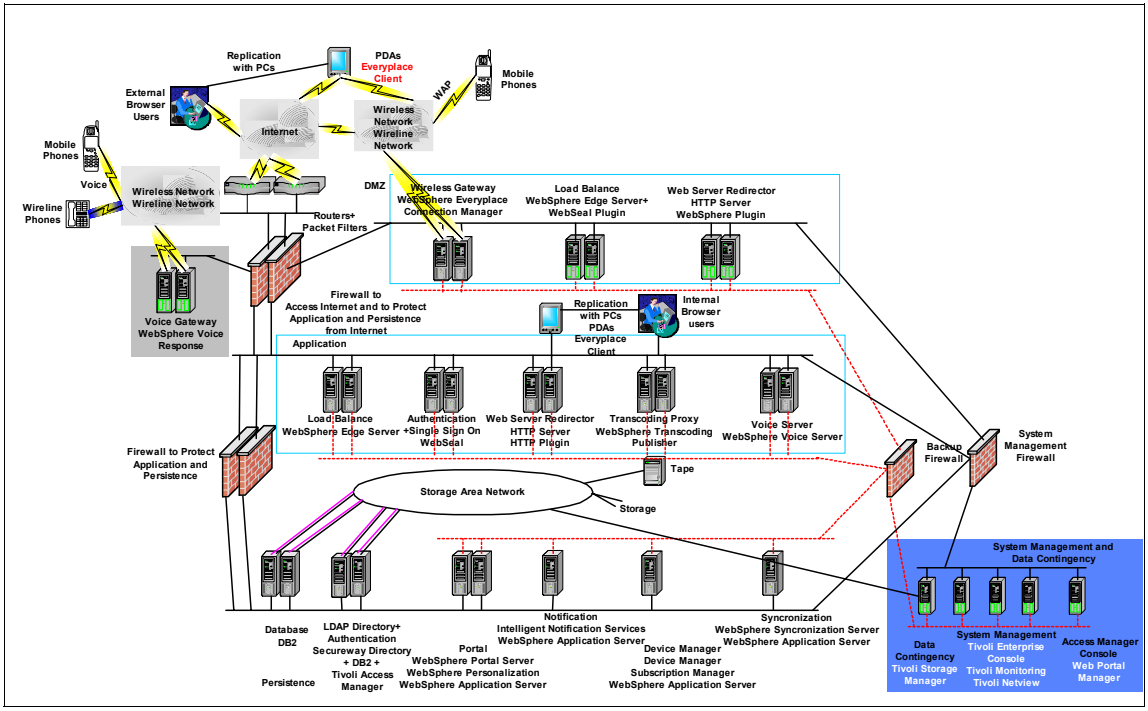


Figure 9-1 Security issues for the components of a Pervasive Portal solution

Note: Because of the extensive details of the previous diagram, it is difficult to read some parts on a printout. If you would like to read the details, please use the original PDF document and magnify the area you want to see.

9.1.1 Boundary components

The boundary components include the firewalls and routers that implement the “physical” security separating the solution layers and implementing the security policy. Each security layer created uses different principles, compatible with the security level required. Even the security policy of each layer will be different and specific to the relative functionality and operational level.

We will use a generic diagram block representing a multi-tier production environment to explain the characteristics of each security layer. It is considered that the solution non-functional requirements include high availability. For more detailed information about high availability and performance, refer to Chapter 11, “Performance and availability” on page 239.

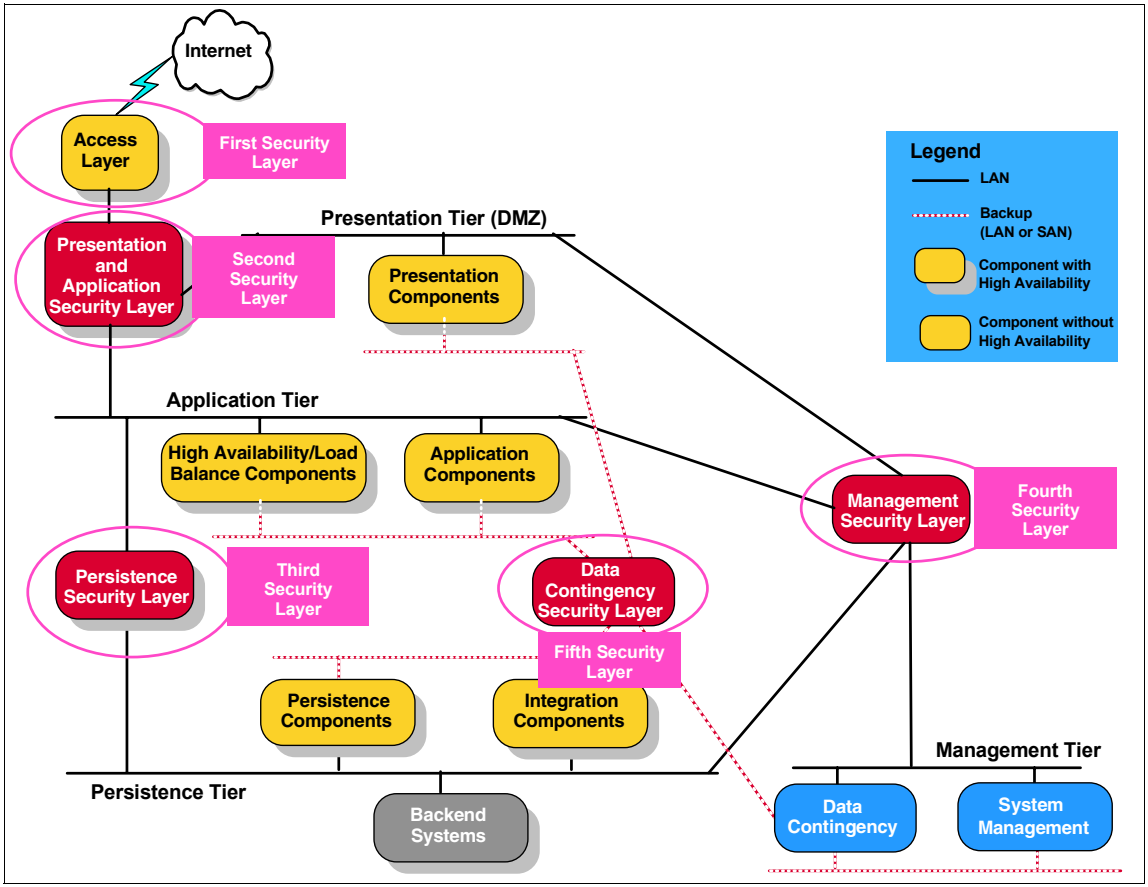


Figure 9-2 Security layers for a generic n-tier solution

First security layer

This layer includes routers connecting to Internet and protecting in a basic level through the use of packet filters. Besides creating an initial barrier, it avoids some improper accesses that could compromise the performance of the LAN. That layer increases the security level, but by itself it is not enough to protect the entire solution.

Second security layer

This security level includes two firewalls separating the DMZ (presentation layer for Internet users) from the application tier. That layer will have the following characteristics:

- A second level of security to protect against malicious Internet users.

- This layer does not allow Internet users to access the application layer directly. Using this model, every request from external users will take place in the Presentation components (HTTP Server in the Web Server Redirector node) on the demilitarized zone (DMZ) and the requests will be redirected to the application layer. It is important to create rules so that only the presentation components can access the Application components.

It is recommended that the two firewalls be working at high availability, guaranteeing service availability, even in case of problems with one of the firewalls.

Third security layer

This security level includes two firewalls separating the application tier from the critical components and persistence tier that the end user does not need to access directly.

Every information request performed by the user takes place on the presentation tier that reaches the persistence tier by the application tier. Rules will be created in order to define which components of the application tier should access the persistence tier. It is assumed that access will be denied to any end user trying to access the Persistence components.

It is recommended that the two firewalls be working at high availability, guaranteeing service availability, even in case of problems with one of the firewalls.

Fourth security layer

A separate network was created for the site management. This layer corresponds to the security level implemented to separate the management tier from the presentation (DMZ), application and persistence tiers. This is possible through the use of one firewall. The reason for the existence of this layer lies with the different rules, and mostly because of the right policy that should be implemented. Implementing the management security policy in the second and third layer would represent a security exposure. Keeping in mind that every request performed on the application tier goes through the second and third security layer, this exposure could reduce the level of security and cause a performance degradation. The firewall prevents a break-in attempt on the persistence tier through the management tier.

Fifth security layer

Separated networks were created for the site's back-up for performance reasons (it does not use up the bandwidth from the presentation, application and persistence tier). Those were not connected because this would create a security leak, and a possible break-in on the persistence tier (critical data) through the

back-up network. Therefore, the use of a firewall connecting the back-up network of the various layers is strongly recommended.

9.2 WebSphere Everyplace Connection Manager

WebSphere Everyplace Connection Manager is composed of a Messaging Gateway (SMS Gateway and Push Proxy Gateway), a WAP Gateway, Remote Access enabler (VPN-IP tunneling), a wireless client and an administration interface (Wireless Gatekeeper).

Note: WebSphere Everyplace Connection Manager is not part of the WebSphere Everyplace Access 4.2 product; it is a separate product from IBM.

Following are some security solutions supported by each one of these components. Figure 9-3 also shows in a generic way the several network connections that WebSphere Everyplace Connection Manager supports.

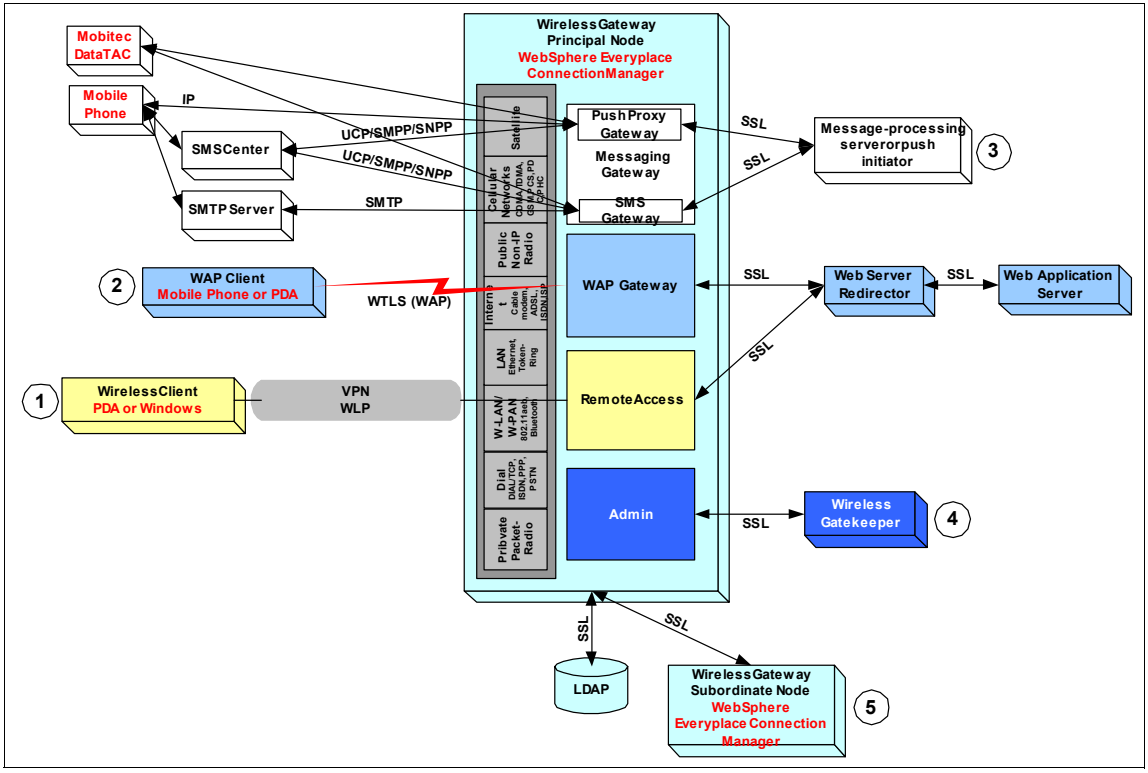


Figure 9-3 WebSphere Everyplace Connection Manager security

1. Wireless Client running on a PDA or a Windows accessing the WebSphere Everyplace Connection Manager through an IP tunnel solution.

   This function provides a secure data communication between a client using a PDA or Windows over a public network and a secure network.

   Considering that the user wants to access a Web Application that is behind the Wireless Gateway, the following considerations should be taken into account:

   – A secure IP tunnel will be implemented between the Wireless client and the Wireless Gateway
   – All the traffic inside the tunnel is protected
   – If the Web Application supports SSL, it will establish an HTTPS connection between the Web browser on the client and the Web server, and the content will be transported through the IP tunnel.

   Authentication is a prerequisite for the communication link between the Wireless Gateway and Wireless Client to be encrypted. The Wireless Gateway uses a modified Point-to-Point Protocol (PPP) called *wireless optimized link protocol* (WLP) to authenticate the connection between itself and the Wireless Clients. WLP provides data compression, IP header reduction, selective packet filtering, and TCP retransmission optimizations that make data communications over these networks more efficient and cost-effective.

   The Wireless Gateway supports:

   – Single-party key distribution protocol. The Wireless Client is authenticated to the Wireless Gateway.
   – Two-party key distribution protocol. The Wireless Gateway and the Wireless Clients authenticate each other.
   – Diffie-Hellman key agreement algorithm. Both the Wireless Gateway and the Wireless Client are given the means to compute the same key. Note that this choice does not perform authentication, only encryption.

   In terms of encryption, the WebSphere Everyplace Connection Manager and Wireless Client support the advanced encryption standard (AES), digital encryption standard (DES), RC5 and Triple-DES.

2. Secure connection between a WAP client and a WAP Gateway

   This functionality provides secure Web access to WAP clients such as mobile phones and PDAs using a WAP browser.

   The solution used to provide encryption between the WAP client and the Web Application is as follows:

– Wireless transport layer security (WTLS) protocol using public-key algorithms to manage key agreement and symmetric key algorithms to encrypt the communication link between the WAP clients and the WAP gateway.
– SSL protocol to encrypt the communication link between the WAP gateway and the Web Application.

This solution has two secure connections: WTLS and SSL. The encrypted data from WTLS has to be decrypted in order to be re-encrypted under SSL, and vice-versa. The reason is that SSL is defined at the transport level (TCP), and that transport level protocol is different from the one in WAP. The consequence of this is that at one moment the data may pass through the gateway, and in some contexts this might create a security issue.

For solutions on these issue, refer to the IBM Redbook *Mobile Applications with IBM WebSphere Everyplace Access Design and Development* - SG24-6259-00.

The WAP Gateway supports several methods of authentication to validate a WAP client.

– Native PPP connections: the Wireless Gateway accepts connections from dial-in devices that support the Point-to-Point Protocol (PPP) protocol. When users are first configured by the administrator, they are assigned a password that is stored securely by the gateway and is communicated to the person using the ID (by phone, secure e-mail, or postal mail) so both parties know the password.

  Using either the PAP (password authentication protocol) or CHAP (challenge handshake authentication protocol) authentication protocols, users negotiate authentication with the Wireless Gateway. The Wireless Gateway validates the user's credentials using a persistent user storage database in an LDAP-compliant directory server, or using an external authentication server that uses the RADIUS protocol.

  The authentication for establishing a PPP session to Wireless Gateway serves as the authentication mechanism for WAP proxy transactions.
– Device resolver: though not a method of authentication, the device resolver serves to uniquely identify a WAP client. Authentication of the WAP client takes place elsewhere in the environment, usually at the server that provides network access.

  In a typical service-provider environment, the Network Access Server (NAS) has direct access to the provider's wireless network infrastructure, where unique information about a device, such as its phone number, is available. The NAS uses the RADIUS protocol to inform the WAP proxy of that unique identity. The WAP proxy uses the device resolver to match this

information to WAP requests and check that the device has previously been authenticated.

- Account resolver: the account resolver is similar to the device resolver function, but instead of using the RADIUS protocol to get information about the WAP client, the WAP proxy uses HTTP to query the NAS for information about the device.
- HTTP challenge: the Wireless Gateway can require that user credentials exist in each WAP transaction header.

  When user IDs are first configured in the system by the administrator, they are assigned a password that is stored securely on the Wireless Gateway and is securely communicated to the person using the ID (by phone, secure e-mail, or postal mail) so that both parties know the password.

  Any transaction that does not contain a user ID and password is discarded, and the WAP client browser is sent a reply with a status code of either Proxy-Authenticate or WWW-Authenticate, depending on the environment. The WAP client browser prompts users to enter their user ID and password, and then resubmit the transaction with the credentials included. These credentials, along with the network address of the WAP client, are used to authenticate the user. The Wireless Gateway can validate the user's credentials using a persistent user storage database in an LDAP-compliant directory server, or using an external authentication server, via the RADIUS protocol.

The encryption between the WAP Gateway and the Web Application is supported using SSL version 2 or 3, and the authentication is provided by x.509 certificates.

3. Secure connection between the Message Gateway and message-processing server or push initiator.

   This functionality provides encryption and authentication between a message-processing server or push initiator and the Messaging Gateway.

   The encryption between the Message Gateway and the message-processing server or push initiator is supported using SSL version 2 or 3, and the authentication is provided by x.509 certificates.

   In the WebSphere Everyplace Access package, the message-processing server and the push initiator function is provided by the Intelligent Notification Services. However, the actual version of the Intelligent Notification Services does not support SSL at this moment, but it will be supported in a future release.

   The messages flowing between the messaging gateway and the client devices are not encrypted.

4. Secure connection between the Wireless Gatekeeper administration console and the Wireless Gateway.

   The Wireless Gateway uses its access manager subsystem to communicate with the Wireless Gatekeeper administration console. The access manager is a process (wgmgrd) that manages interactions between Wireless Gatekeeper, the persistent data store, and Wireless Gateways.The encryption supported is SSL version 2 or 3, and the authentication is provided by X.509 certificates. In addition to X.509 certificate validation, each administrator ID has an associated password. The Wireless Gateway can validate the administrator's credentials against a persistent user storage database in an LDAP-compliant directory server.

5. Inter-gateway cryptography

   For load balancing and high availability performance, several Wireless Gateways can work together in a clustered configuration. The cluster manager subsystem controls the communications among the nodes. The encryption between the Wireless Gateways is supported by using SSL version 2 or 3, and the authentication is provided by X.509 certificates.

   For more information about the load balancing and high availability solution for WebSphere Everyplace Connection Manager, refer to 11.4, “Applying to a Pervasive Portal solution” on page 249.

9.3 WebSphere Edge Server

WebSphere Edge Server acts as a front end to a cluster of Web application servers, serving static and limited dynamic content from its cache and performing load balancing. It supports SSL-encrypted requests and, if needed, exploits cryptographic hardware accelerators to relieve application servers of CPU-intensive public key infrastructure (PKI) authentication chores.

WebSphere Edge Server addresses the authentication and authorization requirements by the use of the Tivoli Access Manager for Edge Server plug-in. It identifies users to all the site’s back-end components, and controls the access to cached static or slowly changing dynamic content, as well as to back-end highly dynamic or interactive content. The Access Manager plug-in can identify users via X.509 certificates, forms-based login or the traditional ID-password basic authentication challenge protected by an SSL-encrypted session. It integrates with access policy administration tools such as Policy Server and LDAP.

9.4 WebSphere Everyplace Access and its components

WebSphere Everyplace Access provides the core components to create a Pervasive Portal solution. Considering that it is composed of several different products and based on WebSphere Application Server, the product architecture is mostly based on a centralized security solution, facilitating the management and development of new components. Refer to 5.2, “Products” on page 60 for more information about the WebSphere Everyplace Access components.

The authentication and Single Sign-On features are provided by WebSphere Application Server, WebSphere Portal Server and Tivoli Access Manager. Refer to 9.5.1, “Tivoli Access Manager and Single Sign-On” on page 210 to obtain more information about authentication and Single Sign-On methods.

In order to protect data transferred between servers and mobile clients, security must be enabled on both the HTTP server, the application server, and the mobile client.

WebSphere Everyplace Access supports Single Sign-On using LTPA (Lightweight Third Party Authentication) which is part of WebSphere. It also supports SSL for securing the connections and communication for the HTTP protocol.

Note: WebSphere Edge Server is not part of WebSphere Everyplace Access V4.2; it is a separate IBM product.

Everyplace Client

Everyplace Client includes an authentication proxy that handles translations between the client and a secure server. This proxy is transparent to the servers, therefore each client component must be configured to use the proxy. The proxy then acts as an agent between the client and servers, providing the necessary authentication information to the server.

Device Manager

In order to provide encryption between the Device Manager and the mobile devices, these are the available solutions:

- Palm OS does not supply any SSL functions, so the device agent provides the elements needed to implement secure connections. It implements 128-bit encryption as well as data integrity checking of communications between the device agent and the plug-in.
- Device Manager relies on the SSL features supplied with Microsoft Windows CE. It implements 128-bit encryption of communications between the device agent and the plug-in.

Everyplace Synchronization Server

The Synchronization Server uses WebSphere Application Server basic authentication for user authentication. It also supports MD5 and basic authentication at the SyncML layer and performs user authentication before permitting access to back-end databases.

The Lotus Domino Adapter and Microsoft Exchange Adapter require authentication to synchronize with the back-end databases. For caching back-end servers, the adapters need write, edit, create and delete authority in order to write updated PIM data to the back-end servers. Adapter authentication is configured using the Lotus Domino and Microsoft Exchange Adapter portlets.

9.5 Tivoli products for security

The following Tivoli products are not part of the WebSphere Everyplace Access product; they are optional components of a solution and they are widely used to implement security.

9.5.1 Tivoli Access Manager and Single Sign-On

From the user's perspective, Single Sign-On (SSO) is the ability to move between applications without being prompted for a user ID and password (or certificate) when moving from one application or datasource to another during the same user session. The solution that provides authentication, authorization and SSO is the Tivoli Access Manager.

There are a number of commonly encountered business requirements that justify the use of an authentication and authorization solution like Tivoli Access Manager:

- Different back-end and Web content hosting systems require users to authenticate multiple times, which generates a negative user experience. In order to improve customer satisfaction, a method for single-user authentication has to be implemented.
- Web security policies must be consistently applied across the business. Without a common security infrastructure, Web content and application security policies tend to be applied differently by various parts of the business.
- The costs of Web security management must be predictable.
- Threats of inadvertent security compromises or hacker attacks represent significant risks to business operations and company goodwill.

Access Manager's base functions are provided through a set of core components and various management components.

Core components

Access Manager is fundamentally based on two components:

- An user registry that is based on a LDAP directory and provides:
  - A database of the user identities
  - A representation of groups in Access Manager (roles) that may be associated with users
- An Authorization Service, consisting of an authorization database and an authorization engine. This is the foundation and it is responsible for permitting or denying access to protected objects (resources) based on the user's credentials and the access controls placed on the objects.

  The authorization database is a special database containing a virtual representation of the resources that it protects and the definition of the security mechanisms. Some of these security mechanisms are:

  - Access control list (ACL) policy templates - ACLs are special Access Manager objects that define policies identifying user types that can be considered for access, and specify permitted operations. In the Access Manager model, ACLs are defined separately from and then attached to one or more protected objects. Access Manager uses an inheritance model in which an ACL attached to a protected object applies to all other objects below it in the tree until another ACL is encountered.
  - Protected object policy (POP) templates - A POP specifies additional conditions governing the access to the protected object, such as privacy, integrity, auditing, and time-of-day access. POPs are attached to protected objects in the same manner as ACLs.

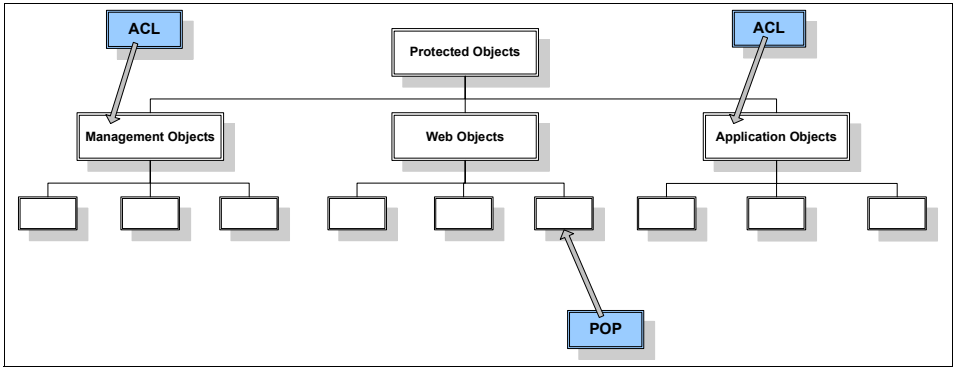


Figure 9-4 Relationship between protected objects, ACLs and POPs

Management components

The Access Manager environment requires certain basic capabilities for administrative control of its functions. Management facilities are provided through the following base components:

- The Policy Server, which supports the management of the authorization database and its distribution to Authorization Services. The purpose of the Policy Server is to maintain the master authorization database that contains the protected object space with the access control information (ACLs and POPs).
- The pdadmin utility, which provides a command line capability for performing administrative functions, such as adding users or groups.
- The Web Portal Manager, which provides a Web browser based capability for performing most of the same functions provided by the pdadmin utility.

WebSEAL

WebSEAL is a high-performance, multi-threaded reverse proxy, front-ending back-end Web service that applies a security policy to a protected object space. WebSEAL can provide Single Sign-On and incorporate back-end Web application server resources into its security policy. Being implemented on an HTTP server foundation, it listens to the typical HTTP and HTTPS ports.

In order to apply a security policy to any resource located in a back-end server, it has to be defined in WebSEAL a junction with this resource.

For example, suppose a junction on the WebSEAL host www.server1.com is defined such that a request for any URL specifying the path /content/xyz is to be proxied to the back-end Web server w3.server2.com. /content/xyz is the junction point and from the perspective of the browser, the request is processed by www.server1.com. To support this, WebSEAL performs various transformations of the response sent to the browser to assure that the back-end server names are not exposed (this is called virtualization of the back-end Web server and improves security).

Figure 9-5 on page 213 shows the components of the Tivoli Access Manager, the relationship between them and Single Sign-On solutions.

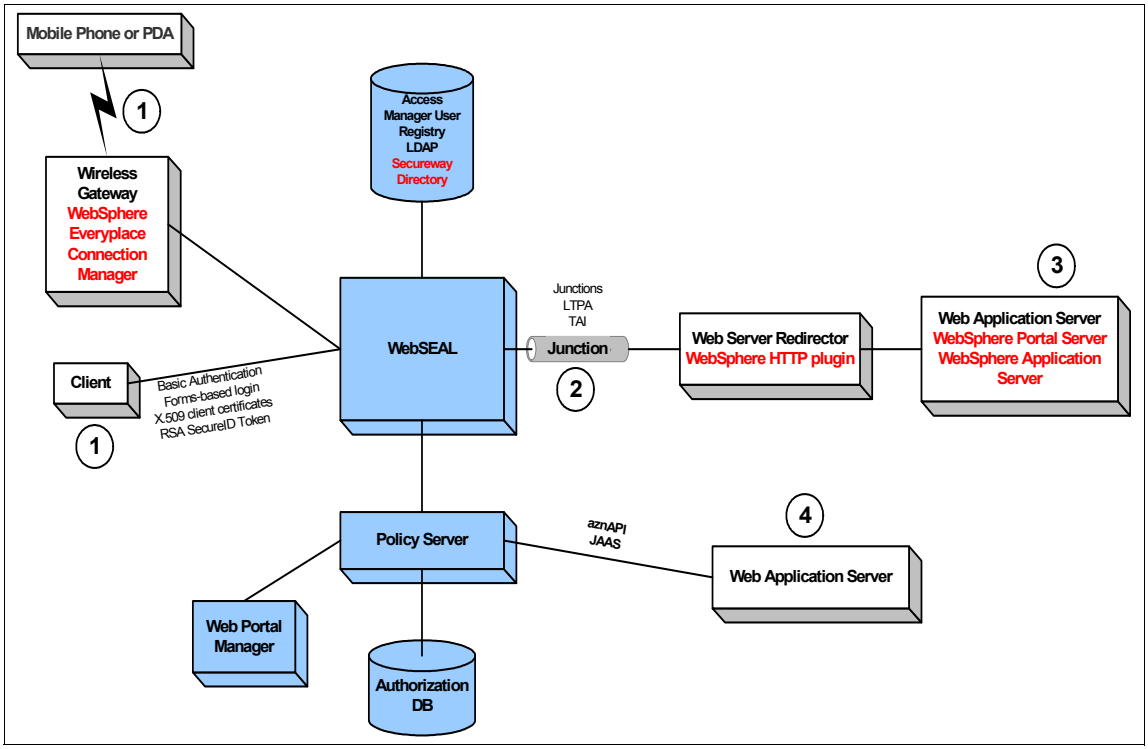


Figure 9-5 Tivoli Access Manager and Single Sign-On architecture

1. User authentication mechanisms

   WebSEAL supports a number of client authentication mechanisms to protect access to a Web environment. WebSEAL can communicate with the clients with both encrypted (SSL) and unencrypted (TCP) protocols. The supported encryption types are SSL V1, SSL V2, SSL V3, and TLS V1. It also supports SSL hardware acceleration that can minimize the CPU impact of SSL and improve the overall performance of the system. Some of these mechanisms are:

   – Basic authentication with user ID/password - Basic authentication (BA) is part of the HTTP standard and defines a standardized way in which user ID/password information is passed to a Web server.

   – Forms-based login with user ID/password - The alternative to using basic authentication is the forms-based login. Rather than sending a basic authentication challenge in response to a client request, WebSEAL responds with a sign-in form in HTML format.

Another benefit to using the forms-based login process is that you can enforce a time-based logout for authenticated sessions. The time values can be customized in the WebSEAL configuration files.

- Authentication with X.509 client certificates - In response to a certificate request from WebSEAL, as part of the SSL V3 tunnel negotiation, the browser prompts the user to select a certificate from the local certificate store or smartcard.
- Authentication with RSA SecurID token - Access Manager supports authentication of clients using user name/token pass code information from an RSA SecurID token authenticator (TAR), a physical device that stores and dynamically generates a piece of authentication data (a token).
- Authentication integration with WebSphere Everyplace Connection Manager- Access Manager provides an authentication mechanism for clients using a Wireless Gateway as WebSphere Everyplace Connection Manager; it can receive from it an “authenticated ID” and does not need to re-authenticate the user.

2. WebSEAL authentication with junctioned servers and delegation mechanisms

   WebSEAL can authenticate itself to a junctioned server using either server certificates, forms-based authentication, or HTTP basic authentication. When using an SSL communication channel for this junction, WebSEAL and the junctioned server can also mutually authenticate one another. This is very important in order to establish the trust relationships between WebSEAL and back-end application servers.

   WebSEAL can communicate with the back-end servers with both encrypted (SSL) and unencrypted (TCP) protocols. The supported encryption types are SSL V1, SSL V2, SSL V3, and TLS V1. It also supports SSL hardware acceleration that can minimize the CPU impact of SSL and improve the overall performance of the system.

   After a user has been authenticated by WebSEAL and an authorization decision has been made, WebSEAL has to forward the user’s request to a back-end Web application server. The mechanisms to forward that information and also provide a Single Sign-On solution are:

   - WebSEAL junctions

     WebSEAL performs authentication while the back-end server handles its own authorization needs. It is recommended that you use a shared user registry to eliminate duplication of data.

     WebSEAL is configured to talk to a Web or application server behind it with a smart junction, which is a portion of a request URL that directs WebSEAL to forward the remainder of the URL path to the back-end Web or application server. A smart junction:

- Allows the integration of multiple servers (WebSEAL or third-party) into one unified Web space.
- Extends Access Manager security to third-party Web servers. WebSEAL provides authentication and authorization checking and enforcement services.
- Allows growth of Web clusters with load balancing and fault-tolerant HTTP and HTTPS junctions.
- Provides both TCP and SSL connections to back-end servers.

Smart junctions can be configured to let WebSEAL modify the request header contents and basic authentication credentials before passing the request to a subsequent Web server. That is, WebSEAL can determine how the user ID and password are forwarded on.

– Web Trust Association Interceptor (TAI)

This Single Sign-On method is supported by WebSphere and implies that WebSphere's security application recognizes and processes HTTP requests received from WebSEAL. WebSphere and WebSEAL engage in a contract in which the former will give its full trust to the latter, which means that WebSEAL will apply its authentication policies on every Web request that is dispatched to WebSphere.

This trust is validated by the interceptors that reside in the WebSphere environment for every request received. The method of validation is agreed upon by WebSEAL and the interceptor.

For detailed information on these options, refer to the IBM Redbook *Enterprise Business Portals with IBM Tivoli Access Manager*, SG24-6556-00.

– LTPA authentication

This is a token-based lightweight third-party authentication mechanism (LTPA) that is supported by WebSphere Application Server and Lotus Domino.

Considering an environment without WebSEAL, the token is created by WebSphere Application Server when a user authenticates, and is stored in a cookie that is passed back to the browser with the HTTP response. On subsequent accesses by the user, the token is extracted from the cookie and used to authenticate the user. In this way, the user enters their user ID and password only once, and the LTPA token identifies them after that. The drawbacks of this approach are twofold:

- LTPA is supported only by WebSphere Application Server and Domino, and has not received industry-wide acceptance. Kerberos is a more widely used third-party authentication mechanism.

- The user's credentials (in the form of the token) are passed all the way back to the browser and then back to the server with each request. This creates some vulnerability considering that the cookie is stored in the user machine and particularly if SSL is not used between the browser and Web server.

With WebSEAL, the token never passes to the browser, avoiding a potential security vulnerability.

When a user makes a request for a WebSphere resource, the user must first authenticate to WebSEAL. After successful authentication, WebSEAL generates an LTPA cookie on behalf of the user. The LTPA cookie, which serves as an authentication token for WebSphere, contains the user identity, key and token data, buffer length, and expiration information. This information is encrypted using a password-protected secret key shared between WebSEAL and the WebSphere server.

WebSEAL inserts the cookie in the HTTP header of the request that is sent across the junction to WebSphere. The back-end WebSphere server receives the request, decrypts the cookie, and authenticates the user based on the identity information supplied in the cookie.

To improve performance, WebSEAL can store the LTPA cookie in a cache and use the cached LTPA cookie for subsequent requests during the same user session. It can be configured with lifetime timeout and idle (inactivity) timeout values for the cached cookie.

The trust association interceptor mechanism (TAI) is preferred to LTPA, because LTPA support is currently limited to WebSphere Application Server and Domino, and the TAI is faster considering that a token does not have to be encrypted and decrypted each time.

3. Single sign-on to WebSphere Portal Server

In order to provide Single Sign-On, the WebSphere Portal Server trust of WebSEAL is maintained through the junction between WebSEAL and the Web server. There are essentially two options for achieving Single Sign-On, and each handles trust of the requester differently. The two options are:

– Running WebSphere Portal Server in Trust Association mode (TAI), which allows WebSEAL to act as a front-end authentication server while WebSphere Portal Server applies its own authorization policy onto the resulting credentials that WebSEAL passes to it.

– Passing mapped usernames and passwords to WebSphere Portal Server, with which the target Web server can do its own authentication and add its own authorization policy.

With both options, both aspects of trust are maintained.

WebSphere Portal Server contains applications called portlets. Although WebSEAL can provide access control to these portlets, the portlets themselves often need to make further access control decisions, finer-grained than those controlled by WebSEAL. For example, these finer-grained access control decisions can help personalize the requesting user's Web browser screens. So, for flows coming through WebSEAL, to enable fine-grained processing, WebSEAL can be configured to pass user information, group information, or Access Manager credential information to the target portlets.

In conclusion, Access Manager and WebSphere Portal Server are based on different security models. The options below help Access Manager and WebSphere Portal Server users choose between one of the security models, or a mix:

- Access Manager as an end-to-end solution. You can choose to use Access Manager to manage access to all WebSphere Portal Server resources. Though this has the advantage of using a single security model, the disadvantage is that authorization cannot be as granular as in WebSphere Portal Server. Developers have to code to enforce fine-grained authorization on portal resources.
- Access Manager and WebSphere Portal Server for providing security solution. You can choose to have Access Manager and WebSphere security coexist. This has the advantage of no coding to enforce fine-grained authorization on portal resources. The disadvantages are redundant effort in defining ACLs, and maintaining and administering two security solutions.

4. Programmatic integration
   - Java Authentication and Authorization Service (JAAS)

     The IBM Tivoli Access Manager (Tivoli Access Manager) authorization Java classes provide an implementation of Java security code that is fully compliant with the Java 2 security model and the Java Authentication and Authorization Service (JAAS).

     The Java 2 security architecture is policy-based, and allows for fine-grained access control. When code is loaded, it is assigned permissions based on the security policy currently in effect. Each permission specifies a permitted access to a particular resource, such as read access to a specified file, or connect access to a specified host and port.

     With the Java 2 and JAAS support delivered in Tivoli Access Manager, Java applications can invoke the Tivoli Access Manager-supplied JAAS LoginModule to acquire authentication/authorization credentials from Access Manager.

     This offers Java application developers the following advantages:

- The security of Java applications is managed using the same consistent model as the rest of the enterprise.
- Java developers do not need to learn anything additional beyond Java 2 and JAAS.
- Updates to security policy involve Tivoli Access Manager-based administrator actions, rather than any code updates.

– Authorization Application Programming interface (aznAPI)

The Access Manager aznAPI provides a standard programming and management model for integrating authorization requests and decisions with applications. Use of the aznAPI allows applications to utilize fined-grained access control for application-controlled resources.

Application-specific resources may be individually defined, added to the protected object space and maintained in the authorization database in the same manner that WebSEAL defines their respective resources. ACLs and POPs may be attached to these application objects, and aznAPI calls may then be used to access the Access Manager Authorization Service to obtain authorization decisions.

Access Manager provides a C version of the API and a Java version of the API, although the Java version is really a wrapper for the original C code.

9.6 Where to find more information

- For more information about IBM Tivoli Access Manager, refer to the IBM Redbook *Enterprise Business Portals with IBM Tivoli Access Manager*, SG24-6556-00.
- For more information about WebSphere Everyplace Connection Manager for AIX, refer to the *Product Administrator's Guide.*

10

System management

System management is a critical discipline for any e-business solution. It involves pre-implementation and post-implementation activities.The pre-implementation activities are part of the design phase and occur during the deployment of the solution. The post-implementation activities occur on a routine basis and do not really begin until an application has been deployed into a production environment. This chapter covers the pre- and post-implementation system management design, technologies and activities.

© Copyright IBM Corp. 2003. All rights reserved.

10.1 System management activities

The multi-tier model provides the separation of the presentation, business logic, persistence and integration components of an application, facilitating the system management and data contingency. Figure 10-1 shows that it was complemented with two new tiers:

- System Management: this tier is responsible for the components that will manage the solution. The management range goes from solution availability, performance evaluation and alert generation to software distribution and inventory management.
- Data Contingency: this tier is responsible for back-up/restore of the solution.

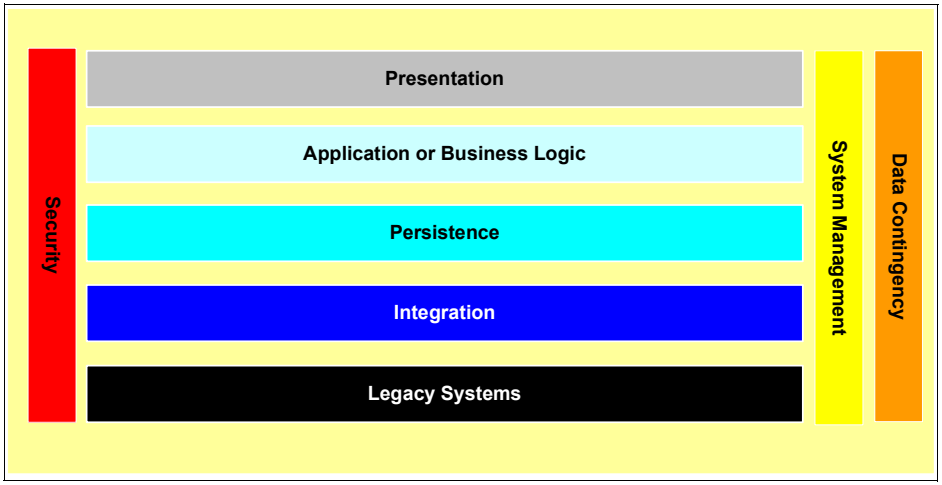


Figure 10-1 n-tier model with System Management and Data Contingency

System management can be divided into pre- and post-implementation activities:

1. Pre-implementation activities occur during the Solution Design phase and involves the answer of some questions that will drive the post-implementation activities. These answers are very related to availability requirements and used technologies. Some of these questions are:
   - Does the solution need management? Is it critical? What are the impacts of service outage?
   - What level of management is needed for each component? It is interesting to analyze each isolated component and qualify how critical that component is to the solution, considering technical and business issues.The main components are:

- WAN and LAN Connectivity
- Security
- Server hardware resources
- Operating systems
- Foundation software, like WebSphere Application Server
- On-the-shelf applications, like WebSphere Portal Server
- Developed applications
- Databases
- Clusters
- Storage

– How much money can be spent on system management? Is it worth it to manage a specific component if the cost for it is particularly high? What are the benefits of each component in business and technical terms of management?

– Is data contingency by implementing a back-up and restore policy necessary? Is an online and totally automatized process needed?

– Is a disaster recovery solution necessary?

– Will the development and testing of new applications be in-house?

– Will the stage be in-house?

– What will be the process to develop an application, stage and deploy it in the production environment?

– What is the required Service Level Agreement (SLA) established between the users and the solution provider? SLAs cover system availability hours, system utilization and problem resolution response time.

With the answers to some of these questions, it is possible to design a solution that supports and implements the required level of system management. Part of the design phase includes matching the solution requirements, the existing tools and technologies, and the cost of the solution. From this matching, the solution that best fits the technical and business requirements can be developed.

2. Post-implementation activities occur after the solution is deployed. These activities require highly specific skills and professional experience to perform competently. Some of the core activities that are performed depending on the required level of system management are:

   – Application management

   – Performance monitoring

- Availability management
- Security management
- Disaster recovery
- Operating system and network administration
- Asset management
- Software distribution
- Problem reporting
- Change management

A detailed explanation of each pre- and post-implementation activity is beyond the scope of this redbook. Our focus is on the key system management activities related to the implementation of WebSphere Everyplace Access and its components for a Pervasive Portal solution.

10.2 WebSphere Everyplace Access management

This chapter will not focus on system management of products that are the foundation of the WebSphere Everyplace Access, like WebSphere Application Server and WebSphere Portal Server.

For information on system management of WebSphere Application Server, refer to the IBM Redbook *IBM WebSphere V4.0 Advanced Edition Handbook* - SG24-6176-00 and to *User-to-Business Patterns Systems Management Guidelines*, REDP0401.

For information on system management of WebSpahere Portal Server, refer to the IBM Redbook *Access Integration Pattern using IBM WebSphere Portal Server* - SG24-6267-00.

Some components of the WebSphere Everyplace Access which will be discussed are:

- Everyplace Synchronization Server
- Intelligent Notification Services
- Device Manager

10.2.1 Everyplace Synchronization Server

Everyplace Synchronization Server is a solution for synchronizing data to back-end databases.The Synchronization server uses WebSphere Portal Server

to provide administration and configuration by the Synchronization Server Administration portlets. The administration portlets are:

- Manage Servers - This portlet allows the administrator to monitor the Synchronization Server status, stop and start servers, and view a list of actively synchronizing users.
- Remove User Preferences - This portlet allows the administrator to remove user profiles from the Synchronization Server database.
- Device Profiles - This portlet provides sample device configurations that users can choose to use or to modify.
- Lotus Domino Adapter - This portlet allows the administrator to edit Lotus Domino Adapter specific information, including adapter authentication, Lotus Domino Servers to monitor, caching and polling frequency, Field Conversion Databases location, and server cache enabling.
- Microsoft Exchange Adapter - This portlet allows the administrator to edit Microsoft Exchange Adapter specific information, including adapter authentication, Microsoft Exchange Adapter servers to monitor, caching and polling frequency, and server cache enabling.
- Server Settings - This portlet allows the administrator to configure server message logs and trace logs.

The relational database adapter is not managed using the portlets described above. It uses an embedded version of DB2 Everyplace, so most administration tasks are accomplished using the Mobile Devices Administration Center. The relational database adapter only uses the WebSphere Portal interface for user and group management.

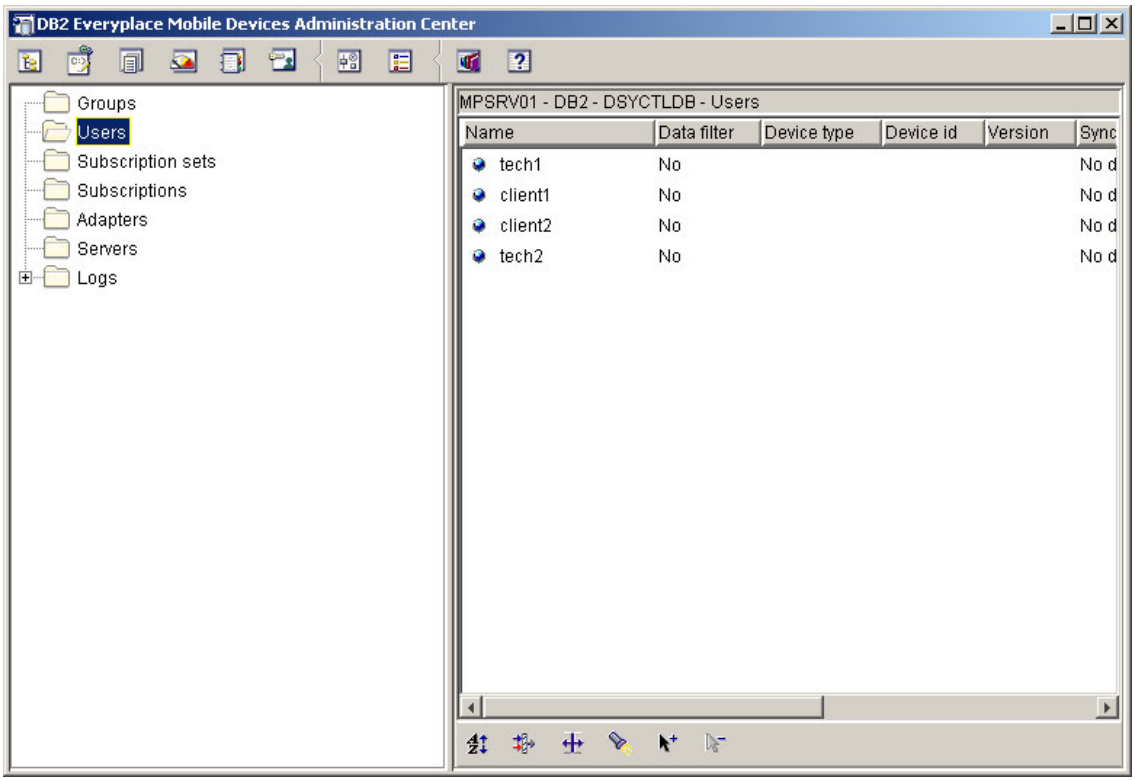


Figure 10-2 Mobile Devices Administration Center

10.2.2 Intelligent Notification Services

Intelligent Notification Services deliver notifications to users via multiple delivery channels based on user preferences and subscriptions. It uses WebSphere Portal Server to provide administration by the Administration and User portlets.

Administering Intelligent Notification Services involves managing Intelligent Notification users, configuring e-mail subscriptions and managing servers, gateway adapters, and content adapters. Servers must be configured, started, stopped, and monitored. Gateway adapters and content adapters must be configured. Custom gateway adapters and content adapters can be added.

Administration portlets

The Intelligent Notification administrator uses a set of administrative portlets to manage servers, configure gateway adapters, configure e-mail subscriptions, and remove the preferences of deleted users. The administrative portlets are as follows.

- Manage Servers portlet - Administrators use the Manage Servers portlet to start, stop, and configure Intelligent Notification servers.
- Remove Users Preferences portlet - Administrators use the Remove User Preferences portlet to remove the user preferences of users that have been deleted from WebSphere Portal.
- Configure Gateways portlet - Administrators use the Configure Gateways portlet to view and configure gateway adapters.
- Configure Subscription portlet - Administrators use the Configure Subscriptions portlet to specify configuration settings for e-mail subscriptions.

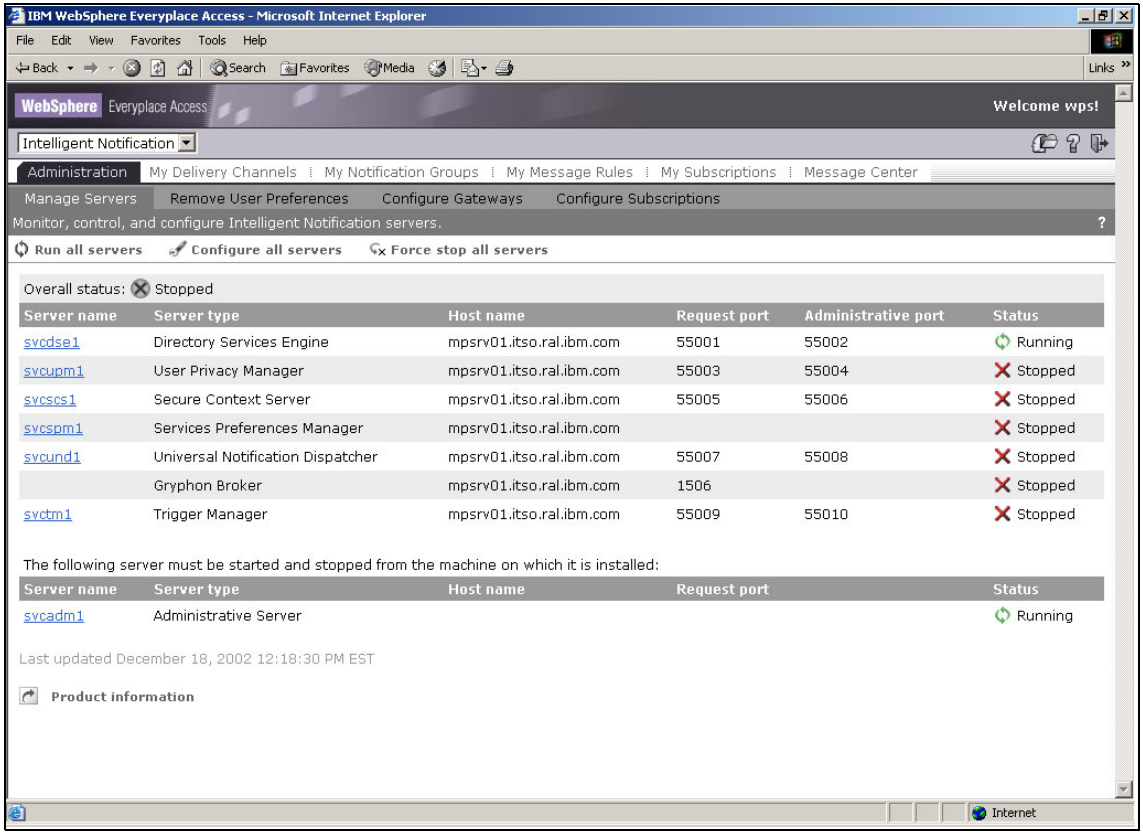


Figure 10-3 Intelligent Notification Services Administration portlets

User portlets

Users work with portlets to manage delivery channel settings, manage notification groups, specify rules for message delivery, manage notifications, and subscribe to content sources. The user portlets are:

- My Delivery Channels - On the My Delivery Channels page, users can add or delete delivery channels and edit delivery channel settings. There are multiple delivery channel portlets on the My Delivery Channels page, one for each type of delivery channel.
- My Notification Groups - With the My Notification Groups portlet, users add and remove notification groups. Users also add and remove users from those groups.
- My Message Rules - With the My Message Rules portlet, users create and modify rules for receiving messages. The rules specify what priority messages are to be received, on which delivery channels, and from which notification groups.
- My Message Center - With the My Message Center portlet, users can view a list of recent notifications and content for any of the messages listed. Users can also use this portlet to send simple notifications to other users of Intelligent Notification Services.
- My Subscriptions - On the My Subscriptions page, users subscribe to content sources, such as sample news, stock, weather, and e-mail sources. Users add, edit, and remove subscriptions for content sources. There are multiple portlets on the My Subscriptions page, one for each content source.

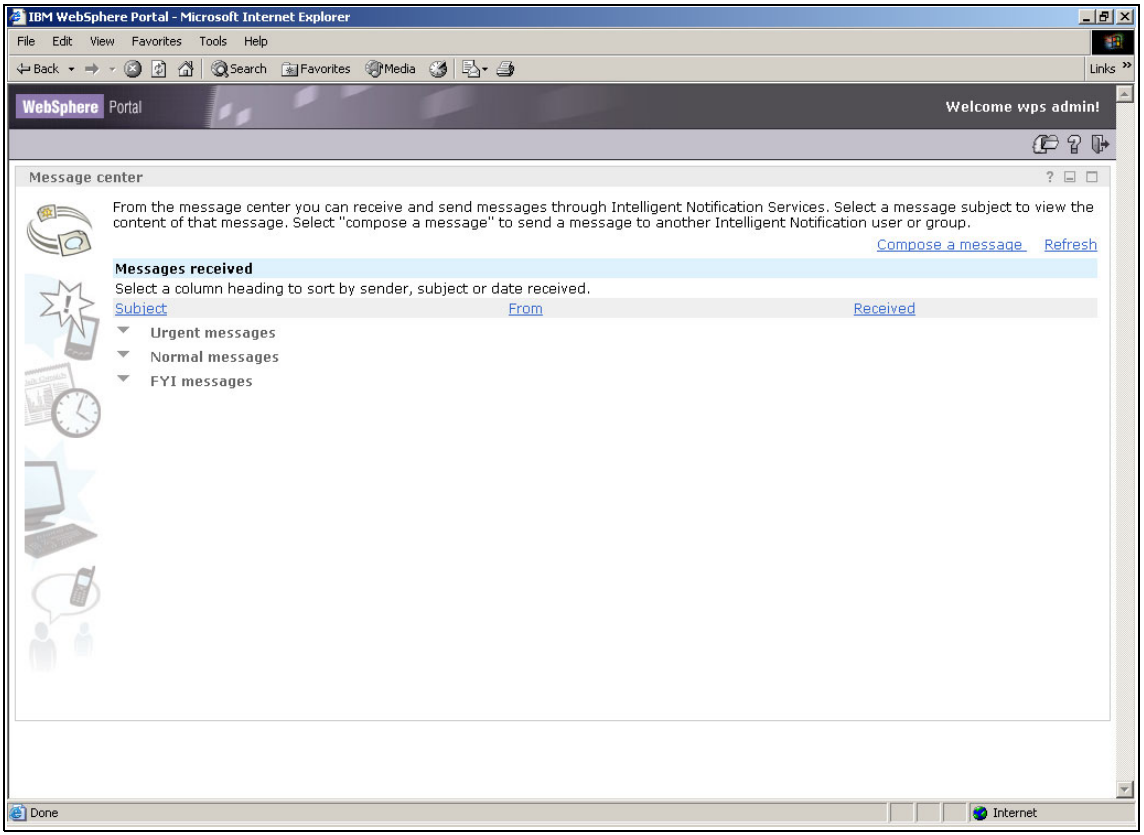


Figure 10-4 Intelligent Notification Services User portlets

10.2.3 Device Manager

Device Manager is comprised of a set of servlets running in an application server and a database that is the repository for device related data.The Device Manager can manage PALM OS PDAs, generic Windows CE devices and SyncML DM capable devices.

Device Manager provides three administration components:

- Device Manager Console
- Customer Care application
- Self Care application

Device Manager Console

The Device Manager Console is a client-server Java application that runs on Windows and provides a graphical user interface.

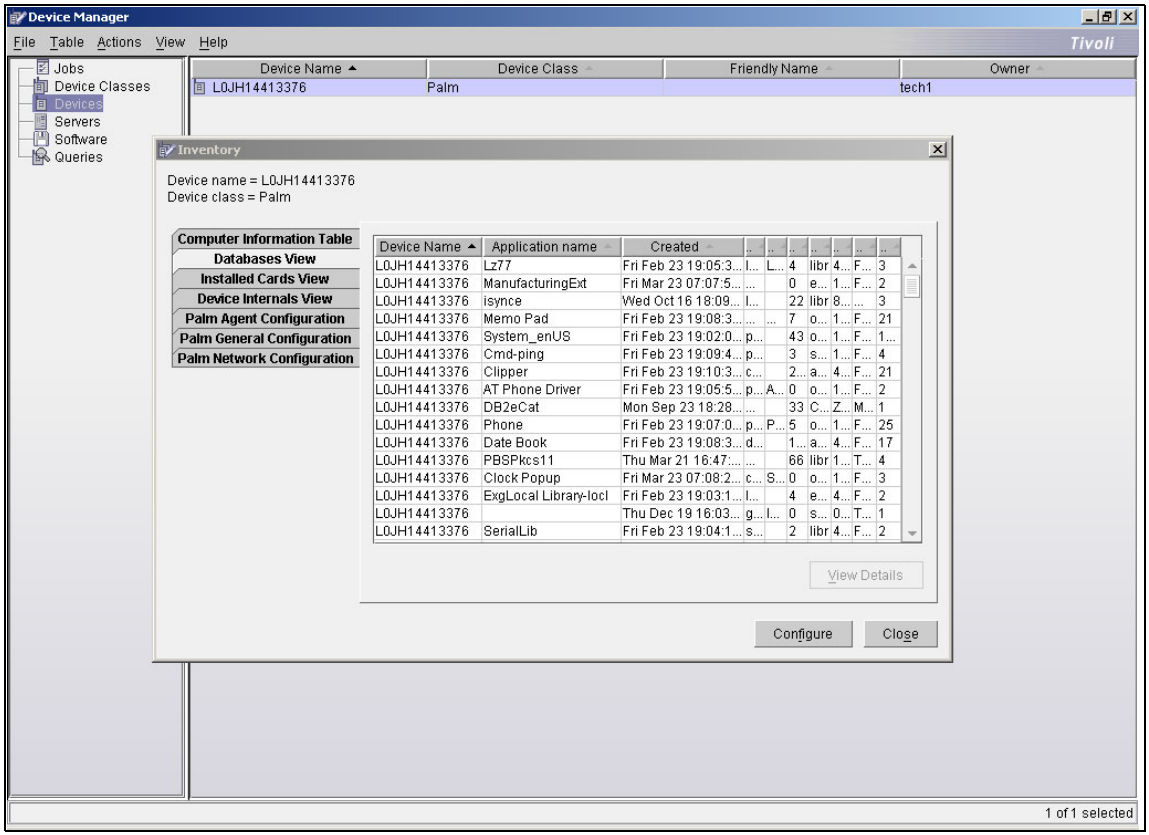


Figure 10-5 Device Manager console

From this console, the following elements can be managed:

- Jobs - A job is specialized processing initiated by the Device Manager. The job types that can be managed using the console are device configuration (updates the configuration of a device, including network parameters), inventory collection (collects a list of software applications, hardware and current configuration parameters present on a device) and software distribution (send a software package to a target device). These jobs are submitted and take place the next time the device connects.
- Devices - The console provides management of device-related information, software, inventory and jobs.

- Device Class - Device Class is a collection of devices that have similar characteristics and that can be managed similarly. The console provides management of device class information, and device-class-related software and jobs.
- Server - A server is a Device Manager server that processes jobs for devices. The console displays the name and port number of each registered Device Manager server.
- Software - This relates to a software package to be sent to a device. A software package resides on a Web server and this information (URL) is stored in the Device Manager database.
- Queries - The console uses queries to request device information or to target a job to devices with particular characteristics.

Administrators using the Device Manager Console can submit jobs to a single device, all devices of a device class or a selection of devices of the same device class.

Customer Care application

The Customer Care application is a Web application accessed by browser, provided by the subscription manager component. It is mainly used by Customer Service Representatives (CSRs) to manage devices, jobs and software to these devices. It allows a subset of the tasks that an administrator performs with the Device Manager Console and performs these tasks one device at a time.

Self Care application

The Self Care application is similar to Customer Care but is used by the device owners to manage their own devices. It is also a Web application accessible by a browser. The available tasks for Self Care are a subset of the tasks available for Customer Care.

10.3 System Management and monitoring using Tivoli products

In order to provide an overview of how to implement performance, availability, configuration and operation management, core IBM Tivoli products will be listed and described at a high level. It is not the objective of this redbook to discuss these system management products in detail.

IBM Tivoli Enterprise™ Console (TEC)

The IBM Tivoli Enterprise Console® product is a powerful, rules-based event management application that integrates network, systems, database, and application management. It offers a centralized, global view of your computing enterprise while ensuring the high availability of your application and computing resources. It collects, processes, and automatically responds to common management events, such as a database server that is not responding, a lost network connection, or a successfully completed batch processing job. It acts as a central collection point for alarms and events from a variety of sources, including those from other Tivoli software applications, custom applications, network management platforms, and relational database systems.

IBM Tivoli Monitoring

IBM Tivoli Monitoring provides monitoring for essential system resources such as performance, including disks, CPU and applications, to detect bottlenecks and potential problems, and to automatically recover from critical situations.

IBM Tivoli Monitoring for Web Infrastructure

IBM Tivoli Monitoring for Web Infrastructure is a critical tool to help ensure the optimal performance and availability of both application servers and the associated Web servers that feed them. It provides a single point of control to enable IT organizations to understand the health of a Web-based environment's key elements. It allows administrators to quickly identify problems, alert appropriate personnel as required, and offer a means for automated problem correction.

IBM Tivoli Monitoring for Databases

The IBM Tivoli Monitoring for Databases ensures the availability and optimal performance of DB2, Oracle, and Informix® database servers.

IBM Tivoli Monitoring for Messaging and Collaboration

IBM Tivoli Monitoring for Messaging and Collaboration monitors the status of Domino servers, identifies server and system problems in real time, notifies administrators and takes automated actions to resolve Domino server problems. The product also collects monitoring data to help you analyze performance and trends, and helps you address problems before they affect end users.

IBM Tivoli Netview

IBM Tivoli NetView® discovers TCP/IP networks, displays network topologies, correlates and manages events and SNMP traps, monitors network health, and gathers performance data. Tivoli NetView meets the needs of managers of large networks by providing the scalability and flexibility to manage mission-critical environments.

IBM Tivoli Storage Manager

IBM Tivoli Storage Manager protects data from hardware failures, errors, and unforeseen disasters by storing back-up and archive copies in offline and offsite storage. It is responsible for implementing the data contingency (back-up/restore) policy.

10.3.1 Integrating System Management in the Pervasive Portal solution

Figure 10-6 on page 232 shows an example of how to integrate the system management infrastructure in a Pervasive Portal solution using the Tivoli products. In order to understand the security layer for Data Contingency and System Management, refer to 10.1, “System management activities” on page 220.

The system management solution in this example is composed of the Tivoli Netview that manages the network infrastructure, the Tivoli Monitoring that manages operational systems, hardware resources such as memory and disk and specific applications like WebSphere and DB2, and Tivoli Enterprise Console, which consolidates all the events created by Tivoli Netview and Monitoring and generates alarms. To manage the specific applications, you will require Tivoli Monitoring as the server and the Tivoli agents (Tivoli Monitoring for Web Infrastructure, for Databases, for Messaging and Collaboration) running with each specific application.

The data contingency is provided by the Tivoli Storage Manager that backs up data using the Storage Area Network for the equipment connected to it, using a LAN (dotted line) for the other equipment that also needs backing up. This sample diagram does not take care which components need back-up and includes all of them in the back-up LAN or SAN. Usually, databases have a lot of data to back up and it is recommended, in terms of performance and security, to use a SAN. So as not interfere in the data access performance and integrity of the databases while performing a back-up, some components can be used that allow the online back-up, such as the Tivoli Storage Manager for Databases.

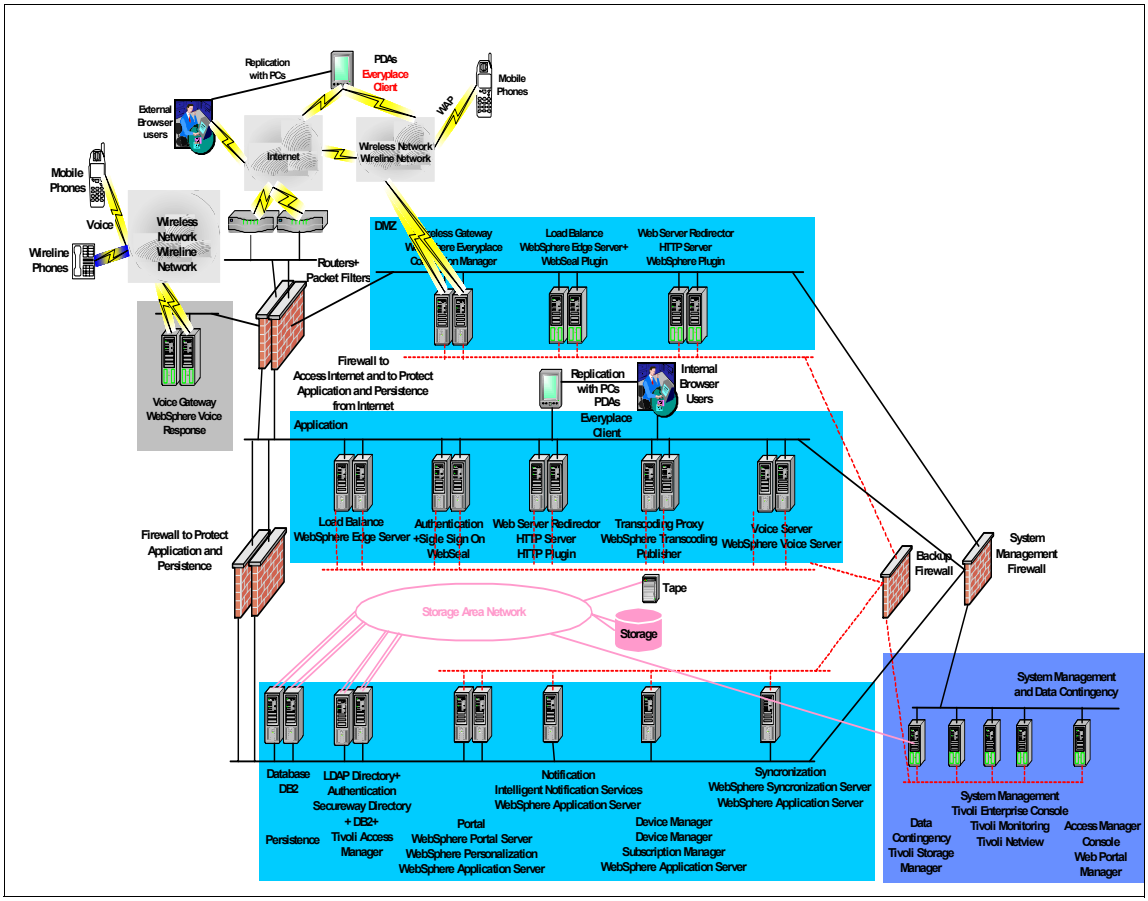


Figure 10-6 System Management and Data Contingency represented in the operational model

Note: Because of the extensive details of the previous diagram, it is difficult to read some parts on a printout. If you like to read the details, please use the original PDF document and magnify the area you want to see.

10.4 Production, Staging and Development environment

Keeping in mind the growth possibility of a solution and the insertion of new components, new functionalities or even the need for performance testing, access, security or environment availability, the solution should include, besides the Production environment, a Staging and Development environment. The diagram represents a generic Production environment.

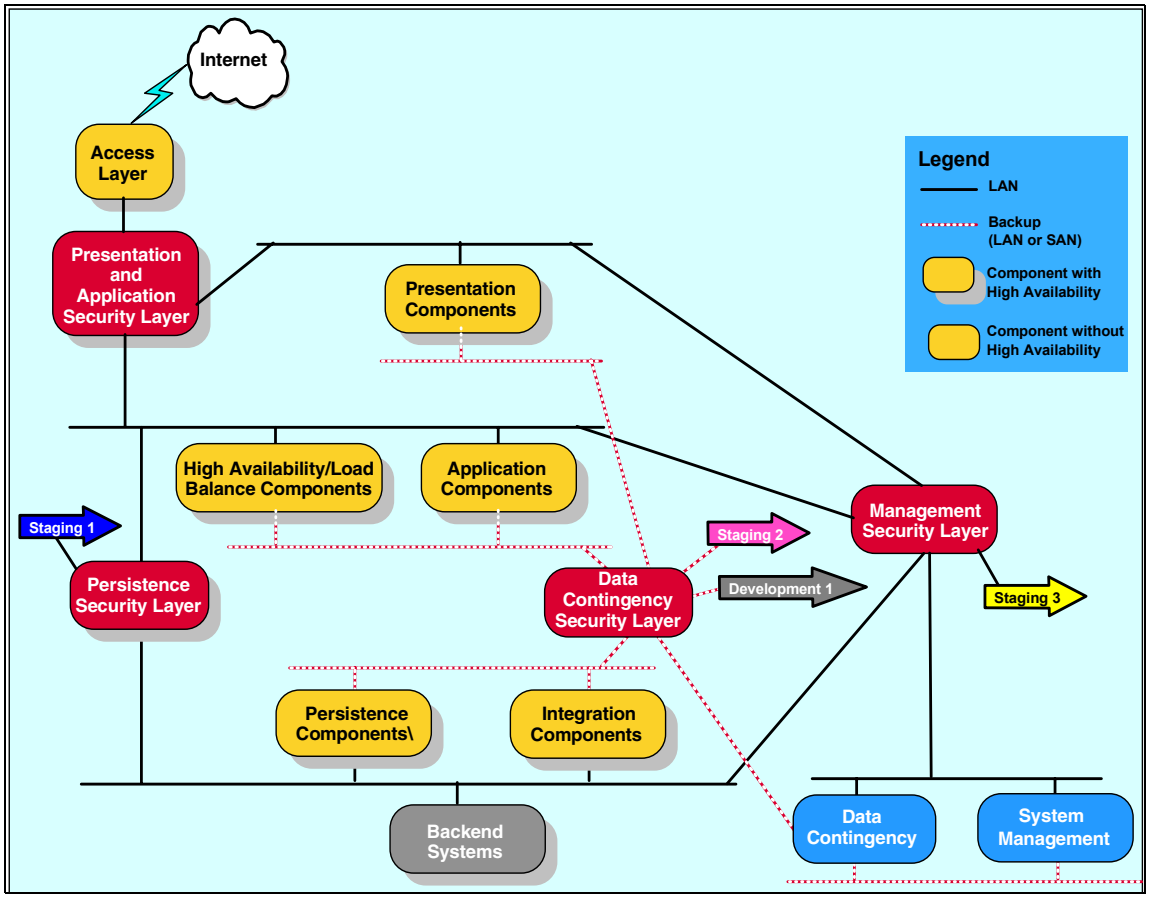


Figure 10-7 Generic n-tier Production environment

With the Staging environment, it is possible to perform all kinds of tests without interfering with the current production of the solution. After the testing and approval of new functionalities or new definitions of non-functional requirements such as performance, it can be migrated to the Production environment.

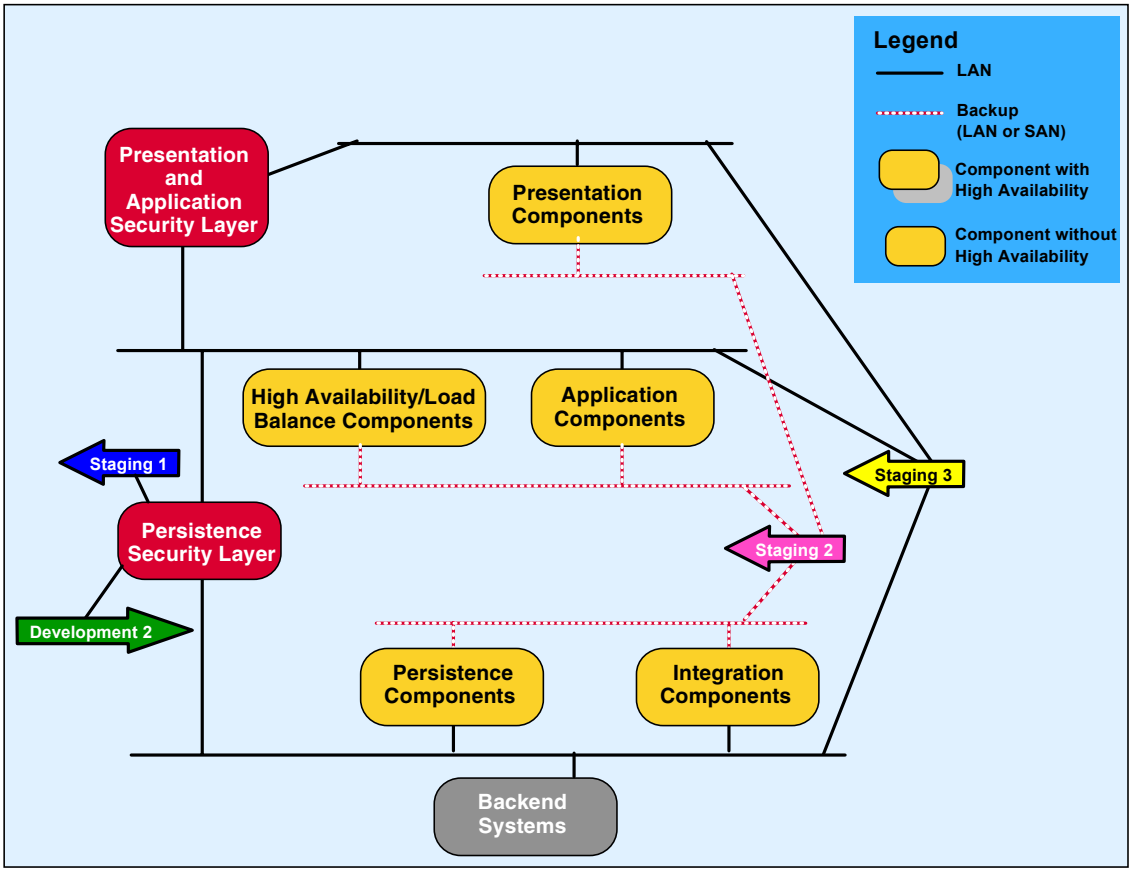


Figure 10-8 A generic staging environment

The Development environment provides all the necessary tools for creation and implementation of the new functionalities.

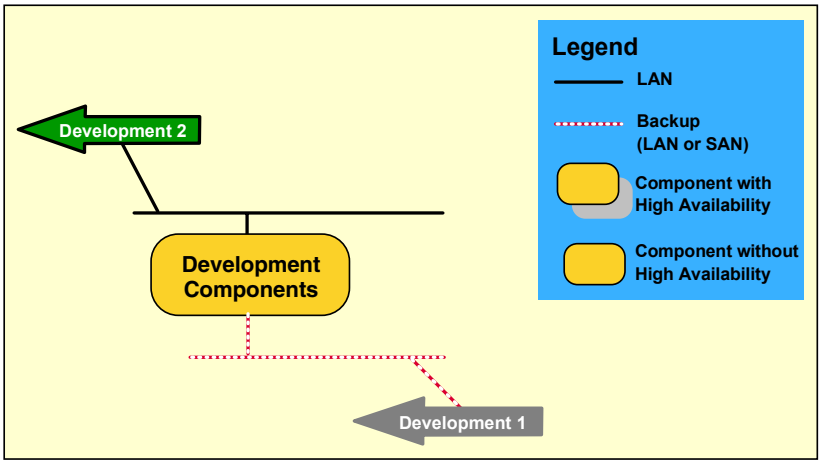


Figure 10-9 A generic development environment

Figure 10-10 on page 236 shows a possible connection of all three environments. Since the people in charge of the production are not, in general, the same who test the new solutions and develop the new functionalities, the environments are protected by security layers.

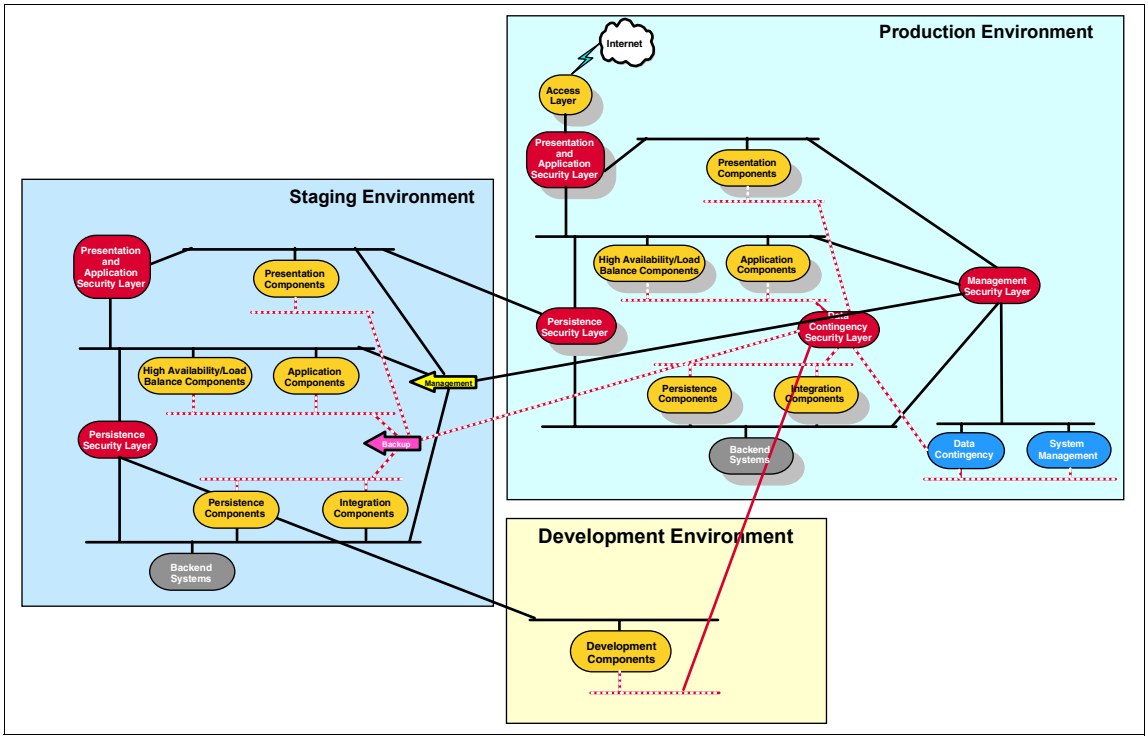


Figure 10-10 Connection between environments

10.5 Where to find more information

- For more information about IBM Tivoli products, go to:
  http://www.tivoli.com/
- For more information on WebSphere Everyplace Access, refer to the WebSphere Everyplace Access infocenter at:
  http://www.ibm.com/software/pervasive/products/library/ws_everyplace_access.shtml
- For more information about test and development environment, refer to: *WebSphere Application Server Test Environment Guide,* SG24-6817

11

Performance and availability

The e-Business model is very different from the traditional models. Placing an e-Business application on the Internet means you have potentially millions of users, all around the world, operating at all times of the day. If this is extended to a Pervasive solution, it increases the complexity and requirements to support a good quality of the offered services.

This chapter discusses the non-functional requirements of performance, availability and scalability related to each component of a Pervasive Portal solution.

© Copyright IBM Corp. 2003. All rights reserved.

11.1 Concepts

The multi-tier model provides the separation of the presentation, business logic, persistence and integration components of an application, facilitating the scalability and implementation of availability and performance. As shown in Figure 11-1, these non-functional requirements complement the picture.

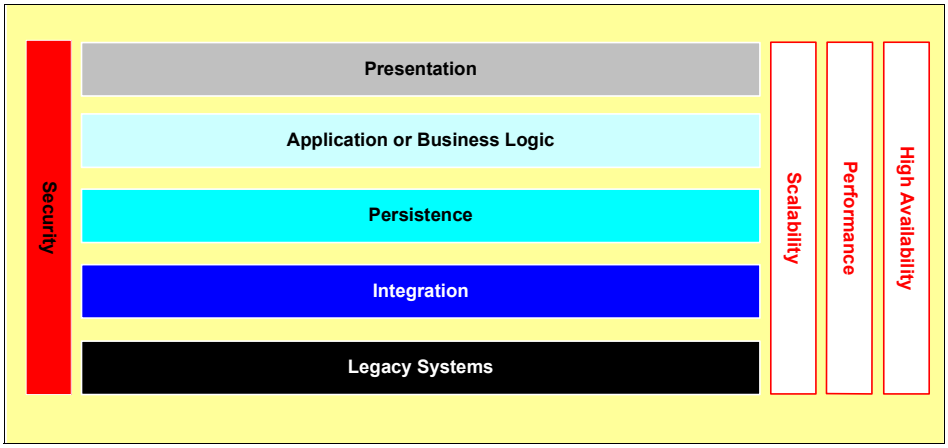


Figure 11-1 Multi-tier model with high availability, performance and high availability

To understand these non-functional requirements, a brief description is provided below.

Availability

In IT, a system, application, or component that can be used is considered to be *available*. Availability is the measurement of the time during which the element is out of use, that is, experiencing an outage. Availability is usually expressed as a percentage of time the element is not out of service; the availability measure is calculated by subtracting the duration of the outage from the base time and dividing the results by the base. *High availability* is the term usually associated with the ability to run for extended periods of time with no (or minimal) unplanned outage. High availability refers to a system or component that is continuously operational for a desirably long length of time. Availability can be measured relative to "100% operational" or "never failing". A widely-held but difficult to achieve standard of availability for a system or product is known as "five 9s" (99.999 percent) availability.

Measure	Outage per year
99.9999%	32 seconds
99.999%	5 minutes
99.99%	53 minutes
99.9%	8.8 hours
99%	87 hours (3.6 days)
90%	876 hours (36 days)

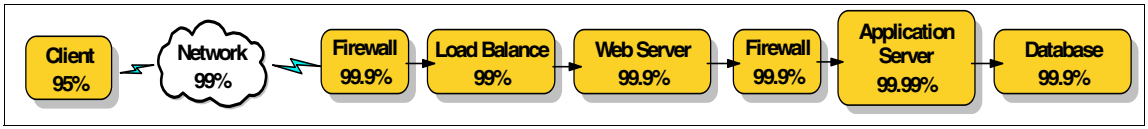


Figure 11-2 Availability chain

To provide continuous availability, every element of the infrastructure must support continuous availability. The level of service delivered can be no higher than the availability of the weakest link. Improving one element to deliver near 100% availability while ignoring other elements does not provide as much benefit as a more balanced approach. The total availability of an infrastructure is calculated by multiplying the availability values of all components. In Figure 11-2, the availability of the entire solution is 93%. The value of each node does not represent reality; these are only samples.

Performance

Performance has three definitions:

1. The speed at which a computer operates, either theoretically (for example, using a formula for calculating Mtops - millions of theoretical instructions per second) or by counting operations or instructions performed (for example, millions of instructions per second (MIPS)) during a benchmark test. The benchmark test usually involves tasks that attempt to imitate the kind of work the computer does during actual use.
2. The total effectiveness of a computer system, including throughput, individual response time, and availability.
3. In a transactional system, how long it takes to get a response to a request.

Scalability

Scalability is the capability of a system or component to adapt readily to a greater or lesser intensity of use, volume, or demand while still meeting business objectives such as acceptable levels of performance and availability. There are two types of scalability:

- Vertical - In terms of hardware, this is achieved by adding more power to the hardware, like CPUs, memory, etc. The advantage of this type of scalability is the use of the same system running on the same hardware, only with more power. It is important to check that the system supports and recognizes all the power features added. The disadvantages are that the scalability is limited to the hardware growth capability and the solution support for that growth.

  In terms of a Web application, vertical scaling provides a straightforward mechanism for creating multiple instances of an application server, and hence multiple JVM processes.

- Horizontal -In terms of hardware, this is achieved by adding more hardware equipment in parallel to support the load. For this type of solution, a load balance component is required to distribute the requests between each piece of equipment that is running the solution. The advantage of this type of scalability is the endless growth achieved by adding new hardware and the possibility to use less powerful machines. If there are at least two machines attending the requests, it will improve the availability, because if one fails, the other can assume the workload and also improve the performance since there are more machines attending the load. The disadvantage is that some solutions do not support load balancing.

  In terms of a Web application, in horizontal scaling, clones of an application server are created on multiple physical machines. This enables a single application server to span several machines yet still present a single system image. Horizontal scaling can provide both increased throughput and failover support when compared to vertical scaling topologies.

11.2 Techniques

There are some known techniques used to provide availability and performance. This section is just a quick overview of the techniques with some specific details in regards to “Pervasive Portals”. For more information about scalability and availability, refer to the redbook *Patterns for the Edge of Network*, SG24-6822.

Clustering

In a computer system, a cluster is a group of servers and other resources that act as a single system and enable high availability and, in some cases, load balancing and parallel processing (performance).

Clustering configurations for high availability

The goal of these configurations is to improve availability. There are two basic configurations for high availability.

1. The simplest high-availability cluster configuration is a two-node cluster. There is one primary system for all cluster resources and a second system that is a back-up, ready to take over during an outage of the primary system.

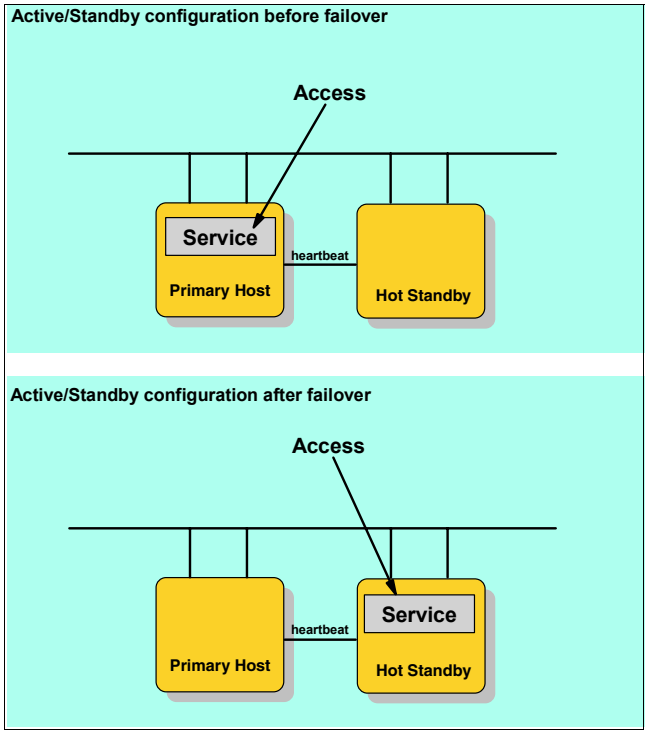


Figure 11-3 Active/standby configuration

2. Another typical configuration is the mutual takeover cluster. Each node in this environment serves as the primary node for some sets of resources and as the back-up node for other sets of resources. With mutual takeover, every system or node is used for production work, and all critical production work is accessible from multiple systems, multiple nodes, or a cluster. The goal of this technique is to improve availability and scalability.

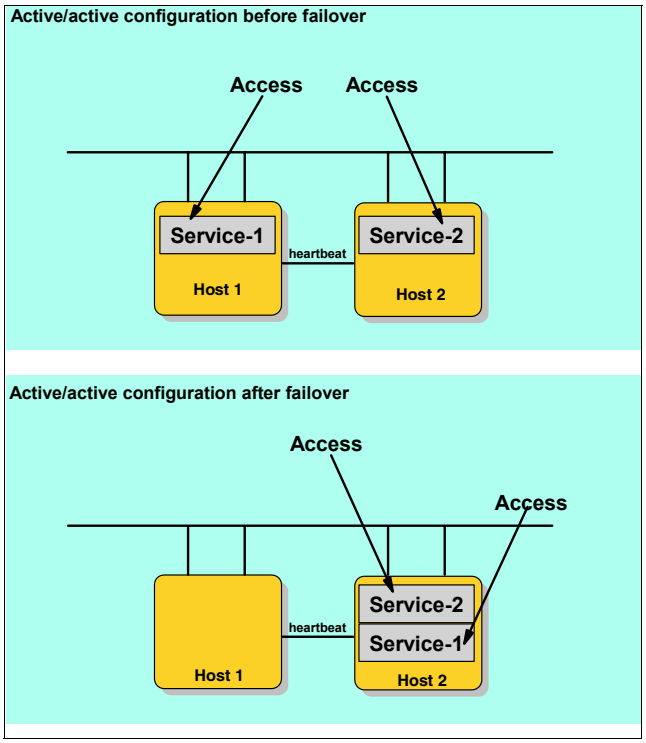


Figure 11-4 Active/active configuration

In both of these scenarios, replication is key. Replication means that a copy of something is produced in real time, for instance, copying objects from one node in a cluster to one or more other nodes in the cluster. Replication makes and keeps the objects on your systems identical. If you make a change to an object on one node in a cluster, this change is replicated to other nodes in the cluster.

Load balancing

The goal of this technique is to improve performance and availability by supporting horizontal scalability. Load balancing is dividing the amount of work that a computer has to perform between two or more computers so that more work gets done in the same amount of time and, in general, all users are served faster.

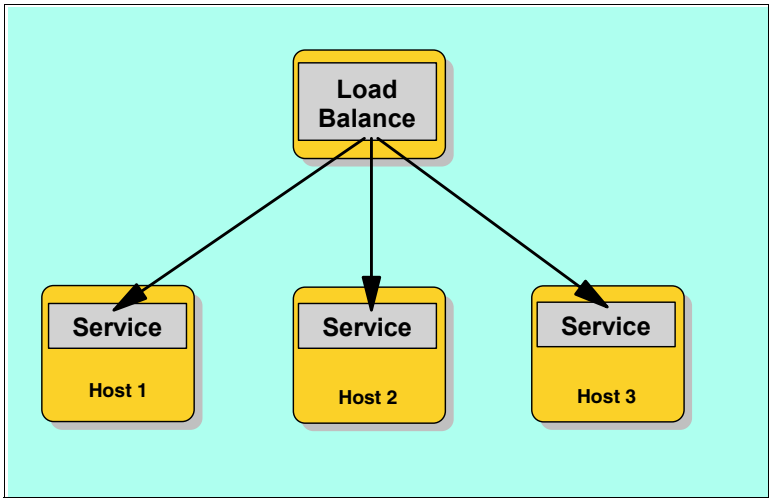


Figure 11-5 Load balance configuration

On the Internet, companies whose Web sites get a great deal of traffic usually use load balancing. To load balance Web traffic, there are several approaches. For Web serving, one approach is to route each request in turn to a different server host address in a domain name system (DNS) table, round-robin fashion. Usually, if two servers are used to balance a work load, a third server is needed to determine which server to assign the work to. Since load balancing requires multiple servers, it is usually combined with failover and back-up services. In some approaches, the servers are distributed over different geographic locations.

Caching

The goal of this technique is to improve performance. A cache is a special high-speed storage mechanism. It can be either a reserved section of main memory or an independent high-speed storage device. Two types of caching techniques are commonly used: memory caching and disk caching.

When data is found in the cache, it is called a cache hit, and the effectiveness of a cache is judged by its hit rate. Many cache systems use a technique known as smart caching, in which the system can recognize certain types of frequently used data.

Content delivery

The goal of this technique is to improve performance and availability. Content delivery (sometimes called content distribution) is the service of copying the pages of a Web site to geographically dispersed servers and, when a page is

requested, dynamically identifying and serving page content from the closest server to the user, enabling faster delivery. Typically, high-traffic Web site owners and Internet service providers (ISPs) hire the services of the company that provides content delivery.

A common content delivery approach involves the placement of cache servers at major Internet access points around the world and the use of a special routing code that redirects a Web page request (technically, a Hypertext Transfer Protocol (HTTP) request) to the closest server. When the Web user clicks a URL that is content-delivery enabled, the content delivery network re-routes that user's request away from the site's originating server to a cache server closer to the user. The cache server determines what content in the request exists in the cache, serves that content, and retrieves any non-cached content from the originating server. Any new content is also cached locally. Other than faster loading times, the process is generally transparent to the user, except that the URL served may be different from the one requested.

The three main techniques for content delivery are: HTTP redirection, Internet Protocol (IP) redirection, and domain name system (DNS) redirection. In general, DNS redirection is the most effective technique.

11.3 Products

WebSphere Everyplace Access does not support caching in this version. Components like DB2e, Device Management, and synchronization require having unique sessions that prevents using caching.

Other components in WebSphere Everyplace Access, for example portal, can use external caching. There are some products that implement these techniques in order to provide availability and performance. Some of them are described below.

IBM WebSphere Edge Server

IBM WebSphere Edge Server V2 for Multiplatforms is a Web infrastructure software that addresses the scalability, reliability and performance needs of e-business applications in both local and geographically distributed environments. Its functions incorporate robust, leading-edge caching and load balancing that together compensate for the inherent weakness of the Internet in supporting critical business applications and expectations. In addition, IBM WebSphere Edge Server V2 introduces significant new functions, defined by the Edge Services Architecture, which offer additional capabilities.

The Edge Services Architecture has been defined to enable the delayering of application programs so as to more fully exploit the distributed execution environment created when edge servers are placed at the edge of the network.

Delayering means decomposing an application into smaller modules for placement on appropriate platforms throughout a distributed network topology. It is advantageous in accommodating a variety of business models which may be evolving over time. Delayering applications in “edgable” components opens significant new opportunities for e-businesses, service providers, and independent software vendors, allowing application scalability and good user response times.

IBM WebSphere Edge Server is made up of four main components which allow you to reduce Web server congestion, increase content availability and improve Web server performance:

- Caching and filtering - The caching and filtering component, known as the Caching Proxy, is a server that provides highly scalable caching and filtering functions used in receiving requests and serving URLs. Since tunable caching is capable of supporting high cache hit rates, this component can reduce bandwidth costs and provide more consistent rapid customer response times. Additionally, the Caching Proxy is programmable via plug-ins, and it caches and invalidates dynamic content generated by the IBM WebSphere Application Server’s dynamic cache.
- Load balancing - The load balancing component, known as Network Dispatcher, is a server that is able to dynamically monitor and balance TCP servers and applications in real time. It improves a Web site’s availability, scalability and performance by transparently clustering edge, Web and application servers. The main advantage of the load balancing component is that it allows heavily accessed Web sites to increase capacity, since multiple TCP servers can be dynamically linked in a single entity that appears in the network as a single logical server.
- Content Distribution - The Content Distribution Framework provides an infrastructure that can be used to distribute static, dynamic, and multimedia Web content, along with application components to production servers, rehosting servers and edge server caches throughout the network.
- Application Offload - A typical Web application setup consists of three tiers: Presentation, Business logic, and Data Store, co-located at the origin application server. Application Offload moves some of the Presentation and Business logic related processing to the edge of the network. It enables the Edge Server to perform page composition from cached and rehosted fragments and Web objects.

For detailed information about IBM WebSphere Edge Server, refer to the IBM Redbook *IBM WebSphere Edge Server: New Features and Functions in Version 2* - SG24-6511.

High Availability Cluster Multi-Processing for AIX

IBM's tool for building UNIX®-based mission-critical computing platforms is the HACMP software. The HACMP software ensures that critical resources are available for processing. HACMP has two major components: high availability (HA) and cluster multi-processing (CMP).

High Availability

Until recently, the only avenue for achieving high availability in the UNIX realm was through fault tolerant technology. Fault tolerance relies on specialized hardware to detect a hardware fault and instantaneously switch to a redundant hardware component, whether the failed component is a processor, memory board, power supply, I/O subsystem, or storage subsystem.

Although this cutover is apparently seamless and offers non-stop service, a high premium is paid in both hardware cost and performance because the redundant components perform no processing. More importantly, the fault tolerant model does not address software failures, by far the most common reason for down time.

High availability views availability not as a series of replicated physical components, but rather as a set of system-wide, shared resources that cooperate to guarantee essential services. High availability combines software with industry-standard hardware to minimize down time by quickly restoring essential services when a system, component, or application fails. While not instantaneous, services are restored rapidly, often in less than a minute.

The difference between fault tolerance and high availability, then, is this: a fault tolerant environment has no service interruption, while a highly available environment has a minimal service interruption. Many sites are willing to absorb a small amount of down time with high availability rather than pay the much higher cost of providing fault tolerance. Additionally, in most highly available configurations, the back-up processors are available for use during normal operation.

High availability systems are an excellent solution for applications that can withstand a short interruption should a failure occur, but which must be restored quickly. Some industries have applications so time-critical that they cannot withstand even a few seconds of down time. Many other industries, however, can withstand small periods of time when their database is unavailable. For those industries, HACMP can provide the necessary continuity of service without total redundancy.

Cluster Multi-Processing

Cluster multi-processing is a group of loosely coupled machines networked together, sharing disk resources. In a cluster, multiple server machines cooperate to provide a set of services or resources to clients.

Clustering two or more servers to back up critical applications is a cost-effective high availability option. You can use more of your site's computing power while ensuring that critical applications resume operations after a minimal interruption caused by a hardware or software failure.

Cluster multi-processing also provides a gradual, scalable growth path. It is easy to add a processor to the cluster to share the growing workload. You can also upgrade one or more of the processors in the cluster to a more powerful model. If you are using a fault tolerant strategy, you must add two processors, one as a redundant back-up that performs no processing during normal operations.

There are two basic types of cluster configurations:

• Standby configurations—These are the traditional redundant hardware configurations where one or more standby nodes stand idle, waiting for a server node to leave the cluster.

• Takeover configurations—In this configuration, all cluster nodes do useful work, processing part of the cluster's workload. There are no standby nodes. Takeover configurations use hardware resources more efficiently than standby configurations since there is no idle processor. Performance can degrade after node detachment, however, since the load on remaining nodes increases.

For detailed information about High Availability Cluster Multi-Processing for AIX, refer to the IBM Redbook *Configuring Highly Available Clusters Using HACMP 4.5*, SG24-6845-01.

11.4 Applying to a Pervasive Portal solution

An end-to-end solution has many components. Providing availability and scalability to an end-to-end solution really means managing each component's capacities, performance and availability. Increasing the performance and availability of one component may change the dynamics of the transaction, thereby moving the bottleneck to another component. To increase the scale of a solution, many or all components of that solution must scale to service the increasing number of requests. There are some rules to help provide availability and performance:

1. Define the availability that is required for each component of the solution. This can be done by understanding how critical each component is to the solution.

It is important to remember that if high availability is required for one component, every other component that has dependency relationships with this component has to provide at least the same level of availability. Some components of a solution support load balancing; this provides availability and at the same time increases the performance, considering that the requests will be held by more than one server at the same time.

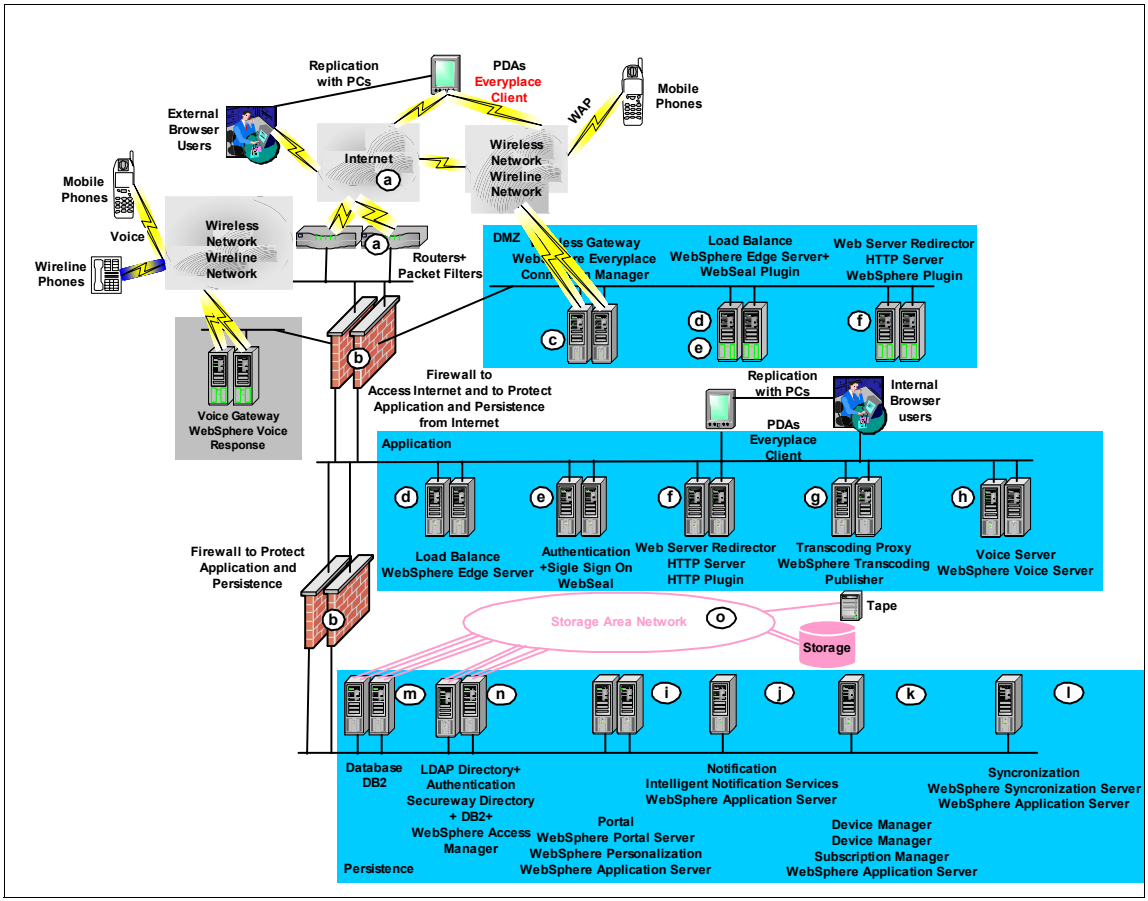


Figure 11-6 Performance and Availability for Pervasive Portal Solution components

Note: Because of the extensive details of the previous diagram, it is difficult to read some parts on a printout. If you would like to read the details, please use the original PDF document and magnify the area you want to see.

a. ISP and routers - If Internet access is critical to the solution, it is important to have redundant Internet links, if possible with different service

providers. To provide transparent high availability of the inbound and outbound communication, one of the solutions is to use technologies such as Border Gateway Protocols (BGP) and Hot Standby Router Protocol (HSRP).

BGP performs interdomain routing in Transmission-Control Protocol/Internet Protocol (TCP/IP) networks. BGP is an exterior gateway protocol (EGP), which means that it performs routing between multiple autonomous systems or domains and exchanges routing and reachability information with other BGP systems.

Using HSRP, a set of routers works in concert to present the illusion of a single virtual router to the hosts on the LAN. This set is known as an HSRP group or a standby group. A single router elected from the group is responsible for forwarding the packets that hosts send to the virtual router. This router is known as the Active router. Another router is elected as the Standby router. In the event that the Active router fails, the Standby assumes the packet-forwarding duties of the Active router.

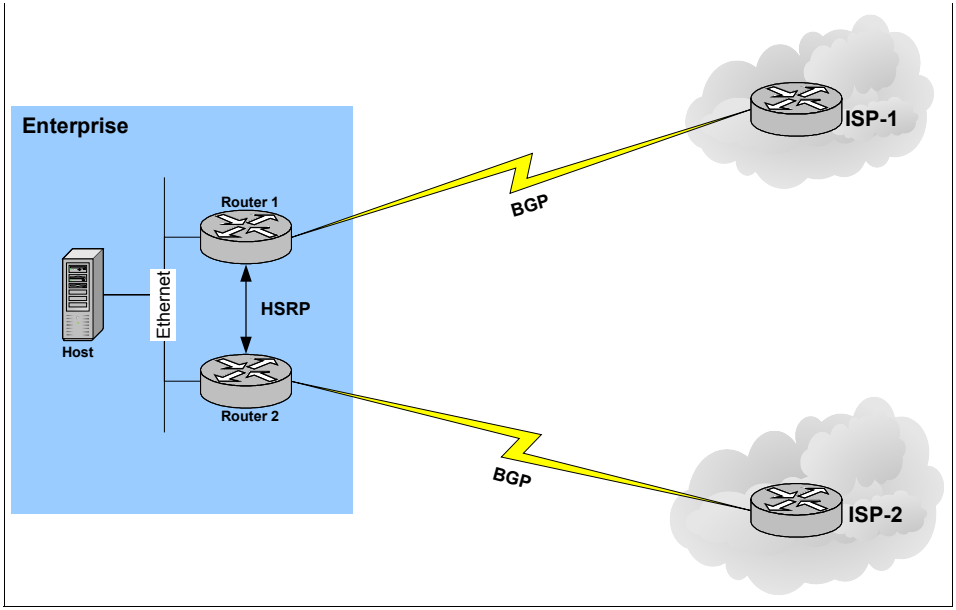


Figure 11-7 ISP and routers high availability

It is very important to provide routers with redundant components such as processors, fans, power supplies, management modules, etc.

b. Firewall - A possible solution to provide high availability and increase performance for the firewalls is the use of a load balance solution such as WebSphere Edge Server. Considering that a firewall has at least two

communication adapters, the WebSphere Edge Server has to take care of all interfaces and one possible implementation is the WebSphere Edge Server running in the same machine as the firewall. It is recommended that WebSphere Edge Server be dedicated for the firewalls.

c. Wireless Gateway (WebSphere Everyplace Connection Manager)- Figure 11-8 on page 253 shows a scenario of Wireless Gateways configured to distribute workload. Each box is a Wireless Gateway, configured as either a subordinate node or a principal node. The principal node is a Wireless Gateway configured to receive traffic from a mobile network connection and distribute the workload among its subordinate nodes. The subordinate nodes are configured to accept and process traffic from the principal node based on a configurable distribution algorithm. The distribution algorithms include:

   - Round-robin - The principal node continuously repeats the sequence of distributing traffic to a series of subordinate nodes, one after the other.
   - Weighted round-robin - The principal node continuously repeats the sequence of distributing traffic to a series of subordinate nodes, based on configurable CPU utilization thresholds, called low and high water marks.
   - Device/type of network connection based - The principal node distributes traffic to subordinate nodes based on the type of network connection or unique device identifier from which it came.

   To provide back-up for the principal node, High Availability Cluster Multi-Processing (HACMP) may be used with a second machine. It is not necessary to back up the subordinate nodes.

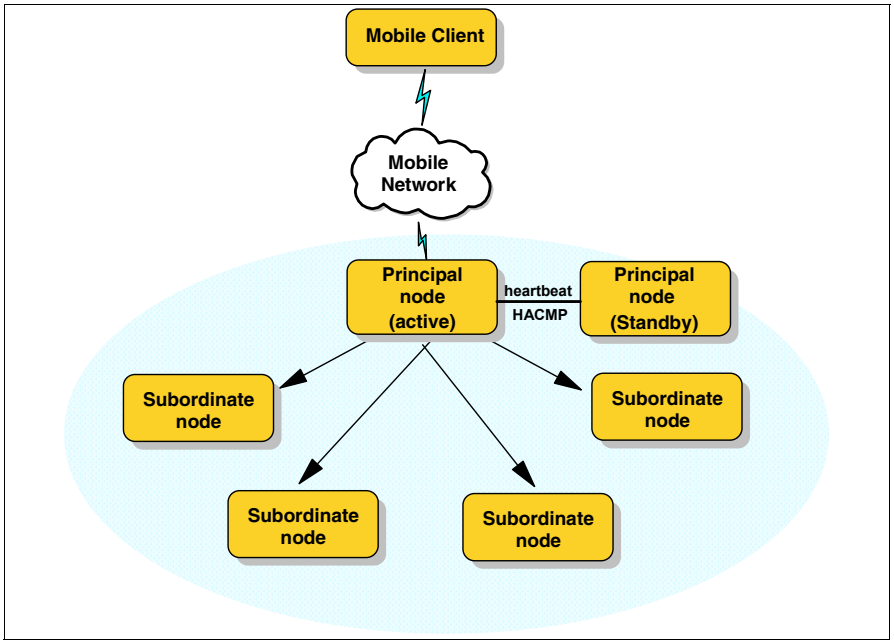


Figure 11-8 Wireless Gateway cluster solution

In order to improve performance on WAP services, the Wireless Gateway acting as a WAP gateway can use the HTTP proxy caching facilities provided by the Everyplace Wireless Gateway WML caching plug-in for IBM WebSphere Edge Server Caching Proxy (formerly Web Traffic Express). Caching facilities minimize network traffic and ensure that WAP clients spend less time retrieving repeated requests.

The Wireless Gateway provides a virtual private network tunnel between the wireless gateway and a wireless client, sensitive to the limited bandwidth capabilities and varying latencies found in wireless environments, so IP traffic can be routed efficiently and securely. In order to improve performance, this VPN solution implements data compression, IP header reduction, selective packet filtering, and TCP retransmission optimizations.

d. Load Balance (WebSphere Edge Server)- This supports high availability by the use of a second machine that monitors the main, or primary machine and stands by to take over the task of load balancing if the primary machine fails at any time. The standby machine is ready to take over and preserve connections in case the primary machine fails. If the back-up machine detects that the active machine has failed, it will take over and begin load balancing. At that point, the statuses of the two machines are reversed: the back-up machine becomes active and the

primary machine becomes standby. A “heartbeat” mechanism between the two Dispatcher machines is used to detect a Dispatcher failure. The standby machine makes the decision to take over if it detects that all heartbeats have stopped for two seconds. The standby machine then changes its state to active. It also broadcasts gratuitous Address Resolution Protocol (ARP) commands so that everyone on the subnet (including the router) will now send packets for the cluster addresses to the standby (now active) machine.

For detailed information about IBM WebSphere Edge Server, refer to the IBM Redbook *IBM WebSphere Edge Server: New Features and Functions in Version 2* - SG24-6511-00.

e. Authentication (Tivoli Access Manager - WebSeal).

Tivoli Access Manager is composed of:

- User Registry, which in most cases is implemented by an LDAP directory as IBM Secureway Directory. It supports the concept of master and replica LDAP servers. A master server contains the master directory from which updates are propagated to replicas. All changes are made and occur on the master server, and the master is responsible for propagating these changes to the replicas.
- The Policy Server, which supports the management of the authorization database and its distribution to Authorization Services.

  For availability purposes, a standby server can be configured to take over Policy Server functions in the event of a system failure. This can be supported using an appropriate high-availability product (for example, HACMP on AIX platforms).

  The Policy Server replicates the authorization database to all other Access Manager Authorization Servers. Every application, configured in local cache mode, that uses this Authorization Service (like WebSEAL) has its own local copy (replication) of the master authorization database and can therefore provide authentication and Authorization Services, even if the Policy Server is not available for a brief period of time.
- The Web Portal Manager, which provides a Web browser based capability for performing administrative functions. In order to provide availability, more than one Web Portal Manager can be deployed.
- The WebSEAL reverse proxy that is the front-end to the back-end Web servers and is responsible for applying the security policy. Increasing the availability of the WebSEAL starts with at least two front-end WebSEAL servers. Replicated front-end WebSEAL servers provide the site with load balancing during periods of heavy demand, as well as fail-over capability: if a server fails for some reason, the remaining

replica server(s) will continue to provide access to the site. The load balancing mechanism is handled by a mechanism such as the Network Dispatcher (WebSphere Edge Server).

In order to provide availability and performance to access the back-end Web server, WebSEAL supports defining more than one target server for a unique junction. In this case, WebSEAL can load balance among the servers, and if a back-end server is unavailable, WebSEAL can continue forwarding requests to the remaining servers for the junction. For situations where it is important that subsequent requests for a particular user continue going to the same back-end server, WebSEAL is capable of providing support using what are called *stateful junctions*.

For more detailed information on Tivoli Access manager, refer to 9.5.1, “Tivoli Access Manager and Single Sign-On” on page 210.

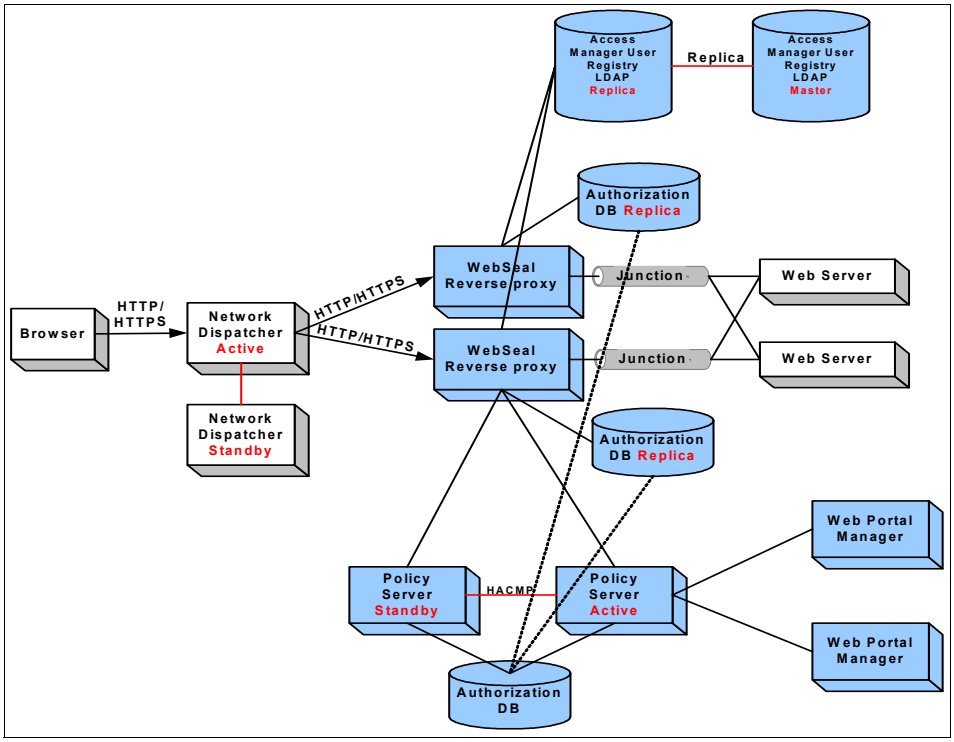


Figure 11-9 High Availability for the Tivoli Access Manager components

f. Web Server Redirector (Web Server and WebSphere HTTP plug-in)- This supports load balancing provided by the WebSphere Edge Server or the Authentication component (WebSeal). See Figure 11-10 on page 256 and Figure 11-11 on page 257.

g. Transcoding proxy (WebSphere Transcoding Publisher)- This supports load balancing provided by the WebSphere Edge Server.

h. Web application server - There are several topologies to implement the Web application server and provide availability and performance. For this solution, the separation of the HTTP server, the Application Server and the Database was chosen. For more detailed information on the existent topologies, refer to Chapter 10 of the IBM Redbook *IBM WebSphere V4.0 Advanced Edition Scalability and Availability*, SG24-6192-00.

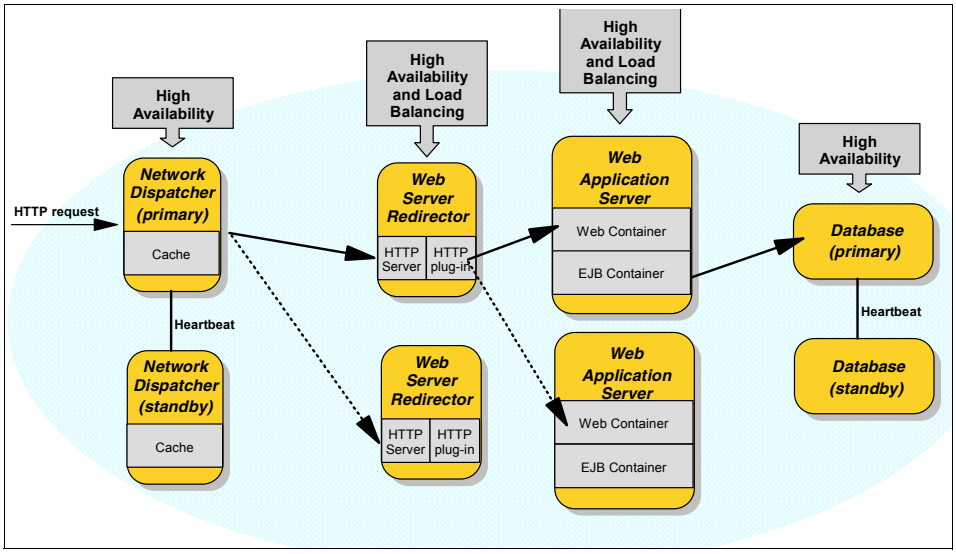


Figure 11-10 Web application server availability

In Figure 11-11 on page 257 the authentication node that provides load balance for the HTTP server is added. The authentication node is load balanced by the Network Dispatcher.

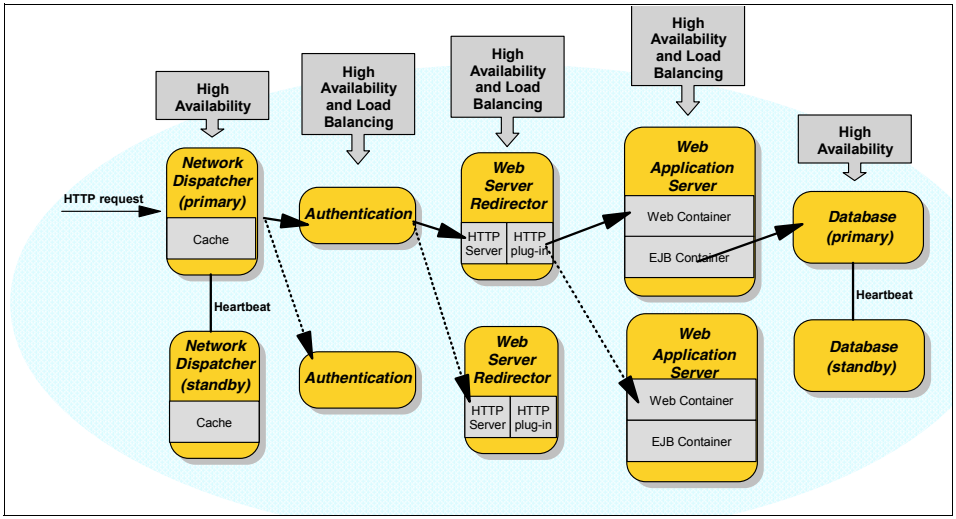


Figure 11-11 Web application server and Authentication proxy availability

The Portal Server, Device Manager and Synchronization are Web applications and use the solutions described in Figure 11-10 on page 256. These components support WebSphere Application Server cloning.

i. Portal (WebSphere Edge Server)- This supports load balancing provided by the WebSphere Edge Server or WebSphere HTTP plug-in.

j. Notification (Intelligent Notification Services)- This supports load balancing for the administration and user portlets using the same mechanism of WebSphere Portal Server.

k. Device Manager - This supports load balancing provided by the WebSphere Edge Server or WebSphere HTTP plug-in.

   Before a device (PDA) can use the dispatching capability, that device must be configured with the server URL of the network dispatcher cluster. In the dispatcher, a cluster is a group of TCP or UDP servers that are used for the same purpose and are identified by a single host name or IP address, the cluster address. A server URL is a Web address stored on a device and used by the device agent program to direct the device to a Device Management server, possibly through a network dispatcher (by using the host name of the dispatcher's appropriate cluster address) for job processing.

   When the device connects to the network dispatcher, the dispatcher forwards the HTTP request from the device to the most available Device Manager server. If jobs are scheduled for the device, the Device Manager server redirects the device back to its own server URL. This prevents the device from returning through the network dispatcher and being rerouted

to a different Device Manager server whenever multiple HTTP connections are required to complete the job.

l. Synchronization (Everyplace Synchronization Server)- This supports load balancing provided by the WebSphere Edge Server or WebSphere HTTP plug-in. In order to improve performance, it includes:

- A high performance cache for systems with heavy workloads. When caching is enabled, the Synchronization server replicates the back-end data locally in order to deal with client synchronization requests quickly.
- Filters that limit the synchronization to subsets of data. Filtering can also help control client database size by synchronizing the minimum amount of data necessary for a specific client.

m. Database - If the Relational Database Management System (RDBMS) is running on AIX, you can place the server in an HACMP cluster. If the application is using a database client that supports connection pooling and will reopen severed connections with the database, it is possible to have the database fail over without causing the application as a whole to fail.

n. LDAP Directory - A directory is often described as a database, but it is a specialized database that has characteristics which set it apart from general purpose relational databases. One special characteristic of directories is that they are accessed (read or searched) much more often than they are updated (written). Lightweight Directory Access Protocol (LDAP) is a fast growing technology for accessing common directory information. In a way similar to building an HA database for WebSphere, you can build an HA LDAP service with clustering software such as HACMP.

o. Storage - There are several modes and technologies of storage. The high availability can be provided by a redundant array of independent disks (RAID) and combined with performance and scalability provided by a storage area network (SAN).

2. Define the capacity and performance that are required for the solution. This can be done using:

– Benchmarks

- **TPC-C** - this is one of the most used benchmarks and relates to a computing environment where a population of users executes transactions against a database.The TPC-C is sponsored by the Transaction Processing Performance Council. For more details or information on other TPC benchmarks, refer to:

  `http://www.tpc.org`

- **SPECweb** - this is specifically used for Web solutions. There are already two versions of the SPECweb: the 96 and the 99 versions.

The SPECweb96 measures the number of Web operations (get, put) per second that a machine can perform. The SPECweb96 is widely used on the comparison and sizing of Web solutions and this pattern is being substituted for the SPECweb99 described below.

The SPECweb99 is the next generation of the SPEC benchmarking for WWW servers evaluation. Because of the quick advances of Web technologies, the benchmark SPECweb99 includes several improvements in order to respond to Web users' needs now and in the future. The SPECweb99 measures the number of Web pages per second that a machine can provide.

For more details, refer to:

http://www.spec.org/

– Solution and business requirements
  - User visits
  - Concurrent users
  - Response time
  - Peak time
  - Hits per second
  - Pageviews per second

Using this information, it is possible to size the necessary hardware and use the rights techniques to provide the desired performance. Sizing is not a precise method. It uses benchmarks and performs experiences as to how each component works under determined conditions. There are so many influencers that can change the performance of a solution, such as network latency, basic software, applications developed not using the best practices, that the sizing will be the first estimate to have the solution definitions.

IBM has for many years been designing and running e-Business solutions. With this experience, it was possible to create patterns as to the typical load and use of each type of solution. This intellectual capital was used to develop internal tools to help define a solution sizing with more accuracy. It is not the focus of this IBM Redbook to explain sizing techniques, but we do provide a sample using an internal IBM tool for a Pervasive Portal solution using IBM WebSphere Portal Server V4.1. It is assumed that:

– The solution will be 3-tier (Presentation, Application and Persistence)
– The performance objectives are: four user visits per second and a contingency percentage of 10%
– The 3-tier will be running on pSeries™ machines

– A load balance component such as WebSphere Edge Server will be provided

– There is a delay of two seconds between the server and the pervasive devices

The first step is to choose a pattern. In our case, it will be a Portal (WebSphere Portal Server V4.1).

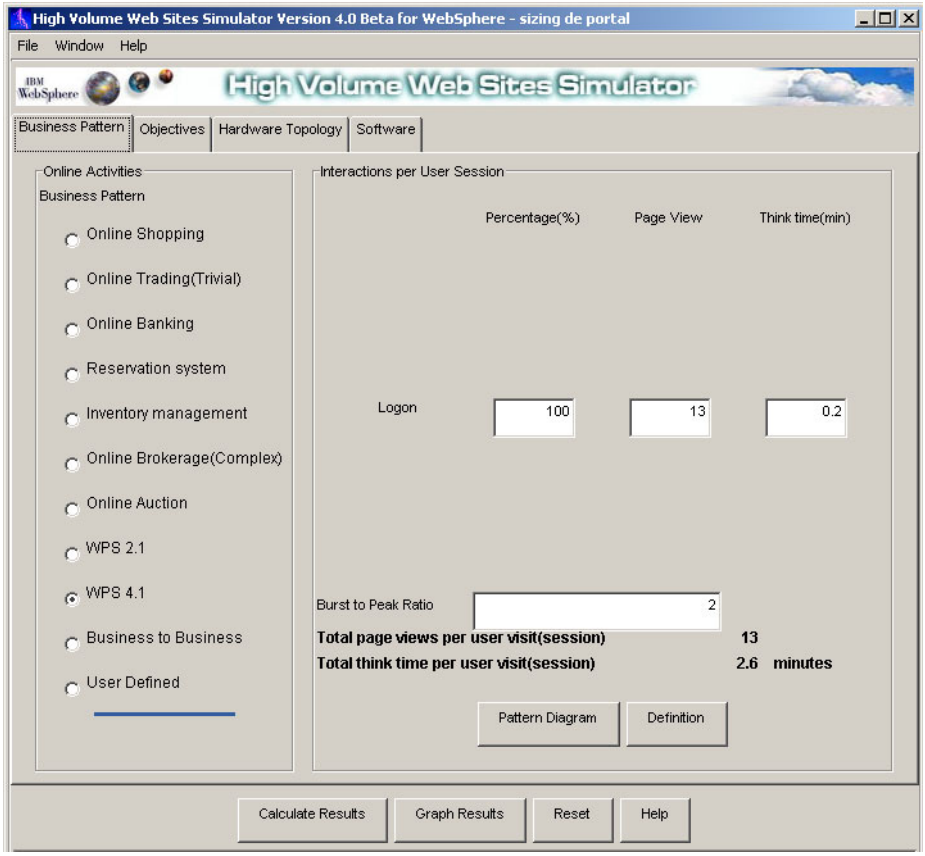


Figure 11-12 Choosing a pattern

Figure 11-13 on page 261 shows a Web portal pattern diagram.

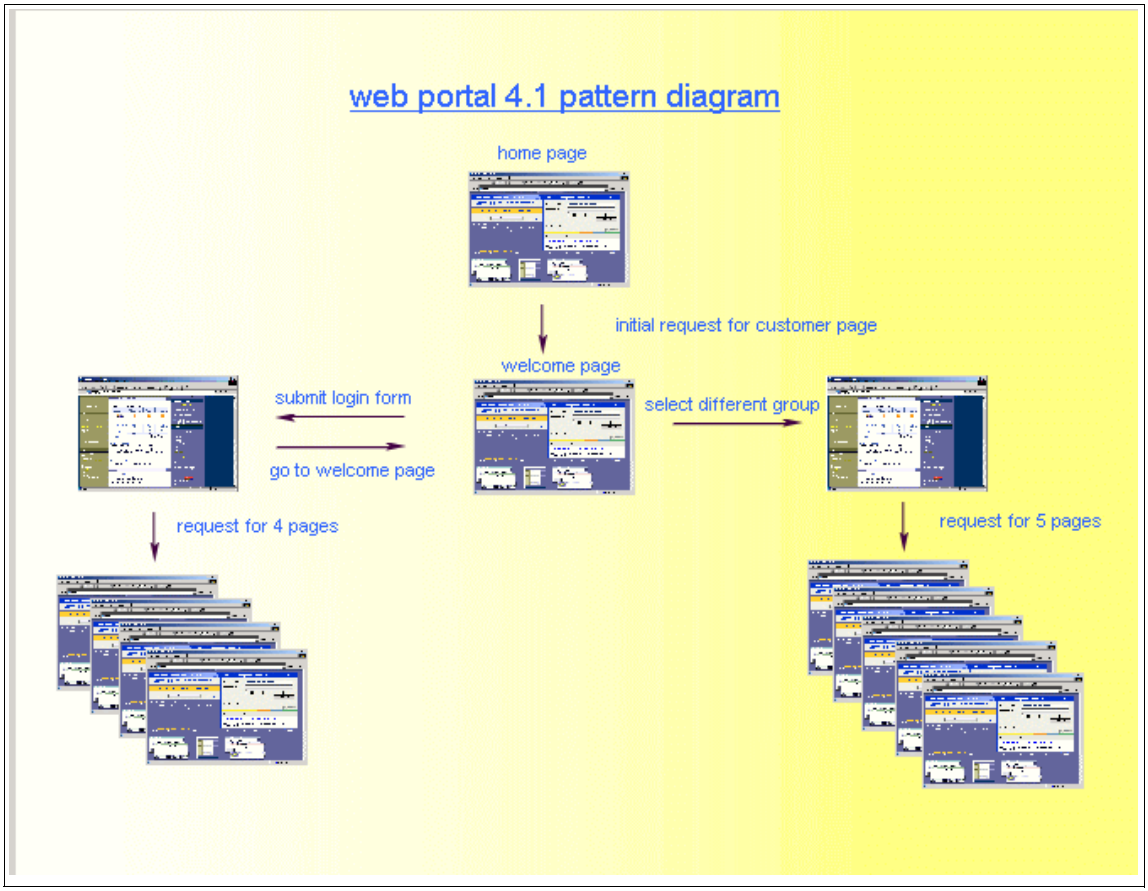


Figure 11-13 Portal pattern

The second step is to set the performance and capacity objectives and number of tiers.

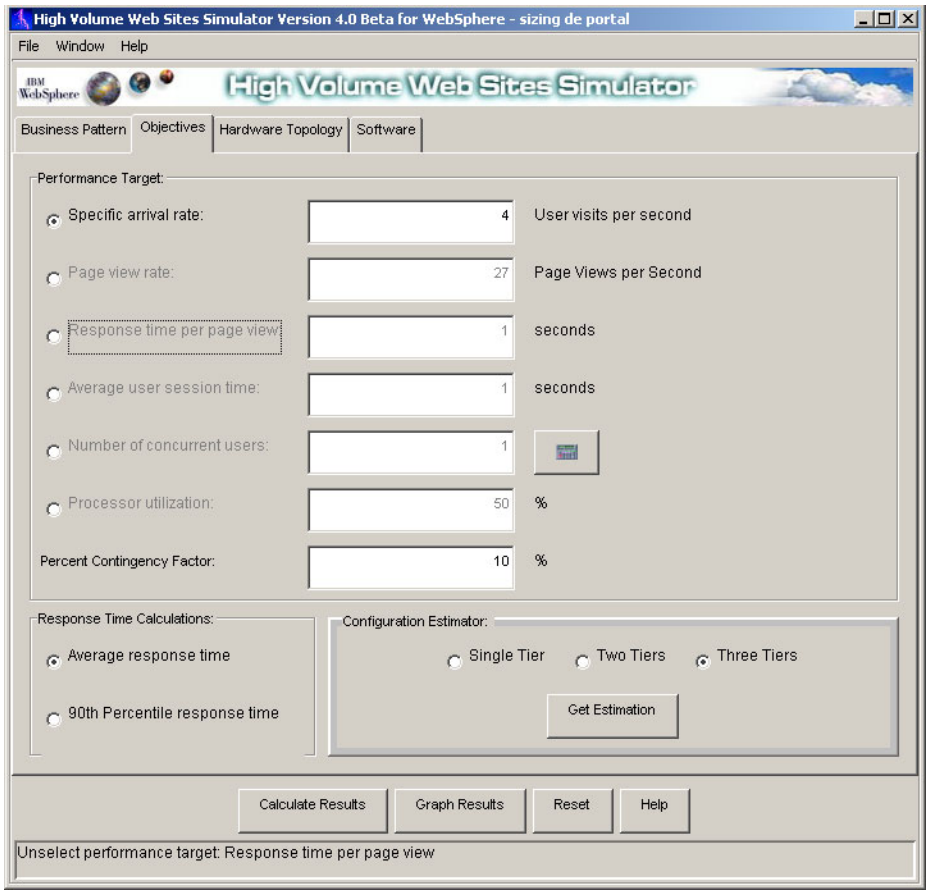


Figure 11-14 Setting Performance and capacity objectives

The third step is to set which version of WebSphere will be used and if it is being used with SSL.

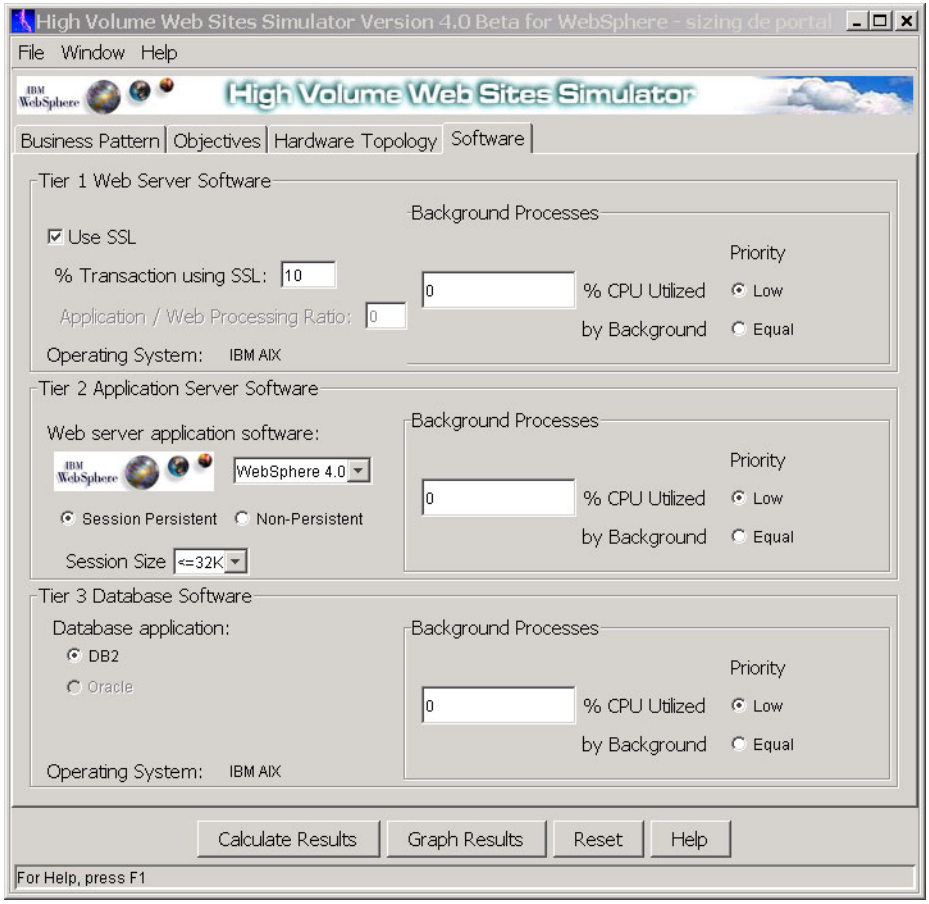


Figure 11-15 Software settings

The fourth step is to choose the hardware that will be used.

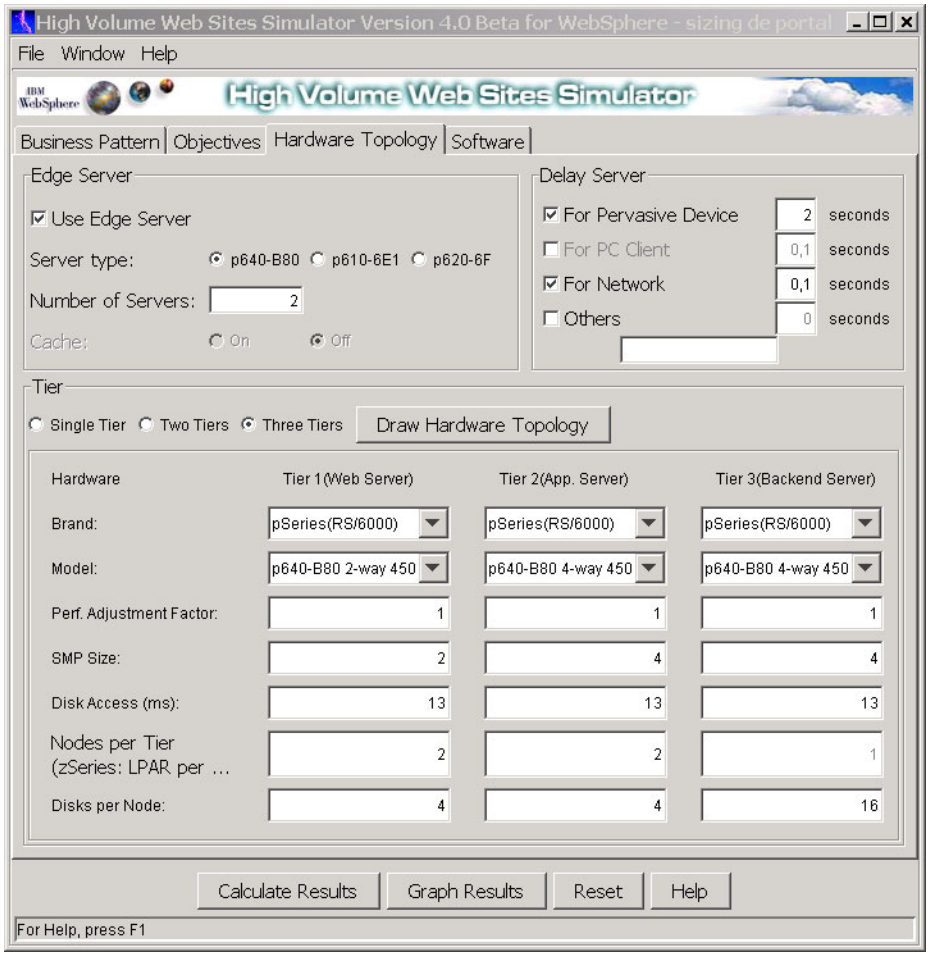


Figure 11-16 Hardware choice

The last step is to check whether the hardware and software definitions meet the performance objectives defined for the solution.

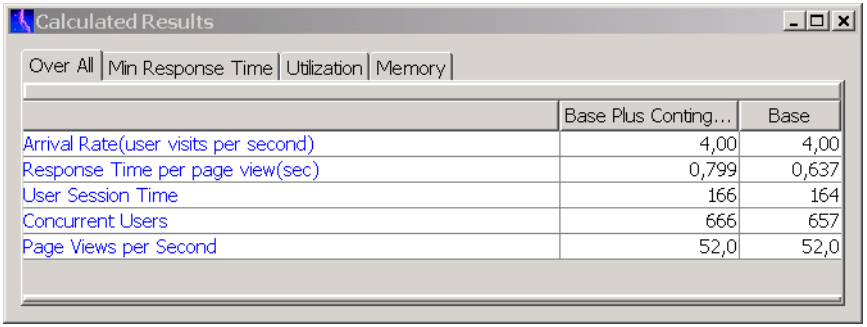


Figure 11-17 Performance results

It is possible to create graphics and visualize the capacity and performance of the solution.

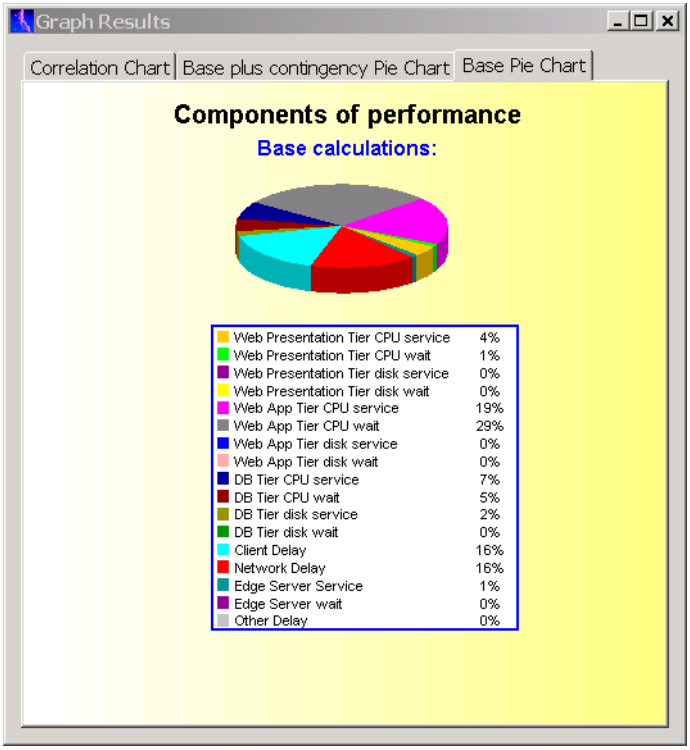


Figure 11-18 Performance graphics

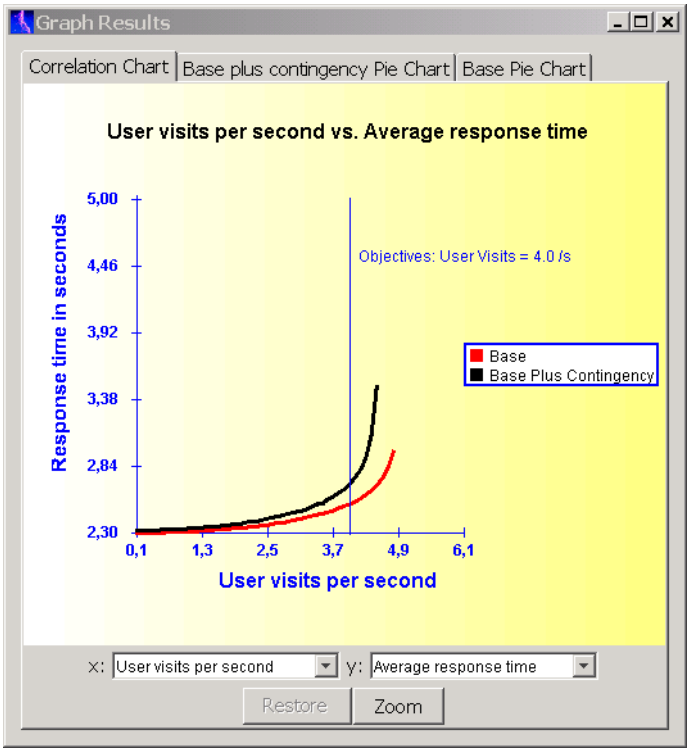


Figure 11-19 Performance graphics

11.5 Where to find more information

- For more information about WebSphere Edge Server, refer to the IBM Redbook *IBM WebSphere Edge Server: New Features and Functions in Version 2,* SG24-6511-00.
- For more information about HACMP, refer to the IBM Redbook *Configuring Highly Available Clusters Using HACMP 4.5*, SG24-6845-01.
- For more information about WebSphere Scalability and Availability, refer to IBM Redbook *IBM WebSphere V4.0 Advanced Edition Scalability and Availability*, SG24-6192-00
- For more information about Scalability and Availability, refer to IBM Redbook *Best Practices for High-Volume Web Sites*, SG24-6562-00

Part 3

Implementation

© Copyright IBM Corp. 2003. All rights reserved.

12

Technical scenario

In this chapter, we provide information on how to deploy the sample application in a runtime environment which was developed for the project.

© Copyright IBM Corp. 2003. All rights reserved.

12.1 Deploying the sample application

The following sections and step-by-step instructions will guide you through the sample application installation and configuration.

12.1.1 Prerequisites for the application

The sample application runs on WebSphere Everyplace Access V4.2. You can use the setup manager to install the product. For other installation details, refer to the product documentation or to the IBM Redpaper *IBM WebSphere Everyplace Access V4.1.1 Installation*, REDP3587.

The recommended configuration method for WebSphere Everyplace Access V4.2 is to use the setup manager provided with the product. This is the supported way of performing the installation; at this stage, there is no support for incremental installation on top of WebSphere Portal Server or other existing components.

12.1.2 Database configuration

The application requires a database populated with sample data. The following steps will help you to set up the sample database.

1. After you have downloaded SG246876.zip from the Redbooks site and followed the **Additional Materials** link, create a directory on your local hard drive (for example: c:\pervsamp) for the sample code, then unzip the file into the directory.
2. Open a DB2 command window.
3. Change the directory to the sample code directory: C:\SG246876\database.
4. Run the setup.bat script.
5. In order to populate the database, run the populate.bat script.
6. Close the window.

12.1.3 Installing the EJB components

The sample application consists of multiple parts. The back end is implemented as an EJB application. This section will guide you in installing the EJB components for the sample.

1. Create an operating system user *dbuser* with the password *password* for database access.
2. Launch the WebSphere Advanced Administrative Console.

3. Create a datasource under **Resources -> JDBC providers -> Sample DB Driver**.
   - Name: `Pervasive`
   - JNDI name: `jdbc/Pervasive`
   - databaseName: `PERPORDB`
   - user: `dbuser`
   - password: `password`
4. Click **Console -> Wizards -> Install Enterprise Application**.
5. Select the **Install stand-alone module (*.war, *.jar)** option. Browse for the EJB .jar file **PervasiveEJB.jar**. Provide the Application name: `PervasiveEJB`.
6. Click **Next** until you get to the Selecting Application Servers panel. Highlight the `PervasiveEJB` line, then select the **WebSphere Portal** server from this item. Click **Next**.
7. Click **Finish**.
8. Click **Yes** when installation asks whether to regenerate the application code. Set the database name to `PERPORDB`, set the schema name to `DBUSER`.
9. When the installation is complete, start the PervasiveEJB under the Enterprise Applications.
10. The portlets should find the EJB stubs and skeletons; you need to copy the file, PervasiveEJB.jar, to a place where portlets can load the class files. One suggestion is to use <WAS_HOME>\lib\ext to copy the .jar file over. You should restart your application server in order to get the class loader to load the new libraries.

12.1.4 Installing and configuring the portlets

A major part of the sample application is the portal application. This section will explain how you can install the portal application.

The first step is to install the portlet.

1. Open the browser and type:

   `http://<server_fully_qualified_name>/wps/portal`
2. Click the key on the right side of the page and log on as administrator (the default is wpsadmin) with a password (the default is wpsadmin).
3. In the navigation menu, select **Portal Administration** then select **Portlets -> Install Portlets**.

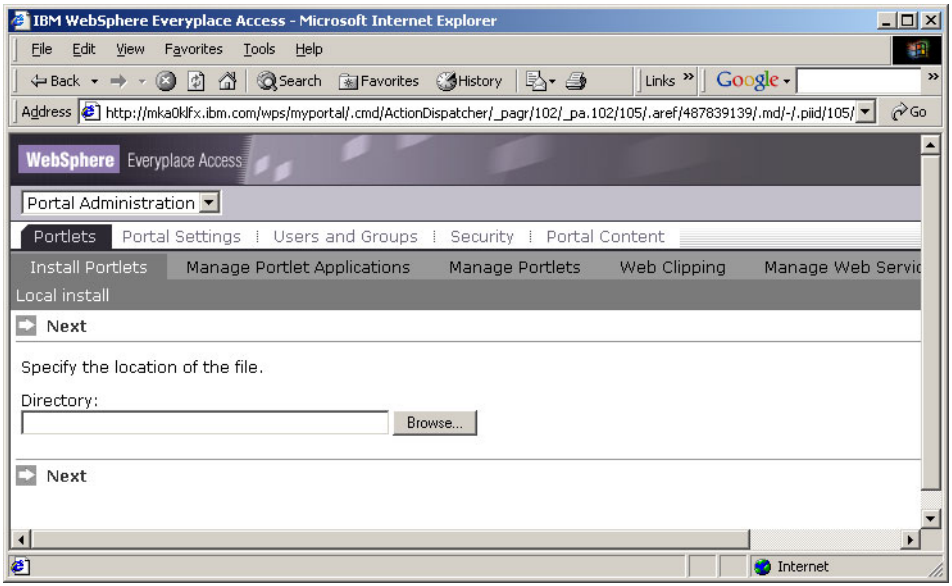


Figure 12-1 Install portlet

4. In the Directory field, type `C:\SG246876\PervasivePortal.war` and click **Next**.
5. You should see one list with one portlet application (PervasivePortal application) and portlets (we have four portlets: ITSO close defect and invoice customer portlet, ITSO search knowledge portlet, ITSO register defect portlet, ITSO list problem pending portlet). Click the **Install** link.
6. You need to create a place to insert the portlets that you have created in the previous step. In the navigation menu, select **Work With Pages** then click **Manage Place and Pages**.
7. Click **Create Place**.
8. In the Place name and default locale title field, type `Help Desk ITSO Application` and for Support markups, select the **html**, **wml** and **chtml** options and click **OK**.

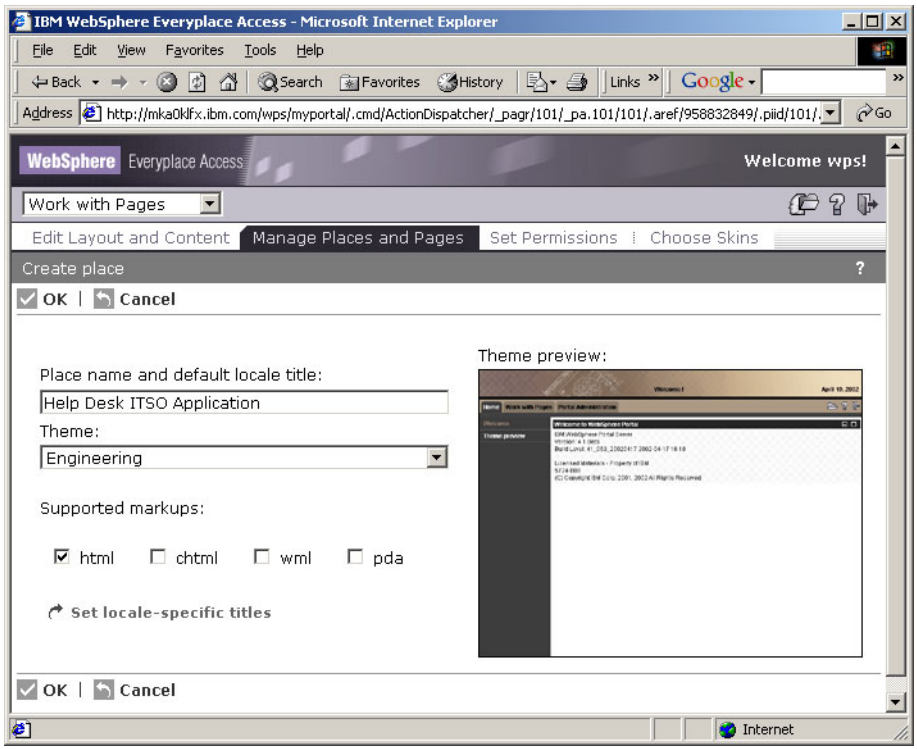


Figure 12-2 Create a place

9. Click the **Manage Places and Pages** tab, click **Create page** then select **Create new**.

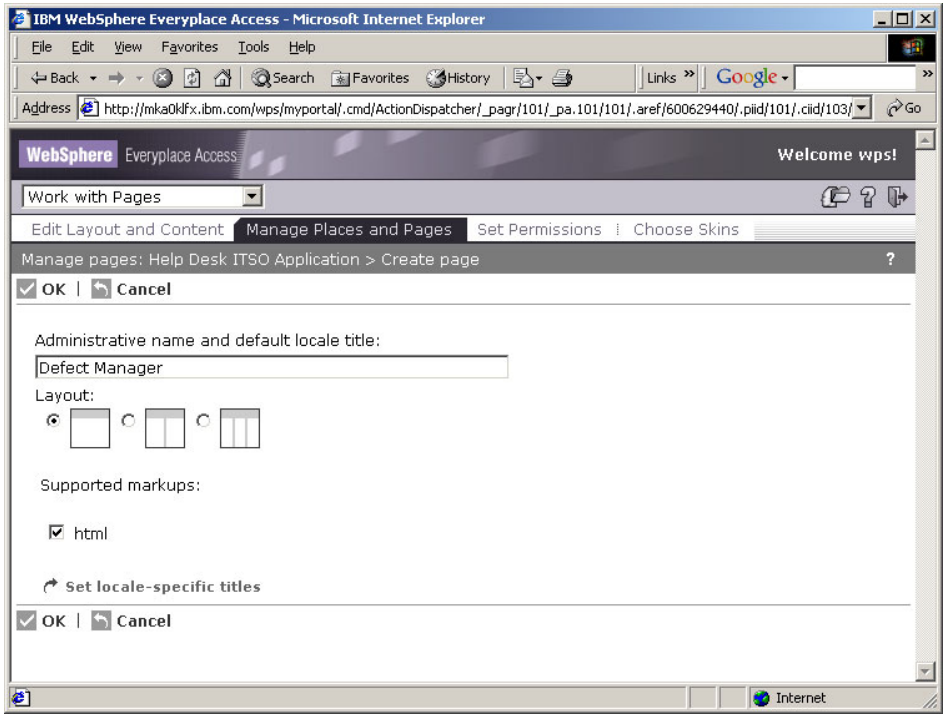


Figure 12-3 Create a page

10. In the Administrative name and default locale title field, type `Defect Manager`.
11. For the layout, select the first option.
12. For the Supported markups, select **html**, **chtml** and **wml** and click **OK**.
13. Click the **Edit and Layout Content** tab and select the place and the page that you have created.

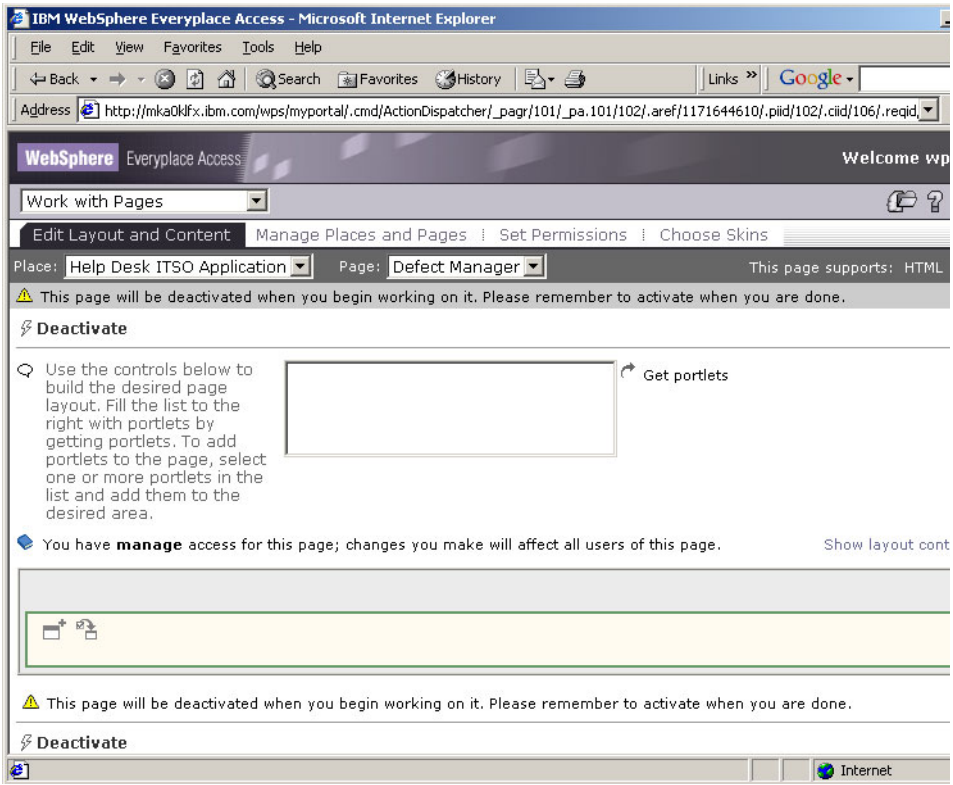


Figure 12-4 Page to define the lay out for portlets

14. Click **Get portlets**.
15. On the next page, select the **Show all portlets** option and click the **Go** link.
16. You will see the list of portlets in your WebSphere Portal Server. You should find the portlets for this application and click the plus sign next to the items to add them to the Portlet list (you can find the name of the portlets in step 5).

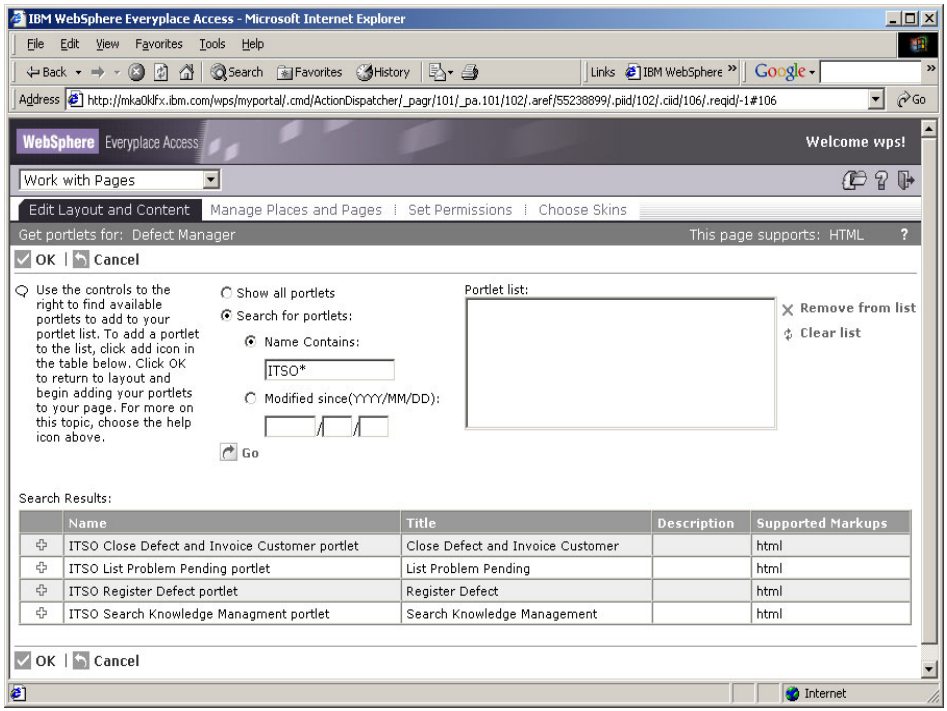


Figure 12-5 Page for select the portlets

17. Click **OK** after selecting all four portlets.

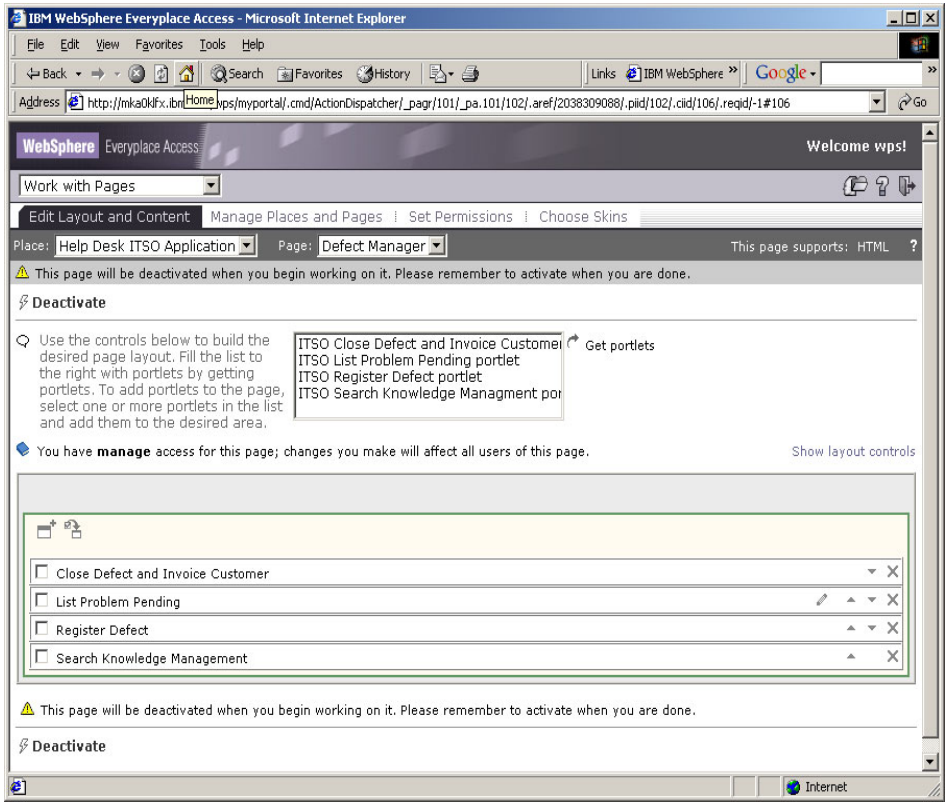


Figure 12-6 Page after selecting the portlets

18. Select the portlets one by one and click the button (the small window with the plus sign). By adding the portlets to the layout, you also set the sequence according to which the portlets will show on the page.
19. At the end, click the **Activate** link to active this place.
20. The next step is give the permission to portal users to access the place. On the navigation menu, select **Portal Administration** and then click the **Security** tab.
21. Select the **Selected users and groups** option and click **Get Users and Groups**.
22. Select the **Search for groups** option and search for the group Technician, then click **OK**.
23. Under Select the objects for the permissions, select **Pages**.
24. Type `Defect Manager` for the Name Contains field then click **Go**.

25. Under the Minimum column, select **View** for the Help Desk ITSO Application and the Defect Manager then click **Save**.

26. Under Select the objects for the permissions, choose **Portlets** this time.

27. Type `ITSO` for the Name Contains field then click **Go**. You should see the list of ITSO portlets for the application.

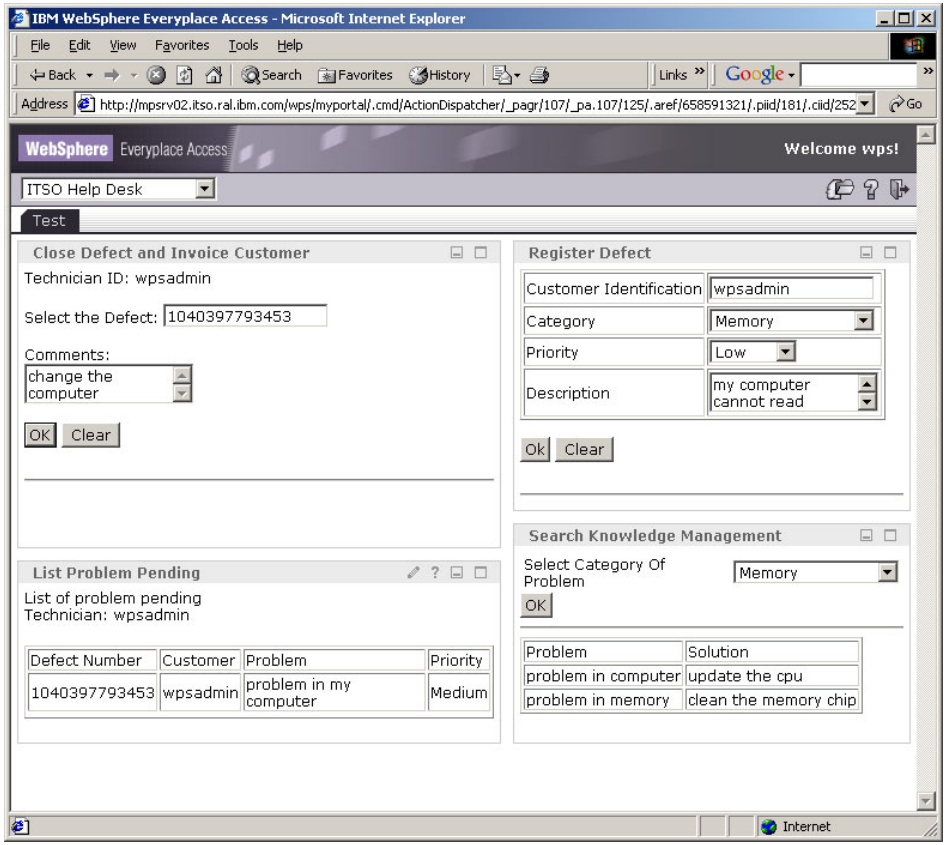


Figure 12-7 ITSO Web Application

12.1.5 Application users

The users for the application are stored in an LDAP directory. In order to create the necessary users and groups, refer to 8.3.2, “User registry” on page 176.

12.1.6 Mobile client application and database synchronization

The sample application also has a mobile client application for database synchronization.

The client application for PalmOS is distributed together with the source code. For detailed installation steps, refer to 8.3.3, “Using Transcoding Technology” on page 178.

On the server side, the mobile client application requires a synchronization server. For detailed configuration steps, refer to 8.5.1, “Using DB2 Everyplace” on page 185.

Part 4

Appendixes

© Copyright IBM Corp. 2003. All rights reserved.

A

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/SG6876

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, SG246876.

© Copyright IBM Corp. 2003. All rights reserved.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
sg246876.zip	Sample application for the book

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	10MB at least
Operating System:	Windows 2000 Server
Processor:	Intel Pentium® III or 4, 1GHz or higher
Memory:	1GB or more

How to use the Web material

Create a subdirectory (folder) on your workstation, for example: `C:\SG246876`, and unzip the contents of the Web material zip file into this folder.

For more information about how to install and configure the sample application refer to Chapter 12, “Technical scenario” on page 271.

Abbreviations and acronyms

ACL	Access Control List
API	Application Programming Interface
BA	Basic Authentication
CDC	Connected Device Configuration
CDLC	Connected Limited Device Configuration
CDMA	Code-Division Multiple Access
CDPD	Cellular Digital Packet Data
CGI	Common Gateway Interface
CHTML	Compact HTML
CICS®	Customer Information Control System
CMP	Container Managed Persistency
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
DHTML	Dynamic HTML
DMZ	Demilitarized Zone
DN	Distinguished Name
DNS	Domain Name Server
DOM	Document Object Model
DTD	Document Type Definition
DTMF	Dual Tone Multi-Frequency
EAI	Enterprise
EDI	Electronic Data Interchange
EJB	Enterprise Java Bean
ERP	Enterprise Resource Planning
GOF	Gang of Four
GPRS	General Packet Radio Service
GSM	Global System for Mobile telecommunication
GUI	Graphical User Interface
HA	High Availability
HACMP	High Availability Cluster Multiprocessing
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IIOP	Internet Inter-ORB Protocol
IM	Instant Messaging
IP	Internet Protocol
IrDA	Infrared Data Association
ISP	Internet Service Provider
ITSO	International Technical Support Organization
JAAS	Java Authentication and Authorization Service
JAR	Java Archive
JDBC	Java Database Connection
JDK	Java Development Kit
JNDI	Java Naming and Directory Interface
JSP	JavaServer Page
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
LTPA	Lightweight Third Party Authentication
OS	Operating System
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure

© Copyright IBM Corp. 2003. All rights reserved.

PSTN	Public Switched Telephone Network
RDBMS	Relational Database Management System
RMI	Remote Method Invocation
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
SSO	Single Sign-On
TAI	Trust Association Interceptor
TDMA	Time Division Multiple Access
UDDI	Universal Description Directory and Integration
URL	Unified Resource Locator
VPN	Virtual Private Network
WAN	Wide Area Network
WAP	Wireless Access Protocol
WEA	WebSphere Everyplace Access
WML	Wireless Markup Language
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language

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 292.

- *A Portal composite pattern Using WebSphere Portal V4.1*, SG24-6869.
- *Access Integration Pattern Using IBM WebSphere Portal Server*, SG24-6267.
- *Self-Service Patterns using WebSphere Application Server V4.0*, SG24-6175.
- *Patterns: Connecting Self-Service Applications to the Enterprise*, SG24-6572
- *Self-Service Applications using IBM WebSphere V4.0 and IBM MQSeries Integrator*, SG24-6160
- *Patterns on z/OS: Connecting Self-Service Applications to the Enterprise*, SG24-6827
- *Patterns: Building Messaging-based and Transactional Applications*, SG24-6875
- *User-to-Business Pattern using WebSphere Personalization Patterns for e-business Series*, SG24-6213
- *Mobile Commerce Solutions Guide using WebSphere Commerce Suite V5.1*, SG24-6171
- *Mobile Applications with IBM WebSphere Everyplace Access Design and Development*, SG24-6259
- *WebSphere Personalization Solutions Guide*, SG24-6214
- *Enterprise Business Portals with IBM Tivoli Access Manager*, SG24-6556
- *IBM WebSphere V4.0 Advanced Edition Handbook*, SG24-6176
- *WebSphere Application Server Test Environment Guide*, SG24-6817
- *IBM WebSphere Edge Server: New Features and Functions in Version 2*, SG24-6511
- *Configuring Highly Available Clusters Using HACMP 4.5*, SG24-6845

© Copyright IBM Corp. 2003. All rights reserved.

- *IBM WebSphere V4.0 Advanced Edition Scalability and Availability*, SG24-6192
- *Best Practices for High-Volume Web Sites*, SG24-6562
- *Transcoding Technologies in IBM WebSphere Everyplace Access Version 4.1.1*, REDP3592
- *Relational Database Synchronization in WebSphere Everyplace Access V4.1.1*, REDP3590
- *User-to-Business Patterns Systems Management Guidelines*, REDP0401
- *IBM WebSphere Everyplace Access V4.1.1 Installation*, REDP3587

Referenced Web sites

These Web sites are also relevant as further information sources:

- Patterns for e-business Web site

  `http://www.ibm.com/developerWorks/patterns`
- Symbian's corporate Web site

  `http://www.symbian.com`
- Sun's J2ME Web site

  `http://java.sun.com/j2me`
- Qualcomm's BREW Web site

  `http://www.qualcomm.com/brew`
- IRDA organization's Web site

  `http://www.irda.org`
- Bluetooth official Web site

  `http://www.bluetooth.com`
- IEEE official Web site

  `http://www.ieee.org`
- ITU Official Web site

  `http://www.itu.int`
- ETSI official Web site

  `http://www.etsi.org`
- UMTS official Web site

  `http://www.umts-forum.org`

- 3GPP official Web site

  `http://www.3gpp2.org`
- W3C's official Web site

  `http://www.w3c.org`
- WAP Forum's official Web site

  `http://www.wapforum.org`
- Open Mobile Alliance Web site

  `http://www.openmobilealliance.org`
- Tivoli's Web site

  `http://www.tivoli.com`
- IBM's Web site

  `http://www.ibm.com`
- Lotus's Web site

  `http://www.lotus.com`
- TPC's official Web site

  `http://www.tpc.org`
- SPECweb's official Web site

  `http://www.spec.org`
- IBM Redbook's Web site

  `http://www.redbooks.ibm.com`
- Microsoft's PocketPC Web site

  `http://www.microsoft.com/mobile/pocketpc`
- Microsoft's Smartphone Web site

  `http://www.microsoft.com/mobile/smartphone`
- PalmOS Web site

  `http://www.palmsource.com`
- Wi-Fi Allieance Web site

  `http://www.weca.net`
- Apache's XML Web site

  `http://xml.apache.org`

- VoiceXML Web site

  `http://www.voicexml.org`

- SyncML official Web site

  `http://www.syncml.org`

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the **CD-ROMs** button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

© Copyright IBM Corp. 2003. All rights reserved.

D

E

F

G

H

I

J

L

M

N

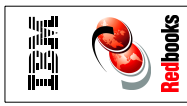
O

P

R

S

X



Redbooks

Patterns: Pervasive Portals Patterns for e-business series

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Patterns: Pervasive Portals
Patterns for e-business Series

Using the Access Integration pattern to build Pervasive Portal solutions

Using WebSphere Everyplace Access V4.2

Technical scenario with a sample application

This IBM Redbook focuses on the Access Integration pattern, specifically on portals with pervasive access. The book is a valuable source for IT architects, IT specialists, application designers, application developers and consultants who wish to know more about Pervasive Portal solutions. The application framework for this book includes WebSphere Portal and WebSphere Everyplace Access.

Part 1, “Patterns for e-business”, introduces the Patterns for e-business concept, focusing particularly on the Access Integration pattern.

Part 2, “Guidelines”, provides guidelines for Pervasive Portal applications, including application design and development, and some of the non-functional requirements for such applications, including security, system management and performance.

In Part 3, “Implementation“, you will find details on how to set up and configure a system for the sample application introduced in this book.

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

SG24-6876-00 ISBN 0738427772