WebSphere® software

IBM

Patterns: Pervasive and Rich Device Access Solutions

- Using Patterns for e-business to design pervasive solutions
- WebSphere Everyplace Access and related products
- Sample applications for various scenarios

Peter Kovari
Phillip Dermody
Daniel Ehrle
Stefan Fassmann
Richard Jacks
George Kroner
LindaMay Patterson
Sami Serpola

Redbooks

ibm.com/redbooks

International Technical Support Organization

Patterns: Pervasive and Rich Device Access Solutions

March 2005

SG24-6315-00

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

First Edition (March 2005)

This edition applies to WebSphere Everyplace Access V5.0 on Windows 2000 Server and AIX 5.2 platforms; WebSphere Studio Application Developer V5.0 on Windows and Linux platforms; WebSphere Studio Device Developer V5.7 on Windows platform; and Worplace Client Technology, Micro Edition on Windows platform.

Contents

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.

Trademarks

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

AIX®
Approach®
Cloudscape™
DB2®
developerWorks®
Domino®
DPI®
@server®
eServer™
Eserver™
Everyplace®
ibm.com®
IBM®
Lotus Notes®
Lotus®
Notes®
PartnerWorld®
Perform™
Rational®
Redbooks (logo) ™
Redbooks™
Sametime®
Tivoli®
TME®
ViaVoice®
WebSphere®
z/OS®

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 is part of the Patterns for e-business series. The focus of this book is to provide an in-depth overview of the currently available pervasive solutions using various IBM products.

Part 1, “Pervasive solution patterns” on page 1, discusses the application patterns, runtime patterns, and runtime Product mappings for each of the scenarios introduced in the third part of the book. Pervasive solutions are Access Integration solutions that focus on front-end integration. The infrastructure with which pervasive solutions must integrate can be thought of as a “black box,” which can be implemented in a number of ways. For example, it can be a Web server, a Web application server, or a portal server. The black box used in this book was implemented as a portal server and color-coded blue in the figures. Readers with a different black box implementation should be able to apply the delta orange nodes and Product mappings to their own configurations.

Part 2, “Guidelines” on page 111, is a collection of application design and development guidelines that can be applied to each of the scenarios.

Part 3, “Scenario implementations” on page 149, consists of eight different pervasive scenairos. Each scenario provides technical details and samples for implementation.

This book is primary a source for IT architects and IT specialists planning to design or implement pervasive solutions.

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.

The Team (lef to right): Phillip Dermody, Stefan Fassmann, Sami Serpola, Daniel Ehrle, LindaMay Patterson, Richard Jacks, Peter Kovari, George Kroner

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.

Phillip Dermody is a Speech Solutions Specialist working in Technical Sales in Pervasive Computing, Software Group, ANZ. He joined IBM in 2003 and has been working with the full range of WebSphere voice and speech products with both financial groups and with Telecoms. Prior to joining IBM, Phillip was an R&D manager in speech technology and speech user interfaces, and also worked as a consultant designing and technically managing a range of speech-based projects, including the development of voice portals. Phillip works with customers on underlying speech technology and interface requirements, architectures for speech deployment, and product solutions. He has a PhD from the University of NSW.

Daniel Ehrle is an IT Specialist in the software group for pervasive computing in Switzerland. After he received his degree in computer science in 1996, he joined IBM as a Security Specialist in the security services group of IBM Global Services. He changed to the software group as technical presales for WebSphere in 2000 and specialized in pervasive computing in 2002. Since then he has been engaged in various customer pilots and proof of concepts.

Stefan Fassmann is an IT Specialist in the Pervasive Computing service group at the IBM Lab Boeblingen, Germany. After he received his degree in electrical

engineering in 1996, during which he specialized on radio networks, he started working on mobile workforce and remote access solutions in the mobile computing group of IBM Global Services. He changed to the Pervasive Computing service group in the IBM Lab in Boeblingen in 2000. Since then he has been engaged in various customer projects using embedded software client technology.

Richard Jacks is an IT specialist in the Pervasive Comupting services group at the IBM Lab in Hursley, UK. After he received his degree in Computer Science in 2001, he started working on WebSphere MQ products within IBM Software Group. In 2003 he moved to the Pervasive Computing services group. He has since been engaged in various customer projects using the WebSphere Everyplace suite of products.

George Kroner is a second year Co-op IT Specialist at the IBM ITSO Center in Raleigh, North Carolina. He is currently pursuing a Bachelor of Science degree in Information Sciences and Technology at Pennsylvania State University. His interests include mobile computing, Web applications, and intelligent interfaces.

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

Sami Serpola is a IT Architect in Helsinki. He has over five years of experience with mobile architectural infrastructure design. Sami has worked closely with Customers and Business Partners to integrate and migrate their existing environments and solutions towards the WebSphere family. Sami's areas of expertise include Internet technologies and protocols, mobile infrastructures with an emphasis on remote synchronization, authentication and authorization, and mobile solutions usability and interfaces.

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

Juan Rodriguez
Margaret Ticknor
Jeanne Tucker
International Technical Support Organization, Raleigh Center

Julie Czubik
International Technical Support Organization, Poughkeepsie Center

Jonathan Adams
Joan Boone
Curtis Ebbs
Bill Glendenning
Joe Hansen
Nichelle Hopson
Christian L Hunt
Christian Kirsch
David Lection
Peggy Nethery
Eric Otchet
Jake Palmer
Dr. Werner Schollenberger
Kermit Taylor
Dave Van Voorhis

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

Pervasive solution patterns

1

Patterns for e-business

This publication 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, and 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 helps 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.

The IBM Patterns for e-business are further augmented with guidelines and related links for their 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 pattern's process, the architect can further define the additional functional pieces that the application will need to succeed. Finally he can build the application using the 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 reuse 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 such 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.

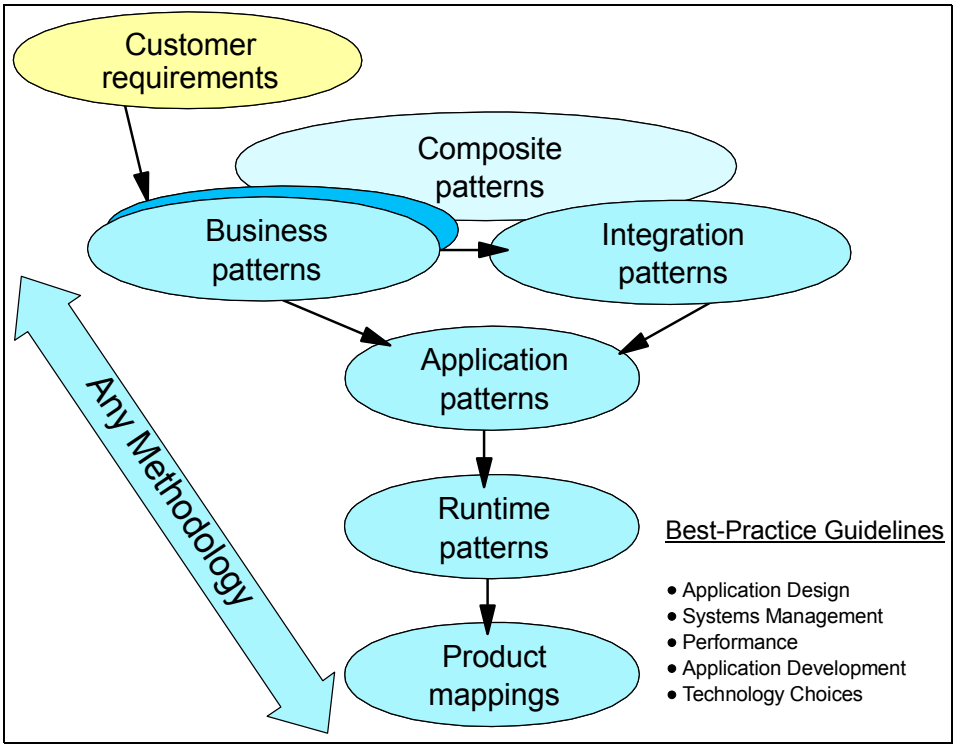


Figure 1-1 The Patterns for e-business layered asset model

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 is a layered structure where each layer builds details 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 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 get 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, they will be able to cut back significantly on the resources spent handling this by phone. The objective is to allow policy holders 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 us 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 explained 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 the iconic representation shown in Figure 1-4.

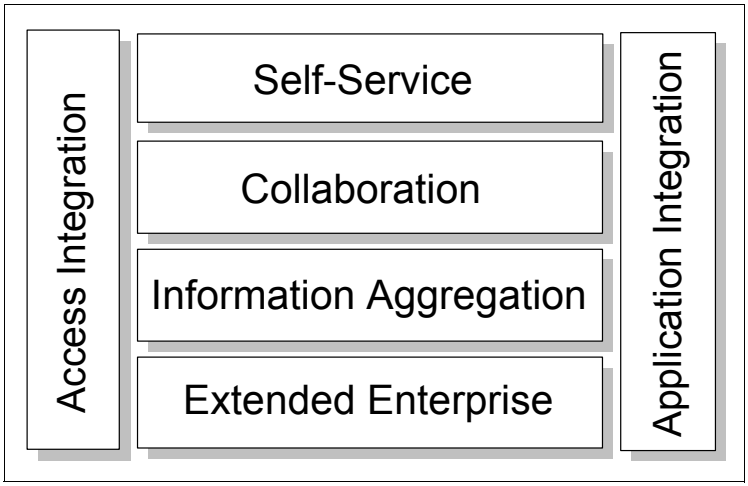


Figure 1-4 Patterns representing a custom design

If any of the Business or Integration patterns are not used in a custom design, we show that in the figures by making a block or blocks lighter than another. For example, Figure 1-5 shows a custom design that does not have a Collaboration business pattern or an Extended Enterprise business pattern for a business problem.

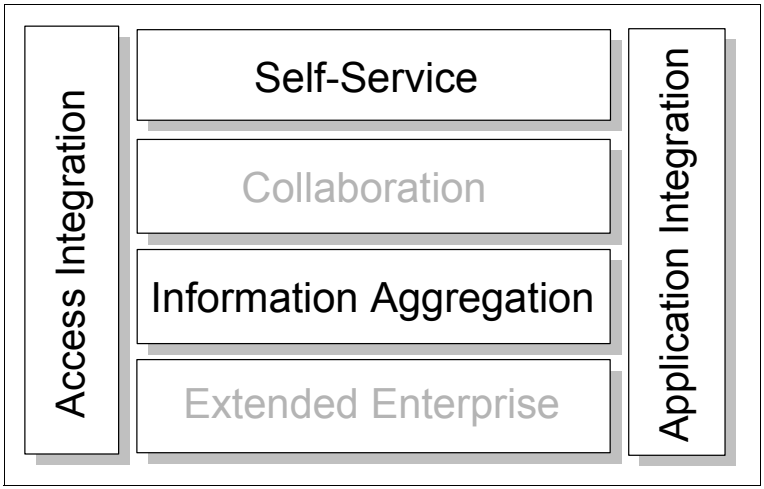


Figure 1-5 Custom design with Self-Service, Information Aggregation, Access Integration, and Application Integration

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	Provides 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 Self-Service::Directly Integrated Single Channel application pattern shown in Figure 1-7 fulfills this requirement.

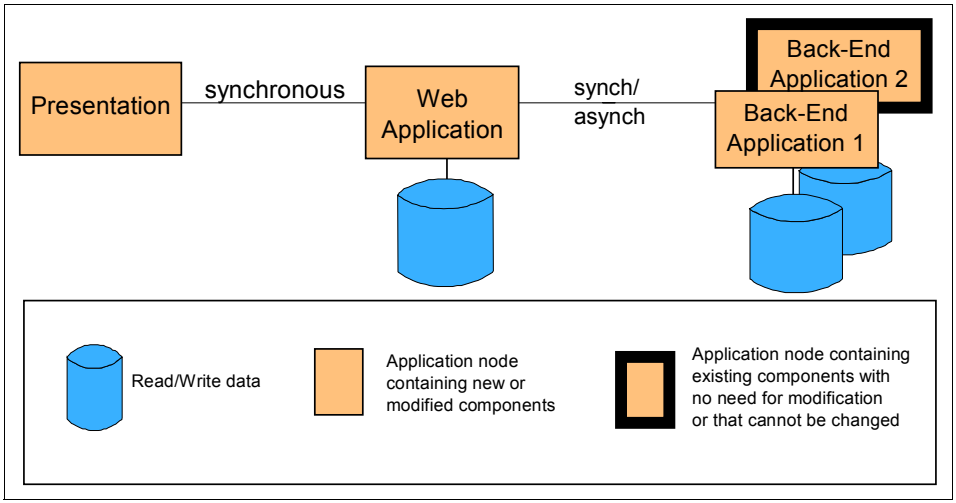


Figure 1-7 Self-Service::Directly Integrated Single Channel

The Application pattern shown in Figure 1-7 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 us 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 Self-Service::Decomposition application pattern shown in Figure 1-8 would be more appropriate.

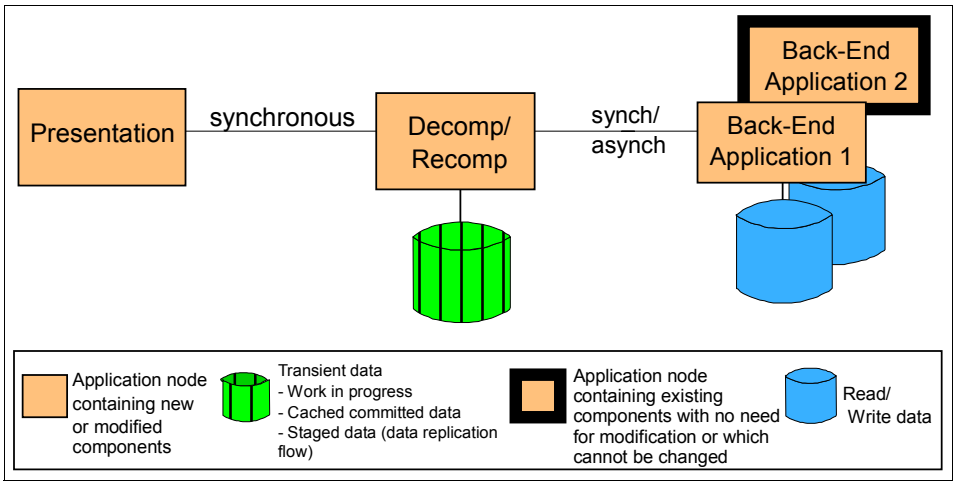


Figure 1-8 Self-Service::Decomposition

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 coexist 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 us 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 which Runtime pattern is the 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.

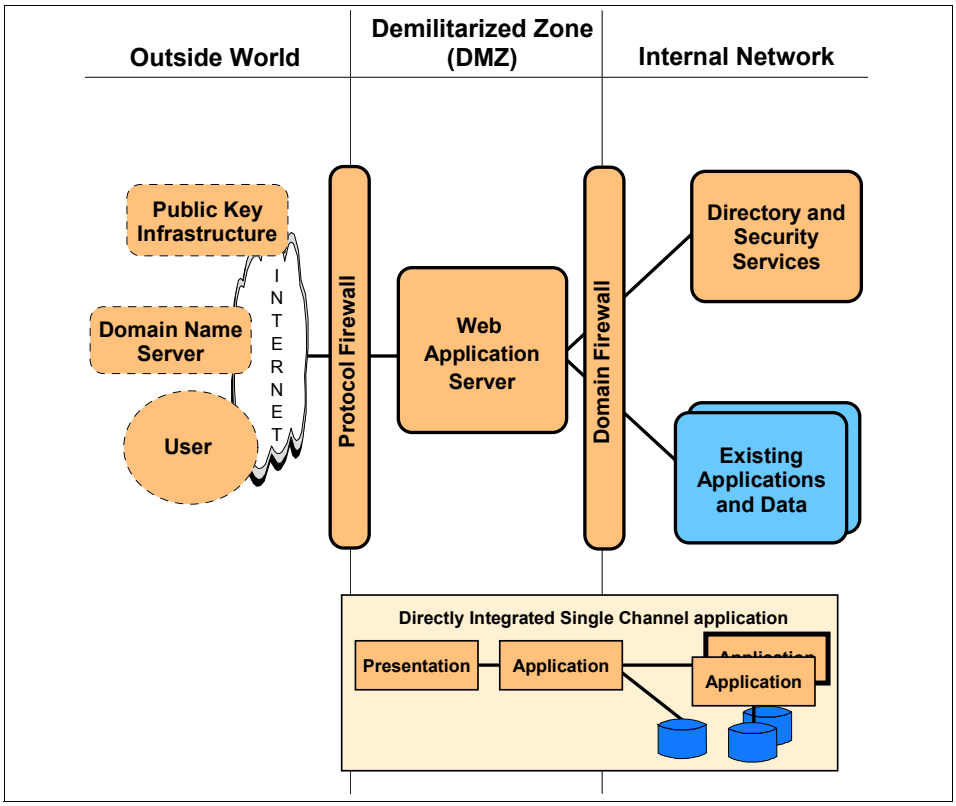


Figure 1-9 Directly Integrated Single Channel application pattern::Runtime pattern

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 of 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.

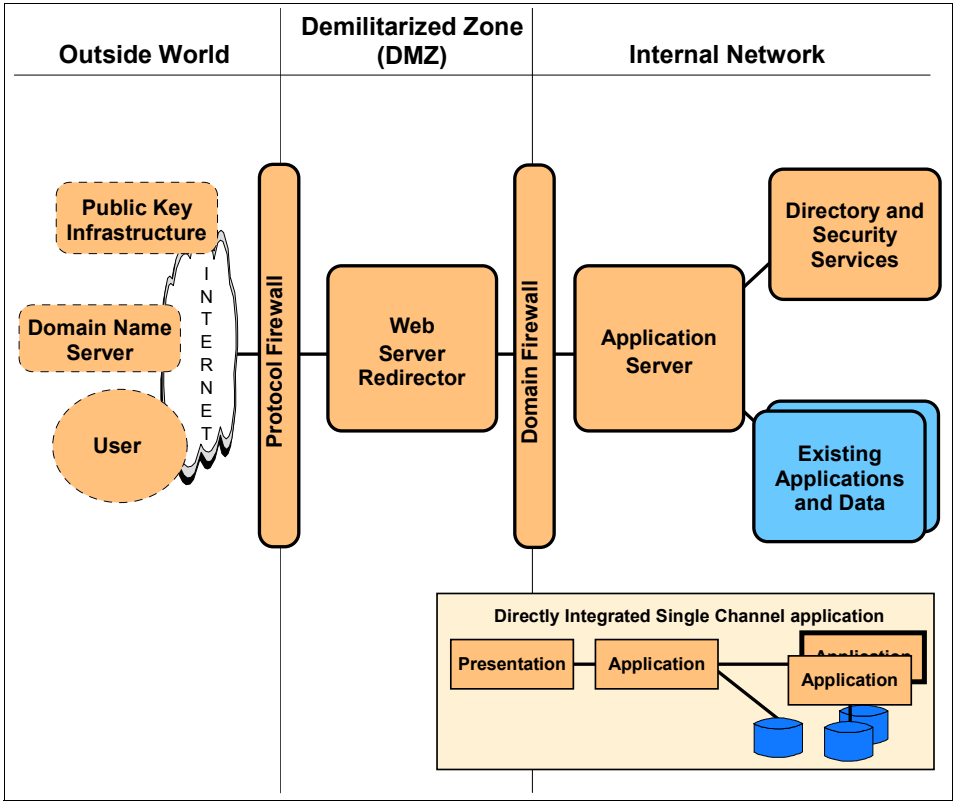


Figure 1-10 Directly Integrated Single Channel application pattern::Runtime pattern: Variation 1

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, 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 that 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.

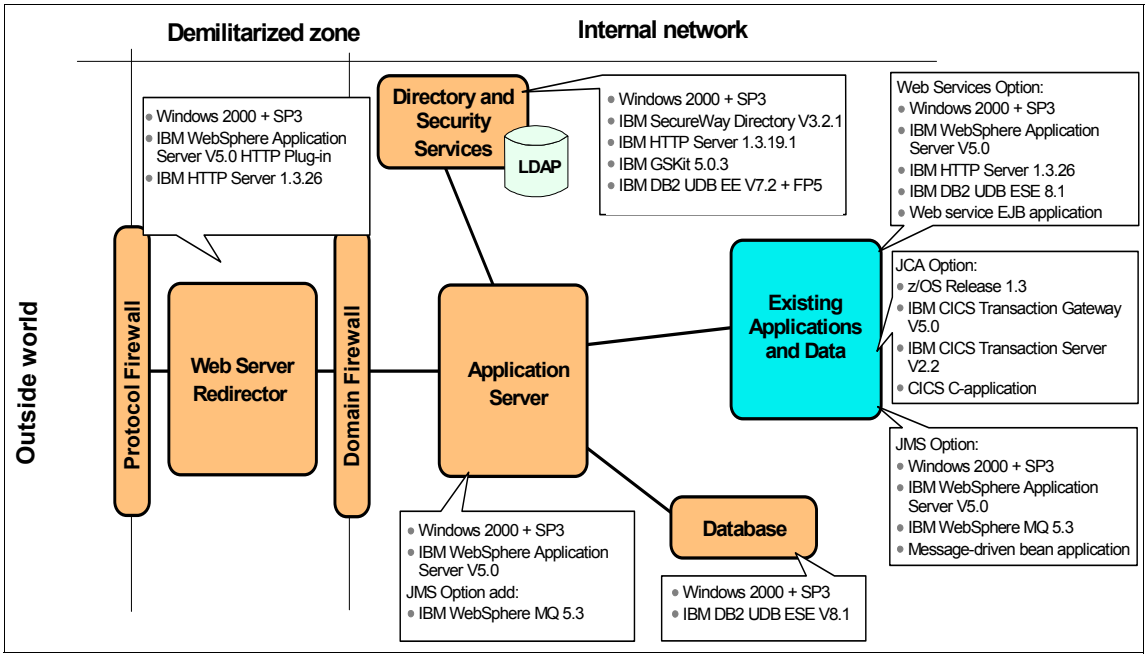


Figure 1-11 Directly Integrated Single Channel application pattern: Windows 2000 Product mapping

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 give you 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, backup 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

Application patterns for pervasive solutions

The Access Integration application patterns for pervasive solutions show application topologies for a wide spectrum of pervasive devices and applications.

The Application patterns introduced in this book (a subset of the Access Integration patterns) focus on pervasive solutions and enable the front-end integration of applications with pervasive and rich client devices.

2.1 Pervasive access applications

Pervasive access applications are those that allow you to access your business resources on the road from pervasive devices such as PDAs or cellular telephones.

Pervasive access applications have a common set of features to fulfill the business and IT requirements required by an organization desiring mobile access to its resources. These include:

- Online and/or offline application capabilities for maintain access to certain resources even without network connectivity
- A client-side repository such as a database or file directory to store data to be used and updated on the road
- A means of synchronizing data on the mobile devices with a back-end resource
- Mechanisms for securing the data and applications accessed via the device, including authentication, authorization, and secure communication
- Content adaptation based on the device and/or customer behavior
- Access to existing back-end data or applications

Typical examples for pervasive access applications could be:

- Synchronizing the company's Personal Information Management (PIM) and e-mail databases to allow employees to maintain current schedule, e-mail, and other information wherever they are.
- The use of PDAs by mobile workers to load information relating to their daily tasks from the enterprise server. These mobile workers can work offline all day and can synchronize their updated/new data from their PDAs back into Enterprise applications.
- In rail transportation, the railway switching system is critical to the movement of trains within the train yard. A malfunction causes additional work for the yard workers because they must find alternate ways to move cars and connect to the right trains. Sensors can monitor these switches and alert workers whenever anything is not functioning properly.
- Stock brokers want to be instantly notified when any of their stocks rise above 14 percent or drop more than 10 percent of their previous value.
- Commuters may want to access the most recent scheduling information for a railway system using a voice application from their telephone when they leave the office.

2.2 Pervasive Device Adapter application pattern

Note: This application pattern was formerly called Pervasive Device Access pattern. The name has changed because the primary function of this pattern is not accessing the content but adapting the content. Content access is a general requirement and function of the Access Integration patterns, including the Rich device application pattern and its variations.

The Access Integration pattern is used to provide consistent access to various applications using multiple device types. In order to allow pervasive devices to access an existing Business pattern, we therefore need to use an Access Integration application pattern. The Pervasive Device Adapter 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 HTML issued by the application presentation logic into a format appropriate for the pervasive device. In this way, the Pervasive Device Adapter 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.

The Pervasive Device Adapter application pattern is shown in Figure 2-1. This Application pattern represents the applications where the capabilities of the pervasive clients are limited and the focus is on how to enable the application and back-end data for pervasive access.

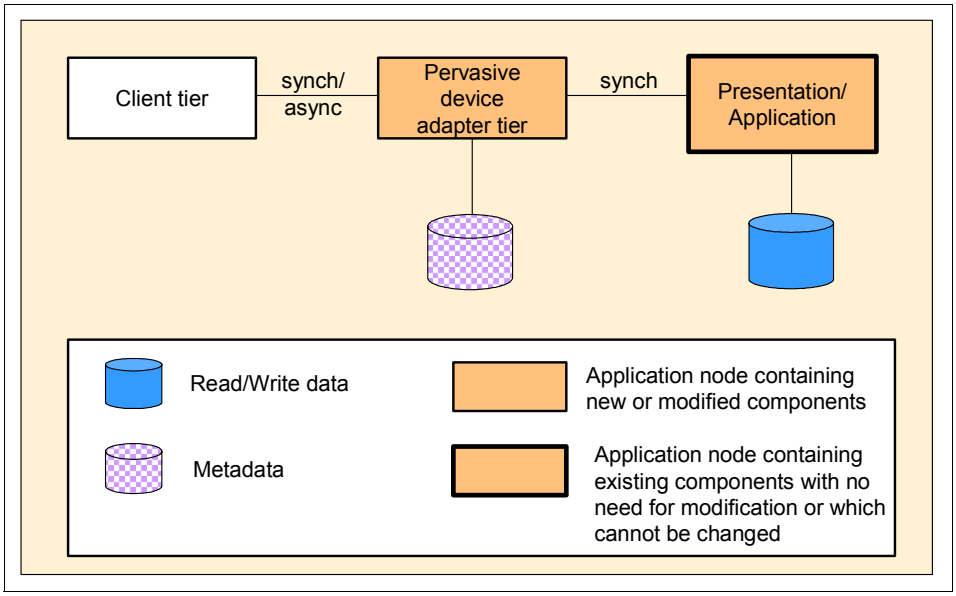


Figure 2-1 Pervasive Device Adapter application pattern

Because the pervasive client does not have any specific application running and the capabilities are not relevant, the client node is represented with an empty infrastructural (no application) node.

Application patterns for pervasive solutions are discussed in this book under two particular solution types:

- Pervasive devices
- Rich devices

In the Pervasive Devices Adapter application pattern an intermediary application node provides the necessary services for pervasive devices to access back-end applications and data. The pervasive devices vary in many aspects, including display size, memory size, networking functionality, etc. The pervasive device adapter tier provides the broadest set of services possible to match the pervasive devices on one side to the applications on the other.

The pervasive device adapter tier provides the following services, among others:

- Connectivity
- Security
- Synchronization
- Content adaptation
- Access to enterprise resources

Business and IT drivers

These:

- Provide universal access to information and services.
- 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.

Solution

The Pervasive Device Adapter application pattern is built using three logical tiers: Pervasive device, pervasive device adapter, and application.

- The pervasive device tier represents devices such as PDAs and mobile phones that can render data formats such as WML and iMode.
- The pervasive device adapter tier receives requests from pervasive devices and converts them into the appropriate requests that can be understood by existing applications and converts the response from these existing applications into formats that can be rendered by the pervasive device. The rules that govern the transcoding from one format to the other are captured by the metadata data shown in the above diagram.

  In providing pervasive device support, this tier implements the device support service for protocol adaptation and data stream transcoding and the security and administration service to ensure that the pervasive device users can achieve a single sign-on to existing applications.
- The application tier may represent a new application, a modified existing application, or an unmodified existing application. Predominantly these are browser-based applications that must be made available on wireless devices. They may represent applications that automate Self-Service, Collaboration, or Information Aggregation.

As an example, consider extending a browser-based application to be accessed by a WAP-enabled mobile phone that can render a WML data format over a cellular network. In this scenario the pervasive device adapter tier can be implemented using packages such as WebSphere Everyplace Suite. The pervasive device adapter tier can be further divided into a WAP gateway and a transcoding node.

In this case the WAP gateway is responsible for protocol conversion from WAP to HTTP and vice versa. The gateway must also handle state management issues since today's WAP devices do not support cookies.

The content adaptation node is responsible for converting the HTML response from the existing browser-based application into a WML response to be sent to the pervasive device. In doing so, this node usually uses device-specific style sheets to determine what type of information can be made available on this device. For example, an HTML page that displays "Breaking News" items with headline, summary, and a link to a detailed report obviously does not render itself well on a mobile phone display. Under this scenario one may select only the headline and the link to be displayed on the mobile phone. Similarly, downloading large graphics icons and images over the wireless network may not be the best use of the low bandwidth available. Here one may decide to either completely eliminate graphics or to convert them into image formats that are optimized such as WBMP. Device-specific style sheets can capture such content selection, conversion, and elimination criteria.

Guidelines for use

The implementation of this Application pattern calls for a careful examination of the placement of the content adaptation logic and its influence on the dialogue mapping. The content adaptation logic placed within an enterprise's infrastructure (on its Web server) allows flexible or tight control over the mapping of a dialogue to the form and size of the access device, such as a mobile phone or a Personal Digital Assistant. This placement assumes a tight linkage of the user community to the enterprise and some IT sophistication to run a wireless infrastructure.

Alternatively, the logic can be placed on the ISP infrastructure, supporting a loose coupling with the target informational Web site (the device user may access many sites using the same mapping service). In this case the owner of the dialogues (the enterprise) provides a device-specific sequence of the dialogue, which is then transcoded by the ISP service.

Benefits

This Application pattern gives users a considerable choice of devices to access their applications and data sources.

Limitations

This Application pattern is unlikely to optimize the user interface for any particular device type. If this is required, additional application changes will be necessary.

Putting the Application pattern to use

An insurance company has a team of claims assessors visiting policyholders to check the validity and value of their insurance claims. The claims assessors need frequent and fast access to the policyholder policies, claims details, and so on, plus they need to initiate contacts with garages and rental car companies through their extended enterprise applications. On the road their preferred access is through a wireless-connected palmtop device. At home or in the office they use a laptop computer for general activities like writing reports. Hence, the insurance company chooses the Pervasive Device Adapter application pattern to extend the existing claims applications to be accessed through palmtop devices.

2.3 Rich Device application patterns

The second Application pattern, and its variations, addresses the rich device application set. Rich devices are running specific rich client applications supported by a broad set of services. There are many different pervasive devices on the market— it is difficult to find the common denominator and provide a solution for an intermediary node to serve all the different clients. Rich devices, therefore, try to overcome this problem by targeting a smaller set of devices and defining a common client framework. The framework provides:

- Graphical User Interface API
- General application API
- Pervasive application API
- Access to the device hardware

The basic Rich Device application pattern consists of only one node, the client application. This node is capable of providing all the business functionality required.

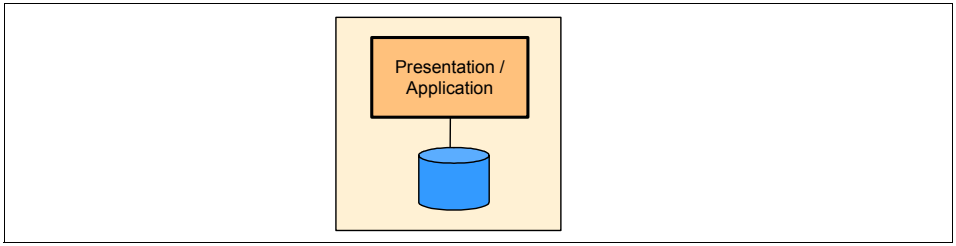


Figure 2-2 Rich Device application pattern

A client application by itself is simply not sufficient in most cases without connectivity to back-end applications. There are two variations to the Rich Device application pattern.

Application pattern variations

The first variation covers the scenario where the client is always (at least for the application execution) connected.

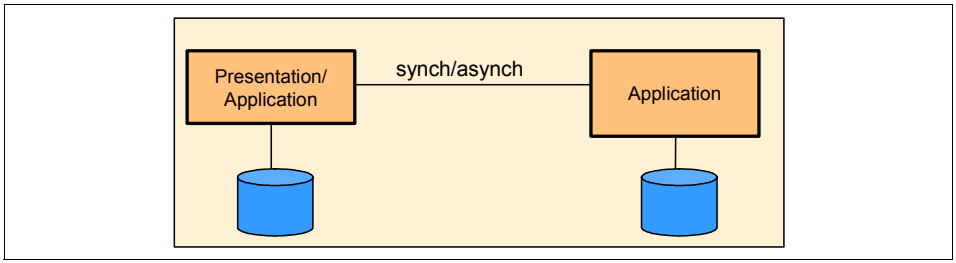


Figure 2-3 Rich Device = Online variation

The second variation represents the client scenarios with offline (not always online, or occasionally connected) capabilities. These types of clients process data in a store and forward mode where the data is stored locally until a connection can be made and the data can be transmitted to the server for onward processing.

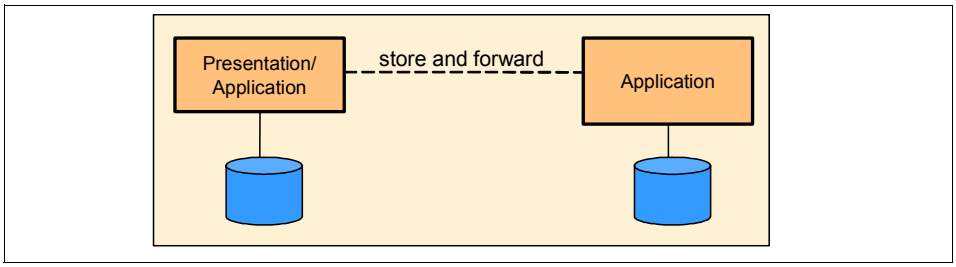


Figure 2-4 Rich Device = Store and forward variation

Business and IT drivers

Business and IT drivers:

- Provide universal access to information and services.
- Richer and more advanced user experience.
- Time to Market.
- Reduce Total Cost of Ownership (TCO).

Solution

The key in rich device applications is the application itself. As opposed to other Web-based solutions (see 2.2, “Pervasive Device Adapter application pattern” on page 21), this application pattern focuses on one logical tier: The client tier.

The client application uses services and accesses components on the server side. It is built on top of a common runtime and common framework that can run on numerous hardware devices and operating systems.

Many rich client applications are capable of operating in both online and offline mode. These applications are always available for the user with full functionality, even if the network is down for any reason. Once the network is available, the data and information can be updated on the server and on the client side. Note that only the business function is available all the time, not the most current data.

Guidelines for use

Rich clients are similar to the clients of client-server times. These devices provide fat client functionality with the advantage of server-based management. Some of the application logic is implemented on the client side where they can use services that are not available for thin clients, for example, transactionality, reliability, security, offline data store, and so on.

Rich device applications do not have the usability constraints that thin clients have. Interactions can be longer and more complex. Rich clients can have controls (user interface components) that are not available to thin clients.

Beyond the rich user experience, there are various quality of services available to these clients. For example, a client and a server can exchange information over a reliable communication, and the user can make sure that the interaction was successful.

Rich applications should ideally be designed in a way so that they can operate both in online and offline networking mode. Imagine the online operation as a special case of the offline operation where information is exchanged (synchronized) instantaneously, not in a long-running store and forward manner.

Benefits

The rich device applications provide a richer and better user experience with extended functionality. Rich device applications can enable clients to operate in both online and offline mode. The applications can also use additional services, including transactionality, reliability, security.

Limitations

The rich device applications are limited to a set of devices that are capable of operating the runtime environment together with the framework required by the clients.

Putting the Application pattern to use

An example use of the Application pattern would be the PIM and e-mail synchronization scenario. Users can access calendar, e-mail, and other personal applications on their pervasive devices and use these applications in a connected or disconnected environment. You can find more information about this sample in Chapter 9, “PIM and e-mail synchronization” on page 151.

2.4 Other Access Integration patterns

The Access Integration pattern 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.

In the rest of this section you will find details about the remaining Access Integration application patterns that are not discussed in further detail in this book.

2.4.1 Single sign-on

The single sign-on application patterns provide a framework for seamless application access through unified authentication services. Two Application patterns for single sign-on are shown below: A basic pattern where the single sign-on functions are performed in the Web tier, and an extended pattern where the security context is extended to include the back-end systems.

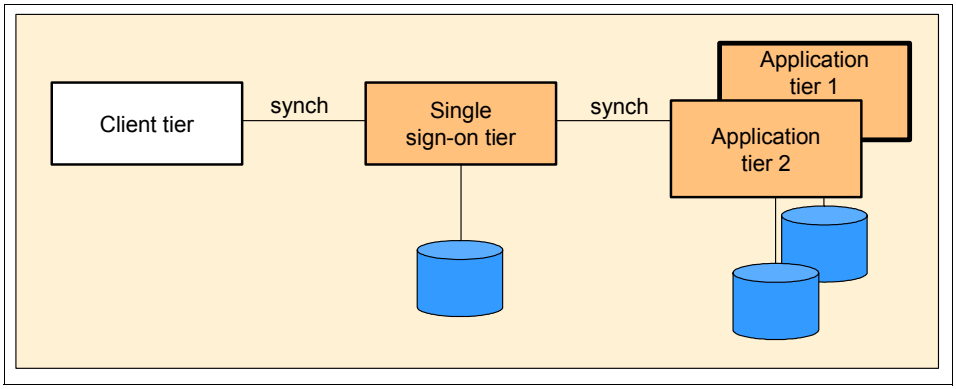


Figure 2-5 Web Single Sign-On application pattern

Business and IT drivers

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 is the main IT driver for single sign-on.

Solution

The Single Sign-On application pattern uses the security and administration service discussed above.

This Application pattern is built using three logical tiers: Client, Single Sign-On, and Application.

- The client tier represents the user interface client such as a browser, mobile phone, or PDA.
- The Single Sign-On tier implements the security and administration service, which provides a seamless sign-on capability across multiple applications. This tier uses a user profile data store, which is primarily read-only. However, this data store can also be used in a read/write manner to keep track of the last sign-on, the number of invalid sign-on attempts, and so on. The SSO tier intercepts all sign-on requests, authenticates the user, and establishes a user credential upon successful authentication. Subsequently, if the user tries to access another application that also requires a sign-on, this service automatically passes the user credential on to that application. The target application recognizes the user credential established by the security service and uses it for authorization locally. As a result, users can sign on once to access all the applications integrated using this Application pattern.
- The application tier may represent a new application, a modified existing application, or an unmodified existing application.

Guidelines for use

Some guidelines are:

- Having a single source for authentication services could create a single point of failure for dependent applications. Care must be taken to provide for high availability of this service.

- Typically, single sign-on works well to support authentication services only, leaving the supported applications to handle their own authorization as appropriate. Combining these services is generally possible only with new applications that can make use of the common services from the start.

Benefits

Some benefits are:

- Users can access their application portfolio 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.

Putting the Application pattern to use

An insurance company wants to create an Enterprise Information Portal (EIP) that consolidates various applications and information sources. Such a portal must provide single sign-on capability. To implement the requirement the insurance company chooses the Single Sign-On application pattern.

2.4.2 Extended Single Sign-On application patterns

Extending the security context to include the back-end systems enables non-repudiation of back-end system transactions. For solutions with strong privacy and/or audit requirements, this approach is needed. As shown in the figure below, these solutions almost always require a centralized user administration model. Examples include financial services transactions and access to health care clinical document systems.

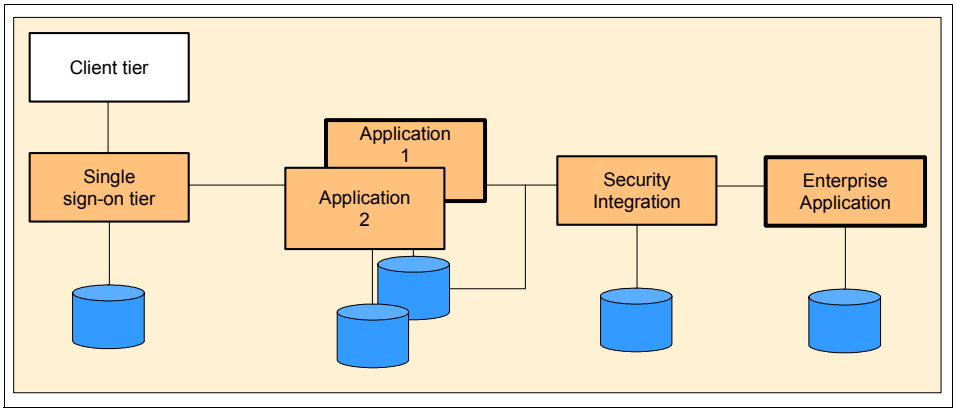


Figure 2-6 Extended Single Sign-On application pattern

2.4.3 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 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.

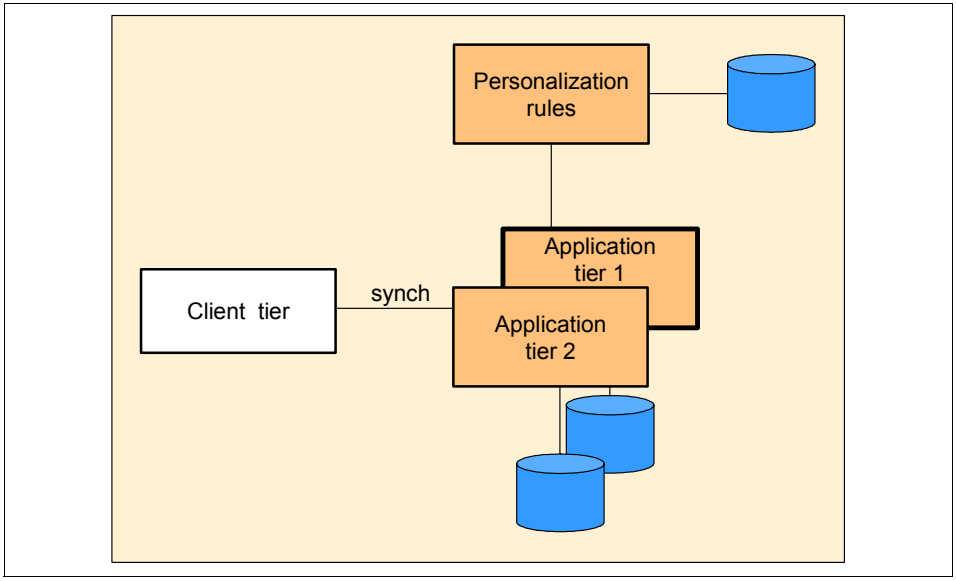


Figure 2-7 Personalized delivery

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.

Solution

The Personalized Delivery application uses three of the previous four common services for Integration business patterns discussed above:

- Personalization
- Security and administration
- Pervasive device support

This Application pattern is built using three logical tiers: Client, Personalization, and Application.

- The client tier represents the user's access device, such as a browser, PDA, phone, etc.
- The Personalization tier works in concert with the application or portal in question to tailor the application components and data presented to the user based on the desired approach (participatory, predictive, prescriptive). Personalization services typically provide a centralized repository for user profile information related to preferences, access history, and aggregate use statistics. The services also give developers the capability to define and store rules and filters, which can be used by applications to provide personalized delivery of content and applications.

  This tier implements the personalization service for data/rule/preference storage and collection and the security and administration service to determine a user's identity.
- The application tier may represent a new application, a modified existing application, or an unmodified existing application.

Guidelines for use

Successful implementation of the Personalized Delivery pattern requires a careful examination of business rules, business objectives, and applications' ability to interact with the personalization services. Without defining clear, measurable success criteria for implementation and careful results tracking, costs can quickly spiral beyond those planned for, without recognizing tangible benefits.

Benefits

The benefits are:

- 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.

Putting the Application pattern to use

The insurance company introduced in the Single Sign-On application above wants to extend their Enterprise Information Portal (EIP) such that it provides a managed window for all customer-facing employees such as customer service reps, agents, and brokers. Such a portal must personalize the welcome screen of the portal based on the user's identity. To implement these requirements the insurance company chooses the Personalized Delivery application pattern.

3

Runtime pattern

For many audiences, it is difficult to get a clear understanding of how the different types of applications can fit into the existing environment and, furthermore, how to implement them. Runtime patterns exist to help these audiences realize how they can harness and implement a pervasive environment based on their business and IT requirements.

Runtime patterns use nodes to group functional and operational components together and display those components at an abstract level. The nodes are interconnected to solve specific business problems. Each of the pre-defined Application patterns leads to one or more underlying Runtime pattern.

3.1 An introduction to the node types

A Runtime pattern consists of several nodes that represents specific functions. Most Runtime patterns consist of a core set of common nodes with the addition of one or more nodes unique to that pattern. To understand the Runtime pattern, you need to review the node definitions described in the following sections.

3.1.1 User node

The user node is most commonly a person who wants to use a specific device for accessing a business application from his local device and has either online or offline access to a back-end server. The user node could also be another client or a form of automated component such as a signaling sensor.

3.1.2 Client node

The client node could be any device that provides the network connectivity such as LAN, WAN, mobile networks (GSM/GPRS/CDMA, etc.), or telephone networks (h.323, etc.). Clients usually contain a user interface (UI), CPU, and storage to process, interact with, and store the data and request. Most frequently, the client node is a personal computer (PC), Personal Digital Assistant (PDA), smartphone, or another handheld device.

3.1.3 Pervasive client services node

This node contains those components that are designed to support device applications and interactions. The pervasive client services node is linked very tightly with the pervasive extension services. It contains those components that interact with the server side. This node also includes the layer that is responsible for running the pervasive applications on the device.

3.1.4 ISP Gateway (Pervasive services) node

The Internet Service Provider Gateway provides a network access point to certain types of devices. An example would be using a GPRS connection on a Blackberry device. ISP Gateways can contain the device recognition and user authentication information or they could act as a proxy to provide a communication channel between the device and the server.

3.1.5 Protocol firewall node

A firewall is a hardware and 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 that are connected to the Internet and can block some virus attacks. A firewall can also separate two or more logical parts of a local network to control data exchange between departments. Components of firewalls include filters or screens, each of which controls the transmission of certain classes of traffic. 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)

A pair of firewall nodes provides increasing levels of protection at the expense of increasing computing resource requirements. The protocol firewall is typically implemented as an IP router.

3.1.6 Connectivity and access for pervasive services node

This node allows an encrypted, secured, and trusted channel between client devices and corporate back-ends. It also offers seamless roaming to the corporate intranet from any location. In certain cases, this node can be used for decreasing data transmission over the wireless networks through various compression techniques. This is not a mandatory service, but security and connectivity requirements may make it necessary.

3.1.7 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 servers 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 sever behind the domain firewall into the secure network, where it is more protected than within the DMZ.

3.1.8 Telephony connector

The telephony connector node is used to maintain voice-related information such as vocabularies of words, pronunciations of words, and statistics on how certain words are used. It also contains the logic for speech recognition and can convert audio into text.

This node is required to connect the telephony network to enterprise telephony. The node would include any switch (PBX) and might support analogue, digital (E1/T1), and VoIP connectivity depending on enterprise requirements.

3.1.9 Voice gateway node

The voice gateway node supports the call control and management via an IVR and speech technology set (speech recognition and TTS). In our current partnerships we support Genesys Enterprise Edition, Avaya IR, and a range of Cisco products. We also support WebSphere Voice Response V3.1. There are other partners who are also providing IVR connectivity to IBM's speech resources. It is valid to say that any VoiceXML 2.0 and MRCP client compatible system can be supported on this node.

3.1.10 Presentation server node

The presentation server node provides services to enable a unified user interface. It is responsible for all presentation-related activity independent of local client-based applications. 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 based on the user's role.

3.1.11 Personalization server node

The personalization server node works with the Web presentation server node to customize the presentation with data that matches the users interests. 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 other back-end sources is selected for presentation to the user. It provides the mapping function of user classification to content data.

3.1.12 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 the Web-based 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 implementations validates the access to the Web application server part of the Web server, but

this node also authenticates for access to the database server and synchronization services.

3.1.13 Application server node

The application server node provides the infrastructure for application logic and can be part of a Web application server. It is capable of running both presentation and business logic but generally does not serve HTTP requests. When used with a Web server redirector, the application server node can run both presentation and business logic. In other situations, it can be used for business logic only. This application server node is the base for the pervasive extension services node.

3.1.14 Pervasive extension services node

Note: This node was formerly called the *Pervasive devices services node* in previous redbooks.

This node contains all of the pervasive components on the server side. This node is required to communicate with the pervasive client services on pervasive devices. It takes care of data transfer, content adaptation, rendering (cHTML, PDA, VoiceXML), synchronization, device management, etc. Components for notification are also included in this node. Pervasive extension services act as a server for the services and actions that are performed on the client side. This node is invoked based on the commands that the pervasive client services node sends, and returns the needed responses back to the client. Examples of this include data updates, query results, instant notifications, and confirmations.

3.1.15 Existing data and applications node

Existing applications are run and maintained on nodes that are installed on the internal network. These applications provide for business logic that uses data maintained in the internal network. The number and topology of these existing application and data nodes is dependent on the particular configuration used by these legacy systems.

3.1.16 Database node

The function of this node is to provide persistent data storage and retrieval in support of the user-to-business transactional interaction. The data stored is relevant to the specific business interaction, for example, bank balance, insurance information, and current purchases by the user.

It is important to note that the mode of database access is perhaps the most important factor determining the performance of this Web application in all but the simplest cases. The recommended approach is to collapse the database accesses into a single or a few calls. One approach for achieving this is by coding and invoking stored procedure calls on the database.

3.1.17 Collaboration server node

This node represents the existing e-mail and PIM repository. In most cases this would be a Lotus Domino or Microsoft Exchange server. In a pervasive environment, this node talks with the application server node and the Pervasive extension server node. In some cases it is also possible to connect to this node without any other nodes. An example of this is that it is possible to configure many smartphones to use the IMAP protocol, which can directly connect into the collaboration server node to retrieve e-mail messages if it supports IMAP connections.

3.2 Runtime patterns for pervasive access

Runtime patterns detail at an abstract level those components (nodes) required to enable the pervasive extensions to an existing environment.

We are going to follow the notation of Access Integration::Application pattern::Runtime pattern=(variation), but leaving the Access Integration part out to make the names shorter.

Pervasive solutions are Access Integration solutions that focus on front-end integration. The infrastructure with which pervasive solutions must integrate can be thought of as a “black box,” which can be implemented in a number of ways. For example, it can be a Web server, a Web application server, or a portal server. The black box used in this book was implemented as a portal server and color-coded blue in the figures. Readers with a different black box implementation should be able to apply the delta orange nodes and Product mappings to their own configurations.

3.2.1 Pervasive Device Adapter::Runtime pattern (composed with Portal runtime pattern)

This Runtime pattern allows users to access online portal information through their pervasive devices. This is also the most commonly used access type for users who want to log into their environment using pervasive devices.

The goal of mobile computing is to make business data and applications available on different mobile devices. Figure 3-1 describes the Pervasive Device Adapter::Runtime pattern (composed with the Portal runtime pattern) and the nodes that are required for the implementation.

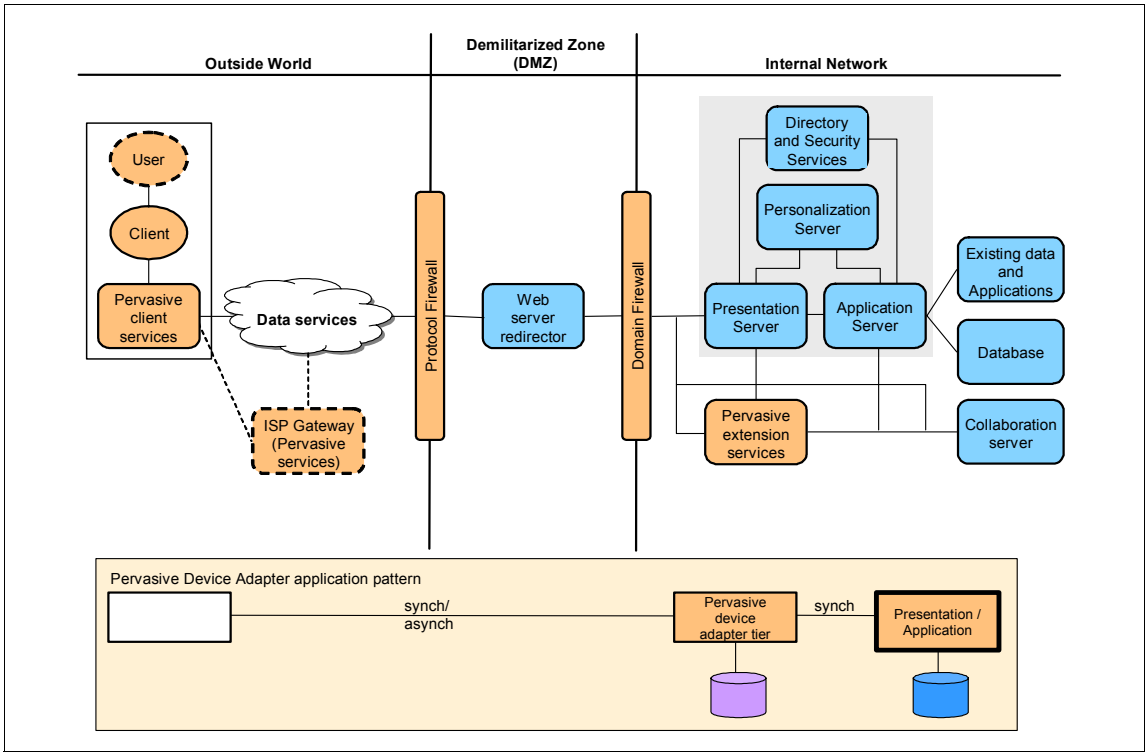


Figure 3-1 Pervasive Device Adapter::Runtime pattern (in orange) composed with Portal (in blue)

3.2.2 Pervasive Device Adapter=Voice::Runtime pattern

The Pervasive runtime pattern for voice is used when enterprises would like to extend their applications to support voice interaction.

The Figure 3-2 on page 42 shows a basic set of nodes to consider for voice solutions. There are two different clients shown on the diagram.

- One client is the traditional phone client. It could be any type of phone, including cell phone, landline phone, etc.
- The other client is a rich client application with VoIP support. The client can make phone calls over the IP protocol. In this case VoIP service providers need to provide switching from the IP network to the phone network.

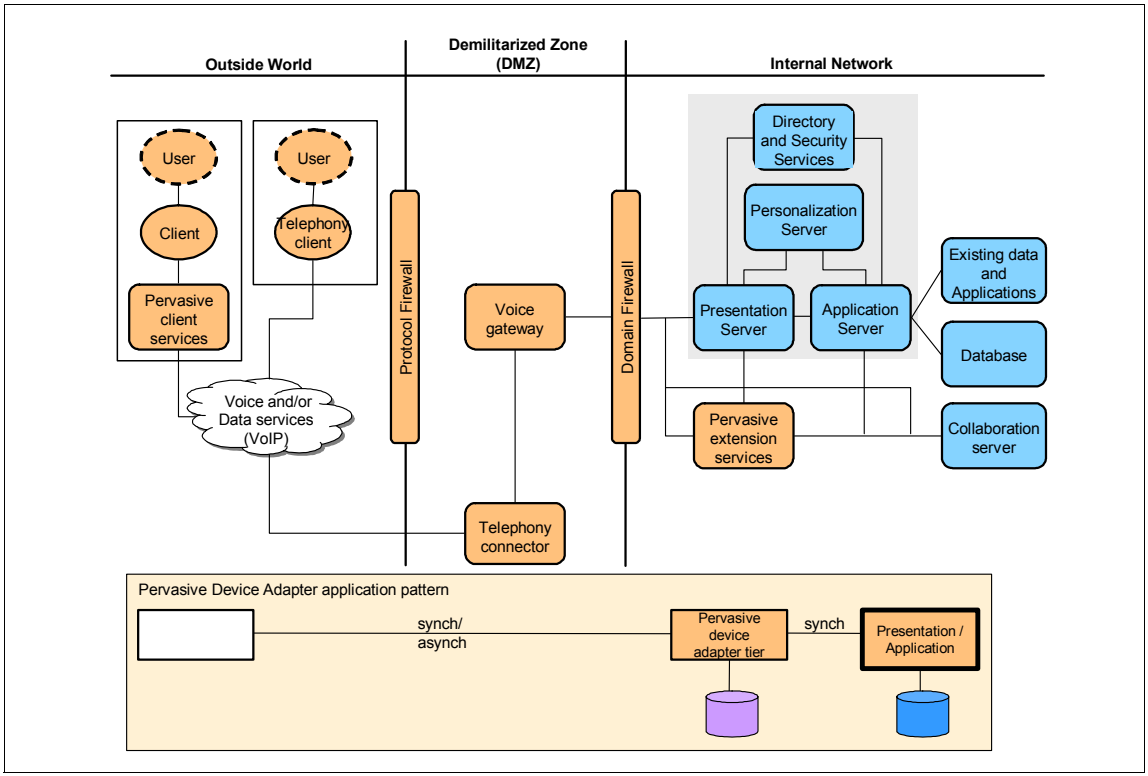


Figure 3-2 Pervasive Device Adapter=Voice::Runtime pattern

3.2.3 Rich Device::Runtime pattern

This Runtime pattern is only a starting point for more complex Runtime patterns for the variations of the Rich Device application pattern.

The basic pattern only includes the client tier and focuses on one particular node, the rich device client. The diagram also includes the “outside”, public network with data services.

This pattern depicts the scenario when the user is running a standalone client application on the pervasive device, and it is not connected to any server component.

Note that the network is available even for standalone applications. It is possible for these applications to communicate with each other in a peer-to-peer manner without involving any server component (for example, direct connection via Bluetooth).

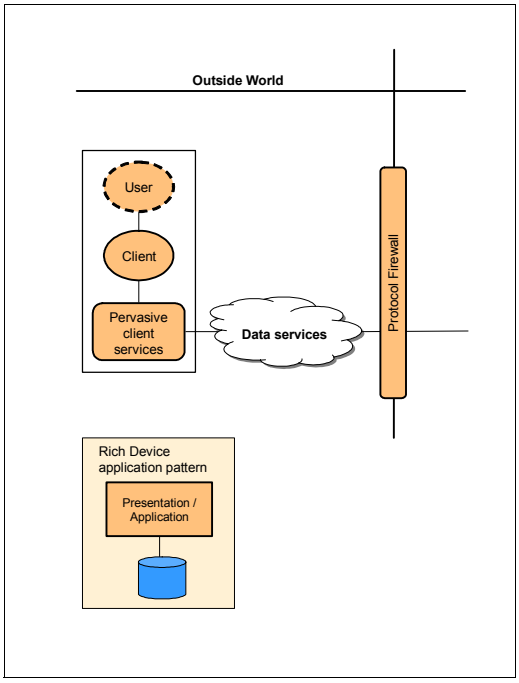


Figure 3-3 Rich Device::Runtime pattern

3.2.4 Rich Device=Online::Runtime pattern

Rich client applications can also operate in an online environment, where the application has a live connection to the network.

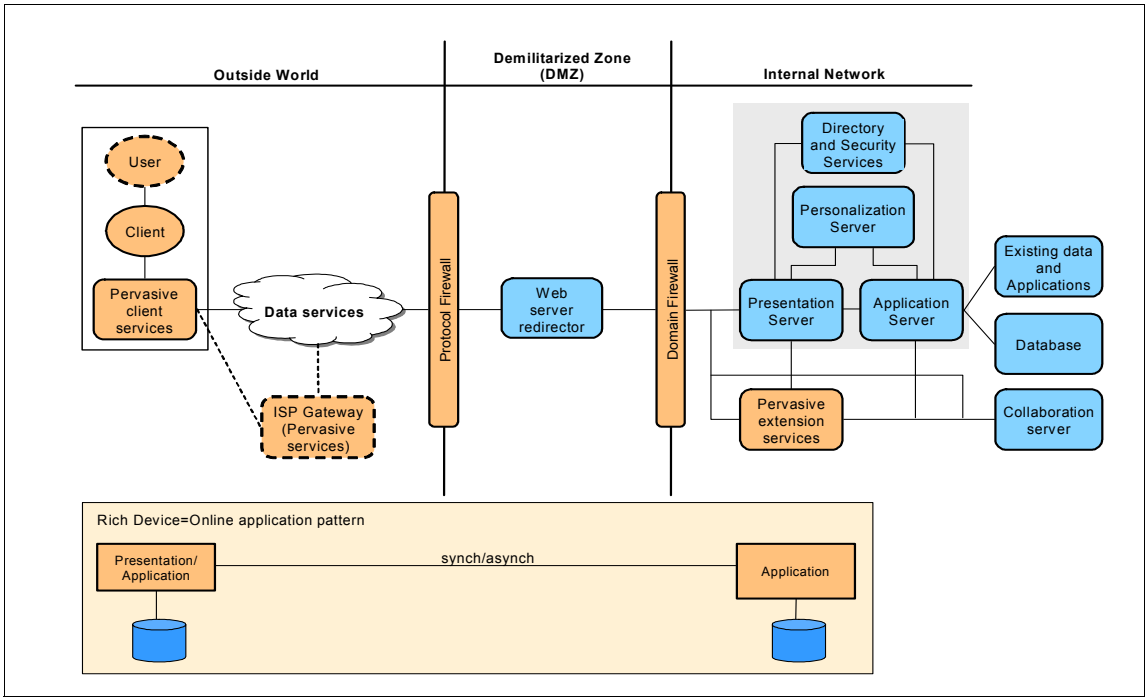


Figure 3-4 Rich Device=Online::Runtime pattern

This Runtime pattern is a nice fit for Device management solutions. Device management requires online connection to the central management system to perform system and application updates and configurations on the client side.

Another example for this Runtime pattern is instant messaging, including the intelligent notification services using instant messaging scenario. In this case the client communicates with a server application or with another client using instant messaging.

3.2.5 Rich Device=Store and forward::Runtime pattern

The following Pervasive Runtime pattern can be used for any offline applications that use a local repository (either a database, a file, or a message queue) to access back-end services through the pervasive extension services node.

This Runtime pattern can be a realization for many different scenarios. The store and forward mechanism does not necessarily imply offline operation. An application performing store and forward operations in an online environment (instant replication, synchronization, or forward) can be considered as an online

application. So basically this pattern can operate in both online and offline environments.

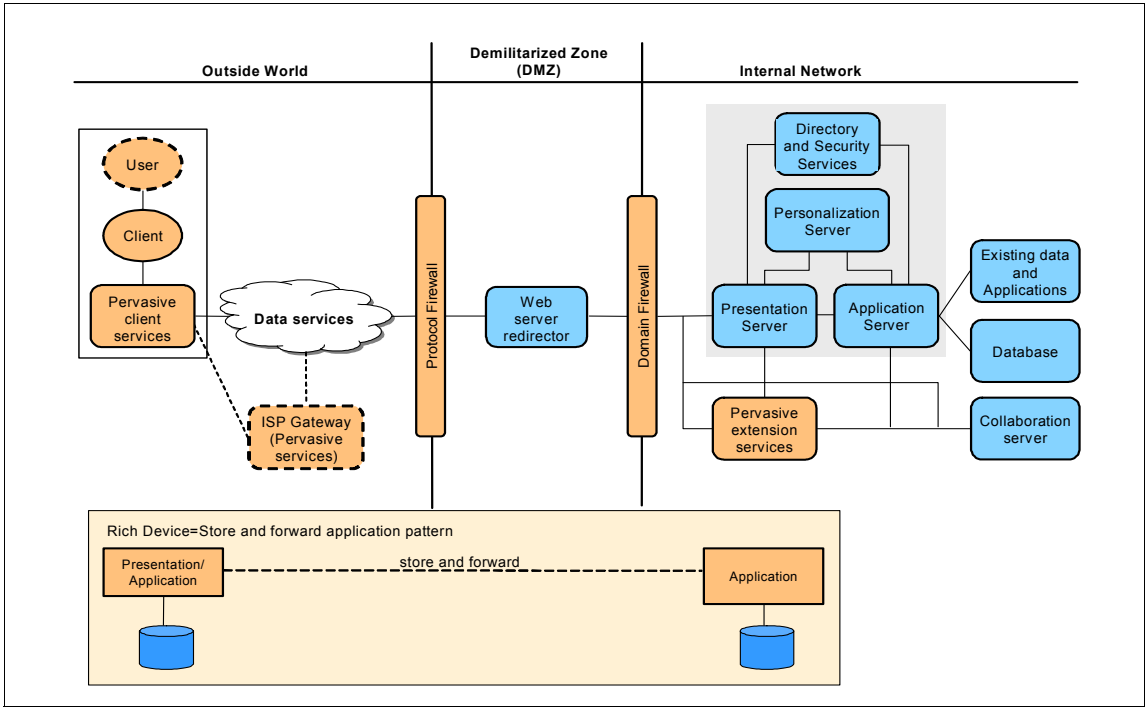


Figure 3-5 Rich Device=Store and Forward::Runtime pattern (in orange) composed with Portal (in blue)

3.2.6 Rich Device=Store and forward::Runtime pattern variation 1

The PIM and e-mail synchronization service is generally a service included in pervasive solutions software packages. PIM and e-mail synchronization is an application type using the store and forward type of operation. It allows remote access to existing e-mail and calendar information, which can be used in e-mail and calendar applications on the pervasive devices.

There are different ways to alter the PIM and e-mail runtime pattern for scalability purposes, decreasing response times, and providing high-availability support for multiple users. The most efficient way to do this is to give users access to multiple different physical collaboration servers. The following diagram shows the separated pervasive extension services where PIM and e-mail support is distributed over multiple nodes.

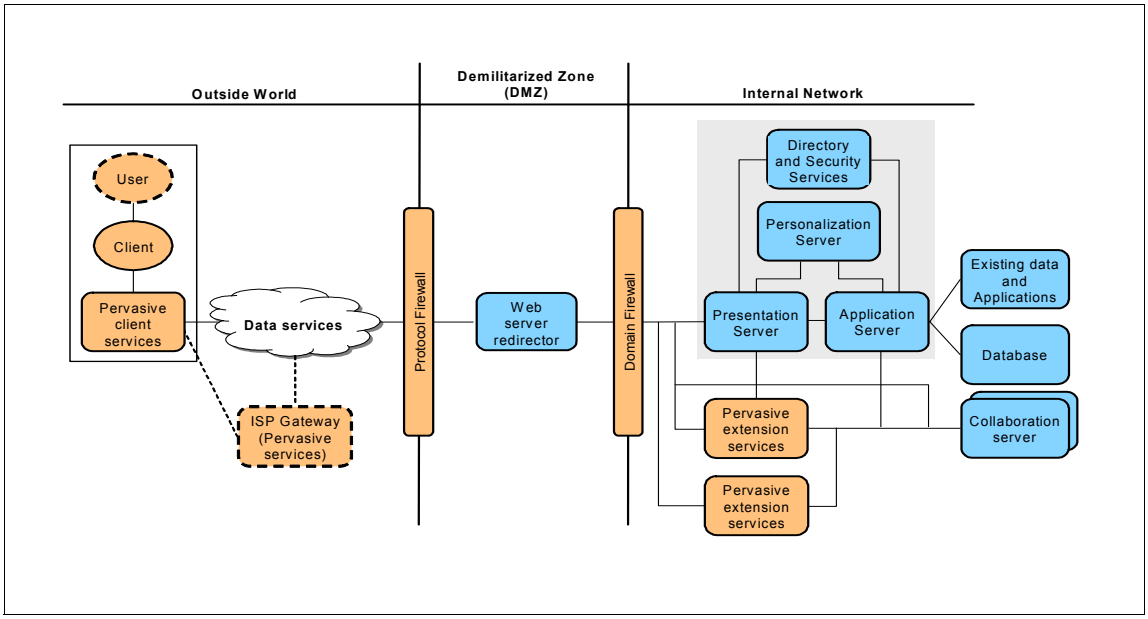


Figure 3-6 Rich Device=Store and forward::Runtime pattern variation 1 (in orange) coupled with Portal (in blue)

3.2.7 Composite Rich Device=Online and PDA=Voice::Runtime pattern

This section describes the variation of a Runtime pattern for a voice solution that can be used for a multimodal approach. Multimodality allows a user to utilize multiple means of interaction (for example, voice and Web browser) with an application at the same time. Using this pattern, an environment could use applications either through data or voice. The application itself could also support both behaviors at the same time; for example, users could send requests through the voice channel (using speech for request) and get results through the data channel (getting the response in WML, for example). An example scenario of using a multimodal application:

1. The user logs onto the application using voice recognition.
2. The voice application that uses the existing back-end system asks what actions the user would like to perform. The options are: Users account statement, latest bank transactions, loans, and customer service.
3. The user selects the latest bank transactions and sends a query using voice. This query could contain the day's information or describe the days for those transactions, for example, the past ten days' bank transactions.

4. The voice service transforms requests into a data format. Using this format, the request passes through the backend presentation layer, which can read the request and launch an existing application to create a response.
5. When the application is ready to send the response, it will check if the user is logged in using a Web browser. When the application knows that the user is using a Web browser, it will send the response through the data gateway. The presentation server node takes care of the content translation based on the device and routes that content through the proper gateway.
6. The user will then get his latest bank transactions for the past ten days shown in the Web browser.

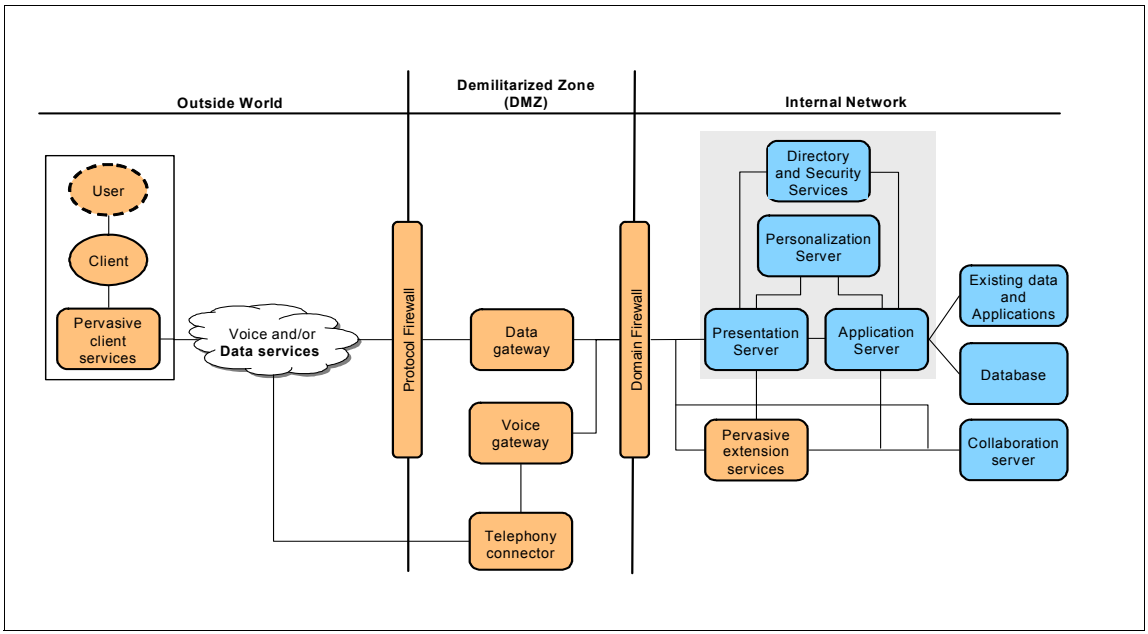


Figure 3-7 Composite Rich Device=Online and PDA=Voice::Runtime pattern

Voice solutions allow the user to query the system naturally using voice and retrieve this information efficiently by receiving it visually on a device. The multimodal approach is a good combination to take advantage of both channels, voice and data. It allows a very flexible and user-friendly approach.

3.2.8 Pervasive Connectivity::Runtime pattern

This Runtime pattern focuses on establishing a secured and trusted connection between the client and server. Security and roaming are two very important issues when designing and implementing a mobile environment. Today, the devices themselves do not possess a means to secure a connection to the

Internet that meets generally accepted company policies. Because of this, companies are forced to find solutions that fit their business requirements.

Figure 3-8 describes the nodes that are required for the Runtime pattern for pervasive connectivity. The connectivity and access for pervasive services node could be divided into two parts. The first handles the security and authorization towards the network. The second is used for accessing the application data based on the user authentication and authorization as per the directory and security services node.

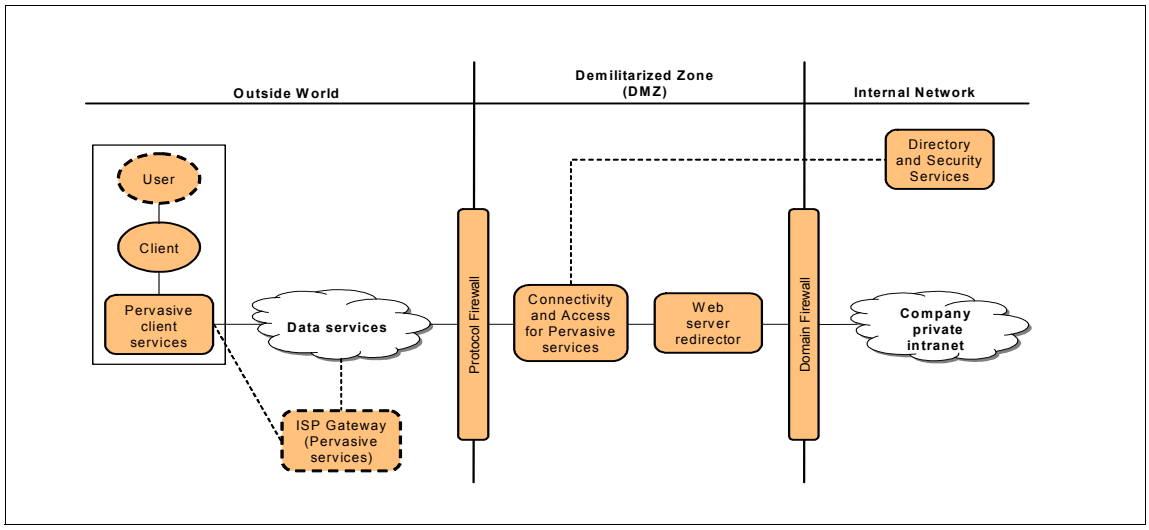


Figure 3-8 Pervasive Connectivity::Runtime pattern

3.2.9 Composite Pervasive and Rich Device solution::Runtime pattern

This Runtime pattern collects all the nodes that are available within the pervasive environment. Figure 3-9 on page 49 describes all of these pervasive nodes in one Runtime pattern.

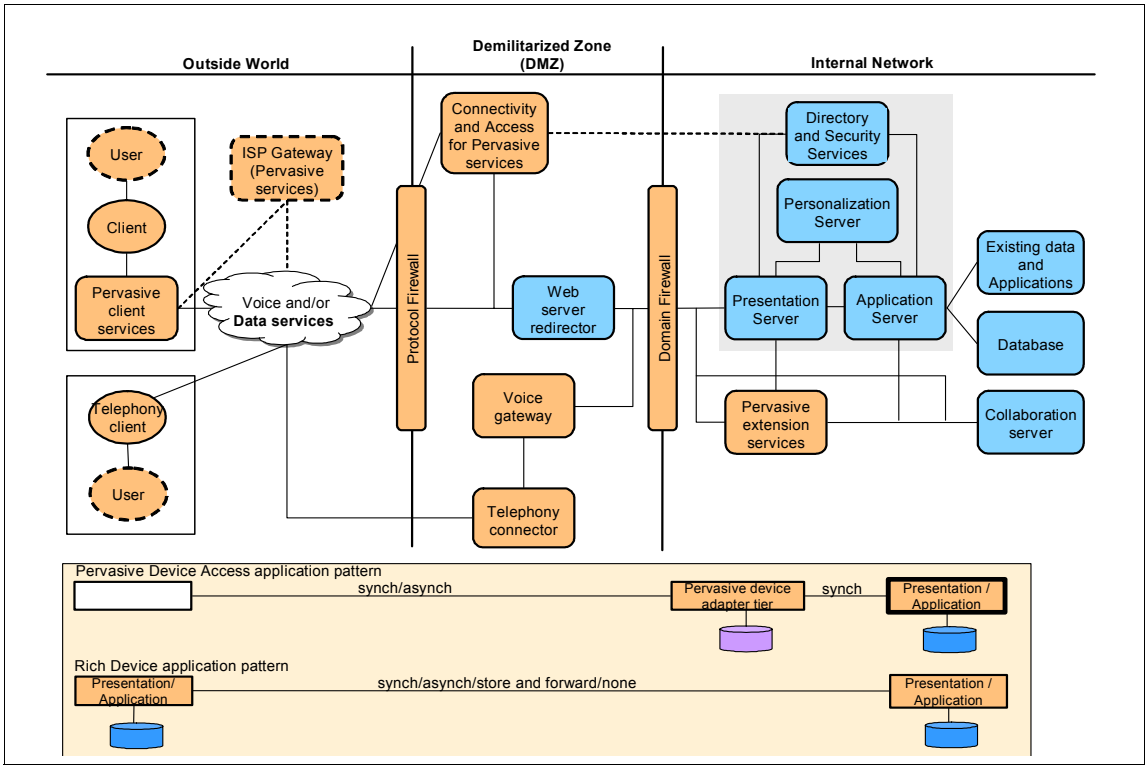


Figure 3-9 Composite Pervasive and Rich Device solution::Runtime pattern

Pervasive runtime patterns provide a controlled and secure environment by locating all sensitive and persistent data behind firewalls. It does not provide scalability or failover capabilities, but it is a good starting point from which to provide this functionality. It also describes those nodes that are related to Internet Service Providers, which are not often clear when starting pervasive projects.

Composite Pervasive and Rich Device solution::Runtime pattern variation 1

For performance reasons, it is also possible to divide and distribute the pervasive extension services node into multiple different nodes and direct certain specific functions away from each other. This is an effective way if one group will use only PIM and e-mail synchronization, another group will only use data synchronization, and another group only uses online capabilities.

Figure 3-10 describes the variation of the Runtime pattern that is used for high availability and load balancing based on different group-related behaviors.

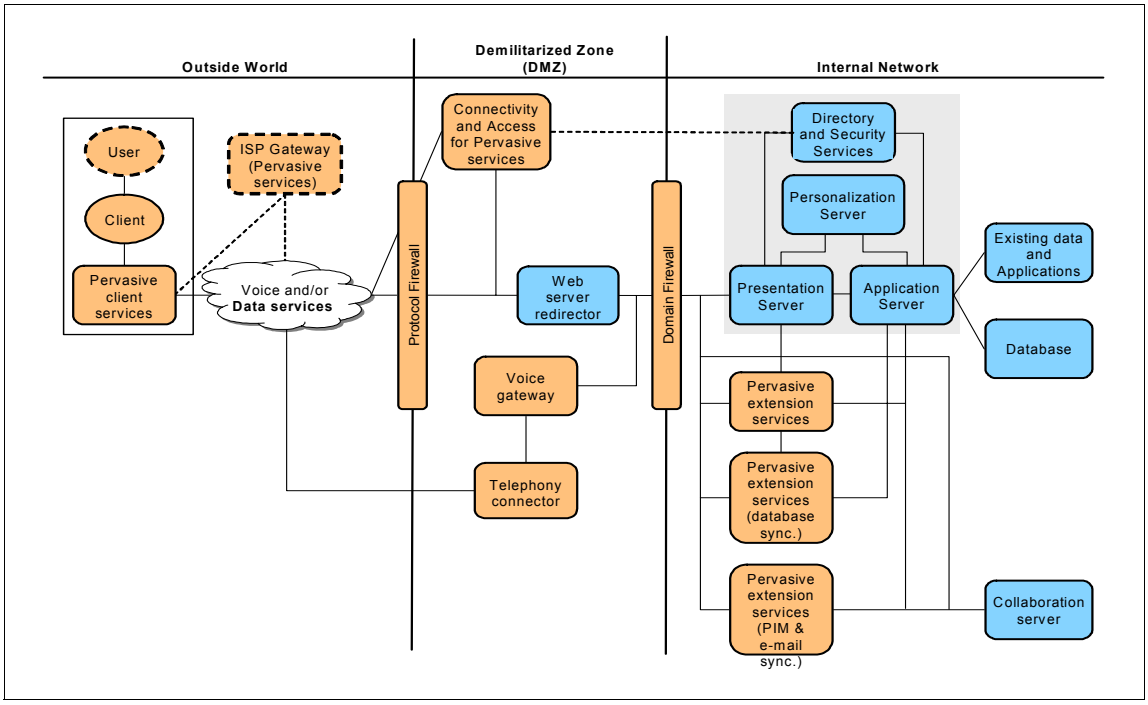


Figure 3-10 Composite Pervasive and Rich Device solution::Runtime pattern variation 1

4

Product mappings

After we have found a suitable Runtime pattern and customized, it we need to find actual products and platforms that will support the planned environment. It is recommended that the final platform selection be based on the following considerations:

- Existing systems and platform investments
- Customer and developer skills that are available
- Customer choice
- Supported platforms for each component
- New extensions that will fit with the existing customer environment

The selected platform should fit into the customer's environment and ensure quality of service, scalability, and reliability so that the solution can grow along with the businesses and remain in line with the company’s on demand strategy.

4.1 Overview of IBM pervasive software products

For the sample scenario, ITSO Railways, introduced in this book, the following products were used:

- On the server side:
  - IBM WebSphere Application Server V5.0
  - Websphere Everyplace Access V5.0
  - WebSphere Portal Server V5.02
  - WebSphere MQe 2.0.1
  - Lotus Domino 6.5.1
  - Sametime 3.0 and Sametime Everyplace 3.1 in Microsoft Windows 2000 environment on a server side
- On the client side:
  - WebSphere Everyplace Access client V5.0
  - WebSphere Client Technology, Micro Edition V5.7
  - Sametime Connector 3.1 in Pocket PC 2003 and 2002

We will create a Product mapping for each Runtime pattern based on our scenarios and recommend different platform alternatives for each: One for Windows and one for AIX or Linux.

The scenarios that we use for the Product mappings are:

- Online pervasive portal
- PIM and e-mail synchronization
- Device-based solutions
- Voice application
- Device management
- Connectivity
- Pervasive solution and protocol mapping

This chapter introduces the major products used in this application and provides an overview of the products.

4.1.1 WebSphere Everyplace Access V5.0

WebSphere Everyplace Access provides a platform for mobile employees to have access to the information they need. For example, think of a field sales professional getting remote access to Personal Information Management (PIM) data and information about customers, inventory, and delivery times. Employees

are able to confirm delivery times and then check their own calendar to set up a follow-up appointment. This shortens cycle time and makes sales calls more effective, in turn leaving the customer more satisfied.

Everyplace Access makes information available to mobile employees based on connectivity and the type of data they want to access on a selected set of mobile clients.

WebSphere Application Server

IBM WebSphere Application Server Enterprise V5.0 provides the features in IBM WebSphere Application Server Network Deployment V5.0 plus programming model extensions for sophisticated application designs.

It offers advanced capabilities such as application adapters, application workflow composition and choreography, extended messaging, dynamic rules-based application adaptability, internationalization, and asynchronous processing.

WebSphere MQ is bundled with the package (except on z/OS).

More information about IBM WebSphere Application Server Enterprise V5.0 can be found at:

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

WebSphere Portal Server

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.

IBM DB2

IBM's DB2 database software is a full-featured, robust, scalable, and easy-to-use relational database. DB2 provides the foundation of information on demand on Linux, UNIX, and Windows platforms.

Innovative manageability DB2 Version 8.1 provides automation capabilities, including self-configuring, self-optimizing, and self-managing capabilities.

- New levels of integrated information across the entire enterprise leverage federated Web Services and XML to help solve critical business problems. New federated capabilities enable customers to integrate information as Web Services. XML enhancements make it easier for programmers to integrate DB2 and XML information.
- Robust e-business foundation: Performance, scalability, and availability enhancements continue with cross-workload and cross-platform leadership,

improving overall application performance and making information highly available.

DB2 Everyplace

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).

Using DB2 Everyplace users can synchronize data stored in any JDBC-compliant relational database with the Everyplace Client application on the mobile device. To demonstrate this functionality, a sample DB2 Everyplace application is provided with the Everyplace Client. Everyplace Access allows application developers to create custom relational database synchronization applications using DB2 Everyplace and the DB2 Everyplace SDK

DB2 Everyplace is part of IBM's solution for pervasive computing. With DB2 Everyplace, mobile professionals (such as sales people, inspectors, auditors, field service technicians, doctors, realtors, and insurance claim adjusters) can keep in touch with vital data that they need access to when away from the office.

Organizations are able to deliver their DB2 enterprise data to mobile and embedded devices. With DB2 Everyplace, you can access and perform updates to a database on your mobile device. With DB2 Everyplace Sync Server, you can synchronize data from the mobile device to other data sources in your enterprise. The file adapter capability enables you to distribute files and applications to mobile users.

Everyplace Synchronization Server

Everyplace Synchronization Server enables handheld computing devices to link remotely to desktop applications. Mobile users can easily synchronize data with Microsoft Exchange, Lotus Notes. The mobile device can synchronize using modem, cellular phone, Internet, Wireless, intranet, local area network (LAN), or wide area network (WAN).

Synchronization Server and Everyplace Client enable users to synchronize back-end Personal Information Manager (PIM) data such as e-mail, calendar, tasks, and memos between supported mobile devices and Lotus Notes and Microsoft Exchange databases. Users customize synchronization preferences using the provided synchronization settings portlet. Synchronization enables users to connect, synchronize, and go. They can view their e-mail, calendar, or other PIM data later, when they are no longer connected. From their mobile devices, users can respond to messages when they are offline. During the next synchronization session, their responses are copied to the server.

Everyplace Intelligent Notification Services

A *notification* is a message sent to the user by Intelligent Notification Services. This message can either be a simple notification originating from another user or application, or it can be a subscription-based notification notifying groups of users of information events to which they subscribed. Notifications are sent to users via delivery channels. *Delivery channels* are mechanisms for receiving messages, such as portlets, Lotus Sametime, e-mail, and Short Messaging Service (SMS). Intelligent Notification users can set *preferences* that effect when, how, by whom, and with which delivery channels the users are notified.

- *Simple notifications* are 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* are messages that are triggered by events to which the user subscribes. For example, users can subscribe to a stock notification that alerts them when the stock price for a specific company has gone above a set number. Another example is where users subscribe to Web news services and receive notifications when any articles are published by the Associated Press that relate to a company, product, or subject that the users define.
- Delivery channels are the mechanisms through which users of Intelligent Notification receive messages. Intelligent Notification Services supports the following types of delivery channels:
  - Message Center - A user portlet with which users view messages, delete messages, and send simple notifications to other Intelligent Notification users.
  - Lotus Sametime - Channel for sending messages through Lotus Sametime's instant messaging server and client pair.
  - Pager Wireless Communication Transfer Protocol (WCTP) - Channel for sending messages to pagers using the WCTP protocol. E-mail channels can also be used to send notifications to pagers.
  - E-mail Simple Mail Transfer Protocol (SMTP) - Channel for sending e-mail messages using the Simple Mail Transfer Protocol.
  - Short Message Service (SMS) - Channel for sending short text messages to mobile devices.
  - Wireless Application Protocol (WAP) - Channel for sending messages to wireless devices such as cell phones and Personal Digital Assistants (PDAs).
  - Voice - Channel for sending messages to a telephone through the voice server.

  - Server-Initiated Actions - Delivery channel that is used to push commands from the server to a client; for example, the command to synchronize e-mail can be initiated by the server.
  - AOL Instant Messenger - Channel for sending messages on America On-Line Instant Messenger service.
  - BlackBerry - Channel for sending messages to a BlackBerry device.
- Preferences and subscription settings
  - Intelligent Notification Services allows subscribers to specify many types of preferences and subscription settings. These parameters affect with what, where, when, and how the subscribers are notified. The following list details the notification parameters that subscribers can set.
  - Delivery channel information - Delivery channel properties such as name, e-mail address, Sametime user ID, phone number, and time periods when the user does not want messages delivered.
  - User-defined groups - Groups of Intelligent Notification users that the user creates.
  - Group members - Which Intelligent Notification users belong to a user-defined group.
  - Message rules - The priority level of messages that are allowed to be delivered from the members within a group to each of the user's delivery channels.
  - Subscription settings - The criteria used to match content for a particular subscription, and the delivery preferences used when a match is made.

Device Manager

Device Manager is device management technology that helps enterprises and private networks, or providers, manage devices.

For Device Manager, devices are personal digital assistants (PDAs), handheld PCs, PCs, sub-notebooks, cellular phones, set-top boxes, in-vehicle information systems, and other devices used in pervasive computing.

Device Manager is used to enroll devices into a database and perform many tasks for managing devices, such as configuration, inventory collection, distribution of software, and initial provisioning of devices. Device Manager is used with a subscription manager component to control access to the user interfaces for administrators and device users.

Management actions are referred to by Device Manager as jobs. Jobs may be applied to (or targeted to) a single device or a group of devices. Groups of devices are defined by a list of devices, or by characterizing the group. A group

of devices is characterized by the device owner or owner group, or by some attributes of the device inventory, or both.

Jobs are targeted to a device and the job is run when the device connects to the Device Manager server. The server maintains a history of jobs status for all jobs and all devices. For certain device types, the server can notify the device that a job is waiting for the device.

Device Manager is built on a Web application server model. The Device Manager server is a set of J2EE servlets, running on WebSphere Application Server. Device management data storage is in a relational database, such as DB2 or Oracle. The Device Manager server requires an agent on the managed device. The agent can be a Device Manager supplied agent, such as Pocket PC, Windows 32-bit, and PalmOS, or can be supplied by a device manufacturer.

Device agents communicate with the Device Manager server using HTTP or HTTPS. The protocol running on top of HTTP is either a proprietary protocol, which is used with Pocket PC and PalmOS devices, or a standard protocol, such as the OMA DM protocol used with OSGi devices. With HTTP and HTTPS communications between devices and Device Manager, requests and responses can pass through various network elements, such as firewalls.

Device Manager provides an OMA DM Management Server that is an OMA DM V1.1.2 implementation. Available management functions for SyncML DM devices include inventory collection, device configuration, and running custom-built SyncML DM command jobs.

Device Manager also supports software bundle management for OSGi-enabled devices.

Everyplace Client

Everyplace Client provides the client application that is needed on the mobile device to accomplish synchronization and access to the back-end system. The Everyplace Client provides a single user interface for multiple client synchronization functions. The provided synchronization applications currently include the following.

- E-mail and PIM is provided for PIM data synchronization with Lotus Notes and Microsoft Exchange. Synchronization server provides the required interface with the backend databases.
- Database synchronization is provided for relational database synchronization with supported back-end JDBC and ODBC compliant databases. DB2 Everyplace provides the required interface with the backend relational database.

- Offline Portal Pages is provided to synchronize designated offline browsing content and forms to the device. The WebSphere Everyplace Access offline browsing feature provides the required interface with the portal.

4.1.2 WebSphere Everyplace Connection Manager

WebSphere Everyplace Connection Manager can help boost the productivity of mobile workers by giving them highly secure, uninterrupted access to the data they need. Offering a distributed, scalable, multipurpose communications platform, WebSphere Everyplace Connection Manager V5 can help enterprises optimize bandwidth, reduce costs, and ensure security by efficiently extending their existing applications to workers in the field over many different wireless and wireline networks. This:

- Provides compression and other network optimizations that decrease user response time and lower network costs.
- Features seamless cross-network roaming, making it possible to dynamically switch network connections without interrupting applications.
- Encrypts data over vulnerable wireless LAN and wireless WAN connections. It provides a Virtual Private Network (VPN) between communicating partners.
- Integrates standard Internet Protocols (IPs) and non-IP wireless bearer networks, server hardware, device operating systems, and mobile security protocols.
- WebSphere Everyplace Access provides an entry-level starting point so your enterprise can start small and expand to richer wireless application functions to meet your company's needs.
- WebSphere Everyplace Connection Manager allows service providers to offer highly encrypted, optimized, and scalable solutions to enterprise customers.

WebSphere Everyplace Connection Manager also ships client-side components for numerous different devices. The client component provides connectivity and security services for the client.

4.1.3 WebSphere MQ Everyplace

WebSphere MQ Everyplace V2.0 connects mobile and wireless applications with the enterprise using secure and dependable application messaging. MQ Everyplace provides the mobile transport for the IBM enterprise integration bus and connects seamlessly with WebSphere Business Integration offerings. It

extends robust messaging to fragile mobile and wireless networks to address the problem of intermittent network connectivity. It supports:

- Broad mobile device support: Supports a wide range of devices with small, customizable footprint. Offers a choice of languages, APIs, and environments including Java, C, JMS, and J2ME.
- Robust mobile integration: Once-only messaging so transactions are not lost or duplicated between mobile applications. Peer-to-peer, synchronous, and asynchronous support. Rich encryption, non-repudiation, and authentication features.
- Extensive customizablility: Configure rules to transmit during off-peak rates or at specific times. Suitable for unmanaged networks. Message caching and compression features help lower communication costs. JMX support for increased systems management.

4.1.4 WebSphere Client Technology, Micro Edition

Workplace Client Technology, Micro Edition is a platform onto which the next generation of e-business applications can be deployed. It provides a server-managed Java-powered platform with access to many enterprise applications, data, and transactions.

Workplace Client Technology, Micro Edition uses industry standards and middleware to combine the convenience of pervasive devices with the power of e-business and the productivity of IBM Workplace. Several elements can be applied to a wide array of device types, including desktop computers, laptop computers, mobile handsets, PDAs, controllers, gateways, set-top boxes, and others to deliver the right combination of productivity and integrity required for a specific application.

Customers can quickly select the data services that enhance the productivity of end users. IBM and its business partners are ready to assist you in the development of pilot applications and services for any new and existing device, as well as offering the end-to-end device-to-services expertise that can deliver integrity to the device.

4.1.5 Domino Server V6.5.1

Lotus Notes is a workgroup computing environment that helps people work together more effectively. With Notes, people can work together regardless of computer platform or technical, organizational, or geographical boundaries. Notes-based information can be shared across any distance, at any time.

The Domino server provides services to Notes workstation users and other Domino servers including storage of shared databases and mail routing.

Domino server has the ability to transform Lotus Notes databases into interactive World Wide Web applications. It supports security from standard user name and password authentication to Secured Socket Layer (SSL) encryption.

Domino allows users to rapidly develop World Wide Web (WWW) applications using Lotus Notes' built-in database design features. Once created, users on both Notes and the WWW can access and interact with these databases. The Access Control that Notes provides is carried to the Web, allowing simple, and very flexible control of who can access the site, the views, and even the individual documents. And because Domino creates all the HTML code on the fly, all document and view updates are immediately exported to the Web.

Sametime Server V3.1

Lotus Sametime is a family of products that allow organizations to share and collaborate on documents as well as immediately find and converse with colleagues and partners worldwide in real time. Lotus Sametime, which also includes secure, reliable management features, enables organizations to exploit the knowledge that exists within the organization by providing a cost-effective, secure method of instantly exchanging knowledge with others, regardless of their geographic location.

Sametime Everyplace Server V3.1

IBM Lotus Sametime Everyplace V3 is a wireless instant messaging solution and the latest member of the industry-defining Lotus Sametime family of instant messaging solutions. Sametime Everyplace extends the instant messaging capabilities and presence awareness of Sametime to mobile phones and a wide range of wireless devices including Pocket PC, Palm, and MIDP devices.

As presence awareness and instant messaging become a core component of a company's communications infrastructure, mobile access to that infrastructure is critical. Sametime Everyplace helps mobile employees to be more accessible to each other and to co-workers at the office—without sacrificing the security of their company's network. In addition, mobile access to presence awareness and instant messaging opens the door for the next generation of wireless applications, such as smart notifications and access to interactive agents (often called *bots*), which allow employees to tap mission-critical applications and information immediately. With bots, users of Sametime Everyplace will be able to send simple queries to office applications such as the corporate directory. The information is then automatically sent back to their supported wireless device of choice, all without having to modify back-end applications.

4.1.6 WebSphere Voice Server

WebSphere Voice Server provides speech recognition and speech synthesis (Text-To-Speech or TTS) engines, voice application development tools, and telephony platform connections to develop and deploy applications for use over telephones.

IBM WebSphere Voice Server for Multiplatforms V4.2 extends WebSphere Voice Server V3.1.1 with an array of new features and enhancements. It allows companies to leverage their existing Web infrastructures to enable easy voice access to existing Web applications by wireline and wireless phones. The software includes tools that enable developers to quickly develop and deploy voice-enabled e-business solutions, using industry standard technology such as Java technologies and VoiceXML.

WebSphere Voice Server for Multiplatforms V4.2 includes:

- Connection to many telephony platforms, including both WebSphere Voice Response for AIX and WebSphere Voice Response for Windows, Intel Dialogic, Cisco or Siemens HiPath, and Voice Server Speech Technologies for Windows and Linux.
- Voice Toolkit V4.2 for WebSphere Studio, which includes a VoiceXML editor, grammar editor, and a pronunciation builder, allows application developers to easily add voice technology to middleware applications. Tools for prototyping VoiceXML applications on a PC without a telephony server, and the necessary speech recognition and TTS engines for testing applications.

4.1.7 WebSphere Voice Application Access Server

WebSphere Voice Application Access (WVAA) V5 extends the popular WebSphere Portal infrastructure and programming model to a voice user interface (VUI), enabling users to access a wide range of business applications and data through a standard telephone or cell phone. The WebSphere Voice Application Access voice portal platform supports VoiceXML 2.0 using the WebSphere Voice Response for AIX VoiceXML 2.0 browser or other interactive voice response (IVR) platforms approved by IBM and compatible with VoiceXML 2.0.

WebSphere Voice Application Access V5 allows users to take advantage of the personalization features of WebSphere Portal to tailor their individual voice portals to fit their needs.

For the administrator, it provides a consistent framework for administering users and extending the same portal security features for authentication and authorization across multiple channels.

For the developer, WebSphere Voice Application Access V5 leverages the same Eclipse-based programming model as WebSphere Portal to build applications using VoiceXML technology.

With WebSphere Voice Application Access V5, developers can reuse existing code, business logic, and infrastructure, and enhance the development process to build VoiceXML applications.

4.1.8 Voice Response Server

WebSphere Voice Response is capable of supporting simple to complex applications and can scale to thousands of lines in a networked configuration. Applications can be developed using the native development environment or using standards-based development tools for Java or VoiceXML. WebSphere Voice Response Server is used to link the telephony users into the corporate Web environment.

4.2 Pervasive Device Adapter::Product mappings

This section describes the software components needed to move from designing a pervasive environment using patterns for the online pervasive portal scenario to implementing it.

To use this approach effectively, there are several questions that you need to consider:

- What kind of service would I like to offer? Does it require online access or could it also be used remotely when no connection is available (offline browsing)?
- What kind of back-end infrastructure is available? Usually online mobile device implementations are based on existing J2EE applications.
- How well would I like to support different devices? Select and target devices to focus on. It is more difficult to cover a wide range of different devices and their operations.
- What applications do I want to support? It is good to specify a subset of applications that will be mobilized. There is no need to enable all existing Web applications onto handheld devices.
- Are there enough skills to mobilize existing applications in my organization? Developing mobile applications requires knowledge of mobile-specific APIs and a focus on developing user-friendly interfaces to existing Web-based applications.

Pervasive Device Adapter::Product mapping=Windows

Figure 4-1 describes the products that are suitable for online pervasive portal solutions on the Windows platform.

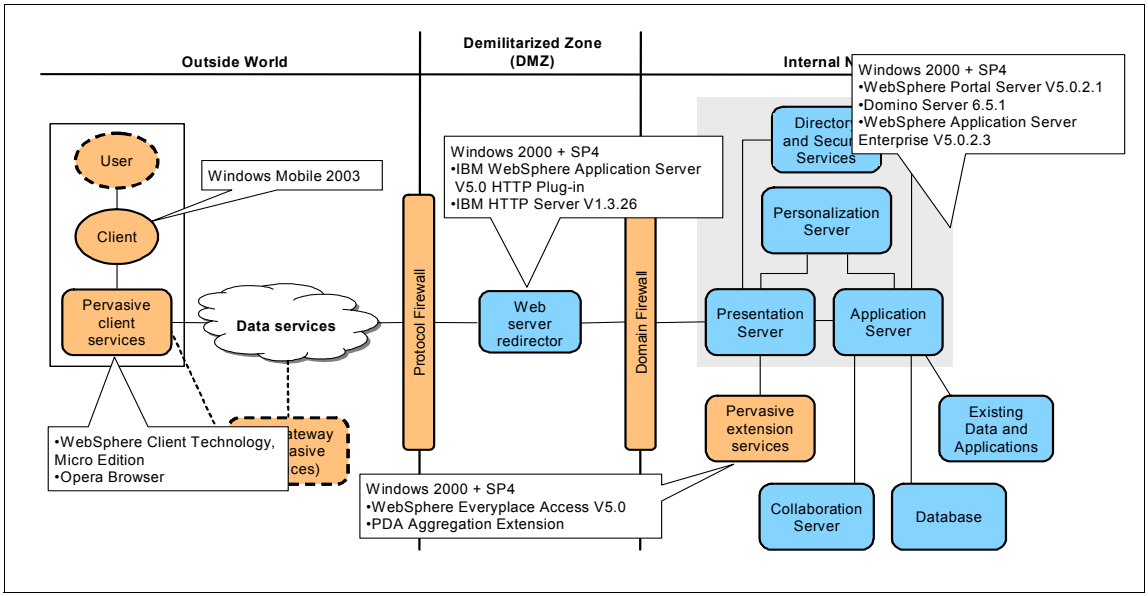


Figure 4-1 Pervasive Device Adapter::Product mapping=Windows

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.

By using a Web server redirector node, we can place the majority of the business logic on the internal network behind two firewalls. The redirector is implemented using the IBM HTTP Server and WebSphere Application Server Web server plug-in. The redirector serves static HTML pages and forwards requests for dynamic content to a WebSphere application server using the HTTP protocol.

The pervasive extension services node is used to extend the presentation server node. The pervasive extension services node contains the functions that interpret device platform and configuration information, and based on that information adapt and render the content for the specific device. It includes a PDA aggregator that extends the regular portal environment. PDA aggregation can customize existing portlets for display on PDA devices.

The back-end services like collaboration, database, and existing applications do not have any particular software mappings. In the case of the collaboration node,

the software used could be Lotus Domino server or Microsoft Exchange server. The Database node could be DB2, Oracle, or Sybase, for example. Existing applications could be any enterprise application.

Pervasive Device Adapter::Product mapping=AIX

Figure 4-2 shows the same implementation using AIX as the primary platform.

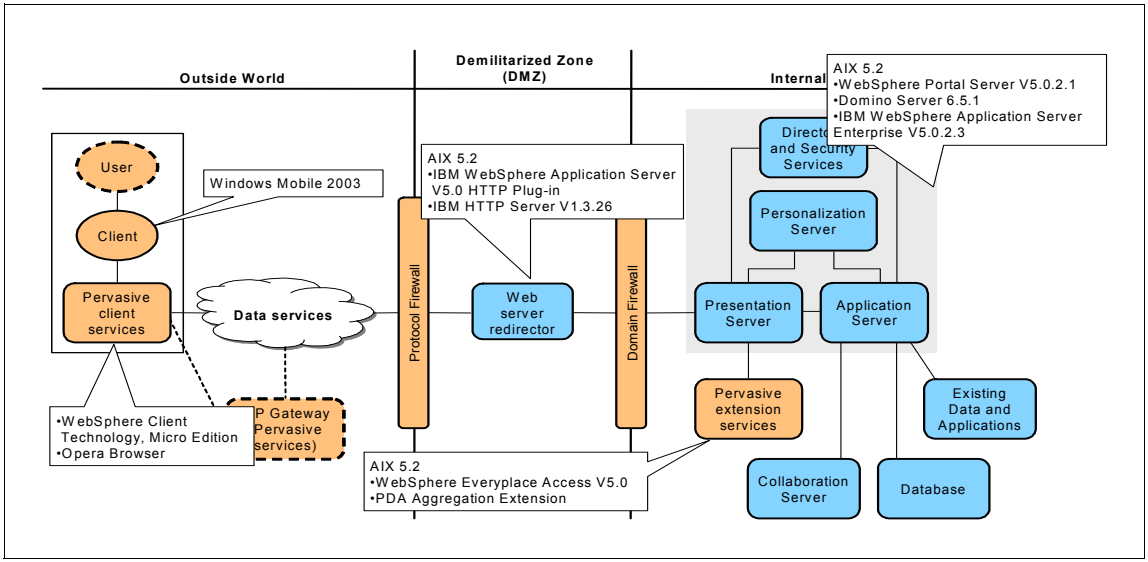


Figure 4-2 Pervasive Device Adapter::Product mapping=AIX

4.3 Pervasive Device Adapter=Voice::Product mapping

Voice-based solutions allow access to an enterprise and its data through the telephone networks. Companies have already made significant investments in developing their legacy applications and their e-business framework. By voice-enabling these already existing applications, companies can extend these same applications and databases to everyone, including customers and employees who may not have had access previously due to lack of access to the Internet. To make this implementation possible, we developed the Runtime pattern for voice solutions; see 3.2.2, “Pervasive Device Adapter=Voice::Runtime pattern” on page 41.

This scenario requires that the following issues be addressed:

- The structure of the voice application has to be designed carefully. The voice user interface and its usability are especially important issues in voice applications. There must be an easy way to access the data. This means that

the application should not contain too many selections or have too deep a hierarchy tree.

- The speech recognition infrastructure can be a complex process that relies on understanding the vocabulary and grammar usage for the application including the particular pronunciation of the vocabulary required.
- It is important to understand the iterative nature of application development. This involves incorporating as much data as possible about what users will say to the system and then gradually “tuning” the recognition to adjust to the full variation of speaker interactions with the system. Speaker variation can be the result of the “persona” of the system (how the user interprets the way the prompts are spoken), the user’s understanding of the requirements of the application, and the success of the system in providing error correction, etc.
- The use of text-to-speech versus recorded natural speech needs to be carefully evaluated in the application particularly for its use with mobile phone users.
- There is enough hardware power. Voice recognition requires significant processor speed and memory for operation. It is important to use the appropriate capacity planning tools for the application.

Pervasive Device Adapter=Voice::Product mapping=Windows

Figure 4-3 on page 67 describes the products that can be used to create voice solutions based on this pattern.

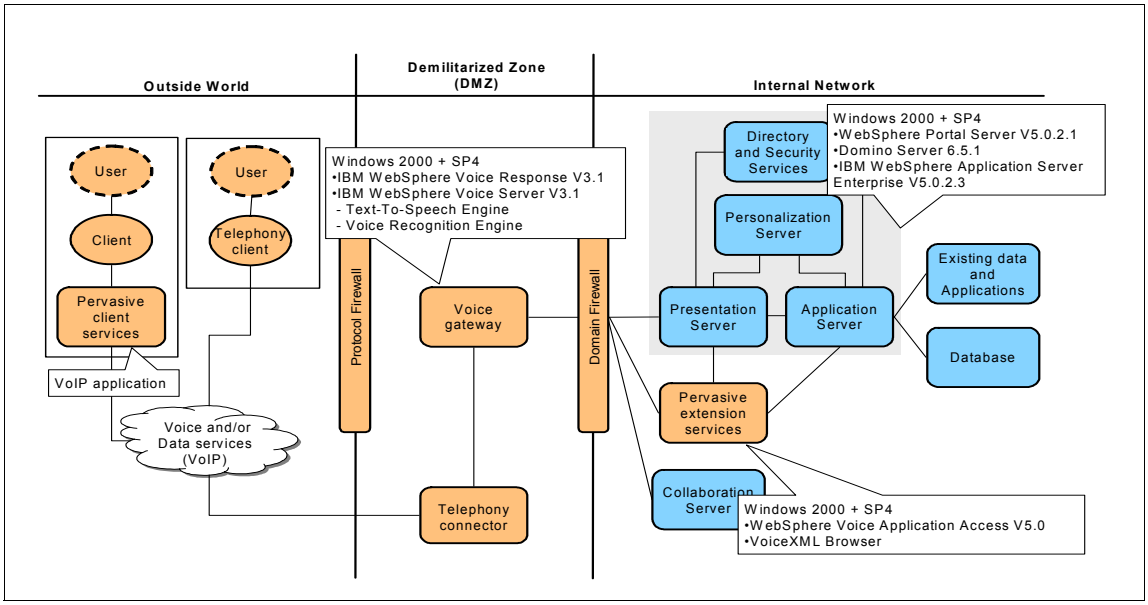


Figure 4-3 Pervasive Device Adapter=Voice::Product mapping=Windows

The most important nodes that are used in voice applications are the voice gateway, voice services, and pervasive extension services nodes.

- *Voice gateway node* is responsible for establishing, connecting, and routing the incoming phone calls from the phone network (analog, digital, or Voice over IP), and permitting access to the backend system. The telephony gateway node connects the phone input to the system and triggers a VoiceXML application based on the rules for the call and the user requests. The VoiceXML controls the *call flow* and manages the voice services to present prompts to the user and interprets user requests via voice services.
- *Telephony connector* is responsible for recognizing the user's selections and clarifying what the user would like to do. The telephony connector node has its own database where data is stored that is used for voice recognition purposes. For example, it contains a large vocabulary of words, their pronunciations, and their grammars. This node is used to send recognized requests to the application. Voice service contains two different engines:
  - *Speech recognition* recognizes caller utterances by means of one or more application-specific grammars or statistical language models.
  - *Text-To-Speech* is used to convert dynamic text into audible speech. Concatenative TTS uses short duration natural speech to create speech output. This technology sounds more natural because it is comprised of pre-recorded information from both a male and female voice.

- The *pervasive extension services node* is used to extend the presentation node. It means that the pervasive extension services node contains components that can be used for enabling voice support for the existing applications. Most often it means that a component can send and render the incoming and outgoing requests into the format that is suitable for voice services, for example, VoiceXML. The extension is basically an aggregator that transforms standard HTML pages and content into the VoiceXML format when the requests come through the Telephony gateway and vice-versa when responses need to be sent through that telephony channel.

The Presentation node and pervasive extension services node are connected to an existing application node using the application server node. Using this model, it is possible to reach a much larger audience—anyone with a telephone. With this pattern, company employees and customers can access information and conduct transactions without connecting to the Internet. Using a telephone, they can fully interact with the applications and Web site.

Pervasive Device Adapter=Voice::Product mapping=AIX

Figure 4-4 shows the same implementation using AIX as the primary platform.

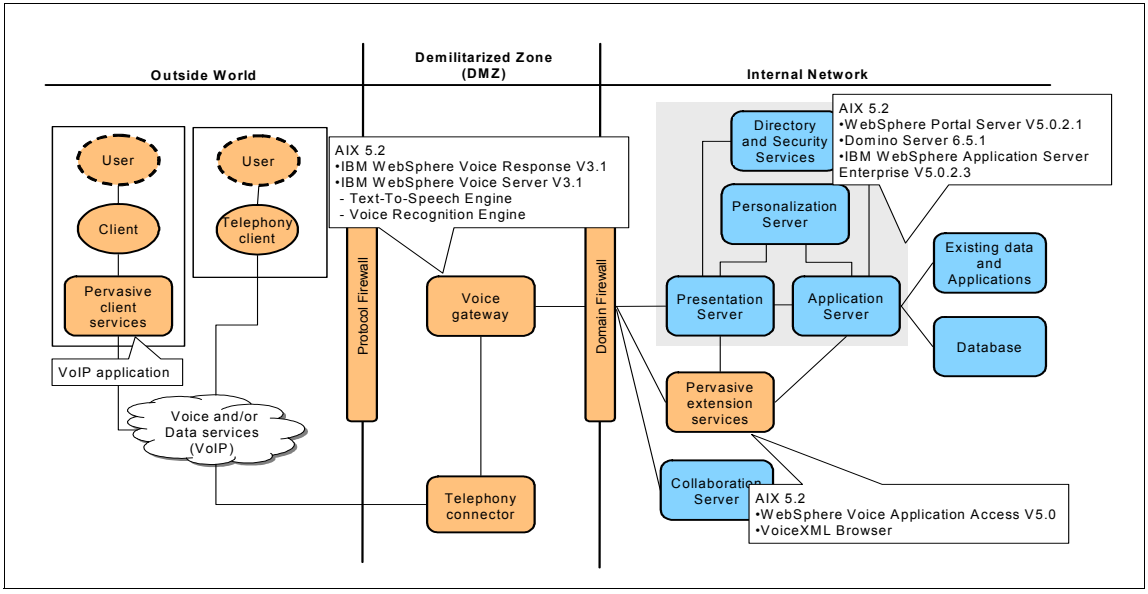


Figure 4-4 Pervasive Device Adapter=Voice::Product mapping=AIX

4.4 Rich Device::Product mapping=Pervasive device OS

It would be impossible to list all the device and operating system combinations on this diagram, and it is not the objective of this book anyway. Deciding on the client device and operating system for a company is one of the most difficult decisions when designing pervasive solutions.

Windows Mobile 2003 and Palm OS are the two major pervasive device operating systems supported by most of the IBM pervasive products. IBM and its pervasive products also support a few other devices and operating systems. Their support and availability vary product by product.

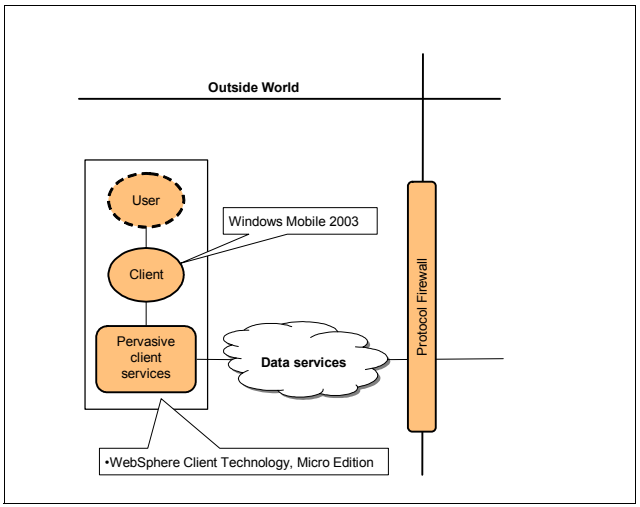


Figure 4-5 Rich Device::Product mapping=Pervasive device OS

Take the diagram as a starting point for other Product mappings.

4.5 Rich Device=Online::Product mapping=Device Management

Device management is used to manage mobile devices and maintain their applications by performing such actions as updating software, downloading configurations, and inventorying devices.

Device management is usually a complex task because of the wide range of devices that can be used and the rapidly changing nature of pervasive computing.

When planning this Runtime pattern, consider the following:

- How many different devices should we support? Explore how the devices will be used and what capabilities the devices will need. Supporting five types of devices is much easier than supporting 25. Realize that device management will cut costs in the long run.
- What are the applications that we need to implement? List all the applications that require some form of management operation.
- How can we standardize the company's device profile? To make device management easy, create different software and hardware profiles for the devices and standardize the supported devices in a manageable way.
- How are we going to control the devices? There should be a way to inventory the devices.

IBM WebSphere Everyplace Access offers a device management tool that is used for inventorying, packaging, updating software, and remotely installing software on mobile devices. This component can be migrated to the existing Tivoli Configuration Manager.

Rich Device=Online::Product mapping=Device Management, Windows

The following diagram shows Product mapping for the Windows platform.

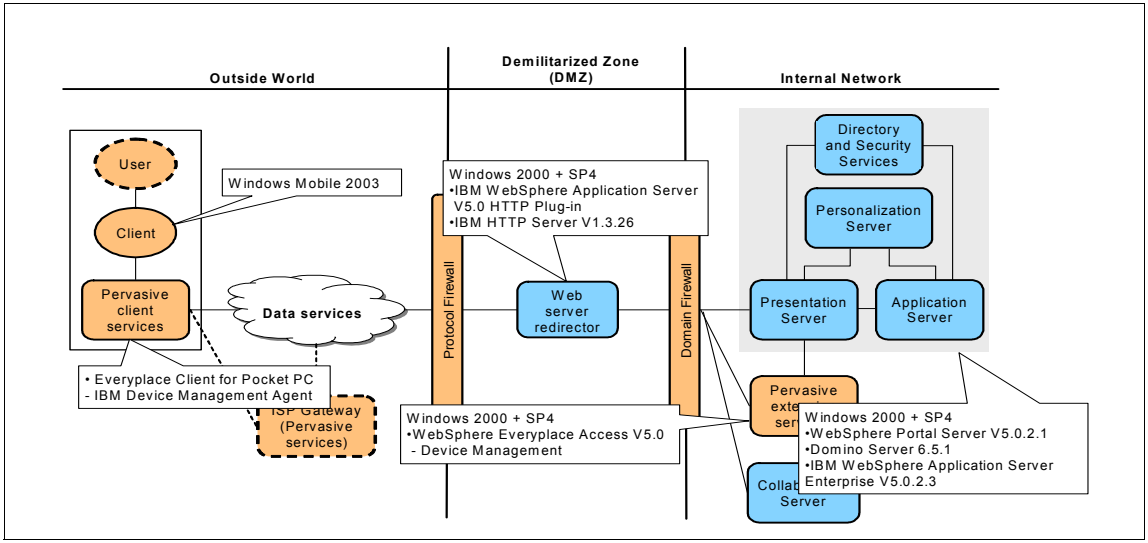


Figure 4-6 Rich Device=Online::Product mapping=Device Management, Windows

The pervasive client services node receives any update or inventory requests from the pervasive extension services node and updates the device applications or device operating systems.

Rich Device=Online::Product mapping=Device Management, AIX

Figure 4-7 shows this same implementation using AIX as the primary platform.

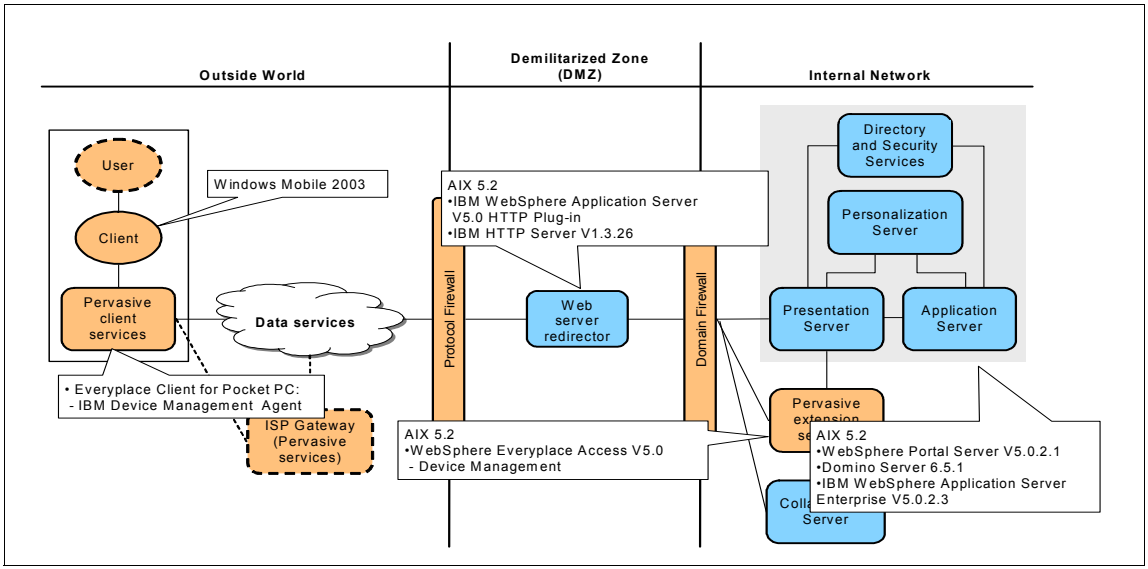


Figure 4-7 Rich Device=Online::Product mapping=Device Management, AIX

4.6 Rich Device=Store and forward::Product mapping

When using this pattern you need to consider the following:

- What devices do you intend to support? Focus on a specific set of devices that suit the business requirements and support the technology used in your particular implementation.
- What are the business-critical applications that you need to support? For example, you do not want to mobile-enable an entire CRM application. You only want to enable those parts that could promote and extend the application.
- How can you ensure data security and trusted synchronization between the client and server? You should consider how the data is transferred between the client and server.

- What is the complexity of the applications you would like to implement? You need to consider the structuring of your application. Which business logic is suitable to include on the client side and what should be left on the server side?

The rich device application could contain a client-based local application with business logic in it and a local repository for accessing the data. It could also be a very simple offline forms application that allows users to input the data in offline mode through the form and submit it to a local repository (usually it is file system). The data, which is stored into the local database or file system is later synchronized with the back-end server. It is also possible to use a verified transaction messaging operation for sending data. This scenario is recommended if the data requires once-only messaging. This kind of mechanism is used when the data is very critical, for example, credit card information.

This pattern can be used to describe a very complex pervasive application that is used by businesses to support their mobile sales force workers in their daily routines. It is made up of the following parts:

- Rich device application: This contains the user interface, business logic, and local repository on the device side. Sales workers could find their customer data, create future tasks based on their selections, and store this data for future handling. These functions are described in the user, client, and pervasive client services nodes.
- Pervasive services: This takes care of transferring and synchronizing data between the client device and the back-end server. This function needs to authenticate and authorize the user and client towards the existing corporate environment and update the existing back-end application with the data that is being sent from the client. It is also responsible for sending updated data back to the client.

More information about this simple offline forms scenarios can be found in 13.4, “Sample application development” on page 295.

This same Runtime pattern could also be used to simplify implementations for offline applications. The next example describes the workflow steps for simple offline applications, which would use this Runtime pattern.

1. User fills in an offline-based form using his pervasive device.
2. After filling in that form, he can submit it and continue to fill in the next offline form with different content.
3. The data from the filled-in and submitted forms is stored in the device’s local storage location (file or database) for future handling.

4. When the device is connected to the network, the user could synchronize that device with the server-side application by sending the completed form data from the client to the server.
5. The server accepts the incoming data and sends a confirmation that the data has been received as well as any additional responses back to the client.

More information about this simple offline forms scenario can be found in 11.3.1, “General considerations for intermittently connected applications” on page 228.

If the data being used on the mobile devices is very crucial or high availability is needed, then clustering your servers is an appropriate solution. Clustering distributes certain logical nodes onto different physical machines. The following Runtime pattern illustrates an effective way to visualize the environment by building up the functional and operational nodes. This makes it easy to pinpoint which nodes require splitting and which parts might require clustering to solve high-availability problems.

Rich Device=Store and forward::Product mapping=Windows

The following diagrams shows the Product mapping for the Windows 2000 platform.

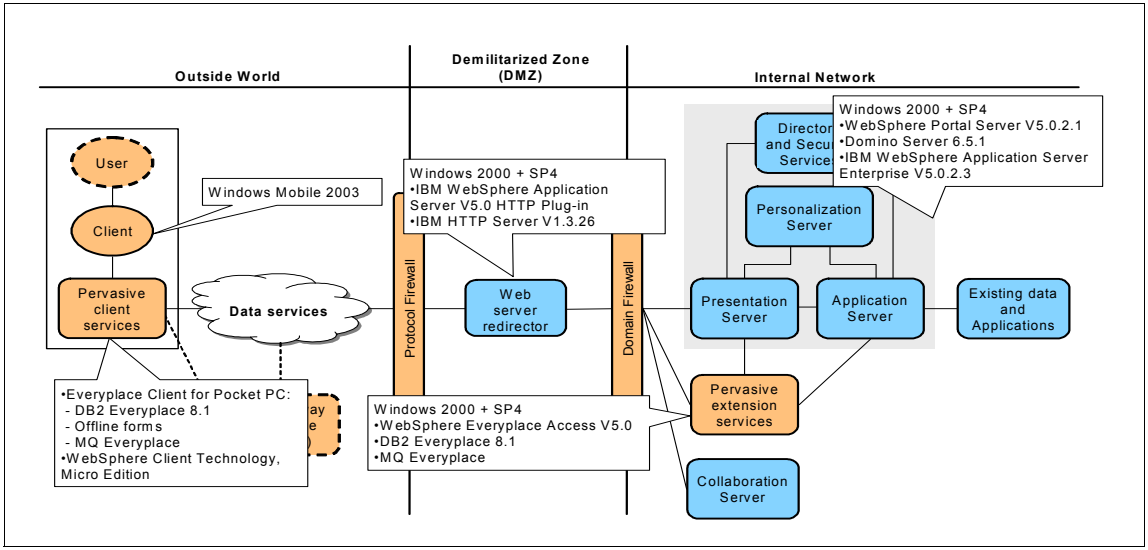


Figure 4-8 Rich Device=Store and forward::Product mapping=Windows

The pervasive client services node contains a lot of components that are used to support rich device applications. WebSphere Client Technology, Micro Edition can support many standardized technologies that are used in the pervasive world. DB2e and MQe can support alternative transactional quality of service

between client and server. Offline forms can be used to gather data on the client side. When a network connection is available, these stored offline forms are sent to the server.

On the server side, the pervasive extension services node handles the incoming data that is sent from the client. The pervasive extension services then treat the data and sends it to an existing application or database for further processing.

Rich Device=Store and forward::Product mapping=AIX

Figure 4-9 shows the same implementation using AIX as the primary platform.

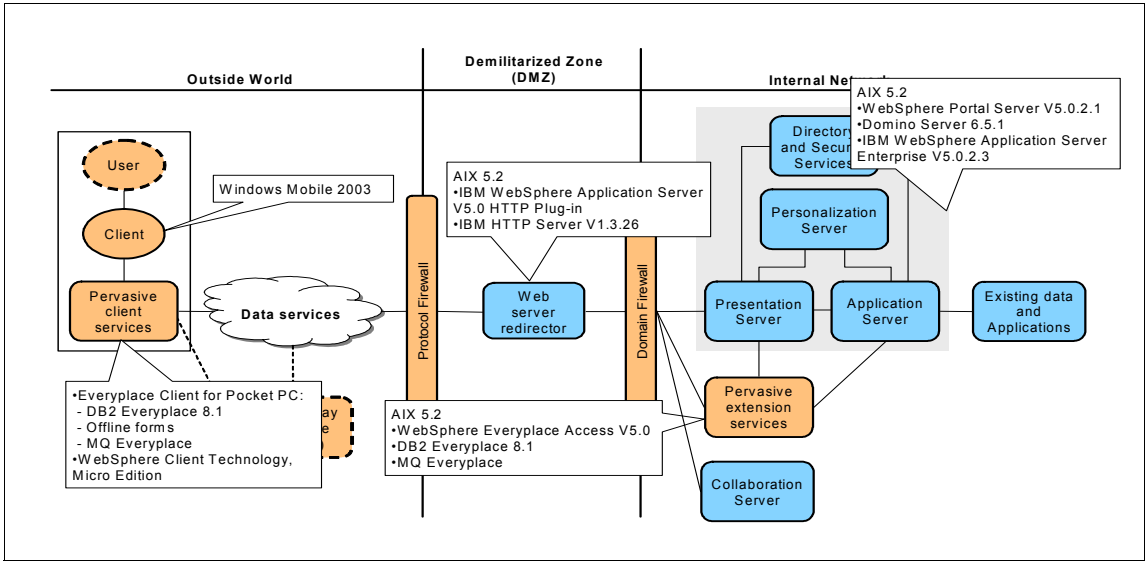


Figure 4-9 Rich Device=Store and forward::Product mapping=AIX

4.7 Rich Device=Store and forward::Runtime mapping=PIM and e-mail

Because synchronization requires a lot of I/O capacity for the server, the pervasive extension services node is split into two different nodes. By splitting that node, we get a more reliable and scalable environment. By dividing the node into separate nodes we could offer online portal users, and PIM and e-mail users a better response time and separate environments. Because of this, if the synchronization service goes down, it does not affect the portal side, and vice versa.

There are several questions that you need to consider with this pattern:

- Do we need to support offline access? In many cases, online PIM and e-mail access is adequate and offline access is not needed.
- Who is the targeted audience for using this service? In many cases this service is not extended to everyone within a company. We want to limit the users who have access based on certain criteria.
- What are the effects of extending synchronization capabilities into the existing environment? We want to make sure that this service does not harm the existing environment. For example, synchronization can be affected by unstable mail servers and slow or overloaded networks.
- Should I use both online and offline access for accessing PIM and e-mail data? In some cases there might be a reason to use partial online capabilities. You need to consider this option if this kind of connection is needed and how this will impact data synchronization services.
- What are the specific devices that you are going to consider in your implementation? We need to specify the devices and make sure that those devices can properly support these functions.

Rich Device=Store and forward::Runtime mapping=PIM and e-mail, Windows

The following diagram shows the Product mapping for the Windows 2000 platform.

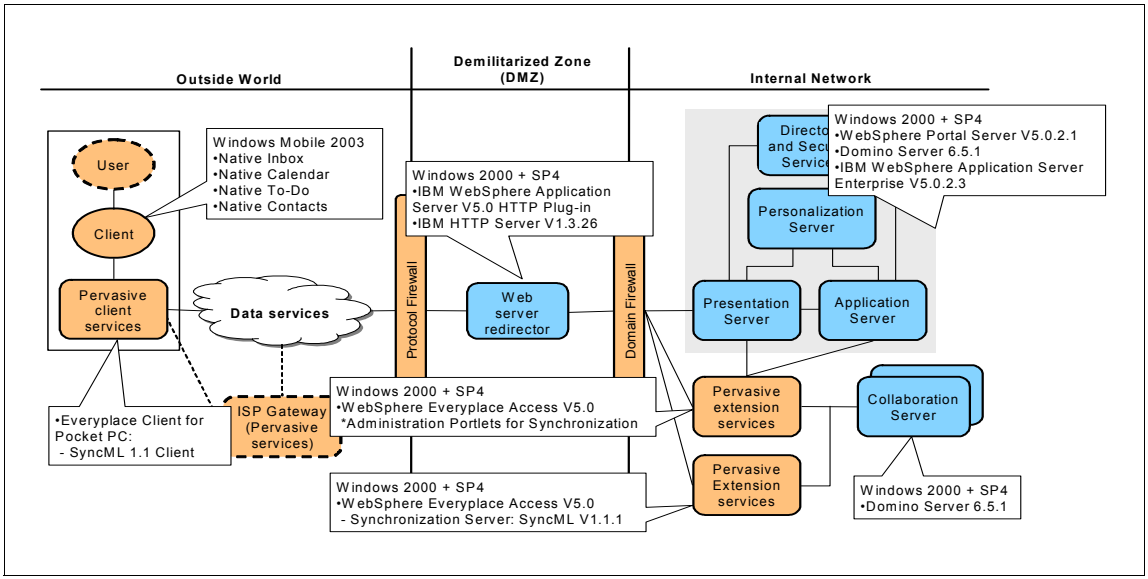


Figure 4-10 Rich Device=Store and forward::Runtime mapping=PIM and e-mail, Windows

By using a Web server redirector node, we can place the majority of the business logic on the internal network behind two firewalls. The redirector is implemented using the IBM HTTP Server and WebSphere Application Server Web server plug-in. The redirector serves static HTML pages and forwards requests for dynamic content to a WebSphere Application Server using the HTTP protocol.

In this case, we do not use WebSphere Everyplace Connection Manager V5 to tunnel the requests between the client and server, although it is the recommended way to access a corporate network from the outside world.

In a real-world scenario, allocating resources in this manner is a good idea for those companies that intend to serve PIM and e-mail services to their employees without giving full access to other portal-based online solutions. In that case, there would be a resulting bottleneck at the synchronization server, which offers offline access to employers PIM and e-mail databases.

Rich Device=Store and forward::Runtime mapping=PIM and e-mail, AIX

Figure 4-11 on page 77 shows the same implementation using AIX as the primary platform.

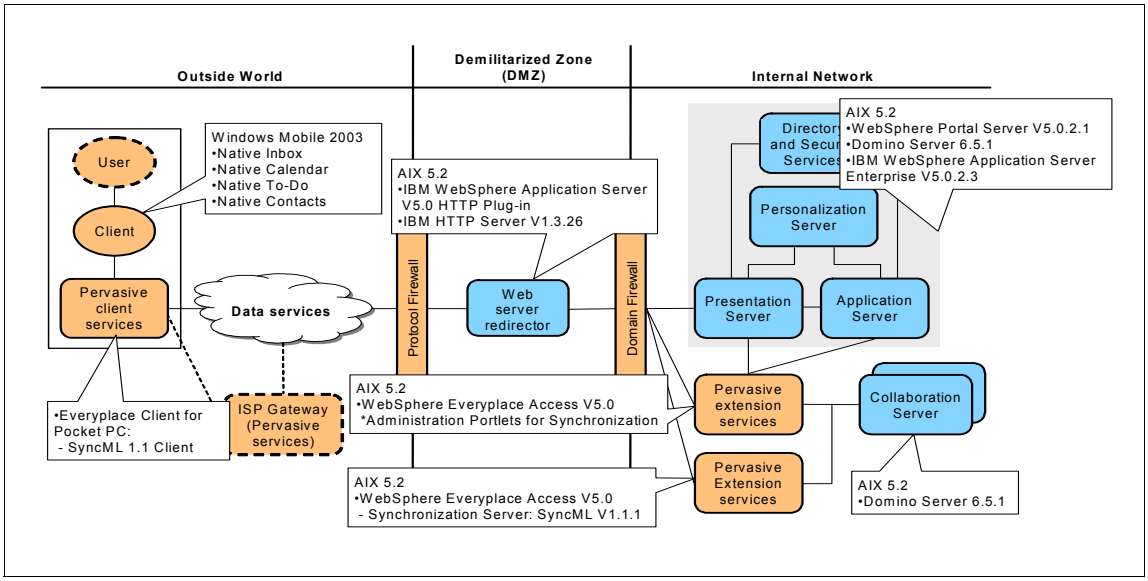


Figure 4-11 Rich Device=Store and forward::Runtime mapping=PIM and e-mail, AIX

4.8 Pervasive Connectivity runtime pattern::Product mapping

This section describes the products that are used for encrypted, trusted, and securely tunnelled connections between the device and server. When planning for this pattern, consider the following:

- What kind of security and connections will we support? Usually, business data is critical, and customers require a secured and encrypted method of accessing it. The most common way to do this is to create a virtual private network (VPN) connection. A VPN creates a secure tunnel on the public Internet to support the transactions between the client and back end.
- How can we cover areas with little or inconsistent network coverage? We need to cover and maintain the user's sessions and provide for a user-friendly experience if the network is not available all the time.
- Do we have to authenticate our users to the network and the device? We need to consider the level of authentication and authorization. Usually it is the case that companies require authentication for both the applications and the network as well.
- Of all connectivity options available at a particular location, can we offer the most appropriate network? We would like to offer the most suitable network that is available for the device. If a device notices that it is now on a corporate

wireless LAN, we can allow the device to use that network rather than a slower and potentially more expensive cellular network.

The IBM WebSphere Everyplace Connection Manager V5 does not create a tunnel for every application. It secures the whole communication between the client and the server. This means that the user is able to use multiple different applications (for example, Instant Messaging, mobile enabled online portlets, data synchronization, and PIM and e-mail synchronization) through the tunneled connection without configuring each application's security settings separately.

Pervasive Connectivity runtime pattern::Product mapping=Linux

Figure 4-12 describes how the connectivity products are mapped into the Runtime pattern for Connectivity using the Linux platform.

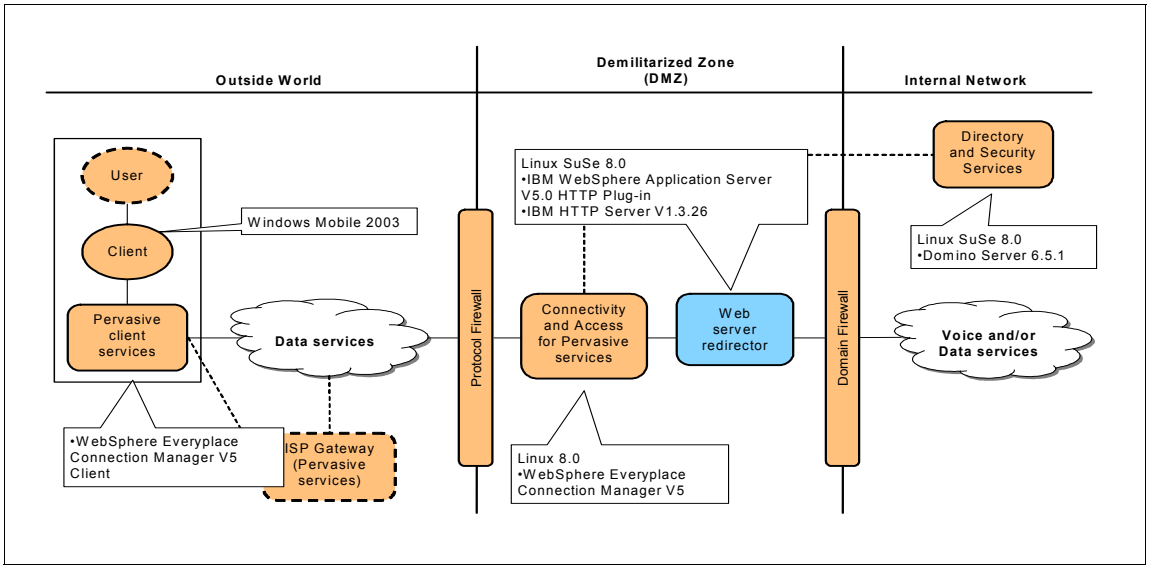


Figure 4-12 Pervasive Connectivity runtime pattern::Product mapping=Linux

The connectivity and access for pervasive services node lies in the DMZ to provide a more secure implementation for the whole pervasive environment. The Pervasive client services is needed because it communicates with the connectivity and access for pervasive services node using a tunneled connection.

Pervasive Connectivity runtime pattern::Product mapping=AIX

Figure 4-13 shows this same implementation using AIX as the primary platform.

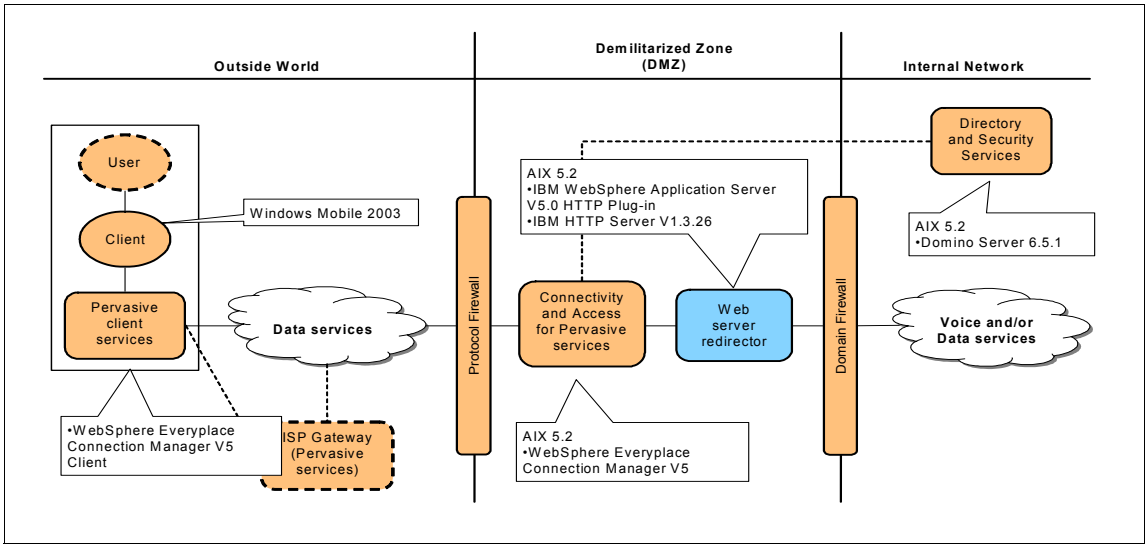


Figure 4-13 Pervasive Connectivity runtime pattern::Product mapping=AIX

The connectivity and access for pervasive services node is added in front of the Web server redirector node. This is done so that all incoming requests will be routed from the client to the corporate intranet through the trusted tunnel. The connectivity and access for pervasive services node controls the user and device network authentication and authorization. When the network access is trusted, the client can authenticate towards the application layer and can get the access to the backend applications. The single sign-on (SSO) mechanism is also possible. Single sign-on means that the network layer and application layer authentication and authorization is done inside the connectivity and access for pervasive services node. If this is the case, the connectivity and access for pervasive services node sends the application layer authentication request to the existing directory and security services node automatically and the user does not have to insert her application layer authentication information.

To enable the single sign-on between the connectivity server and existing authentication and security server you could use the following technologies:

- Trust Association Interceptor (TAI) to enable companies to incorporate their IBM WebSphere environments, including IBM WebSphere Application Server and IBM WebSphere Portal, into a unified and centrally managed security infrastructure. Connection Manager V5 provides a TAI plug-in for use with WebSphere Application Server. This allows single sign-on from mobility clients connecting to WebSphere Application Server through a VPN.
- Lightweight Third Party Authentication (LTPA), a mechanism for achieving single sign-on across the Internet domain that contains users' resources.

Connection Manager is able to generate LTPA tokens that may be used as the mechanism for establishing single sign-on between Connection Manager and other supported applications for HTTP Access Services using HTTP access client authentication method. This is supported by both RADIUS and LDAP-bind type authentication profiles.

4.9 Pervasive Solutions composite pattern::Product mapping

This section discusses a full mapping for products with the Pervasive Solutions Runtime pattern.

This Runtime pattern is very powerful when the customer also would like to use certain kinds of intelligent notification messaging such as that which could be used to send notifications to employees when they receive an urgent e-mail message. Notification services could also trigger an alert to a user whenever a stock price reaches a certain level.

Notification works as follows:

1. Notification tracking checks via an existing database or Web services application that a current value has gone past a certain threshold. These functions are described in nodes: Existing data and application, application server.
2. It then triggers an action that launches the notification. These functions are described in nodes: Application server, pervasive extension services.
3. Now the notification engine knows what to send, but it still needs to find the proper device to send that data to. This operation is done inside the pervasive extension services node.
4. The final step is to deliver that notification to the proper device application using SMS, e-mail, or an Instant messaging application where the user can receive that message. These functions are described in nodes: Pervasive extension services and Pervasive client services.

When notification services are used within the environment, there must be well-defined interfaces for channels to display these notifications. For example, Instant Messaging requires a supported PDA or mobile phone and a Web-enabled cellular network connection. Voice-based notifications require a network connection with which to dial a phone or cellular phone. These need to be in place for any notifications to be sent through these channels.

Implementation considerations

When someone starts thinking about implementing a pervasive environment, the user should be aware of certain issues:

- What devices is your organization going to support? Although many middleware implementations can support multiple devices, it is always a good practice to consider the most important features that will be used and ensure that the selected devices support these features. Deciding to support a variety of different devices usually means that you must make some compromises on application support.
- How do you manage the devices? Distributing updated applications and device management are two major issues to be considered when building a properly functioning and easily maintainable pervasive environment. It is no use having a good client-based application if it is hard to distribute and update on many devices.
- Internet Service Provider services and the network connections into the corporate DMZ must be in place. In many cases there is little focus on this aspect of the project. Only at the end of the project do certain issues arise, such as having proper gateways to support different channels such as Blackberry, GPRS, or voice.
- Have security and authorization issues been completely and adequately addressed? Security is one of the most crucial aspects of a pervasive environment. Usually very critical information is exchanged through many different channels. There must be a way to create a secure and encrypted tunnel from the device to the corporate network. Furthermore, it is very common for employees to lose or misplace their handheld devices or have them stolen. Because of these cases, there should always be a well-defined client and device authentication method provided on the server-side.
- Building a pervasive environment usually means that we want to extend the existing environment to serve different devices. This means that we want to take advantage of the existing components that are in place already. This includes aspects such as user authentication and interfaces to ERP and CRM applications. Very careful and exact definition and implementation of these allows an organization to benefit from these components without any major changes to the existing environment or architecture.

Pervasive Solutions composite pattern::Product mapping=Windows

Figure 4-14 describes the products that you can use to implement a comprehensive pervasive environment on the Windows 2000 platform.

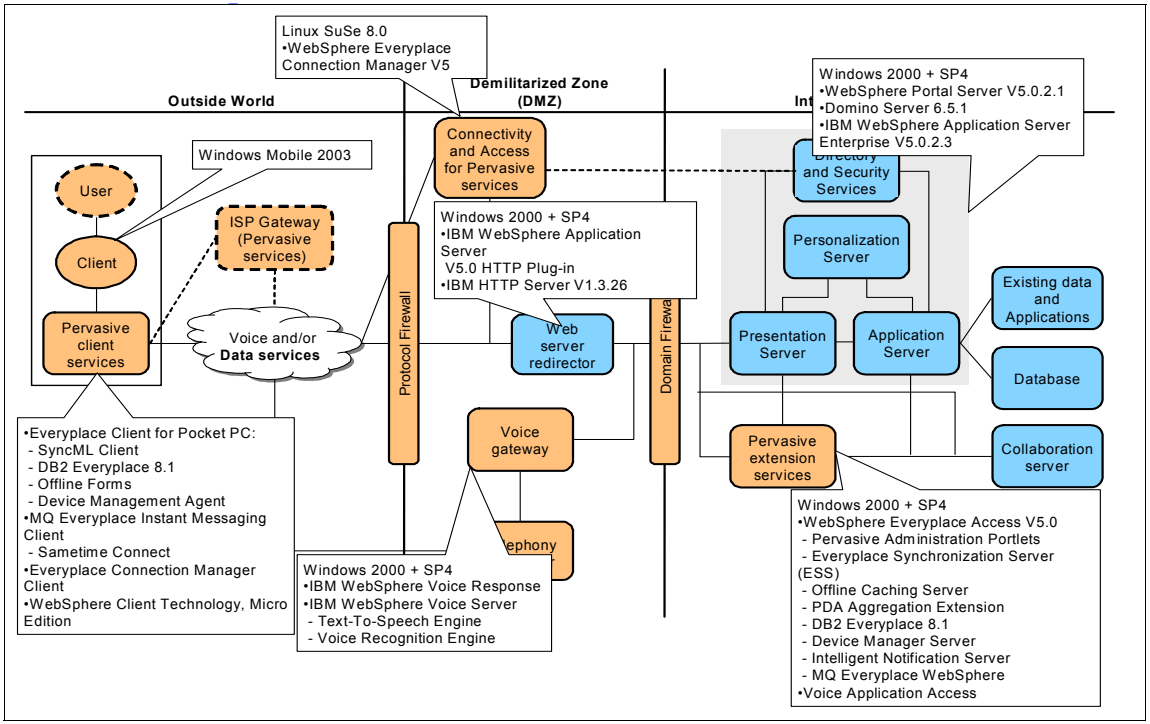


Figure 4-14 Pervasive Solutions composite pattern::Product mapping=Windows

Pervasive Solutions composite pattern::Product mapping=AIX

Figure 4-10 shows the same implementation using AIX as the primary platform.

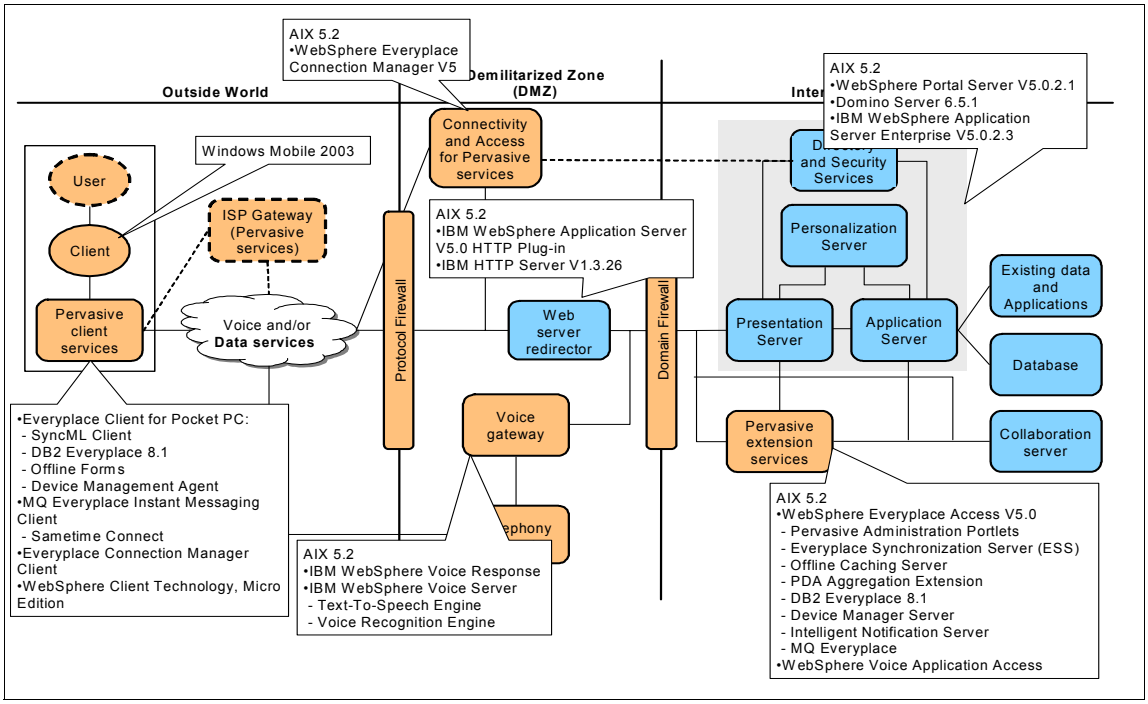


Figure 4-15 Pervasive Solutions composite pattern::Product mapping=AIX

In order to get a good understanding of the protocols that are used for Pervasive solutions, Figure 4-16 on page 85 describes the network protocols used for the Windows 2000/AIX 5.2 and Windows Mobile 2003, Palm OS implementations:

- GPRS/UMTS: General Packet Radio Service (GPRS) and Universal Mobile Telecommunications Service (UMTS) carry the end user's packet data from mobile devices to external packet data networks and visa versa.

  UMTS is a third-generation (3G) broadband, packet-based transmission of text, digitized voice, video, and multimedia at data rates up to 2 megabits per second (Mbps) that offers a consistent set of services to mobile computer and phone users no matter where they are located in the world.

- HTTP/HTTPS: Hypertext Transfer Protocol (HTTP V1.1), or Hypertext Transfer Protocol Secure (HTTPS - HTTP V1.1/SSL V3), is used from the user's Web browser to the HTTP server in the Web server redirector node.

  HTTP, or HTTPS, is also used from the WebSphere Web server plug-in in the Web server redirector node to the Web container in the application server node.

- LDAP: The application server uses Lightweight Directory Access Protocol (LDAP V3) to access the LDAP server in the directory and security services node.
- JDBC: The application server uses a Java Database Connectivity (JBDC V2.0) driver to access the database.
- IMAP: Internet Message Access Protocol is a method of accessing electronic mail or bulletin board messages that are kept on a (possibly shared) mail server. In other words, it permits a "client" e-mail program to access remote message stores as if they were local.
- IIOP: The IIOP (Internet Inter ORB Protocol) specification defines a set of data formatting rules, called Common Data Representation (CDR), which is tailored to the data types supported in the CORBA Interface Definition Language (IDL). Using the CDR data formatting rules, the IIOP specification also defines a set of message types that support all of the ORB semantics defined in the CORBA core specification. It provides a single protocol for interoperability between the Enterprise and the Web.
- DIIOP: Domino IIOP allows Domino and the browser client to use the Domino Object Request Broker (ORB) server program. The Domino ORB processes the applet requests and transmits the information to the browser client to communicate.
- OMA DM is a specification created by the Open Mobile Alliance (OMA) organization for device management of wireless devices. It is a standardization that allows Device Manager to write one protocol engine to encode and decode the messages passed between Device Manager and the OMA DM device agent.
- Voice over Internet Protocol (VoIP) is a protocol that allows you to make telephone calls using a computer network, over a data network like the Internet. VoIP converts the voice signal from a user's telephone into a digital signal that travels over the Internet and then converts it back at the other end. This allows the user to speak to anyone with a regular phone number.
- Mobile Internetworking Control Protocol (MICP) is a protocol that is used to exchange control information between the voice gateway and voice service.

Figure 4-16 on page 85 shows the protocol and technology mappings for pervasive solution solutions.

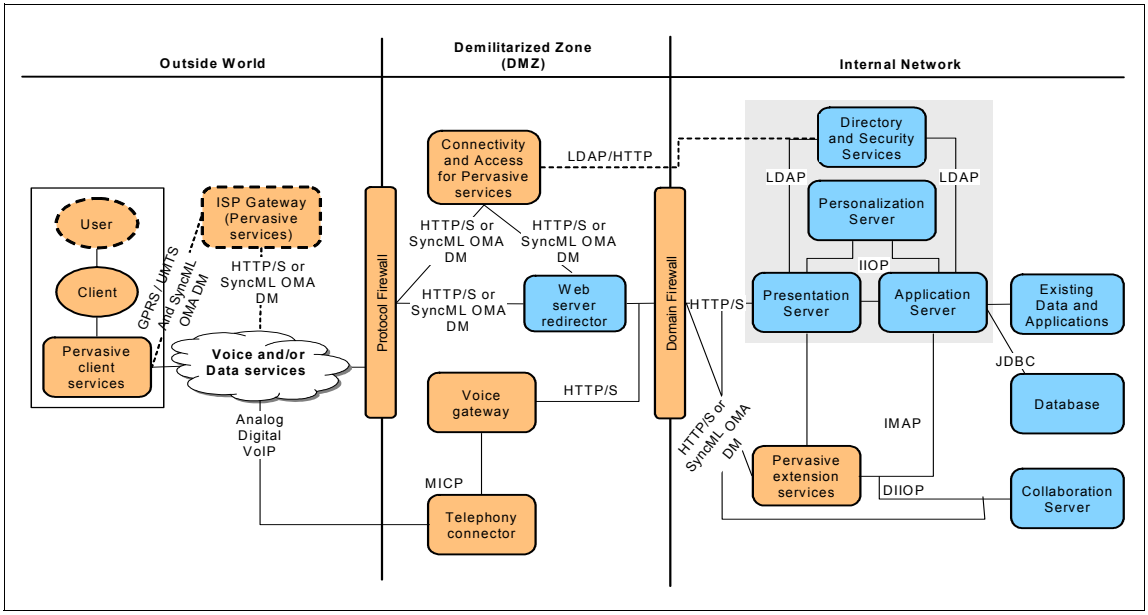


Figure 4-16 Network protocol mapping for pervasive solutions

5

Chapter 5.

ITSO Railway sample overview

This chapter provides an overview of the business domain used in the technical scenario chapters in this section. This chapter identifies the enterprise and the business problems they intend to solve by incorporating pervasive computing technology into their business environment. A subset of these business problems is detailed in the technical scenario chapters, which further defines and implements a portion of the mobile solution.

5.1 ITSO Railway

ITSO Railway, a fictitious company, wants to modernize various aspects of their business and take advantage of the capabilities pervasive computing has to offer. They want to give their mobile workers and customers mobile/wireless access to enterprise applications and data. ITSO Railways has a variety of mobile employees such as traveling executives, delivery personnel, and train conductors. ITSO Railways analysis has identified the following situations that will benefit from pervasive computing technology:

- Mobilize inventory management by replacing existing paper-based Train Supply forms with mobile solution.
- Provide executives mobile PIM and e-mail support while they are on the road.
- Monitor critical equipment for potential problems or malfunctions.
- Alert maintenance personnel of potential equipment problems or malfunctions.
- Provide mobile customers access to railway information by extending existing railway portal to mobile devices.
- Automate ticketing on train by providing train conductors with a mobile ticketing application.
- Maintain mobile devices easily by providing remote maintenance of mobile devices.
- Secure mobile devices by providing security for critical data and applications on the mobile device.
- Provide voice access to railway information to customers using phones (voice) to obtain information.

5.1.1 Business value to ITSO Railways

ITSO Railways believes the business value of using pervasive computing technology will improve their business and their bottom line. The overall business values provided by using the technology are:.

- Automate and stream line various business processes
- Increase data accuracy and reduce data latency.
- Increase mobile worker productivity by providing them with immediate access to information.
- Increase customer satisfaction and quality of service by providing them with various types of access to frequently requested information.
- Reduce paper forms and paper handling by automating manual processes.

- Enable mobile workers to work more efficiently and effectively by providing them with access to enterprise applications and data.
- Reduce call center costs generated by mobile phone users calling in to get timetable data with peaks occurring at specific times of the day.

5.2 General requirements

ITSO Railways has established the following general requirements for any mobile solutions being created.

The key requirements are:

- Enhance existing applications and methodologies currently available.
- Use standards-based technologies and products wherever possible.
- Integrate all new solutions with the existing business applications and databases.
- Use the techniques and the technology appropriate to the end-user needs.
- Provide devices and applications that are user friendly and easy to learn, use, and maintain.
- Ensure the security of the enterprise and any business-critical data and applications wherever they are located within the ecosystem.
- Use tools and automated techniques in any development projects that streamline and support the development life cycle.

5.3 Provide executive PIM and e-mail support

ITSO Railway executives need access to their Personal Information Management (PIM) and e-mail while traveling. Because executives have very dynamic information needs, they must have access to their schedules and mail whenever and wherever the need arises. They must be able to maintain a single copy of their e-mail and be assured the PIM updates will be posted to their office PIM application server.

5.3.1 Key requirements

Key business requirements for giving executives mobile access to PIM and e-mail are:

- Provide executives with access to office services such as PIM and e-mail.

- Ensure that any modifications are handled so the executives only work with the information once.
- Protect the user from any technical details

Key IT requirements are:

- The pervasive solution integrates with the existing PIM and e-mail system.
- Provide an easy-to-use and learn user interface that is familiar to them.
- Buy an existing mobile PIM and e-mail solution.
- Ensure that the new PIM and e-mail solution is easy to install and maintain.

5.3.2 Example application scenario

Peter, a railway executive, has a very dynamic schedule and is constantly getting e-mail that he needs to respond to quickly. Peter uses the ITSO Railways PIM applications to keep his business files up to date. Peter is constantly getting requests from his superiors to create reports for them. In order to keep track of these requests Peter uses the To Do list PIM application. When he is in the office he uses his desktop computer to review, update, and track these to-do list items. He wants the list to be in sync and accurate on both his mobile device and his office computer.

Example use case

In this example use case:

1. Peter logs onto his mobile device.
2. Peter selects his To Do list application.
3. Peter reviews the existing items.
4. Peter updates the status of an item to done.
5. Peter adds a new item to the to-do list.
6. Occasionally the mobile device is synchronized with the server, so Peter's updates are processed on the PIM application server.

5.4 Mobile customer access

Customer satisfaction is key to ITSO Railways success. They would like to give customers mobile access to railway schedules, time tables, price changes, and any promotions currently available.

5.4.1 Key requirements

The key business requirements are:

- Increase customer satisfaction by providing them anywhere access to railway information using a variety of devices.
- Provide customers with mobile access to railway information.
- Reduce calls to customer service.
- Make information available 24 hours a day 7 days a week.
- Provide the latest information to allow customers to make informed decisions.

The key IT requirements are:

- Integration with and extend existing railway information system and train scheduling system.
- Make the application easy to use and learn.
- Extend the existing Web/portal system to mobile devices.
- Use tools that support the end-to-end development process and simplify development.

5.4.2 Example application scenario

George uses the railway systems frequently. He recently purchased a PDA and uses it to access various Web sites. He would like to use the PDA to access the railway information so he can use his time better when he is out adventuring.

Example use case

In our example use case:

1. George logs on to his PDA.
2. He accesses the ITSO Railways Web site.
3. He selects train schedules.
4. He looks for the train schedule for trains from Raleigh to Charlotte.
5. He reviews the train tables available.

5.5 Mobile inventory management

ITSO Railways delivery personnel provide supplies such as food, dining ware, soap, and toiletries to each train when it enters a station. As the delivery person replenishes the supplies, he fills out a paper form indicating the quantity for each item provided to that train. At the end of the day, these forms are turned into the

station office, then input into the Inventory management system. Sometimes the delivery trucks do not have all the necessary supplies for a train, causing unnecessary delays or trains in transit without the necessary supplies for their customers.

5.5.1 Key requirements

Key business requirements are:

- Automate record keeping for train supplies by eliminating paper forms.
- Streamline inventory management process.
- Provide a user-friendly application for the delivery personnel.
- Provide delivery personnel with their own truck's item inventory.
- Increase inventory accuracy by reducing entry errors.

Key IT requirements are:

- Integration with existing inventory management system.
- Provide simple user-friendly interface and application.
- Minimize the programming and technical skills to create and maintain the solution.
- Make the application easy to administer.

5.5.2 Example application scenario

Dan, the train delivery person, provides supplies to trains when they enter the Raleigh station. He is responsible for checking the supplies in each passenger car of the train and replenishing any storage bins that are low on supplies. He must document the number of supplies he replenishes on each car of each train he services. As he replaces the supplies he updates the Train Supply Form with the exact item and the item's quantity for each car on a train. Because Dan is very busy going from train to train all day long, Dan turns in his Train Supply Forms at the end of each day. The information on the forms is then entered into the inventory management system so the supply inventory in the station warehouse can be kept accurate and supply orders can be made as needed. When Dan does not have enough supplies for a particular train, he must call another delivery person to provide the needed supplies or allow the train to leave the station without all the supplies it should have.

Example use case

In this example use case:

1. Dan (the delivery person) logs onto his mobile device.
2. Dan accesses the Train Supply Application.

3. Dan sees the Train Supply Form and enters the train number.
4. As Dan goes through the train, he checks each car's supply bins to see if any particular items are needed. As he replaces each item, he updates the online form with the quantity of each item replenished in each car.
5. Occasionally the mobile device is synchronized with the server, so Dan's updates can be processed.

5.6 Monitor critical equipment

Various equipment is critical to the railway, such as trains with complex engines and rail lines with very sophisticated electrical systems. This equipment must be operational at all times. ITSO Railways would like to place sensors at key locations in their railway system and on trains to collect data and monitor the equipment for potential malfunctions before they happen.

5.6.1 Key requirements

The key business requirements are:

- Provide sensors that monitor railway business-critical equipment.
- Sensors must signal any changes in equipment status.
- Sensors must be reliable and available 24 hours a day for 7 days a week.
- Sensors must identify the particular piece of equipment they are associated with and clearly state the situation as it occurs.
- Sensors must be roughed and inexpensive.

The key IT requirements are:

- Integration with maintenance system.
- Sensors must be easy to install and maintain.
- Sensors must be programmable so they can be reused.

5.6.2 Example application scenario

The railway switching system is critical to the movement of trains within the train yard. If a switch malfunctions it can delay the movement of trains and cars. A malfunction causes additional work for the yard workers because they must find alternate ways to move cars so they can be connected to the right trains. Sensors can monitor these switches and create signals when anything is not functioning properly.

Example use case

In this example use case:

1. The sensor monitors various aspects of the switch.
2. When the sensor identifies a problem or the potential for a malfunction, it sends a message to the maintenance system.

5.7 Alerts to maintenance workers

The railway maintenance teams must be notified of any train or railway problems as soon as possible. This time-critical data allows the maintenance team to provide immediate service at the point of need and to ensure the proper person(s) is dispatched to resolve the problem.

5.7.1 Key requirements

The key business requirements are:

- Enable preventive maintenance on business-critical equipment.
- Reduce down time of equipment.
- Ensure the right (skilled) personnel are dispatched to perform the maintenance.
- Provide the maintenance personnel with specifics on the equipment malfunction to aid in diagnosis and resolution.

The key IT requirements are:

- Provide an extension to the existing maintenance system.
- Use existing communication channels or convenient channels to send alerts, preferably using Sametime.
- Incorporate generalized software to ensure flexibility in service provider selection.

5.7.2 Example application scenario

Steven, a railway maintenance man, must be immediately informed when a switching system is in the wrong position. The message must identify the location of the switch, so he can go to that specific location within the railway yard. The message is sent to Steven's device and displayed as a Sametime message.

Example use case

In this example use case:

1. Steven receives a Sametime message about an equipment failure.
2. He reads the message and immediately notifies the dispatcher that he is on his way.
3. He heads off to the location of the switch that is about to malfunction.

5.8 Automated on-train ticketing

There are various ways customers buy ITSO Railway tickets, such as using the online ticket purchasing system, buying them from the ticket agents at the train station, and buying them from the conductor on the train. The conductor needs a mobile solution that allows him to sell tickets and accept customer credit cards. The conductor's mobile devices may be occasionally connected to the ticketing system.

5.8.1 Key requirements

The key business requirements are:

- Increase customer satisfaction by using mobile devices to sell tickets and collect customer credit card information.
- Streamline on-train ticket selling and payment collection process.
- Validate customer credit card information against the credit card black list.

The key IT requirements are:

- Integration with and extend existing ticketing application.
- Make the application easy to use and learn.
- Use tools that support the end-to-end development process and simplify application development.

5.8.2 Example application scenario

Richard, a train conductor, walks through the train cars checking tickets and selling tickets to anyone who does not have a ticket. Richard uses a paper-based system to record the customer name, credit card information, and signature. With the new system Richard will use a PDA to select the route (which then determines the price of the ticket), collect the customer credit card information and the credit card PIN number, and validate the credit card information over

wireless connection. At the end of the transaction, the mobile device will generate a paper ticket for the customer.

Example use case

In this example use case:

1. Richard logs on to his PDA.
2. He accesses the ITSO Railways Ticket Application.
3. With the customer travel plans, he selects the departure and arrival station and number of tickets.
4. The application displays the ticket price. Richard enters the customer credit card information and submits the ticket transaction.
5. Occasionally the mobile device is synchronized with the server, so Richard's updates are processed by the Train Station Ticketing application.

5.9 Provide voice access to customers

The ITSO Railway has many calls to its timetable call center at peak periods from mobile phone users who want to check the availability of trains because they need to re-schedule their journeys to accommodate meeting changes, etc. Because of the large number of call center consultants needed at the peak times the call center costs have a significant impact even though they remain under-utilized at less busy times of the day. ITSO Railways is keen to reduce this cost but wants to maintain an easy-to-use alternative. In addition to automating the current timetable information access, they want to be able to extend it in the future using the same framework and technologies for booking and payment transactions.

5.9.1 Key requirements

The key business requirements include:

- Easy access to railway timetable information from phones
- Effective and efficient system for use by mobile phone users
- Technology solution to provide future extensibility to additional services including booking, payment, booking reviews, etc. using the same underlying technology infrastructure
- Technology solution to provide extensibility for common data and interface for use by customers on the Internet, mobile devices, and phones

The key IT requirements are:

- Framework and infrastructure to permit current requirements and future additions using a portal approach
- Tools for application development that are open source and standards based

5.9.2 Example application scenario

Marion has finished a meeting early and wants to know whether she should rush to catch an early train home if available or complete some notes over a coffee.

Example use case

In this example use case:

1. Marion uses her mobile phone to connect to the ITSO Railways information line.
2. Marion responds to the queries for departure and destination locations and obtains a readout of available times of trains for the afternoon.

5.10 Maintain the mobile devices

Many ITSO Railways mobile workers are away from the office for long periods of time. However, the applications and operation environments on these mobile devices must be maintained on a continual basis. ITSO Railways needs a way to roll out, update, maintain, and inventory software on the mobile devices while they are in the field.

5.10.1 Key requirements

The key business requirements are:

- Ensure mobile devices have the correct levels of software.
- Inventory software on the devices.
- Update and remove software on mobile devices as needed.

The key IT requirements are:

- Provide the administrator with tools to provision software to mobile devices.
- Provide the administrator with a tool to inventory mobile devices.
- Provide the administrator with tools to remotely update the device operating system.

5.10.2 Example application scenario

Steven, a system administrator, determines which software levels are to be provisioned to the mobile devices. He selects the users and groups to receive software updates and submits the jobs to provision the selected mobile devices. He can monitor the progress of these jobs. As needed, Steven requests an inventory of the devices to review the software levels on the devices.

Example use case

In this example use case:

1. Steven logs on to the mobile device management system.
2. He selects the software to be provisioned to the device.
3. He submits a job to perform the provisioning.
4. When a mobile worker connects to the enterprise, the device is checked to see if software updates are required. If updates are required, a job executes to send those updates to the device.
5. Once the updates are on the device, a device agent processes the software updates.

5.11 Secure mobile device

Many of the mobile devices used by ITSO Railways mobile workers have business-critical and business-sensitive data and applications on them. Therefore, ITSO Railways wants to ensure the security of these devices and the data and applications on the device, in case the device gets lost or stolen.

5.11.1 Key requirements

The key business requirements are:

- Ensure that business-critical and business-sensitive data is secure on the mobile devices.
- Notify the server if the login is unsuccessful after X tries.
- Wipe devices that have been lost or stolen.

The key IT requirements are:

- Provide the administrator with tools to wipe lost or stolen devices.
- Provide the administrator with tools to track down lost devices.
- Automate the clearing/wiping of the device.

5.11.2 Example application scenario

Steven, a system administrator, is notified that a device has been lost and the user does not think she will be able to find the device. Steven will issue a request to have the device locked and access to the enterprise network revoked. He will then issue the user a new device.

Example use case

In this example use case:

1. Steven accesses the administration system.
2. He determines the user and device in question.
3. He submits a job to lock and wipe the device as soon as it connects to the network.

6

Pervasive application types

The previous chapters discussed how patterns can help to define and clarify application and middleware components and functions. The goal of this chapter is to give a high-level overview of the different pervasive applications that can be implemented.

Our purpose with this book is to help simplify the implementation of such an environment. Patterns can help you to understand how any new mobile applications and connections will fit into and affect your existing environment.

6.1 Application types

What are the types of online and offline applications? What are the scenarios that demonstrate such pervasive application types? What are the common design patterns for offline and online implementations of the appropriate application types?

These are the questions we are looking to answer in this chapter.

6.1.1 Solution space

Figure 6-1 on page 103 shows the different application sets in the solution space for pervasive applications. A pervasive application to be implemented will most likely fall into one of the application sets:

- *Standalone applications* refers to the applications running on a device without the need to connect to a server application.
- PIM and e-mail applications - This is really a specific implementation of the general applications.
- General applications - This includes almost all of the pervasive solutions.
- Voice applications - These are specific pervasive applications where the user uses voice, natural speech to interact with the system.
- Multimodal applications - This is a combination of voice applications and general application solutions. The user can interact with the system using voice and can receive the response in a visual form (for example, Web page).
- Pervasive services - There could be additional components to a pervasive solution. These are also represented in the diagram:
  - Device management - This takes care of the various device and application management tasks on the pervasive clients.
  - Device connectivity - This provides connectivity services for the pervasive devices, including security and roaming.

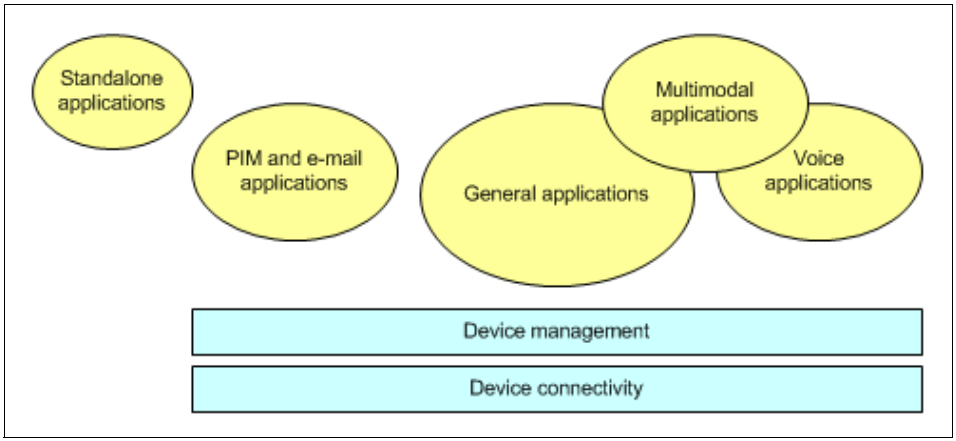


Figure 6-1 Application sets

We are going to use the application sets and services described above to provide more details about the available options to design a pervasive application. An application set can contain many application types. Each application set and its application types are discussed below.

The following is a list of the application types corresponding to the application sets:

- Standalone applications are not in the scope of this book.
- PIM and e-mail applications:
  - Online and offline PIM/e-mail
- General applications:
  - Online browser
  - Online browser (using stored pages)
  - Online and offline forms (using browser and a proxy)
  - Online and offline applications including transactions
  - Online instant messaging
  - Online notification (shared service)
  - Online location-aware application
- Voice application
  - Alternative user experience (voice)
- Multimodal application
  - Alternative user experience (multimodal)

Each application type is shown under the upcoming section, and they are mapped to Application and Runtime patterns. You will also find a list of valid devices for each application type.

Pervasive services are a different set of application types. These services relate to system management and they are always supporting other application types. The pervasive services types include:

- Device management
- Device connectivity

6.1.2 Application types mapped to Runtime patterns

The upper part of the following table represents the various devices that are capable of running the different variations of PIM and e-mail applications. At the bottom you can find the Application and Runtime patterns that you can apply to the solution.

Table 6-1 PIM and e-mail applications

Devices	**Online PIM and e-mail**	**Online and offline PIM and e-mail**
Windows client	T	R
Linux client	T	R
Pocket PC	T	R/N
Palm OS	T	R/N
Blackberry	T	R/N
Symbian	T	R/N
Smartphone	-	-
Patterns		
Application and Runtime patterns	Pervasive Device Adapter	Rich Device=Store and forward variation

Note: The T in the table means that the solution is supported by the thin client on the device listed on the left side (first column). The R means rich client support. The N means native application support. The different client application supports (thin, rich, native) can be combined.

The minus sign (-) indicates that the solution is not supported on the device.

The question marks (?) in Table 6-2 mean that there is no known solution for a certain device, but technically it is a possible solution. There could be third-party solutions.

The next application set is General applications. The next table includes several application types. The structure of the table is identical to the previous one.

Table 6-2 General application solutions

Devices	**Online browser**	**Offline browser (using saved pages)**	**Online + offline forms***	**Online + offline application including transactional**	**Online instant messaging**	**Online notification (shared service)**	**Online location-aware application**	**Alternative user experience (multi-modal)**
Windows client	T	T	T/R	R	R	R	R	R
Linux client	T	T	T/R	R	R	R	R	R
Pocket PC	T	T	T/R	R	R	R	R	R
Palm OS	T	T	T/R	R	R	R	R	R
Blackberry	T	?	T	?	?	?	?	?
Symbian	T	?	T	?	?	?	?	?
Smartphone	T	-	T	-	-	N	-	N
Patterns								

Devices	Online browser	Offline browser (using saved pages)	Online + offline forms*	Online + offline application including transactional	Online instant messaging	Online notification (shared service)	Online location-aware application	Alternative user experience (multi-modal)
Application and Runtime patterns	Pervasive Device Adapter			Rich Device =Store and forward variation	Rich Device=Online variation. For last column add Voice			

Note: The online + offline forms application type can be implemented using either a thin client solution or a rich client solution. With certain technologies you can implement the same thin client application (browser) in both online and offline environments.

Voice applications are also represented here in one column. It is a bit different from previous tables, although it is represented the same way. On the client side, voice applications really just require a phone for user interaction; therefore, listing a bunch of pervasive devices does not make much sense. The reason why these devices are still there is because some of them have either native phone connection built in, or they are capable of running applications with VoIP support to use voice interactions.

As shown in 3.2.9, “Composite Pervasive and Rich Device solution::Runtime pattern” on page 48, a composite pervasive network can support any combination of the Application and Runtime patterns shown in these tables.

Table 6-3 Voice applications

Devices	Alternative user experience (Voice)
Windows client	T/R (with VoIP support)
Linux client	T/R (with VoIP support)
Pocket PC	T/R (with VoIP support)
Palm OS	T/R (with VoIP support)

Devices	Alternative user experience (Voice)
Blackberry	?
Symbian	T/R (with VoIP support)
Smartphone	T/R/N
Patterns	
Application and Runtime patterns	Pervasive Device Adapter=Voice variation, Rich Device=online variation

The last table shows the Pervasive services solutions: Device management and Device connectivity. Read the table as described before.

Table 6-4 Pervasive services

Devices	Device management	Device connectivity
Windows client	R	R
Linux client	R	R
Pocket PC	R	R
Palm OS	R	R
Blackberry	-	-
Symbian	R	-
Smartphone	-	N
Patterns		
Application patterns	Rich Device=Online variation	Pervasive Device Adapter, Rich Device=Online variation
Runtime pattern	Rich Device=Online variation	Connectivity runtime pattern

Having reviewed the options we can now map the ITSO Railway scenarios to the appropriate application type, device type, and matching patterns.

6.1.3 Scenario implementations using various pervasive technologies

The following two tables are two parts of the same scenario - solution mapping table. Because of the extensive horizontal size of the table, it has been cut into two parts for improved readability.

The heading lists the different pervasive application types that we have defined in the previous section. The first column is a list of scenarios (use cases) we have chosen in this particular book. The table is a mapping between the scenarios and application types to show which approach could be used to implement each scenario. As you will see, there is more than one option in some cases. In these cases the business and IT requirements can decide which solution is the best choice for implementation. The options that we selected are shown by the chapter number in the associated cell of the table.

Table 6-5 Scenario implementation options - 1

Scenario	**Online + offline PIM/e-mail**	**Online browser**	**Offline browser (using stored pages)**	**Online + offline form (using browser + proxy)**	**Online + offline application including transactional**
Inventory management				T - Chapter 10 R - Chapter 11	R
Executive PIM and e-mail	T/R/N - Chapter 9	T			
Sensor notification	R/N (using e-mail)				R (using MQe)
Look-up train schedule		T - Chapter 10	T/N - Chapter 10		
Ticket purchasing	R/N	T	T/N	T/R/N	R - Chapter 13
Device management					

The following table continues the list of solutions.

Table 6-6 Scenario implementation options - 2

Scenario	Online instant messaging	Online notification (shared service)	Online location-aware application	Alternative user experience (voice)	Alternative user experience (multi-modal)	Online device management
Inventory management						
Executive PIM and e-mail						
Sensor notification	R (using IM)	R - Chapter 12				
Look-up train schedule			T/R/N	T/N - Chapter 14	R/N	
Ticket purchasing						
Device management						R - Chapter 16

Note: T means thin client solution. R means rich client solution. N means native client solution.

IM stands for Instant Messaging.

MQe stands for the IBM technology called MQ Everyplace.

Connectivity

You may have noticed that the connectivity scenario is not listed on any of the previous two tables. The reason is that connectivity can be applied to any of the scenarios. Connectivity is an infrastructural service that comes into consideration with networking, either online or offline operation. Remember that the offline applications have to go online sometime to perform synchronization or other data

exchange tasks. For more information about Device connectivity refer to Chapter 15, “Connectivity and access” on page 359.

Content adaptation

A corner-stone technology for pervasive solutions is transcoding (as defined before) or content adaptation (the new term). For more information about content adaptation refer to the IBM Redbooks *Mobile Applications with IBM WebSphere Everyplace Access Design and Development*, SG24-6259; and *Patterns: Pervasive Portals Patterns for e-business Series*, SG24-6876.

Solutions using the content adaptation technology can be found under the online browser application type and occasionally under the alternative user experience (voice) application type.

Part 2

Guidelines

7

Technology options

To mobilize your business, you must consider which technologies are used on the client side, on the server side, and which technologies can be used to connect and support the two.

These include options such as what mobile device types to use, which options there are to develop and host wireless Web applications for these devices, which technologies can be used to access these applications from the devices, and which services your pervasive environment will support.

As an emerging technology, pervasive computing is an area that is constantly evolving. It is important to consider your existing infrastructure as well as new technologies to make a best decision for your implementation.

7.1 Client-side technologies

Client-side technologies are those that can be used by your employees and customers to access your business's mobile applications.

7.1.1 Devices

There are many devices that can be used to access your business's mobile applications. It is important to consider which devices may be used to access your content to ensure that it is rendered correctly on the devices.

Cellular telephone

Cellular telephones are becoming more common as a way for customers to access information from your company. All major cellular telephone providers offer connectivity to the Web via cellular telephone.

Because they are in such widespread use, cellular telephones may offer a way to mobilize your business's workforce without the purchase of additional devices.

Access to the Web on a cellular telephone is provided via a microbrowser. A standard cellular phone includes voice capabilities, messaging features (SMS or WAP push), and data functionality (you can access Internet content through a microbrowser).

Laptop PC

With wireless Internet access becoming more common at home and on-the-go, the wireless laptop PC is considered a mobile device.

This category includes laptops, notebooks, or portable PC browser clients that have a wireline-connected or 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.

Personal Digital Assistant (PDA)

PDAs are handheld computers, usually with a wireless interface, that serve as an organizer for personal information. PDAs often have a pen-based stylus to tap selections on menus and to enter text. The PDA might also support handwriting as a means to interact with the device. The unit may also include a small on-screen keyboard that is tapped with the pen or with your thumbs.

The PDA offers the convenience of a small device with a comparatively larger screen than the cellular telephone.

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, Symbian OS (formerly EPOC), and Windows CE.

When developing mobile applications for certain industries, you should consider some industry-specific ruggedized devices. They have functionality that is similar to PDAs, but they are more robust and designed to work in hazardous environments. Symbol and Intermec are two manufacturers specializing in these devices.

Smartphone

This is a generic name for voice-centric mobile phones with information capability. Smartphones attempt to combine the functionality of the PDA and of the cellular telephone into one device.

Tablet PC

The tablet PC attempts to combine the handwriting capabilities of the PDA and the versatility of the Laptop PC into a large screen device than can be written on and read like a sheet of paper.

Telephone

With the increasing use of Interactive Voice Response (IVR) systems to handle business activities such as customer service, the telephone and the cellular telephone should each be considered as a pervasive device.

Sensors

Sensors can be connected to an enterprise network to collect data from the field. Depending on the type of the sensor data, a signal can be transmitted into the system for further processing.

Actuators

As opposed to sensors, actuators out on the field receive signals and data from a system and take actions based on the input. These devices can be integral parts of an enterprise system and can be controlled as pervasive devices.

7.1.2 Operating systems

These devices can run on a wide array of operating systems. However, in many cases, user interaction with a mobile application will be through a Web browser or via a Java-based application and will not be directly affected by the choice of operating system.

Linux

Linux is an open-source operating system based on UNIX. Versions used on mobile devices are smaller versions of the full Linux operating system. It is not currently in widespread use on pervasive devices but is experiencing growth in this area—especially on PDAs and on smartphones.

Palm OS

The 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 latest being Palm OS 5. It is important to note that some Palm 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 OS 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 used primarily in smartphones.

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

http://www.symbian.com

Windows Mobile for Pocket PC

The PocketPC operational system is based on the Microsoft Windows CE 3.0 (WinCE) operating system. It 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

Windows Mobile for SmartPhone

Like PocketPC, Microsoft has 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 on disconnected applications.

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

http://www.microsoft.com/mobile

Windows XP Tablet PC Edition

Windows XP Tablet PC Edition is a popular choice for tablet PCs. It provides the familiarity of the Microsoft Windows operating system with the ability to interact with the device with a stylus.

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

http://www.microsoft.com/windowsxp/tabletpc

7.1.3 Device Platforms/Frameworks

There are many platforms and frameworks on which mobile applications can be built and run.

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 a runtime environment to handle the user interface, security, and network protocols.

Prior to Java on the mobile device, applications for mobile devices were created in device-specific languages. These device languages provide performance advantages because they are tuned to the device; however, they had limited portability, and the application components had limited reusability across devices or applications.

J2ME allows the enterprise to create mobile applications that are easily migratable to multiple devices and types of devices. J2ME addresses portability issues by providing Java packages via configurations and profiles targeted at specific device configurations. J2ME consists of a customized subset of J2SE technology components (which will be discussed in “J2SE” on page 121), a set of standard hardware configurations, and profiles for targeting small devices.

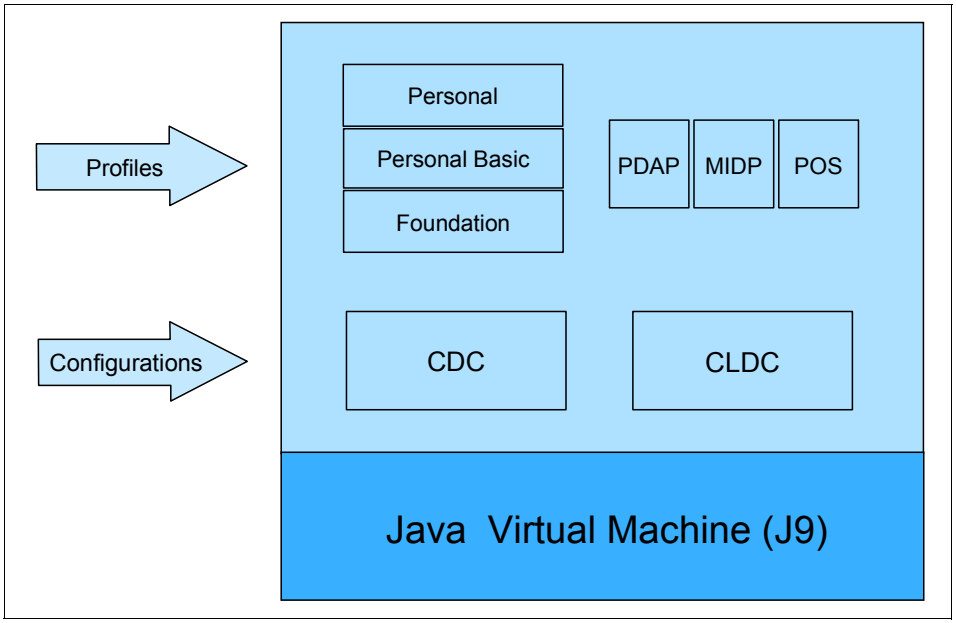


Figure 7-1 J2ME configurations and profiles

J2ME has two configurations:

- Connected Limited Device (CLDC), the smaller of the two J2ME configurations, is targeted for devices with intermittent network connectivity, slow processors (16 to 32 bit processors), and limited available memory (as low as 160 kilobytes).
- Connected Device (CDC), a full feature (J2ME) JVM, is much closer to the full J2SE functionality. CDC supports devices with a minimum of 2 megabytes of memory, including both RAM and flash or ROM available for the JVM and the Java applications, 32-bit processors, and greater network bandwidth. The CDC target devices include TV set-top boxes, residential gateways, in-vehicle telematics systems, and high-end mobile devices such as personal digital assistants (PDAs).

Any number of profiles can be made based on these two configurations. These profiles extend the configurations and define the class libraries available for building applications. In the above diagram, we can see the following profiles:

- Mobile Information Device (MIDP) profile for any limited capability device such as mobile phones and low-end PDAs
- Personal Digital Assistant (PDAP) profile for mid-range to upper-end PDAs
- Point of Sale (POS) profile for embedded point of sale devices

- Foundation profile for more powerful non-graphic devices
- Personal profile (extends the Foundation Profile) for more powerful devices with GUI support

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

http://java.sun.com/j2me

Microsoft's .NET Compact Framework

Microsoft also provides a framework to develop applications for mobile devices such as PDAs and smartphones. The .Net Compact Framework is a subset of the .NET Framework. It aims to simplify the process or create and deploy applications for mobile devices.

For more information on the .NET Compact Framework, see:

http://msdn.microsoft.com/mobility/prodtechinfo/devtools/netcf

QUALCOMM's 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.

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

http://brew.qualcomm.com

OSGi

The OSGi Alliance is an industry group that defines and promotes an open standard for the networked delivery of managed services to local networks and devices. Open standards enable different manufacturers to adhere to the same set of standards, which minimizes the number of products that are incompatible. The OSGi standard complements other residential standards and is open to most other protocols and transport and device layers.

The OSGi Service Platform Release 3 specification defines a Framework on which applications can run. Developers can write new applications and adapt existing applications to run on the Framework. The Framework enables operators to deploy multiple applications on a single Java Virtual Machine. Application developers partition applications into services and other resources. Services and resources are packaged into bundles, which are files that serve as

the delivery unit for applications. These bundles have manifests with special headers that enable you to share classes and services at the package level.

For more information on OSGi, refer to:

http://www.osgi.org

7.2 Server-side technologies

The services that the client devices connect to will reside on a server. Here we discuss the many options available on the server side.

7.2.1 Services

The following technologies are used to host and serve content to mobile devices and manage data exchange with mobile devices.

Application server

An application server allows devices to access and use the applications and other backend resources managed by the application server. In a mobile environment, an application server may host Web applications that allow devices to access information from servlets or portlets, or it might be used to provide mobile devices with access to services such as device management or synchronization.

Device management

Device management allows you to distribute software to mobile devices, update software on mobile devices, and collect inventory information such as application, hardware, and configuration information in a centralized manner.

Notification services

Notification services allows you to deliver notifications to users based on user preferences and subscriptions. These can be sent via a delivery channel such as SMS, e-mail, or instant messaging. Notifications can also be configured to alert certain groups of users to certain events. For example, server administrators can be alerted when a database server goes down, or financial brokers can be alerted when a stock falls below a certain price.

Synchronization

Synchronization allows the user to maintain accurate copies of data on both the server and on her device. For instance, synchronization of e-mail ensures that the most recent messages display on the device and synchronization of calendar

information ensures that a user's most recent appointments are viewable on both the device and the PC. Synchronization may also extend to replicating databases or processing message queues between the device and a server.

Transcoding/content adaptation

Transcoding (also referred to as content adaptation) allows standard HTML-based Web content to be transformed into a size and markup language appropriate for the type of mobile device trying to access this content.

Location Aware Services

Applications using the Location Aware Services are taking advantage of the location of the client device. Based on the location, applications have an additional input data to combine into the processing.

Voice services

Special "voice servers" provide voice services for the system, including:

- Automatic Speech Recognition (ASR)
- Text-to-Speech (TTS)
- Natural Language Understanding (NLU)

7.2.2 Java-based technologies

Because Java allows developers to write an application once and run it anywhere, Java-based technologies are very significant in the area of pervasive computing where there are many varied devices on which to run the same or similar applications.

J2SE

Java 2 Platform Standard Edition (J2SE) is an edition of Sun's Java platform designed to allow applications to be written once and run anywhere. It defines the Java Virtual Machine and core class libraries, and it provides the foundation for building applications, browser-based applets, and other Web applications.

For more information on J2SE and the following Java-based technologies, refer to Sun's Web site:

http://java.sun.com

J2EE

Java 2 Platform Enterprise Edition (J2EE) is a superset of J2SE that enables the development of robust and complex server-based enterprise applications. J2EE

provides such technology components as Enterprise JavaBeans (EJBs) and Web application services.

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.

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.

JavaServer Pages

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.

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.

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.

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 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

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.

Java Message Service (JMS)

Java Message Service is an API that allows developers to create Java-based applications that can exchange data with an enterprise messaging system in a standardized way.

JMS defines a set of common enterprise messaging concepts in order to maximize portability of the same code base among different enterprise messaging systems.

For more information on JMS, see Sun's Web site:

http://java.sun.com/jms

Portlets

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.

Portlets are 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 the IBM Redbook entitled *IBM WebSphere Portal V5: A Guide for Portlet Application Development*, SG24-6076.

7.3 The mobile Web

There are many technologies that relate to the area of the mobile Web. Some of these are markup languages. Others play a role in user interaction or controlling how data is exchanged with mobile devices.

7.3.1 HTML

HyperText Markup Language (HTML) 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/

7.3.2 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 maps, 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, and 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

7.3.3 XML

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 Extensible Stylesheet Language (XSL), 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.

7.3.4 XML Device-Independent Markup Extensions (XDIME)

XDIME is a markup language that attempts to enable developers to code an application once and run it on any device.

7.3.5 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. For more information on XForms, see:

http://www.w3.org/TR/xforms

7.3.6 XHTML 1.1 (HTML 4.01)

Extended HyperText Markup Language (XHTML) 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.

7.3.7 XSLT

Extensible Stylesheet Language Transformations (XSLT) is a W3C specification for transforming XML documents into other documents, including other XML documents, HTML documents, and WML documents. The XSLT is built on top of the Extensible Stylesheet Language (XSL), a stylesheet language for XML (such as CSS2 for HTML). 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, and it is expressed in the form of a stylesheet.

7.3.8 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`

7.3.9 SyncML DS and DM

SyncML stands for Synchronization Markup Language. It is a an open standard protocol that comes in two flavors: One for data synchronization and one for device management.

The goal of SyncML DS 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.

SyncML DM is based on the SyncML protocol and is designed to enable the customization, personalization, and servicing of mobile device, taking into account the same limitations as SyncML DS does for mobile environments.

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

http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html

7.3.10 VoiceXML and X+V

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.

7.4 Connectivity technologies

There are also many options to connect the client devices to their server counterparts. In a mobile environment, wireless technologies are the ones most often considered.

7.4.1 Wireless technologies

In a mobile environment, wireless connectivity technologies are the most likely to be used. When choosing which wireless connectivity technology, it is important to factor speed, cost, and coverage into your decision.

Cellular

Cellular networks provide the greatest areas of coverage. There are many different cellular technologies that allow Web access to wireless devices such as GPRS, GSM, CDMA, and UMTS. Because this area is always changing and evolving, it is hard to give exact specifications on cellular coverage currently offered by wireless providers. Current speeds are near that of a 56-k modem. In the near future, these speeds should approach near-broadband speed in metropolitan areas.

Wireless Ethernet

Wireless Ethernet (802.11x) is the most popular technology for providing wireless access in buildings, shops, homes, and workplaces. It is faster than cellular, but not as fast as wired Ethernet. It is reliable and can be easily added to an existing wired network infrastructure. Speeds range from 11mbps to 54 mbps but can degrade the farther a user gets from a wireless access point.

Wireless Ethernet standards such as 802.16 are currently under development for wide-scale implementations of wireless Ethernet in the form of a Metropolitan Area Network.

More information on wireless Ethernet can be found at:

http://www.ieee802.org

Bluetooth and Infrared (IR)

Bluetooth is another wireless standard used to allow devices within relatively close proximity to exchange data.

For more information, see the Bluetooth Web site:

http://www.bluetooth.org

Infrared uses a beam of infrared light to exchange data in the same way a remote control is used to change a television channel. It is slower than Bluetooth, and the devices must be positioned in direct line-of-sight from each other to communicate.

Other

Other means to connect wirelessly to the Internet include packet radio and satellite. These can be useful in areas that do not gave good cellular or other wireless coverage. These technologies can be expensive.

7.4.2 Wired technologies

Below is a list of wired technologies used in today's networks.

Plain Old Telephony Services (POTS)

Telephony services can be used to connect to IT networks, for example, modem connection from a desktop. Even though this is an old technology, it is still a valid and working method today.

Digital Subscriber Line (DSL), Cable

On digital networks data can be transmitted with a higher rate. DSL is a a successor of the analog phone network, and makes a digital connection between subscriber and provider. Cable networks are based on the digital Cable TV networks.

Cradle

Mobile devices such as PDAs come with cradles that connect to a PC to synchronize data. Often, this same connection can enable the device to access other network resources and the Internet. For situations where wireless connectivity is not an option, the cradle will provide an opportunity for the user to synchronize data with a network server or PC.

Ethernet

Some devices can be connected to a network via Ethernet in the same way that many PC workstations are. Ethernet provides relatively inexpensive connectivity, easy integration with an already existing wired network, and high speeds.

7.4.3 Issues with connectivity

When implementing a mobile environment, you must give due consideration for:

- Network security - Who is authorized to use your services, how are they authenticated, and if your Data needs to be encrypted.
- Connectivity management - Will users roam between different wireless and wired networks?
- Bandwidth - How much bandwidth will be used and how expensive will this be?

7.5 IBM-specific pevasive-related technologies

The following technologies are IBM proprietary technologies.

7.5.1 Service Management Framework (SMF)

Service Management Framework is an implementation of the OSGi Service Platform Release 3 specification described in “OSGi” on page 119. It is a production-ready software management framework for network-delivered applications.

The OSGi specification defines a set of services for application bundles to use. Of those services, Service Management Framework implements the following:

- Configuration Admin Service - Enables an operator to set the configuration information of deployed bundles
- Device Access - Supports automatic detection of attached and detached hardware devices and can automatically download and start appropriate device drivers
- HTTP Service - Provides an embedded HTTP server that is capable of serving HTML and servlets
- Log Service - Provides a general purpose message logger for the OSGi environment
- Package Admin Service - Enables a management bundle to provide the policies for package sharing

- Permission Admin Service - Enables a management bundle to administer a bundle's permissions and provides defaults for all bundles
- Preferences Service - Provides a persistent data store for bundles
- User Admin Service - Provides minimum authentication functionality
- Start Level Service - Enables a management agent to control the relative starting and stopping order of bundles
- URL Handlers Service Support - Enables you to dynamically register multiple URL and content handler services, which shield bundles from Java limitations

Service Management Framework Bundle Developer does not implement the following OSGi Release 3 specifications:

- IO Connector Service
- Wire Admin Service
- Namespace
- Jini
- UPnP
- Initial Provisioning

7.5.2 Workplace Client Technology, Micro Edition (WCTME)

Workplace Client Technology, Micro Edition (WCTME) provides developers with an integrated development environment based on WebSphere Studio and allows Java developers to leverage their existing skills to create mobile applications.

Workplace Client Technology, Micro Edition provides a Java runtime environment that offers the ability to extend new and existing Web applications to mobile devices. This platform provides a programming model, which incorporates the following technology and services:

- Approach based on Java 2 Enterprise Edition (J2EE) open standard programming model.
- Component-based architecture via Service Management Framework (SMF), IBM's implementation of the Open Services Gateway Initiative (OSGi) Service Platform Release 3 specification. The OSGi Alliance defines and promotes this open standard for network delivery of managed services to local networks and devices.
- IBM's J9 small footprint Java Virtual Machines (JVMs) and WebSphere Studio Micro Environment Foundation or WebSphere Studio Custom Environment Max and RM class libraries.
- Data storage capabilities by incorporating the IBM DB2 Everyplace and IBM Cloudscape small footprint local databases. Java Database Connection (JDBC) 2.0 Data Access APIs are used to access and manage the data within

these databases. Also included is DB2 Everyplace's bi-directional synchronization service, which synchronizes the data between the device and the enterprise data store.

- SyncML Data Synchronization (DS) protocol, an XML dialect, providing an alternative way to synchronize data between the mobile device and the enterprise server.
- Transactional messaging service via MQ Everyplace Version 2.0 and Java Messaging Service (JMS) for situations where the server application architecture expects all data updates and changes as transactions. MQ Everyplace provides a secure and dependable transport mechanism for once and only once (single) delivery of transactions to the enterprise.

7.5.3 Extension Services for WebSphere Everyplace (ESWE)

IBM's Extension Services for WebSphere Everyplace (ESWE) provides the Java runtime environment needed to create mobile applications and extend enterprise applications to mobile devices.

Extension Services for WebSphere Everyplace (ESWE), currently an IBM technology, extends the "write once, run anywhere" Java mantra from the server-to-mobile devices. ESWE has affinity to J2EE, which means that ESWE supports selected J2EE components and services. In particular, ESWE extends the J2EE Web application model, Servlet 2.3, and Java Server Pages 1.2 capabilities to mobile devices.

The ESWE platform provides a component-based Java runtime environment that incorporates Service Management Framework, and extends IBM's J2ME and WebSphere Custom Environment offerings. WCE is a complete Java runtime environment for embedded applications used in real-time control systems and on various devices in closed networks. WCE should be considered for applications where code size or runtime speed is more important than the cross-device compliance J2ME provides.

8

Application development toolkits

This chapter provides an overview of IBM's toolkits for developing pervasive solutions. The chapter shows how these toolkits integrate with WebSphere Studio's family of products and the specific tools that are provided by each toolkit.

This chapter includes:

- The pervasive tools strategy
- The toolkits and how they relate to WebSphere Studio
- The toolkit definition and an overview of the tools that are within each toolkit

8.1 Pervasive tool strategy

The pervasive tools are charged with accomplishing these three key goals:

- Extend IBM's application development framework to mobile workers by providing tools to enable developers to create mobile On Demand solutions.
- Make it easy to extend existing applications to the pervasive environment and to create new pervasive solutions.
- Provide tools that empower developers to deliver pervasive solutions.

The objectives for the tools are to:

- Support open standards based technologies.
- Support the complete application development life cycle.
- Increase the productivity of the development team building the pervasive solution(s) by providing Rapid Application Development (RAD) support, wizards, cheat sheets, and easy-to-use tools.
- Enable a wide variety of developers and business professionals with diverse skills to create mobile solutions.
- Provide state-of-the-art tools and toolkits that enable developers to build server-based, device-based, and end-to-end mobile applications.
- Support a same or similar programming model regardless of the target runtime environment (server-based, device-based, or multi-device mobile application models).

8.1.1 WebSphere Studio and pervasive toolkits

The WebSphere Studio product offering provides an integrated development environment (IDE) that supports WebSphere and J2EE application development, including Web services, Web applications, XML-based applications, and portlets. WebSphere Studio provides tools that support the complete development life cycle featuring an end-to-end test environment for applications that can be created using the IDE.

As shown in Figure 8-1 on page 137, the Eclipse-powered WebSphere Studio family provides the foundation for the pervasive toolkits. WebSphere Studio products are built on top of Eclipse, an open extensible universal tools platform, which is open source and managed by the eclipse.org. Eclipse enables third parties and open source developers to create additions to the framework and customize and integrate additional tools, utilities, APIs, and plug-ins to meet their needs.

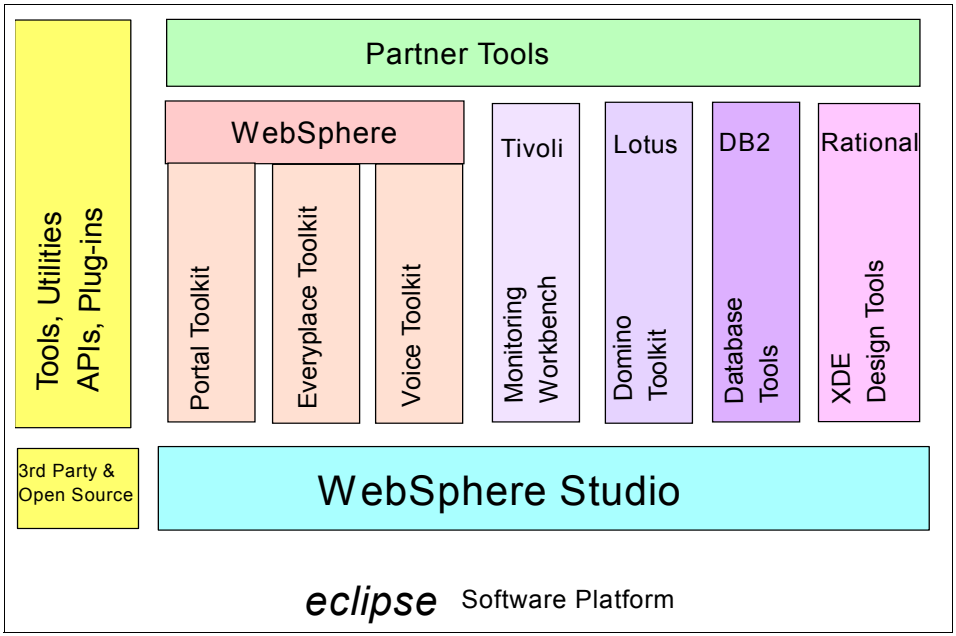


Figure 8-1 WebSphere Studio family and product-specific toolkits

Figure 8-1 shows the various WebSphere (pervasive) toolkits built on top of WebSphere Studio products. The pervasive toolkits are:

- *Portal Toolkit*, which provides an IDE for creating a portal and portlets. The IDE includes capabilities to create, test, debug, and deploy individual portlets and Web content.
- *Everyplace Toolkit*, which provides tools to enable developers to build a variety of mobile applications (both the server and device-based applications).
- *Multimodal Toolkit*, which provides tools to create XHTML combined with Voice XML (X+V) applications and speech-related tools.
- *Voice Toolkit*, which provides tools to create, debug, test and deploy voice (Voice XML) based applications.

Other IBM products, such as Tivoli, Lotus, DB2, and Rational provide development tools and toolkits based on the WebSphere Studio's offering, as shown in Figure 8-1.

8.2 Everyplace Toolkit

The Everyplace Toolkit provides development tools for developers creating a variety of pervasive applications. The Everyplace Toolkit plugs into either WebSphere Studio Site Developer or WebSphere Studio Application Developer.

The Everyplace Toolkit supports an end-to-end development experience. It consists of tools that support server application development, device applications development, and a variety of examples showcasing services within WebSphere Everyplace Access, WebSphere Everyplace Mobile Portal, and WebSphere Client Technology, Micro Environment. Figure 8-2 shows the various tools and examples provided in the Everyplace Toolkit.

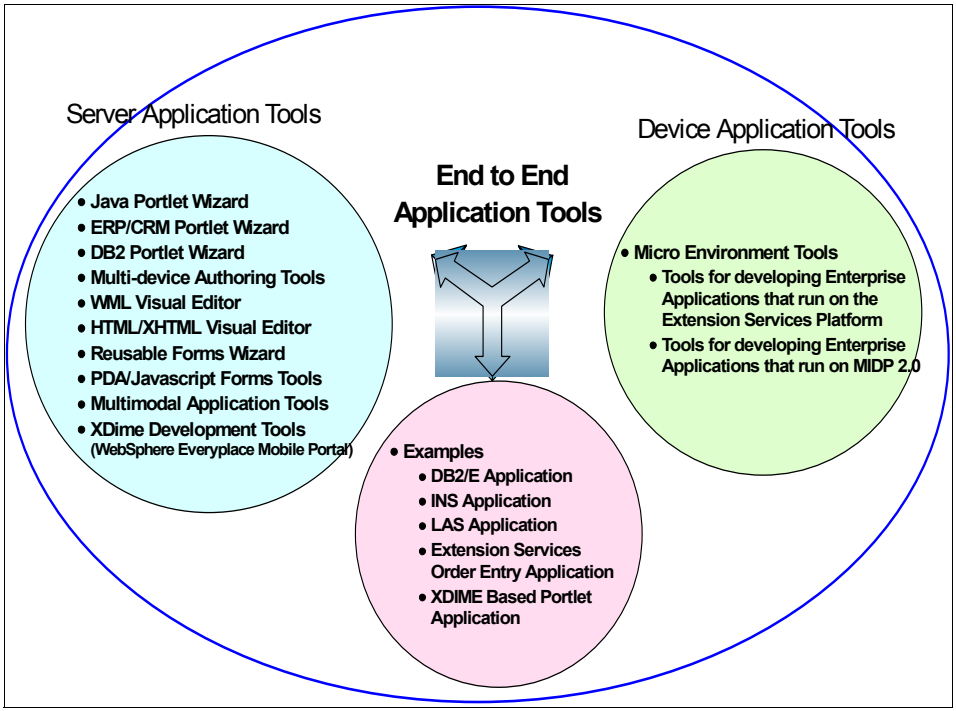


Figure 8-2 Everyplace Toolkit

The Everyplace Toolkit includes the Portal Toolkit, which provides tools to create, test, debug, and deploy portlets. The Everyplace Toolkit has tools to aid in server-based application development, such as:

- Java Portlet Wizard - A wizard that creates a mobile portlet based on the developer answers to a few questions. The portlet applications can be debugged using the WebSphere Studio Unit Test Environment. Completed

applications are easily deployed to production servers from WebSphere Studio.

- *ERP/CRM Portlet Wizard* - Creates mobile portlet applications that extend ERP or CRM application access from PDA class mobile devices.
- *DB2 Portlet Wizard* - Creates DB2-based mobile portlet applications that access DB2 tables and display information in the designated report format.
- *Multi-device Authoring Tool* - Visually develop portal and Web applications and adapt those applications for multiple mobile devices. It supports device-specific application flows, and custom device formatting or branding may be applied to the applications. It provides a very simple process for specifying new devices for support. The application is generated at design time for maximum runtime performance. It includes a rich set of device profiles allowing generation for all popular PDAs, cell phones, and other hand-held devices. The tool is shown in Figure 8-3 on page 140.
- *WML Visual Editor* - Supports WYSIWYG development of WML-based applications.
- *HTML/XHTML Visual Editor* - Supports WYSIWYG development of HTML and/or XHTML-based applications.
- *Reusable Forms Wizard* - A wizard used to quickly develop forms-based applications using the quick start or template-based approach.
- *PDA/Javascript Forms Tool* - A tool used to create forms in either PDA markup or Java script.
- *Multimodal Tools* - Tools to create XHTML and Voice XML (X+V)-based applications.
- *XDIME Development Tools* (for WebSphere Everyplace Mobile Portal) - Tools to create applications that can be dynamically tuned to the target device.

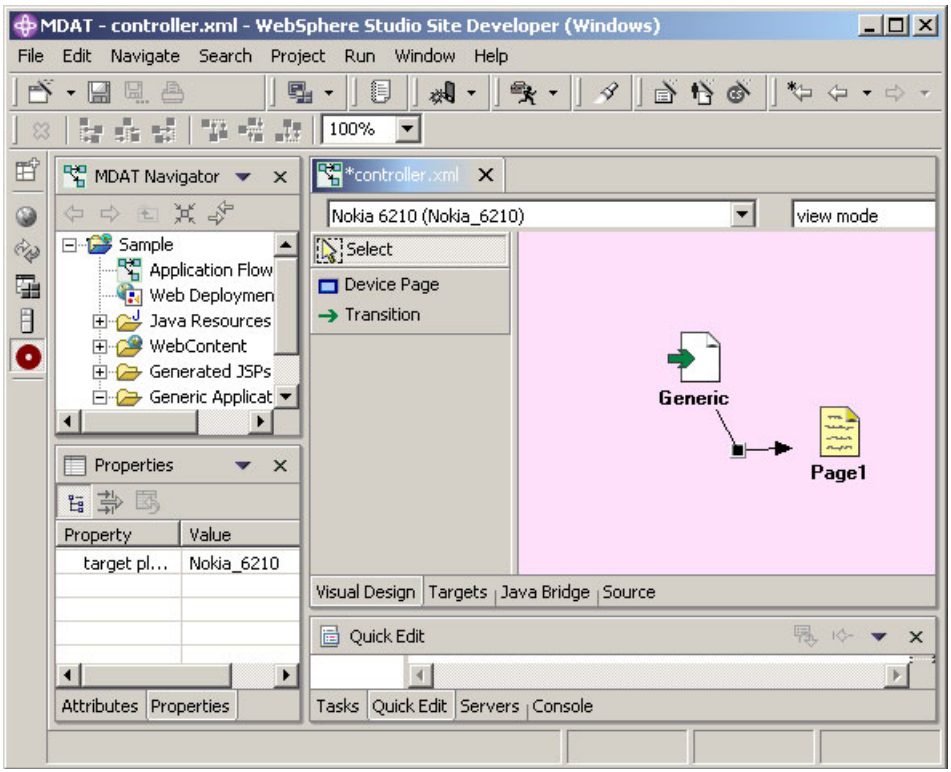


Figure 8-3 Multi Device Authoring Tool

The Everyplace Toolkit has tools to aid in device-based application development, that rely on the WebSphere Studio Device Developer and tools that support Workplace Client Technology, Micro Edition. These tools support developing applications that run on the Extension Services Platform or the MIDP 2.0 runtime environment.

The Everyplace Toolkit provides a variety of examples the developer can leverage in their application development. The examples support these services:

- DB2 Everyplace
- Intelligent Notification Service (INS)
- Location Aware Service (LAS)
- Extension Services Order Entry (a device-based example)
- XDIME based Portlet (for WebSphere Everyplace Mobile Portal)

For more details on the Everyplace Toolkit go to:

http://www-306.ibm.com/software/pervasive/everyplace_toolkit/

8.3 Multimodal Toolkit for WebSphere Studio

The Multimodal Toolkit for WebSphere Studio is an IDE that plugs into WebSphere Studio. It uses the XHTML + VoiceXML (X+V) proposed standard and Embedded ViaVoice speech recognition and speech synthesis to enable both visual and voice access to Web applications. It contains these tools:

- *Multimodal X+V Editor* - For creating and editing XHTML and VoiceXML within the same application.
- *Grammar Editor* - A text editor that provides syntax checking based on Speech Recognition Grammar Specification (SRGS) standards. It provides the ability to specify a user-provided Document Type Definition (DTD) used in content assist and validation and provides an 'Unknown Pronunciation' view to show those words that are not recognized within the grammar. Also, It provides the ability to generate SRGS grammars for VoiceXML applications, provides conversion capability from other grammars to SRGS and allows you to customize grammar compilation options. The graphical grammar test tools work with compiled grammar to provide debug assistance.
- *Pronunciation Builder* - A generator to create pronunciations from keyboard input, microphone input, or audio files builder. It includes a pop-up to assist in generating pronunciations based on phonemes defined by the International Phonetic Association. It offers multiple choices for default pronunciation generated using the recognition and Text to Speech (TTS) engines. It generates pronunciation files for recognition and TTS engines (exception dictionaries) and provides audio assistance to hear the generated pronunciation and tuning prior to application testing.
- *Audio recorder* - Allows for the creation of audio files from microphone input and provides a means to play a previously recorded audio file.
- *Multimodal Browser Launcher* - A tool that launches the browser used to test multimodal applications.
- *Application Deployment capabilities* - Extends WebSphere Studio's deployment tools to publish a multimodal application.
- *Reusable Dialog Components* - Pre-written X+V building blocks of code that provide common functions for use in application development.

For more details on the Multimodal go to:

http://www-128.ibm.com/developerworks/subscription/descfiles/mmtk4122.htm

8.4 Voice Toolkit for WebSphere Studio

The Voice Toolkit for WebSphere Studio is an IDE that plugs into WebSphere Studio. It contains these tools:

- *Call Flow Builder* - A graphical tool used to visually compose speech applications. It provides a drag-and-drop means of building the application from a palette of common speech dialog components. It includes support for call flow simulation to test the call design and anticipated dialogs. The tool facilitates the creation of a visual application flow, as well as the creation of associated dialog scripts, prompts, and audio files. It generates standards-based VoiceXML for deployment, and allows for text or recording as input to prompts. This tool is shown in Figure 8-4 on page 144.
- *CCXML Editor* - A text editor that provides syntax checking of call control. It provides content assist through pop-ups with valid CCXML elements and attributes. It assists with code development through color coding the CCXML elements. It supports code formatting and provides the ability to specify a user-provided DTD for use in content assist and validation.
- *VoiceXML Editor* - A text editor that provides syntax checking based on VoiceXML 2.0 standards. It provides content assist through pop-ups with valid VoiceXML elements and attributes. It provides a conversion capability from VoiceXML 1.0 to 2.0 and source code formatting with color coding of VoiceXML elements. It provides the ability to specify a user-provided DTD used in content assist and validation. It has an integrated VoiceXML simulator to test and debug your code. It can verify pronunciations for unknown words and creates custom pronunciations. It launches the grammar editor to define application grammars or the RDC Wizard to import reusable code or the audio recorder to record and play audio files.
- *Reusable Dialog Components (RDC)* - Pre-written VoiceXML building blocks of code that provide common functions for use in application development.
- *RDC Wizard* - A wizard allowing the user to select and customize Reusable Dialog Components (RDC). This fully integrates with the VoiceXML Editor.
- *Pronunciation Builder* - A generator to create pronunciations from keyboard input, microphone input, or audio files builder. It includes a pop-up to assist in generating pronunciations based on phonemes defined by the International Phonetic Association. It offers multiple choices for default pronunciation generated using the recognition and Text to Speech (TTS) engines. It generates pronunciation files for recognition and TTS engines (exception dictionaries), and provides audio assistance to hear the generated pronunciation and tuning prior to application testing.
- *Grammar Editor* - A text editor that provides syntax checking based on Speech Recognition Grammar Specification (SRGS) standards. It provides the ability to specify a user-provided Document Type Definition (DTD) used in

content assist and validation, and provides an ‘Unknown Pronunciation’ view to show those words that are not recognized with the grammar. Also, It provides the ability to generate SRGS grammars for VoiceXML applications, provides conversion capability from other grammars to SRGS and can allow you to customize grammar compilation options. The graphical grammar test tools work with compiled grammar to provide debug assistance.

- *Voice Portlet tools* - Adds a Voice portlet perspective that brings the portlet and the voice application creation together. It provides creation and validation of fragment VoiceXML portlet content. To aid in problem determination, log viewers are available to show what occurs when the voice portlet is running. Portlet wizard creates the framework for a voice portlet and provides local test and debug support that mirrors the WebSphere Voice Application Access environment.
- *Audio recorder* - Allows for the creation of audio files from microphone input and provides a means to play a previously recorded audio file.
- *Analysis Tools* - Provides the ability to examine recognition log files for call analysis and the ability to verify audio quality of audio files.
- *National Language Understanding Tools* - Provides the ability to train an NLU application to understand free-form user interactions, using an integrated, database-driven environment. It includes a testing environment for trying out the NLU statistical models using a keyboard or microphone.
- *Lexicon Editor* - A development tool for Lexicon markup, for purposes of creation and modification of pronunciation files. This editor provides a set of functions similar to the VoiceXML editor except that it is based on proposed Lexicon standards.

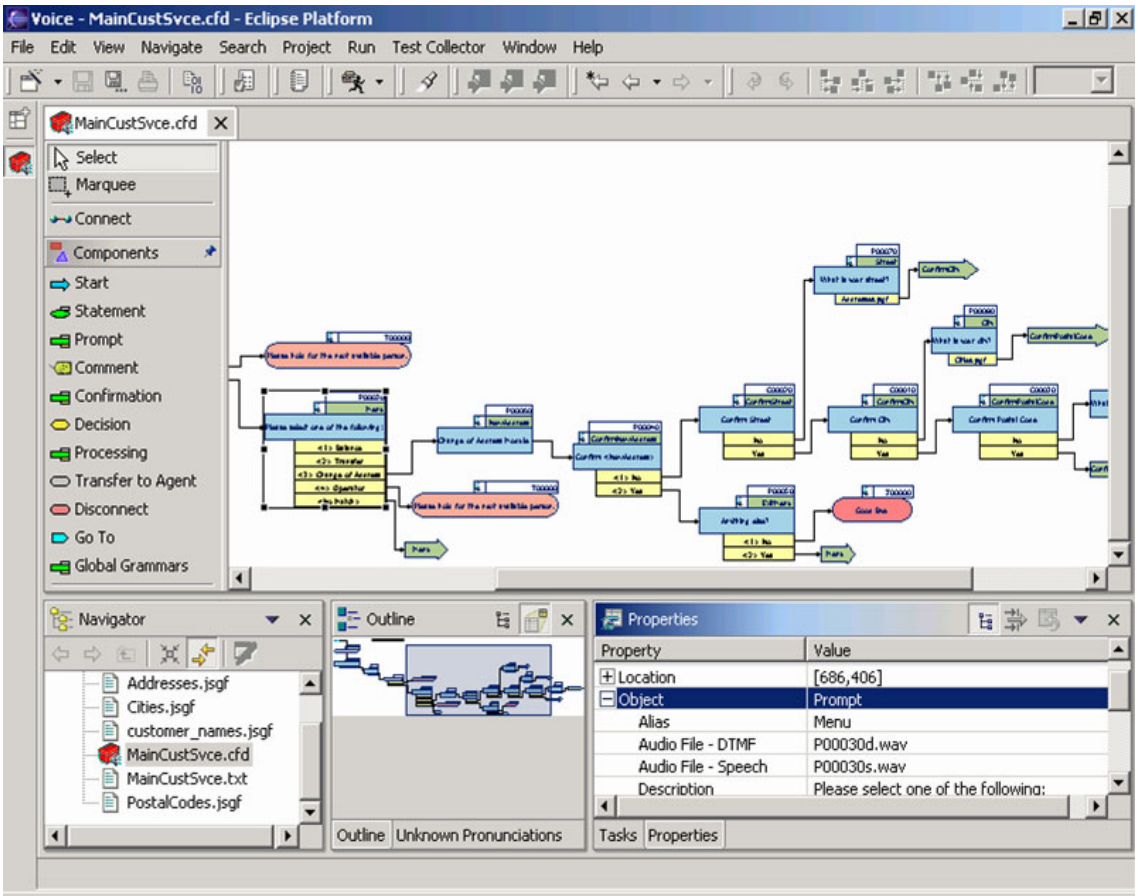


Figure 8-4 Call Flow Builder

For more details on the Voice Toolkit for WebSphere Studio go to:

http://www-128.ibm.com/developerworks/subscription/descfiles/vtws51wx.htm

8.5 WebSphere Studio Device Developer

WebSphere Studio Device Developer, an integrated development environment, is used to develop, test, debug, and deploy standalone Java applications that run on pervasive devices. For creating embedded applications WebSphere Studio Device Developer can be used alone or as a plug-in to either WebSphere Studio Site Developer or WebSphere Studio Application Developer. In conjunction with

WebSphere Studio, WebSphere Studio Device Developer can be used to create Web applications targeting pervasive devices.

WebSphere Studio Device Developer includes various tools:

- *MicroAnalyzer* - Gathers dynamic application runtime information for analysis and review, which helps pinpoint problems in the code and areas in the code where optimization is needed.
- *SmartLinker* - Helps to tune the code and removes unnecessary objecting, making the application as small as possible before it is deployed to devices.
- Test support for multiple devices and emulators.
- *Just-in-time and ahead-of-time compilation* - Technology that speeds up applications in the runtime environment.
- *Web Services Toolkit for Mobile Devices* - Both development tools for creating Web services and a Web Services runtime simulation environment. This toolkit supports development of bundle applications and services that consume and publish Web Services.

8.5.1 SMF Bundle Development Kit

SMF Bundle Development Kit Version 5.7 is used to develop, test, and deploy OSGi bundles. This toolkit provides a runtime and tools to develop applications and services that run on IBM's implementation of the OSGi framework, Service Management Framework (SMF). The SMF Bundle Development Kit adds the SMF perspective to WebSphere Studio. The SMF perspective contains wizards, views, and editors that collectively provide an SMF bundle developer with the tools needed to perform the essential tasks, such as:

- Identifying a bundle package and service imports and exports
- Constructing the OSGi manifest to include package and service imports and exports
- Tagging the bundle with device characteristics that enable the SMF Bundle Server to differentiate target devices
- Submitting bundles to the SMF Bundle Server for testing
- Connecting to any number of SMF Bundle Servers to view the contents of the server's repository
- Launching an SMF runtime from the IDE
- Launching an SMF Bundle Server from the IDE
- Connecting and managing an existing SMF runtime located anywhere on the network

8.5.2 Application Tools for Extension Services

The Application Tools for Extension Services V5.7 extends existing WebSphere Studio tools enabling you to develop, test, and deploy Web applications targeting pervasive devices. Figure 8-5 shows the Extension Services Web application development with WebSphere Studio, and Figure 8-6 on page 147 shows Extension Services test environment.

The Platform Builder tool allows you to create multiple platform projects that target different device configurations or contexts.

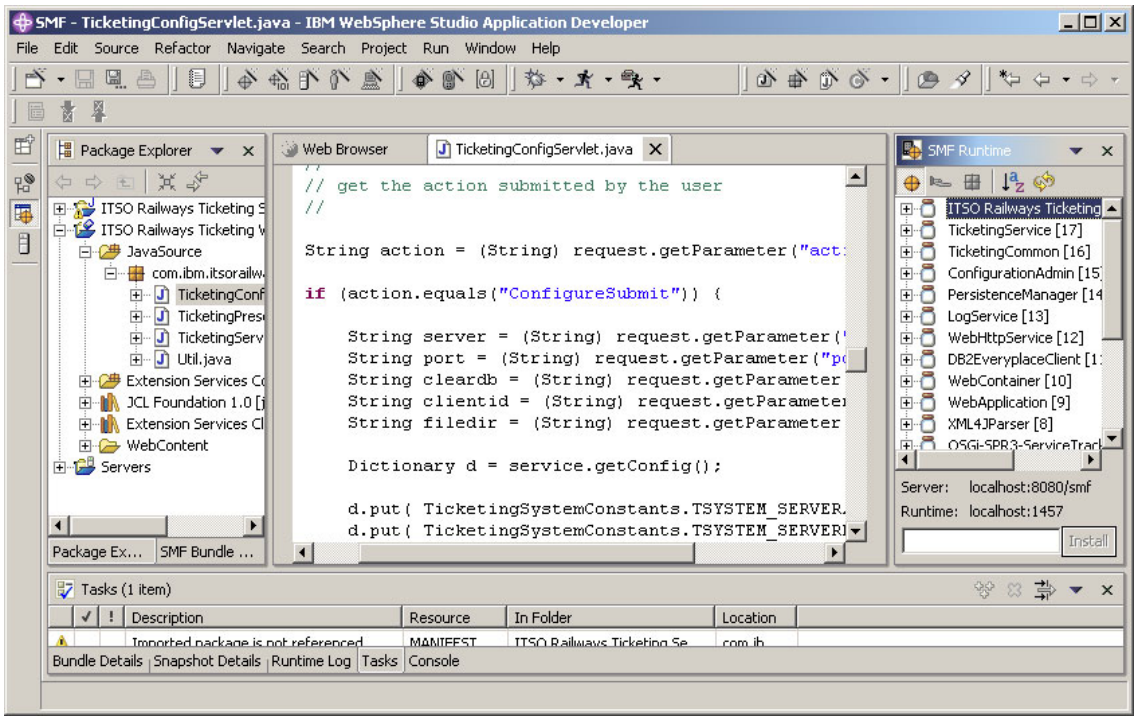


Figure 8-5 Extension Services Web application development

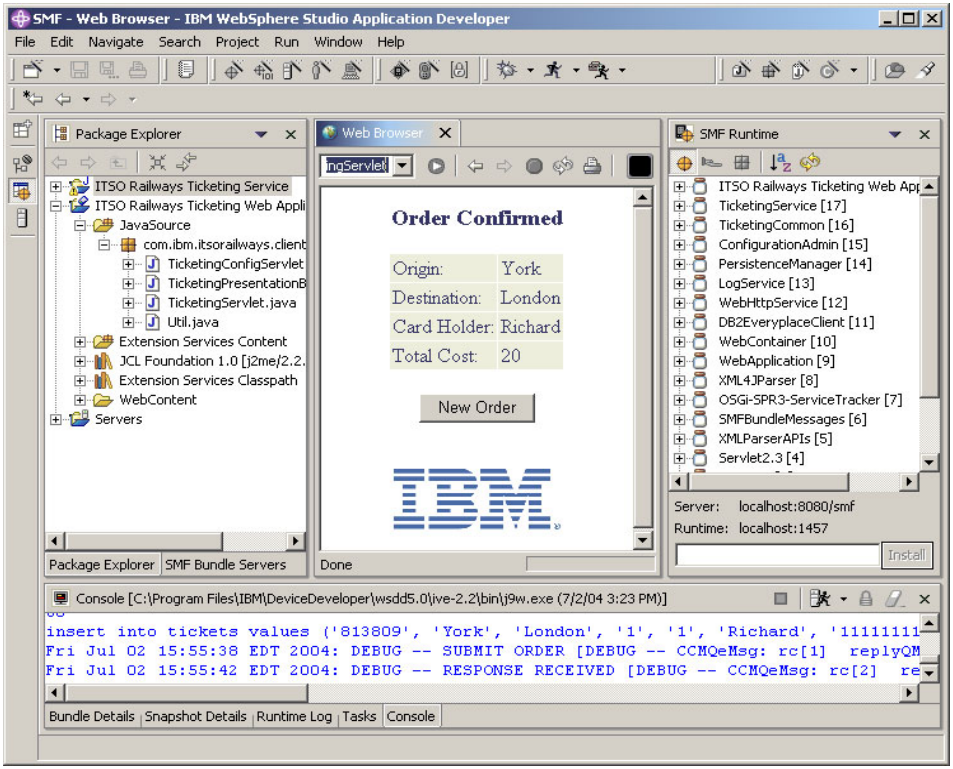


Figure 8-6 Extension Services testing

For more details on the WebSphere Studio Device Developer, the SMF Bundle Development Kit, and Application Tools for Extension Services go to:

http://www.ibm.com/software/pervasive/products/wsdd/index.shtml

Part 3

Scenario implementations

9

PIM and e-mail synchronization

The executive PIM and e-mail support scenario describes the architectural approach to synchronize Personal Information Management (PIM) and e-mail data from servers to mobile devices. Starting from the business case of ITSO Railways, as described in Chapter 5, “ITSO Railway sample overview” on page 87, example use cases are created. Based on these examples, use cases and the Pervasive runtime patterns from Chapter 3, “Runtime pattern” on page 35, an architectural overview diagram and a component model are developed. Furthermore, necessary considerations, administration steps, and the final setup of the PIM and e-mail synchronization are documented.

9.1 Overview

This chapter is a summary of requirements that are needed to support PIM and e-mail synchronization on mobile devices in the ITSO Railway's scenario. It is an implementation based on the following Runtime pattern.

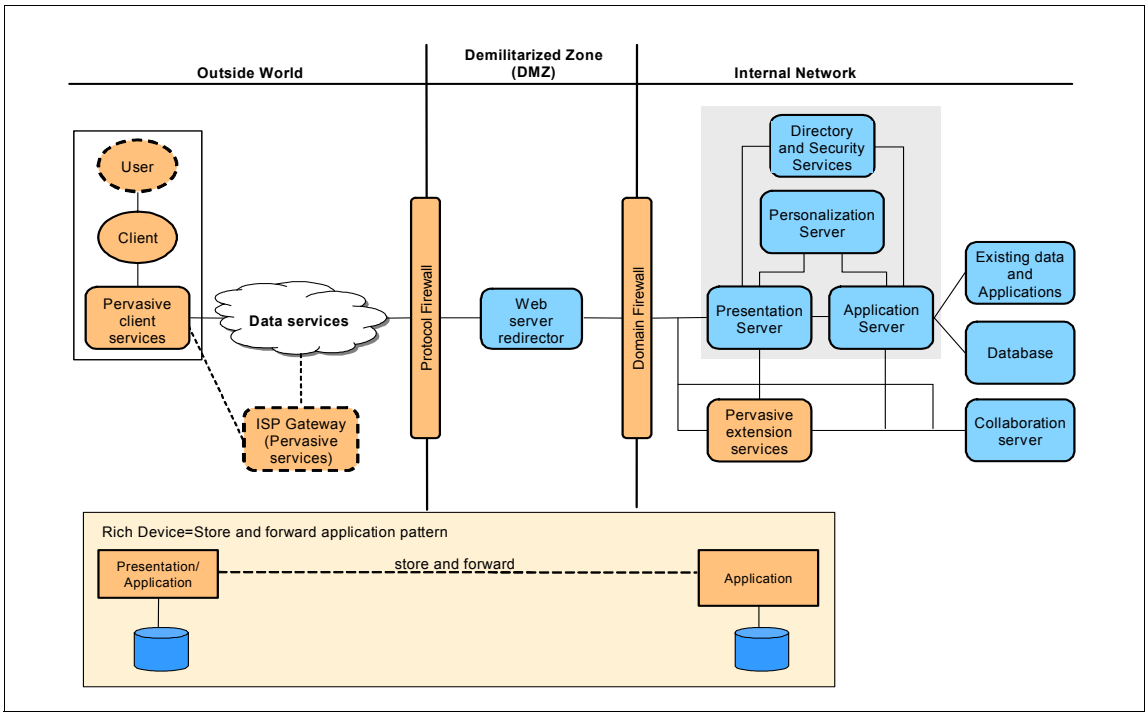


Figure 9-1 Runtime pattern for the PIM and e-mail scneario

The Product mapping for the Runtime pattern we use for this book is found in the following diagram.

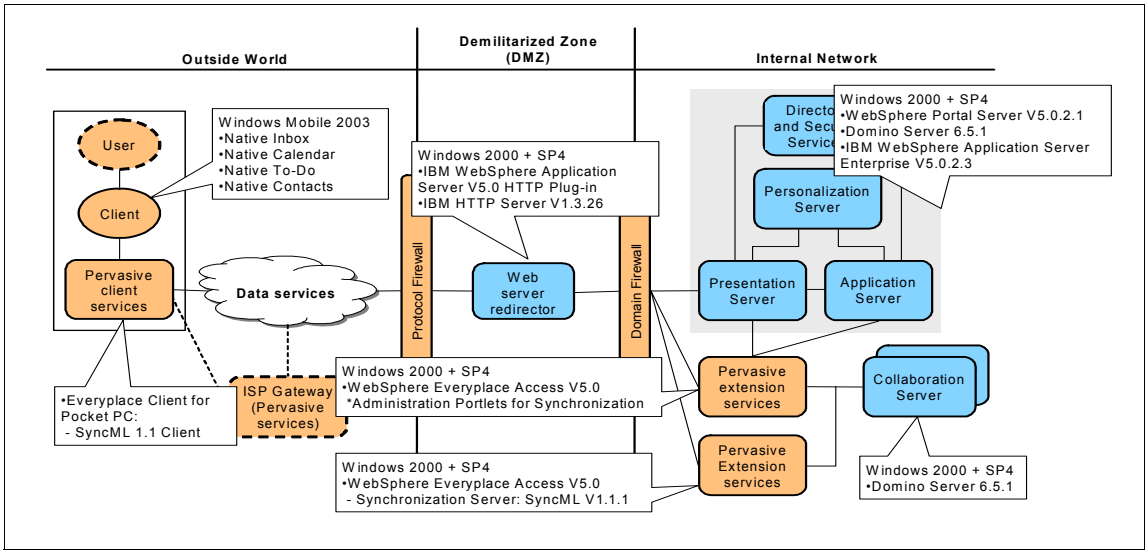


Figure 9-2 Rich Device=Store and forward::Runtime mapping=PIM and e-mail, Windows

9.1.1 Customer requirements

The ITSO Railway company example was created to explain general approaches for mobilizing applications. Examples for customer requirements are described and listed in Chapter 5, “ITSO Railway sample overview” on page 87.

The following business context was identified for the PIM and e-mail synchronization access:

- Currently, ITSO Railway executives need to access their PIM and e-mail data while traveling. They must be able to maintain a single copy of their e-mail and be assured that the PIM updates will be posted to their office PIM application server.
- The current system should extend the access of mobile devices such as PDAs or cell phones without interference with existing PIM and e-mail services.
- The following chapters list the captured functional and non-functional requirements to define the baseline according to which the business system must be designed. A sample use case model with the most important use cases is also created.

The next figure represents the business functions and actors that are related to PIM and e-mail synchronization process. The picture also defines the connectors that are used to link the individual symbols together. The picture provides a

comprehensive way of representing key aspects of the proposed solution and gives a foundation for identifying and applying the components that are required to implement the ITSO Railway PIM and e-mail support to executives.

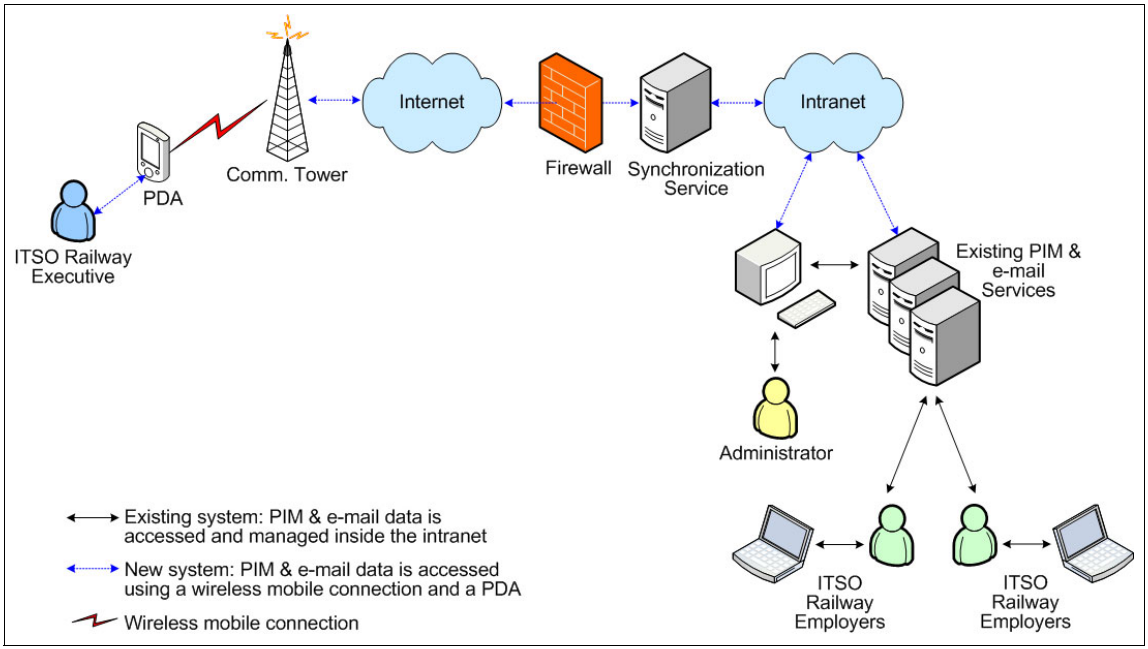


Figure 9-3 ITSO Railway PIM and e-mail example business context: Business, users, and connectors

9.1.2 Functional requirements and use case model

The functional requirements of the ITSO Railway PIM and e-mail support example are extracted from the customer requirements. They provide the main input for the use case model. The use case model work product is used to describe the functional requirements of the system under development. The model uses graphical symbols and text to specify how users in specific roles will use the system (that is, use cases). The textual descriptions describing the use cases are from a user's point of view; they do not describe how the system works internally or its internal structure or mechanisms.

Actor names and descriptions; use case numbers, names, business events, and overviews; and communication associations between the actors and the use cases provide an overview of the functional requirements. The other constructs of the model document the expected usage, user interactions, and behaviors of the system in different styles and depth.

We simplify the use case model for the ITSO Railway PIM and e-mail support example and reduce the use case description to the following constructs:

- Actors (name, description, status, superclass, subclass, and associations)
- Use cases (number, subject area, business event, name, overview, preconditions, description, associations, inputs, outputs, traceable to, usability index, and notes)
- Communication Association diagram

The presented project approach for the ITSO Railway PIM and e-mail support example is mainly based on the IBM Global Services Method recommendation.

Actors

The following actors are interested in using this new mobile-enabled PIM and e-mail support:

- Executive
- Administrator
- Internal employee

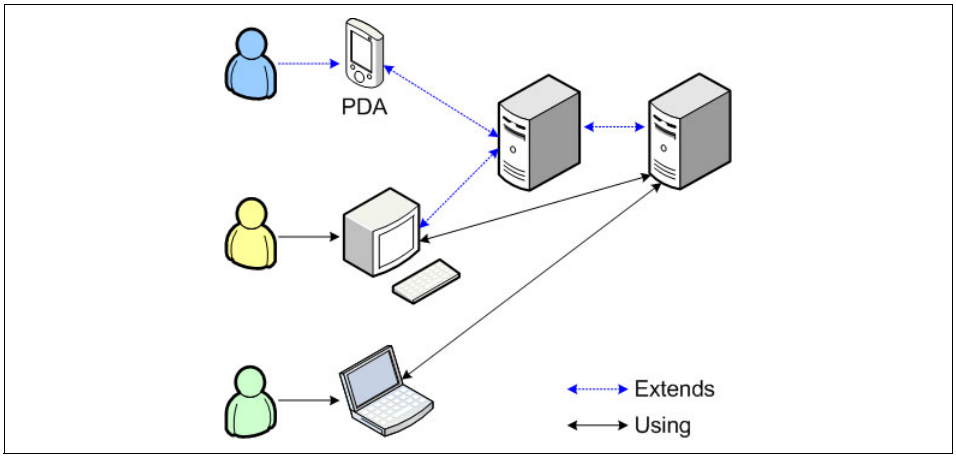


Figure 9-4 ITSO Railway PIM and e-mail support - Use case actors

The executive is the main actor in this example. He must have the ability to access PIM and e-mail information from a PDA. He also must be able to maintain a single copy of his e-mail and PIM data and synchronize that data with his office PIM application server.

The Administrator is a general system administrator who is responsible for the deployment, maintenance, and management of the synchronization and existing PIM and e-mail services. One of his responsibilities is that he also maintains the

user and device accounts that are required for the synchronization and existing mail services.

The internal employee is a actor who can use the existing mail services inside the ITSO Railway network. This actor and its services are not directly related to the mobile device services. The only requirement is that these actors must have continuous access to their e-mail databases even without a mobile connection. The actor could be any ITSO Railway employee, even the executive when he works in his office.

The following tables describe the actors of the ITSO Railway PIM and e-mail support example (see Table 9-1 and Table 9-2 and Table 9-3 on page 157).

Table 9-1 Use case actor executive

Actors name	**Executive**
Brief description	The ITSO Railway executive who needs to have access to his PIM and e-mail data whenever and wherever he is through the PDA. The data must be synchronized towards the existing office PIM and e-mail application.
Status	Primary.
Relationship	
Inheritance	Subclass: Administrator, internal employer.
	Superclass: None.
Association to use cases	Start application, create a new memo and/or entry and/or item, submit memo and/or entry and/or item, synchronize inbox and/or calendar and/or to-do.

Table 9-2 Use case actor administrator

Actors name	**Administrator**
Brief description	The administrator for the synchronization of PIM and e-mail services. Maintains the system, deploys, and makes it available for the PDA devices and users.
Status	Secondary.
Relationship	
Inheritance	Subclass: None.
	Superclass: Executive and internal employee.

Association to use cases	Maintain accesses and services.

Table 9-3 Use case actor internal ITSO employer

Actors name	**Internal ITSO employee**
Brief description	The internal employee can use his PIM and e-mail applications online through the corporate network.
Status	Secondary.
Relationship	
Inheritance	Subclass: Administrator, executive.
	Superclass: None.
Association to use cases	Use PIM and e-mail services online inside the corporate network (same as executive, but online).

Use cases

Use cases describe functional requirements. They are used as input for the system and application design. Furthermore, use cases show the key functionality for the solution delivered to the customer. The final solution test runs utilize the use cases that the customer and the system provider agreed to at project start.

The following use cases were considered in the ITSO Railway PIM and e-mail synchronization support:

1. Start Application.
2. Create and submit new memo and/or entry and/or item.
3. Synchronize inbox and/or calendar and/or to-do.
4. Maintain accesses and services.

The following tables describe these use cases (see IBM Global Services Method).

Table 9-4 Use case 1: Start application

Use case #1 **Subject area**	**Start application** **Data input**
Business event	Executive starts his working day, logs into the PDA, and starts to read his memos, entrie,s and tasks.
Actors	Executive.
Use case overview	Executive starts the native inbox, calendar, and to-do's applications on the PDA, which is used to keep the executives up-to-date. Executive gets requests from his superior to create reports for them.
Preconditions	- PDA is fully charged, - Authentication was successful, - E-mail and PIM data was synchronized,
Termination outcome	**Condition effecting termination outcome**
1) PIM and e-mail data loaded properly	- PDA fully charged. - Authentication successful. - Mobile connection was enabled. - PIM and e-mail data was synchronized successfully.
2) PIM and e-mail data not loaded properly	- PDA cannot be switched on. - Authentication was not successful. - PIM and e-mail data was not synchronized.
Use case description	ITSO Railway executive starts his work day and logs into the PDA, which is fully charged and synchronized during the night. Executive switches on the PDA and authenticates himself. He opens the e-mail and PIM applications and reviews the existing items and tasks for today.
Use case association	- Used by create new memo and/or entry and/or item use case. - Needs the Submit memo and/or entry and/or item and Sync inbox and/or calendar and/or to-do use case.
Inputs summary	- Authentication data.
Output summary	
Usability index	
Use case notes	Use case provides the security features for this application and it is tightly related for the successful synchronization.

Table 9-5 Use case 2: Create and submit memo and/or entry and/or item

Use case #2 **Subject area**	**Submit memo and/or entry and/or item** **Data input**
Business event	Executive updates the status and adds an item.
Actors	Executive.
Use case overview	Executive creates the reports based on his superior's requests and tracks these requests using to-do lists. After he has creates those requests he can submit and store the data.
Preconditions	Start PIM and e-mail applications on PDA.
Termination outcome	**Condition effecting termination outcome**
1) Validation successful	- Data were entered in the correct format. The validation function returns successful when the SUBMIT button is pressed. - The submit button is pressed and the data is locally stored for further synchronization event purposes.
2) Validation unsuccessful	- Data were entered in the wrong format. The validation function returns a failure when the button SUBMIT is pressed. - Data cannot be stored.
Use case description	Executive creates a report based on his superior's requests using the device's native e-mail application. To keep track of these requests, he uses the to-do list PIM application. If there is any reason to create a new calendar entry to arrange a meeting, he could create that as well. When the executive has finalized responses, updated his to-do lists, and created calendar entries, he could submit those forms. Forms and data are stored in a local database until the next synchronization.
Use case association	- Used by Create and Submit new memo and/or entry and/or item use case. - Needs the Start Application use case.
Inputs summary	Memo will contain the following fields: From, To, Subject, Body. To-do will contain the following fields: Subject, When, Priority, Status, Description, Category. Calendar entry will contain the following fields: Type, Subject, Chair, When to start, When to end, Where, Invites, Category, Description.
Output summary	-Responses to executive's superiors. -New calendar entries. -Updated to-do list.

Usability index	This is the most important use case for the end user (executive) point of view.
Use case notes	Use case is a base for the synchronize inbox, calendar, and to-do use case.

Table 9-6 Use case 3: Synchronize inbox, calendar, and to-do

Use case 3# **Subject area**	**Synchronize inbox, calendar, and to-do** **Data input**
Business event	Executive sends out and synchronizes his PDA with the office PIM and e-mail applications.
Actors	Executive.
Use case overview	Executive starts the synchronization client and sends out the stored responses, calendar events, and to-do list updates.
Preconditions	Authentication was successful.
Termination outcome	**Condition effecting termination outcome**
1) PIM and e-mail synchronized	- Authentication was successful. - Connection is established. - The data was exchanged between the client and server. - PIM and e-mail data were successfully synchronized.
2) PIM and e-mail not synchronized	- Authentication was not successful. - Connection is not available or connection is unstable. - Data was not entered on forms successfully (that is, used some special characters). - Server could not synchronize the data in a reasonable amount of time.
Use case description	Executive synchronizes his PDA at certain times during his working day to sent and to receive the latest responses and updates.
Use case association	- Used by Start Application use case. - Needs the Create and Submit new memo and/or entry and/or item use case for synchronization to make sense.
Inputs summary	Synchronize the stored local data and receive the latest updates from the office PIM and e-mail application.
Output summary	Send out the calendar entries and responses to his superiors.

Usability Index	Without this use case the executive could not update and maintain his personal PIM and e-mail data as up to date on the PDA.
Use case notes	Use case relies on a robust, scalable middle ware for PIM and e-mail synchronization.

Table 9-7 Use case 4: Maintain accesses and PIM and e-mail services

Use case 4# **Subject area**	**Maintain accesses and PIM and e-mail services** **System maintenance, user administration**
Business event	It is possible to serve PIM and e-mail accesses for the PDA devices. Executives are able to keep their office PIM and e-mail content up to date whenever and wherever they are.
Actors	Administrator.
Use case overview	Administrator deploys mobile application for ITSO Railway executives and makes available for them to access their personal PIM and e-mail content.
Preconditions	-Mobile connections are available. -Devices are customized and able to serve PIM and e-mail synchronization. -PDAs, PIM, and e-mail users and their access rights are defined.
Termination outcome	**Condition effecting termination outcome**
1) PIM and e-mail synchronization successfully deployed and user assigned	- Client side contains the synchronization services. - Authentication successful. - Application was installed successfully. - User access rights are assigned to executive members.
2) PIM and e-mail synchronization not deployed and is not available to users.	- Authentication was not successful. - Application could not be installed. - Device does not support such synchronization services. - User access rights have not been assigned to delivery staff members.
Use case description	Administrator logs on to administration server, configures the synchronization services, and customizes the users profiles. Administrator assigns user access rights to be able to use PIM and e-mail synchronization services.
Use case association	This use case is a precondition for all other use cases.

Inputs summary	Existing PIM and e-mail services must be in place. Synchronization servers must be able to connect to the existing PIM and e-mail services.
Output summary	
Usability index	PIM and e-mail synchronization service will not be accessible for ITSO Railway executives if this use case is not successful.
Use case notes	

Use case communication-association diagram

Use cases also describe the communication style. This communication is modeled as a communication-association, as shown in Figure 9-5 on page 163. The direction of the arrow shows the direction in which the communication is initiated. In addition, Figure 9-5 on page 163 includes the application workflow for the executive as well as the relationship between the use cases.

For example, the executive starts the native PDA PIM and e-mail applications to read his e-mails and reply to those e-mails. The executive also maintains his personal to-do lists in order to keep track of those entries. During the day after certain periods of time, he synchronizes the locally stored data towards the office PIM, e-mail applications, and databases.

The administrator's workflow and relationship between use cases is also described in Figure 9-5 on page 163. The administrator starts the user access management application and gives the proper rights for the executives. Without those rights, the executives cannot use the PIM and e-mail synchronization services.

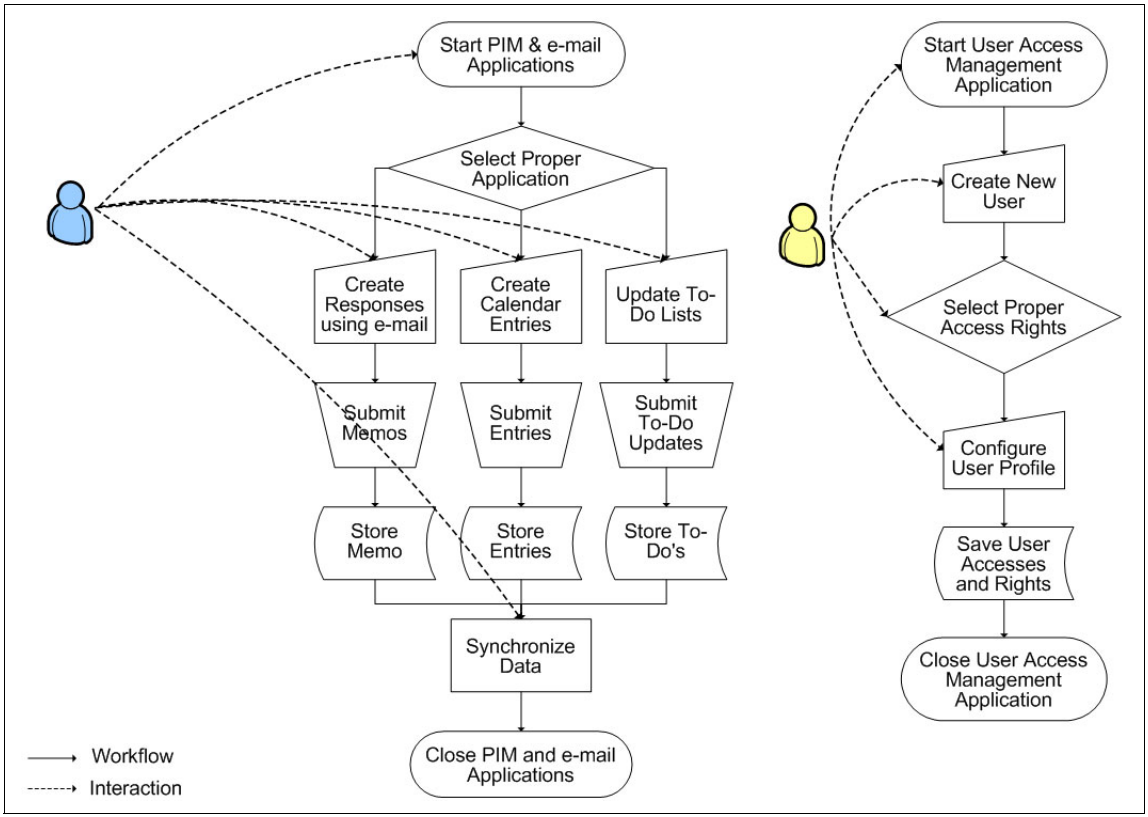


Figure 9-5 Use case communication-association diagram for PIM and e-mail support

9.1.3 Non-functional requirements

This section lists non-functional requirements that apply mostly to applications on mobile devices.

The non-functional requirements for a business system address those aspects of the system that do not directly affect the functionality of the system as seen by the users. Nevertheless, they can have a profound effect on how that business system is accepted by both the users and the people responsible for supporting that system.

The non-functional aspects of a business system cover a broad range of themes. The major non-functional themes identified by the AED Technical Council are listed below:

- Performance
- Scalability

- Availability (including recoverability ability and reliability)
- Maintainability (including Flexibility and portability)
- Security
- Manageability
- Environmental (including safety)
- System usability
- Data integrity (including currency, locality of updating, and data retention)

Generally, all the requirements of the system to be delivered must be understood in each of the listed areas. They are presented in a way that helps to design, develop, and manage the operational model. The operational model defines the involved computers, networks, and other platforms on which the application will execute and by which it is managed. The non-functional requirements also feed into the design of technical and application components. For example, service level requirements may imply component performance requirements.

In our simple ITSO Railway PIM and e-mail support example we are looking at these non-functional requirements from a more generalized view. Many of those are addressed already through the choice of WebSphere Everyplace Access as the middleware for our PIM and e-mail support example. Mainly the WebSphere Portal Server running on WebSphere Application Server covers scalability, availability, security, and data integrity.

The non-functional requirements we are concentrating on in our example are not all inclusive. From our perspective, we focused on most important ones.

Performance

Performance is considered when the PIM and e-mail synchronization capability is used. When the PDA's native PIM and e-mail applications are used without a back-end connection, the performance of the back-end system does not influence the performance experienced by the user. System performance will only be noticed during the synchronization process when PIM and e-mail data is sent back and forth with the office PIM and e-mail applications and client PIM and e-mail applications.

The following response times of the application and system are required:

- Frequency of usage/data complexity:
  - High frequency (10–20 times/per day) / 30 sec
  - Low frequency (1–10 times/per day) / 60 sec

Availability

Availability is frequently an important Service Level Requirement. The table below lists the specification of the desired availability. Availability requirements

typically vary by use case and component; each row represents a collection of use cases grouped by common availability requirements.

Table 9-8 Availability requirements

	Availability requirement	**impact of not being met**
Component view		
Handheld device (PDA)	During working time of ITSO Railway executive (7 days a week, 8 hours a day).	Critical. Without devices, executive does not have the opportunity to read his mail information whenever and wherever he is.
PIM and e-mail applications	During working time of delivery personal (7 days a week, 8 hours a day).	Critical. To be able to read the latest requests from superiors and to react, executives must have access to their PIM and e-mail data.
PIM and e-mail synchronization server	Fully available on the working days (5 days a week, minimum 6:00–20:00) On a weekends 9:00–20:00.	High. Critical responses must send and synchronize outside of the regular working hours.

Maintainability

All application components (PIM and e-mail synchronization client, synchronization server, and office PIM and e-mail application and database) must be independent of each other from a development and maintenance point of view. The following components can be designed, tested, and maintained independently of each other:

- Client side (client software for PIM and e-mail synchronization and native PIM and e-mail applications)
- Synchronization server (middle ware component between client and server)
- Office PIM and e-mail application and database (back-end)

Security

The PDA should be protected with a power-on password. The executive must also authenticate against the local PIM and e-mail synchronization client and against the synchronization server and existing office PIM and e-mail application

on the server side (these three different authentications can be separated or combined into one).

Manageability

Manageability focuses on the application management, which includes synchronization profiles. For example, this could be the range of the calendar entries that can be synchronized (last 30 days), amount of data that can be sent (max 200 K/message or attachments are not allowed to synchronize). This could also cover if there is any reason to automate the synchronization mechanism using server-initiated actions.

The PIM and e-mail synchronization devices must be remotely deployable, configurable, and manageable.

System constraints

System constraints are mainly defined by the used handheld device (PDA) and the support of PIM and e-mail synchronization (SyncML client availability).

The specifications of a chosen PDA define the capabilities for the application and include screen size, resolution, browser capabilities, battery lifetime, and weight.

The example constraints shown on Table 9-9 apply to a Window Mobile 2003 PDA.

Table 9-9 Constraints defined by our chosen PDA

Constraint	Value
Screen size	240 x 310 pixel
Battery lifetime	8 hours
Browser JavaScript capable	Yes
Mobile connection	Yes
Wireless connection	Yes
Cradle for synchronization and battery charging	Yes
Power-on password available	Yes

System usability

This section covers the usability of the PIM and e-mail support example from the user's perspective.

The synchronization must be easily understood and easy to use. A device's native PIM and e-mail application should be used. Easy log information for the synchronization results should be available.

Data integrity

The data entered using the devices must be accurately transferred to the back-end system. The WebSphere Everyplace Access, Everyplace Synchronization service takes care of this function.

We will look at these non-functional requirements and how they are fulfilled in the following sections in more detail.

9.1.4 Solution approach

The fictive ITSO Railway customer requirements above (see "Functional requirements and use case model" on page 154 and "Non-functional requirements" on page 163) were the input for the example solution described in the following sections. The following approach was taken to get to a PIM and e-mail synchronization support solution:

- Identify the problem that has to be solved by considering the requirements and use cases to derive an architectural overview.
- Choose a suitable technology and Runtime pattern to create a component model for the solution.
- Deployment of solution and roll-out of application to users. Installation and configuration of solution on the pervasive server as well as on the handheld device must be performed. This includes setup of user access configuration to the application as well as software distribution to the handheld device.
- Maintenance and operation of the solution. Changes and updates of handheld applications as well as middleware components must be centrally managed.

The solution configuration and technology that are used on the server side are described in the upcoming chapters. The deployment and maintenance of the application is covered in more detail in Chapter 16, "Maintaining mobile devices" on page 393.

9.2 Architectural overview

According to the ITSO Railway customer requirements, the ITSO Railway executives must be able to get and process the latest PIM and e-mail data without a constant connection to the ITSO Railways existing office PIM and

e-mail applications. The data is stored on the local device where it can be modified if it is needed. Occasionally, the mobile device will be synchronized with the server so the executive's updates are processed on the railway office PIM and e-mail application server. The synchronization is done by middle ware.

With this conclusion in mind, the Pervasive runtime pattern for PIM and e-mail synchronization can be applied (refer to "Rich Device=Store and forward::Runtime pattern" on page 44). The high-level solution architecture was derived from the Runtime pattern in Figure 3-5 on page 45. To make a fully understandable and descriptive presentation of how the PIM and e-mail synchronization support example could fit into the selected pattern, we use the required Runtime pattern nodes along with the architecture overview picture. This architecture overview contains five parts:

- Devices

  Devices instantiate the user and client including the pervasive client services node (PIM and e-mail synchronization client) of the applied Runtime pattern in Figure 3-5 on page 45. The device is a tool that allows executives to update their PIM and e-mail data through a handheld device (PDA). The executives could use PDA's native PIM and e-mail applications to create responses to their superiors and to update their personal to-do lists. Furthermore, they are able to create calendar entries to arrange meetings with their peers. The entered data is stored locally and the data is ready to synchronize towards the existing office PIM and e-mail application. The PDA runs a pervasive client services (PIM and e-mail synchronization client) application that communicates and synchronizes the data back and forth between the client and server. Using this function, executives could see their latest e-mail data and react in a short period of time for the critical requests that they might get from their superiors.

  An online connection to the existing office PIM and e-mail application in the back-end was not considered appropriate. The reason for that was that the executives might be in a place where a network connection is not available. The other issue was the response time. Executives must have a good response time to their personal PIM and e-mail data. The most convenient way is to use the PDA's native applications to create new memos, events, or items, and in the end synchronize the data.

- External network services

  External network services are the network services that are used for synchronization purposes. The executives and their PDAs use the existing communication services that are offered by the Internet Service Providers (ISPs). ISPs maintain their Wireless Wide Area Transport Network services that are used for wireless connection purposes (such as GPRS and UMTS). ISPs provide a gateway to the Internet that is used to transfer the synchronization data from the client into the server. In the Runtime pattern for

PIM and e-mail synchronization this is represented in nodes ISP Gateway and Voice and/or data services; see Figure 3-5 on page 45

- Web server redirector

  Web server redirector is placed in the de-militarized zone (DMZ), which is used to cover and protect the ITSO Railways intranet network against any possible attacks. There is a Web server redirector service that routes the incoming synchronization requests from the client to the synchronization server that lies in the high security zone area (internal network).

- Pervasive server

  The pervasive server instantiates the directory and security services, personalization server, presentation server, and application server nodes, including the pervasive extension services node of the Runtime pattern in Figure 3-5 on page 45. It acts as a middleware between the device side and the existing office PIM and e-mail application (Collaboration server). It is responsible for the consistent data synchronization from the device to the office PIM and e-mail application.

- Office PIM and e-mail service

  The office PIM and e-mail services represents the existing PIM and e-mail application and databases in the runtime environment (instance of Collaboration services node of Runtime pattern in Figure 3-5 on page 45). This is the backbone that is used for PIM and e-mail data synchronization. The administrator maintains the existing PIM and e-mail environment and gives access rights to executives who would like to use synchronization services.

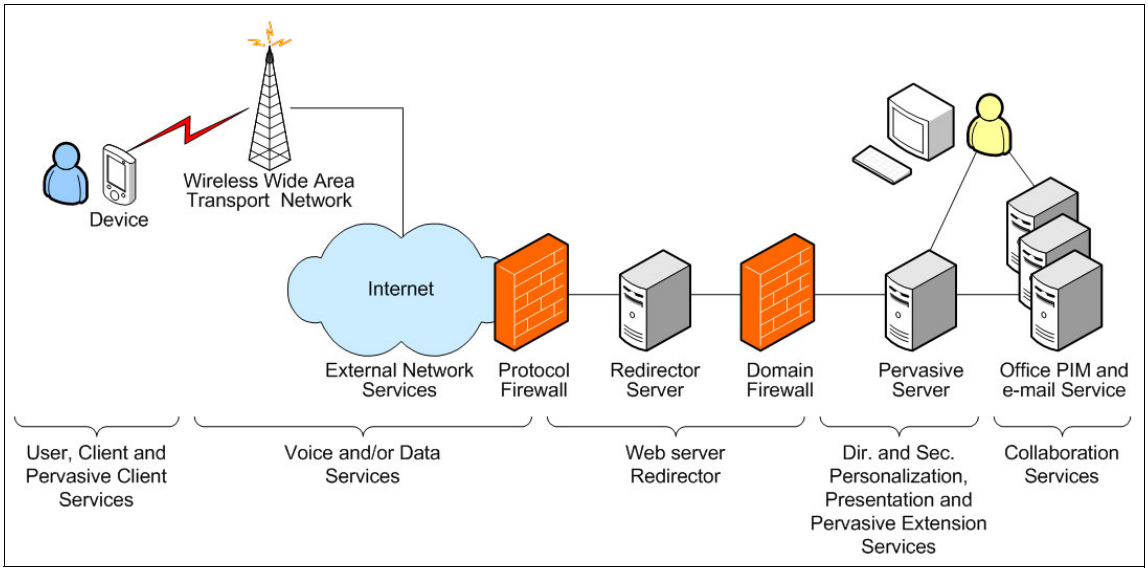


Figure 9-6 Architecture overview diagram for PIM and e-mail synchronization

Taking into account the ITSO Railway business, IT requirements, and use cases, as well as considerations that were described in 3.2.5, “Rich Device=Store and forward::Runtime pattern” on page 44, and mapping proper software into it; see 4.7, “Rich Device=Store and forward::Runtime mapping=PIM and e-mail” on page 74. We can create a more accurate overview of the PIM and e-mail synchronization solution. Figure 9-7 on page 171 shows the necessary components for a WebSphere Everyplace Access based PIM and e-mail synchronization solution, which is presented in solution overview.

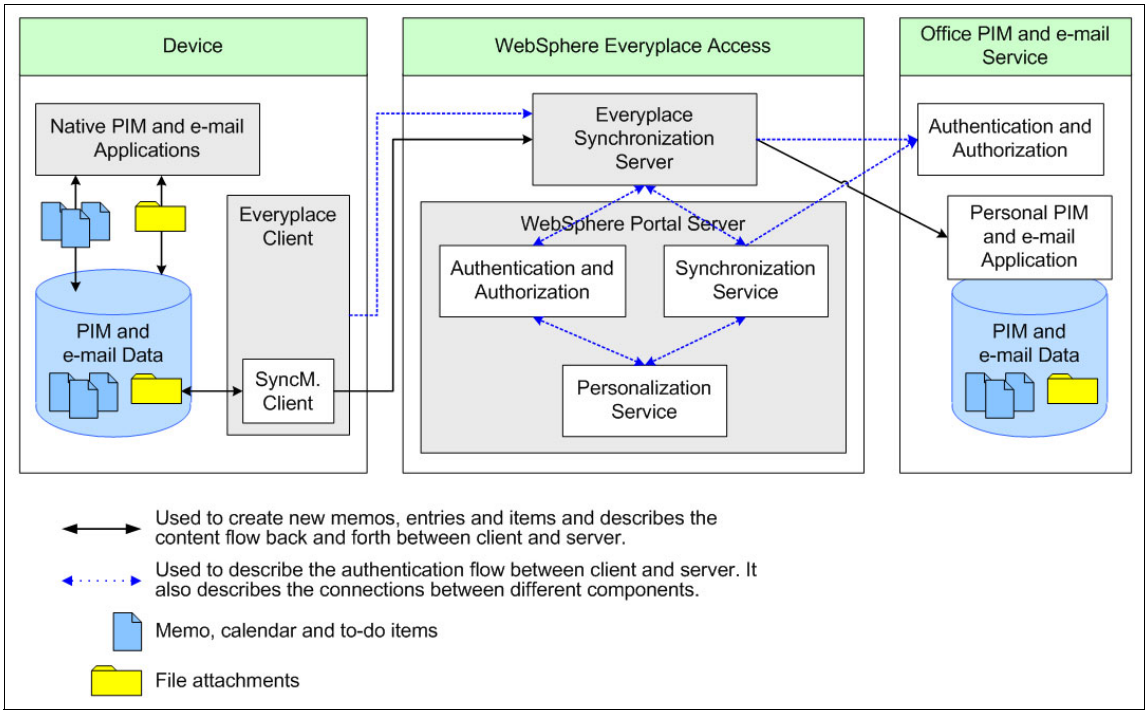


Figure 9-7 WebSphere Everyplace Access PIM and e-mail synchronization content solution components

There are three major blocks involved in this model (3-tier architecture):

- Office PIM and e-mail service

  The office PIM and e-mail service is the existing PIM and e-mail service that stores the ITSO Railways employer's personal PIM and e-mail data. This service is used as a base to synchronize PIM and e-mail data to PDA devices.

  The existing service is used for daily PIM and e-mail transactions by the ITSO railway employees. It must be very stable, and these new synchronization services must not interfere with it in any way.

- WebSphere Everyplace Access (WEA) with WebSphere Portal Server and Synchronization server

  This block is the middleware tier. It contains WebSphere Everyplace Access, including the WebSphere Portal Server, Synchronization server, and Everyplace Synchronization enterprise application. The Everyplace Synchronization enterprise application contains a SyncML servlet that handles the synchronization process. Portal server is used to maintain the user's personalization and authentication information. It is also used to

manage the synchronization services (connections to the back-end and synchronization profiles) between the client and existing office PIM and e-mail services.

- Device

  The device is the interface to the user's personal PIM and e-mail data. Devices contain native PIM and e-mail applications that are used for accessing this data. By using these, the user is able to review his e-mails and entries and create new ones. Device native PIM and e-mail applications take care of the data that is locally stored on the device. The Everyplace Client and synchronization client are the counterparts of the WebSphere Everyplace Access server on the device side. Together they ensure that the user's personal PIM and e-mail data are synchronized properly with the server side. The Everyplace Client also makes sure that the user is properly authenticated and the data information is transferred correctly to the back-end.

This component model uses existing features from WebSphere Everyplace Access and Portal Server. To enable the PIM and e-mail synchronization into the existing environment, the configuration and maintenance is the only effort. No development experience is needed:

- Configuration effort: Configure the existing PIM and e-mail service to support WebSphere Everyplace Access server. Make sure that there are existing network connections in place. Give proper access rights for the users and create new synchronization profiles if needed.
- Effort on the device: Installation and configuration of the Everyplace Client.

9.3 System design overview

This section discusses the chosen solution design of the ITSO Railway PIM and e-mail support for their executives.

9.3.1 General considerations for synchronized enabled applications

There are certain questions that have to be considered when enabling a synchronization-based solution with an existing environment. Taking the questions from the applied Runtime pattern in 3.2.5, "Rich Device=Store and forward::Runtime pattern" on page 44, the following five decision points were addressed in the ITSO Railway PIM and e-mail synchronization support example.

- Decision about the connectivity:
  - Interview your corporate employees and ask for their opinion regarding offline support.

  - Quite often, online capability is good enough if it is available outside the corporate private network.
  - How much we have to pay for an online connection versus an offline connection.
  - It is good to consider what network connections will be used. Should you use partial or completely mobile networks that are more expensive than normal Ethernet service?
- Decision about handheld devices:
  - Consider customer requirements (usability, security, weight, battery lifetime, screen size, etc.).
  - This decision leads to a device with certain computing capabilities (operating system, Web browser, RAM/ROM size, CPU performance, etc.). For example, the screen size and the CPU performance influence the battery capacity, which is proportional to the device's weight. In order to meet a certain battery lifetime the battery capacity has to be adjusted, which again effects the weight of the device.
- What is the targeted audience? Who is going to use this service?

  Usually there are certain units or groups that need PIM and e-mail synchronization services for their business needs. Identifying those groups and units and customizing the required services based on their needs can reduce the implementation expenses. You must also consider the company's on demand strategy: "Pay only for what you use and extend your services together with your business."
- What is the affect of extending the synchronization mechanism into the existing environment?

  Synchronization services must not affect or harm an existing environment, for example, by increasing network response time, causing existing mail services crashes, or introducing questionable security decisions.
- Which security level must be addressed on the device and through the existing office PIM and e-mail applications?
  - This influences the handheld device selection and software choice (password capabilities).
  - Security on the application level could be managed by the pervasive server. How is this managed over insecure networks?

Basically, there are two different approaches for accessing user e-mail databases:

- Partially online and synchronization access for user e-mail databases

This type of behavior allows users to synchronize some of their e-mail content data with the client side. For example, a user would like to synchronize only the title's information into his device. When the user opens a specific memo by clicking the title, the network connection is initiated and will access that memo in online mode. This technique can be achieved by using an Internet Mail Application Protocol (IMAP) based protocol. There are still many restrictions for using that technique. One is that users are not a able to synchronize their PIM data (to-do's, contacts) because IMAP does not support it. The other issue is that this requires the IMAP server to run on the back-end side, which will increase the mail servers I/O load and may increase the response times for existing e-mail services (even for those users who use their e-mail databases through the corporate intranet network). This could also be a security issue. Some companies do not want to allow IMAP connections into the existing e-mail services.

- Offline PIM and e-mail access and synchronization

  The offline PIM and e-mail access and synchronization service between client and server is one of the most often used mobile device applications. This means that the users are able to read and create new e-mails, calendar entries, and to-do lists; and update their contact databases when they are not connected into the existing PIM and e-mail servers. Everything is done locally in their devices, using the device's own PIM and e-mail applications. When users want to send the changes that they made into the server side to maintain e-mail and PIM information, they start the synchronization client, which connects to the network and sends that data to the back-end synchronization server, which authenticates the incoming user/device request. After successful authentication, the data is sent to the existing PIM and e-mail applications. After updating, the server could send new updates back to server, for example, new calendar or e-mail events that the user might have received.

  PIM and e-mail synchronization between client and server uses a standardized SyncML protocol. It is a protocol that is designed to send users e-mail and PIM information over the air and to keep a single copy of the data without any duplications.

  On the client side, the SyncML requires its own client that communicates with the device's native PIM and e-mail applications. Furthermore, the SyncML client is used for sending the data to the back-end synchronization server.

  On the server side, SyncML uses a synchronization server that is responsible to get the incoming requests from the SyncML client, authenticating the incoming requests, and sending PIM and e-mail data into the existing PIM and e-mail application on the server side.

The offline capability gives a more user-friendly experience without any network delays. Synchronization is a very effective way to keep a single copy of PIM and e-mail data up to date both on the server and client side.

9.4 Runtime configuration and deployment

This section explains the necessary steps to enable the PIM and e-mail synchronization for mobile devices. The configuration is divided into two different parts: Server configuration and client configuration. To deploy PIM and e-mail synchronization, we go through one synchronization step previously described in a use case—see Table 9-6 on page 160.

Note: This chapter uses a Domino 6.5.1 server as an e-mail server. A synchronization server is used with WebSphere Everyplace Access 5.0 for Windows 2000 with Service Pack 4. The Everyplace client is installed on top of a Pocket PC 2003. More information on how to configure synchronization in your environment can be found in the WebSphere Everyplace Access V5.0 InfoCenter.

9.4.1 Enable PIM and e-mail server to support synchronization server connection

To enable the synchronization services towards an existing corporate e-mail server, there are few steps that need to be configured and checked.

- The network connection between the synchronization server and the existing e-mail server is enabled, and servers can connect with each other using fully qualified host names.
- Access rights are in place and the synchronization server can access the existing e-mail server.
- The proper ports are opened for data transaction purposes (1352, DIIOP, 80).
- The fully qualified Internet hostname must be entered in the names.nsf file for each Lotus Domino Server that synchronizes with the synchronization server, such as your authentication server and any Lotus Domino e-mail database servers.
- Add the Domino administration ID that the synchronization server uses to read and write to the Domino server to the ACL of the mail.box database on the Domino server.

9.4.2 Configure PIM and e-mail synchronization

After installation and basic configuration of Everyplace Access Services, there are certain steps needed to configure for enabling the PIM and e-mail synchronization between client and server.

WebSphere Everyplace Access is used as a middleware integration layer to enable the synchronization between the client and existing e-mail and PIM server. This middleware layer perspective is a very modular and effective way to extend existing corporate environments to support mobile devices. One of its advantages is that WebSphere Everyplace Access is a separate component that can be added quite easily without interference to existing services that run in the corporate environment.

The administrator has four basic tasks to enable the PIM and e-mail synchronization:

- Configure the synchronization server and existing back-end e-mail services.
- Give proper access rights for users that use the PIM and e-mail services.
- Create a new synchronization profile that is used for synchronization.
- Configure Everyplace Client and synchronization on th eclient side.

Configure server towards existing back-end e-mail server

The Everyplace Synchronization Server contains two main server components: The synchronization server and the Everyplace Synchronization Server enterprise application. The synchronization server contains the PIM adapters for connecting to the back-end servers. The Everyplace Synchronization enterprise application contains the SyncML servlet, which handles the synchronization process. For users to be able to synchronize, these two components must be running along with the supporting servers.

The synchronization server is managed by the synchronization service, which you can access through the Manage Servers portlet. This service is used to start and stop the synchronization server and monitor the synchronization server activity. You can use the Manage Server portlet to monitor all installed synchronization servers and their active users through the service. You can start and stop synchronization servers from this portlet as needed.

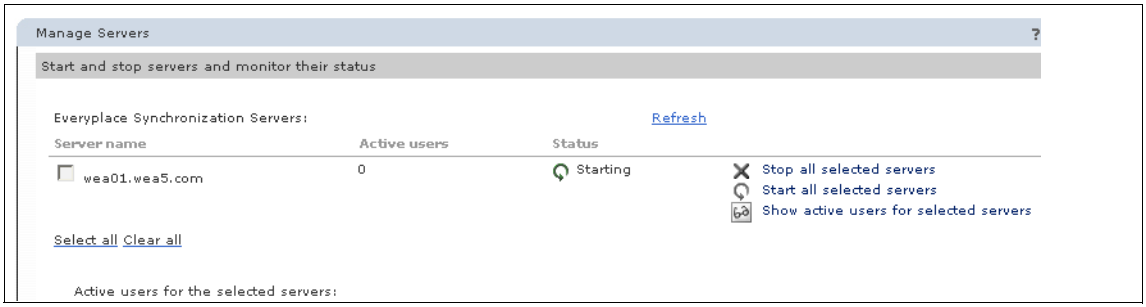


Figure 9-8 Everyplace Synchronization Server - Manage Servers display: Synchronization server

The portlet displays the current status for each synchronization server. It provides the following status states:

- Starting is displayed when the server is loading.
- Running is displayed when the server is operational.
- Stopping is displayed when the server is shutting down.
- Stopped is displayed when the server is not operational but can be restarted.
- Not available is displayed when the portlet cannot get status from the server. Administrator services are not available and the portlet cannot restart the server. When a server is not available, verify that the synchronization server is running.

This section provides an overview of the procedures for configuring and administering the PIM adapter for Lotus Domino. Other adapters like the Microsoft Exchange 5.5 Adapter and the Microsoft Exchange 2000 Adapter are also available. More information about these adapters and how to configure those using WebSphere Everyplace Access server can be found in the WebSphere Everyplace Access V5.0 InfoCenter.

Use the Lotus Domino Adapter portlet to configure the synchronization server to work with Lotus Domino servers. The Lotus Domino Adapter allows data to be synchronized between Lotus Domino servers and mobile devices. A single Lotus Domino Adapter can support multiple Lotus Domino servers and domains.

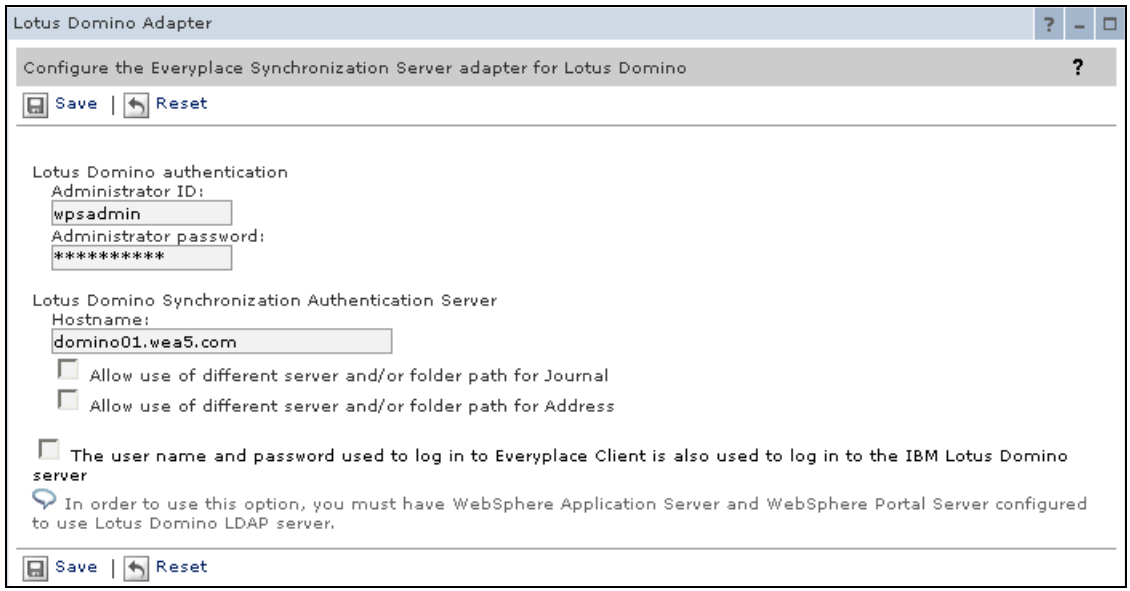


Figure 9-9 Lotus Domino Adapter

It is possible to view the status of any PIM server configured to use the PIM adapter using the Manage Server portlet. The portlet displays a description of the PIM servers and the state of the server. The possible states are Stopped, Running, Starting, Stopping, and Not available.

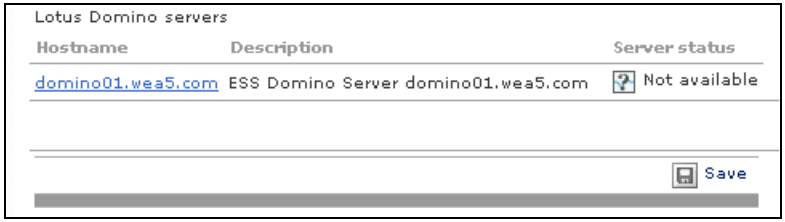


Figure 9-10 WebSphere Everyplace Access - Manage Servers display: PIM adapter

The Lotus Domino adapter requires that the Lotus Notes client is installed on the same computer as WebSphere Everyplace Access. The synchronization server and the Lotus Domino Adapter use this Lotus Notes client to communicate with the Lotus Domino e-mail database that the users synchronize with.

When you have managed to configure these steps and are able to see the synchronization server and Domino PIM adapter in running mode, you can give access to users to use this service.

Access rights for PIM and e-mail synchronization users

WebSphere Everyplace Access V5 provides administrators with the functionality to create multiple users at the same time. This is a very convenient way to create and handle users that already exist in a corporate environment and who would like to extend synchronization services onto their daily business package's application selection.

Using the Bulk Load Users portlet, the administrator can create new synchronization users from existing WebSphere Portal users or from a user list imported from Microsoft Exchange or Lotus Domino.

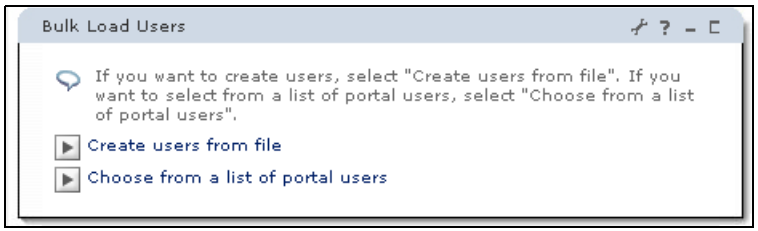


Figure 9-11 Bulk load portlet

The Bulk Load Users portlet lets you create new synchronization users from a user list you import from Microsoft Exchange or from Lotus Domino, and then notify each user of their new WebSphere Portal username and password by e-mail. For information about how to export a user list from Microsoft Exchange or Lotus Domino and use it to create new users, see the portlet help for the Bulk Load portlet on the Administration page and exporting users for use in the Bulk Load portlet.

Create synchronization profiles

There are basically three different ways to create synchronization profiles. One option is to use Bulk Load portlet. The second option is for an administrator to create their own, corporate standard based profile. The last is to allow users to create their own profiles.

- Using the Bulk Load portlet, the administrator can assign a device profile to multiple new or existing users. He can create a new device profile or edit an existing bulk load profile and assign the profile to existing synchronization users or new users created from an imported user list. In addition, the administrator can send the users an e-mail informing them of the new device profile. For more information about using the Bulk Load portlet to assign administrator profiles, see the portlet help for the Bulk Load portlet on the Administration page.

- The second option is to allow the administrator to create his own synchronization profile for using the Synchronization Administrator Profiles portlet.

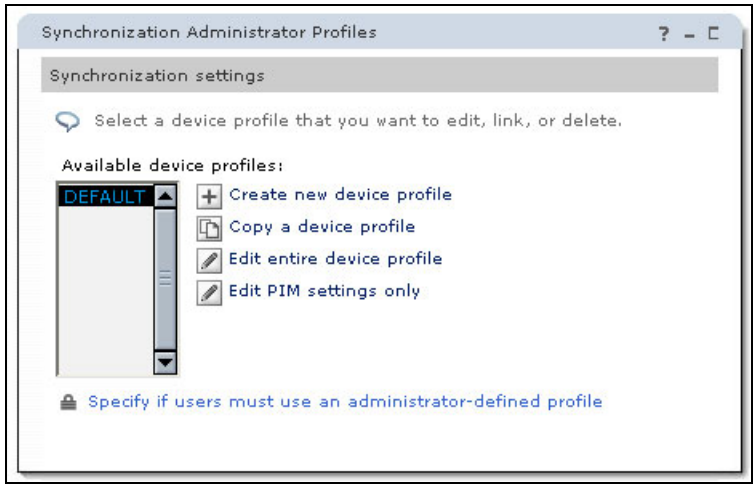


Figure 9-12 Synchronization Administrator Profiles

By creating a new profile, the administrator could specify more detailed behavior that can be used for different devices. Most often, this is very useful to do due to different device functions and their requirements. For example, users who use devices with very limited memory will not want to receive their whole calendar entries and mail attachments on the client side. They only want to view the current week's entries.

Figure 9-13 Synchronization Administrator Profiles: Create new profile

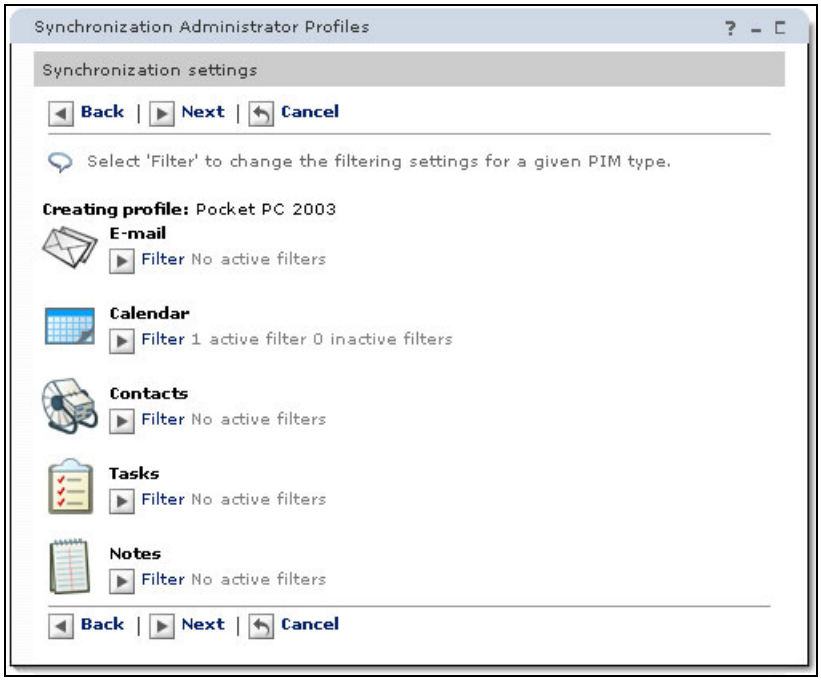


Figure 9-14 Synchronization Administrator Profiles: New profile with calendar filter in it

After the administrator has created the profile, he has to determine if there are any conflicts between client data or server data. After these steps are configured, there is one new synchronization profile that can be used for your implementation.

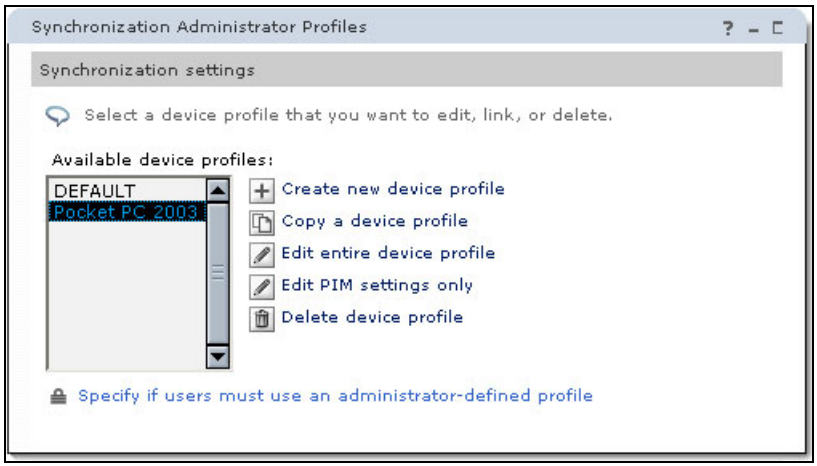


Figure 9-15 Synchronization Administrator Profiles: PocketPC 2003 profile

- The third option to create synchronization profiles is to let users create their own profiles. This is good for users who would like to use personalized synchronization preferences instead of using the default corporate profiles. To do this, users have to go to the Mobile setup tab and open the Synchronization User profiles-portlet.

  A user has to identify that the current profile is his own by typing the user name and password.

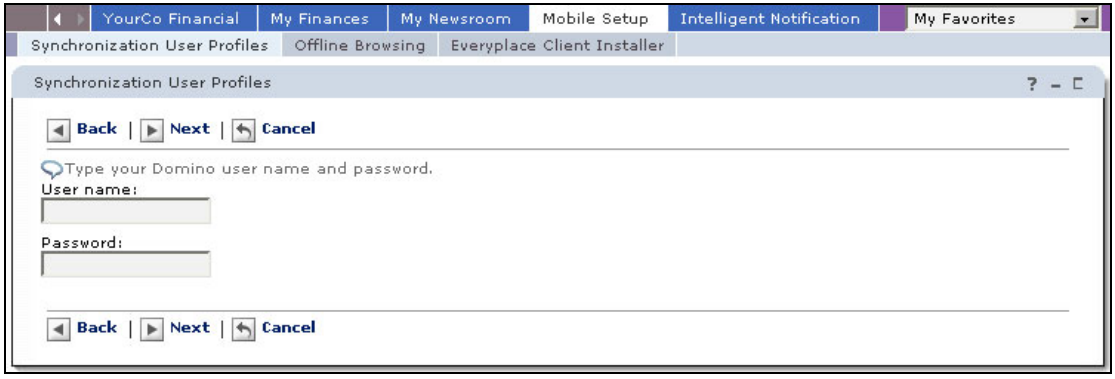


Figure 9-16 Synchronization User Profiles

After the user has inserted his user name and password, he can see the existing profiles that the administrator has created previously. Users could choose existing ones or create his own. The steps for creating a new personal profile is the same as it is in the administrator synchronization

profile. After the user has finished his profile and selected the appropriate locations, he should get the information in his portlet, as shown in Figure 11-14.

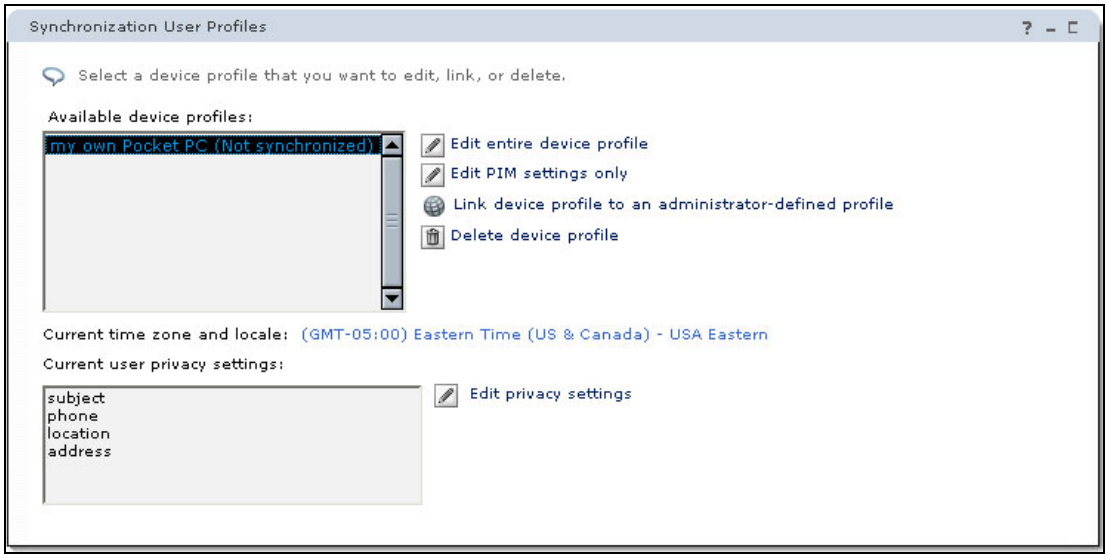


Figure 9-17 Synchronization User Profiles: Personal profile

9.4.3 Configure Everyplace Client and synchronization on client side

This section describes the configuration steps that are needed to be done on the client side.

Everyplace Client software is easy to install, configure, and operate. All you have to do is start the setup program for your device type and put your PDA in the docking cradle. The installation program copies the files to the PC. Everyplace Client files are installed on your PDA the next time you synchronize.

WebSphere Everyplace Access contains different Everyplace clients for different devices. This chapter will cover only the Pocket PC device. More information about the Everyplace Client and how to install and configure it can be found in the WebSphere Everyplace Access V5.0 InfoCenter.

The Everyplace Client can be launched using the Setup.exe in the appropriate directory for your device type located on the Everyplace Client installation CD. After language selection, a Welcome to the Install Shield panel is displayed. Select **Next** to continue. Remember that you accept the terms of the Software License Agreement by selecting **Yes**. If you choose No, the installation will be terminated. Select the destination where the installed files will be located and

select **Next** to continue. At this point, you should be able to see the Components panel that allows you to select all the components that you want to install.

For synchronization purposes, you only need the PIM and e-mail synchronization component. Select that current component and select **Next**. On the Start Copying the Files panel select **Next**, and on the InstallShield Wizard Complete panel select **Finish**.

You then need to perform these steps on the Pocket PC side:

1. Put your Pocket PC on the docking cradle and perform the *Activesync*.
2. Input the username and password (twice) for Everyplace Client, and select **OK**.
3. Input the name and e-mail address for the e-mail application on the Sync Client panel, and select **OK**.

9.4.4 Using the PIM and e-mail synchronization

Everyplace Client uses a browser-based user interface. In order to open the Everyplace Client user interface, click the Everyplace Client icon and enter your username and password on the login panel. The main view will open.

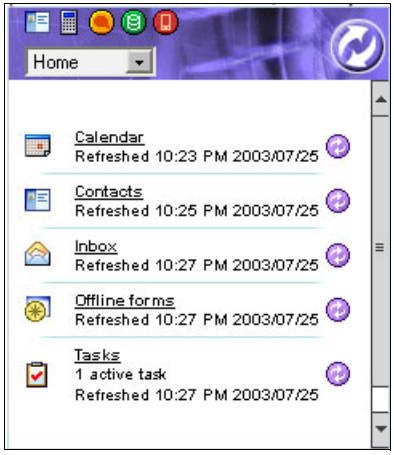


Figure 9-18 Everyplace Client: Main view

The Everyplace Client enables you to synchronize data with WebSphere Everyplace Access servers using industry-standard SyncML technology. The Everyplace Client uses the term *Refresh* instead of *Synchronize*. Click the Refresh icon on the PDA device to refresh the data.

After clicking the Refresh icon, the Everyplace Client communicates with the server to determine which synchronization type the client should use. The Everyplace Client supports two types of synchronization: Slow synchronization and normal synchronization.

- Slow synchronization

  During a slow synchronization, all the items in the client databases are compared with all the corresponding items in the server databases on a field-by-field basis. Everyplace Client sends all the client data to the server, and the server does a field-by-field analysis, comparing its own data with the data received from the client. After analyzing the data, the server returns all the modified information to the client. Because a slow synchronization is a time-consuming process, you should only use this method when the PDA device is connected via a high-speed method such as in the docking cradle. A slow synchronization occurs in the following situations:

  – The first time you synchronize the Everyplace Client

    When the server requests a slow synchronization as a result of data inconsistency after the client has initiated a normal synchronization. This can occur if the server encounters synchronization errors.

  – When you request a full refresh from configure PIM and e-mail settings

  Before initiating a slow synchronization, the Everyplace Client displays a confirmation pop-up. To start the slow synchronization process, select Refresh in the pop-up window to begin the slow synchronization.

- Normal synchronization

  During a normal synchronization, the client and server exchange information about any modifications to the data on the client and the server. Therefore, only new and changed information is synchronized from the server to the client and from the client to the server. The Everyplace Client always requests a normal synchronization except under the circumstances noted above for a slow synchronization. Since only modified information is exchanged, a normal synchronization is less time consuming than a slow synchronization.

The *My Settings* (user preferences) provides quick access to user settings. From this page, users can personalize their Everyplace Client interface and settings. This section provides the following information:

- The Overview provides a starting point where users can modify their Everyplace settings.
- Working with custom categories allows users to create new categories to organize the user's information and application data as needed.
- Personalizing shortcuts is used to add up to eight applications that are most used by the user into the shortcut bar.

- Working with themes allows users to personalize and customize the look of the interface and choose from several different installed themes.
- The *Synchronization Settings* allow you to change the time and the way the synchronization is handled. There are three different options:
  - Timed refresh - When connected, the device will refresh based on the user-defined interval, set in minutes.
  - Cradled - The device is refreshed only when connected.
  - Manually only - The device is refreshed only when the user taps the connection status and refresh button in the main view.

  It is also possible to get information for each synchronization by enabling the radio boxes.
- Working with network profiles lets users control how many items get synchronized during a refresh and to which servers to synchronize. Users can switch profiles using the profile drop-down selector under *My Settings*. To work with profiles, users need to click *Network profiles*. On the Network profiles page, users can add, edit, and delete profiles, similar to working with *Categories*. Click the *Edit* icon to modify the name of a profile.
- Mobility Client allows users to configure the *Connection Manager* settings.
- *Security* defines the username and password for accessing the client on the device and for accessing your portal. Changing the username and password on the client does not change the user's portal username and password.
- *Replace Data* gives options to replace device data or configure free memory.
- Using *Software Update*, you can update the Everyplace Client and device software.
- *Servicing* enables or disables tracing for field support.

9.5 Summary

This chapter summarizes the basic steps to add PIM and e-mail functionality into an existing e-mail environment.

Customizing and extending the existing enterprise environment to cover PIM and e-mail synchronization is very often a straightforward process. It rarely requires development experience because there are a lot of existing products that support PIM and e-mail synchronization. PIM and e-mail synchronization is based on the SyncML Data Synchronization standard. SyncML has been an initiative recently consolidated into the Open Mobile Alliance (OMA).

More information about SyncML can be found at:

http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html

The following table describes the actions in high level that you need to consider when researching and implementing PIM and e-mail synchronization functionalities. The table below also describes all those parties who are related to those actions.

Table 9-10 Enable PIM and e-mail synchronization and related parties

Task/action	Supplier services	Administ-rator	Security	ISP	CTO	User
Define used channels and create contracts	X			X	X	
Create and configure needed channels	X	X	X	X		
PIM and e-mail sync. installation	X	X				
Enable access in existing PIM and e-mail services		X	X			
Configure PIM and e-mail sync. towards existing e-mail services	X	X	X			
Access rights for PIM and e-mail sync. users		X				
Create sync. profiles		X				X
Install sync. client on user devices		X				X
Configure sync. clients		X				X

It is not necessary to load users for installing and configuring their devices. That can also be done remotely by using the *Device Management* services. More information about the Device Management can be found in Chapter 16, “Maintaining mobile devices” on page 393.

The table above describes the high-level tasks that need to be in place and the interested parties that are related to achieve the final goal.

The PIM and e-mail synchronization requires a client-to-server connection. It means that each client has to have a SyncML-based client application. The client needs to be able to send and get messages from the server and route them to the client's native PIM and e-mail applications. On the server side, there must be a synchronization server that handles the incoming and outgoing requests and routes the data to existing e-mail servers.

To extend the enterprise environment to use PIM and e-mail synchronization, we highly recommend discussing the solution with all involved parties (security and system administrators and the users).

You should also carefully consider the possible extra loads that this new service generates. An enterprise does not want to degrade its existing services by bringing new services into it. This has been covered in 3.2.6, "Rich Device=Store and forward::Runtime pattern variation 1" on page 45.

Further information for how to enable PIM and e-mail synchronization and the configuration steps can also be found in the WebSphere Everyplace Access InfoCenter and other WebSphere Everyplace Access IBM Redbooks.

10

Web access to ITSO Railway's timetables

Using a mobile portal approach is potentially the least difficult way to make information available to users of pervasive devices. It is useful for displaying content that has been adapted for smaller screens to customers without requiring them to log in.

This scenario describes an approach to make train schedule information available to ITSO Railways customers who use Web browsers, PDAs, and Web-enabled cellular telephones. This is accomplished by using an existing time table database and by developing a single Web application that can display this content on multiple devices.

The approach takes into account customer wants, needs, and options for implementation of such a system. System architecture and system design are discussed. At the end of this chapter is a description of the steps needed to develop such an application.

10.1 Overview

ITSO Railways wants to provide access to train schedules to its customers in their homes, offices, and on the go. Realizing that these customers may be using a wide range of devices to access this information from Web browsers on their PCs to micro-browsers on their Web-enabled mobile phones, ITSO Railways has decided to implement a Web application that will accommodate many devices and markup languages.

ITSO Railways wants to be able to show a customer the times that a train is scheduled to depart from a particular station.

In the future, ITSO Railways also wants to be able to extend this application to show the most recent data in the ITSO Railways train schedule database so that customers will know if a train is running ahead of schedule, slightly behind schedule, or on schedule in real-time. We keep this in mind when developing the sample application.

The following diagram is the Runtime pattern we used for the implementation of this scenario.

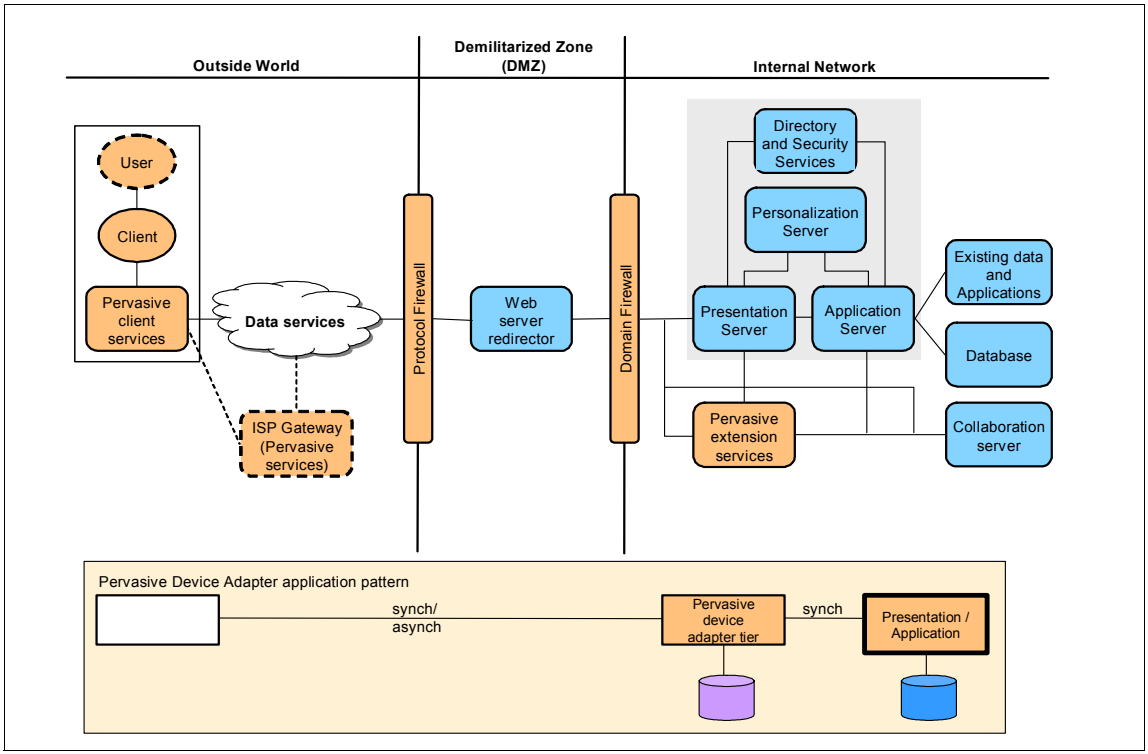


Figure 10-1 Runtime pattern for the Pervasive Web access scenario

The runtime mapping that we used for this book is seen on the following diagram.

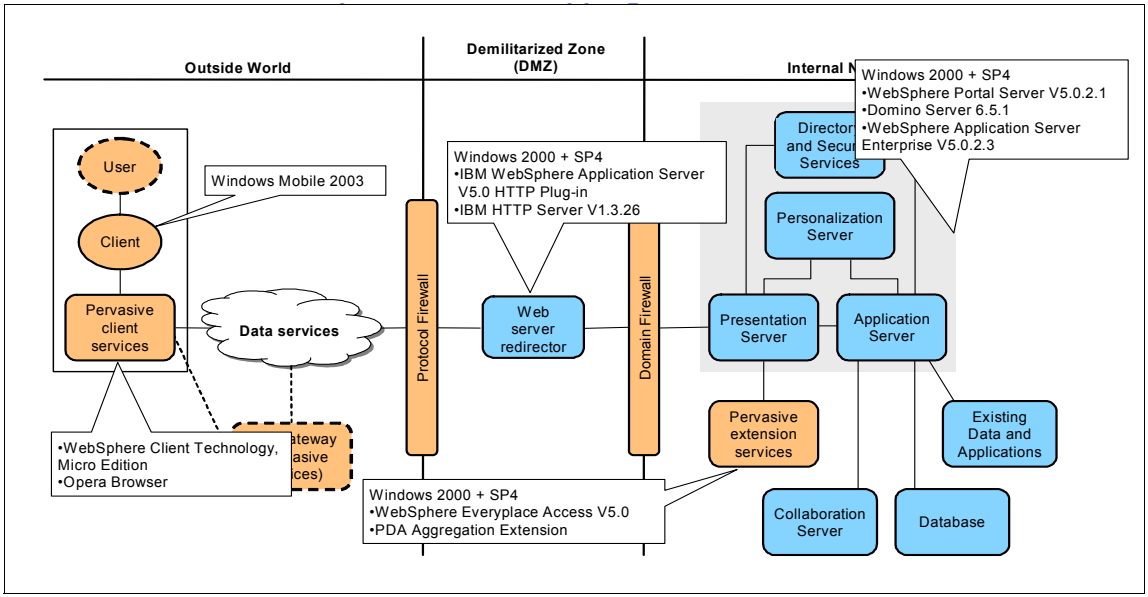


Figure 10-2 Pervasive Device Adapter::Product mapping=Windows

10.1.1 Customer requirements

Typical ITSO Railways customers vary and can include frequent users, occasional users, business users, vacationers, and commuters. These users all desire accurate and timely information of train schedule data.

Some users may use a standard Web browser to access this information from their home or office computers. Other users may desire access to this information from a Web-enabled cellular telephone or their PDA while they are on the go.

10.1.2 Use case model

A customer uses the railway systems frequently. The customer recently purchased a PDA and uses it to access various Web sites. The customer would like to use her PDA to access the railway information so that she can manage her time better when travelling.

Example use case

In our example use case:

1. The customer logs onto the PDA.
2. The customer accesses the ITSO Railways Web site.

3. The customer selects the train schedules.
4. The customer selects her departure and destination locations.
5. The customer selects when she wants to ride.
6. The customer reviews the trains schedule.

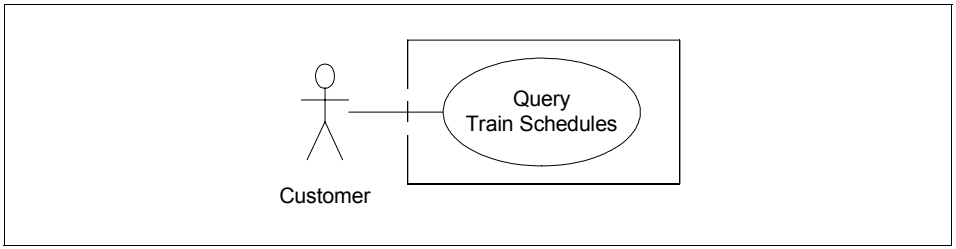


Figure 10-3 Use case scenario

10.1.3 Key requirements

Following is a list of business and IT requirements.

Business requirements

These are:

1. Increase customer satisfaction by providing customers with anywhere access to railway schedule information.
2. Provide customers with mobile access to railway schedule information.
3. Reduce information-only customer requests.
4. Make information available 24 hours a day, 7 days a week.
5. Provide the latest information relating to the train schedules.

IT requirements

These are:

1. Integration with and extending existing railway information systems and train schedule systems.
2. Make the application easy to use and learn.
3. Extend the existing Web/portal system to mobile devices.
4. Use tools that support the end-to-end development of such an application and simplify development.

10.2 Architectural overview

This system is designed and implemented much like that of a standard Web server or Web portal. Instead of a Web server, ITSO Railways would host and serve this content from a pervasive access server. This server would be able to determine what type of device is accessing the railway timetable application based on the header information sent when the device attempts to connect to the server. It would then be able to offer to the device content encoded in a markup language supported by the requesting device.

In between the pervasive server and the Internet resides a load balancing and connection management server. This server would be responsible for distributing heavy loads and could be responsible for managing connections with devices that have the capability of connecting to the Internet via more than one type of connection (for instance, when roaming from a cellular connection to wireless Ethernet/802.11x).

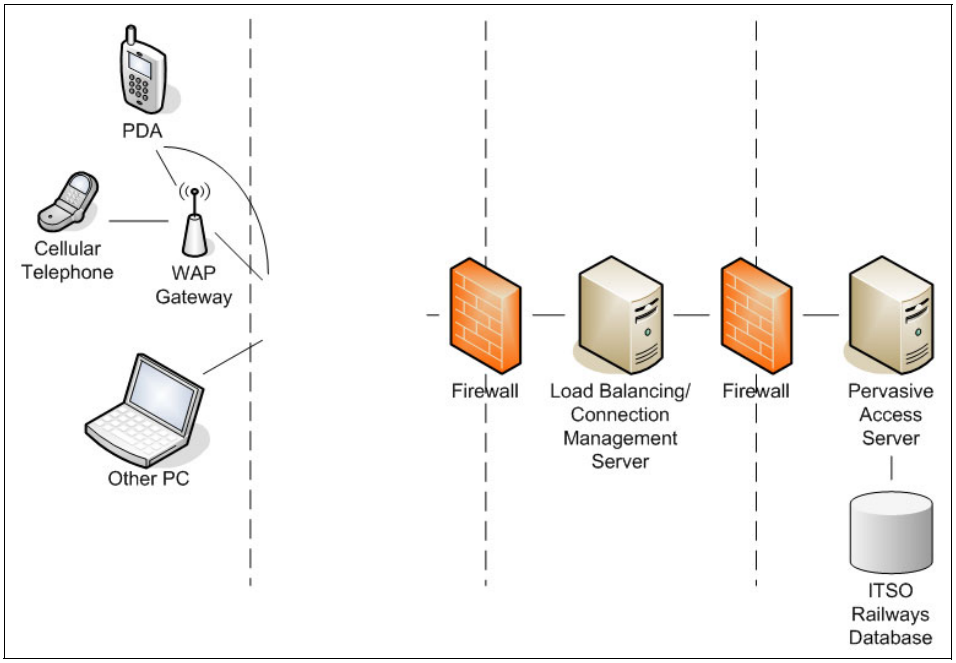


Figure 10-4 Architecture

In this arrangement, mobile devices can connect to the pervasive server by any means that they can connect to the Internet. Cellular telephones and cellular-enabled PDAs connect to the Internet on a cellular network through a device called a WAP Gateway. A WAP gateway is responsible for encoding content in such a way that it is usable by mobile devices. PDAs, laptop PCs, and

other devices can connect directly to the Internet via wireless Ethernet (802.11x), wired Ethernet, cradle, or other means.

10.3 System design overview

In this scenario, we concentrate on developing the content to be hosted on the pervasive access server for the mobile devices shown in the above architecture. We want to enable our application to query a database of train schedule data that resides on the internal corporate network. The application introduced here consists of a series of JSP files that have been specially modified to display content correctly according to the markup languages used by Web browsers (HTML), PDAs (PDA markup), and Web-enabled cellular telephones (WML, cHTML). Each of these pages allows the customer to query train schedule information based on departure location, arrival location, and the time of day that the customer wants to use the train.

10.3.1 Application flow diagram

Mobile applications developed for WebSphere Everyplace Access are portlets just like those in WebSphere Portal that are extended to use multiple markup languages. They use the same business logic and event handling as WebSphere Portal portlet applications with certain constraints.

In this scenario, querying the train schedule is conducted as a series of four steps:

1. Selecting a departure location
2. Selecting an arrival location
3. Selecting a time to leave
4. Reviewing the train schedule based on the previous selections

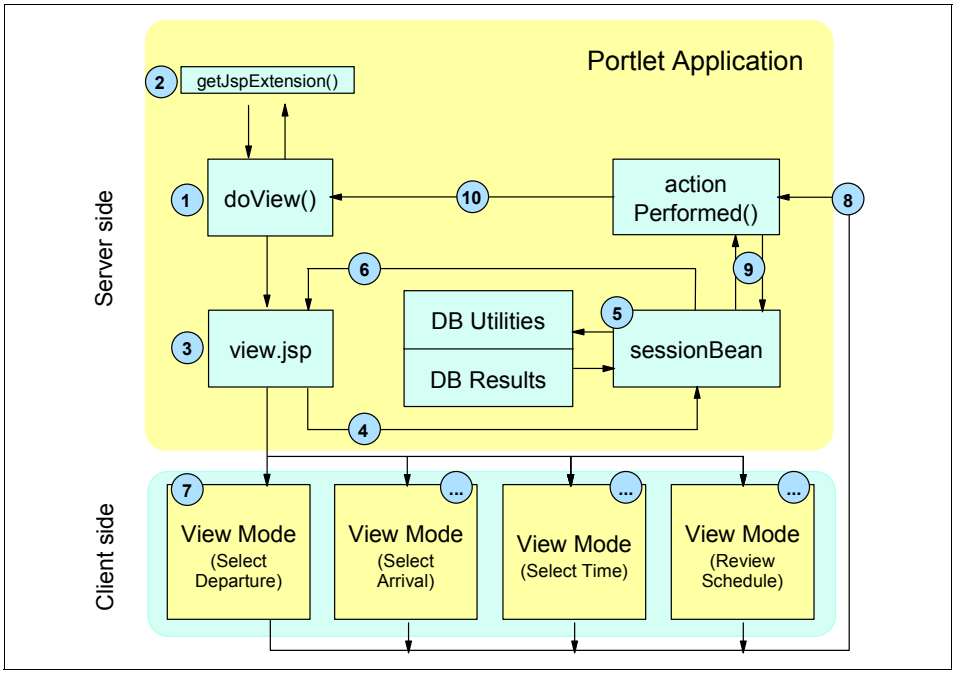


Figure 10-5 Event handling

The sequence flow for this scenario through the application is as follows:

1. Initially, the doView() method is executed. It is the responsibility of the doView() method to display a portlet to the user in view mode.
2. The doView() method calls the getJspExtension() method. This enables the portlet to determine which set of JSPs to use when rendering the content based on the markup languages supported by the requesting device.
3. The view.jsp file is called to render an initial screen.
4. The view.jsp calls the sessionBean to get a listing of departure locations.
5. The sessionBean accesses a listing of departure locations by using the database utilities and database results classes to query the ITSO Railways database.
6. This information is returned to the view.jsp for processing.
7. The portlet is shown in view mode to the user. In this scenario, it lists a series of links showing the available departure locations for ITSO Railways.
8. The user selects the desired departure location. The action is sent as part of the portlet URI. Upon submission, the actionPerformed() method is executed to process this action.

9. The data sent with the action indicating the desired departure location is passed to and stored in the sessionBean.
10. The doView() method is called again to perform step number two of the process used to query the train schedule.

The sequence flow above is iterative. The same process occurs when the user chooses an arrival location and is shown the resulting train schedule information.

10.3.2 Design considerations

There are several ways to adapt existing information to mobile devices. When designing your application, you will want to consider the capabilities of each method.

Transcoding technology/content adaptation

One common approach is to adapt pre-existing Web content using transcoding technology (sometimes referred to as content adaptation). The transcoding technology included with WebSphere Everyplace Access transforms content based on the capabilities of the devices accessing the content. For example, devices with a small screen will receive a scaled-down version of the information, and devices requiring a different markup language (for instance, WML on cellular phones) will receive the information in the markup best suited for that device. Features included with these transcoding technologies allow you to adapt and customize content in a variety of ways.

It is important to note that transcoding technology has limited capabilities, especially when it comes to user interaction with a mobile application.

Attention: See the chapter on Transcoding Technology in *WebSphere Everyplace Access Version 4.3 Handbook for Developers*, SG24-7015-01, for more information on transcoding technology.

Customized markup-specific JavaServer Pages

A second approach for developing content for mobile devices is to create an application with a set of customized JSPs for each markup language used to access and interact with the content. This allows the developer to customize his mobile Web applications for the devices used to access the content. Currently, the WebSphere Everyplace Toolkit allows the developer to create custom JSPs for HTML (for Web browsers), cHTML (for iMode-capable mobile phones), WML (for Web-enabled mobile telephones), and PDA markup (for PDA devices).

Customized device-specific JavaServer Pages

Maintaining Web applications for such a wide variety of screen sizes, markup languages, display capabilities, and means of interactivity on each supported device is very complex.

To further the concept of developing *markup-specific* applications, the Multi-Device Authoring Tools included with the Everyplace Toolkit allow you to automatically convert a generic mobile application into *device-specific* applications. These multi-device applications are based on Struts, which is based on the Model-View-Controller (MVC) paradigm. A multi-device application consists of a controller.xml file; a DLG file for each of the views; generated JSP files for each supported device; and Java Source that contains the business logic, methods, and beans.

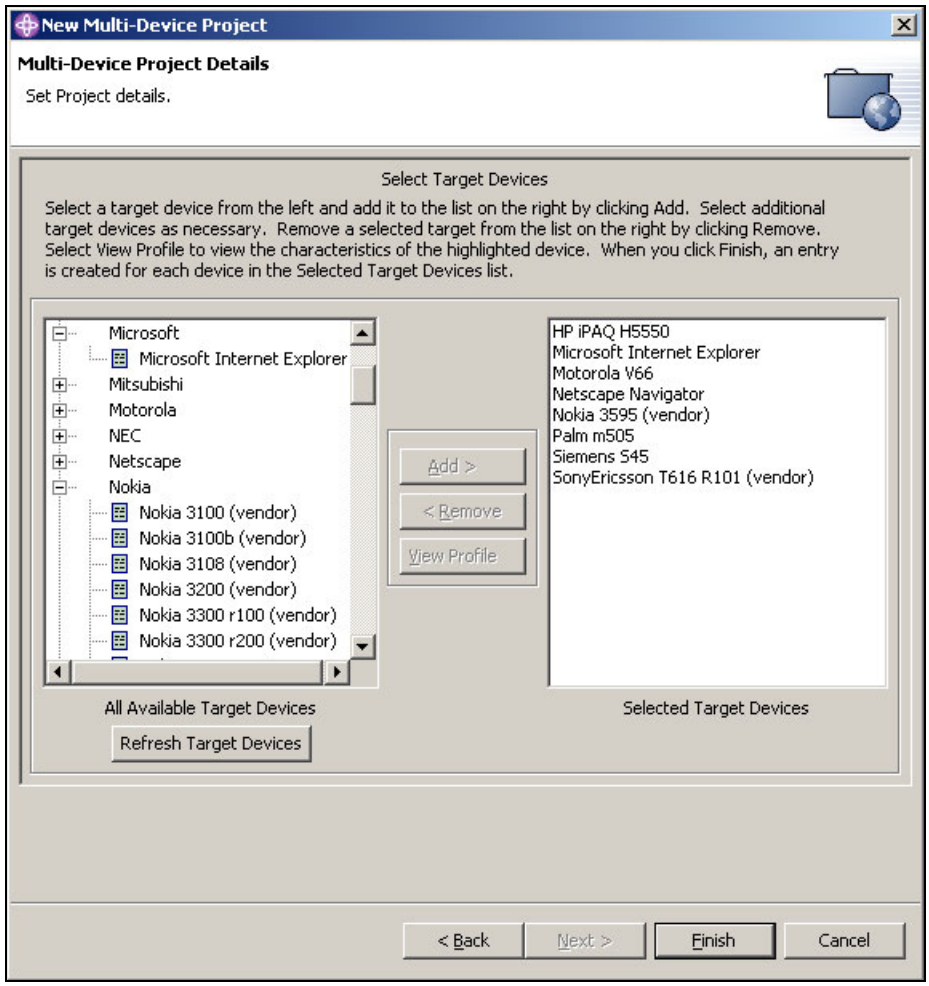


Figure 10-6 Selecting supported devices

The Multi-Device Authoring Tools also include an editor to allow you to define both a generic and a target-specific application flow diagram.

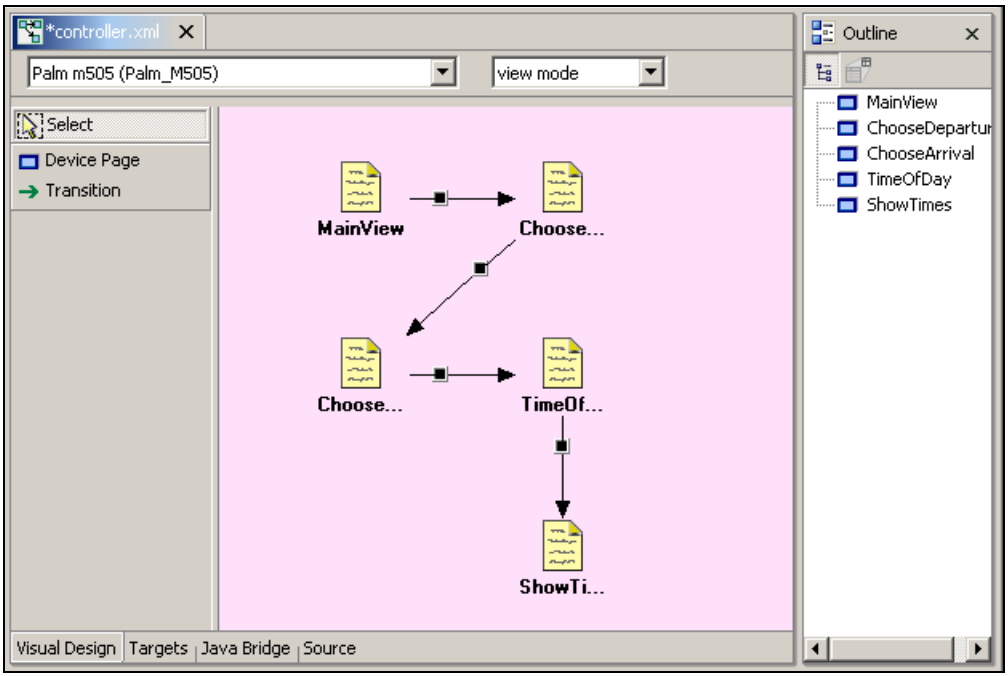


Figure 10-7 Application flow editor

> **Tip:** For more information on developing multi-device portlet projects, refer to the section titled “Developing multi-device applications” in the WebSphere Studio Site Developer InfoCenter.

10.4 Application development

Here we describe the development of a sample application that meets the customer requirements described previously. The source code to this application can be found in Appendix A, “Additional material” on page 435.

For this application, we use the Everyplace Toolkit to create a framework for our mobile application that supports HTML, PDA, and WML markup languages. We then test this application using a Web browser, PDA simulator, and a Web-enabled mobile telephone simulator.

> **Tip:** For more information on developing applications for mobile devices, refer to *WebSphere Everyplace Access Version 4.3 Handbook for Developers*, SG24-7015-01.

10.4.1 Create the portlet application project framework

To create the application framework, create a Portlet Application Project using the Everyplace Toolkit wizard. On the screen that asks you to select the supported markup languages for the project, select the WML and PDA markup languages. You can optionally select cHTML markup for iMode-capable devices.

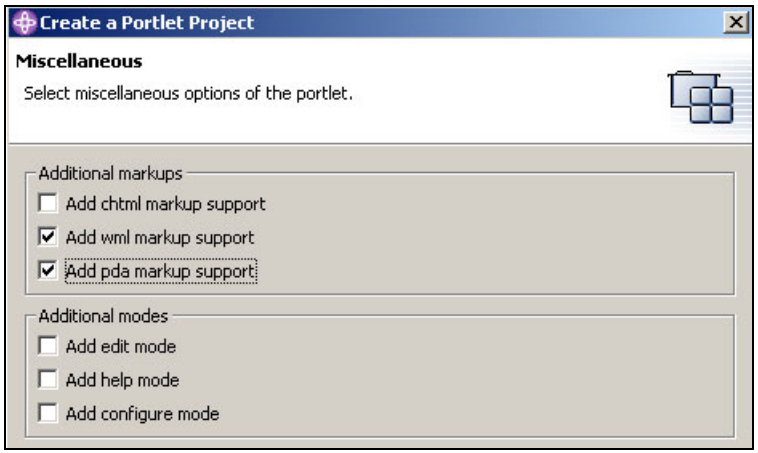


Figure 10-8 Supported markup language selection

The wizard will generate a framework for your project. It will generate a separate folder for each set of JSPs corresponding to each markup language supported by the project.

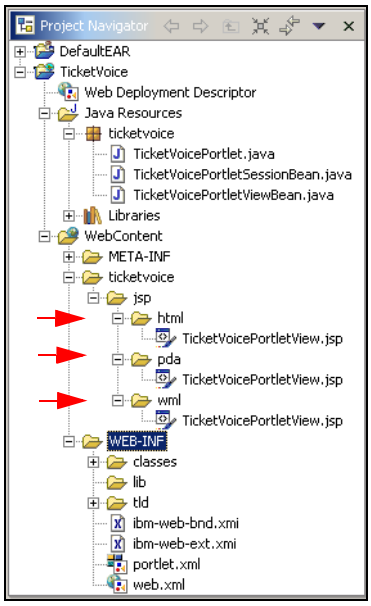


Figure 10-9 Portlet application framework highlighting markup-specific JSP folders

10.4.2 Add supporting files and business logic

To support your application, you will need to add and modify files to handle the business logic and data management. For this application, we have added a JavaBean to store data from user input and navigation, and a database utilities class to manage interaction with the ITSO Railways train schedules database.

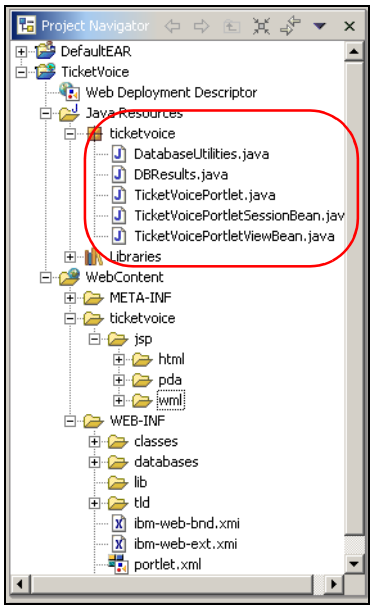


Figure 10-10 Adding support files

10.4.3 Add connectivity to the existing train schedule database

The sample database that we are using for this scenario contains two tables: One for locations and one for times. They are structured as follows.

Note: The files needed to create this database are in Appendix A, “Additional material” on page 435. Please follow the steps described in the sample application’s directory to set up the database.

The locations table has three records for location names: Raleigh, Durham, and Wilmington.

Table 10-1 The times table contains entries for possible schedule information

NAME1	NAME2	TIMEOFDAY	TIMES
Raleigh	Durham	am	6 am
Raleigh	Durham	am	8 am
Raleigh	Durham	pm	6 pm
Raleigh	Durham	pm	8 pm

NAME1	NAME2	TIMEOFDAY	TIMES
Raleigh	Wilmington	am	7 am
etc	etc	etc	etc

The database utilities class can be modified to point to any database on any supported database management system. WebSphere Studio Site Developer provides the Cloudscape database management system for local development of database-driven applications.

10.4.4 Customize and add JSPs for specific markup languages

When all business logic and connectivity to the database is in place, it is time to customize the JavaServer pages to display information to the user of the application.

As shown in Figure 10-9 on page 202, there are separate folders for each supported markup language. You should add to and modify these files in accordance with the specifications for each markup language used to build your applications. For markup languages such as WML it is extremely important to correctly code the markup, or the content will not display and/or function correctly on the device. The following sample shows the code used to dynamically generate a listing of departure locations for ITSO Railways in WML. In WML, all text to be displayed on a phone display must come between and tags. In addition, line breaks are indicated by
.

Example 10-1 Sample dynamically generated WML

```
<% sessionBean.setDb_sqlstring("SELECT NAMES FROM APP.LOCATIONS");
sessionBean.getJDBCResult();%>

<p>
Welcome to ticket portlet, please select your departure location:
<% for(int x=0; x<sessionBean.getDb_rowCount(); x++){%>
   <br/><a href="<portletAPI:createURI><portletAPI:URIAction
   name="submitDeparture"/><portletAPI:URIParameter name="start"
   value="<%=sessionBean.getDb_SQLresult(0,x)%>"/></portletAPI:createURI>">
   <%=sessionBean.getDb_SQLresult(0,x)%></a>
<% } %>
</p>
```

Tip: See http://www.w3schools.com/wap/ for more information on the WML markup language.

10.4.5 Test and debug the application

You can test and debug the application as you would a normal portlet development project by running it in the WebSphere Portal Test Environment.

> **Tip:** See *IBM WebSphere Portal V5: A Guide for Portlet Application Development*, SG24-6076, for help setting up the WebSphere Portal Test Environment and running your application.

Running the portlet application project loads the portlet into the Web browser in the WebSphere Studio Site Developer IDE. By selecting a departure location, an arrival location, and a time of day, a user can view the trains scheduled to depart.

Figure 10-11 Sample application in Web browser

While the WebSphere Portal Test Environment is running, you can also access this portlet through any external PDA or cell phone simulators by entering the following URL:

```
http://localhost:9081/wps/portal/!ut/p/.cmd/LoginUserNoAuth?userid=wpsadmin&password=wpsadmin
```

For testing applications that have been developed for the PDA markup, you can use a simulator such as the Palm OS simulator, which is downloadable from:

http://www.palmsource.com/developers/

Microsoft also has SDKs and emulators available for download for Pocket PC-based PDAs at:

http://msdn.microsoft.com/windowsmobile

After configuring the simulator to access the Internet, you can enter the above URL to access your portlet application's content.

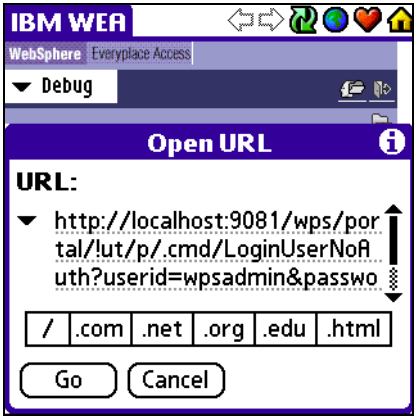


Figure 10-12 Entering URL into Palm Simulator

After clicking the **Go** button, you can interact with your application just as if you were using it on a real Palm OS-based device.

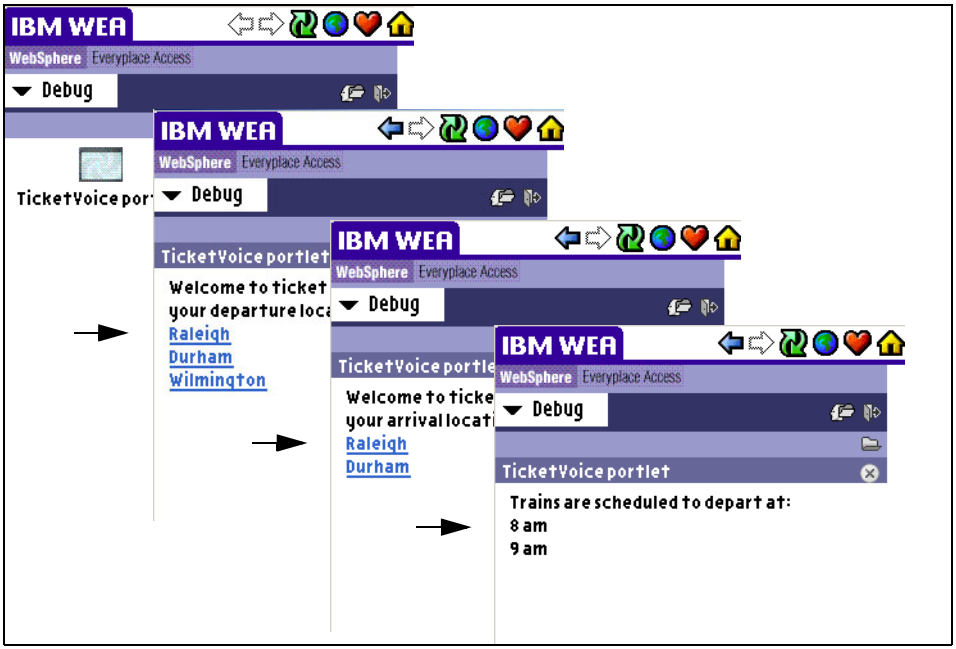


Figure 10-13 Sample application running in Palm Simulator

The same can be done when developing applications for Web-enabled mobile telephones. Companies such as Nokia offer several development tools that can help you test your content. By downloading the Nokia Mobile Internet Toolkit and the Series 40 Developer Platform SDK, you can run a test environment such as the one shown below.

First, enter the URL above to access the content.

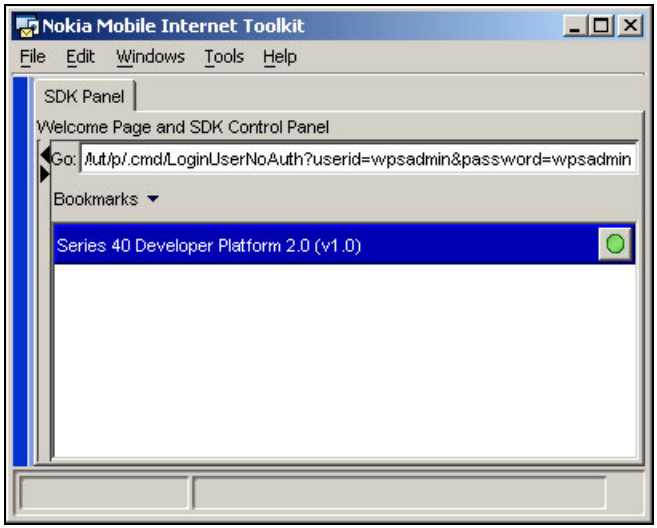


Figure 10-14 Phone simulator, entering the URL

Once the portlet application loads, you can interact with it in the simulator as you would on the actual phone.

Figure 10-15 Sample application running in Nokia Series 40 Simulator

While navigating through your application, some tools allow you to see the data and source code that is generated. This is especially useful for locating errors in the source code and correcting markup language syntax errors where necessary.

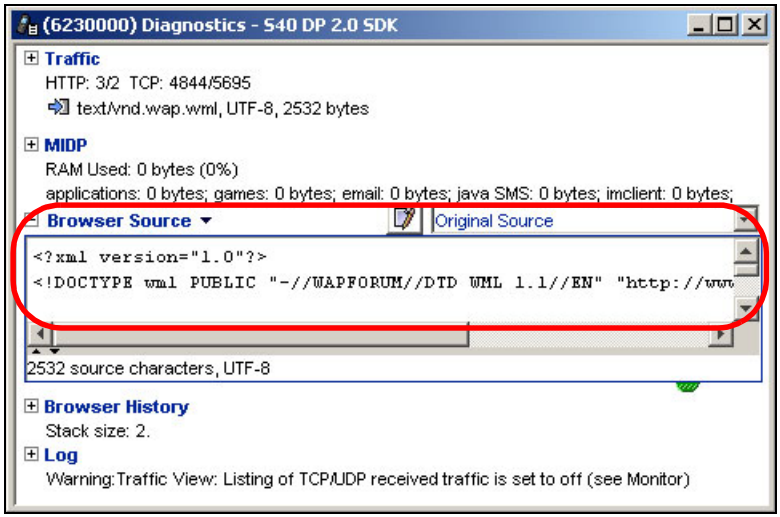


Figure 10-16 Diagnostic data

10.5 Summary

In this chapter, we have discussed various methods to display information on mobile devices. We then discussed the steps necessary to develop an application for ITSO Railways to make train timetable information available to its customers.

Using a mobile portal solution is an easy and effective way to make content available to users and customers on the go.

11

Chapter 11

Mobile Inventory Management with offline forms

The Mobile Inventory Management scenario for inventory stock tracking of used train supplies is used to describe the architectural approach for forms-based applications on mobile devices. Starting from the fictive business case of ITSO Railway, this example's use cases are created (see Chapter 5, "ITSO Railway sample overview" on page 87). An architectural overview diagram and a component model are developed based on these example use cases and the Pervasive runtime patterns from Chapter 3, "Runtime pattern" on page 35. Furthermore, necessary application development steps and the final setup of the new mobile extension of the inventory system (Mobile Supply Tracking System) are documented.

11.1 Overview

This section is a summary of the requirements extracted from ITSO Railway's inputs for the mobile extension of their inventory management system. This mobile extension will be called Mobile Supply Tracking System (MSTS).

The following diagram is the Runtime pattern we implemented in this particular scenario.

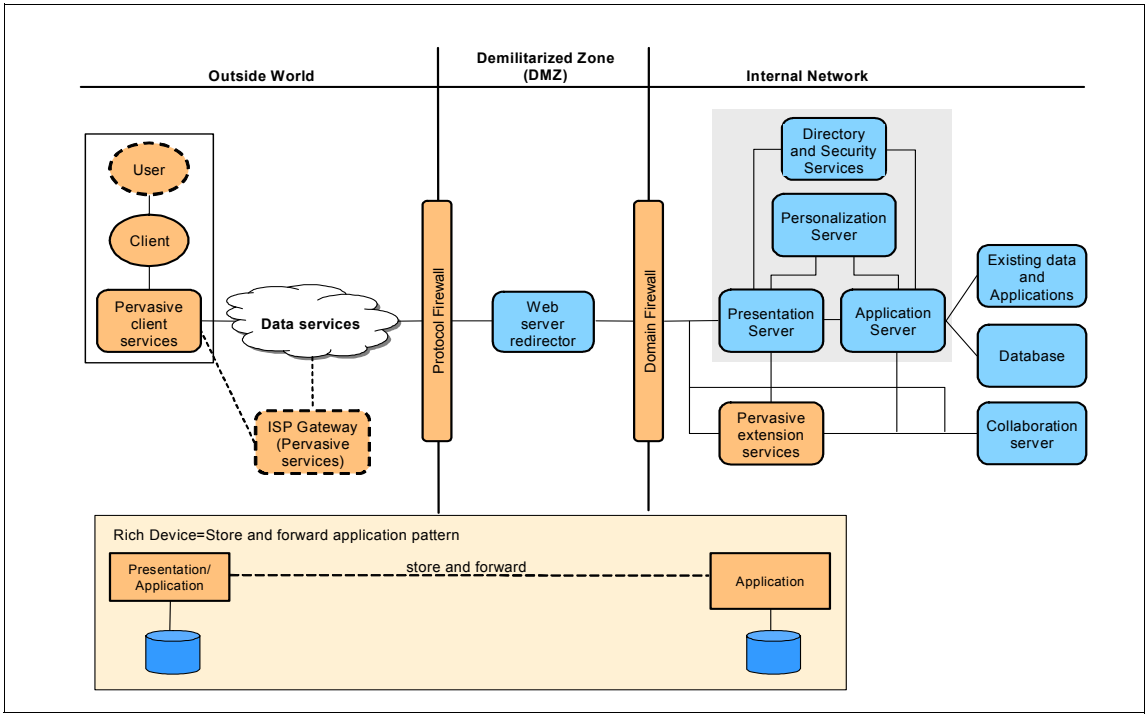


Figure 11-1 Runtime pattern for the offline forms scenario

The runtime mapping with the products described in this chapter can be found in the following diagram.

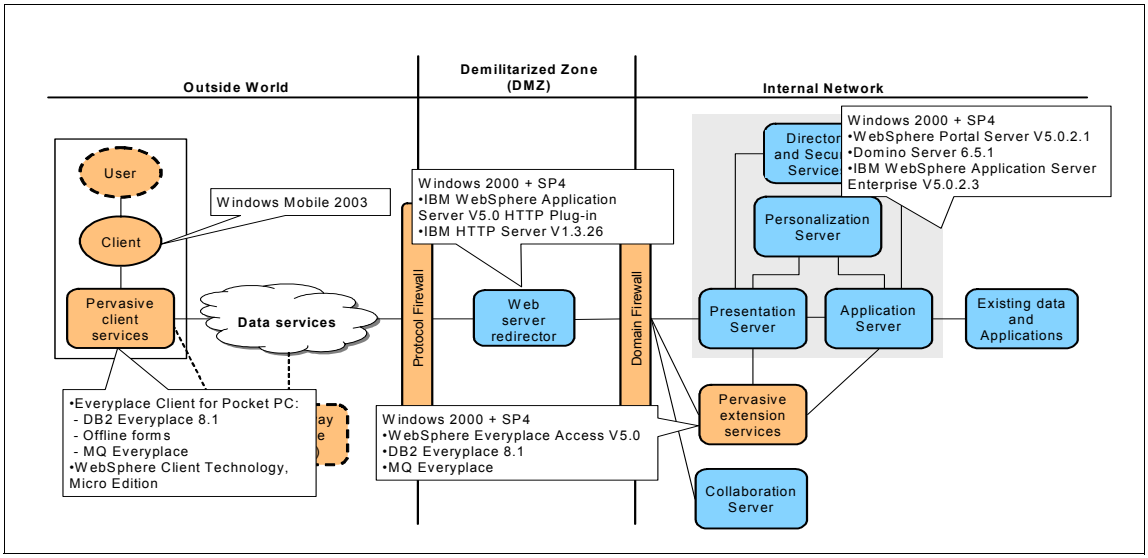


Figure 11-2 Rich Device=Store and forward::Product mapping=Windows

11.1.1 Customer requirements

The ITSO Railway company was created as a way to explain the general approaches for mobilizing applications through a series of scenarios. Examples for customer requirements are described and listed in Chapter 5, “ITSO Railway sample overview” on page 87.

The following business context for the inventory stock tracking of used train supplies was identified (see Figure 11-3 on page 214). Currently, the ITSO Railway delivery staff tracks train supplies on paper forms. At the end of the day, this paper form is delivered to the service station. The data entry clerks use these forms to enter the amount of used supplies into the inventory system. This allows ITSO Railways to track the amount of supplies leaving the existing stock in the inventory system. Supplies running out of stock can be automatically identified and re-ordered (for example, by the ERP system).

In order to improve the existing inventory stock tracking business process, it is planned to equip the delivery staff with PDAs. This implies that the existing inventory system must be extended with a Mobile Supply Tracking System, which consists of a mobile extension to the inventory system (server part of the MSTS) and a Mobile Supply Tracking client application (client part of the mobile extension running on the PDA). By doing this, a faster, more stable process will be achieved.

The following sections list the functional and non-functional requirements to define the baseline by which the new Mobile Supply Tracking System must be designed to extend the existing inventory system. A sample use case model with the most important use cases is also created.

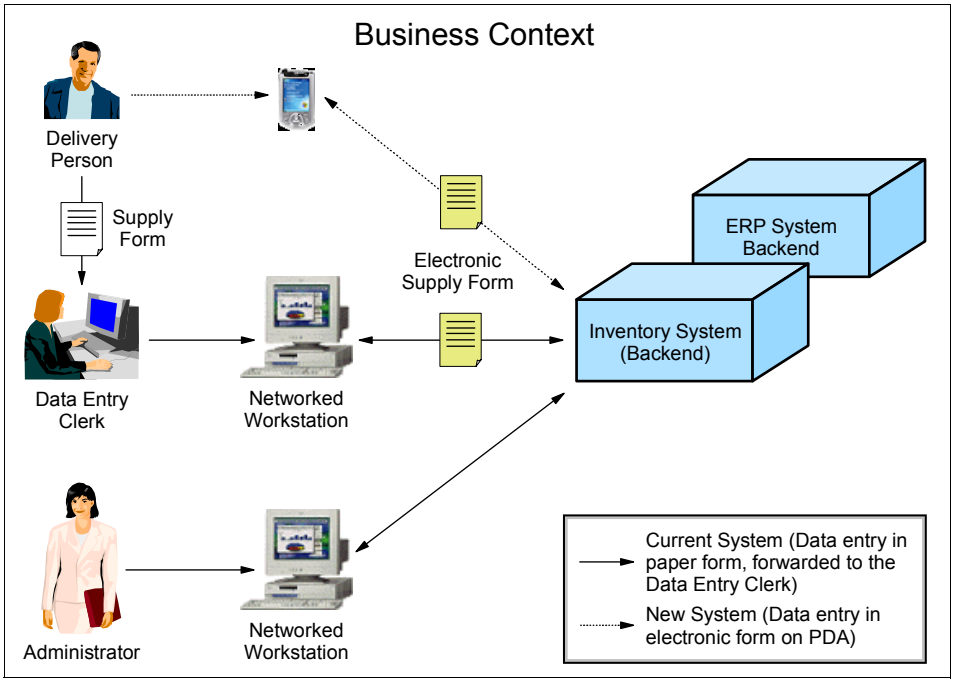


Figure 11-3 ITSO Railway Business Context for inventory stock tracking process of used train supplies

11.1.2 Functional requirements and use case model

The functional requirements of the ITSO Railway Mobile Supply Tracking System example are extracted from the customer requirements. They provide the main input for the use case model. The use case model work product is used to describe the functional requirements of the system under development. The model uses graphical symbols and text to specify how users in specific roles will use the system (that is, use cases). The textual descriptions define the use cases from a user's point of view. They do not describe how the system works internally or its internal structure or mechanisms.

Actor names, actor descriptions, use case numbers, use case names, use case business events, use case overviews, and the communication associations between the actors and the use cases provide an overview of the functional

requirements. The other constructs of the model document the expected usage, user interactions, and behaviors of the system in different styles and depth.

We simplified the use case model for the ITSO Railway Mobile Supply Tracking System example and reduced the use case description to the following constructs:

- Actors (name, description, status, superclass, subclass, and associations)
- Use cases (number, subject area, business event, name, overview, preconditions, description, associations, inputs, outputs, usability index, and notes)
- Communication Association diagram

The presented project approach for the ITSO Railway Mobile Supply Tracking System example is mainly based on the IBM Global Services Method recommendation.

Actors

The following actors are of interest for the new Mobile Supply Tracking System:

- Delivery person
- Administrator

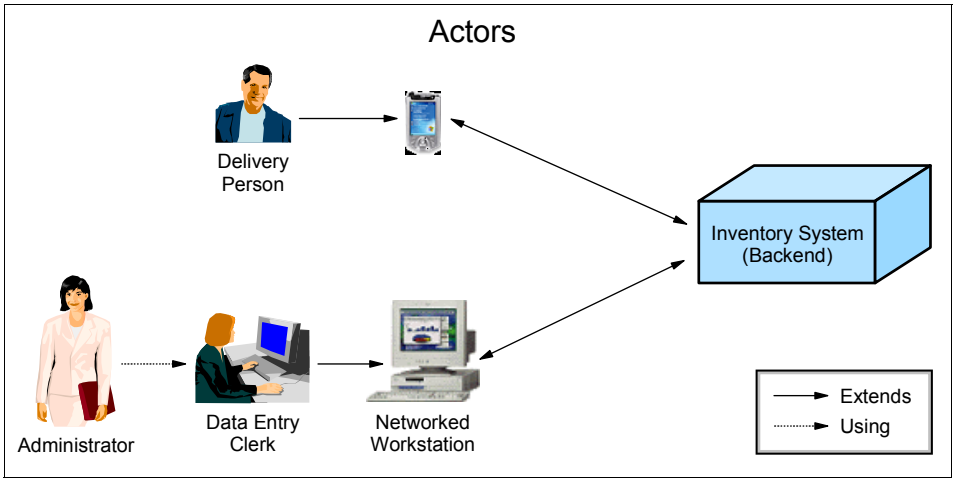


Figure 11-4 ITSO Railway Mobile Supply Tracking System

The delivery person is the main actor in this example. At the beginning of the working shift the user picks up his fully charged PDA containing the Mobile Supply Tracking client application (the electronic Train Schedule page and the Supply Consumption Record forms). The Train Schedule page contains the information of trains needing to be equipped with the new supplies during their

stop at the station. The delivery person uses this schedule to check the supplies of the trains assigned to him. If during these checks supplies needed to be restocked, the consumed amount must be entered onto the electronic forms (Supply Consumption Record). At the end of the working shift, the electronic forms are synchronized back to the Mobile Supply Tracking System server where these forms can be further processed. The PDA is placed in its networked cradle for this synchronization and is charged at the same time.

The administrator is a general system administrator who is responsible for the deployment, maintenance, and consistent operation of the inventory system and its supply tracking application for the delivery staff. This person is a user administrator in the sense that he provides the delivery staff with access to this application.

The following tables describe the actors of the ITSO Railway Mobile Supply Tracking System example (see Table 11-1 and Table 11-2).

Table 11-1 Actor delivery person

Actors name	**Delivery person**
Brief description	The delivery person equips the train with supplies to replace those used up during a train journey (for example, hand towels, soap). He must record the amount of used supplies for the inventory system.
Status	Primary.
Relationship	
Inheritance	Subclass: Administrator.
	Superclass: None.
Association to use cases	Start application, input data and submit form, sync forms.

Table 11-2 Actor Administrator

Actors name	**Administrator**
Brief description	The administrator of the inventory system maintains the system, deploys the Mobile Supply Tracking client application (electronic train schedule, supply consumption record, and synchronization), and makes it available to the delivery person's PDA.
Status	Secondary.

Relationship	
Inheritance	Subclass: None.
	Superclass: Delivery person.
Association to use cases	Deploy and manage application.

Use cases

Use cases describe functional requirements. They are used as input for the system and application design. Furthermore, use cases are the key proof points for the solution delivered to the customer. The final solution test runs utilize the use cases to which the customer and the system provider agreed to at the project's start.

The following use cases were considered in the ITSO Railway inventory system example:

- Start application
- Input data and submit form
- Synchronize forms
- Deploy and manage application

Table 11-3 on page 218 through Table 11-6 on page 222 contain more detailed descriptions.

Use case diagram

Actors and use cases communicate to execute the interaction described in the use case. This communication is modeled as a communication-association, shown in Figure 11-5 on page 218. The direction of the arrow shows the direction in which the communication is initiated. In addition, Figure 11-5 on page 218 includes the application workflow for the delivery person as well as the relationship between the use cases.

The administrator makes the Mobile Supply Tracking System application available to the appropriate delivery people. Each delivery person starts the client application on the PDA at the beginning of their shift (working day). The delivery people enter the amount of used supplies onto the Supply Consumption Record forms in their PDAs and submit them locally (the submitted forms are stored locally in the PDA in a file or database). At the end of the shift, the locally submitted forms are transmitted through the Mobile Supply Tracking System server to the inventory system through synchronization.

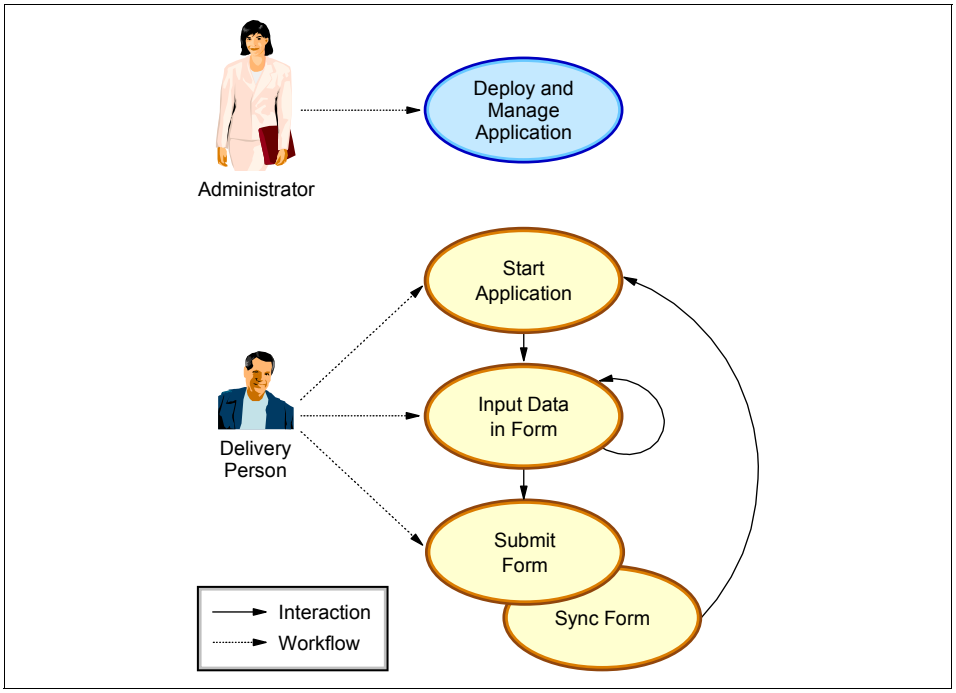


Figure 11-5 Mobile Supply Tracking System application use cases

The following tables describe use cases for the new ITSO Railway Mobile Supply Tracking System in detail.

Table 11-3 Use case 1: Start application

Use case #1 **Subject area**	**Start application** **Data input**
Business event	Delivery person starts his working day and equips the first train car with additional supplies.
Actors	Delivery person.
Use case overview	Delivery person starts the Mobile Supply Tracking client application on the PDA, which is used to record the amount of used supplies.
Preconditions	- PDA is fully charged. - Mobile Supply Tracking client application successfully deployed on PDA.

Termination outcome	Condition effecting termination outcome
1) Train Supply Consumption form loaded	- PDA password entered correctly. - Authentication with Mobile Supply Tracking client application was successful.
2) Supply Consumption form not loaded	- PDA cannot be switched on (not charged). - PDA password incorrect. - Authentication with application was not successful.
Use case description	Delivery person starts the shift with picking up the PDA from the service station (fully charged and synchronized). He switches on the PDA, enters the PDA password, and starts the Mobile Supply Tracking client application (authentication necessary): - Delivery person opens up train schedule page to find out which train on which platform must be checked. - First supply consumption record form is loaded.
Use case association	- Relies on deploy and manage application use case. - Used by Input Data and Submit Form and Sync Forms use cases. - Needs sync forms use case.
Inputs summary	- PDA password and application authentication data.
Output summary	-
Usability index	-
Use case notes	Use case provides the device and application access security features for the Mobile Supply Tracking client application.

Table 11-4 Use case 2: Input data and submit form

Use case #2 **Subject area**	**Input data and submit form** **Data input**
Business event	Delivery person equips and refills trains with the supplies needed.
Actors	Delivery person.
Use case overview	Delivery person enters amount of used supplies into the Supply Consumption Record form.
Preconditions	Start application on PDA (Mobile Supply Tracking client application).
Termination outcome	**Condition effecting termination outcome**

1) Validation successful and form submitted	- Data entered in correct format. Validation function returns successful when **Submit** button is clicked. - Form is locally stored for synchronization.
2) Validation unsuccessful, form not submitted	- Data entered in wrong format. Validation function returns failure when **Submit** button is clicked. - Form cannot be submitted and locally stored.
Use case description	Delivery person equips the train car with additional supplies according to train schedule form. He opens up the Supply Consumption Record form and enters the train number and the amount of used supplies. The form on the PDA must provide the supply's name and a data input field for the used amount of the supply. The amount must be entered as a numeric number. The **Submit** button invokes a validation function for the amount field. If the validation is successful, the form is stored locally. It is ready to be sent to the inventory system in the backend during Mobile Supply Tracking System synchronization.
Use case association	- Relies on Deploy and Manage Application use case. - Used by Sync Forms use case. - Needs Start Application use case.
Inputs summary	Supply consumption record form: - Train number (numeric, 1 letter + 5 digits, fixed label preferred) - Item type (fixed label) and corresponding used amount of this item (numeric, min. 1 digit): soap, hand towels, paper rolls, tissues
Output summary	- Supply consumption record form (all validated input data ready to be sent to the backend).
Usability index	Data input must be easy to understand and to handle for user.
Use case notes	This is the most important use case for the end user (delivery person). It is tightly coupled with the Sync Forms use case.

Table 11-5 Use case 3: Sync forms

Use case #3 **Subject area**	**Sync forms** **Data input**
Business event	Delivery person finishes supplying scheduled trains (end of a working shift).
Actors	Delivery person.

Use case overview	Delivery person synchronizes the submitted supply consumption records with the backend inventory system.
Preconditions	- Authentication with Mobile Supply Tracking client application was successful (see use case #1) - PDA placed in network-attached cradle
Termination outcome	**Condition effecting termination outcome**
1) Train Supply Consumption Record forms synchronized	- Mobile Supply Tracking System (synchronization middleware), inventory system, and network connection are available and functioning. - Locally provided username and password are correct.
2) Forms not synchronized	- Mobile Supply Tracking System (synchronization middleware), inventory system, and network connection are not available and/or functioning. - Locally provided username and password are incorrect.
Use case description	Delivery person finishes the day and all filled-in forms were successfully submitted (stored locally). Delivery person puts PDA in network-attached cradle in service station and synchronizes forms (inventory data is sent to the backend). Mobile Supply Tracking client application is stopped and PDA is left in the cradle for recharging.
Use case association	- Relies on Deploy and Manage Application use case. - Used by Start Application use case. - Needs Input Data and Submit Form use case for synchronization.
Inputs Summary	Filled-in supply consumption record forms.
Output Summary	Data entered in supply consumption record forms is transmitted to inventory system.
Usability Index	Synchronization should be possible by pressing a button.
Use Case Notes	This case is the most important use case from the inventory systems point of view. All data entered by the delivery person must be transmitted to the inventory system without errors (relies on a robust, scalable middleware for forms synchronization).

Table 11-6 Use case 4: Deploy and manage application

Use case #4 **Subject area**	**Deploy and manage application** **System maintenance, user administration**
Business event	New Mobile Supply Tracking client application for delivery staff is available and/or delivery staff members have changed.
Actors	Administrator.
Use case overview	Administrator deploys Mobile Supply Tracking client application for delivery staff and makes it available to appropriate staff members.
Preconditions	Mobile Supply Tracking client application was developed and is available. Application users and their access rights have been defined.
Termination outcome	Condition effecting termination outcome.
1) Mobile Supply Tracking client application successfully deployed and users are assigned	- Authentication successful. - Mobile Supply Tracking client application was installed successfully. - User access rights were assigned to delivery staff members. - Forms have been made available as offline content.
2) Application not deployed and is not available to users	- Authentication not successful. - Mobile Supply Tracking client application could not be installed. - User access rights were not assigned to delivery staff members. - Train Schedule page and Supply Consumption Record form are not made available as offline content.
Use case description	Administrator logs on to administration server, installs Mobile Supply Tracking System and starts it. Administrator assigns user access rights to delivery staff members. Makes Mobile Supply Tracking client application available for delivery staff members.
Use case association	This use case is a precondition for all other use cases.
Inputs summary	Mobile Supply Tracking client application has to be provided.
Output summary	Mobile Supply Tracking client application is available.
Usability index	User administration should be conform to existing user management.

Use case notes	Mobile Supply Tracking client application will not be accessible for delivery staff members if this use case is not successful.

11.1.3 Non-functional requirements

This section lists non-functional requirements that apply mostly to applications on mobile devices.

The non-functional requirements for a business system address those aspects of the system that do not directly effect the functionality of the system as seen by the users. Nevertheless, they can have a profound effect on how that business system is accepted by both the users and the people responsible for supporting that system.

The non-functional aspects of a business system cover a broad range of topics. The major non-functional themes identified by the AED Technical Council are listed below:

- Performance
- Scalability
- Availability (including recoverability and reliability)
- Maintainability (including flexibility and portability)
- Security
- Manageability
- Environmental (including safety)
- System usability
- Data integrity (including currency, locality of updating, data retention)

In the ITSO Railway Mobile Supply Tracking System example, we are looking at these non-functional requirements from a more generalized view. Many of those are addressed already through the choice of WebSphere Everyplace Access as the Mobile Supply Tracking System middleware for our forms-based Mobile Supply Tracking client application example. The WebSphere Portal Server running on WebSphere Application Server covers scalability, availability, security, and data integrity.

Performance

Performance is considered to be addressed through the offline forms capability of the Mobile Supply Tracking client application itself. Since the Supply Consumption Record form can be filled-in without a backend connection, the performance of the backend system does not influence the performance experienced by the user. System performance will only be noticed during the synchronization process of the filled form with the backend system.

The following response times of the Mobile Supply Tracking client application and system are required.

Table 11-7 Target end-to-end and local response time

Target end-to-end and local response time		
Frequency of usage	High frequency (5–10 times/per day)	Low frequency (1–5 times/per day)
Locally entered data in form	2 sec.	1 sec.
End-to-end synchronization of form	Not applicable	300 sec.

Availability

Availability is frequently an important Service Level Requirement. The table below lists the specification of the desired availability. Availability requirements typically vary by use case and component; each row represents a collection of use cases in the form of a component with common availability requirements.

Table 11-8 Availability requirements

Component view	**Availability requirement**	**impact of failure**
Handheld device (PDA)	During working time of delivery personal (7 days a week, 8 hours a day)	High - Used supplies would not be recorded with PDA - Work around: Old paper forms could be used, data must be entered manually
Mobile Supply Tracking client application (supply consumption record, train schedule)	During working time of delivery personal (7 days a week, 8 hours a day)	High - Used supplies would not be recorded with PDA - Work around: Old paper forms could be used, data must be entered manually
Mobile Supply Tracking System server (Synchronization)	Fully available at the end of each shift of the delivery staff (7 days a week, minimum 6:00-7:00, 14:00-15:00, 22:00-23:00)	High, forms can be synchronized at the beginning of the next shift, trolley with supplies might not be adequately filled

Maintainability

All application components (Mobile Supply Tracking client application, Mobile Supply Tracking System server and backend) must be independent of each

other from a development and maintenance point of view. The following components can be designed, developed, tested, and maintained independently of each other:

- Middleware components (client and server software for offline forms applications and synchronization)
- Offline forms application (supply consumption record and train schedule forms)
- Inventory system (backend)

Security

The PDA should be protected with a power-on password. The delivery person must also authenticate against the local Mobile Supply Tracking client application.

Manageability

Manageability mainly concentrates on the application management. It contains the way the forms-based application is managed from an administrators point of view.

The Mobile Supply Tracking client application must be remotely deployable, configurable, and manageable.

System constraints

System constraints are mainly defined by the used handheld device (PDA) and the size of the Mobile Supply Tracking client application.

A chosen PDA defines capabilities for the application, such as screen size, resolution, browser capabilities, battery lifetime, and weight. The browser capabilities heavily influence the design of the offline forms application (Supply Consumption Record). It is beyond the scope of this book to investigate all the different types of devices and their constraints.

System usability

We are covering the usability of the Mobile Supply Tracking System example from the user's perspective. The Mobile Supply Tracking client application including the Supply Consumption Record form must be easily understood and easy to navigate for the user.

Data integrity

Data entered in the Supply Consumption Record form must be stabley transferred to the backend system. The WebSphere Everyplace Access Offline

forms Synchronization service as part of the Mobile Supply Tracking System server takes care of this requirement.

We will look at these non-functional requirements and how they are fulfilled in the following chapters in more detail.

11.1.4 Solution approach

The ITSO Railway customer requirements were the input for the example solution of the Mobile Supply Tracking System described in the following sections. The following approach was taken to get to a forms-based solution.

- Identification of the problem that has to be solved by considering the requirements and use cases to derive an architectural overview.
- Choice of suitable technology and Runtime pattern to create a component model for the solution.
- Application software development. Choice of a client programming model. Adaptation of this model to the customer use cases, identification of programming tools, and choice of software technology.
- Deployment of solution and roll-out of application to users. Installation and configuration of solution on the pervasive server as well as on the handheld device must be performed. This includes setup of user access configuration to the application as well as software distribution to the handheld device.
- Maintenance and operation of the solution. Changes and updates of handheld applications as well as middleware components must be centrally managed.

The solution development and deployment on the server side (Mobile Supply Tracking System server) are described in the following chapters. The roll-out and maintenance of the Mobile Supply Tracking client application is covered in more detail in Chapter 16, “Maintaining mobile devices” on page 393.

11.2 Architectural overview

Regarding the ITSO Railway customer requirements (see 11.1, “Overview” on page 212), the Mobile Supply Tracking client application for the delivery person must be able to work without a constant connection to the Mobile Supply Tracking System sever and Inventory system. The used supplies must be recorded by entering the consumed amount on the handheld device. This implies that a local repository must be present on the device. Data stored in this repository must be transmitted to the inventory system at a certain time via some middleware.

With this conclusion in mind, the Pervasive runtime pattern for Device-based solutions can be applied (please refer to “Rich Device=Store and forward::Runtime pattern” on page 44). The high-level solution architecture in Figure 11-6 on page 228 was derived from the Runtime pattern in Figure 3-5 on page 45. This architecture for the replacement of the paper forms for tracking used train supplies consists of three parts:

- Device running the Mobile Supply Tracking client application

  It instantiates the client including the pervasive client services node of the applied Runtime pattern in Figure 3-5 on page 45. The device is the physical replacement of the paper form through a handheld device (PDA). The Mobile Supply Tracking client application running on the PDA provides an electronic form with nearly the same look and feel as the paper form. The user (delivery person) will enter the amount of the used supplies into this electronic form (Supply Consumption Record). The entered data is stored locally (Offline Content Cache in Figure 11-6 on page 228). An online connection to the inventory system in the backend was not considered as appropriate because it would have implied an always available radio network connection. Unfortunately, the coverage of radio networks cannot be guaranteed within a train passenger car. It must be possible to enter data in disconnected mode.

- Pervasive server (mobile supply tracking system server)

  The pervasive server instantiates the directory and security services, personalization server, presentation server, and application server nodes including the pervasive extension services node of the Runtime pattern in Figure 3-5 on page 45. It acts as middleware between the device side and the inventory system in the backend (enterprise server). It is responsible for the consistent data transmission from the device to the inventory system (synchronization of cache content).

- Enterprise server

  The enterprise server represents the runtime environment of the inventory system (instance of existing data and applications node of Runtime pattern in Figure 3-5 on page 45). Data entry clerks have direct access to this system to enter all data from the paper supply consumption record. This data is stored and maintained in a local data storage. Other applications, for example, Enterprise Resource Planning applications, could track all changes and trigger further activities (for example, order low-running supplies).

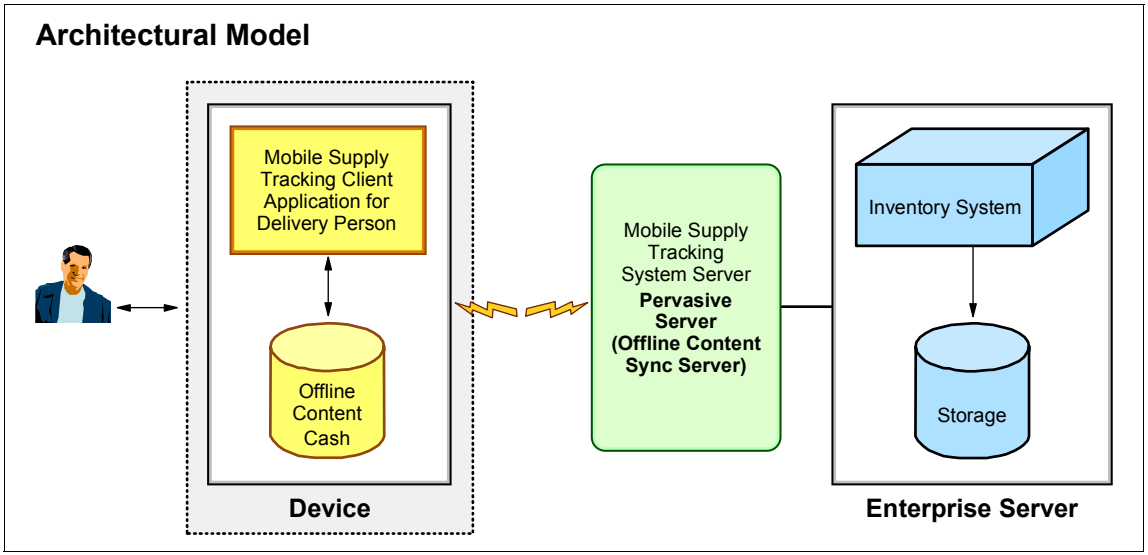


Figure 11-6 Architectural overview diagram

The pervasive server accesses the inventory system in the same way as the data entry clerks do. In doing this, it extends the inventory system access to mobile clients (delivery staff). Data is entered offline in the electronic form (Supply Consumption Record) on the handheld device and transmitted to the inventory system later on. For this data transmission, the handheld device is placed in a cradle with a network connection to the pervasive server (for example, directly through Ethernet or a network-attached service PC).

11.3 System design overview

This chapter discusses the chosen solution design of the ITSO Railway Mobile Supply Tracking System solution for the delivery staff. After a more detailed architectural overview, examples for a component and an object model are also given.

11.3.1 General considerations for intermittently connected applications

The following three decision points were addressed in the ITSO Railway inventory system example:

- Decision about handheld device

Consider customer requirements (usability, security, weight, battery lifetime, screen size, etc.). This decision leads to a device with certain computing capabilities (operating system, Web browser, RAM/ROM size, CPU performance, etc.).

- What intelligence needs to be implemented in the application on the device? Which business-critical processes must be covered?
  - These questions also influence the handheld decision.
  - Some applications allow having more computing power on the server side rather then on the client side (also see Figure 11-12 on page 238).
  - The amount of data entered in the handheld device and the amount transmitted back and forth between the handheld and the server must be considered.
- Which security level must be addressed on the device and through the application?
  - This influences the handheld hardware and software choices (password and encryption capabilities).
  - Security on an application level could be managed by the pervasive server. How is this managed over insecure networks?

Two approaches for offline capable applications on handheld devices were introduced in 3.2.5, “Rich Device=Store and forward::Runtime pattern” on page 44.

- Complex applications on handheld device with appropriate computing power (with more client-side application logic in Figure 11-12 on page 238).

  This includes powerful local data storage, such as a relational database (such as IBM DB2e). This database would contain data copies of the backend database. Data is kept consistent between backend database and device database by an advanced synchronization mechanism. Comprehensive application logic is implemented and managed by an application framework on the device, such as OSGi. Secure transaction and data exchange between device and backend can be handled through an extension of the backend messaging system (such as IBM WebSphere MQ).

  Refer to Chapter 13, “Using Workplace Client Technology, Micro Edition” on page 283, where such a solution approach is described in detail.
- Simple offline forms actions-based applications for data entry (with more server-side application logic in Figure 11-12 on page 238)

  What does it mean for form actions to be performed offline? Form actions can either be performed server-side or client-side. Form actions that are performed client-side can be referred to as being performed offline. The client

does not have to be connected to the server for offline actions to be performed.

Simple applications for data input without needing a database for storing and retrieving data in offline mode can be addressed by an offline forms action approach. The handheld device can be less powerful than the one needed for the first solution since only HTML forms are used to collect data on the device. The handheld device must provide a forms-capable Web browser. It also needs the possibility to store the filled-in and submitted forms locally. A pervasive client or agent must be present to synchronize the local cache content (locally submitted forms) with the originator of the form. This can be applied to many low-performance mobile devices with Web browsers.

In addition, validations for field entries are a good choice to be performed offline, because the validation function takes a small amount of computing power and is performed without exchanging information with the database. Offline validations utilize a light-weight JavaScript validation library file that is cached on the device. For instance, a user connects to the Web and downloads a form that contains offline field validation capability. When the user downloads the form, the validation library is cached on the device. The user can then disconnect from the Web and continue working with the form, entering values into fields and having those values validated offline (for more information about form field validations, refer to 11.4.2, “Development of forms-based applications for mobile devices” on page 237.)

The first approach is feasible for the relatively complex applications (see the ticketing application from Chapter 13, “Using Workplace Client Technology, Micro Edition” on page 283). The second approach is suitable for the simple ITSO Railway Mobile Supply Tracking System for the delivery staff. The technology used is also referred to as mobile access using offline forms actions. Details about the different client programming models can also be found in Chapter 6, “Pervasive application types” on page 101.

11.3.2 Mobile Supply Tracking System solution outline

Taking into account the ITSO Railway customer requirements and use cases as well as considering the two possible approaches (see 11.3.1, “General considerations for intermittently connected applications” on page 228), a solution based on offline forms is an appropriate solution option for the Mobile Supply Tracking System. Figure 11-7 on page 231 shows a detailed architectural overview based on WebSphere Everyplace Access.

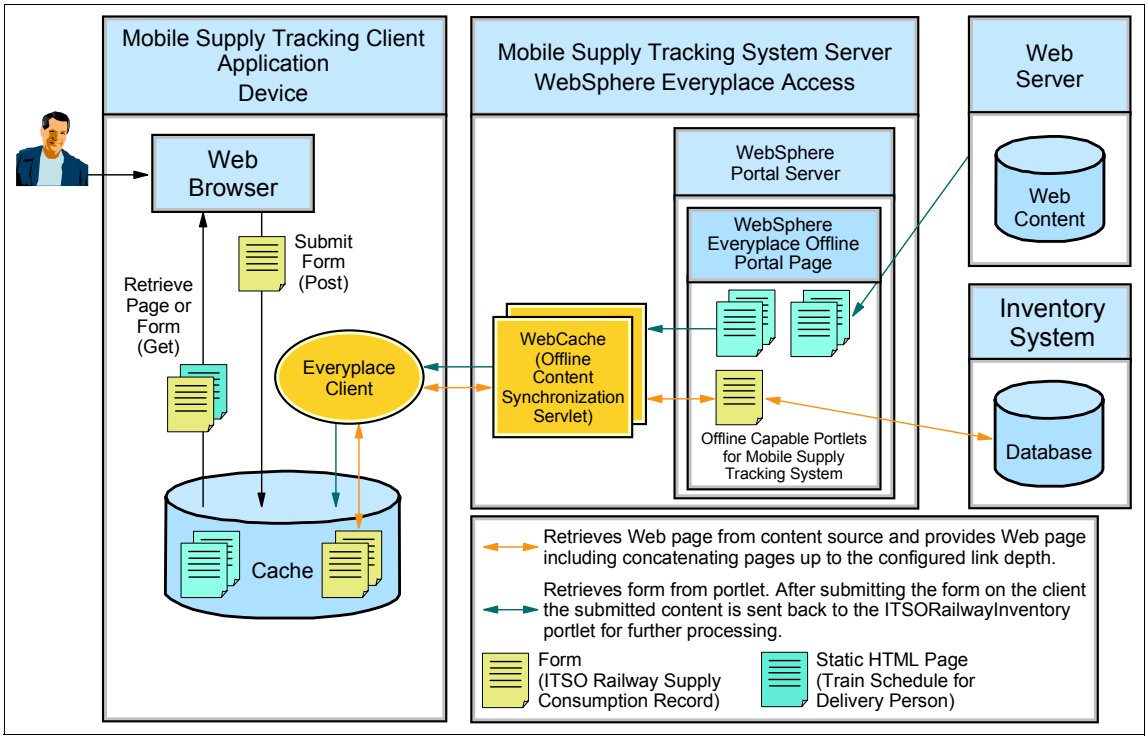


Figure 11-7 ITSO Railway Mobile Supply Tracking System solution outline using WebSphere Everyplace Access

There are three major blocks involved in this model (a 3-tier architecture):

- Inventory system and Web server

  The inventory system is the legacy application that stores the amount of used supplies in a relational database. The database content is maintained through a Web interface.

  The Web server stands for any HTML content serving source in the ITSO Railway network. In our example, it contains the Web page with the train schedule and the trains that have to be supplied by the delivery staff (daily schedules for all delivery people).

- WebSphere Everyplace Access with WebSphere Portal Server (Mobile Supply Tracking System Server)

  This block is the middleware tier. It contains WebSphere Everyplace Access including the WebSphere Portal Server. Portlets can display any content from the Web Server. Forms-based portlets, such as the ITSORailwayInventory portlet, are used to access the inventory system. They contain the ITSORailwayInventory application with the Supply Consumption Record form,

which is made available to the mobile workers (delivery staff). WebSphere Everyplace Access simply manages this mobile extension by retrieving the Web content from the Web Server and the forms on behalf of the user.

11. Device (Mobile Supply Tracking client application)

    The Mobile Supply Tracking client application provides the human interface to the portal content provided by the portlets (HTML pages and forms). The Everyplace Client is the counterpart of the WEA server on the device. Together they assure that the portlet-generated HTML content and forms are locally stored on the device (train schedule HTML page and Supply Consumption Record form). To assure offline browsing of the HTML pages with a configured link depth, the URLs included in the chosen Web pages (placed on the offline page of the WebSphere Everyplace Access portal) are rewritten and pointed to the locally stored sub pages. The Web browser on the device can now retrieve the HTML pages and forms directly from the local cache. The Everyplace Client on the device and the WebCache on the WebSphere Everyplace Access server take care of the submitted ITSO Railway Supply Consumption Record forms.

11.3.3 Component model

The component model was derived from the solution outline in Example 11-7 on page 231. It uses existing features from WebSphere Everyplace Access:

- WebSphere Portal Server (existing component)

  The ITSO Railway Mobile Supply Tracking application was implemented based on portal technology. It follows the Model-View-Controller pattern (see ITSORailwayInventoryPortlet, ViewBean, SessionBean, and View JSP). The ITSO Railway Supply Consumption Record is a form contained in the ITSORailwayInventoryView JSPs. Two JSPs are necessary for the two supported device classes, one for PDA browsers supporting PDA markup and one for HTML browsers. Both browsers must support JavaScript to utilize the field validation function (also contained in the JSP code).

- WebSphere Everyplace Access Offline Forms Support

  The support of offline browsing and offline forms is achieved by the offline content feature shipped with WebSphere Everyplace Access (WebCache and the Everyplace client components). Portlets placed on the Offline Page in WebSphere Everyplace Access are made available for offline viewing on the PDA.

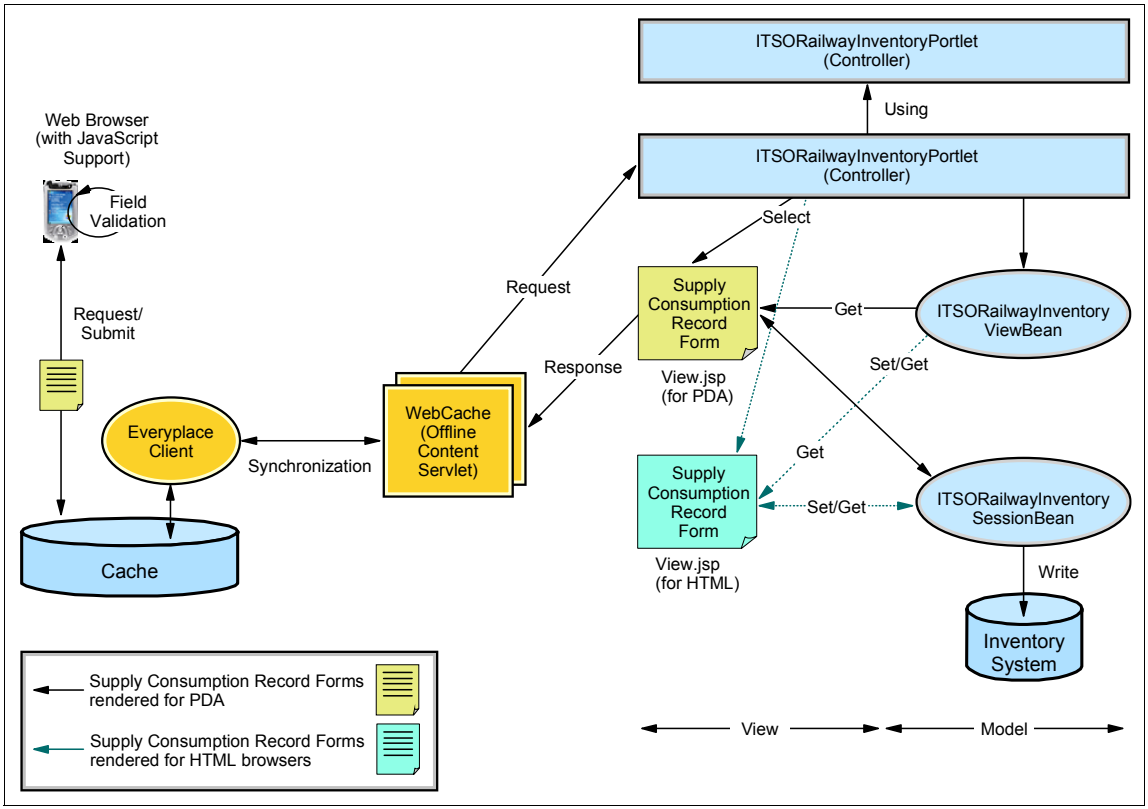


Figure 11-8 ITSO Railway Mobile Supply Tracking System components

The following tasks must be carried out for the Mobile Supply Tracking System implementation:

- Development effort: Portlet development for access to the inventory system (Mobile Supply Tracking System portal application: ITSORailwayInventory).
- Configuration effort: Developed portlets and Web pages must be configured as offline content on the WebSphere Everyplace Access server.
- Effort on the device: Installation and configuration of the Mobile Supply Tracking client application (Everyplace Client and the Offline Page/Forms applications).

11.3.4 Object model

The following classes are necessary for the Mobile Supply Tracking System portal application (ITSORailwayInventory):

- ITSORailwayInventoryPortlet

This portlet is the controller and receives the HTTP requests. It selects the appropriate View JSP depending on the user agent field. The PDA View JSP is for PDA browsers and the HTML View JSP is for full HTML browsers.

- ITSORailwayInventoryViewBean

  The view bean saves the action URI and the actions that are performed on the form for later evaluation.

- ITSORailwayInventorySessionBean

  The session bean captures the data entered in the form after the form was submitted. For our example, the database access to the ITSO Railway Inventory system is also included in here.

Furthermore, two JSPs, one for each supported browser type (PDA or HTML), must be available to provide the appropriate HTTP response to the requesting client. The JSPs create the content for displaying the form on the user's browser. They also capture the entered data (TRAIN_NO, SUPPLY_1, AMOUNT_1) and evaluate the data entered in Amount field (JavaScript function validate()) after clicking the **Submit** button on the form:

- ITSORailwayInventoryView JSP for HTML
- ITSORailwayInventoryView JSP for PDA

11.4 Application development

In this section, the implementation of a sample application which meets the ITSO Railway customer requirements is described. After a short introduction to the development tools and their usage, the process of the generation of the sample application for the Mobile Supply Tracking System is explained in more detail.

11.4.1 Introduction to WebSphere Everyplace Toolkit

The WebSphere Everyplace Toolkit helps developers write portlet applications for wireless and mobile clients. Details about the toolkit's functions have already been introduced in 8.2, "Everyplace Toolkit" on page 138.

Version 5.0.1of the toolkit, which was used for this book, contains support for the following markup languages:

- WML 1.1, 1.2 and 1.3
- XHTML Basic 1.0, XHTML Mobile Profile 1.0
- HTML 3.2 and 4.01
- i-mode HTML 1.0

The Everyplace Toolkit also supports various device emulators. Refer to the release notes for supported versions. We used the Palm Simulator V5.3, which is available for free download from PalmOne's Web site:

http://www.palmone.com

Version 5.0.1 of WebSphere Everyplace Toolkit was successfully installed with the following method (refer to InstallGuide.htm shipped with the toolkit):

1. Install WebSphere Studio Application Developer V5.1.1.

   Use a short path name (less than eight characters, for example, c:\wsad511 for the installation directory).

   Choose to install the **Integrated Test Environments for WebSphere Application Server 5.0.2**.

   Select **Examples for Eclipse Plug-in Development** under Additional Features (leave the other options unchecked; see Figure 11-9).

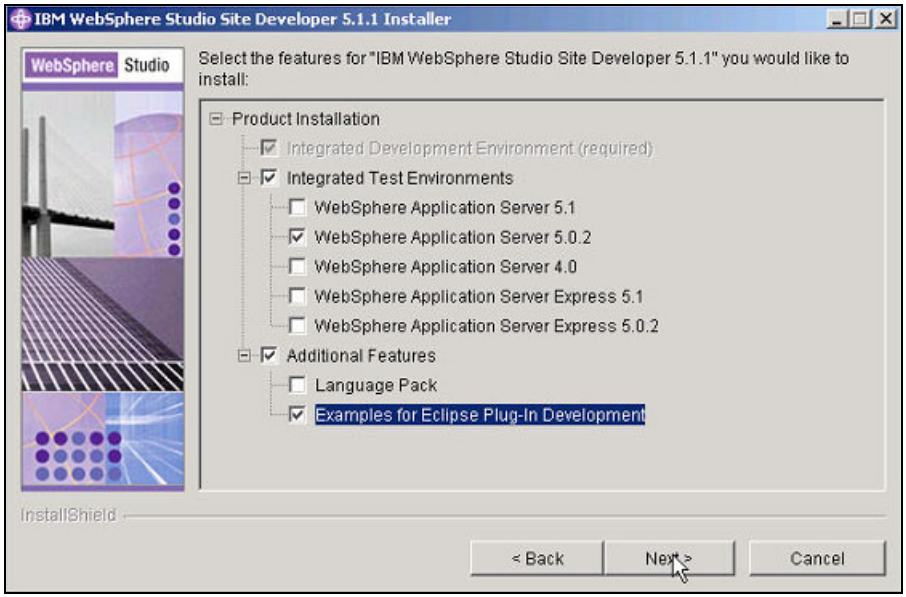


Figure 11-9 WebSphere Studio Installation Features

2. Install Interim Fix 002 for WebSphere Studio (WebSphereStudioInterimFix002.zip). Follow the installation instructions and do not change any default settings.
3. Install interim fixes for WebSphere Application Server (WASInterimFixes.zip).
4. Install **WebSphere Everyplace Toolkit** (EveryplaceToolkit501.exe).

a. The toolkit needs WebSphere Portal Toolkit. Choose to install it (see the screen shot in Figure 11-10), including the WebSphere Portal Test Environment (Figure 11-11).

b. Leave all directories to the suggested defaults.

c. Mounting of reference CDs into the local file system is recommended.

d. Choose **Typical Install**.

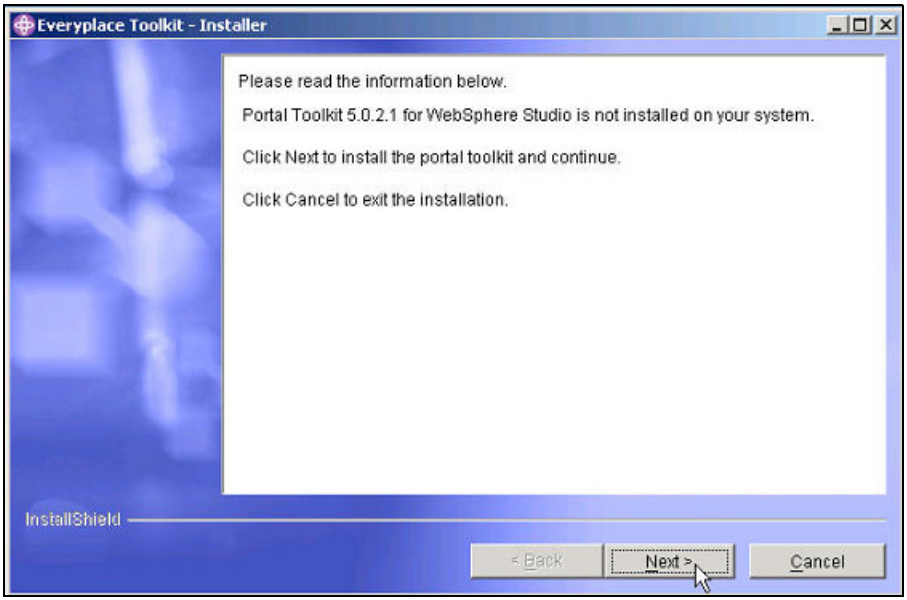


Figure 11-10 Everyplace Toolkit must install Portal Toolkit

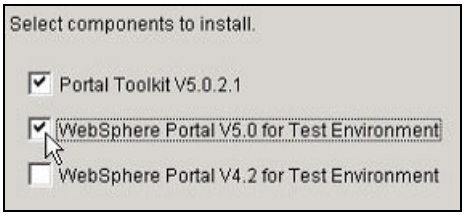


Figure 11-11 Everyplace Toolkit installation options

5. Install WebSphere Portal 5.0 FixPack 2 for local debugging option (updatePortalTestEnv502.bat).

6. Install WebSphere Portal 5.0.2 Cumulative Fix 1 (updatePortalTestEnv5021.bat).

7. Optionally install device emulators for PocketPC and Palm OS.

11.4.2 Development of forms-based applications for mobile devices

This section provides information about developing form based applications.

Forms and business logic

Forms-based applications offer a high potential of reusability. They are very flexible in terms of the amount of business and application logic, which they have to cover. The goal is to avoid changes in existing applications or keep changes as minimal as possible and reuse them. Figure 11-12 on page 238 shows the relationship between business logic and forms. It explains that the application logic can be flexibly deployed on either the client or the server side (enterprise application side). In the case that the mobile device is not capable of running complex client application logic, an application connector is used on the server side to support forms. The application connector is the piece of software that adopts existing enterprise applications to be forms compatible (see the first row in Figure 11-12 on page 238). If the mobile device is a more powerful device capable of running application logic, the application logic can be directly implemented on the device (client) to make the existing enterprise application accessible with forms (see the bottom row in Figure 11-12 on page 238). The middle row in Figure 11-12 on page 238 shows an in-between approach. Some logic is implemented on the client, and other logic is done by the application connector.

A suitable approach out of the three possibilities for an application is chosen depending on the device's capabilities. The main advantage is that in all three cases the form is the same. This makes it very flexible and portable. The same forms application can run on different devices with different capabilities. Depending on the customer and the device requirements, the application logic is implemented through the application connector or the client application or through both (according to Figure 11-12 on page 238).

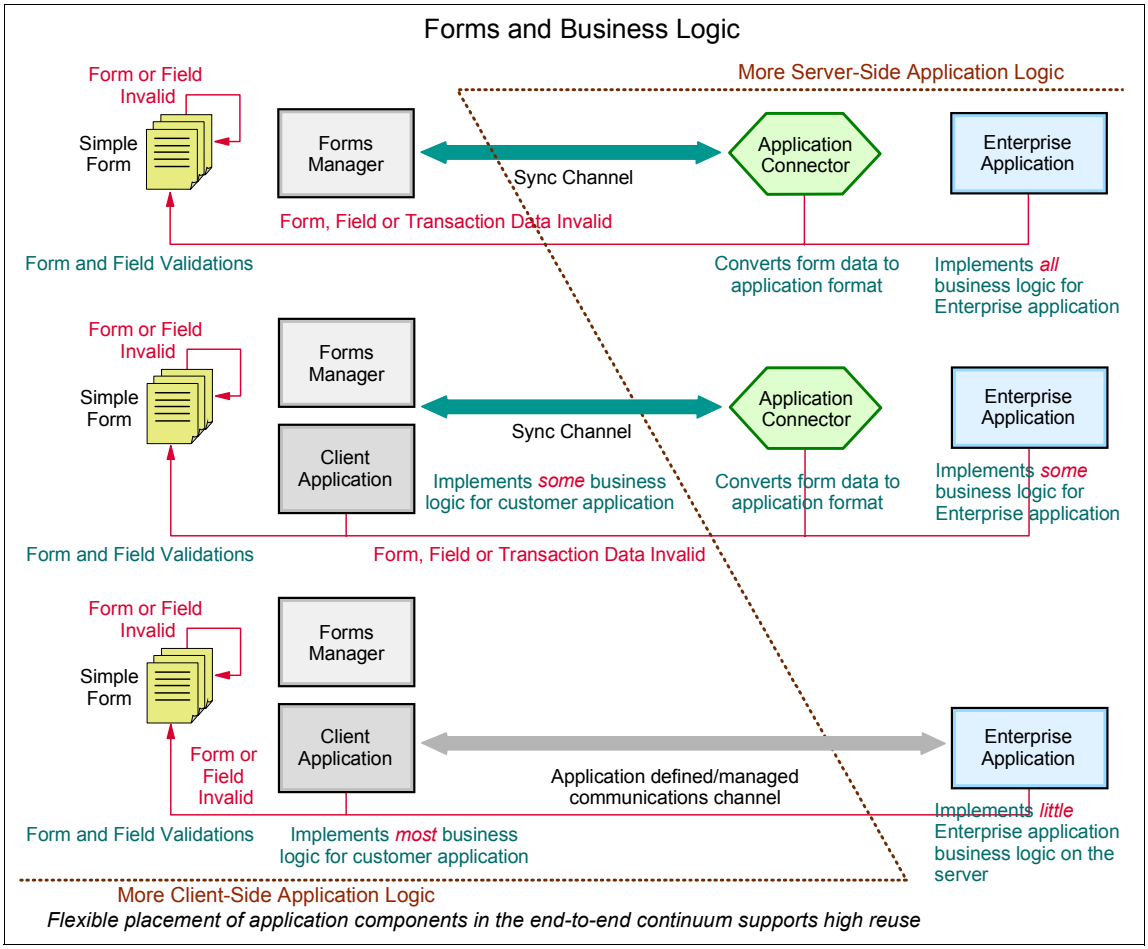


Figure 11-12 Relationship between forms and business logic

Offline forms for WebSphere Everyplace Access

The following steps describe step-by-step the creation of an example for the electronic Supply Consumption Record form used in the ITSO Railway Mobile Supply Tracking System (refer to “Introduction to WebSphere Everyplace Toolkit” on page 234 for installation details of WSAD):

1. Run the project wizard and create a new portlet project. Choose **Portlet Development** and **Portlet Application Project** and click **Next**.
2. Assign a name to project: `ITSORailwayInventory` (Figure 11-13 on page 239). Click **Next** using the default settings on the upcoming screens until the event handling screen appears.

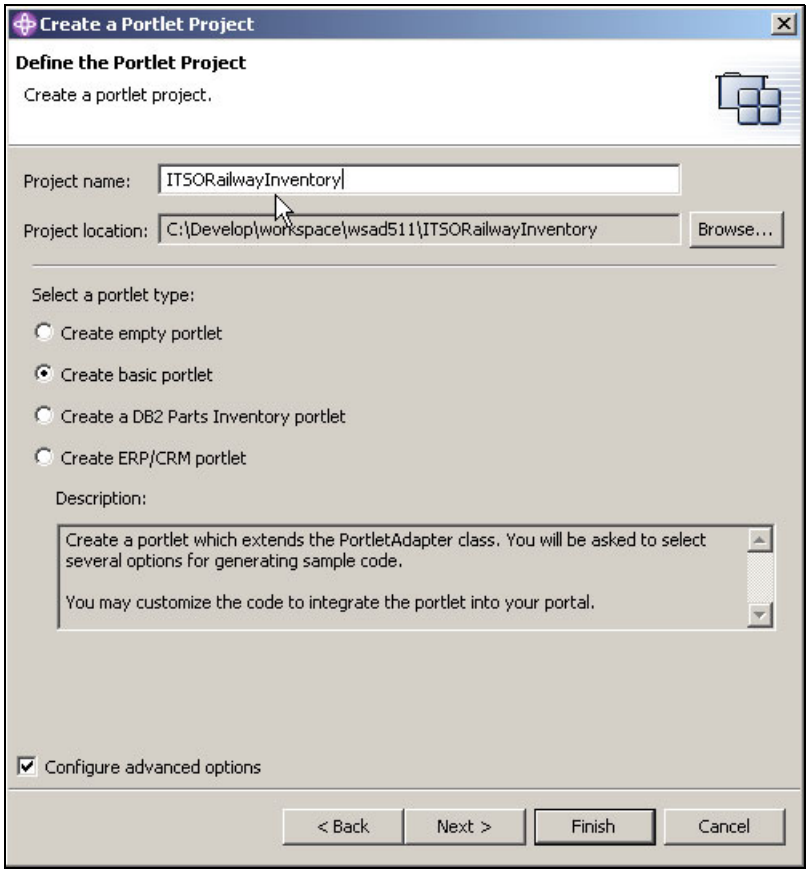


Figure 11-13 Enter project name in WebSphere Studio Portlet Project wizard

3. Add an action listener and forms example (see Figure 11-14 on page 240). Click **Next**.

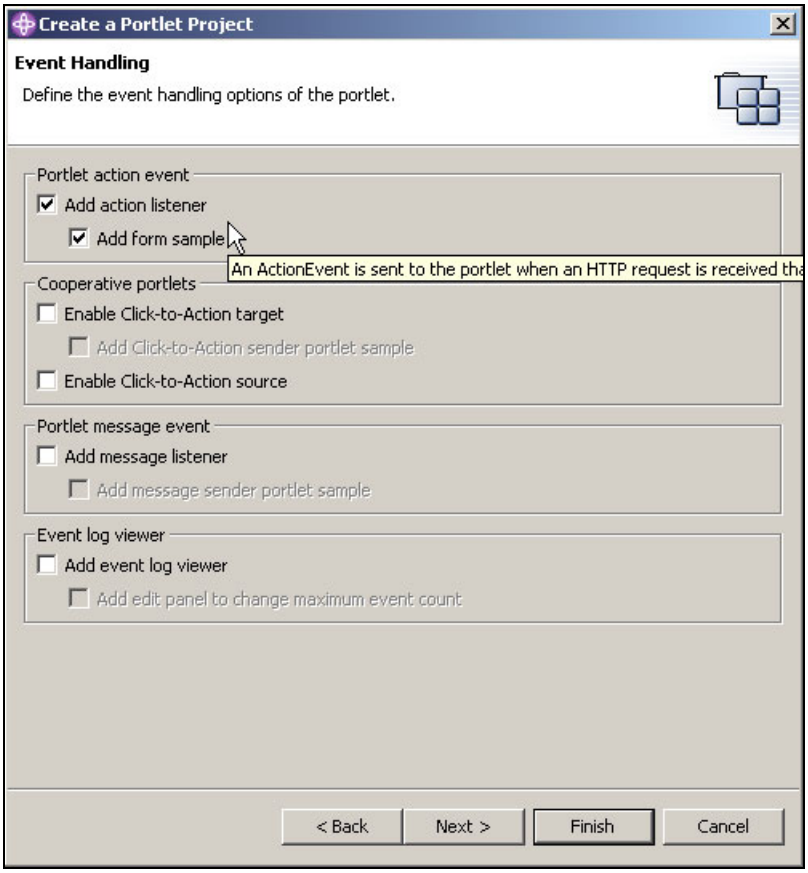


Figure 11-14 Add action listener and form sample in WebSphere Studio Portlet Project wizard

4. Check PDA markup support and click **Finish** (Figure 11-15 on page 241).

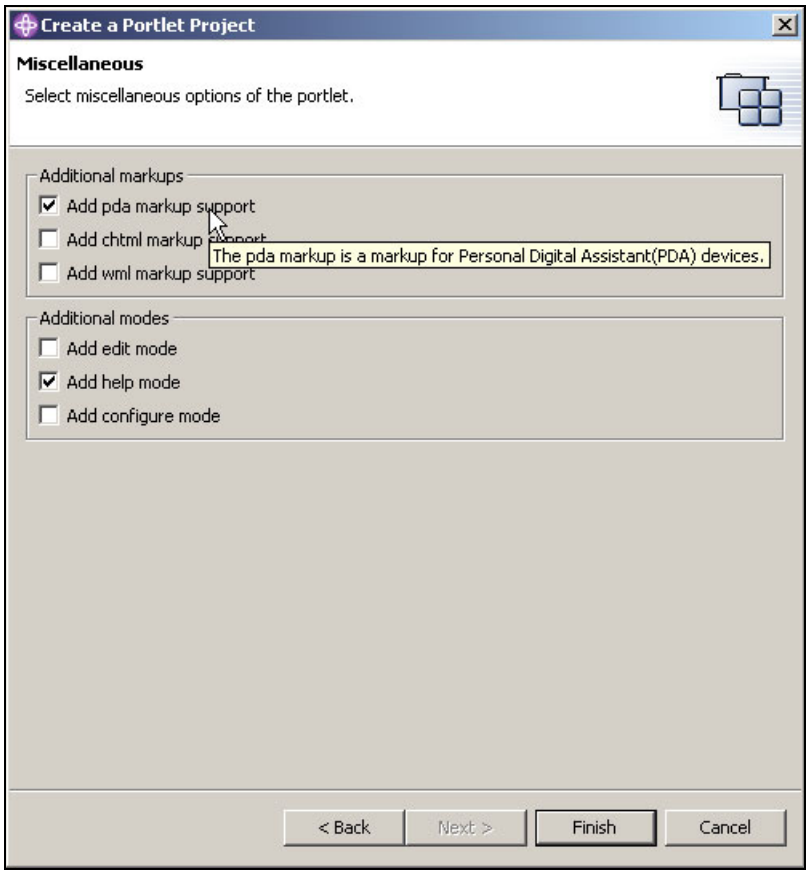


Figure 11-15 Add PDA markup support in WebSphere Studio Portlet Project wizard

The wizard automatically creates the whole project including the EAR files and the portlet.xml file. The PDA markup support was automatically included in the portlet.xml file (see Figure 11-16 on page 242).

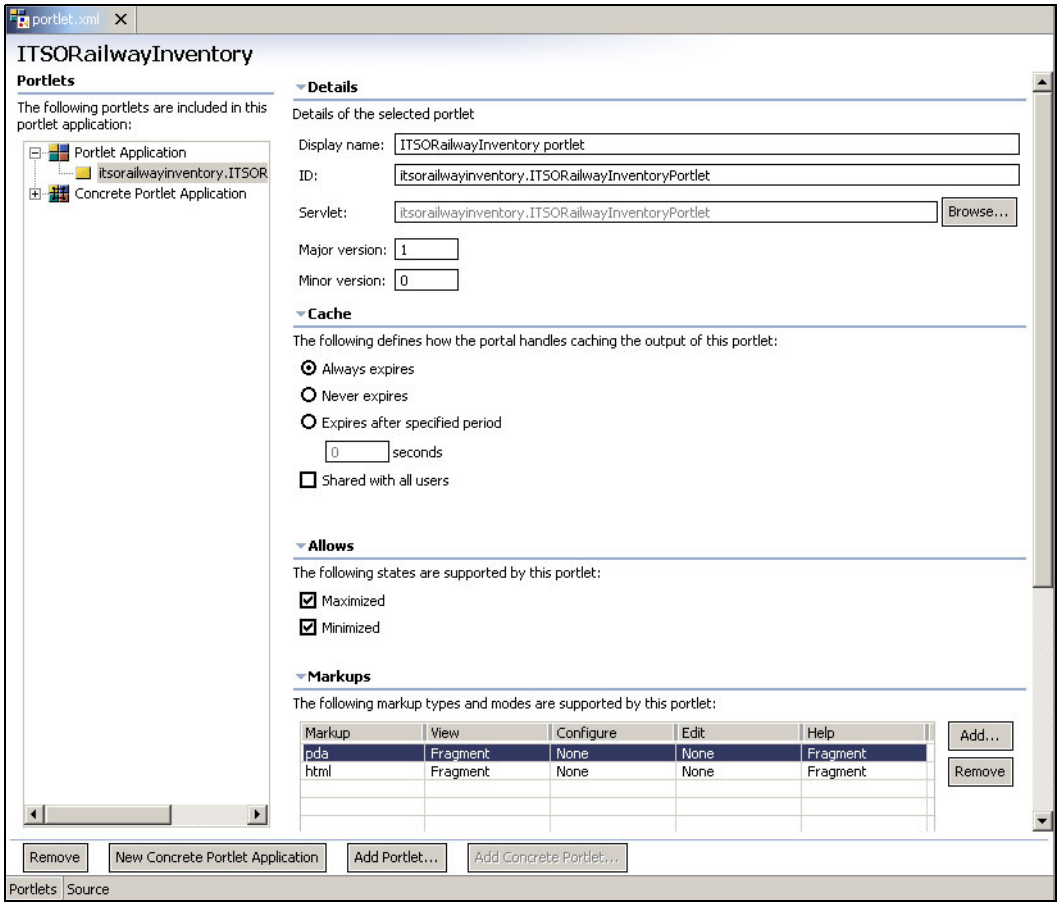


Figure 11-16 Automatically created portlet.xml

All necessary Java and Web resource templates including JSPs for the PDA markup are also automatically provided (see Figure 11-17 on page 243).

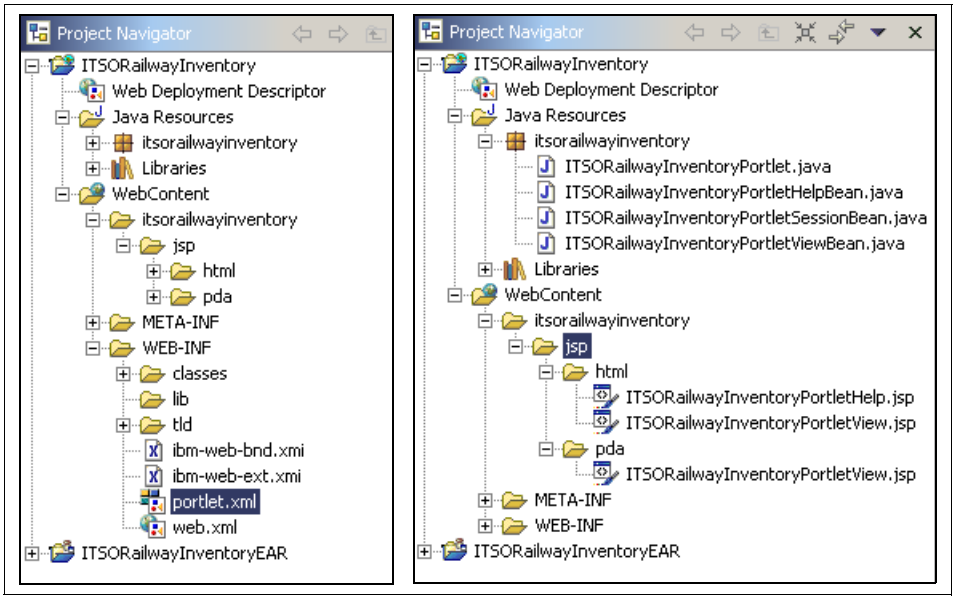


Figure 11-17 Automatically created file templates

5. Run this wizard-generated code on the integrated WebSphere Portal Server test environment. (Right-click the project folder, and choose **Run on Server** from the context menu; see Figure 11-18 on page 244.) Create a new server for WebSphere Portal v5.0 Test Environment using port 9081 (see Figure 11-18 on page 244).

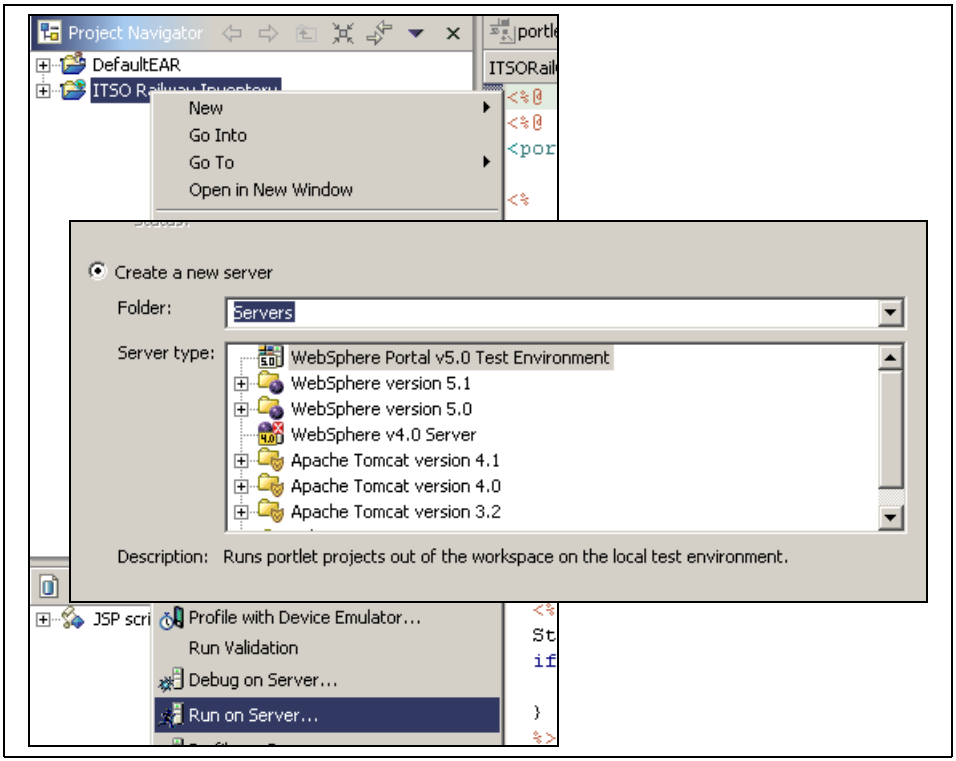


Figure 11-18 Run code created by the wizard in the integrated WebSphere Portal test environment

6. Test the wizard-generated template code in the test environment. Enter a number in the order id field and notice that this number is processed after clicking the **Submit** button (displayed in the line above the order id field; see Figure 11-19 on page 245 and Figure 11-20 on page 245). This code runs in the integrated test environment. It was changed for the ITSO Railway Supply Consumption Record form.

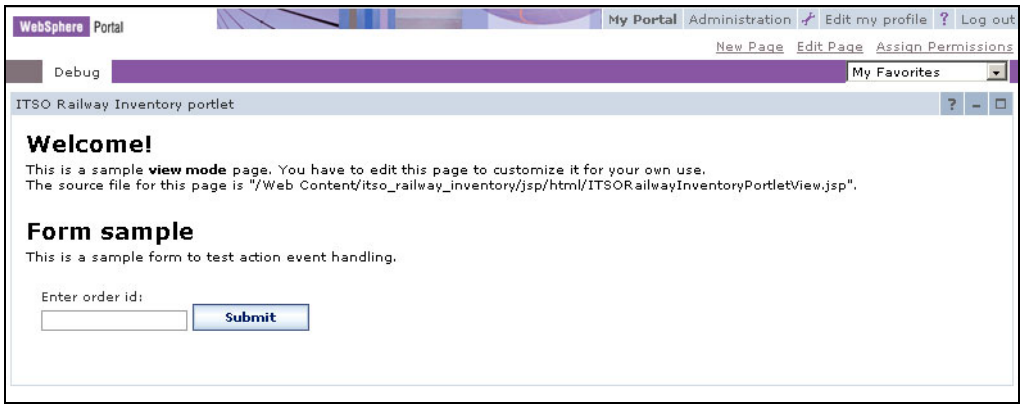


Figure 11-19 View of automatically generated forms portlet

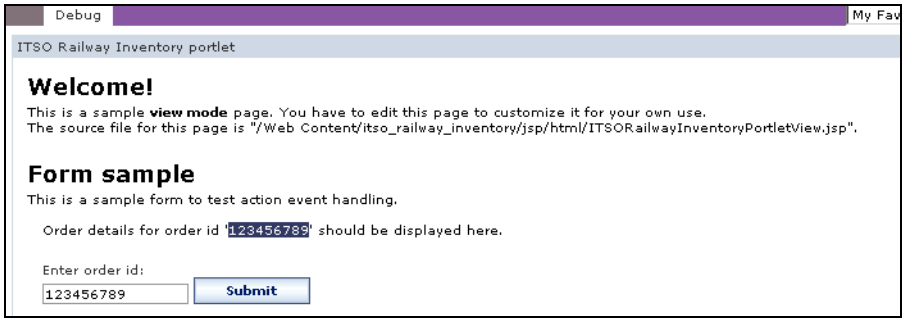


Figure 11-20 View of forms portlet after entering 123456789 in order ID field and clicking Submit button

7. Change this code so that the used supplies from the ITSO Railway Mobile Supply Tracking System example can be recorded with a Supply Consumption Record form. Just change the layout of the form with WebSphere Studio. The JSP editor provides a design view that allows adding HTML and JSP-specific tags by dragging and dropping them onto the form screen. In our example (ITSORailwayInventory project), only the form's title was changed. A drop-down list for the train numbers and a drop-down list for the supplies, as well as an input field for the amount of the used supplies, were added (changes in the ITSORailwayInventoryView.jsp; see listing in Example 11-1).

Example 11-1 Layout changes in the ITSORailwayInventoryView.jsp

```
<FORM method="POST" action="<%=viewBean.getFormActionURI()%>" name="myForm_0"
onsubmit="return validate()">
   <LABEL class="train_Num"  for="<portletAPI:encodeNamespace
```

```
   value='<%=ITSORailwayInventoryPortlet.TRAIN_NO%>'/>">train number: </LABEL>
   <SELECT class="Train_no" name="<portletAPI:encodeNamespace
         value='<%=ITSORailwayInventoryPortlet.TRAIN_NO%>'/>" id="train_no">
      <OPTION>IC 2393</OPTION>
      <OPTION>IR 481</OPTION>
      <OPTION>EC 101</OPTION>
      <OPTION>CIS 259</OPTION>
   </SELECT><BR>
<BR>
   <TABLE border="1"><TBODY><TR>
         <TD width="98"><B>  Supply</B><BR> (select name)</TD>
            <TD width="83"><B> Amount</B><BR> (packages)</TD>
         </TR>
         <TR>
            <TD width="98">
       <SELECT name="<portletAPI:encodeNamespace
         value='<%=ITSORailwayInventoryPortlet.SUPPLY_1%>'/>" id="supply_1">
                  <option value="Soap">Soap</option>
                  <option value="Hand towels">Hand towels</option>
                  <option value="Toilet roles">Toilet roles</option>
                  <option value="Tissues">Tissues</option>
               </select></TD>
            <TD width="83">
       <INPUT class="wpsEditField"  name="<portletAPI:encodeNamespace
         value='<%=ITSORailwayInventoryPortlet.AMOUNT_1%>'/>" type="text"
         size="3" maxlength="3"/><!--Validate-->
            </TD>
         </TR>
      </TBODY>
   </TABLE>
<BR>
   <INPUT class="wpsButtonText" name="<portletAPI:encodeNamespace
         value='<%=ITSORailwayInventoryPortlet.SUBMIT%>'/>"
         value="Submit" type="submit" />
</FORM>
```

Furthermore, the entered data (train number, type of used supply, and amount of the used supply) are stored in the session bean after clicking the **Submit** button on the handheld device (bold in Example 11-1 on page 245). This data is stored in the ITSORailwayInventorySessionBean by the ITSORailwayInventoryPortlet (see code changes in Example 11-2 and Example 11-3 on page 247).

Example 11-2 ITSORailwayInventorySessionBean: Setter and getter method example for storage and retrieval of form data (here the train number)

```
// store train number
public void setTrainNum(String trainNum) {
   this.trainNum = trainNum;
```

```
   /* Here, a database connection should be added
    * to store the entered data from the form into
    * the ITSO Railway inventory system database.
    */
}
// return train number
public String getTrainNum() {
   return this.trainNum;
}
// store the typ of the supply
public void setSupply_1(String string) {
   supply_1 = string;
   /* Here, a database connection should be added
    * to store the entered data from the form into
    * the ITSO Railway inventory system database.
    */
}
// return typ of the supply
public String getSupply_1() {
   return supply_1;
}
// store the amount the supply
public void setAmount_1(String string) {
   amount_1 = string;
   /* Here, a database connection should be added
    * to store the entered data from the form into
    * the ITSO Railway inventory system database.
    */
}
// return amount of supply
public String getAmount_1() {
   return amount_1;
}
```

Example 11-3 Changes in the actionPerformed() method of ITSORailwayInventoryPortlet for storing forms data in session bean

```
if( FORM_ACTION.equals(actionString) ) {
   // Set form text in the session bean
   sessionBean.setTrainNum(request.getParameter(TRAIN_NO));
   sessionBean.setSupply_1(request.getParameter(SUPPLY_1));
   sessionBean.setAmount_1(request.getParameter(AMOUNT_1));
}
```

8. Finally, a field validation was added to the Supply Consumption Record form using the WebSphere Studio field validation wizard. The form can only be submitted if the entered amount of supplies is a digit. An error message is displayed if a non digit character (for example, a letter) is entered into the

amount field. It must be remembered that the validation function is a JavaScript implementation that only works on JavaScript-capable browsers.

The validation function is easily added by right-clicking the amount field in the design window of the ITSORailwayInventoryView.jsp (see Figure 11-21). Right-clicking the amount field shows a context menu from which **Create Field Validation** is chosen (Figure 11-21).

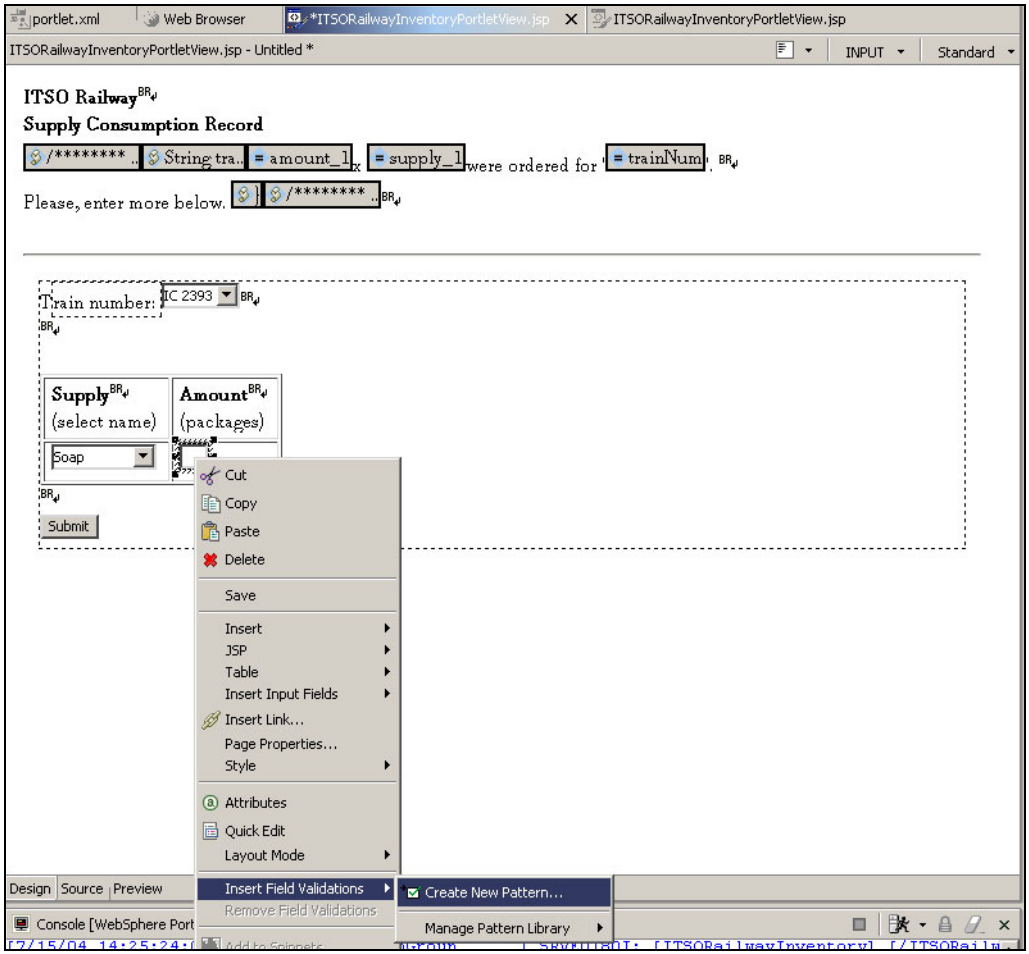


Figure 11-21 Add field validation to ITSORailwayInventoryView.jsp

Select **Create New Pattern** and enter the pattern appropriately (see Figure 11-22 on page 249; refer to “Validation patterns” on page 252 or the WEA InfoCenter for details). We added `\d+` so that a value of a minimum of one digit must be entered before the form can be submitted. If this

requirement is not met, an error message is fired (see Figure 11-36 on page 260).

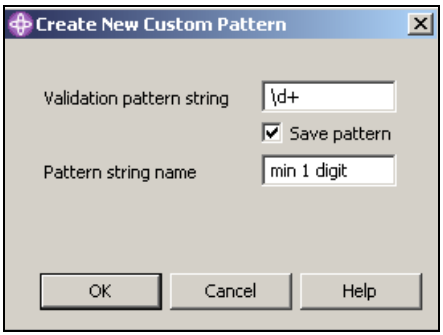


Figure 11-22 Create new validation pattern dialog

Note that the JavaScript code for the validation function was automatically created by the WebSphere Studio (see Example 11-4). The call of function validate() was also automatically added to the Submit button (Example 11-5).

Example 11-4 Automatically generated field validation script

```
<SCRIPT type="text/javascript">
   function validate() {
      var form = document.forms["myForm_0"];
      if (!pattern('\\d+',form.elements["<portletAPI:encodeNamespace
               value='<%=ITSORailwayInventoryPortlet.AMOUNT_1%>'/>"].value)) {
      return false;
      }
      return true;
   }
</SCRIPT>
```

Example 11-5 Created field validation is automatically integrated in POST method

```
<FORM method="POST" action="<%=viewBean.getFormActionURI()%>" name="myForm_0"
onsubmit="return validate()">
```

The error message (alert) that is displayed if the validation of the amount fails was included manually (see Example 11-6).

Example 11-6 Alert added to validation function

```
<SCRIPT type="text/javascript">
   function validate() {
      var form = document.forms["myForm_0"];
      if (!pattern('\\d+',form.elements["<portletAPI:encodeNamespace
               value='<%=ITSORailwayInventoryPortlet.AMOUNT_1%>'/>"].value)) {
```

```
            alert (form.elements["<portletAPI:encodeNamespace
                  value='<%=ITSORailwayInventoryPortlet.AMOUNT_1%>'/>"].value + "
                  is an invalid amount." );
            return false;
         }
         return true;
      }
</SCRIPT>
```

9. Export the portlet containing the Supply Consumption Record form for deployment on the WebSphere Everyplace Access Server. Select the ITSORailwayInventoryProject in the project navigator view of WSAD, right-click, and choose **Export** from the context menu. Select the WAR file format as the destination, click **Next** to enter a WAR file name, and click **Finish** to finally create the file (remember the destination; see Figure 11-23).

Figure 11-23 Export portlet application as WAR file for later deployment on WebSphere Everyplace Access server

This was a short summary of how the mobile forms-based ITSO Railway Mobile Supply Tracking System example application was created. Refer to 11.5.1, “Configuration for offline forms-based applications” on page 253, for the deployment; and to 11.5.2, “Using the application” on page 258, for screen shots of this application.

Further considerations for offline forms applications

Not all portlets work well offline. Only portlets that adhere to the guidelines specified in this topic should be used offline. To prevent administrators and users from selecting portlets that are not intended for offline use, you must add an offline-capable parameter to the portlet.xml file. Note that administrators can add this parameter to portlets using the Manage Portlets portlet. However, they are instructed to verify that the portlet adheres to the guidelines below before adding the parameter.

When you develop a new portlet and want to make it available for offline use, add the offline-capable parameter. The portlet will not be available for administrators or users to select until this parameter is defined. Add the following to the portlet.xml file for the portlet you developed.

```
<config-param>
<param-name>offline-capable</param-name>
<param-value>true</param-value>
</config-param>
```

Offline content development guidelines

Consider the following guidelines when creating portlets for offline use:

- Check the JavaScript support.

  It is important to note that not all clients can support identical levels of JavaScript. Therefore, while the offline functionality does not impose restrictions on JavaScript versions and tag support, the browser software on the client's device may. Please refer to client browser support statements to determine what may or may not be used regarding JavaScript.

- Do not use PortletActions.

  PortletActions are implemented on the Portal server using server-side processing. When the user is browsing these pages offline, this processing cannot be completed in real-time; thus, the resulting link may not be valid. Instead, where state-based decision making was once done with PortletActions, now save that state variable in the request/response object and execute business logic flow decisions internally based on the value of this parameter.

- Avoid action buttons.

  When an offline client synchronizes to obtain the current offline pages, it traverses every link until the link depth is reached. Therefore, if the portlet has a list of items, and beside each item is a button that deletes that individual item, every time the page is processed by the offline servlet, all items will be deleted. Instead, if it is necessary to surface this delete functionality, put the items into a form with check boxes next to each one and then use the offline forms support to delete items in that manner.

- Enable PDA markup.

  The PDA markup must be explicitly supported. When the client connects to the servlet, the servlet then makes a request to the portal for all offline portlets for a given user. The portal responds to this by invoking the offline aggregator to collect all offline-able portlets. For a portlet to be viewable while offline, it must exist on the offline page and it must support the PDA markup. You must update the portlet.xml file to enable PDA markup support. This process is

detailed in the “Adding PDA support to other portlets” section of the WebSphere Everyplace Access InfoCenter.

- Support for cascading forms.

  Forms may now be directly submitted to a URL without having the previous form being submitted in the current session. In other words, the offline servlet will simply post the form data to a URL. Assuming the portlet has been developed such that this URL can accept form submissions that are not dependent on previous session information, the desired functionality will work while offline.

- Use the POST method.

  Only the POST method is supported for submitting forms. Users can only submit one form per page. Embedded forms are not supported, so portlets with embedded forms should not be developed for offline use.

Validation patterns

A validation pattern is a pattern that can be applied to a field in a form. When the form is submitted, the value in the field is evaluated to determine if it follows the pattern. If it follows the pattern, the field value is determined to be valid. If the value does not follow the pattern, the value is determined to be invalid and is rejected; usually an error message is displayed telling the user what is wrong with the value her entered, and the user is given another chance to enter a valid value.

Validation pattern syntax overview (refer to the WebSphere Everyplace Access InfoCenter for details):

- General syntax

  A pattern string follows the XForms pattern validation syntax and consists of a combination of escape sequences, operators, quantifiers, and literals. For example, to specify that any numeric character can be used, type \d. To specify that any alpha character can be used, type \w.

- Examples:
  - Social security number in the form 123-45-6789, type `\d{3}\-\d{2}\-\d{4}`.
  - Seven-digit phone number in the form 123-4567, type `\d{3}\-\d{4}`.
  - Simple e-mail in the form xxx@xxx.xxx, type `\w+@\w+\.\w+`.

11.5 Deployment and runtime configuration

This chapter explains the necessary steps to get the developed forms-based ITSO Railway Mobile Supply Tracking client application example running in a WebSphere Everyplace Access environment. Taking a suggested operational model, the configuration and setup steps are described.

11.5.1 Configuration for offline forms-based applications

After installation and configuration of Everyplace Access Services, there are a few steps needed to configure Offline Portal Content. Administrators will need to configure the link depth and select which portlets can be used offline. Administrators set up and select offline portlets for an offline portal list using the Offline Browsing administrator portlet in the Administrator page group. Then users can select their offline portlets from the administrator-created offline portal list. Users select the portlets they need to use when disconnected using the Offline Browsing user portlet. The user portlet is setup on the Mobile Setup page by default.

Offline Portal Content usage: Administrators and developers

Administrators must configure offline portal content and define which portlets are available for offline use. Administrators will use the Offline Browsing administrator portlet to set up offline portal content. Developers that are creating portlets for offline use need to be aware of offline portal content usage limitations. Administrators and developers should take note of the following points (also see 11.4.2, “Development of forms-based applications for mobile devices” on page 237):

- The Offline Browsing user portlet itself should not be made available for offline browsing. The portlet will not work once it is synchronized to the device.
- Once a portlet has been enabled for offline use, changes to the portlet can cause errors. If changes are made, users may encounter failures when trying to post form submissions. If a failure occurs on the device during synchronization of offline forms data, users should synchronize the offline data and attempt to resubmit the form.
- Users can only submit one form per page. Embedded forms are not supported, so do not select portlets with embedded forms for offline use.
- Only static pages are supported. A dynamically created page with a form cannot be successfully completed while offline. Also, any pages that users must browse in order to get to the form cannot include dynamically generated links.
- Only portlets that support PDA markup can be used with offline portal content.

- Only the POST method is supported for submitting forms.

Summary of administrator (ID) steps necessary for offline viewing:

1. Installation of offline portlet.
2. Define access rights to installed offline portlet for offline users.
3. Allow offline portlet being viewed offline by adding it to the WEA offline portal content.
4. Offline content user steps necessary for offline viewing: Add offline portlet to the users offline portal content.

The following example uses the ITSO Railway Supply Consumption Record offline forms of the Mobile Supply Tracking client application developed in 11.4.2, “Development of forms-based applications for mobile devices” on page 237, to explain the necessary configuration steps. (Tip: Log on using your WEA administrator ID for all administrative steps):

1. Log on as administrator, go to Administration, choose **Portlets** and **Install**. Browse to the location of the previously exported offline portlet (WAR file from 11.4.2, “Development of forms-based applications for mobile devices” on page 237) and click **Install** (see Figure 11-24).

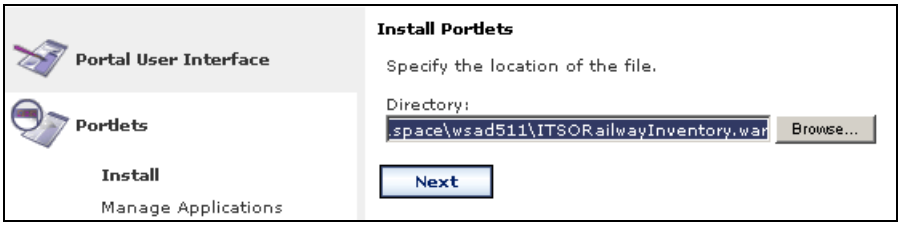


Figure 11-24 Installation of ITSO Railway portlet containing the Supply Consumption Record form in WebSphere Everyplace Access

2. Create the user group (for example, offlineformsusers), which contains all users with access rights to the mobile inventory application (refer to the WebSphere Portal InfoCenter for details).
3. Assign access rights to the installed ITSO Railway inventory portlet. Go to Administration, Access, User and Group Permissions. Search for the offline users group (for example, offlineformsusers), and click the Select Resource Type icon (see Figure 11-25 on page 255).

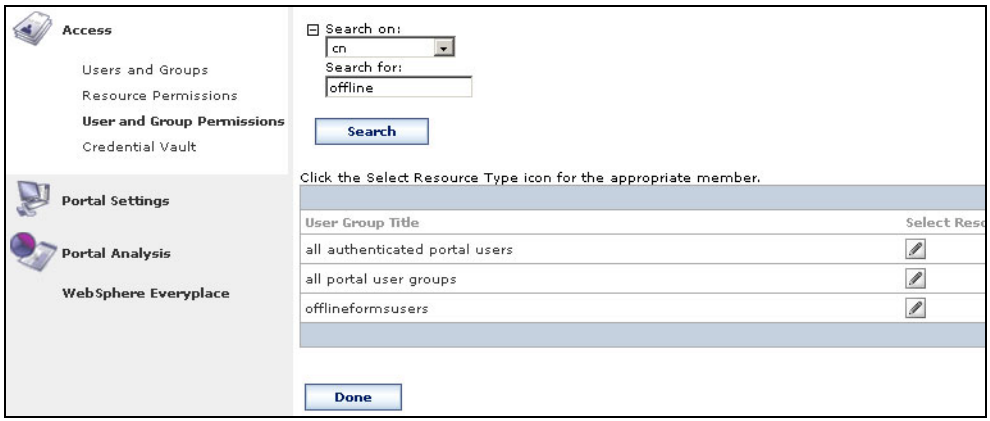


Figure 11-25 Assign ITSO Railway portlet access rights to offline users (1)

Click **Portlets** on the next screen. Search for ITSO to find the installed ITSO Railway inventory portlet containing the Supply Consumption Record form. Click the Assign Access icon on the displayed inventory portlet (see Figure 11-26).

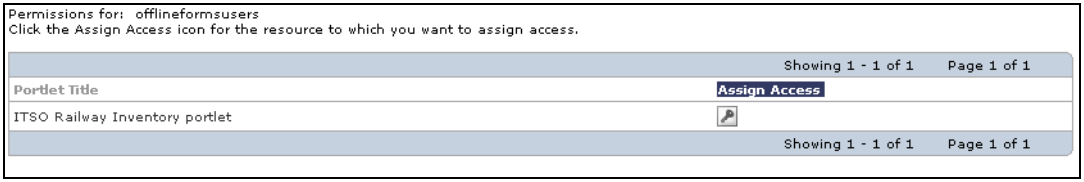


Figure 11-26 Assign ITSO Railway portlet access rights to offline users (2)

Check the **Privileged User** and **User** role check boxes.

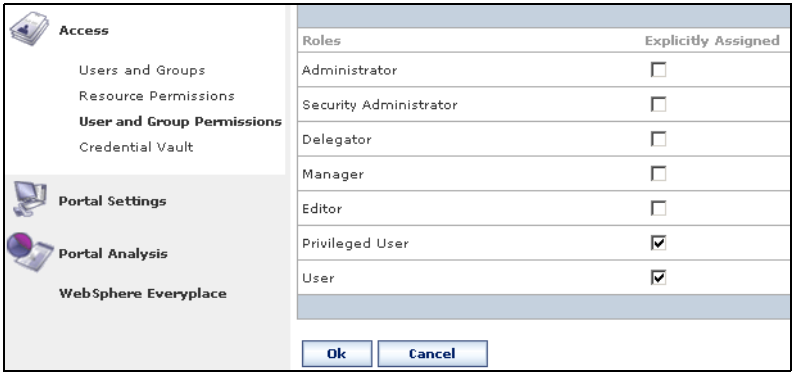


Figure 11-27 Assign ITSO Railway portlet access rights to offline users (3)

Click **OK** and **Done** until the access right assignment was successful.

4. Configure the offline browsing content server and the default link depth. Make the ITSO Railway inventory portlet with the Supply Consumption Record form available as offline content to other users. Go to Administration, WebSphere Everyplace and Offline Browsing (see Figure 11-28) and click on the configure icon of the Offline browsing administration portlet. Enter the server name and the link depth appropriately (see Figure 11-29).

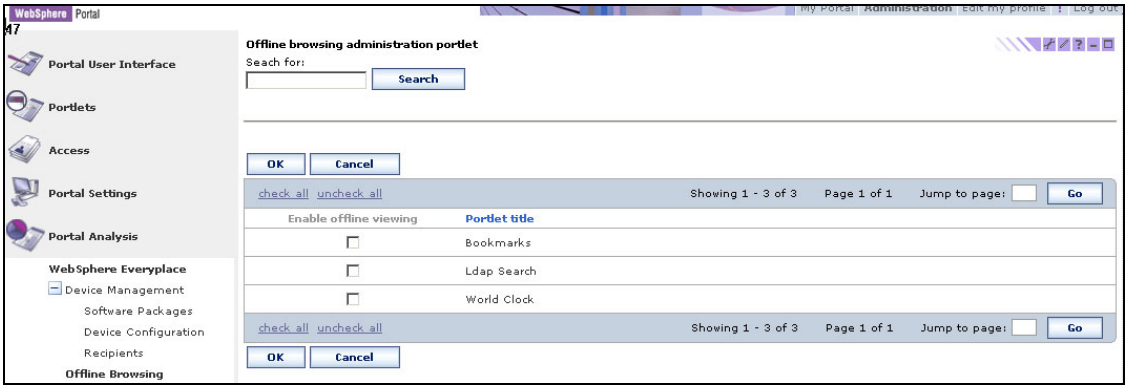


Figure 11-28 WebSphere Everyplace offline browsing configuration

Figure 11-29 Configure offline browsing administration portlet

5. Make the Supply Consumption Record form provided by the ITSORailwayInventory portlet available for offline viewing to users. Check whether the portlet has the parameter offline-capable set to true. For this, go to Administration, Manage Portlet, choose the **ITSORailwayInventory** portlet and click **Modify parameters** (see Figure 11-30).

Figure 11-30 Offline portlets must have offline-capable set to true

The Administrator must allow offline viewing for the Supply Consumption Record (ITSORailwayInventory portlet). For this, go to Administration, WebSphere Everyplace, Offline Browsing, and check ITSORailwayInventory portlet (see Figure 11-31).

Figure 11-31 Administrator allows offline viewing of the ITSORailwayInventory portlet

6. Offline portlet users must configure their own offline content. Log in as a delivery person, for example, Dan Delivery (an offline content user), who must be a member of the offlineformsusers group (defined under point 2). Go to the Mobile Setup page group and the Offline Browsing page. Check the **ITSORailwayInventory** portlet in the displayed Offline Browser User Portlet and click **OK** to make it and the included Supply Consumption Record form available for offline synchronization (see Figure 11-32).

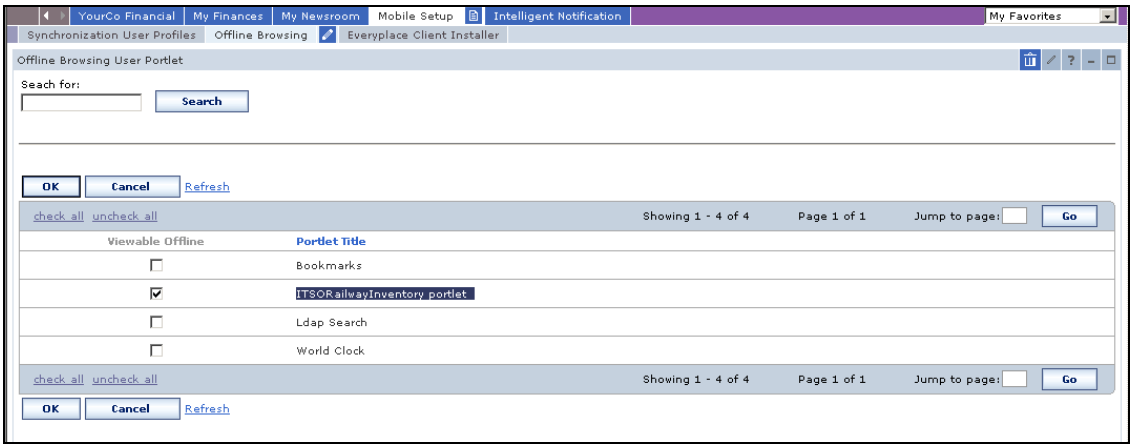


Figure 11-32 User configuration in WEA to make portlet available offline

Now the user Dan Delivery is able to use the ITSORailwayInventory portlet offline.

11.5.2 Using the application

The offline content made available through the Offline browsing user portlet is displayed on the PDA of the delivery person using a device browser, such as Pocket Internet Explorer on Pocket PC devices. The delivery person must enter a power-on password when the PDA is switched on as well as user name and password to access the Mobile Supply Tracking client application. The latter is the WEA client user authentication to assure authorized access to the Mobile Supply Tracking System and the inventory system. This, for example, allows disconnected viewing of a train schedule portlet that produces static HTML fragments. The portal content is downloaded to the device from the network during synchronization if it was configured appropriately (see 11.5.1, “Configuration for offline forms-based applications” on page 253). Each time the portal offline content is synchronized, any new information is retrieved from the portal (for example, changes in the schedule of trains that must be equipped with new supplies).

Since the train schedule for the delivery person is provided as offline portal content as well (offline page), he synchronizes this schedule down to the PDA at the beginning of a shift (PDA is cradled at the service station to connect to the WEA server). For this synchronization, the WEA client is used (see Figure 11-33).

Figure 11-33 WebSphere Everyplace Client: Log on, client menu and synchronization of offline content

The delivery person uses the received train schedule in order to find the train that has to be equipped with new supplies. The used supplies can be recorded immediately in the Supply Consumption Record form on the PDA (see

Figure 11-34 and Figure 11-35). It is assured that the delivery staff can complete and submit the Supply Consumption Record form while being offline.

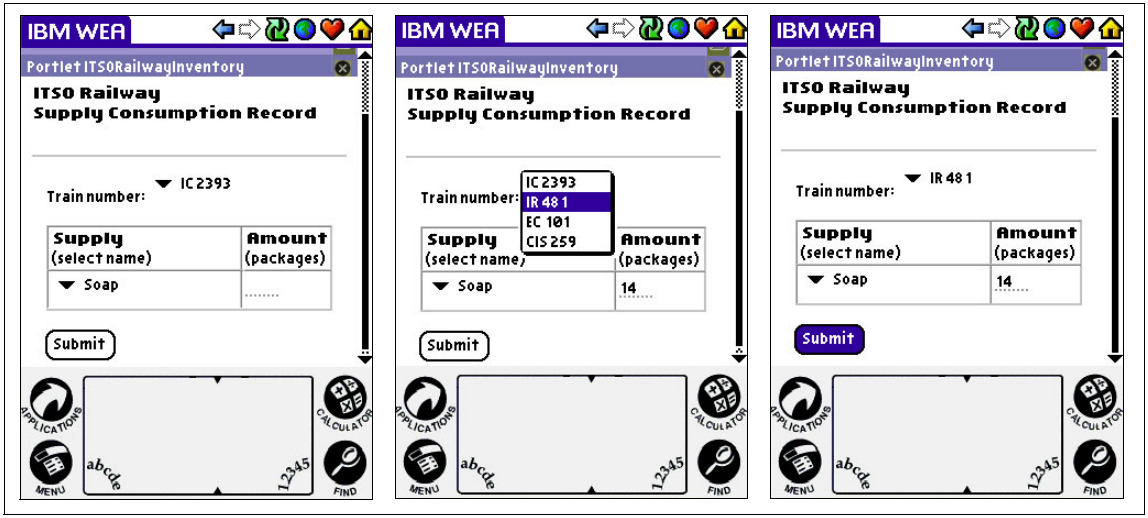


Figure 11-34 ITSO Railway Supply Consumption Record form on a PalmOS device (first screen: start screen, second screen: choose train, third screen: enter used supply and amount)

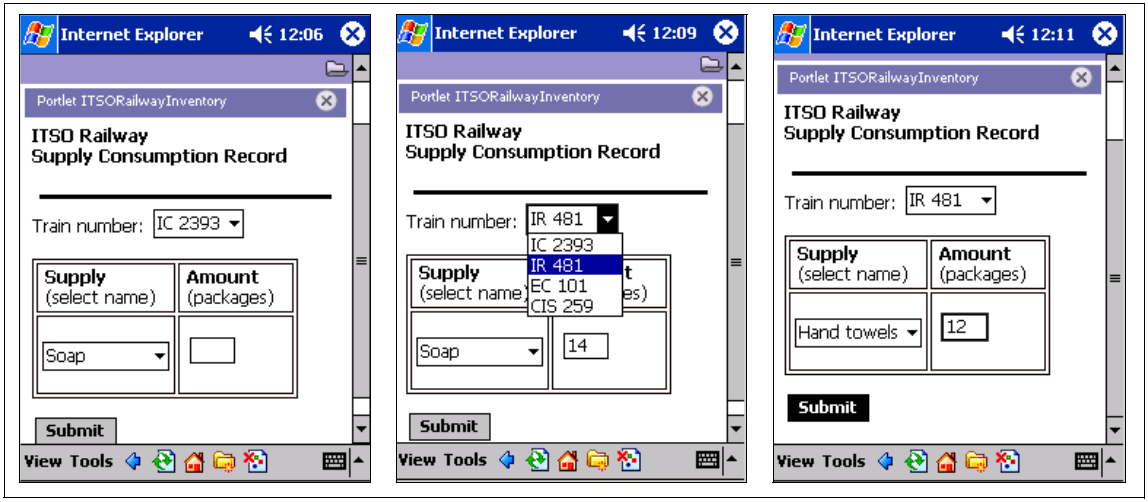


Figure 11-35 ITSO Railway Supply Consumption Record form on a PPC 2003 device (first screen: start screen, second screen: choose train, third screen: enter used supply and amount)

Field validation functions can also be used offline. The handheld device's browser must support JavaScript for this (see Figure 11-36 on page 260).

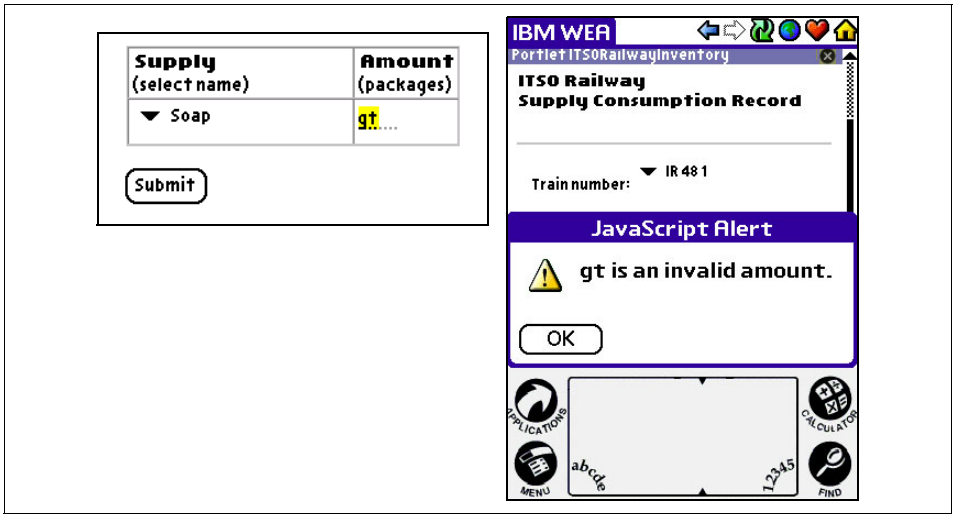


Figure 11-36 Field validation example for offline forms on a PalmOS device with a JavaScript capable browser (only digits are allowed)

The supply data that was entered in the Supply Consumption Record form is stored on the device and then submitted to the server the next time the delivery person connects to the WEA server (for example, at the end of his working shift; see synchronization menu in Figure 11-33 on page 258). The user receives the same confirmation as when submitting the form online. If a failure occurs on the device during the synchronization of offline forms data, WEA will try to synchronize the offline data again (attempt to re-submit the form). Also, if the delivery person's cradled and connected PDA should loose its connection to the WEA server before the form is submitted to the inventory system, WEA attempts to re-submit the form again during the next synchronization. If the connection is lost before the delivery person receives a confirmation for the submitted form, the WEA server will store the confirmation and send it to the user after the next synchronization.

11.6 Summary

This chapter summarized the basic steps for the enablement of existing applications for mobile devices using forms. It also gave an outlook to upcoming technology and discussed other implementation approaches for the ITSO Railway use cases.

The inventory system can be seen as the representative of a general existing legacy application in an enterprise. It is important to understand that no changes

to the existing ITSO Railway inventory system were made. Of course, this example is very simple, but nevertheless the introduced approach applies to any application that is made available to mobile users.

Two approaches are discussed in this book:

- A complex application on a handheld device with appropriate computing power (see Chapter 13, “Using Workplace Client Technology, Micro Edition” on page 283)
- A simple offline forms actions-based applications for data entry

The second approach was discussed in this chapter. Enterprises, which already use WebSphere Portal Server as their consolidated user front-end for application access, face minor changes and/or extensions to portlets for mobile accessibility in most cases with this approach. 11.3.1, “General considerations for intermittently connected applications” on page 228, and 11.4.2, “Development of forms-based applications for mobile devices” on page 237, explain in detail guidelines that must be considered.

It is highly recommended to discuss the mobile enablement of an application with all involved parties (security and system administrators, application developers, and users). The application is extended with view modes for mobile devices. If the existing application follows the Model-View-Controller design pattern already, an extension to mobile devices will be easier. For example, the portlet must be changed to use forms rather then PortletActions. Additionally, PDA view JSPs as well as PDA markup support must be added (read 11.4.2, “Development of forms-based applications for mobile devices” on page 237, for details). Keep in mind that online interactions between the user and the system are not possible during offline mode. Field validation, for example, can be used in offline mode for simple field content checks.

The technical community is working on a standard for forms-based applications, called xForms. This includes standard validation functions based on XML as well as server-side support for offline forms. Scripting will also be considered for appropriate devices.

Further information for development of forms-based applications can also be found in the WebSphere Everyplace Access InfoCenter and the WebSphere Everyplace Access Handbooks.

12

Using Intelligent Notification Services

The objective of this chapter is to describe how to build a pervasive solution that notifies maintenance workers about new maintenance orders. This chapter discusses how to use the Intelligent Notification System (hereafter called INS) of WebSphere Everyplace Access and WebSphere Studio to create this solution.

This chapter discusses the following topics:

- Business context
- Architectural overview model
- The system design overview
- Sample application development
- Deployment and runtime configuration

12.1 Business context

The railway maintenance teams must be notified of any train or railway problems as soon as possible. This is time-critical data and allows the maintenance team to provide immediate service at the point of need and to ensure that the proper persons are dispatched to resolve the problem.

Intelligent Notification System (INS) enhances the ITSO Railways maintenance and support system to notify the maintenance team when a critical event occurs. The maintenance system is responsible for collecting and maintaining all service data from different resources like sensors, service systems, and support centers. INS enhances the existing maintenance system with the capability of alerting a service technician when an urgent service request occurs. INS delivers the messages to the service technician based on his subscription preferences.

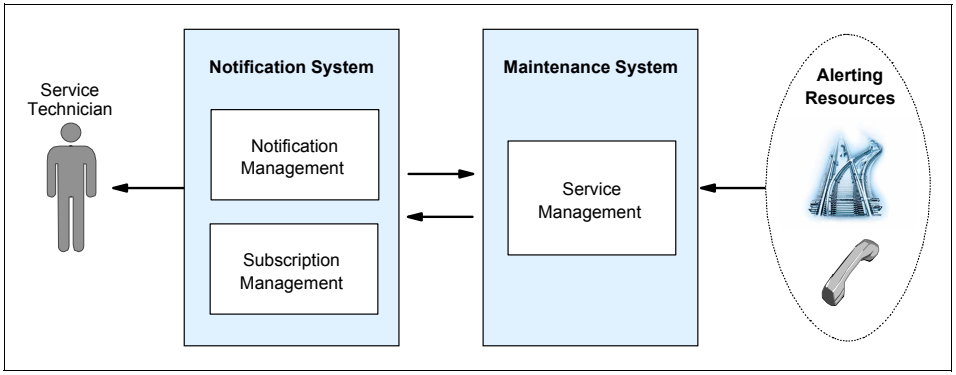


Figure 12-1 ITSO notification business context diagram

12.2 Architectural overview model

This section describes how the Intelligent Notification System extends the existing maintenance system to notify a service technician when a new service request was issued. The architectural overview model covers the entire system and shows the high-level components involved in the overall system.

The following diagram is the Runtime pattern we implemented in this particular scenario.

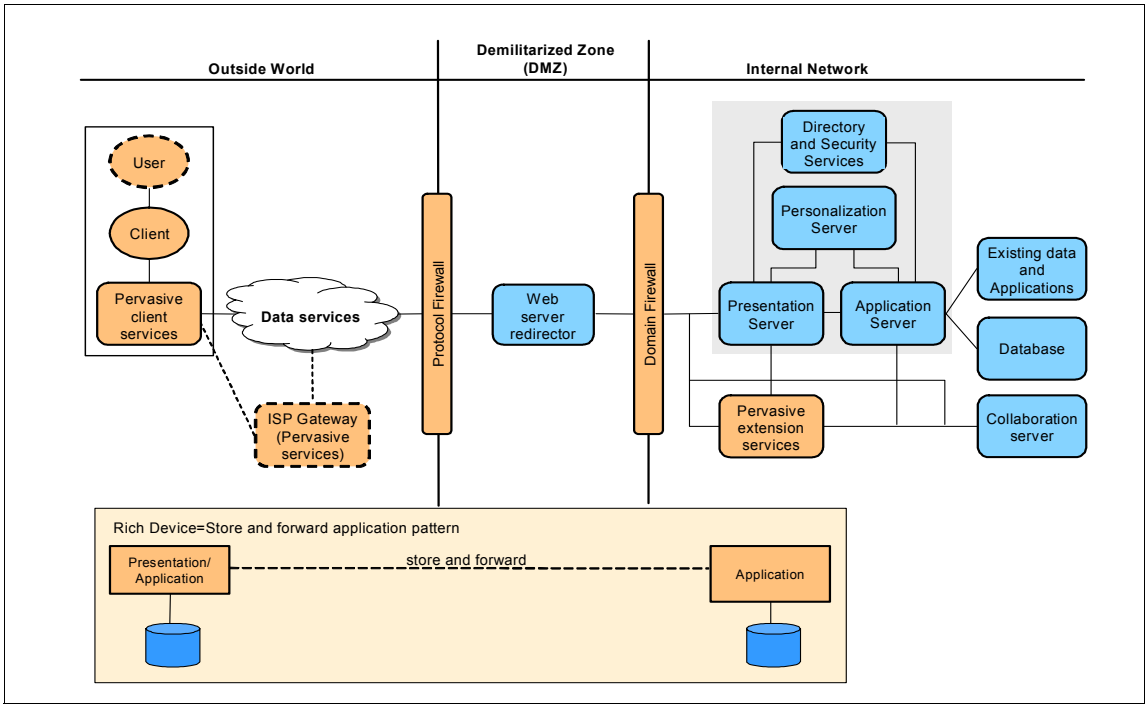


Figure 12-2 Runtime pattern for the intelligent notification scenario

The maintenance system contains the complete support and service infrastructure such as a call center for end-user support, a service request system to manage service requests, and a sensor data collection system to collect and process data from the sensors. In our scenario we assume these systems are in place to trigger the maintenance database for new service requests. A new service request can be created either manually through the call center support or automatically through the sensor system.

The notification system contains both subscription and notification management. The subscription manager allows the service technician to subscribe to new service requests in the maintenance database. It specifies filters to direct service requests received from the content adapter. The content adapter is an application that captures data from the maintenance database and converts that data into a format that the subscription manager can read. The content adapter publishes the data to the subscription manager to match against user subscriptions. When a match occurs, the Subscription Manager calls the appropriate trigger handler, which notifies the targeted service technician through the Notification Manager. The Notification Manager is a component of Intelligent Notification Services that delivers notifications to the service technician based on their preferences.

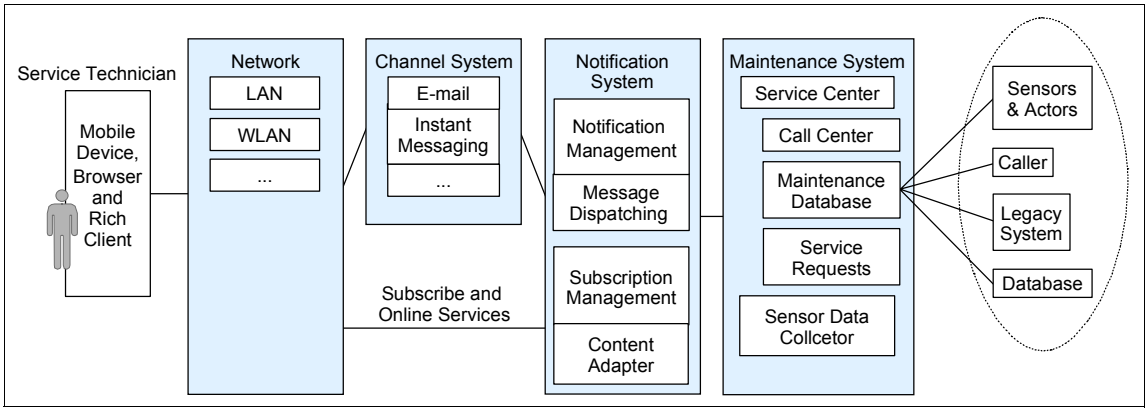


Figure 12-3 ITSO notification architectural overview model

The products we used for this scenario can be seen on the following Product mapping diagram.

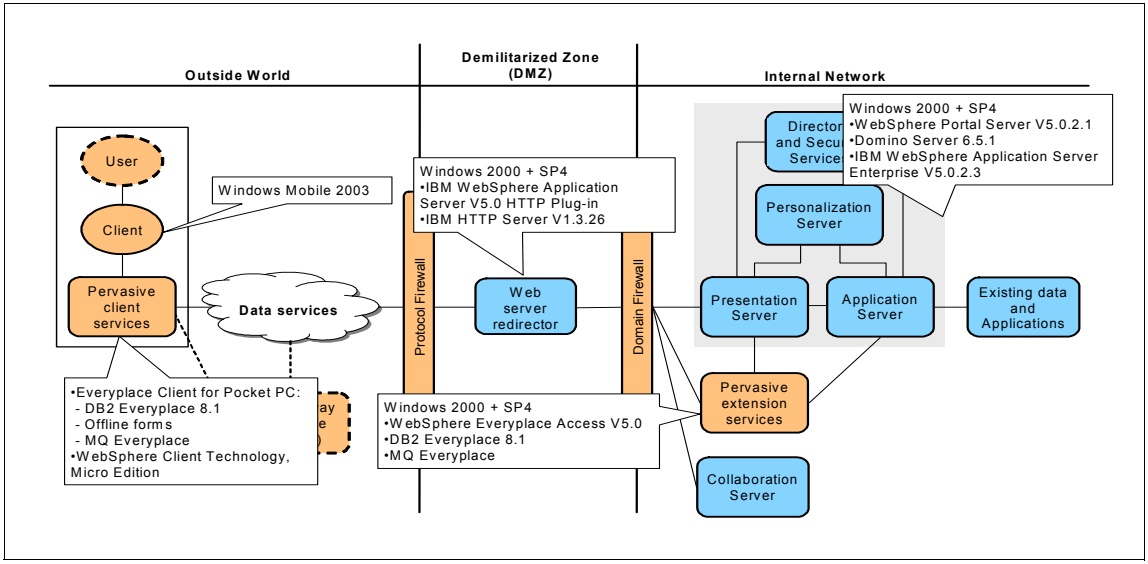


Figure 12-4 Rich Device=Store and forward::Product mapping=Windows

12.3 System design overview

The Intelligent Notification System will be used to monitor the maintenance database for new incoming service requests. A service technician can subscribe

to one or more resources. A subscription means to compare a trigger value with the actual value. When a match occurs, a notification will be sent.

In this scenario, we will monitor a database field called *Origin*. Origin will tell the system from where the alert were issued. *S* indicates an alert from a Switch (mechanical component in the railway industry) and *C* indicates a service request created by the Call Center.

The service technician has created a subscription that will trigger the database field Origin of the maintenance database. The notification system will periodically check if there is a new record with the trigger value = "S*x*" (where *x* is the switch number).

When a match occurs, the subscription manager will proceed with the notification and start a trigger handler, which creates the notification and hands it over to the notification manager. The notification manager will send the message to the service technician based on his preferences and channel availability. In our example we send a Sametime message (instant messaging product) if the user is online, and an e-mail if the user is offline.

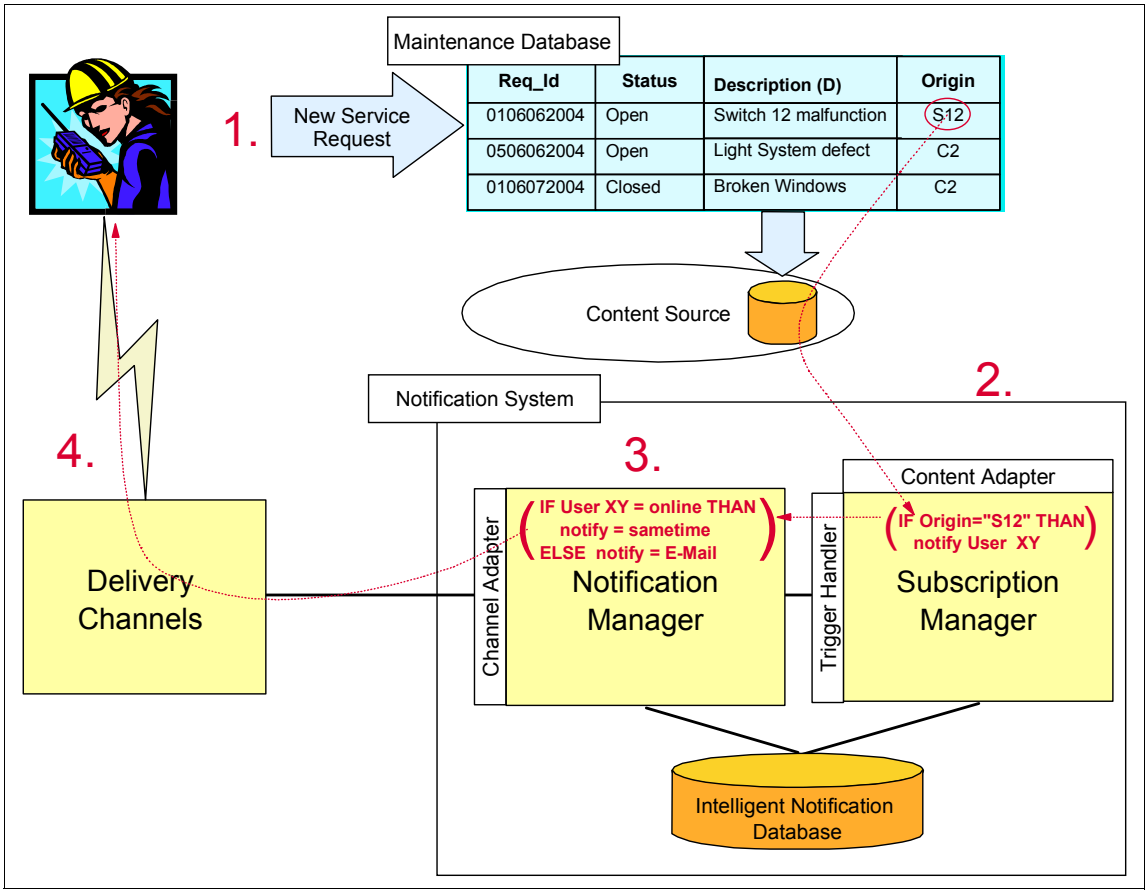


Figure 12-5 ITSO Railways - INS system design overview

Figure 12-5 shows a basic system design consisting of:

- The maintenance database where the new services requests will be placed
- The notification system with its components
  - The subscription manager
  - The notification manager
  - The intelligent notification database, which holds all meta information for the notification system (subscriptions, notifications, and configurations)
- The delivery channels, which are existing mail and instant messaging systems like Lotus Domino and Sametime server

12.3.1 Component model

This section shows the various components of the notification system and their interaction. The context diagram, Figure 12-1 on page 264, shows the two main systems and their surrounding components. The maintenance system has three interfaces for creating and managing service requests. These include the Service Center (self-service and telephone support), a switch monitoring system that will control and maintain the switch sensor functionality, and the maintenance database as central storage for all service requests.

The notification system has two different interfaces for the user and one for the maintenance database. There a two user interfaces supported in our example, the rich browser access to subscribe and access all notification services such as subscriptions, channel configuration, message center, etc. and limited browser access for pervasive devices.

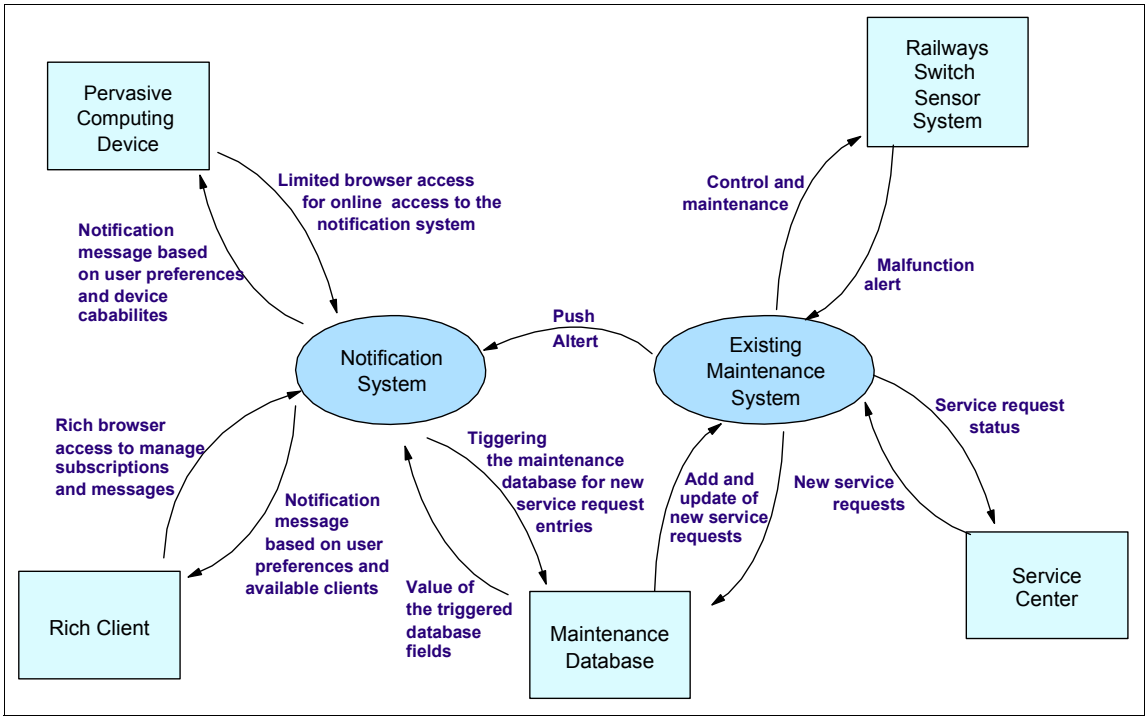


Figure 12-6 ITSO Railways - INS system context diagram

In this example we focus on the notification system. The component interaction diagram, Figure 12-7 on page 270, shows how a possible workflow would look.

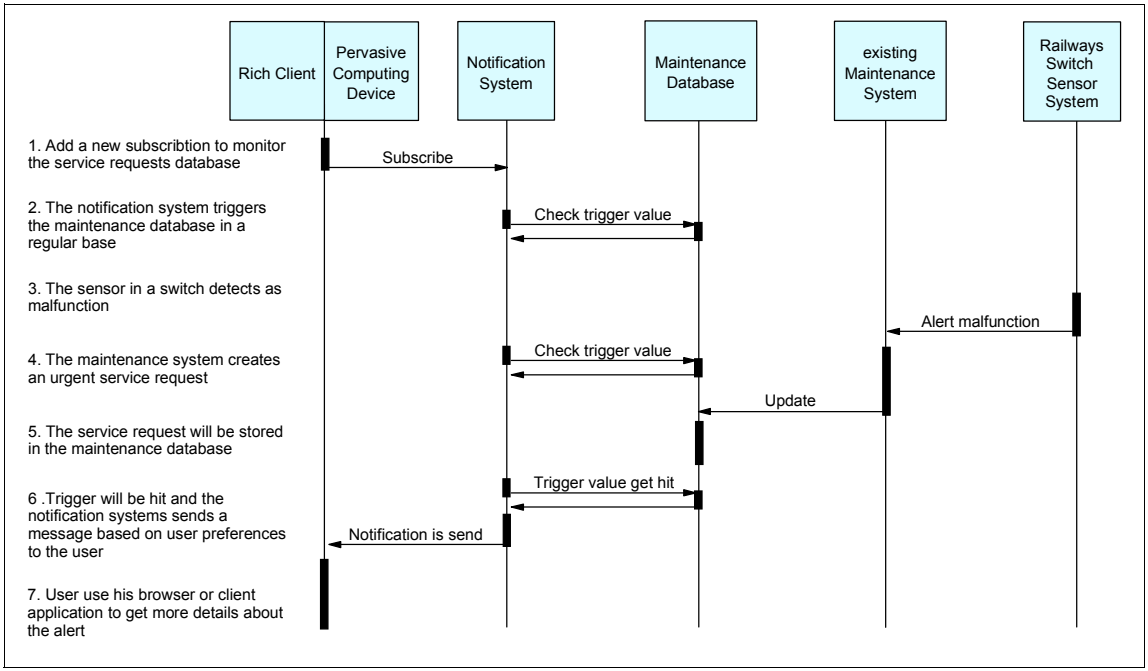


Figure 12-7 ITSO Railways - INS component interaction diagram

12.3.2 Object model

The following classes are descriptions of the classes most frequently used for Intelligent Notification Services development:

- SubscriptionServicesBean - This class contains methods for adding, updating, querying, and removing subscriptions, and publishing content to Subscription Manager.
- TriggerHandler - This class is used to write a custom trigger handler. It contains methods for subscribing to content, activating a subscription, and handling content matches.
- NotificationServiceBean - This class contains methods for sending and canceling notifications to the Notification Manager.
- NmClientFactory - This class creates a local EJB or remote EJB.
- SmClientFactory - This class creates a local EJB or remote EJB.
- PrefMgr - This class contains methods needed to add, delete, retrieve, and update preference data in the database.

12.4 Sample application development

This section discusses the creation of the necessary sample code for the ITSO Railway Intelligent Notification System sample and the development environment used to build it.

Intelligent Notification Services (INS) is an install option of the WebSphere Everyplace Access. After INS is installed, it must be configured correctly. Please see the WebSphere Everyplace Access InfoCenter for detailed information on how to install and configure INS.

A developer for Intelligent Notification Services can develop custom subscriptions, custom content adapters, custom trigger handlers, custom channel adapters, and applications that send simple notifications.

To develop a custom subscription, we need to complete the following tasks:

1. Write a content adapter for the new content source.
2. Deploy the new content adapter.
3. Write a subscription portlet for the new content source.
4. Write a custom trigger handler.
5. Deploy the new subscription portlet.

Developers can create new subscriptions in order to utilize other types of content sources. In order to develop a subscription, a developer must create a subscription portlet, a trigger handler, and a content adapter. The subscription application allows the subscriber to subscribe to the content source and set subscription/matching parameters. The content adapter retrieves data from the content source, transforms it, and sends it to the Subscription Manager. The content adapter APIs are designed to make it as simple as possible for a user to publish data from new content sources as they become available. The content adapter APIs enable the developer to focus on writing code to retrieve data from the content source rather than writing code to interface with the Subscription Manager.

Trigger handlers are also considered part of the subscription because they control what happens when an action is performed on a subscription, such as on add, on update, on query, or when a subscription matches content that is published to the system.

Write the content adapter

The code (.java files) for the following content adapters are packaged in the .ear files found in the <WEA_INS_root>/samples/sm directory, where WEA_INS_root is the root Intelligent Notification Services installation directory. Each .ear file contains a .jar file that contains the .java file for the content adapter. For more

information about packaging and deploying content adapters, see *Deploying a custom content adapter* in the InfoCenter.

1. Create a content adapter file or copy an existing one.
2. Import the required package files into the class.

   ```
   import com.ibm.pvc.ins.sm.client.*;
   import com.ibm.pvc.ins.sm.server.SubscriptionService;
   import java.io.URL
   ```

   These packages are located in /INS/lib/insSmClient.jar.
3. Create the content adapter class.

   ```
   public class MyNewsContentAdapter()
   ```
4. Create the variables.
5. Retrieve the host name of the Subscription Manager and create a Subscription Manager remote EJB client connection.

   ```
   try {
      smClient = SmClientFactory.createRemoteSmClient();
   } catch (IQueueException ce) {
      // ...
   }
   ```
6. Get data from the content source.
7. If necessary, transform the incoming data into an XML document. Then translate the XML document into a format that is acceptable by the Subscription Manager using an XSL style sheet.
8. Publish the XML document to the Subscription Manager.

Deploy the new content adapter

To communicate with Subscription Manager, custom content adapters need to be deployed as one of the following:

- WebSphere Application Server J2EE Client Application
- WebSphere Application Server Enterprise Application

To run a content adapter packaged as a WebSphere Application Server J2EE Client Application, issue the following command:

```
<WebSphere_root>\bin\launchclient.bat
<content_adapter_directory>\<CustomContentAdapter.ear>
-CCclasspath=%INS_HOME%\lib\insSmClient.jar
```

Write a subscription portlet for the new content source

Intelligent Notification Services users use subscription portlets to subscribe to content source notifications. Each subscription portlet displays a list of

subscriptions for the content source to which it corresponds. The subscriber adds, edits, and deletes subscriptions from that list. When a user adds a subscription, he specifies a criteria for the matching content. Then the subscription portlet builds a Subscription Bean, from which the Subscription Manager retrieves content matching criteria and passes them to the Subscription Manager with the add() method of the SubscriptionServiceBean. The Subscription Manager monitors the content source for a content match and notifies the user when a match has occurred.

To develop a custom subscription portlet, you must follow these steps:

1. Compose and maintain a properties file to contain display strings and images.
2. Extend the Subscription Base Controller class.
3. Extend the base Subscription Bean class.
4. Write a JSP front-end, based on the Subscription View template.
5. Write a JSP front-end, based on the Subscription Configuration template.
6. Write a custom trigger handler for the custom subscription portlet.
7. Write a Web.xml file to define the portlet as a Web application with WebSphere Application Server.
8. Write a portlet.xml file to define the portlet as a portlet with the WebSphere Portal.

Compose a properties file for display strings and images

If you are going to make your custom portlet available in different languages, create a properties file to contain key value pairs for all of the translatable strings.

Extend the Subscription Base Controller class

The SubscriptionBaseController class is the Controller in the MVC architecture of the subscription portlets. All the logic for handling action events from the user interface is implemented here. Import the following packages:

```
//portlet libraries
import org.apache.jetspeed.portlets.*;
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlet.event.*;
import com.ibm.wps.portlets.*;
//java libararies
import java.util.*;
import java.io.*;
import java.text.*;
```

Create a class that extends SubscriptionBaseController and implements ActionListener. For example:

```
public class MySubscriptionBaseController extends SubscriptionBaseController
implements ActionListener { ... }
```

Create an initialization method that calls the initialization method of the superclass and initializes all of the instance variables.

Example 12-1 Initialization method

```
public void init(PortletConfig portletConfig) throws UnavailableException {
   super.init( portletConfig );
   contentSource = "myContentSource";
   triggerHandlerURN = "/samples/MyTriggerHandler";
   subKeys = new String[]{
      SubscriptionBean.KEY_NOTIFICATION_OPTION,
      SubscriptionBean.KEY_CONTENT_STORAGE,
      SubscriptionBean.KEY_DEVICE_NAMES,
      NewsSubscriptionBean.NEWS_SOURCE,
      NewsSubscriptionBean.SUBJECT
   };
   config_jsp = jspBaseDir + "/MySubscriptionConfig.jsp";
   help_jsp = jspBaseDir + "/MySubscriptionHelp.jsp";
   view_jsp = jspBaseDir + "/MySubscriptionView.jsp";
   //...
}
```

You must override the following methods: doAddSubscriptionAction(), doModifySubscriptionAction(). The doAddSubscriptionAction() method is called by the ActionPerformed() method when an "add" action is performed. The doModifySubscriptionAction() method is called by the ActionPerformed() method when a "modify" action is performed.

Override the doAddSubscriptionAction() method.

1. Get parameters from the JSP, including the core subscription parameters and parameters specific to the custom subscription portlet. For example:

   ```
   String mySubscriptionParameter = request.getParameter("myParameter");
   ```

   Refer to the file NewsSubscriptionBaseController.java to see how the sample controller gets each of the core subscription parameters and the sample news-specific parameters from the JSP.

2. Get an instance of SubscriptionManager from the subscription manager factory.

   ```
   SubscriptionManager subManager =
      SubscriptionManagerFactory.createSubscriptionManager(userID,
      log,contentSource,triggerHandlerURN);
   ```

3. Create a new subscription bean and populate it with the subscription parameters from the JSP.

Example 12-2 New subscription bean

```
SubscriptionBean newSub = new SubscriptionBean();
   if (notificationOption != null && contentStorageOption != null
      && newsSource != null && subject != null ) {
      newSub.setNotificationOption(notificationOption);
      newSub.setContentStorage(contentStorageOption);
      newSub.setDeviceNames(selectedDCNames);
      newSub.putProperty(MySubscriptionBean.MY_PARAMETER,
      mySubscriptionParameter);
   }
//...
```

4. Pass the populated subscription bean to the subscription manager via the SubscriptionManager.addSubscription() method. For example:

   ```
   subManager.addSubscription(newSub);
   ```

Override the *doModifySubscriptionAction*() method. Perform the same basic steps used in overriding the doAddSubscriptionAction() method with the following exceptions:

- Instead of creating a new subscription bean to populate, find the bean that is being modified and populate *that* bean with the subscription parameters from the JSP.

  ```
  SubscriptionIndexBean[] indexBeans = (SubscriptionIndexBean[])
     request.getPortletSession().getAttribute(SUB_INDEX_BEANS);
     String subscriptionIndex = request.getParameter(SUB_BEAN_INDEX);
     Integer subIndex = new Integer(subscriptionIndex);
     int index = subIndex.intValue();
     SubscriptionBean subBean = indexBeans[index].getSubscriptionBean();
  ```

- Instead of calling the SubscriptionManager.addSubscription() method, call the SubscriptionManager.modifySubscription() method. For example,

  ```
  subManager.modifySubscription(subBean);
  ```

You can optionally override the doView(), doAdd(), doModify(), and doHelp() methods. These methods are called for each of the "modes" that the portlet can be in. The doView() method is called when the portlet is in its default "view" mode. The doAdd() method is called when the portlet is in "add" mode, when a user is adding a subscription. The doModify() method is called when the portlet is in "modify" mode, when a user is modifying the parameters of a subscription.

To override the *doView*() method, follow these steps.

1. Create a subscription manager.

   ```
   SubscriptionManager subManager =
      SubscriptionManagerFactory.createSubscriptionManager(userID,
      log,contentSource,triggerHandlerURN);
   ```

2. Create an array of subscription beans by calling the getSubscriptions() method of the subscription manager class.

```
SubscriptionBean[] subBeans =
   subManager.getSubscriptions(contentSource, myKeys);
```

3. Cache the array of subscription beans. The doModify() method later retrieves the array from the portlet session in order to determine which subscription bean to modify.

To override any of the "mode" methods, register the corresponding JSP to be displayed by the portlet. For example:

```
includeJSP("/WEB-INF/INS/NewsSubscriptionView.jsp",request, response);
```

If you want to perform more specific actions than the default actions of the superclass, you must override the actionPerformed() method of the ActionListener interface.

```
public void actionPerformed (ActionEvent event) throws PortletException { ... }
```

Get an instance of SubscriptionManager from the subscription manager factory.

```
SubscriptionManager subManager =
   SubscriptionManagerFactory.createSubscriptionManager(userID,
log,contentSource,triggerHandlerURN);
```

Write code segments to handle an "add" event, a "modify" event, and a "delete" event. Create conditional expressions to test for and handle each of these cases. Add events happen when the action name is "add." Modify events happen when the action name is "modify." Delete events happen when the action name is ACTION_DELETE.

Get the action name by calling getName() on the result of a call to ActionEvent.getAction(). For example:

```
DefaultPortletAction action =
   (DefaultPortletAction)event.getAction();
if (action != null && action.getName().equals("add")) { ... }
```

When the action name is "add," call the subscription bean constructor method to create a new subscription bean to add.

```
SubscriptionBean newSub = new SubscriptionBean();
```

Then populate the new subscription bean with parameters from the subscription request. For example:

```
String notificationOption = request.getParameter("notificationOption");
if (notificationOption != null) {
   newSub.setNotificationOption(notificationOption);
}
```

Then call SubscriptionManager.addSubscription(), passing in the new subscription bean.

```
subManager.addSubscription(newSub);
```

When the action name is "modify," find the subscription bean.

```
SubscriptionIndexBean[] indexBeans = (SubscriptionIndexBean[])
       request.getPortletSession().getAttribute(SUB_INDEX_BEANS);
   String subscriptionIndex = request.getParameter(SUB_BEAN_INDEX);
   Integer subIndex = new Integer(subscriptionIndex);
   int index = subIndex.intValue();
   SubscriptionBean subBean = indexBeans[index].getSubscriptionBean();
```

Then populate the subscription bean with new parameters from the subscription request. For example:

```
String notificationOption = request.getParameter("notificationOption");
if (notificationOption != null) {
   subBean.setNotificationOption(notificationOption);
}
```

Then call SubscriptionManager.modifySubscription(), passing in the modified subscription bean.

When the action name is ACTION_DELETE, find the subscription bean.

```
SubscriptionIndexBean[] indexBeans = (SubscriptionIndexBean[])
   request.getPortletSession().getAttribute(SUB_INDEX_BEANS);
String subscriptionIndex = request.getParameter(SUB_BEAN_INDEX);
Integer subIndex = new Integer(subscriptionIndex);
int index = subIndex.intValue();
SubscriptionBean subBean = indexBeans[index].getSubscriptionBean();
```

Then call SubscriptionManager.removeSubscription(), passing in the subscription bean to be deleted.

```
subManager.removeSubscription(subBean);
```

Extend the base Subscription Bean class

The Subscription Bean class is the Model in the MVC architecture of the subscription portlets. The SubscriptionBean superclass contains variables for the core subscription properties including the notification option, the content storage option, the names of the preferred delivery channels, the trigger ID (subscription name), and the content source. The subscription bean also contains getter and setter methods for each of these properties.

When you extend the SubscriptionBean class, specify any additional properties that are specific to your custom subscription portlet. For example, the NewsSubscriptionBean class specifies the NewsSource and Subject variables,

which are additional information items that the subscriber provides to help determine which news stories he wants to subscribe to. You can also specify getter and setter methods for each of your custom subscription properties.

Declare this subclass to be part of the following package:

```
package com.ibm.pvc.we.ins.portlets;
```

Import the following packages:

```
import java.util.*;
import java.io.Serializable;
```

Create a class that extends the SubscriptionBean superclass and implements the Serializable interface. For example:

```
public class MySubscriptionBean extends SubscriptionBean implements
Serializable { ... }
```

Declare property keys for the subscription properties specific to the custom subscription portlet.

Note: The trigger handler for this subscription portlet must handle these properties.

You can optionally write getter and setter methods for each of the custom subscription properties. If you do not write getter and setter methods for each of the custom subscription properties, you can instead access the properties from the subscription controller by calling the getProperty and putProperty methods of the superclass of subscription bean.

Compile the new java source code.

The .java source code must be compiled into .class files before it can be executed by the portlet. Use the same java compiler that is used by your version of WebSphere Application Server, which is installed in <WebSphere_root>\java. Several jar files will need to be included in the compiler's classpath. From the lib directory of the Intelligent Notification Services Portlets <WPS_root>\installedApps\INSPortlet_PA_*xxxx*.ear\INSPortlet.war\WEB-INF\lib (where the *xxxx* will be different for each installation) you will need:

- INSPortlet.jar
- insSmClient.jar
- insUtil.jar

From the <WebSphere_root>\lib directory, you will need:

- dynacache.jar

From <WPS_root>\shared\apps directory, you will need:

- portlet.jar
- portlet-api.jar
- wps.jar
- wpsportlets.jar
- servlet.jar

Write a JSP front-end, based on the Subscription View template

Use the Subscription Configuration template to write a JSP that lets the user edit the parameters for a subscription. This JSP is displayed when the user selects to add or modify a subscription. This JSP can be used for both cases, because the subscription parameters are the same whether a user is specifying parameters for a new subscription or editing the parameters of an existing subscription. Users can also use this page to delete the subscription. The template is in the file SubscriptionConfig.jsp. Copy the contents into a new file, and complete the specific sections, as indicated by the comments within the file. Use the code walkthrough below as a guide. Refer to the file NewsSubscriptionConfig.jsp for an example of how the Configuration JSP is implemented for the news sample.

1. Import necessary libraries and beans.

```
<%@ page import="com.ibm.pvc.we.ins.portlets.*" %>
<%@ page import="org.apache.jetspeed.portlet.*" %>
<%@ page import="java.util.*" %>
```

2. Create variables for all of the subscription parameters. The news sample, NewsSubscriptionConfig.jsp, gets a list of available delivery channels and their names from the portlet session.
3. Get the selected subscription bean from the portlet request object. For example:

```
SubscriptionBean subBean =
   SubscriptionBean)portletRequest.getAttribute("selectedSubBean");
```

4. If there is a selected subscription bean, then the user is modifying an existing subscription. Use the getter methods of the subscription bean class to populate the subscription parameter variables. For example:

```
notificationOption = subBean.getNotificationOption();
```

5. If no selected subscription bean exists, then the user is adding a new subscription. Leave the parameter variables blank.
6. Create the OK and Cancel buttons.

Example 12-3 JSP code snippet

```
input type="image" name="<portletAPI:encodeNamespace value='OK'/>"
         VALUE="<portletAPI:text key="Sub_OK" bundle="nls.INS.PortletMessages"
/>"
```

```
      SRC= "<%=portletResponse.encodeURL("/images/INS/header_ok.gif")%>"
      align="absmiddle"
 alt="<portletAPI:text
 key="Sub_OK"
 bundle="nls.INS.PortletMessages" />"
      border="0"/>
 <a style="text-decoration: none;"
href="javascript:document.forms.<portletAPI:encodeNamespace
value='NewsSubscriptionConfig'/>.submit()">
     <span class="wpsTaskIconText">
     <portletAPI:text key="Sub_OK" bundle="nls.INS.PortletMessages" />
     </span>
 </a>
 <img src="<%= portletResponse.encodeURL("/images/INS/divider2.gif")%>"
align="absmiddle" border="0" alt=""/>
 <a style="text-decoration: none;" href="<%= cancelURI %>">
     <img src="<%= portletResponse.encodeURL
   ("/images/INS/header_cancel.gif") %>"
   align="absmiddle"
   border="0"
   alt="<portletAPI:text
   key="Sub_Cancel"
   bundle="nls.INS.PortletMessages"/>"/>
     <span class="wpsTaskIconText">
     <portletAPI:text key="Sub_Cancel"
   bundle="nls.INS.PortletMessages"/>
     </span>
 </a>
```

7. If there is a selected subscription bean, then the user is modifying or deleting an existing subscription. Create a Delete button.

Example 12-4 JSP code snippet

```
<% if (subBeanIndex != null) { %>
         <img src="<%= portletResponse.encodeURL
      ("/images/INS/divider2.gif")%>"
      align="absmiddle"
      border="0"
      alt=""/>
         <a style="text-decoration: none;" href="<%= deleteURI %>">
               <img src="<%= portletResponse.encodeURL
         ("/images/INS/delete_box.gif") %>"
         align="absmiddle"
         border="0"
         alt="<portletAPI:text
         key="Sub_Delete_Subscription"
         bundle="nls.INS.PortletMessages"/>"/>
               <span class="wpsTaskIconText">
```

```
      <portletAPI:text key="Sub_Delete_Subscription"
bundle="nls.INS.PortletMessages"/>
      </span>
 </a>
```

8. Display text entry or other type of input widget for each subscription parameter setting. Populate the input with the value of the corresponding variable. If the user is adding a new subscription, the variable value is blank. If the user is modifying an existing subscription, the variable value is that of the subscription and will be modified by the user.

Example 12-5 JSP code snippet

```
<portletAPI:text key="notification_option" bundle="nls.INS.PortletMessages" />
<INPUT TYPE="radio" VALUE="once", NAME="notificationOption"
<%=notifyOnce%>>
<portletAPI:text key="notification_once" bundle="nls.INS.PortletMessages" />
<INPUT TYPE="radio" VALUE="always", NAME="notificationOption"
<%=notifyAlways%>>
<portletAPI:text key="notification_always" bundle="nls.INS.PortletMessages" />
```

Write a custom trigger handler

When a content match occurs, Subscription Manager uses trigger handlers to notify the appropriate user via the Notification Manager. Trigger handlers process the matched content according to subscription criteria and rules, sending a notification request to Notification Manager. The specifics of how, where, and when the user is notified are controlled by the trigger handler. Each subscription has a trigger handler associated with it.

To develop a custom trigger handler, you must write a Java class that extends the TriggerHandler class. You can view the source code for the sample stock or weather trigger handlers to see an example of how to extend TriggerHandler. The easiest way to create a custom trigger handler is to modify one of the sample trigger handlers to match your needs.

To write a custom trigger handler, complete the following:

1. Create a trigger handler file or copy an existing one.
2. Import the required package files into the class.
3. Create a class that extends the Trigger Handler class.
4. Create variables for resource bundles, delivery options, subscription matching parameters, and strings for constructing the notification.
5. Implement the doGet() method to retrieve information from the request parameters and put them in the response object.

6. Implement the doPut() method to add a new subscription to Subscription Manager.
    a. In the doPut() method, create the matching selector statement for the subscription variables.
    b. In the doPut() method, create the subscription using the subscribe method and activate the subscription using the activate method.
7. Implement the doPost() method to enable Intelligent Notification Services to update an existing subscription, which consists of updating the correct parameters and saving the subscription.
8. Implement the handleMatch() method to build the notification request when a match occurs.
    – In the handleMatch() method, create the Notification Manager client, build the delivery notification object, and submit the message to Notification Manager.
9. Deploy the trigger handler.

Deploy the new subscription portlet

An Intelligent Notification Services administrator must follow these steps to deploy a custom subscription application:

1. Once the war file has been completed, log on to the WebSphere Portal as an administrative user, and install the new .war file using the Administration page.
2. Give Intelligent Notification Services users access to the new subscription portlet by giving the user group access to the new portlet.
3. Add the new portlet to the My Subscriptions page, and lock it down.
4. Sign on as a test user to make sure the page and new portlet display properly.

13

Using Workplace Client Technology, Micro Edition

The objective of this chapter is to describe the design and implementation of the ITSO Railways Ticketing application, a pervasive solution that operates in an intermittently connected environment. This chapter provides an overview of the development of this application built using Workplace Client Technology, Micro Edition.

This chapter describes the following:

- The architecture of the ITSO Railways Ticketing application
- The design of the ITSO Railways Ticketing application
- The implementation of the ITSO Railways Ticketing application
- Key aspects of developing the ITSO Railways Ticketing application
- Testing and running the ITSO Railways Ticketing application in the development environment

13.1 Architectural overview model

The following diagram is the Runtime pattern we implemented in this particular scenario.

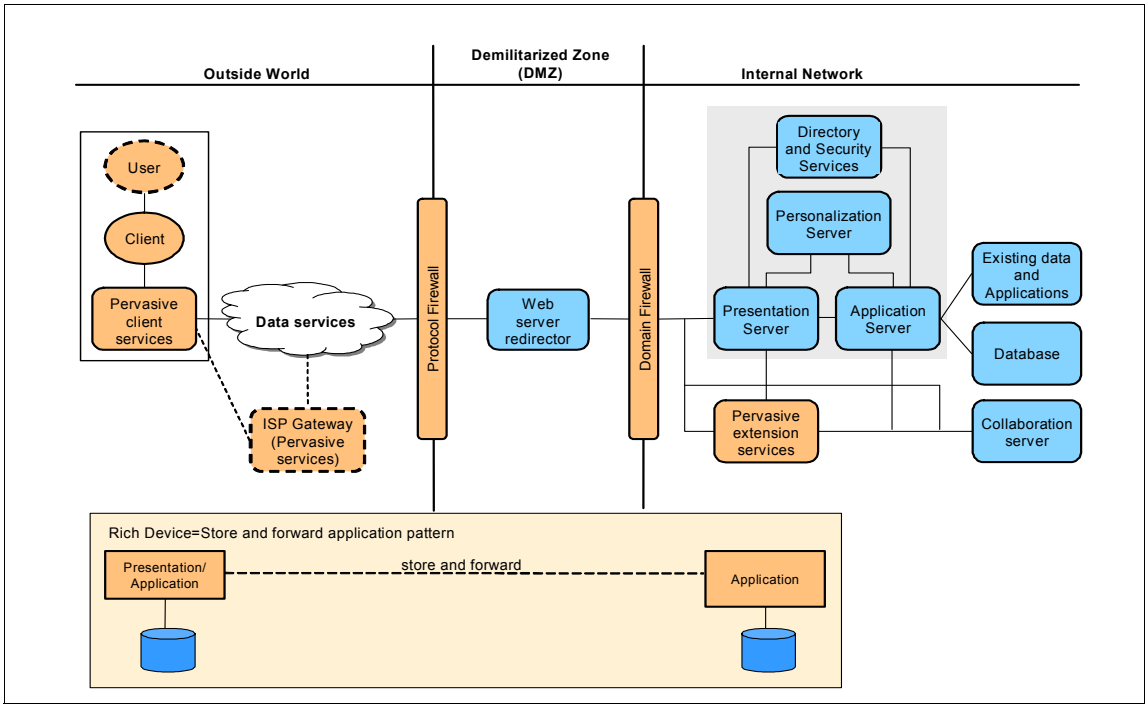


Figure 13-1 Runtime pattern for the ticketing scenario implementation using Workplace Client Technology, Micro Edition

This chapter describes how the Train Station Ticketing application is extended for use by mobile workers, in this case train conductors. The architectural overview model, shown in Figure 13-2 on page 285, demonstrates the entire system and shows the high-level components involved in the system. The user interacts with the ITSO Railways Ticketing application that runs standalone on the mobile device. The ITSO Railways Ticketing application accesses the local database, and when a ticket is purchased, adds a ticket purchase record to the local database. Also, when tickets are purchased the application places a message on the message queue. Any data updates that occur on the mobile device are synchronized to the pervasive server (when the device connects to the network). These data changes are then forwarded to the enterprise application (Train Station Ticketing) for processing.

This chapter focuses on the client-based application, which is highlighted in Figure 13-2.

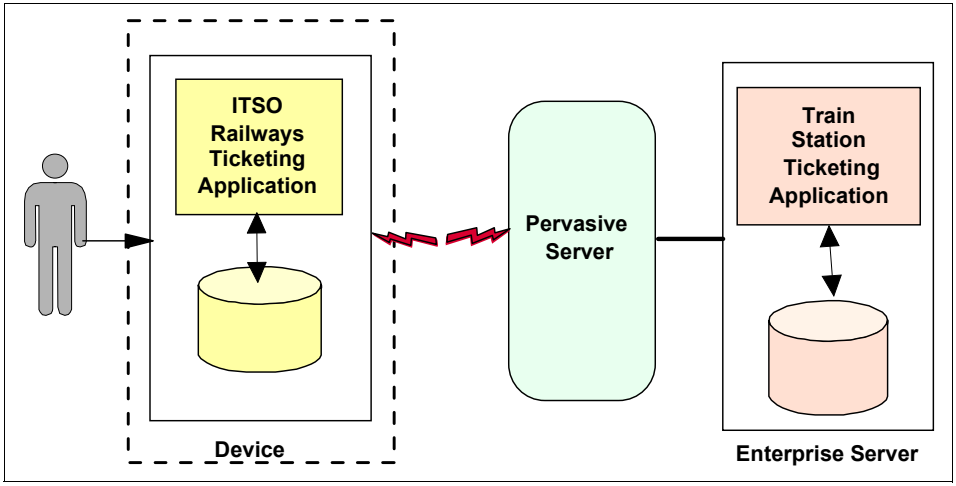


Figure 13-2 Architectural overview model

This scenario is based on the Pervasive runtime pattern for Robust Device Based Solutions runtime pattern and it is implemented using the Workplace Client Technology, Micro Edition, along with DB2 Everyplace, which provides a local database and data synchronization between the device and the server and MQ Everyplace, which provides the messaging service.

We chose Workplace Client Technology, Micro Edition because the ITSO Railways wanted:

- The solution to be able to operate in a standalone mode, which means the application will contain complex business logic.
- The solution to be able to access a local data store.
- The solution to be implemented using a standards based technology, in this case Java.
- The solution be portable across a variety of devices including laptops and PDAs.
- To reuse the existing Java skills the development team already had and possibly to reuse existing Train Station Ticketing application artifacts.
- A robust user interface that Java Server Pages (JSPs) provide.
- The solution to integrate with the existing back-end application, which currently uses a messaging model and has well-defined interfaces.

The ticket prices change frequently because of promotions and discounts. This information is stored in the ITSO Railways enterprise database. To simplify the management of the data used by the new Ticketing application, the team decided to provide a local database and synchronize the enterprise data with the data on the device. DB2 Everyplace meets these needs by providing both a local database (for a variety of pervasive devices) and a data synchronization server.

Ticket purchases are defined as messages and processed by the Train Station Ticketing application. ITSO Railways wanted to extend that model to the pervasive solution. MQ Everyplace provides the messaging service that fits this model. The new Ticketing application will create the ticket purchase messages required by the Train Station Ticketing application.

The actual Product mapping for this scenario can be found on the following diagram.

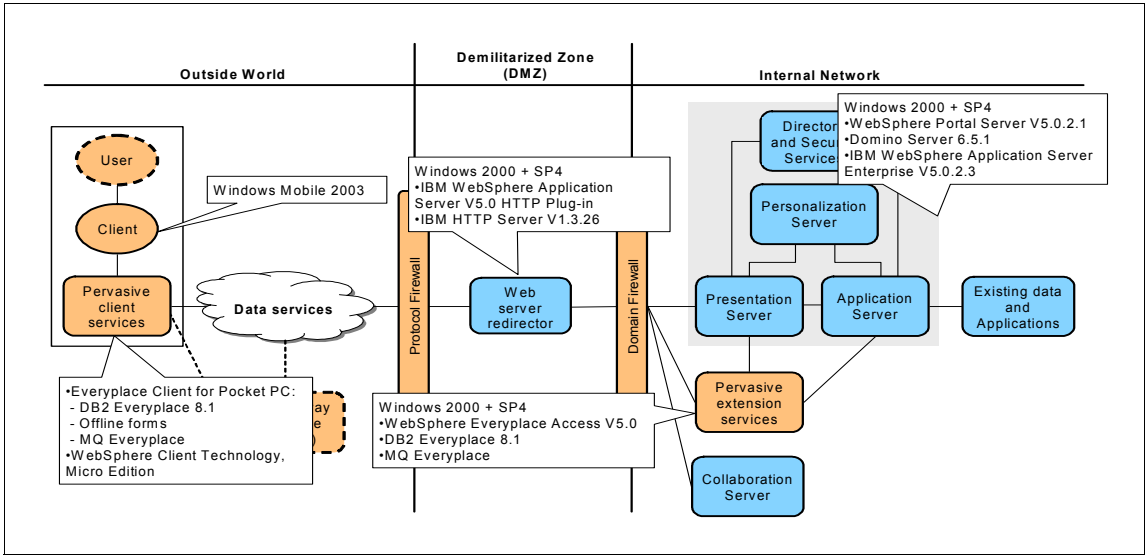


Figure 13-3 Rich Device=Store and forward::Product mapping=Windows

13.2 System design overview

The ITSO Railways Ticketing application will be used by the train conductor while the train travels between stations. The conductor will use this application when he sell tickets to passengers that have boarded the train without previously purchasing a ticket. After determining the passenger's destination, the train conductor will use the Ticketing application to obtain the ticket price(s) for that destination and to enter credit card information as the means of payment. In the

future, the conductor will be able to verify the credit card is not on the credit card black list. We did not implement that portion of the application.

Figure 13-4 shows the high-level system design, which includes the pervasive device and the server environment. Three tiers shown in Figure 13-4 are:

- The ITSO Railways Ticketing application (in Tier 1) runs on the mobile device. This application interacts with the local database and the messaging service.
- The pervasive server (logical tier) contains the runtime middleware, which supports the sharing of data and messages between the device and the enterprise server. It contains the DB2 Everyplace Synchronization Server and MQ Everyplace. The DB2 Everyplace sync server is used to synchronize enterprise data (for this application) to the device. The MQ Everyplace queue manager gets messages from the mobile device's message queue.
- The third tier is the ITSO Railways enterprise server environment, which contains the business application and the business data. This tier contains the Train Station Ticketing application, which processes the ticket purchase messages, creates interfaces to other business applications, and updates the enterprise database as needed.

In this scenario the data synchronization and the message transfer occur over the air whenever a connection is available.

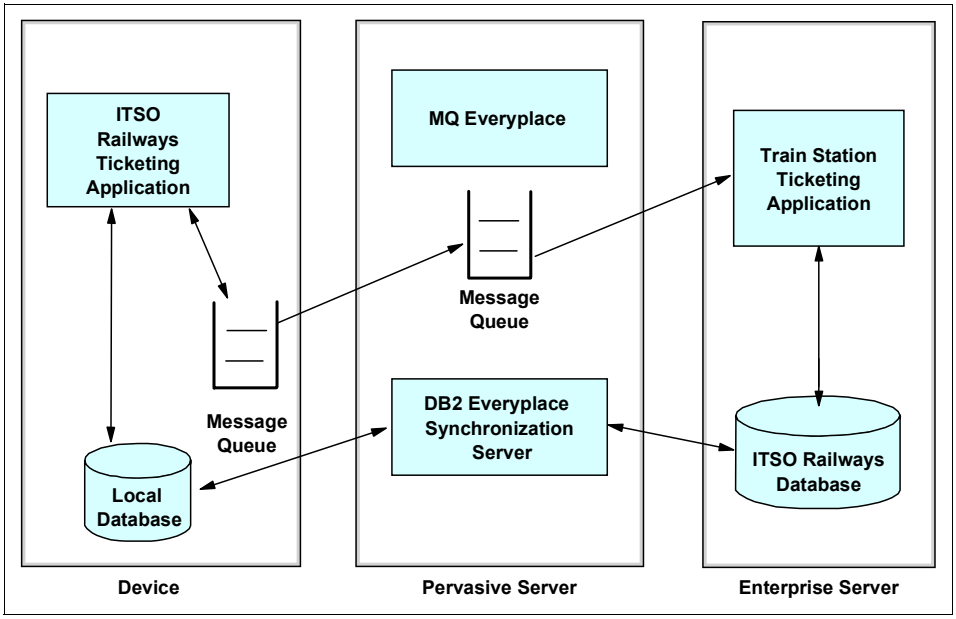


Figure 13-4 System overview

13.2.1 Component model

This component model shows a breakdown of the client application. In this sample application the Ticketing application follows the Model View Controller paradigm, where the controller is a servlet, the view is provided by Java Server Pages (JSPs), and the model consists of classes that interact with the database and the messaging service.

Figure 13-5 shows the component diagram of the sample application. In the sample application, the Ticketing Servlet parses the request from the client browser and determines the actions to take. The TicketingServlet invokes methods on the TicketingService to retrieve data from the local database, update the local database, and to put messages on the message queue.

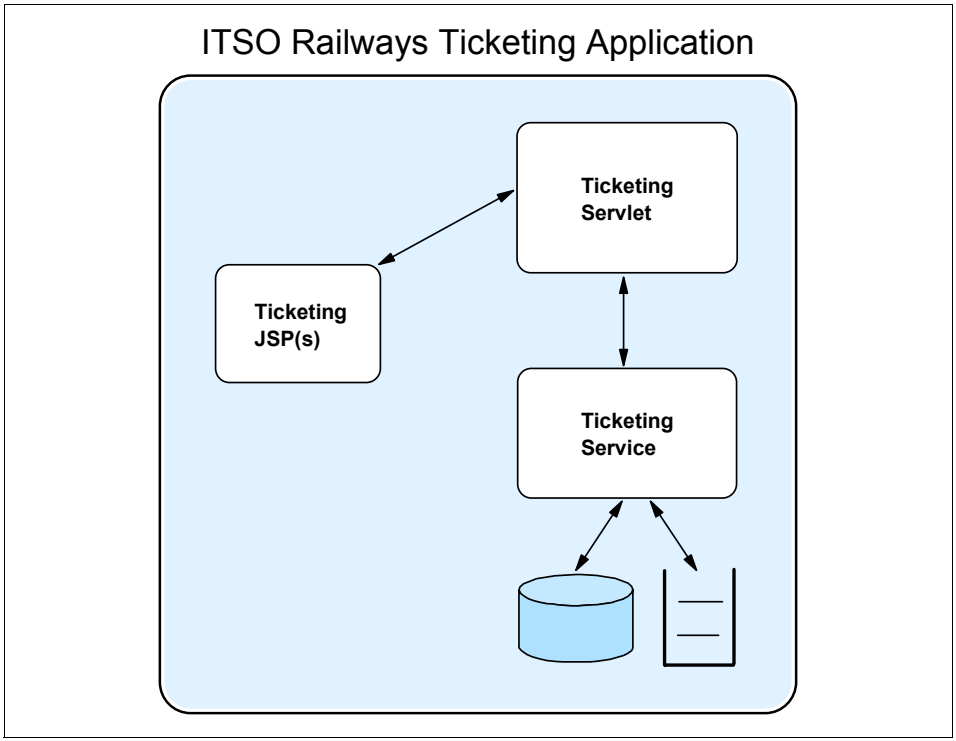


Figure 13-5 Component model

MQ Everyplace is the messaging service used in this application, and DB2 Everyplace provides the small footprint database and the data synchronization service.

13.2.2 Object model

This section provides an object model for our ITSO Railways Ticketing application. The class diagrams show the classes used within the application. The interaction diagram shows the classes directly involved when the conductor starts the application in order to sell a ticket.

Class diagram

Figure 13-6 on page 290 and Figure 13-7 on page 292 show the class diagrams for the ITSO Railways Ticketing application.

The classes shown in Figure 13-6 on page 290:

- TicketingServlet - Controls the flow of the application and drives the interaction with the view, the database, and the message service.
- selectorigin.jsp - Displays the possible origins for the ticket to the user.
- selectdestination.jsp - Displays the possible destinations for the ticket to the user.
- purchasetickets.jsp - Displays the types of ticket that can be purchased and accepts the purchaser's credit card information.
- orderconfirmed.jsp - Displays the summary of the ticketing information and is used to submit the ticket purchase.
- TicketingService - Is an interface defining the methods available for interacting with the database and the message queue.
- ManagedService - (From the OSGi framework) is a service that receives configuration data from a Configuration Admin service.
- Messages - Retrieves strings from the Resource Bundle and formats them.
- TicketingPresentationBundle - Is the activator bundle. At startup time the framework creates an instance of this class and calls its start() method. The activator can then publish services, start its own threads, etc. When the bundle shuts down, the framework calls the activator's stop() method. While the bundle is shutting down, the activator can release resources that were obtained after the start method was called and revoke any services it has published.
- TicketingServiceTracker - Acts as a pass through for the TicketingService to allow an implementation to always exist, but will not attempt to invoke the service unless actually installed.
- ServiceTracker - (From the OSGi framework) a class that simplifies using services from the framework's service registry. It can be used to track all services in the framework's service registry that match the specified search criteria.

- TicketingServiceImpl - Implements the TicketingService methods and provides access to the database and the MQe message service.

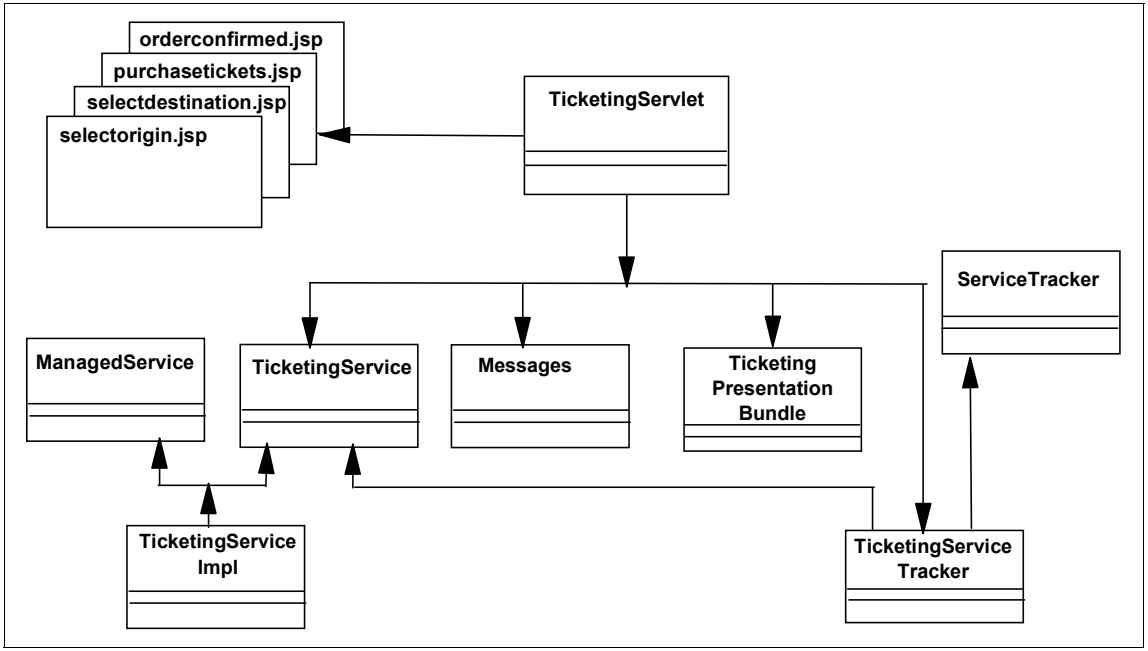


Figure 13-6 Class diagram - 1

The classes shown in Figure 13-7 on page 292 are associated with the TicketingServiceImpl:

- TicketingTransportListener - A listener for messages being sent by the MQ server.
- TicketingMessage - (Implements IMessage and extends MQeMsgObject) provides the getters and setters for this specific message object:
  - IMessage - Defines the basic methods required for messaging to be used
  - MQeMsgObject - The basic message object within MQe used to convey information from one application to another via a sequence of queue managers
- TicketingSystemConstants - An interface that defines the constants used to access system properties as well as those constants used to define queue and queue manager names.
- TicketingMsgConstants - An interface that defines the fields in a message.

- BaseMQeTransport - (Implements ITransport and MQeMessageInterface and extend MQe) an implementation of the ITransport using MQ Everyplace as the transport.
  - ITransport - Defines the basic methods expected of a transport interface
  - MQeMessageInterface - MQe's message interface
  - MQe - Base MQ Everyplace class containing various useful symbolic constant definitions and utility methods to assist with MQe programming
- ServiceRegistration - (From the OSGi framework) an object for the private use of the registering bundle, which should not be shared with other bundles.
- BundleContext - (From the OSGi framework) the context is used to grant access to other methods so that this bundle can interact with the framework.
- TicketingDatabase - Is the database access wrapper providing the methods for accessing and updating of the local database.
- Origin - A class that reflects the elements from the Origin table, which are used in the application.
- Destination - A class that reflects the elements from the Destination table, which are used in the application.
- BaseMessageEnvelope - Implementation of the IMessageEnvelope.
- IMessageEnvelope - Defines the methods required of a message container.
- ServiceReference - (From the OSGi framework) the framework returns ServiceReference objects from the BundleContext.getServiceReference and BundleContext.getServiceReference methods. It may be shared between bundles and can be used to examine the properties of the service and to get the service object.
- ConfigruationAdmin - (From the OSGi framework) this interface is used to store bundle configuration data persistently.
- Configuration - (From the OSGi framework) contains a configuration dictionary and allows the properties to be updated via this object.

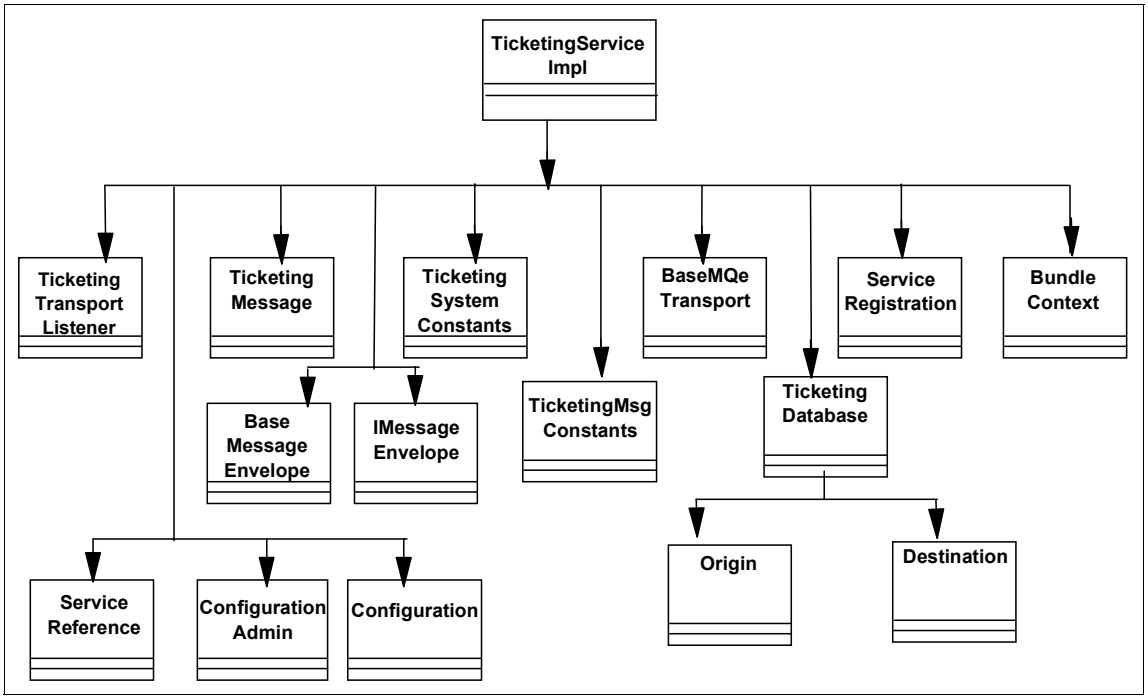


Figure 13-7 Class diagram - 2

Interaction diagram

The interaction (sequence) diagram, shown in Figure 13-8 on page 294, depicts the conductor using the ITSO Ticketing application. Once the conductor starts, the application flow of controls occurs, as follows:

1. The TicketingServlet invokes the TicketingService's getOriginList() method.
2. The Ticketing Service invokes the TicketingDatabase's getOriginList() method, which retrieves the origins from the Origin table.
3. The TicketingServlet invokes the ServletContext's getRequestDispatcher() method to display the selectorigin view to the conductor. The conductor then selects the appropriate origin and clicks Accept. The action is set to SelectDestination.
4. The TicketingServlet invokes the TicketingService's getDestinationForOrigin().
5. The TicketingService invokes TicketingDatabase's getDestinationForOrigin(), which retrieves the possible destinations from the Destination table.
6. The TicketingServlet invokes the ServletContext's getRequestDispatcher() method to display the selectdestination view to the conductor. The conductor

then selects the appropriate destination and clicks Accept. The action is set to PurchaseTickets.

7. The TicketingServlet invokes the TicketingService's getJourneyPrice().
8. The TicketingService invokes TicketingDatabase's getJourneyPrice(), which retrieves the prices from the Destination table.

   Note that the TicketingServlet invokes the getJourneyPrice() sequence twice—once to determine the prices for singles and once for returns.
9. The TicketingServlet invokes the ServletContext's getRequestDispatcher() method to display the purchaseticket view. The conductor then selects the ticket type and collects the customer's credit card information and clicks Submit. The action is set to SubmitOrder.
10. The TicketingServlet invokes the TicketingService's getJourneyPrice() method to obtain the actual price for this ticket using the selected origin and destination as input.
11. The TicketingService invokes the TicketingDatabase's getJourneyPrice() method, which retrieves the price from the Destination table specific to the selected origin and destination.
12. The TicketingServlet invokes the TicketingService's submitOrder() method with the ticket information.
13. The TicketingService invokes TicketingDatabase's storeTicketOrder() method, which invokes the getNextOrderID() to obtain an order ID and then creates a new ticket purchase in the Tickets table. Once the Tickets table has been updated successfully, a message is created and put on the message queue, which is discussed later in this document.
14. The TicketingServlet invokes the ServletContext's getRequestDispatcher() method to display the orderconfirmed view to the conductor. The conductor can then reuse the application for other ticket purchases.

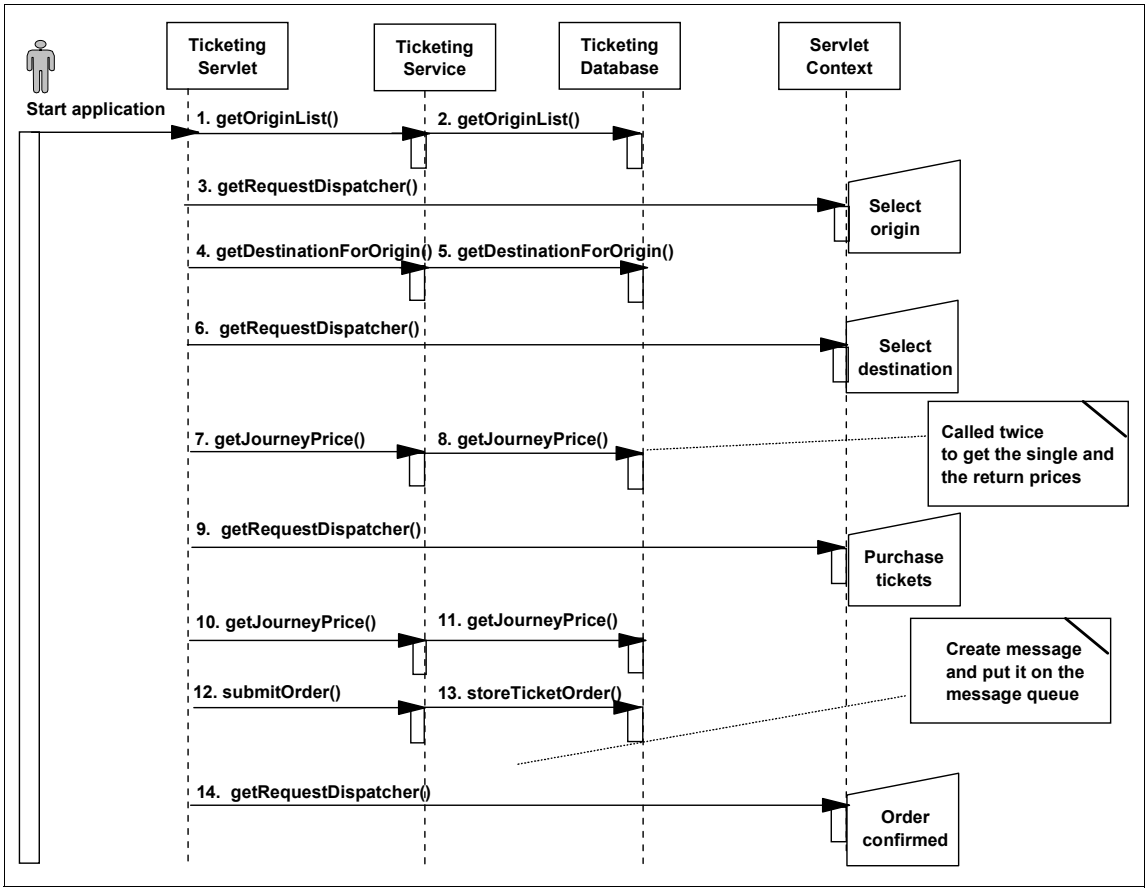


Figure 13-8 Interaction diagram

13.3 Application design

This section describes the architecture and design decisions that went into building our sample application. This sample application is an extension to the Train Station Ticketing application and is targeted to run on mobile devices that operate in an intermittently connected mode. The sample application must therefore be able to operate in standalone mode, thereby allowing the conductor to use the application when connectivity is not available.

There are various architectural decisions that went into designing this application. These include:

- Integration pattern

- Application pattern and Pervasive access pattern
- Design pattern

Integration pattern

The first high-level architectural decision was to select the Access Integration pattern because it met both the business and IT requirements for the solution. We found the pervasive device support service within the access integration services to be particularly appropriate for this architecture. The pervasive device Support service targets the needs introduced by pervasive devices.

Application pattern

The application pattern selected for this scenario is the Pervasive Device Adapter pattern, which provides a structure for extending the reach of the enterprise application (Train Station Ticketing) to pervasive devices.

This scenario uses both the Client to Pervasive Device Interaction pattern and the Pervasive Device to Server Exchange pattern. The client to pervasive device interaction is used because the client, in this case the train conductor, interacts with the application located on the device. The Pervasive Device to Server Exchange pattern is used because the device must exchange data (in the local database) and messages (stored in the message queue) with the server when network connectivity is available.

Design pattern

The classic Model-View-Controller design pattern maps nicely to this application, as was shown in Figure 13-5 on page 288. The model represents the application object that implements the access to the application data and business logic. The View is responsible for formatting the application results and dynamic page construction. The Controller is responsible for receiving the client request, invoking the appropriate business logic, and (based on the results) selecting the appropriate view to be presented to the user.

13.4 Sample application development

This section discusses the creation of the sample application, the IBM products, and the development environment used to build it. This is a standalone Web application running on the mobile device using Workplace Client Technology, Micro Edition.

WebSphere Client Technology, Micro Edition is a platform for deploying mobile applications that can operate in a standalone mode on the pervasive device. WebSphere Client Technology, Micro Edition incorporates WebSphere

Everyplace Micro Environment, which provides a Java 2 Micro Edition (J2ME) powered runtime environment that supports most popular devices, and for smaller devices WebSphere Everyplace Micro Environment delivers a Java Runtime Environment (JRE) that meets Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP 2.0) specifications. For larger devices such as PDAs and handheld computers, WebSphere Everyplace Micro Environment includes the Connected Device Configuration (CDC) Foundation Profile and Personal Profile. WebSphere Client Technology, Micro Edition also provides the ability to use custom profiles for applications where footprint and performance are of major importance.

WebSphere Client Technology, Micro Edition contains a component-based architecture provided by Service Management Framework (SMF). SMF is IBM's implementation of the Open Service Gateway Initiative (OSGi) Service Platform Release 3 specification. The OSGi Alliance defines and promotes this open standard framework for network delivery of managed services to local networks and devices. This approach allows devices to easily be maintained wherever they are and whenever they connect to the network.

WebSphere Client Technology, Micro Edition supports assured messaging by supporting MQ Everyplace and Java Messaging Service (JMS). It also supports IBM DB Everyplace, which provides a local database; and a synchronization service, which allows the local database to sync with the enterprise database when a network connection is available.

To create a WebSphere Client Technology, Micro Edition based application, first set up the development environment with the essential WebSphere Studio products, plug-ins, and features. For this sample, we used WebSphere Studio Application Developer Version 5.1.1 as the base. Next WebSphere Studio Device Developer Version 5.7 was installed as an extension to WebSphere Studio Application Developer. During the WebSphere Studio Device Developer installation you will see this question displayed: `WebSphere Studio Application Developer is installed on the system. The function in WebSphere Studio Device Developer can be enabled in WebSphere Studio Application Developer.` To incorporate the two products, select **Yes, enable it** and click **Next**.

Please note that WebSphere Studio Site Developer could have been used instead of WebSphere Studio Application Developer as the base development environment.

Because we are creating a WebSphere Client Technology, Micro Edition Web based application, we installed the following WebSphere Client Technology,

Micro Edition features on the base (consisting of WebSphere Studio Application Developer and WebSphere Studio Device Developer) in this order:

1. SMF Bundle Development Kit V5.7
2. Extension Services V5.7
3. Application Tools for Extension Services V5.7

Because we are using MQe and DB2e we also installed (from the technologies directory):

- DB2 Everyplace V8.1.4
- MQ Everyplace V2.0.1

13.4.1 Creating the application

The projects for the application artifacts were created using WebSphere Studio Application Developer. The two projects are named ITSO Railways Ticketing Web Application and ITSO Railways Ticketing Service. To create the project select **File** → **New** → **Other**, which invokes the New Project wizard. To create the ITSO Railways Ticketing Web Application project with the New Project wizard, select **Extension Services** with a project type of Extension Services Web Project. Select the Platform Profile of **Extension Services: jclFoundation (5.7.0)** and use the default application services. Because this application is similar to the Order Entry example included with the product, we mirrored their approach and package structure, and reused many of the artifacts that existed in the Order Entry application.

An Extension Services project is different from a standard Java project because the Java Build Path is automatically updated to reflect the project's Platform Profile and Application Service settings. Also, the manifest file located in the Extension Services Content folder is automatically updated with the appropriate OSGi metadata for the project and a bundle activator is created, which is required for registering the Web application with the Web container at runtime. Included is a default bundle activator for this purpose.

The Ticketing Web application relies on numerous classes within the ITSO Railways Ticketing Service Project to interact with the OSGi Framework, SMF services, and the MQ Everyplace messaging services. Even though they are important to this application, we will be focusing primarily on the Java classes with the ITSO Railways Ticketing Web application. As stated earlier, we used the Order Entry example that comes with WebSphere Client Technology, Micro Edition as the base for our sample.

13.4.2 Creating the service interface

The TicketingService is an interface that defines various services that the TicketingServlet uses to interact with the database and the messaging service. The TicketingService interface is created within the ITSO Railways Ticketing Service project and within the com.ibm.itsorailways.ticketing.service package we created. To create the interface select **File** → **New** → **Interface**. Figure 13-9 shows the Java Interface wizard. Enter the interface name `TicketingService` and click **Finish**.

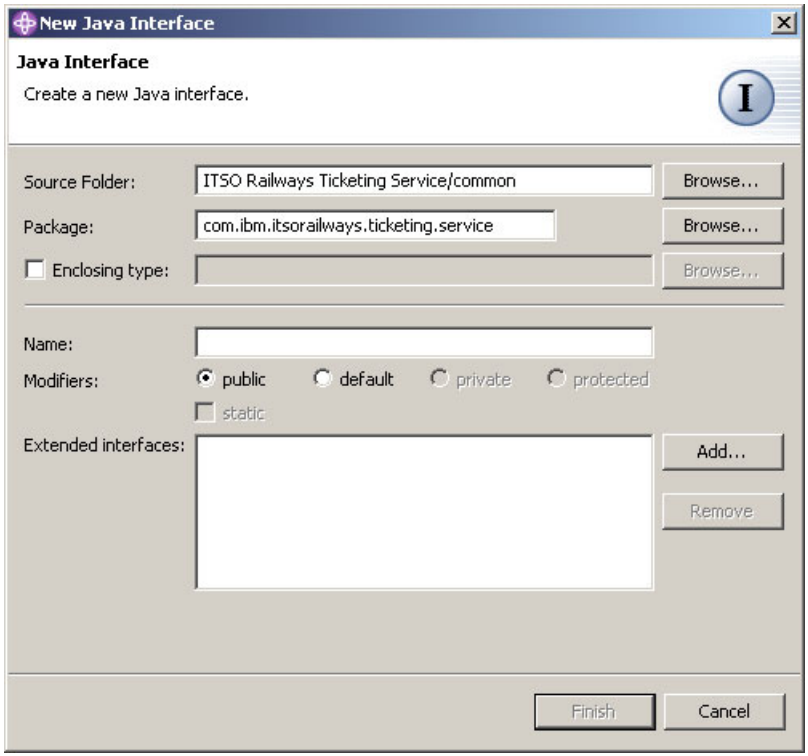


Figure 13-9 Create interface

The methods within this interface are shown in Example 13-1.

Example 13-1 TicketingService methods

```
/**
submits the purchase ticket transaction
**/
public void submitOrder(String origin, String destination, String singles,
String returns, String cardHolder, String cardNumber, String expireMonth,
String expireYear, String totalCost);
```

```
/**
get the possible origins form the Origin table
@return Vector with the possible origins
**/
public Vector getOriginList();
/**
gets the destinations from the Destination table
@return Vector with the possible destinations
**/
public Vector getDestinationsForOrigin(String origin);
/**
get the price for the journey from the destination table
@return an integer with the amount
public int getJourneyPrice(String origin, String destination);
/**
Set the configuration for the service.  The method is used for external
configuration of the service.   The new configuration information is used to
update our current configuration, and that new current configuration is saved
via the ConfigurationAdmin service
@param props The new configuration properties
*/
public void setConfig(java.util.Dictionary config );
/**
Return a copy of our current configuration
@return Dictionary A copy of our current configuration values
*/
public Dictionary getConfig();
```

13.4.3 Create the servlet

The heart of the application is the TicketingServlet, which controls the application flow and drives the interaction between the application and the user (in this case a train conductor). Within the ITSO Railways Ticketing Web Application project open the JavaSource and then open and select the **com.ibm.itsorailways.client.gui.servlet** package we already created. From the Toolbar menu select **File** → **New** → **Other**. In the New wizard select **Web** in the left window and **Servlet** in the right window. Give the new servlet the name `TicketingServlet` and click **Finish**.

The doRequest() method provides the control flow for the application. Each user request is a parameter, which is put into the string named action. The action is used to inform the servlet of the user's response to the screen previously displayed. The possible action values are:

- SelectOrigin
- SelectDestination
- PurchaseTickets

- SubmitOrder

The TicketingServlet uses the TicketingService to obtain data from the local database for display by the appropriate JSP. The doRequest() method is shown in Example 13-2, which nicely maps to the interaction diagram shown in Figure 13-8 on page 294.

Example 13-2 TicketingServlet doRequest()

```
protected void doRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
String nextpage = "/index.jsp";
//get the browser's locale
Final Locale locale = Util.getLocale(request);
Messages.setLocale(locale);
//
// get the action selected by the user
//
String action = (String) request.getParameter("action");
//
// for the SelectOrigin action, get the journey origins from the databse
// and prepare the selectorigin.jsp for display
//
If (action.equals("SelectOrigin")) {
   request.setAttribute( "OriginList", service.getOriginList());
   nextpage = "/selectorigin.jsp";
//
// for the SelectDestination action, get the possible destinations from the
// database and prepare the selectdestination.jsp for display
//
} else if (action.equals("SelectDestination")){
   String origin = (String)request.getParameter("Origin");
   request.setAttribute( "DestinationList",
service.getDestinationsForOrigin(origin));
   nextpage = "/selectdestination.jsp";
//
// for the PurchaseTickets action, get the possible prices from the
// database and prepare the purchasetickets.jsp for display
//
} else if (action.equals("PurchaseTickets")){
   String origin      = request.getParameter("Origin");
   String destination = request.getParameter("Destination");
   request.setAttribute("SinglePrice",
Integer.toString(service.getJourneyPrice(origin,destination)));
   request.setAttribute("ReturnPrice",
Integer.toString(2*service.getJourneyPrice(origin,destination)));
   nextpage = "/purchasetickets.jsp";
//
// for the SubmitOrder action, get the parameters from the request,
```

```
// get the price from the database, update the data base with the ticket and
// place a message on the queue and prepare the orderconfirmed.jsp for display
//
} else if (action.equals("SubmitOrder")) {
          String origin = request.getParameter("Origin");
          String destination = request.getParameter("Destination");
          String singles = request.getParameter("Singles");
          String returns = request.getParameter("Returns");
          String customerName = request.getParameter("CardHolder");
          String cardNumber   = request.getParameter("CardNumber");
          String expireMonth  = request.getParameter("ExpireMonth");
          String expireYear   = request.getParameter("ExpireYear");
          int price = service.getJourneyPrice(origin,destination);
          int totalCost = (Integer.parseInt(singles) * price) +
(Integer.parseInt(returns) * 2 * price);
          request.setAttribute("TotalPrice", Integer.toString(price));
          service.submitOrder(origin, destination, singles, returns,
customerName, cardNumber, expireMonth, expireYear,
Integer.toString(totalCost));
          nextpage = "/orderconfirmed.jsp";
// not used currently
}else if (action.equals("NewOrder")){
          nextpage = "/neworder.jsp";
       }
   RequestDispatcher rd = sc.getRequestDispatcher(nextpage);
   rd.forward(request, response);
} // end method
```

13.4.4 Creating a user interface

The Ticketing application uses various JSPs to display data to the user. Within the ITSO Railways Ticketing Web Application project and within the WebContent/Theme create the JSP by selecting **File** → **New** → **Other** to invoke the New Wizard. Within the New Wizard select **Web** in the left column and **JSP** in the right column, then click **Next**. Enter the JSP name `selectorigin` and click **Finish**. The selectorigin.jsp was developed using Page Designer. Figure 13-10 on page 302 shows a preview of the JSP, so you can see its structure.

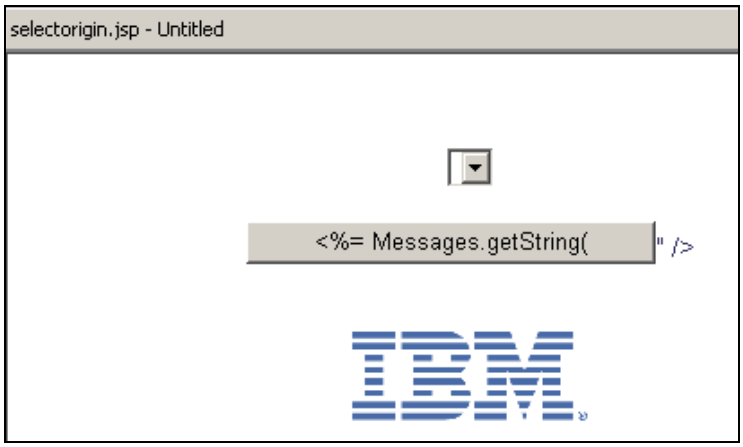


Figure 13-10 selectorigin.jsp preview

Figure 13-11 shows the design view of the JSP.

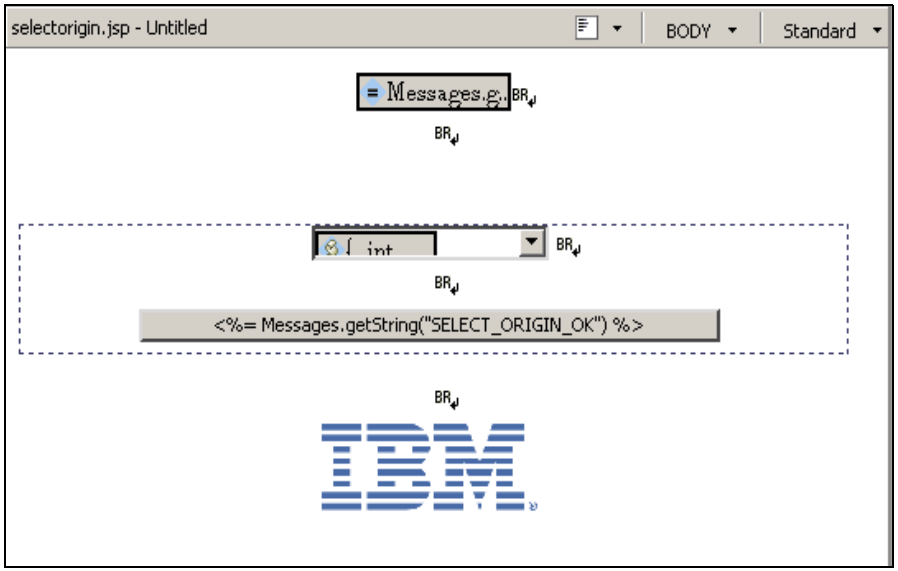


Figure 13-11 selectorigin.jsp design view

Example 13-3 on page 303 shows the code generated for the selectorigin.jsp. The setting of the action parameter is highlighted. The JSP uses the originVector to load the drop-down list with the possible ticket origins.

Example 13-3 selectorigin.jsp code

```
<%@ page contentType="text/html;charset=UTF-8"%>
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<%@ page import="com.ibm.itsorailways.ticketing.common.Origin" %>
<%@ page import="com.ibm.itsorailways.ticketing.common.Messages"%>
<%@ page import="java.util.Vector" %>
<%
      Vector originVector = (Vector)request.getAttribute( "OriginList" );
%>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="Expires" content="Wed, 01-Jan-2003 12:00:01 GMT">
<META name="GENERATOR" content="IBM WebSphere Studio">
<META http-equiv="Content-Style-Type" content="text/css">
<LINK href="theme/Master.css" rel="stylesheet"
   type="text/css">
<TITLE><%= Messages.getString("SELECT_ORIGIN_TITLE") %></TITLE>
</HEAD>
<BODY>
<center>
<font size=+1><b><%= Messages.getString("SELECT_ORIGIN_TITLE")
%></b></font><br><br>
<form action="TicketingServlet" method="post">
<select name="Origin">
<%
      {
         int count = originVector.size();
         for (int counter = 0;counter < count;counter++) {

            Origin o = (Origin)originVector.elementAt( counter );
            if (counter == 0) {
%>
               <option selected="selected" value="<%=o.getName()%>">
<%
            } else {
%>
               <option value="<%=o.getName()%>">
<%
            }
%>
            <%=o.getName()%>
            </option>

<%
         }
      }
```

```
%>
</select>
<br/><br/>
<input type=hidden name=action value="SelectDestination" />
<input type="submit" id="Submit" name="Submit" value="<%=
Messages.getString("SELECT_ORIGIN_OK") %>" />
</form><br>
<img src="images/ibm.gif" alt="ibm.gif">
</center>
</BODY>
</HTML>
```

13.4.5 Accessing the database

The TicketingServlet invokes methods on the TicketingService to access and update the database and to create messages. The TicketingService is implemented by the TicketingServiceImpl. To access and update the database, the TicketingServiceImpl invokes methods on the TicketingDatabase class. Figure 13-4 shows the getOriginList() method (in the TicketingDatabase class), which returns a vector of Origin objects. The Origin object has two elements, which are:

- Id - The ID for the origin
- Name - The literal name of the origin

Example 13-4 Access Origin table

```
public Vector getOriginList() throws IOException {
       Vector originVector = new Vector(8);
       Connection con = null;
       Statement  st  = null;
       ResultSet  rs  = null;
       try {
          con = getConnection();
          st  = con.createStatement();
          rs  = st.executeQuery("SELECT ORIGIN_ID, NAME FROM origins");
          while (rs.next())
          {
             Origin o = new Origin();
             o.setId(rs.getString(1));
             o.setName(rs.getString(2));
             originVector.addElement(o);
          }
       }catch (Exception e){
          System.out.println("******* ERROR " + e);
          e.printStackTrace();
          throw new IOException("EXCEPTION -- Error getting origin list");
       }finally {
```

```
            close(rs);
            close(st);
            close(con);
        }
        return originVector;
    }
```

13.4.6 Creating messages

The Ticketing application creates messages for the message queue once a ticket has been purchased. The TicketingServlet invokes the TicketingService's submitOrder() method, passing it the information from the ticket purchase. First the ticket purchase is recorded (by the TicketingDatabase's storeTicketOrder() method) in the ticket table and then a message is created. Example 13-5 shows the code for the submitOrder() method in the TicketingServiceImpl class.

Example 13-5 TicketServiceImpl submitOrder()

```
public void submitOrder(String origin, String destination,String singles,
   String returns,String cardHolder,String cardNumber,String
expireMonth,String expireYear,String totalCost) {
   String id = getDatabase().storeTicketOrder(origin, destination, singles,
returns, cardHolder, cardNumber, expireMonth, expireYear, totalCost);
   if (id != null)
   {
      BaseMQeTransport transport = getListener().getTransport();
      try {
//** create an instance of the TicketingMessage and set its element values
         TicketingMessage msg = new TicketingMessage();
         msg.setMessageReasonCode(TicketingMsgConstants.CUST_DATA_REQ_CODE);
         msg.setMessageID("ID01");
         msg.setOrderId(id);
         msg.setOrigin(origin);
         msg.setDestination(destination);
         msg.setSingles(singles);
         msg.setReturns(returns);
         msg.setCardHolder(cardHolder);
         msg.setCardNumber(cardNumber);
         msg.setExpireMonth(expireMonth);
         msg.setExpireYear(expireYear);
         msg.setTotalCost(totalCost);
//** identify the queue manager and put the message on the message queue
       msg.setReplyToQueueManagerName(transport.getDefaultQueueManagerName());
      msg.setReplyToQueueName(transport.getReplyQueueName());
      log("DEBUG -- SUBMIT ORDER [" + msg.getDisplayString() + "]");
      IMessageEnvelope env = new
BaseMessageEnvelope(transport.getServerQueueManagerName(),
transport.getRequestQueueName(), msg);
```

```
            getListener().getTransport().send(env);
        }catch (Exception e){
               e.printStackTrace();
        }//* end catch
    }//* end if null
}//* end method
```

For more details on using MQ Everyplace look at the IBM Web site:

http://www.ibm.com/software/wmqe

13.4.7 Setting up the launch configuration

The Ticketing application is a standalone Java application. The following steps must be completed to set up a launch configuration for the program that starts the Ticketing server:

1. In the SMF Perspective, select **Run** → **Run...** to display the Launch Configuration window.
2. Select **Java Application**. Click **New** and provide a name for the application, in this case `Ticketing`.
3. If the Project field is empty, select **Browse** and choose the **ITSO Railways Ticketing Service** project.
4. Click **Search** next to the Main class field to find the com.ibm.itsorailways.ticketing.server.Server server, as shown in Figure 13-12 on page 307.

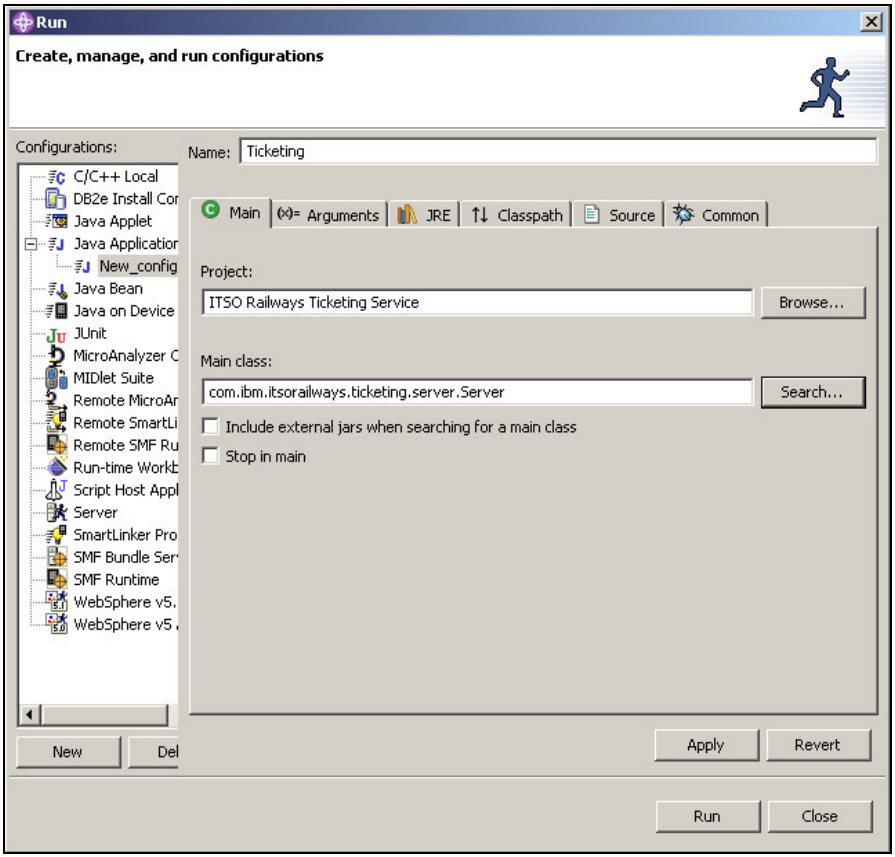


Figure 13-12 Launch configuration

5. In the Choose Main Type window, select **Server** and click **OK**.
6. Select the **JRE** tab.
7. Select the **Extension Services Generated JRE (JCL Foundation)** and click **Run**.

A J9 console window displays after you click Finish. Minimize the window and return to WebSphere Studio. If you close the window, the server terminates and the sample application will not function correctly.

13.4.8 Deploying the application

The following steps set up the Extension Services Local Server:

1. Select **File** → **New** → **Other**. The New Window is displayed.

2. Select **Server** in the left pane and **Server and Server Configuration** in the right pane. Click **Next**. The Create a new server and server configuration window is displayed.
3. Specify a name of `Ticketing` in the Server field. In the Server type field, select **Extension Services** → **Local Server**. Accept the default values for the other fields. Click **Next**.

   If you have not already created a Server Project you might be prompted to create one. Click **Yes** to create a new server project.
4. Accept all default values and click **Next**.
5. Select **Extension Services: jclFoundation (5.7.0)** as the Platform Profile, and click **Finish**.
6. Change to the Server perspective by selecting **Window** → **Perspective** → **Other** → **Server** and open the Server window and verify that the status of the server is Stopped.
7. Right-click your server in the Server configuration window and select **Start**.

Perform the following steps to submit the ITSO Railways Ticketing Service bundles to the Extension Services Local Server:

1. In the SMF perspective, right-click the **ITSO Railways Ticketing Service** project and select **SMF** → **Submit Bundle**.
2. In the resulting Target Submission window, select **Submit Jar**.
3. Select **Admin@localhost:8080/smf** in the Export Targets box.
4. Click **Finish** to submit the bundle to the bundle server.
5. Switch to the Bundle Server view and open **Admin@localhost:8080/smf** → **Bundles**. Verify that bundle names TicketingCommon and TicketingService are listed.

13.4.9 Launching the application

The following instructions are used to launch the Ticketing Web Application:

1. Change to the Server perspective.
2. In the Server Configuration window, right-click the **Ticketing server** and select **Add and remove projects**.
3. Select **ITSO Railways Ticketing Web Application** from the Available projects and click **Add**. Then click **Finish**.
4. In the Servers window, right-click **Ticketing** and select **Restart** to start the bundle server, the SMF runtime, and to load the Ticketing Web Application into the SMF runtime.

5. In the Navigator window, right-click **ITSO Railways Ticketing Web Application** and select **Run on Server** to launch a browser window that loads the application.

13.4.10 Using the ITSO Railways Ticketing application

You can use the Ticketing application to purchase tickets from the journey origin to a particular destination.

When the application starts it displays an initial splash screen. On this screen, click **Start** to begin the application.

Figure 13-13 Start Ticketing application

Next the ticketing Select your Origin screen displays. Select the origin of your trip from the drop-down list and click **Accept**.

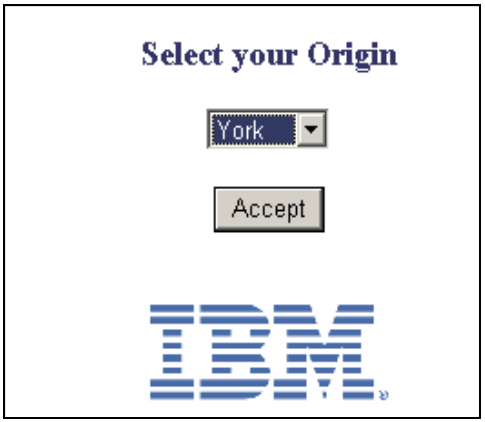


Figure 13-14 Ticketing origin selection

Now the Select your Destination screen displays. Select the destination from the drop-down list and click **Accept**.

Figure 13-15 Ticketing destination

The Purchase Tickets screen displays. Select a single ticket type and a return ticket type from the drop-down lists. Enter the card holder's name, the credit card number, and the expiry date (month and year). Any values for credit card information will be acceptable. Click **Submit**.

Figure 13-16 Ticketing purchase

When the ticketing purchase is processed, the Order Confirmed screen is displayed. Your can click **New order** to purchase more tickets.

Figure 13-17 Ticketing order confirmation

13.5 Deploying the application

This assumes you have installed the WebSphere Everyplace Client, Micro Edition runtime environment. To deploy the ITSO Railways Ticketing application to the target pervasive device perform the following general steps:

1. First define the directory location in the file system to store the application.
2. The ITSO Railways Ticketing Service project contents will be exported as SMF bundles using the SMF Bundle tool. There will be two bundles created, one for the common package and one for the service package.
3. The ITSO Railways Ticketing Web Application project is exported to the same directory as a WAB file.
4. Next a Platform Builder project is created that contains the application bundles. The Platform Builder project can be used to build a target platform for a specific device type that will run the application. To locate the Platform Builder tool select **File** → **New** → **Project** → **Extension Services** → **PlatformBuilder**. This tool is used to resolve any dependencies on other services or bundles.This tool will create the TicketingPlatformBuilder project in the workspace and will immediately launch a build to generate a target platform.
5. To run the target platform (created above) on your device, install the target platform using steps specific to the device. For example, if the output format was a zip file, unpack the platform package into a location in the device file system. Next execute the platform startup script, which is named StartSMF. This script launches the Extension Service platform and installs the application.
6. Once the platform is installed, you must configure the application and possibly modify the configuration for the runtime environment. After the configuration is done, open a browser and enter the location of the Ticketing application to start the application.

14

Chapter 14.

Timetable information by Voice

This chapter describes the rationale, design, and development of a voice access application to ITSO Railways timetable information by customers using phones. The pervasive solution provides ubiquitous access for customers over any telephone devise. The solution is based on the application of WebSphere Voice technologies including WebSphere Voice Response and WebSphere Voice Server as base technologies supporting WebSphere Voice Application Access. The solution approach is compatible with potential extensions to a broad range of services accessible by multi-channels including voice.

This chapter describes:

- Business requirements as use cases for the timetable application
- The Pattern for Voice Access
- The architecture of the ITSO Railways timetable information application
- Design information about the timetable information application and issues in developing a voice access application
- Development of the timetable information application using Toolkits available in WebSphere Studio Application Developer V5.1

14.1 Business requirements

The business requirements are for ITSO Railways to minimize the need for a call center to provide customers with information about its timetables. ITSO has provided a range of printed and Web-based materials for customers but recognizes that customers in transit may need to update their train schedules.

For this purpose ITSO Railways employs a call center operation, but because of potentially large volumes of calls at particular times of the day with only minimum use at other times the call center represents a major cost for ITSO Railways.

In order to meet its current requirements for cost-reduction, ITSO Railways intends to automate the phone access to its timetables. However, it also wants to develop a solution that will in the future provide a common interface for customers regardless of their access channel. In particular, it wants to ensure that the interface to its timetables is similar across phone, visual information presented on mobile devices and on the Internet (as well as potentially information kiosks in the stations).

The following diagram is the Runtime pattern we implemented in this particular scenario.

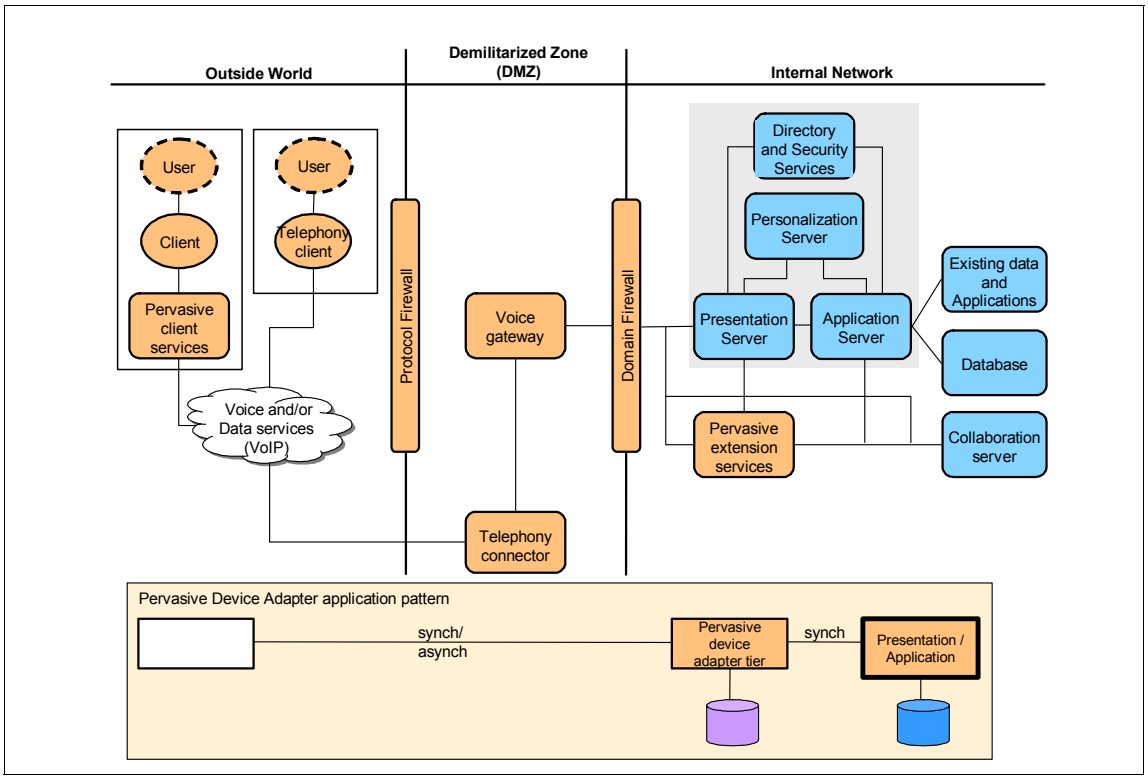


Figure 14-1 Runtime pattern for the Voice scenario

The Product mapping for the scenario that we used in this chapter can be seen in the following diagram.

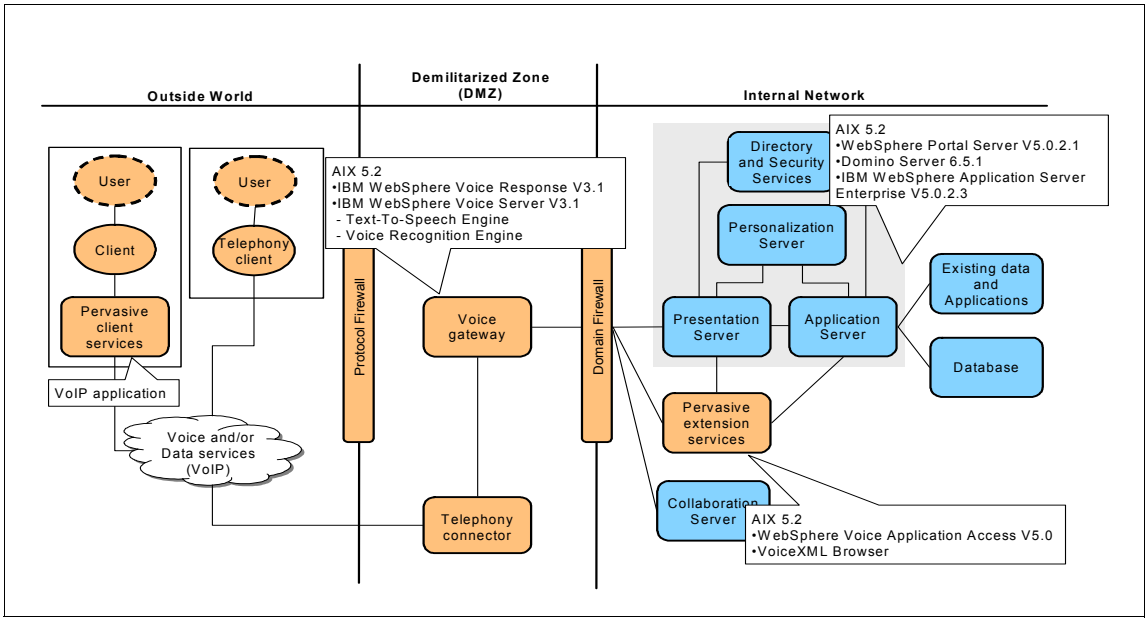


Figure 14-2 Pervasive Device Adapter=Voice::Product mapping=Windows

The high-level requirements are captured in the next figure. This shows a high-level use case for the application.

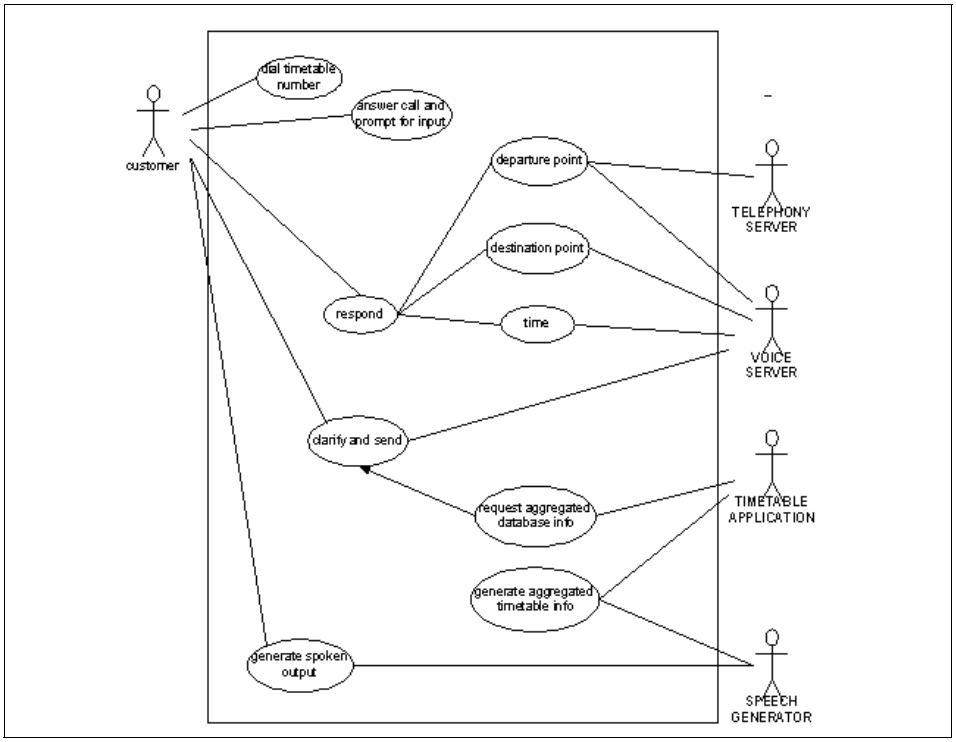


Figure 14-3 Use case diagram

The customer initiates a query by dialing a specific timetable information line. The call is answered (in this case by the IVR) and prompts the customer to enter her request (by collecting information about the caller's departure, destination, and required time). This information is sent to the automated speech recognition system that processes and verifies the input. The collected information is sent as a text request to the timetable application that generates a database request for location and timetables (and in the future for related information, for instance, the fare could be provided as well as duration of the journey depending on the database organization and completeness). The timetable information is generated as text for a speech production system to speak the data to the customer.

The current application requirements could be met by deploying a voice gateway and voice services for recognition and text to speech. However, the other business requirements indicate the need for a utilization of the Pattern for Voice Access because of future requirements for extensibility and multi-channel re-use

of the timetable data. The business decisions that impact the selection of the pattern components include:

- The application requires a telephone answer capability. This could be a PBX that links directly to a speech recognition component. In this case ITSO Railways wants extensibility to other potential applications in the future and therefore wants the system implemented by an Interactive Voice Response (IVR) that would provide the option to offer the customer a range of potential future services from ITSO Railways based on the same underlying infrastructure. So in this use case the IVR is the primary agent for call control, but would link to the existing ITSO Railway PBX system.
- The deployment could utilize either a grammar-based or natural language understanding (NLU) approach. In this case a standard grammar approach could work quite well because the timetable implementation could follow a standard visual application approach (select departure location, select destination location, select approximate time of travel). Alternatively, the interface could simply ask the customer for her travel details and, using an NLU approach, the system could understand a customer who said, “I want to go from Raleigh to Durham at 3 o’clock”. ITSO Railways is interested in NLU interfaces for the future but has decided to build the application using grammars using a menu-driven approach based on its current databases for locations and times. This will avoid the need to collect a range of user responses for the NLU and can be implemented with standard grammar constructions (for example, adding “please” etc. at the end of sentences).
- Another issue is related to the *persona* for the application and how speech should be presented to the user. The successful interaction of the caller with the automated technology will depend on a range of input variables (clarity of caller’s speech, background noise, etc.). The motivation of the user to “work” with the system will depend on their reaction to the persona of the interface. Different persona are required for different applications and company branding. In the case of ITSO Railways it believes that a natural human voice could meet their requirements more than text-to-speech (TTS) systems. However, they also note the value of TTS in the future for presenting more dynamic database information (for example, cause of delays, etc.) in which the information could not be recorded because it would vary. In the case of timetables, however, it is possible to record all locations and times and concatenate them in a naturally sounding presentation.
- Another business issue relates to the type of information that can be provided to the customer from the existing databases. There could be a range of possibilities including some simple value additions such as provision of duration of journey. There could also be a range of extensible services that could permit the customer to book a ticket and pay for it online. ITSO wants to test the application for provision of timetable information in the first instance and, after establishing the return on investment (ROI) for that investment,

wants to ensure that related applications could be supported using the same application approach and infrastructure as the initial application.

- Related to the mandatory business requirement for extensibility, ITSO Railways wants to ensure that the approach to its voice interface is generalizable across the full range of customer interfaces to timetable information. These could include visual interfaces for mobile devices (PDAs), multimodal interfaces (voice and visual interaction) for a range of devices (smartphones, etc.), as well as access by Internet or local information kiosks. Therefore a mandatory requirement is that the application solution demonstrates multi-channel re-use of approach.

While the initial requirements (IVR with grammar-based approach for speech recognition) could be met by use of a standard IVR/speech recognition system, the requirements (including direct extensibility for multiple applications and multi-channel re-use) indicate the need for an alternative approach that guarantees these requirements.

14.2 High-level architectural overview

The use case analysis indicates the need for several high-level architectural components. These are presented on the next diagram.

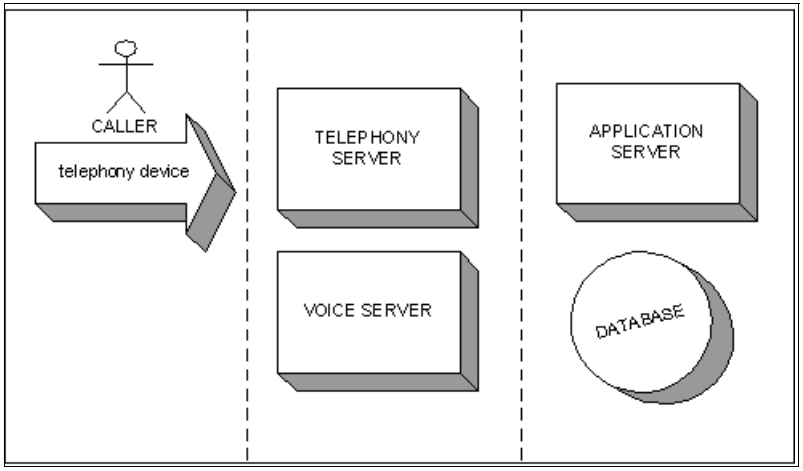


Figure 14-4 High-level architecture diagram

The architecture requirements show that the system would be responsive to any telephony device (landline, mobile, VoIP, etc.) and would require a telephony server for call control and management, and a voice server to respond to customer queries and provide speech responses based on the query. In order to

find and aggregate the information required by the caller, an application server would need to access the ITSO Railway database and aggregate the information for output by the voice server.

14.3 Activity diagram

An activity diagram shows the dynamic interaction of a system in the flow across the application. Activities of classes across the system show a change in the state of the class. Activity diagrams are useful in voice applications because they provide a representation that is intermediate between the call flow of the interface and the objects required by the system. The activity diagram in the figure shows the flow of activities for the ITSO Railway timetable access requirements.

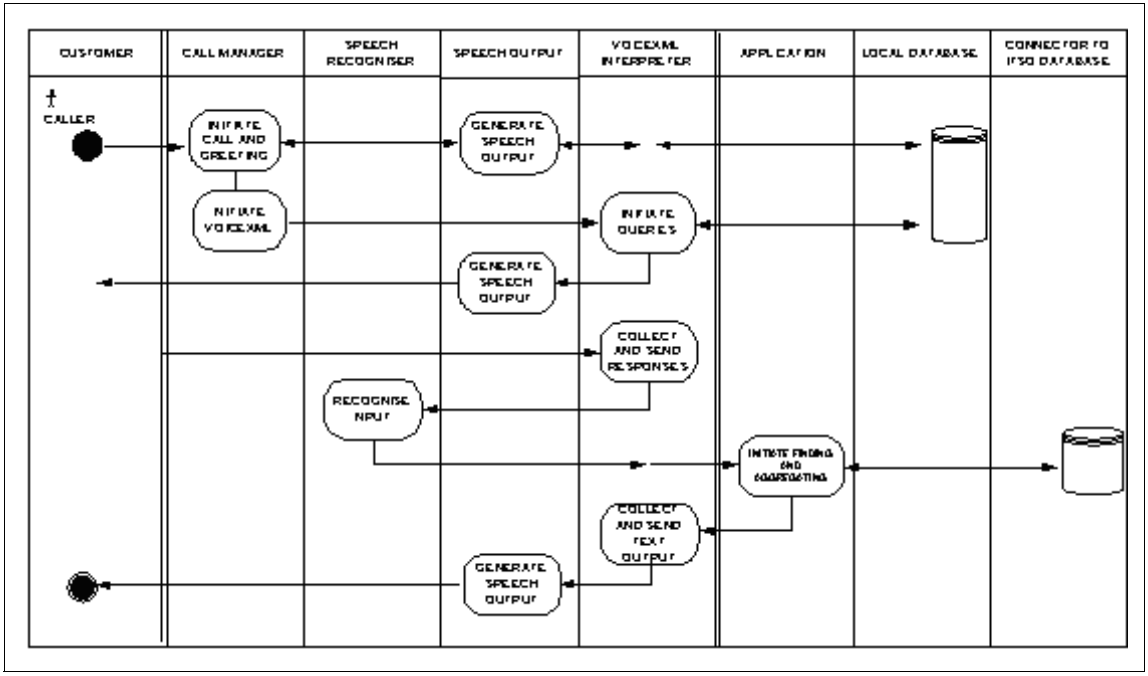


Figure 14-5 Activity diagram

The activity diagram shows a generalizable solution for the call flow and can be extended to meet most information access requirements for callers into the ITSO Railway automated information system.

The caller initiates the call and is given a greeting by the call manager that also initiates an VoiceXML script to handle the call. The greeting and subsequent

queries to the caller are managed via the speech output system that uses local text data as required.

There are two options for this speech output. The greeting and subsequent queries could be written “in-line” in the VoiceXML code or could be called from a local database. In this case the solution could be met by in-line data, but a more generalizable solution is implemented to provide extensibility for future requirements.

The text for presentation is sent to the speech output system for presentation. In this case there are also two options. One is to present the speech as text-to-speech (TTS) from the voice server. The other is to call a speech file (for example, .wav) from the local database for presentation to the caller. In this case ITSO Railways has elected to use natural speech from a selected speaker and so .wav files will need to be presented via the call management system (although any compression format could be supported depending on the network requirements). However, for development the TTS files can be used.

After presenting the prompts to the user, the system collects responses from the caller to the queries. This could be handled in a sequential manner with each query given an answer (what is the departure location, what is your destination location, etc.), or the system could ask a general question (for example, please tell us what journey you want to make), and could recognize the natural language response. These options differ in complexity.

In the first case the approach is based on a prepared grammar. Knowledge about what the user is likely to say is coded into expected utterances by the caller (the grammar) and the input by the caller is matched to the grammar. In the case of the timetable information the grammar would also consist of potential prefixes and suffixes to the utterance (for instance, the caller might say, “Oh, I would like to leave from Raleigh, please”) and the grammars would have to include these potential additions to the target information “Raleigh” to recognize it. In a reasonably constrained application such as the timetable requirements, the actual usage utterances will be reasonably predictable.

The second option is to collect a range of data about the way callers ask for timetable information. For instance, to the general question “Please tell us what journey you want to make,” a caller might say, “I want to go from Raleigh to Durham at around 3 p.m.” Once a sufficient number of potential user utterances to the query have been collected, a statistical language model (SLM) can be built that is used to determine the action intention of the caller. This approach is referred to as a Natural Language Understanding (NLU) approach.

Technologies exist for both approaches (menu/grammar or NLU) as well as mixes of them, and decisions about which to implement will depend on the complexity of the task and the cost benefit analysis. Obviously, callers prefer to

speak as naturally as possible to the system. However, it is also true that users find a form-filling approach quite effective because it makes it clear what needs to be done. However, for a grammar/menu approach the number of fields needs to be limited to maintain user satisfaction.

In the present requirements, because the queries to the caller can be well structured (“where do you want to leave from?”), and there are only three queries, ITSO Railways has decided on the basis of cost benefit to implement a grammar-based approach. However, it recognizes that once the system is in place it could easily collect the required data for the SLM (by initially prompting the caller with “Just say where you would like to go and at what time”) and use the menu presentation as a backup while it collected the data for the necessary SLM. This approach would permit ITSO Railways to extend its automated service to a range of other activities (such as including booking and itinerary checking applications) using more natural dialogue interactions.

In the menu/grammar-based approach adopted here the voice interface components obtain the required information from the caller, and the timetable application interrogates the caller for times of departure for the requested locations. This information is then sent to the timetable application via an VoiceXML script, and after the information has been returned the VoiceXML initiates its outputs to the speech generator (TTS or audio file) for presentation to the caller.

14.4 Components

In order to satisfy the architecture and activity requirements the following set of components can be assembled.

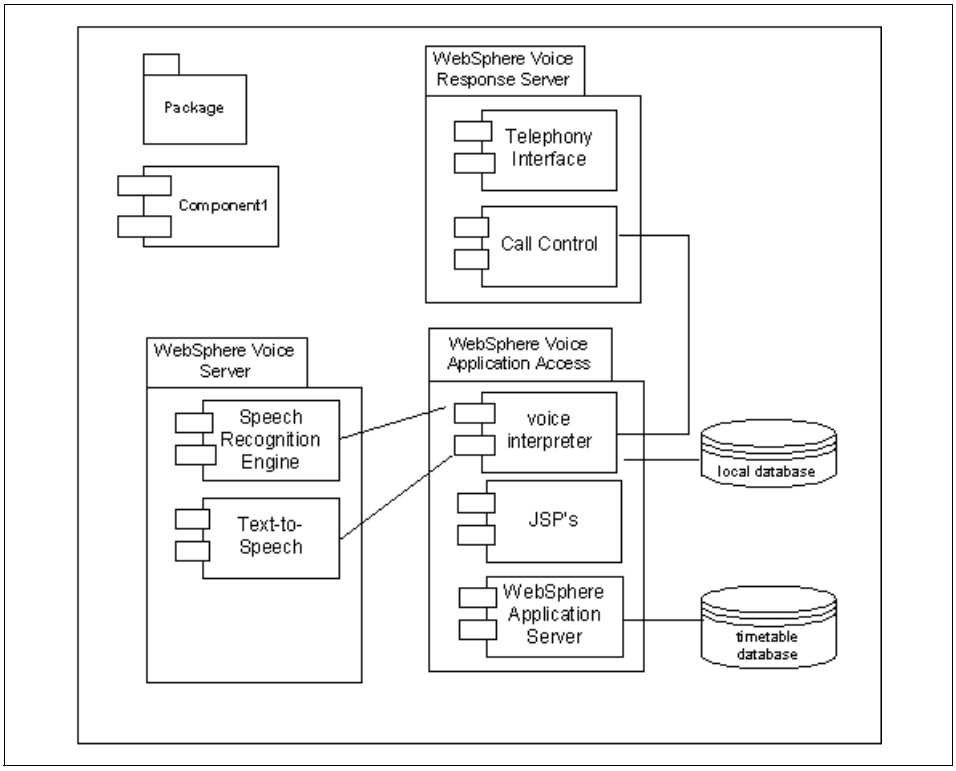


Figure 14-6 Component diagram

Given ITSO Railways requirements for extensibility in the future to both more complex applications and to multichannel re-use of information, WebSphere Voice Application Access was selected as the core technology.

WebSphere Voice Application Access provides:

- A full application development environment - The Voice Toolkit based on the Eclipse platform for the development of VoiceXML and voice portlets
- A VoiceXML interpreter for controlling both the call and voice resources
- WebSphere Application Server for running the application and providing legacy database connectivity

WebSphere Voice Application Access also includes the WebSphere Voice Server that contains both the necessary speech recognition capability as well as TTS for dynamic database presentation. However, these components have been presented separately in the figure to emphasise their specific roles in the application.

WebSphere Voice Application Access does not include the WebSphere Voice Response (WVR) server that provides both connectivity to most PBX systems for call control or that can function independently for call input. WVR is just one of a range of options for managing the call control requirements and any VoiceXML-compatible system could be deployed. WVR provides robust, scalable performance for the ITSO requirements and has options for complete re-use of development if ITSO Railways decides to change from PSTN to Voice over IP (VoIP) in the future (WVR accomplishes this by changing the T1/E1 cards for PBX connectivity to SIP cards for VoIP connectivity). This means that any development completed by ITSO Railways for its current requirements will be available within a VoIP environment.

WebSphere Voice Application Access is a component of WebSphere Portal Server. This provides extensibility to ITSO Railways for presenting future visual information to users via the Web for timetable and other information. Users could experience a similar structured interface when they visit the Web for information or when they call on the phone. In addition, utilization of WebSphere portal information provides access of the timetable information by mobile devices for visual presentation onto PDAs, etc., via WebSphere Everyplace Access (WEA), as well as via multimodal presentation using both visual and voice presentation. Adopting the portlet framework for voice provides a straightforward extensiblity by ITSO Railways into the other devices in the future, as required.

14.5 Interface for call flow

Having determined that ITSO Railway's requirements can be met by available technology, the detailed requirements for ITSO Railway's interface need to be considered.

There are a number of issues to consider in completing a call flow design. These are described below.

14.5.1 Dialogue design

This needs to provide an efficient and effective interface to the information access by the caller. Issues will depend on complexity of menu items if using a grammar-based approach and potential for error recovery if the user becomes lost in the application or if the speech recognition is inaccurate and produces an incorrect response.

Error recovery strategies will depend on the accuracy of the speech recognition engine for the application. All automated speech recognition technologies produce errors, and the ability to recover from them will determine the user satisfaction with the application. If the application is likely to be error-prone

(because of complexity of the task and grammars used), then it is usually best to provide an error recovery system at the start of the system (for example, “If you need assistance at any time just say help. If you want to go to the previous selection say go back.” Etc.).

However, if the system has low complexity and grammar requirements, then an initial announcement about error recovery systems delays the dialogue flow and reduces user satisfaction. This means that it is useful to obtain test data for the operation of the system before incorporating a full error recovery system, basing the design decisions for the error recovery approach on the accuracy of the recognition.

Other options to include are error recovery options within the main menu choices (for example, “If you need more information just say help.”).

14.5.2 Persona selection

Selecting a persona for the application is more complex than just selecting a “voice” that is suitable for branding for the organization. The way the voice resources interact with the customer is strongly influenced by the customer motivation, which is in turn a direct result of the way the customer responds to the “personality” of the presentation.

Many factors can go into successful persona creation including understanding the branding requirements of the organization, the age groups that might be the predominant users of the system, as well as the nature of the task to be completed. Sometimes the persona might need to sound authoritative to give a sense of confidence about the information, and sometimes it might suggest “fun” to encourage the user to enjoy the application.

After an application persona has been developed, the “voice” for the application that matches both the persona criteria and provides clear presentation over a telephone can be selected. Audio prompts are developed from the dialogue design, grammar predictions, and persona criteria.

14.5.3 Usability design

The most critical user requirement for an automated system is to access information (or transactions) effectively (without error) and efficiently (straight access to required information). Voice interfaces need to be as minimal as possible to provide this efficiency and effectiveness. Any delays to explain things to users often lead to frustration. Again a lot of things go into appropriate usability design, but all rely on effective evaluation of the interface before release. This type of testing can be done off-line to ensure that a group of potential users

understands the prompts as they were intended and that the users provide responses that are consistent with the dialogue and grammar designs.

The basic information around which these decisions are made is the design of the interface call flow. An example call flow for the ITSO Railways is presented in Figure 14-7.

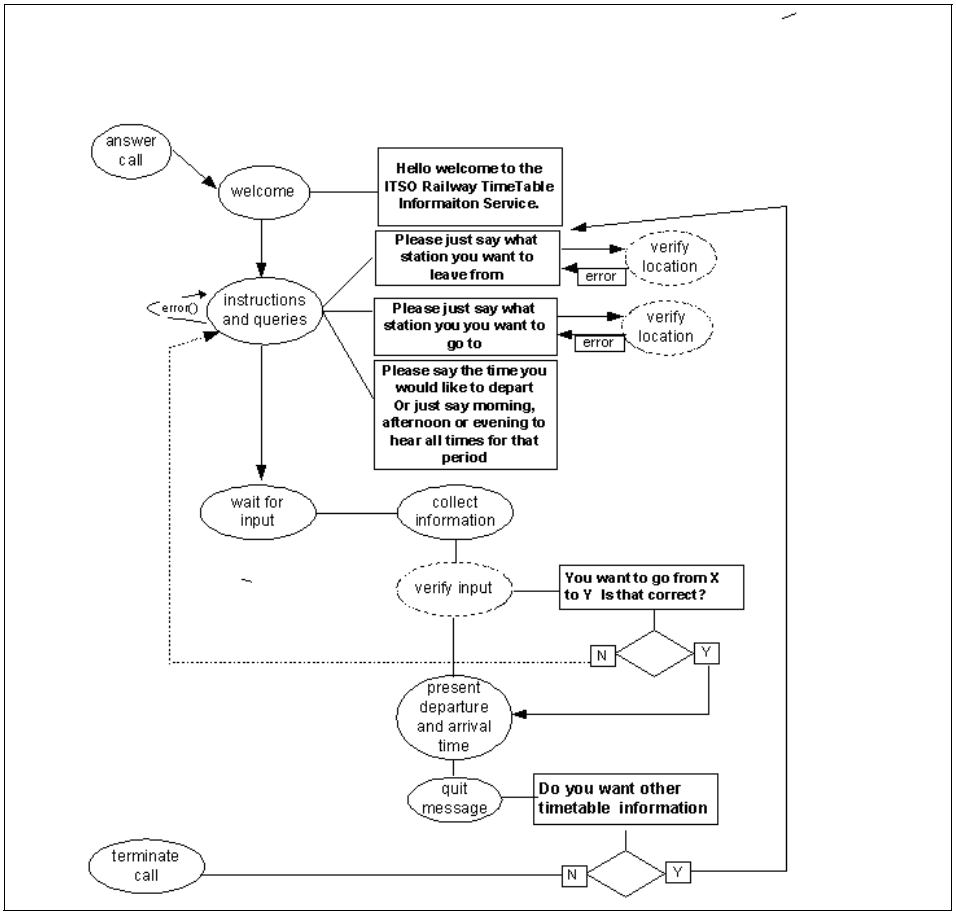


Figure 14-7 Call flow

The decisions made in the design of the call flow include:

- Callers will receive a welcome to the ITSO Railway Timetable Information Service.
- They are asked to speak their responses. (Users can still assume that they might be entering a DTMF IVR system for information, so it is remains useful to emphasise a speech response.)

- If the caller does not respond (silent responses to speech recognition are still prevalent) then it is advisable to repeat the instructions after silence detection (or if recognition confidence is very low).
- In the ITSO Railway design it has been decided to check the user input against station location information. This is because the task puts the burden on the user to know the station location names. It is possible that they might not select a location at which there is a railway station. If a non-match is detected then the user is prompted for another input. This strategy may be acceptable and can be confirmed after usability testing, but in the future after ITSO has established the usage of the system it has the option of including context location information into the system that could suggest the closest station for any locations input by the caller without a station.
- After input of the departure and destination locations the user is prompted to indicate the time of departure or the period over which they would like to hear the times for departure.
- In order to ensure the correct departure and destination locations the caller is prompted with the selections. If there is an error the user is redirected to the start of the queries, otherwise they are given the departure information. An alternative error checking strategy would be to confirm the locations separately so that the user might need to only repeat one of the options if it had not been correctly recognized.
- If the caller receives the information she needs then she is asked if she needs further assistance. For this simple application, if the caller does have other requirements, she is simply redirected to the start of the queries. However, in more complex applications they could select other options for assistance. In particular, a logical extension of the current application would be to include a booking option, a review of the booking option, and a payment option. These would all be straightforward extensions of the activity and call flow patterns discussed in this application.

14.6 Development of timetable access

Following the selection of WebSphere Voice Application Access, the development stage for ITSO Railway's timetable access by phone application can be based on the included voice-related Toolkits for WebSphere Studio.

The toolkits are add-ons for either WebSphere Studio Site Developer V5.1 or WebSphere Studio Application Developer V5.1.

WebSphere Voice Application Access provides a development environment for WebSphere Voice Response (WVR) V4.2 for telephone connectivity and call control functionality using VoiceXML, for WebSphere Voice Server (WVS) V4.2

for speech recognition and text to speech capability, and for WebSphere Voice Application Access V5.0.

The functionality of Voice Toolkit 5.0 that will be deployed for this application includes:

- The Graphical Call Flow Builder providing drag-and-drop call flow design as well as scripts and VoiceXML structures for development.
- A VoiceXML Editor that includes a wizard for selecting and customizing Reusable Dialog Components. These components can provide significant effort savings if they contain the vocabulary and grammar options. For instance, one re-usable grammar contains the names of major US cities. In this case the ITSO Timetable information resides in a proprietary database that will be used to develop the vocabulary with standard grammar constructions added for implementation. In future applications for ITSO the developed grammar will be added to their Re-Usable Dialog Components and re-used for subsequent applications.
- VoiceXML.
- A grammar editor supporting VoiceXML 2.0 formats for building grammar requirements for the timetable application.
- A IBM extension to VoiceXML (based on the International Phonetc Association Representation) Pronunciation Builder that permits modification and creation of vocabulary pronunciations. This is an important component for dealing with names of stations, etc., that might vary from normal language pronunciation of the items. In the current development this tool will be used to add to the vocabulary recognition values (to accommodate multiple pronunciations of station names) and over-ride standard TTS values to approximate local pronunciation usage.
- Audio recorder that will permit recording and testing of user prompts for the application as well as for recording the timetable database and time information.

14.7 Voice portlet development

The ITSO Railway application will run as a voice portlet in the WebSphere Application Access server. In order to develop the voice portlet for the application, the functionality of the Voice Toolkit is employed. This section describes how to set up the Voice Toolkit for developing a VoiceXML application. The application requires grammars to match to the user's utterances in the application, and their development is also outlined.

14.7.1 Setting up Voice Toolkit V5.0 for WebSphere Studio

For the ITSO Railways application the project used WebSphere Studio Application Developer Version 5.1 and the Voice Toolkit Version 5.0. After installing these tools you can access the Voice Toolkit. Our development is for WebSphere Voice Application Access and the application will be built as a voice portlet for WebSphere Voice Application Access.

1. Open Websphere Studio Application Developer and specify a new workspace.
2. After WebSphere has loaded select **Window** → **Open Perspective** → **Voice Portlet**.

   The voice portlet uses the portlet application development so it is necessary to open a portlet application project.

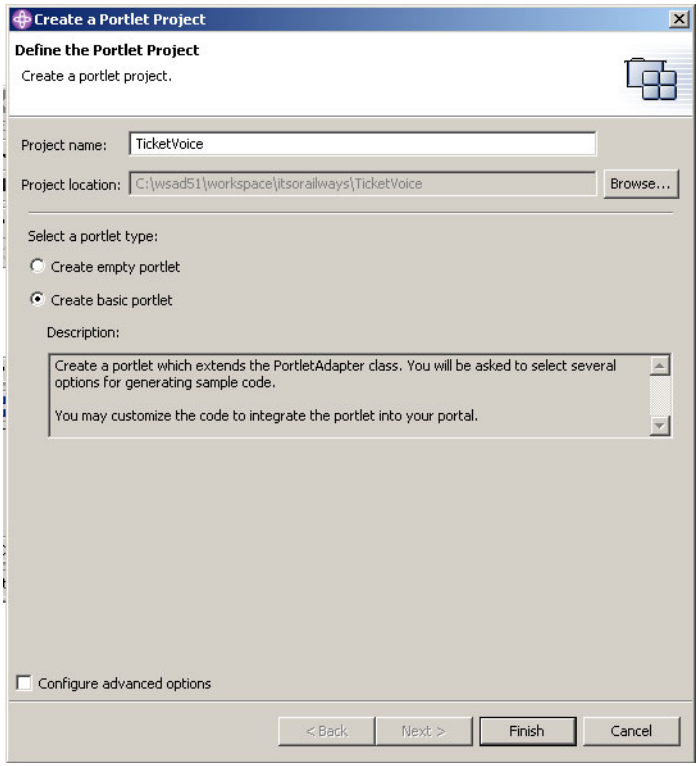


Figure 14-8

3. Select **File** → **New** → **Portlet Application Project** → **Finish**.

14.7.2 Application grammar development

This section assumes the use of Pronunciation Tools in the toolkit to modify the standard recognition based on local pronunciation of stations as required. This may require input by linguistic resources to establish the correct pronunciation requirements. The present application development assumes that the database for station names to be used for speech recognition have been linguistically analyzed and the necessary alternative pronunciations have been prepared using the Voice Toolkit.

The application can be developed using the Voice Toolkit tools described in the following sections.

Before creating the application we can construct a grammar for the requirements.

In order for a JSP to be constructed to utilize an external grammar (compared to an in-line grammar used in the VoiceXML script) it is necessary to construct the external grammar in a standardized format. For all but the simplest applications it is necessary to develop an external grammar approach, and this is described here.

In order to consolidate a number of grammar options that have been available for grammar development (Nuance GSL, IBM BNF, Java Speech Grammar Format) the W3C has proposed two main grammar formats for VoiceXML 2.0. This includes the Augmented Backus-Naur Format (ABNF) and an XML form of a Speech Recognition Grammar Specification (SRGS). ABNF has been used for a long time (for example, HTTP is specified in ABNF) and provides an easy-to-read, compressed view of grammars. SRGF is hierarchically arranged text strings encapsulated in XML elements. This makes the XML considerably longer and more difficult to read. The VoiceXML 2.0 standard only requires SRGF. This is the option employed here and provides portability across a range of applications.

Constructing an SRGF for either in-line or external grammars is similar. The process includes:

- Construct grammar header.
  - Grammar description and version (the grammar description in this case is xml version=1.0 with encoding to be used for English - ISO-8859-1.
  - Language, in this case English - En.
  - Mode of operation (DTMF or voice).
  - Root grammar (grammars can contain subgrammar rules in a single file so it is important to identify the main grammar when the grammar is called).

- Grammar rules.

  Each grammar rule in a file is identified with a unique name. In order to construct a grammar rule for the application we specify the grammar scope *(private* means only local reference within the grammar file, while *public* provides a global reference from another VoiceXML dialog or grammar).

- Creating optional items and lists.

  The creation of grammars is an iterative process that involves using existing knowledge about what users might say (many callers say “um” before answering or finish with a “please” or “thank you”) as well as attempting to constrain the prompts presented to the caller to encourage them to answer precisely. Grammars benefit from collecting “Wizard of Oz” data from potential users to build the initial grammar as well as iterative tuning after implementation to ensure that the richness of the user’s responses is captured by the grammar. Natural Language approaches are differentiated by needing to collect a reasonable amount of user data initially to create the statistical language modelling on which the NL processing is based.

  The ITSO Railway application is a reasonably constrained application (users could just say the station name), but it needs “normal usage” grammar components to take into account typical caller utterances.

  These issues can be reflected in the grammar by including optional grammar elements at the start of the potential utterance (for example, “um”) and at the conclusion of the caller response (for example, “please”). “Tuning” the grammars after implementation will involve additions to these typical usage elements depending on the way users actually respond to the prompts.

  The target words can be differentiated from the optional grammar components by forming a list that is used to search for the requested station for departure and arrival.

  SRGF provides the options for handling repeated options with an extension to the <item> tag:

  ```
  <item repeat="0-1">
  ```

  And a list tag <one-of> to create a list of potential targets for the ITSO Railway stations:

  ```
  <one-of>
  <item>Raleigh</item>
  <item>Durham</item>
  </one-of>
  ```

  The *one-of* tag can also be used for indicating that only one option will be accepted.

  From these guidelines a grammar is developed and created as a file for calling during the application using the Voice Toolkit Grammar.

Creating the grammar file

To create the grammar file:

1. Select **File** → **New** → **SRGS XML Grammar File** from the menu.

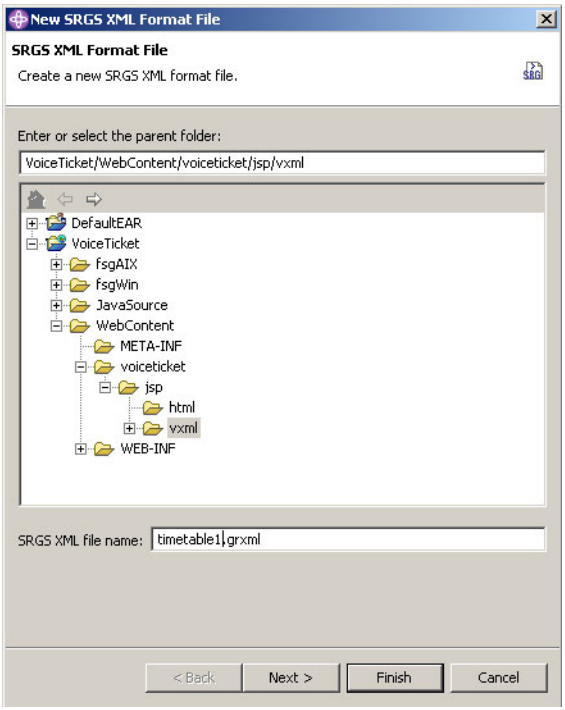


Figure 14-9

2. After naming the grammar select **Next**.

   In normal operation, Text File would be selected (ITSO Railways would have a text list of all their stations available). In this case we list three stations and include them as a list.

3. Click **Finish**.

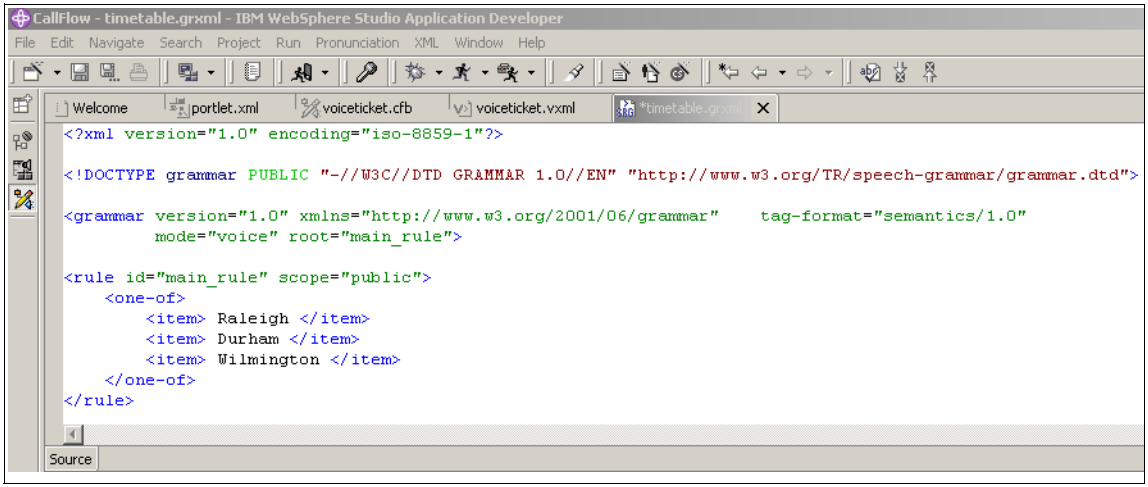


Figure 14-10 .grxml editor view

This shows that the basic grammar file is created with a list of grammar items providing station locations.

For practical applications we need to extend this grammar to include the range of possible usage utterances that users will respond with to the prompt. As an example, in the next example options for “um, I want to go to” are added prior to the target list and “please, thank you” are added as a suffixes to the grammar. In actual practice there could be a range of possible usage grammars that might need to be added to the ensure reliable recognition.

An example of a final grammar file to incorporate some likely usage utterances is shown in the following example.

Example 14-1 Grammar XML source

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
"http://www.w3.org/TR/speech-grammar/grammar.dtd">
<grammar version="1.0" xmlns="http://www.w3.org/2001/06/grammar"
tag-format="semantics/1.0" mode="voice" root="locations">
<rule id="locations" scope="public">
   <ruleref uri="#optionalin"/>
   <ruleref uri="#stations"/>
   <ruleref uri="#optionalout"/>
</rule>
<rule id="optionalin">
   <item repeat ="0-1">
      <one-of>
         <item> um </item>
```

```
         <item> I want to go to </item>
      </one-of>
   </item>
</rule>
<rule id="stations">
   <one-of>
      <item> Raleigh </item>
      <item> Durham </item>
      <item> Wilmington </item>
   </one-of>
</rule>
<rule id="optionalout">
   <item repeat="0-1">
      <one-of>
         <item> thankyou </item>
         <item> please </item>
      </one-of>
   </item>
</rule>
</grammar>
```

The rule *stations* is included here to show what items would be added. However, for the ITSO Railways application these target items for the grammar would be available from an ITSO Railways database that will be called as a Java Server Page to create dynamic availability of the database. This will be demonstrated in the application development.

This section has described only those grammar components necessary to implement the ITSO Railways application. Additional functionality is possible in the SRGF and the details can be found in W3C Speech Recognition Grammar Specification Version 1.0. W3C Recommendation 16 March 2004:

http://www.w3.org/TR/speech-grammar/

14.7.3 Creating a database for the application

In most applications, such as the ITSO Railways application, the target data (in this case station locations) will be stored in a separate DB2 database for access by the application during runtime. Development of the application can proceed with a trial database (for example, constructed as a flat-file in Cloudscape that is used to test the application in the present development).

For the ITSO Railways development a trial database is constructed based on the following table.

Table 14-1 Database content for the stations

Departure location	Arrival location	Segment of day	Departure time
Raleigh	Durham	AM	6
Raleigh	Durham	AM	10
Raleigh	Durham	PM	4
Raleigh	Durham	PM	6
Raleigh	Wilmington	AM	8
Raleigh	Wilmington	PM	10

14.7.4 Creating a call flow for the application

Call Flow Builder is a graphical editor with which to create and test call flow models for the application. It also builds the basic code from the call flow design and creates the voice application.

1. To create a call flow design select **File** → **New** → **Other** from the menu, then provide the file name as shown below.

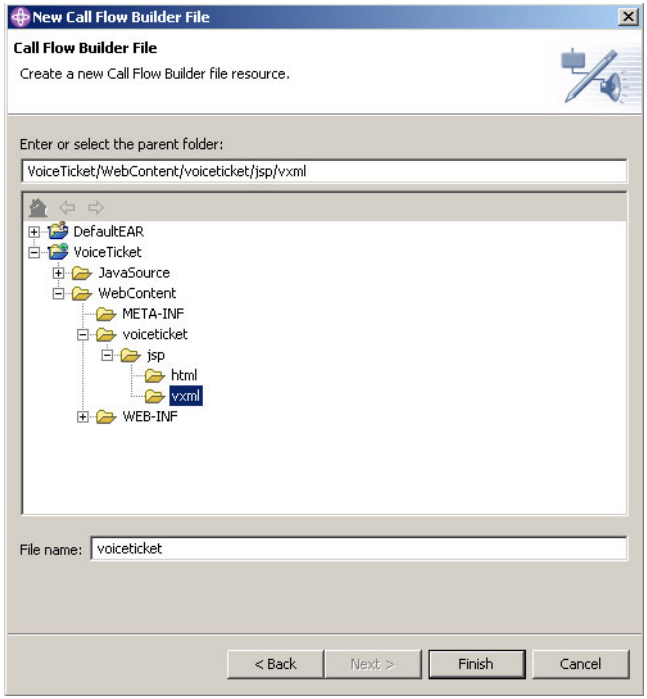


Figure 14-11 New Call Flow Builder file

2. On the next page select **Voice Tools** → **Call Flow Builder**.

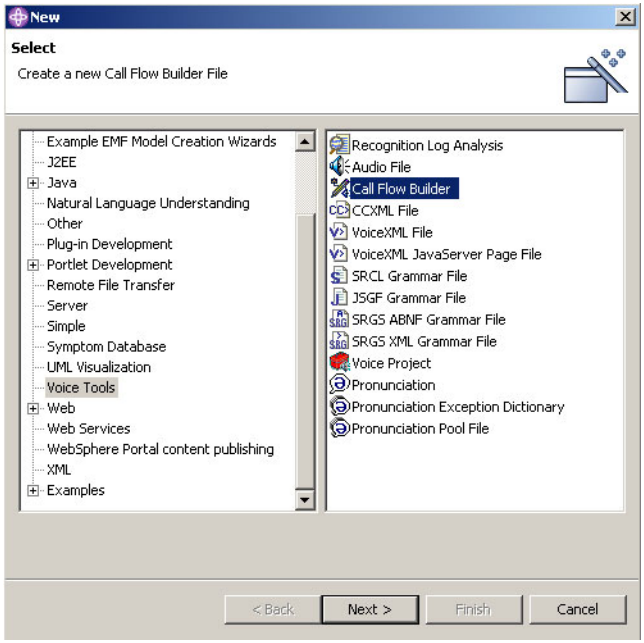


Figure 14-12 New call flow

3. Click **Next**.
4. Switch to the call flow perspective by selecting **Window** → **Open Perspective** → **Callflow**.

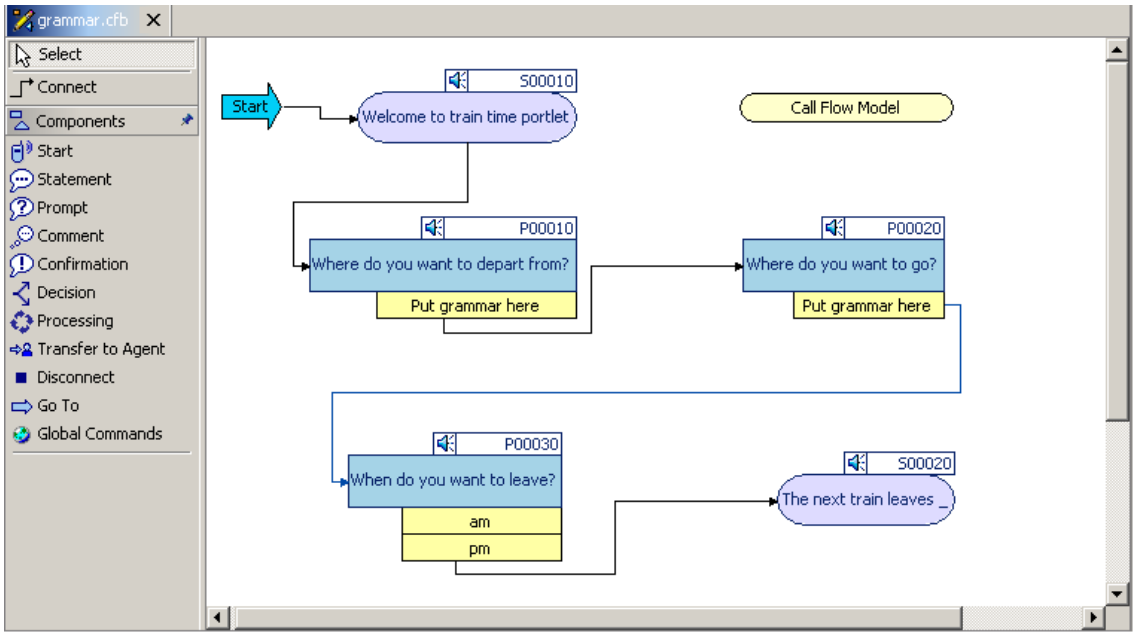


Figure 14-13 Call flow perspective with the call flow editor

In this call flow the dynamic grammar presentation will be added in the “put grammar here” container. In the call flow the time of day request has been constructed as an in-line grammar because, in the example, only two options are presented. If more times were required (for example, every 5 minutes of the day), the grammar would be created as a dynamic external grammar application. This application might, for example, calculate the current time and offered selections based on the criteria of "trains in one hour from now".

14.7.5 Creating speech output

The following picture is an empty call flow in the editor.

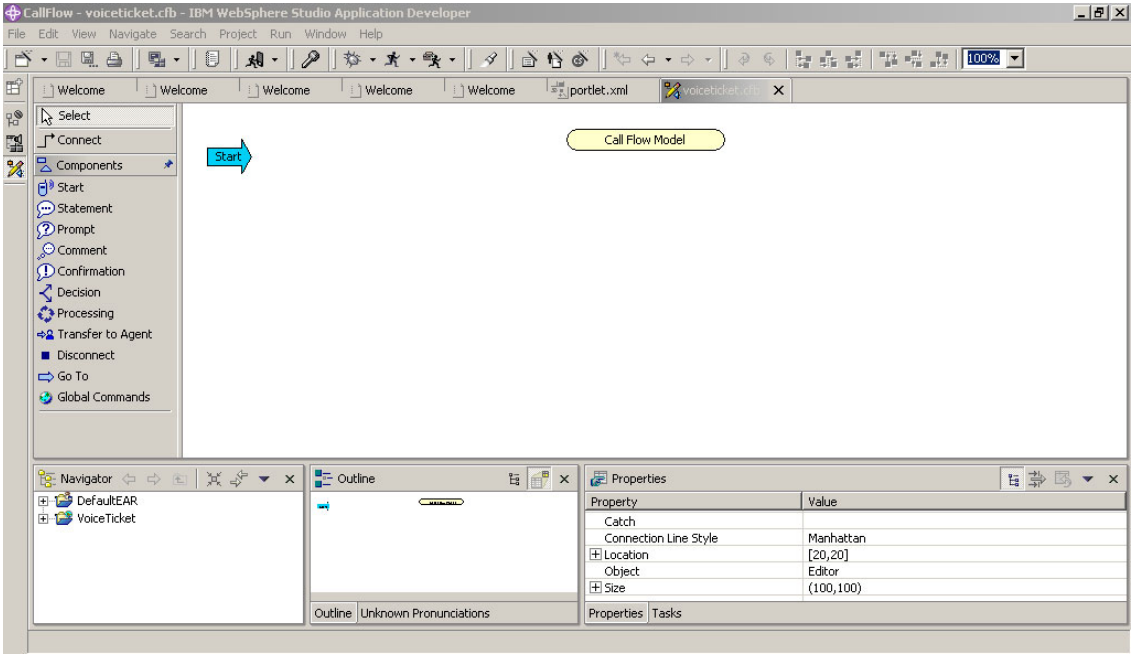


Figure 14-14 Empty call flow

The application can use natural speech recorded from a selected speaker or text-to-speech output from the voice technology platform (WebSphere Voice Server).

ITSO Railways have selected a natural spoken voice for their application. For the full implementation this will involve recording each station name, and each time of departure so they can be played back in response to the user request. The current development of the application will use the TTS option in the call flows (which is automatically invoked for VoiceXML if an audio file is not available). This section briefly describes the process for speech file creation for the prompts for the development and for the prompts and responses in the final implementation.

1. From the Call Builder Perspective open the *Audio File tool* by selecting **File** → **New** → **Audio File**.

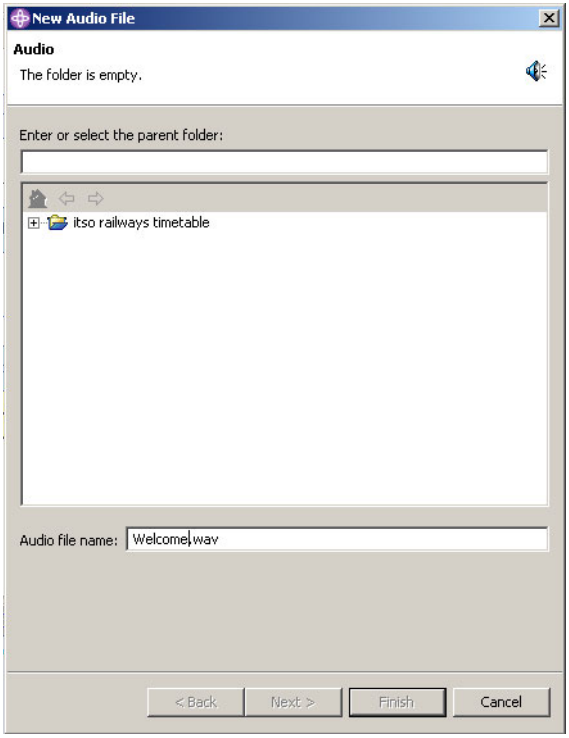


Figure 14-15 New audio file

2. Select the working folder and the appropriate path.
3. Click **Next**.

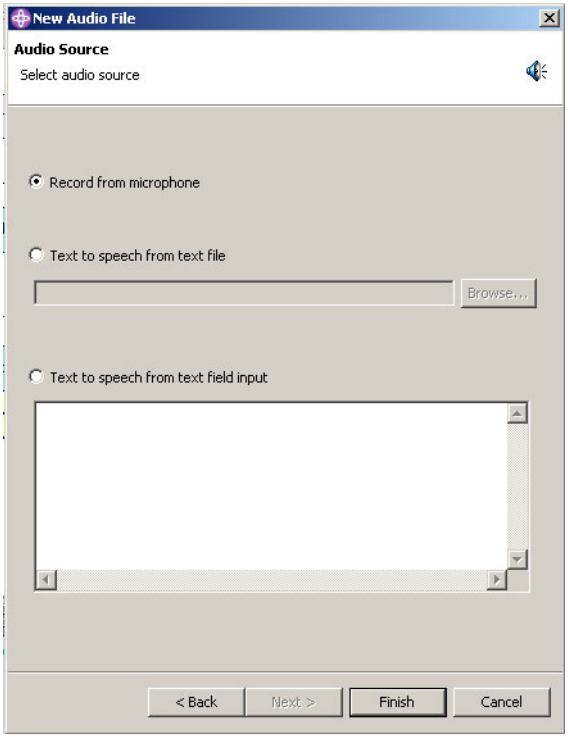


Figure 14-16 Selecting the audio source

4. Select **Record from Microphone** → **Finish**.

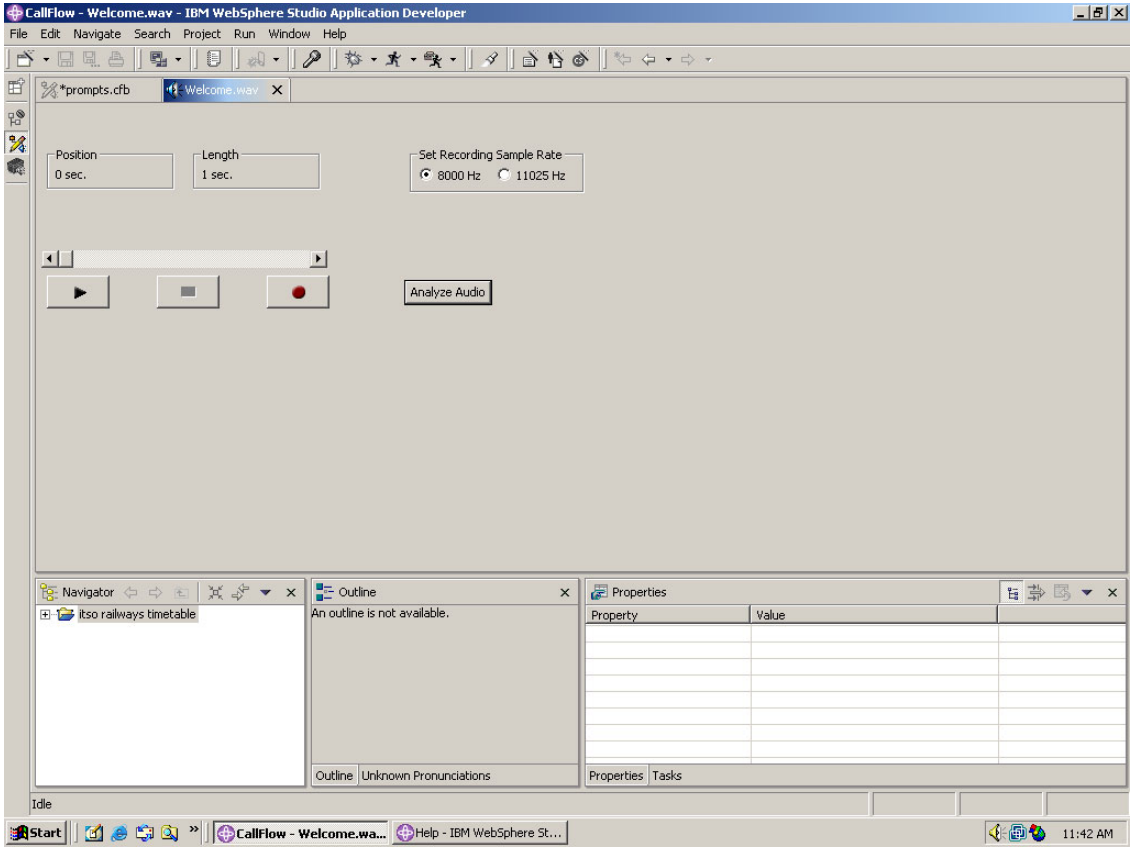


Figure 14-17 Recording Tool in Eclipse

5. Before starting recording the microphone needs calibration. To set the level select the Microphone icon, then select **Script**.

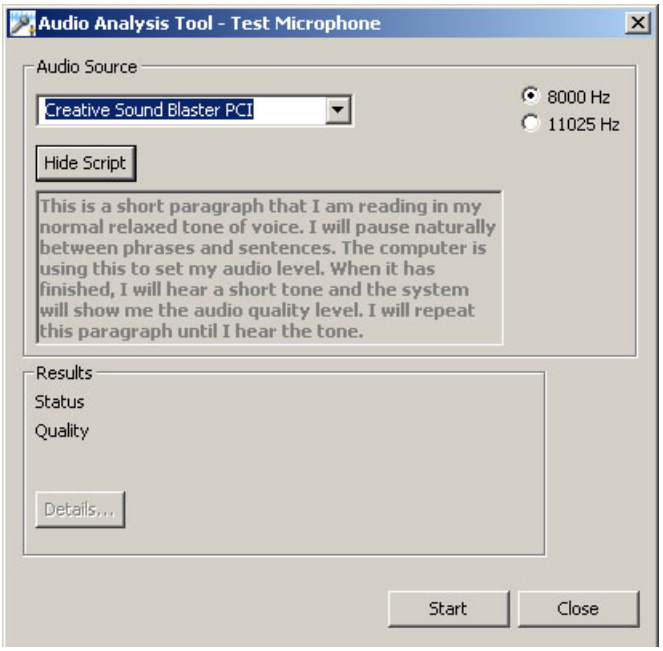


Figure 14-18 Microphone testing

6. After completing calibration click **Close**.
7. On the main recording screen use the controls for recording.
8. On completion of recording select **Analyse Audio**.

The analysis tool provides an automatic rating of quality of the recording.

The Audio File tool could be used to record the prompts during the development as well as for implementation. To construct the natural speech responses to the caller queries (for example, “The next train leaves at ..”, etc.), natural speech files would need to be concatenated to produce the necessary statements. This would require an audio tool with this functionality.

14.7.6 Generating basic VoiceXML code structure from call flow

The next step is to generate the VoiceXML from the flow.

1. When the call flow is completed, the code for the project can be generated by right-clicking the blank space in the diagram.

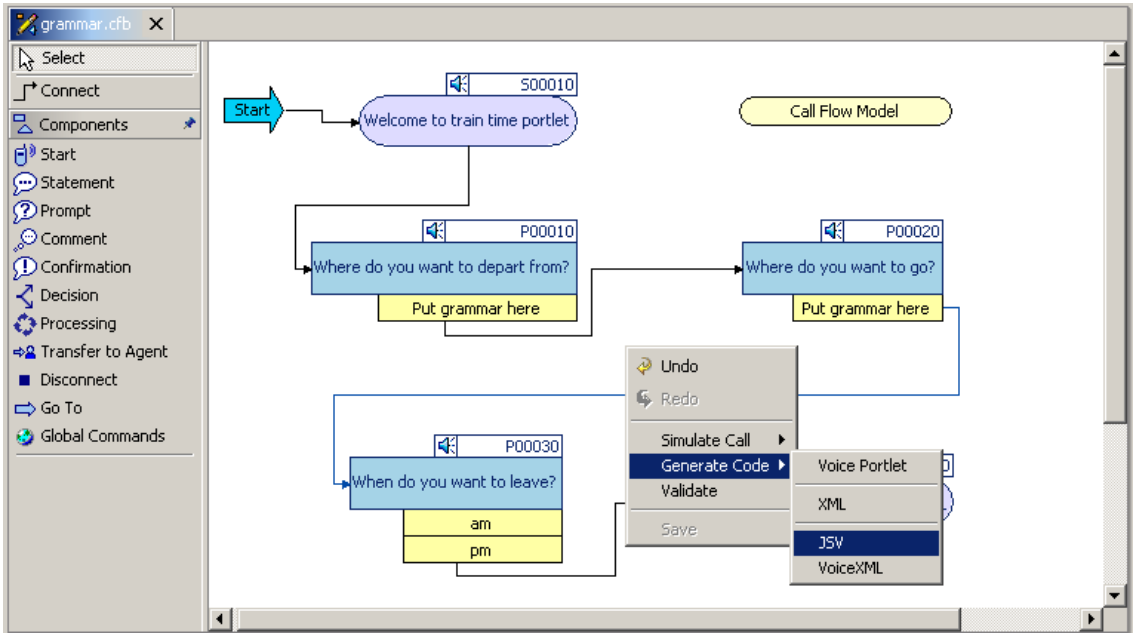


Figure 14-19 Final call flow

2. Select **Generate Code** → **Voice Portlet**. This generates the .jsv page for the project. This can be viewed from the Project Navigator.

Example 14-2 Call flow source

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE vxml PUBLIC "-//W3C//DTD VOICEXML 2.0//EN" "vxml20-1115.dtd">
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xml:lang="en-US">
   <%@ page contentType="application/voicexml+xml" %>
   <!--   Call Flow Model   -->
   <!--Begin Call Flow-->
   <property name="universals" value="help cancel" />
   <property name="audiomaxage" value="1" />
   <var name="mode" expr="'speech'" />
   <var name="previousResult" />
   <form id="S00010">
      <block>
         <audio src="./grammar%20audio/S00010s.wav">Welcome to train time
portlet</audio>
         <goto next="#P00010" />
      </block>
   </form>
   <var name="P00010" />
   <form id="P00010">
```

```
      <field name="fP00010">
         <option value="Put grammar here">Put grammar here</option>
         <prompt cond="mode == 'speech'" bargein="true">
            <audio src="./grammar%20audio/P00010s.wav">Where do you want to
depart from?</audio>
         </prompt>
         <prompt cond="mode == 'dtmf'" bargein="true">
            <audio src="./grammar%20audio/P00010d.wav">Where do you want to
depart from?</audio>
         </prompt>
         <filled>
            <assign name="P00010" expr="fP00010" />
            <assign name="previousResult" expr="fP00010" />
            <goto next="#P00020" />
         </filled>
      </field>
   </form>
   <var name="P00020" />
   <form id="P00020">
      <field name="fP00020">
         <option value="Put grammar here">Put grammar here</option>
         <prompt cond="mode == 'speech'" bargein="true">
            <audio src="./grammar%20audio/P00020s.wav">Where do you want to
go?</audio>
         </prompt>
         <prompt cond="mode == 'dtmf'" bargein="true">
            <audio src="./grammar%20audio/P00020d.wav">Where do you want to
go?</audio>
         </prompt>
         <filled>
            <assign name="P00020" expr="fP00020" />
            <assign name="previousResult" expr="fP00020" />
            <if cond="fP00020 == 'Put grammar here'">
               <goto next="#P00030" />
            </if>
         </filled>
      </field>
   </form>
   <var name="P00030" />
   <form id="P00030">
      <field name="fP00030">
         <option value="am">am</option>
         <option value="pm">pm</option>
         <prompt cond="mode == 'speech'" bargein="true">
            <audio src="./grammar%20audio/P00030s.wav">When do you want to
leave?</audio>
         </prompt>
         <prompt cond="mode == 'dtmf'" bargein="true">
```

```
            <audio src="./grammar%20audio/P00030d.wav">When do you want to
leave?</audio>
         </prompt>
         <filled>
            <assign name="P00030" expr="fP00030" />
            <assign name="previousResult" expr="fP00030" />
            <goto next="#S00020" />
         </filled>
      </field>
   </form>
   <form id="S00020">
      <block>
         <audio src="./grammar%20audio/S00020s.wav">The next train leaves
_</audio>
      </block>
   </form>
<!--End Call Flow-->
```

3. To complete our application we need to include dynamic grammar generation from a database using Java Server Pages and a JavaBean database application to calculate the available times for selected departure and arrival selections. We use the basic code generated from the Call Builder to complete these requirements and convert them to JSP files.

   This process generates the following .jsv file for the voice application.

Example 14-3 .jsv file generated from the flow

```
<%@ page session="false" contentType="text/x-vxml" import="java.util.*,
ticketvoice.*" %>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
<var name="start" />
<var name="end" />
<var name="timeLeave" />
<%
   TicketVoicePortletSessionBean sessionBean =
(TicketVoicePortletSessionBean)portletRequest.getPortletSession().getAttribute(
TicketVoicePortlet.SESSION_BEAN);
%>
   <property name="universals" value="help cancel" />
   <property name="audiomaxage" value="1" />
   <var name="mode" expr="'speech'" />
   <var name="previousResult" />
<%if((sessionBean.getStart().equals(""))){%>
   <form id="welcome">
      <block>
```

```
         <audio
src="<%=portletResponse.encodeURL("grammar_audio/S0001Os.wav")%>">Welcome to
train time portlet</audio>
         <goto next="#Departure" />
      </block>
   </form>
   <form id="Departure">
      <field name="fDepartureLocation">
               <jsp:include page="departureLocations.jsp"/>
         <prompt cond="mode == 'speech'" bargein="true">
            <audio
src="<%=portletResponse.encodeURL("grammar_audio/P0001Os.wav")%>">Where do you
want to depart from?</audio>
         </prompt>
         <nomatch>
            No trains depart from that city. Please try another.
            <reprompt />
         </nomatch>
      </field>
         <filled>
            <assign name="start" expr="fDepartureLocation" />
            <submit next="<portletAPI:createURI><portletAPI:URIAction
name="submitDeparture"/></portletAPI:createURI>" namelist="start"
method="post"/>
         </filled>
   </form>
<% } %>
<% if(!(sessionBean.getStart().equals("")) &&
sessionBean.getEnd().equals("")){%>
   <form id="Arrival">
      <field name="fArrivalLocation">
               <jsp:include page="arrivalLocations.jsp"/>
         <prompt cond="mode == 'speech'" bargein="true">
            <audio
src="<%=portletResponse.encodeURL("grammar_audio/P0002Os.wav")%>">Where do you
want to go?</audio>
         </prompt>
         <nomatch>
            No trains arrive at that city. Please try another.
            <reprompt />
         </nomatch>
         <filled>
            <assign name="start" expr="'<%=sessionBean.getStart()%>'"/>
            <assign name="end" expr="fArrivalLocation" />
            <submit next="<portletAPI:createURI><portletAPI:URIAction
name="submitArrival"/></portletAPI:createURI>" namelist="start end"
method="post"/>
         </filled>
      </field>
```

```
   </form>
<% } %>
<% if(!(sessionBean.getStart().equals("")) &&
!(sessionBean.getEnd().equals("")) && sessionBean.getTimeLeave().equals("")){%>
   <form id="leaveWhen">
      <field name="fTimeToGo">
         <grammar version="1.0" xml:lang="en" mode="voice" root="TimeOptions">
         <rule id="TimeOptions" scope="public">
            <one-of>
               <item>am</item>
               <item tag="am">morning</item>
               <item>pm</item>
               <item tag="pm">afternoon</item>
            </one-of>
         </rule>
      </grammar>
         <prompt cond="mode == 'speech'" bargein="true">
            <audio
src="<%=portletResponse.encodeURL("grammar_audio/P0002Os.wav")%>">When do you
want to go?</audio>
         </prompt>
         <nomatch>
            Please say am or pm.
            <reprompt />
         </nomatch>
         <filled>
            <assign name="start" expr="'<%=sessionBean.getStart()%>'" />
            <assign name="end" expr="'<%=sessionBean.getEnd()%>'" />
            <assign name="timeLeave" expr="fTimeToGo" />
            <submit next="<portletAPI:createURI><portletAPI:URIAction
name="submitTime"/></portletAPI:createURI>" namelist="start end timeLeave"
method="post"/>
         </filled>
      </field>
   </form>
<%}%>
<% if(!(sessionBean.getEnd().equals("")) &&
!(sessionBean.getTimeLeave().equals(""))){%>
   <form id="sayTimes">
      <block>
         The times your train leaves are:
         <% sessionBean.setDb_sqlstring("SELECT TIMES FROM APP.TIMES WHERE
NAMES1 = '"+sessionBean.getStart()+"' AND NAMES2 = '"+sessionBean.getEnd()+"'
AND TIMEOFDAY = '"+sessionBean.getTimeLeave()+"'");
            sessionBean.getJDBCResult();%>
         <jsp:include page="getTimes.jsp"/>
      </block>
   </form>
```

This file incorporates the VoiceXML controls for handling prompts and requests, but also adds the dynamic *includes* required to access the database components. The *includes* are presented, showing the call to the departure grammar and database (Example 16-4), the arrival departure database (with the requested departure station removed) (Example 16-5), and the sort for available times for requested locations and the period of the day (collected from the in-line grammar in the .jsv file) (Example 16.6).

Example 14-4 Included grammar file

```
<%@ page session="false" contentType="text/x-vxml" import="java.util.*,
ticketvoice.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
<%
   TicketVoicePortletSessionBean sessionBean =
(TicketVoicePortletSessionBean)portletRequest.getPortletSession().getAttribute(
TicketVoicePortlet.SESSION_BEAN);
%>
<grammar version="1.0" xml:lang="en" mode="voice" root="dlocations">
   <rule id="dlocations" scope="public">
      <!--Optional grammars can be added at this point
      See section on generating grammar-->
      <one-of>
         <% sessionBean.setDb_sqlstring("SELECT NAMES FROM APP.LOCATIONS");
            sessionBean.getJDBCResult();%>
         <% for(int x=0; x<sessionBean.getDb_rowCount(); x++){%>
            <item><%=sessionBean.getDb_SQLresult(0,x)%></item>
         <% } %>
      </one-of>
      <!--Optional grammars can be added at this point
      See section on generating grammar-->
   </rule>
</grammar>
```

4. As indicated in the comments, it is possible to add optional grammar components similar to those shown in the grammar development section that would account for real utterances by users (“um”, “thank you”). These could be either included as an in-line grammar directly in the .jsv file (similar to the time of day requests) if there were only a few likely instances, or included as a static external file called from the .jsv file. The actual grammar file for the optional grammars would be developed using the grammar editor.

   A similar component is added to activate the arrival grammar. This grammar is the same as the departure grammar except that it does not include the location requested as a departure. (This is a refinement that might be more applicable to other applications—in this case the user is unlikely to ask for the same location for departure and arrival.)

Example 14-5 Arrival grammar

```
<%@ page session="false" contentType="text/x-vxml" import="java.util.*,
ticketvoice.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ taglib uri="/WEB-INF/lib/jspsql.jar" prefix="dab" %>
<portletAPI:init/>
<%
   TicketVoicePortletSessionBean sessionBean =
(TicketVoicePortletSessionBean)portletRequest.getPortletSession().getAttribute(
TicketVoicePortlet.SESSION_BEAN);
%>
<grammar version="1.0" xml:lang="en" mode="voice" root="alocations">
   <rule id="alocations" scope="public">
      <!--Optional grammars can be added at this point
      See section on generating grammar-->
      <one-of>
         <% sessionBean.setDb_sqlstring("SELECT NAMES FROM APP.LOCATIONS WHERE
NAMES <> '"+sessionBean.getStart()+"'");
            sessionBean.getJDBCResult();%>
         <% for(int x=0; x<sessionBean.getDb_rowCount(); x++){%>
            <item><%=sessionBean.getDb_SQLresult(0,x)%></item>
         <% } %>
      </one-of>
      <!--Optional grammars can be added at this point
      See section on generating grammar-->
   </rule>
</grammar>
```

5. Finally, we need to add an application that sorts the database relative to departure time available for the selected locations and period of the day.

Example 14-6 Database tasks code snippet

```
<%@ page session="false" contentType="text/x-vxml" import="java.util.*,
ticketvoice.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ taglib uri="/WEB-INF/lib/jspsql.jar" prefix="dab" %>
<portletAPI:init/>
<%
   TicketVoicePortletSessionBean sessionBean =
(TicketVoicePortletSessionBean)portletRequest.getPortletSession().getAttribute(
TicketVoicePortlet.SESSION_BEAN);
%>
<% for(int x=0; x<sessionBean.getDb_rowCount(); x++){
   if(x>0){%>and<%}%>
   <%=sessionBean.getDb_SQLresult(0,x)%>
<% } %>
```

14.8 Testing the Timetable application

In order to test the application we use the Call Simulator running in the WSAD Portal Test Environment and using the TC/PIP Monitor Tool. This is set up with the following steps:

1. Right-click the **TicketVoice** application, then select **Run on Server**.
2. Select **Create a new Server**, then click **Next**. Fill out the forms as shown in the following two pictures.

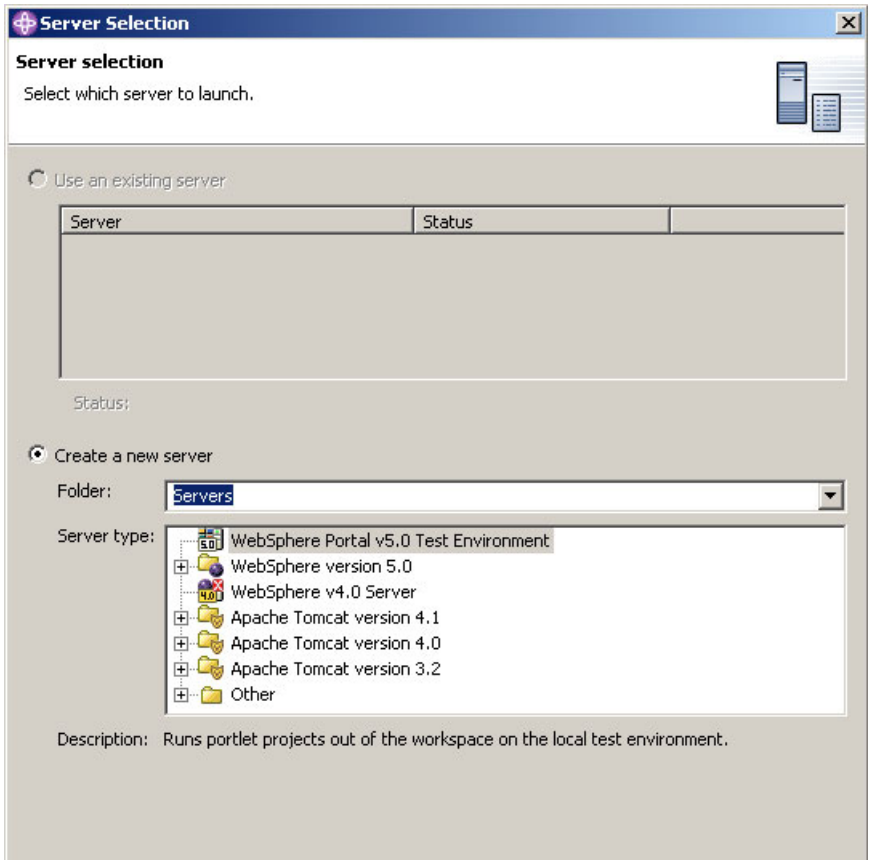


Figure 14-20 Server selection form

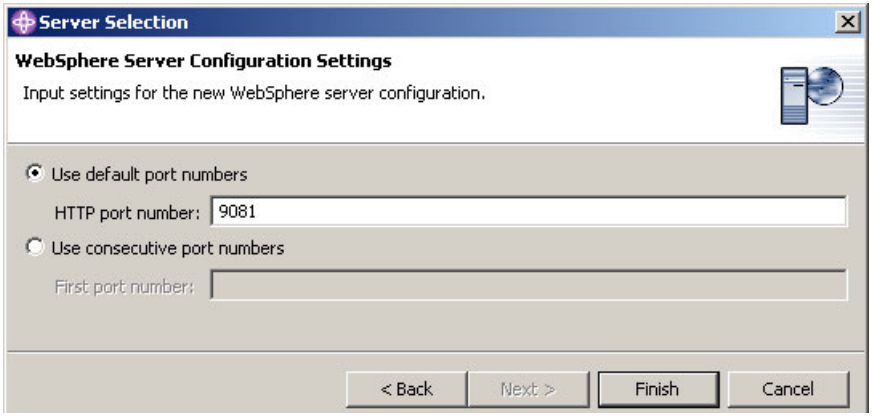
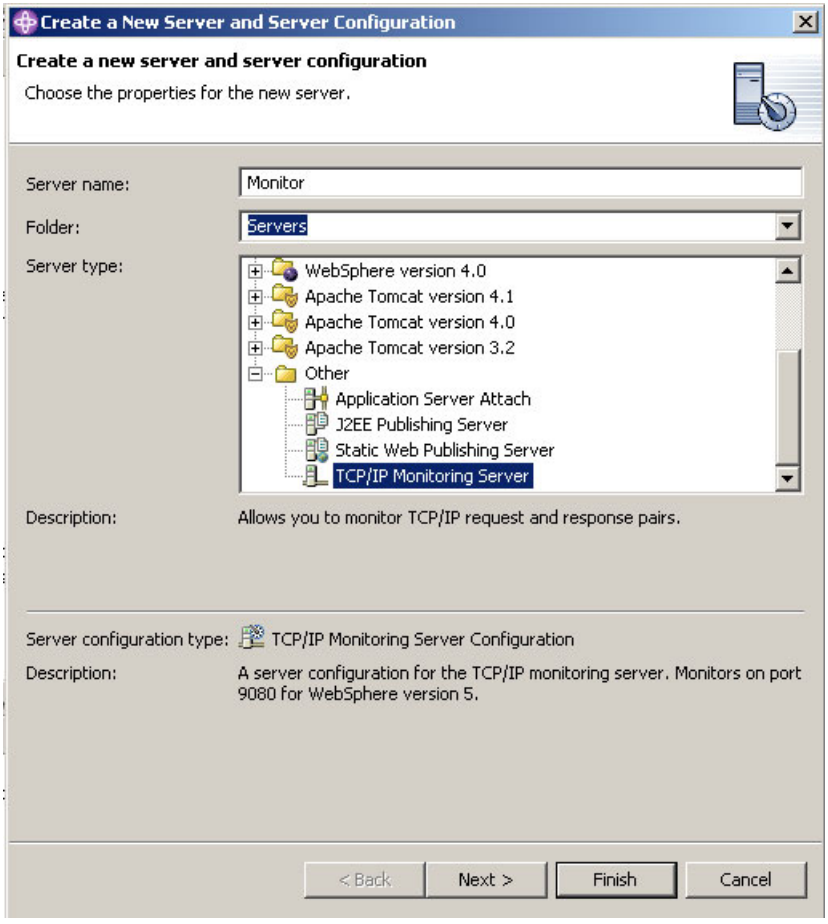


Figure 14-21 Server configuration form

3. Click **Finish**.
4. Under the Server Perspective select **New** → **Server and Server Configuration** to create the monitor server.



5. Select **Monitor** → **Next**.

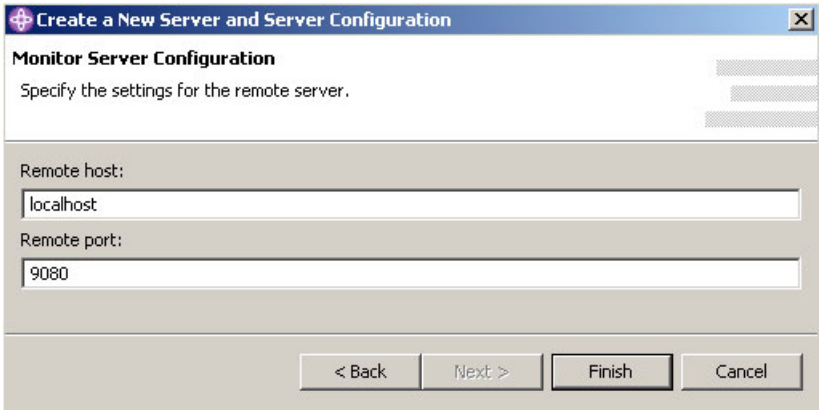


Figure 14-22 Monitor server configuration

6. Click **Finish**.
7. To configure the proxy server for the Monitor, double-click **Monitor**.
8. Select **Configure** switch ports as shown in the next picture.

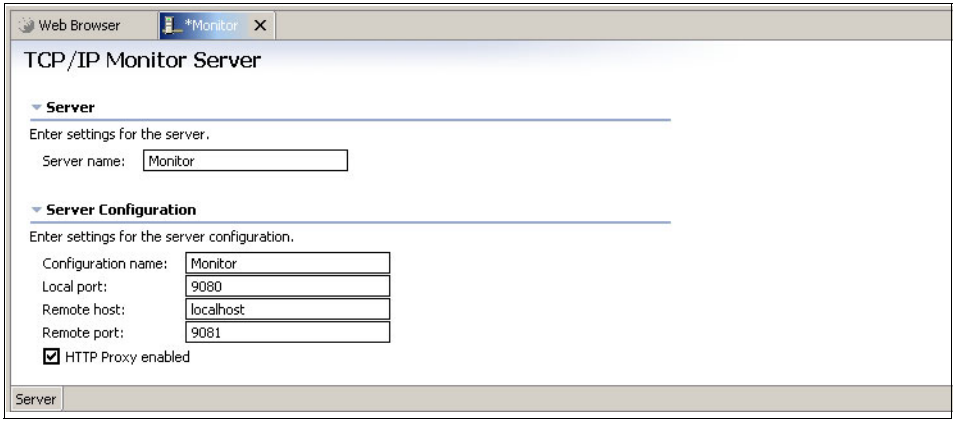


Figure 14-23 Monitor server configuration view

9. Type in the ports. Enter values for the local and remote ports. Set the local port to 9080 and the remote port to 9081.
10. Run the sample application by selecting **Run** from the menu, then **Run** again.
11. We need to configure a new application client. Select **VoiceXML JavaServer Page** → **New**.

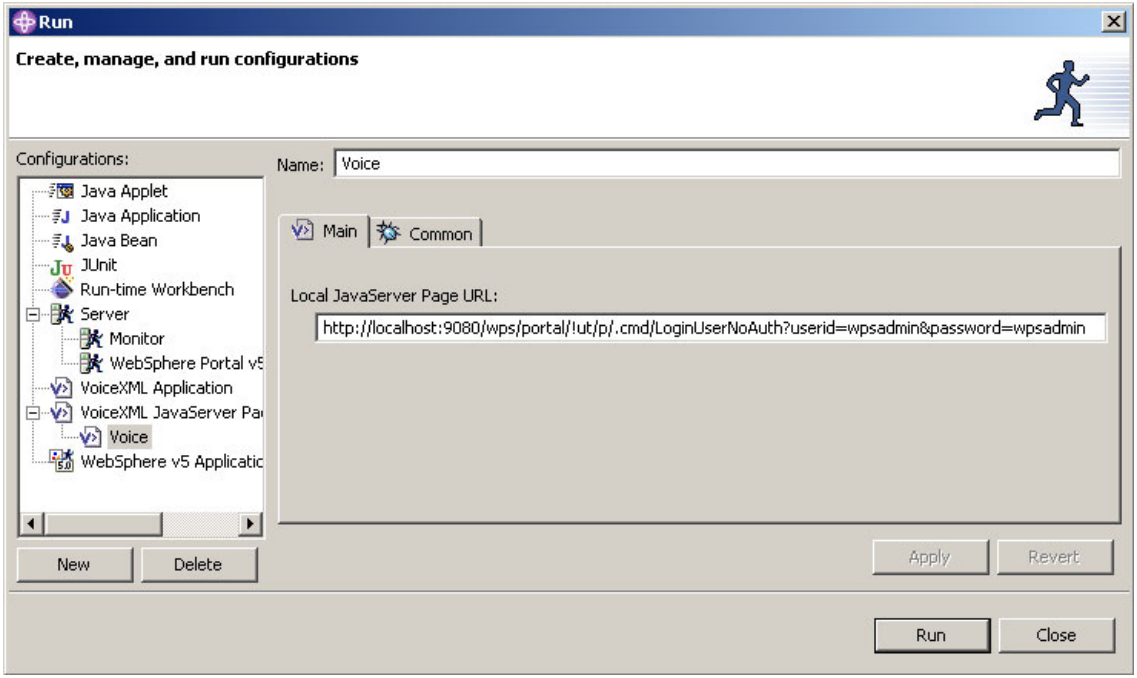


Figure 14-24 New client configuration

12. Select **Voice**, and type in the *JavaServer Page URL* as shown above.
13. Select **Run**. This will display the Monitor Output.

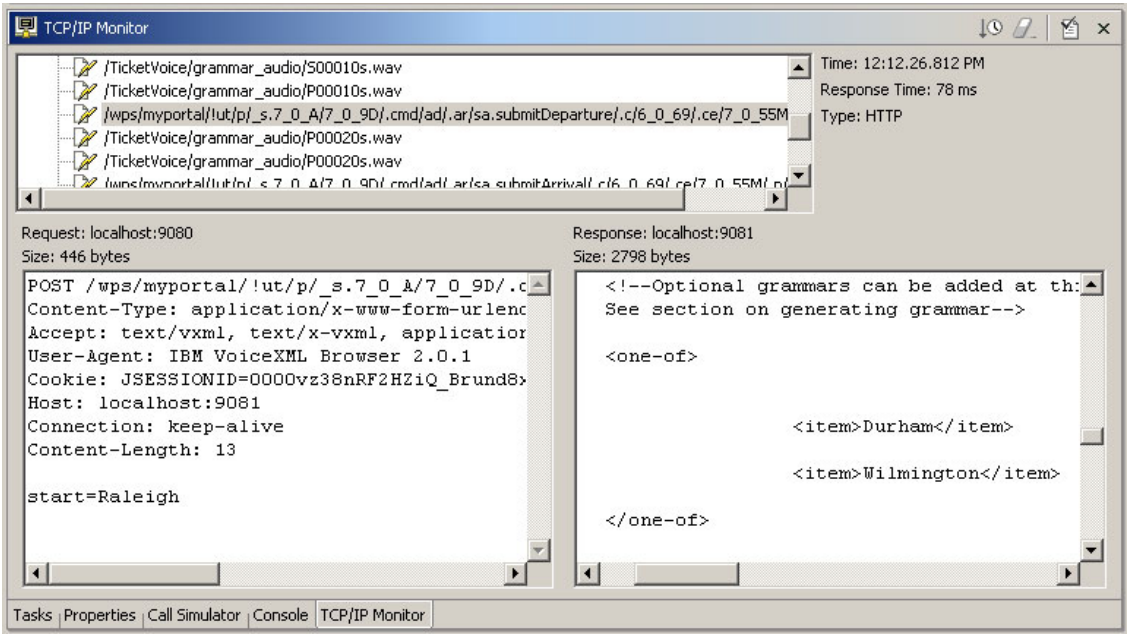


Figure 14-25 Monitor view

14. Toggle output screens to view the *Call Simulator*.

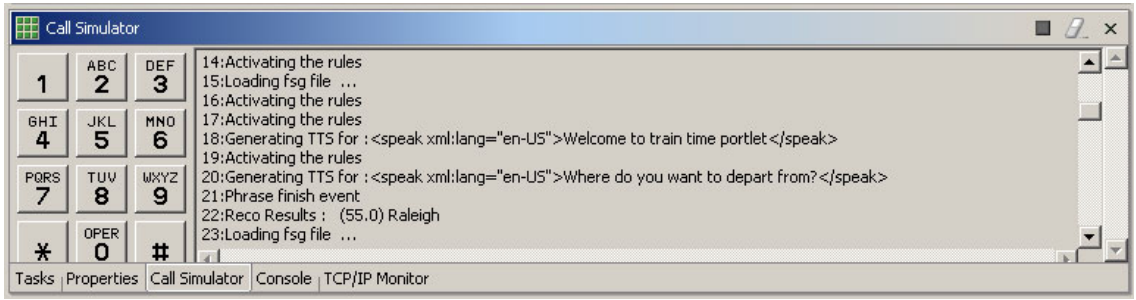


Figure 14-26 Call simulator

14.9 Preparing voice portlet for implementation

After testing and debugging the voice portlet it is necessary to install it for use in WebSphere Portal. This will involve the preparation of the full database (rather than the test database used in the examples) in DB2 and installation in WebSphere Portal. These requirements are not in the scope of the present

example, but once DB2 and Websphere Portal have been installed they require a .war file for the voice portlet.

1. Creation of the .war file from the developed Voice Portlet is described here.
2. Select **Window** → **Voice Perspective**.
3. Right-click the **TicketVoice** project and select **File**, then select **Export**.
4. Select **WAR file**.

Figure 14-27 Export .war file

5. Provide the file destination as shown above, then click **Finish**.

14.9.1 Deploying the voice portlet in WebSphere Portal

Create a new project with the Project Creation wizard in WebSphere Portal.

1. Click **File** → **New** → **Portal Application Project** from the menu.
2. Select **Configure advanced options** on the Define Project page.
3. Select **Add VoiceXML markup support** on the Miscellaneous page.
4. Import the existing project via the WAR file.
5. Associate the Portlet 5.0 API with the project.
6. Select **File** → **Properties** → **Portlet API** from the menu.
7. Select **WebSphere**.

14.10 Meeting ITSO Railways future multi-channel requirements

The current application development has concentrated on the requirements for building a voice portlet that can be used to provide access to ITSO Railway timetable information. A major business requirement for multi-channel re-use (deployment of the same application framework across the Web, phone, PDAs, etc.) indicated the need for use of a portlet approach.

The extension of the voice portlet that provides voice access to the timetable information into other devices is described in Chapter 10, “Web access to ITSO Railway’s timetables” on page 189.

15

Connectivity and access

This chapter discusses the connectivity and access part of the pattern architecture. It consists of mobile access services, which enables mobile devices to connect to the enterprise infrastructure. The connectivity and access node accommodates different services specific to a mobile environment.

Applying the Patterns for e-business approach will help to classify and structure the business and IT needs in order to get a common set of decision criteria and an architectural baseline for connectivity and access services for pervasive devices.

15.1 Business initiatives and environment

To remain competitive, enterprises need to make sure that mobile workers have access to office services and enterprise applications outside of the office. Enterprises have a rich computing environment with a variety of applications, such as sales force and field force automation and important enterprise data. Main business initiatives are:

- Enable corporate workforce seamless mobile access to business application and information.
- Protect business applications and data communication.
- Cost-efficient connection.
- Leverage existing connection and security infrastructure.

An understanding of the business environment and requirements is the first step to build a robust connection service. The environment consists of different domains such as:

- Mobile environment
- Security environment
- Network environment
- Application environment
- Device environment

The following diagram is the Runtime pattern we implemented in this particular scenario.

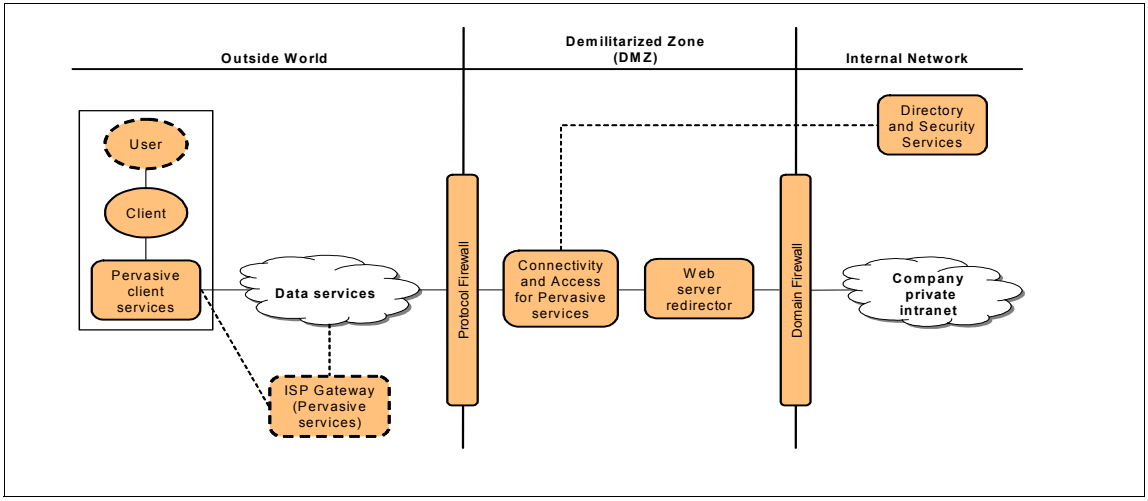


Figure 15-1 Runtime pattern for the connectivity scenario

The actual Product mapping that is used in this chapter to describe the scenario can be seen on the following diagram.

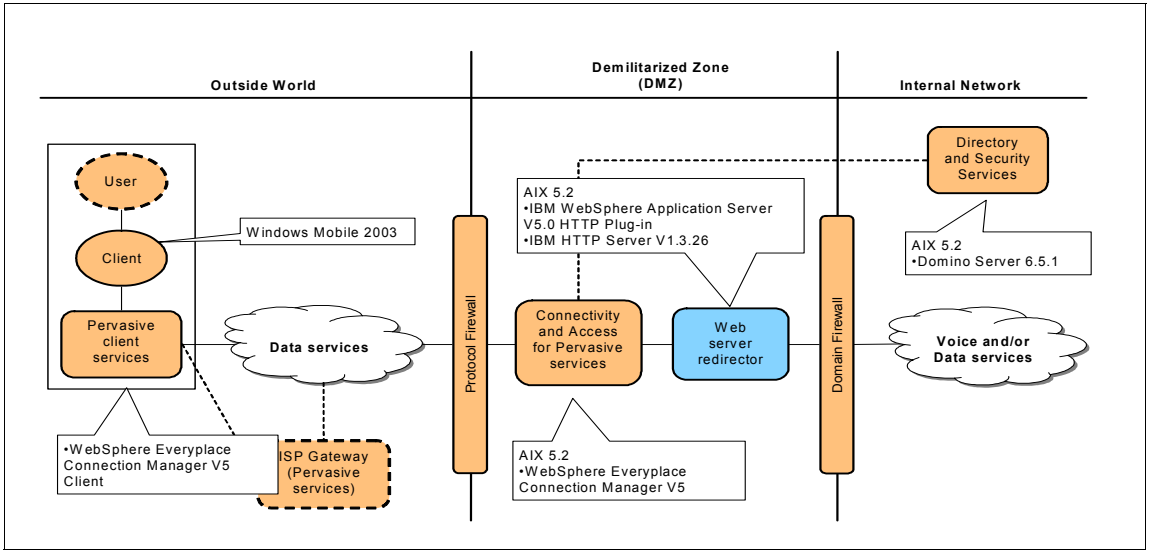


Figure 15-2 Pervasive Connectivity runtime pattern::Product mapping=Linux

15.1.1 Mobile environment

In order to provide access to mobile services a company faces different challenges to set up a reliable and secure access to their business applications. To understand the different areas we will stay in context with the big picture of a mobile workforce shown in Figure 15-3.

The mobile environment diagram shows domains of a typical mobile workforce. Different user groups such as sales, management, service technical, home office workers, and others need access to their business application over different connections using a wide spectrum of devices.

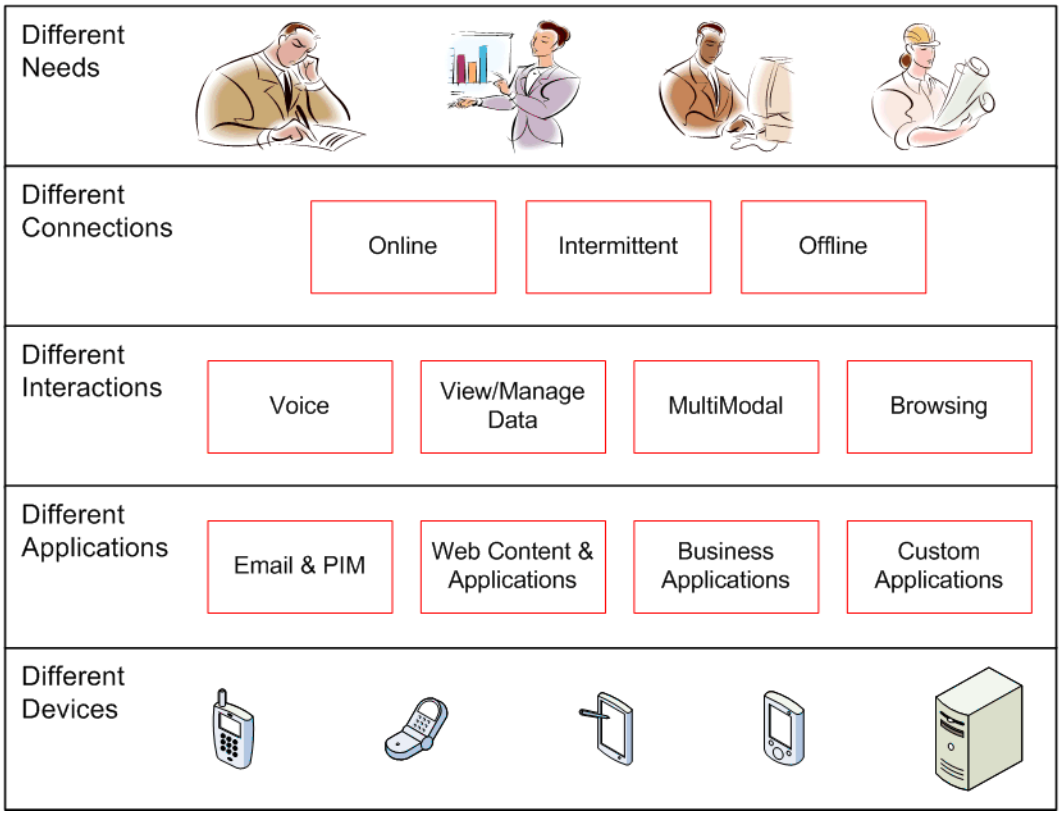


Figure 15-3 Mobile environment

A mobile environment consists of areas that have a high demand on flexibility and the ability to adapt new business needs quickly. Drivers for a mobile environment can be found in the business, social, and technical environments. The next section will talk about key areas of a mobile environment.

15.1.2 Security environment

This section focuses on connectivity and access security. Providing secure access to business applications is an essential function of a mobile access service. Secure access provides different security services to mobile entities such as:

- Identification and Authentication
- Authorization
- Integrity
- Privacy and Confidentiality

- Availability

Security services

This section discusses the security services in further detail.

Identification and Authentication

Identification and Authentication (I&A) services identifies and authenticates entities such as users, devices, or applications by different identification parameters such as user IDs, PINs, device IDs, certificates, and security tokens. Two common types of I&A are Basic authentication and Strong authentication.

Basic authentication is the combination of an identification parameter and a secret such as a password, PIN, or key. Basic authentication provides a base security level at a low cost. *Strong authentication* extends the basic authentication model using additional security parameters such as certificates, biometry system, TAN lists, or hard tokens. Strong security provides a good security level at a higher cost.

The choice of an I&A mechanism must follow guidelines and specifications described in the corporate security policies and standards, considering existing authentication mechanism and devices capabilities.

Authorization

Authorization grants or denies access to information and applications based on security policies and access control. The authorization service creates a security context for a connection and maps the identified entity to the designated access policy. The access policy is represented by access roles and related access control lists (ACLs).

Integrity

Integrity assures that data is unchanged from its source and has not been accidentally or maliciously modified, altered, or destroyed.

Privacy and Confidentiality

Privacy and Confidentiality protects data from unauthorized access by encryption and access control.

15.1.3 Network environment

Today's network environment offers a broad variety of communication options. The border between voice and data networks falls with the rise of new technology like Voice over IP (VoIP) standard or 3G (for example, UMTS) networks. Nevertheless, the deployment of new network technologies varies depending on location, device, and cost characteristics. Even if new technologies

substitute classic networks technologies in the future, in most cases they will coexist in the near future.

The diversity of those network options presents many communication challenges. A heterogeneous wireless communication environment differs strongly in terms of bandwidth, latency, stability, and availability.

Networks

In the example environment shown in Figure 15-4, typical wired networks such as the Enterprise LAN, Dial-In, and Broadband DSL/Cable are extended by wireless networks.

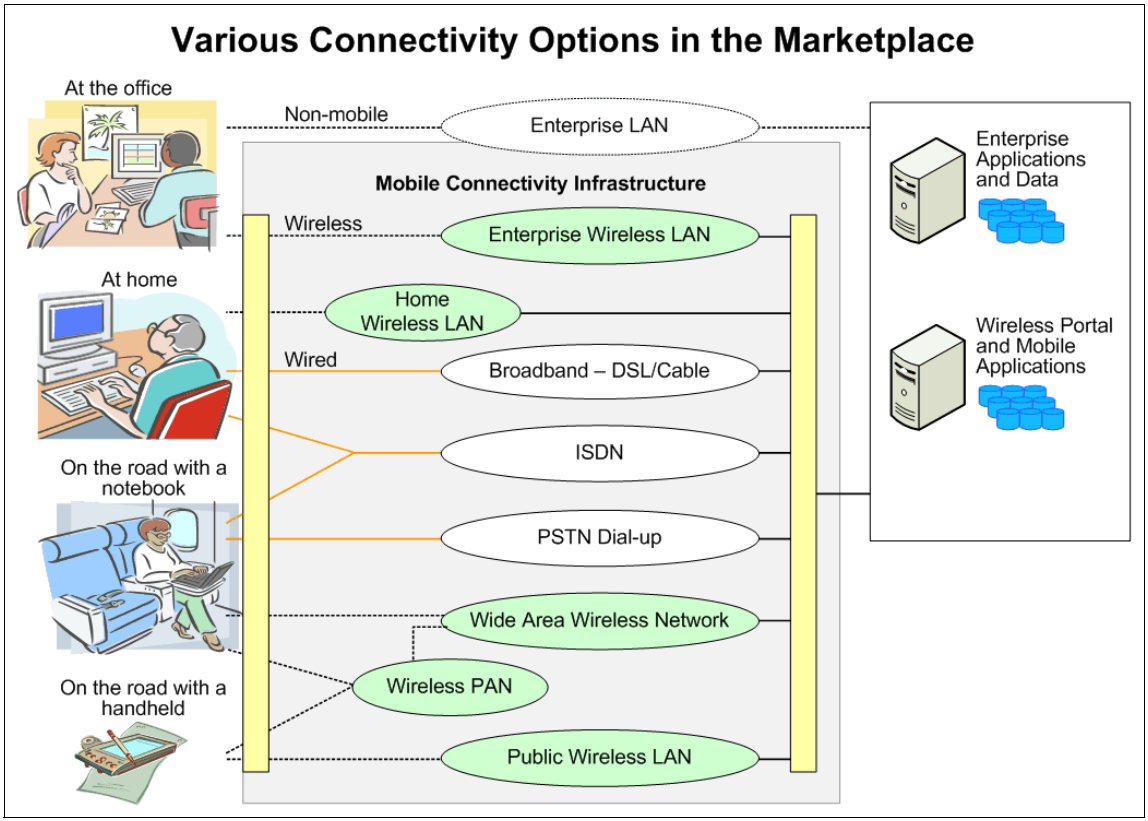


Figure 15-4 Typical network options

The table below shows typical network technologies with theoretical troughput and transport medium.

Table 15-1 Network technologies

Network type	Transport medium	Description	Theoretical troughput Bit/sec
LAN	Wired	10baseT - gigabit Ethernet	10 Mbps–10 Gbps
DSL/Cable	Wired		100 Kbsp–8 Mbps
PSTN Dial-up and ISDN	Wired		28 Kbps–128 Kbps
Wireless network (WLAN)	Wireless/radio frequency	802.11 a,b,g	11 Mbps–54 Mbps
2 and 2.5 Generation Cellular network	Wireless/radio frequency	GSM, CDPD, PDC, HSCD, GPRS	9.6 Kbps–115 Kbps
3 Generation Cellular networks	Wireless/radio frequency	EDGE UMTS,	384 Kbps 2 Mbps
Satellite	Wireless	VSAT	76.8 Kbps (upstream)–40 Mbps (downstream)
Bluetooth	Wireless/radio frequency	PAN	64 Kbps–1 Mbps
Infrared	Wireless/optical	PAN	4 Mbps
USB	Wired	Cradle, USB cable	12 Mbps–480 Mbps

Abbreviations:

- PAN = Personal Area Network
- USB = Universal Serial Bus
- LAN = Local Area Network
- WLAN = Wireless Local Area Network
- PSTN = Public Switched Telephone Networks
- ISDN = Integrated Services Digital Network
- DSL = Digital Subscriber Line
- GSM = Global System for Mobile communications
- PDC = Personal Digital Cellular
- CDPD = Cellular Digital Packet Data
- HSCD = High Speed Circuit Switched Data
- GPRS = General Packet Radio System
- EDGE = Enhanced Data GSM Environment
- UMTS = Universal Mobile Telephone Serv
- VSAT = Very Small Aperture Terminal

Messaging services

Messaging services enables users to stay connected to clients, co-workers, friends, and family even when not able to take a call. It can notify and alert the mobile workforce as well as provide information services. Common services include the following technologies:

- SMS and MMS services
- WAP push

Connection modes

Access to business applications in mobile environments can be classified in two connection modes:

- Online access - Typical browser access, for example. This applies mostly to applications where the business logic is represented on the server side.
- Offline access - Working offline needs more business logic on the device. Data transmission is queued and will synchronize when the network is available.

Roaming

In wireless networking, roaming refers to the ability to move from one network area to another without interruption in service or loss in connectivity. Roaming provides seamless use of voice and data services.

Traditionally, roaming is handled by network operators through contracts and technology bridges. Roaming between a public network and private networks

such as the Internet and the enterprise network are not supported by public network operators.

The sample network setup in Figure 15-5 shows a typical enterprise environment with different locations using private and public networks. Roaming between those networks is becoming more important and should be considered an enterprise mobile access service. Roaming between those networks must be controlled by the enterprise in order to manage security, availability, and network independence.

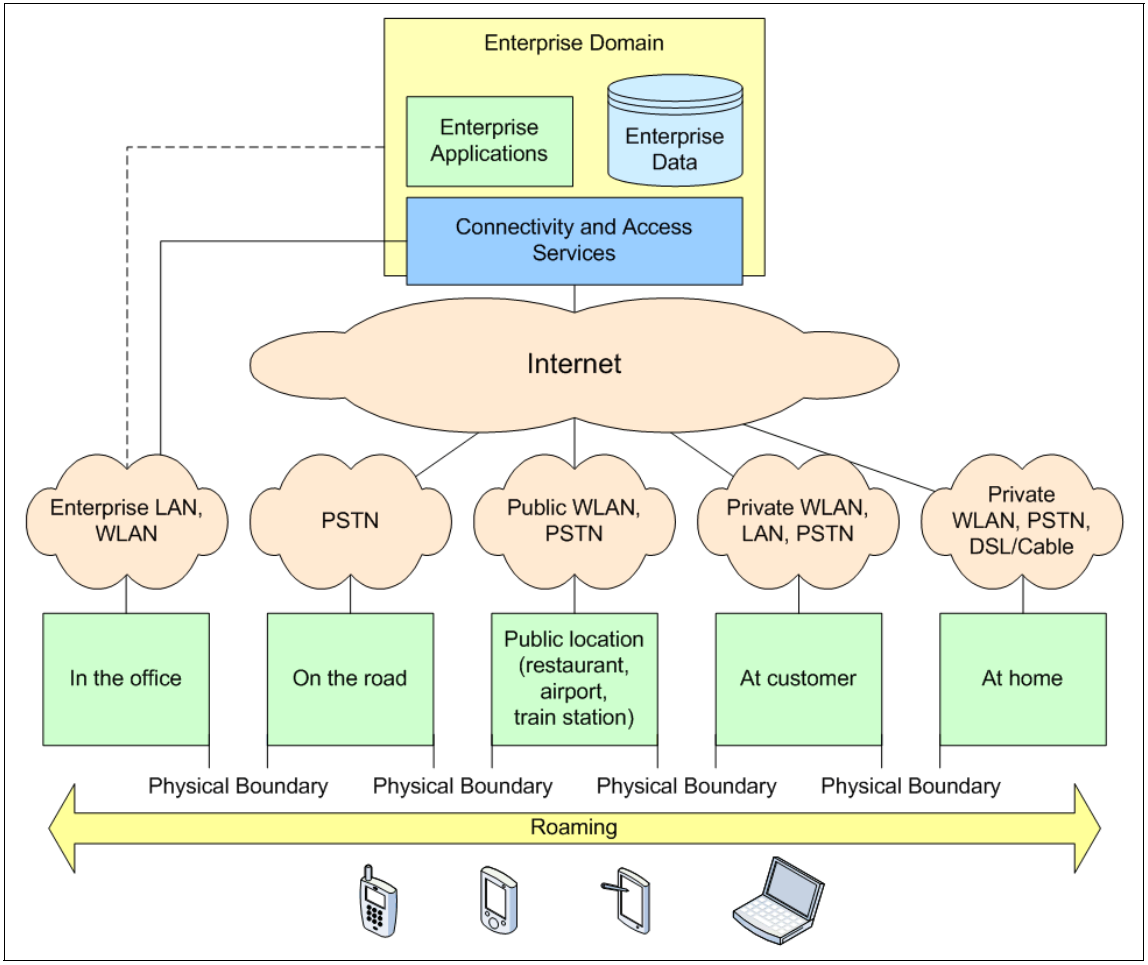


Figure 15-5 Roaming for enterprise networks

Roaming provides seamless connectivity by adding an abstraction layer called *connectivity and access services* in the diagram, which sits between mobile

devices and the enterprise network. This abstraction layer is responsible for handling connectivity to public and private networks. It also enables you to manage connection and session context of mobile applications transparently to user and applications.

Multi-channel and multi-modality

As devices become smaller, modes of interaction other than keyboard and stylus are a necessity. In particular, small handheld devices like cell phones and PDAs serve many functions and contain sufficient processing power to handle a variety of tasks. Present and future devices will greatly benefit from the use of multi-modal access methods.

Multi-channel access is the ability to access enterprise data and applications from multiple methods or communication channels such as a phones, laptops, or PDAs. For example, a user may access his bank account balances on the Web using a browser from the office or from home, and the same user may access the same information over a traditional phone using voice recognition and text-to-speech when traveling on the road.

In contrast, multi-modal access is the ability to combine multiple modes or communication channels in the same interaction or session. The methods of input include speech recognition, keyboard, touch screen, and stylus. Depending on the situation and the device, a combination of input modes will make using a small device easier. For example, in a Web browser on a PDA, you can select items by tapping or by providing spoken input. Similarly, you can use voice or stylus to enter information into a field. With multi-modal technology, information on the device can be both displayed and spoken.

Providing multi-modal access to business data is one of the key topics in the mobile world.

Virtual Private Network (VPN)

A Virtual Private Network (VPN) is an extension of an enterprise's private intranet across a public network such as the Internet, creating a secure private connection, essentially through a private tunnel. VPNs securely convey information across the Internet connecting remote users, branch offices, and business partners into an extended corporate network.

In mobile environment, traditional VPN IP-based solutions are limited in matter of performance, supported client devices, and different network types and services. A mobile access VPN will need to extend VPN services to non-IP networks and devices with limited system resources.

For more information about VPN see the redbook *A Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management*, SG24-5309.

Secure Socket Layer (SSL) offers some basic VPN functions like an encrypted tunnel between client (browser) and server (Web server, Application server, etc.). Instead of encapsulating all data frames in a "VPN" data packet, SSL is limited in protocol and security services. It is a standard encryption method on an application layer, but cannot transport any protocol.

Data compression

Optimizing and reducing the amount of data transmitted over the Internet reduces the response time and connection costs. Data compression reduces the size of data frames to be transmitted over a network link by coding and decoding the transferred packets on both ends of the transport path. A mobile gateway with compression cannot only improve response time, but can also reduce connection cost.

Network Address Translation

Network Address Translation (NAT) and Port Address Translation (PAT) are network technologies that offer flexibility and security. It allows for the separation of the corporate IP address schema from the public Internet address. Enterprise internal IP addresses are hidden from the Internet.

15.1.4 Application environment

As shown in 15.1.1"Mobile environment" on page 361, a mobile environment can consist of different user groups with different needs. Examples are:

- A sales force that needs to get information from many sources while out on the road; access to a groupware application to get e-mail, schedules, and customer information.
- Field force automation workers who need access to technical information and support applications in order to shorten response times and efficiency as well as reduce the paperwork and data quality.
- Delivery and service workers need customer and location information as well as access to the transport backend system for updates and new orders. The delivery company can use location information of the delivery person to rearrange delivery routes and enable a more dynamic business model.
- Home office workers need secure access to their business applications when working from home and abroad. This allows the company to implement new working models in order to increase productivity and decrease costs.

Employees will benefit by more flexibility and the ability to use time efficiently (for example, at the airport or on a train).

- Management workers need to keep in contact with their company to send and receive reports, manage business, schedule events, and receive notifications in urgent situations.

Mobile access provides secure and reliable access to those business applications and information.

Important considerations before building a mobile access are:

- Application architecture
- Programming model
- Communication protocols
- Application security
- Target devices

Application architecture

Knowing the application architecture presented by application overview diagrams, component models, use cases, interaction diagrams, and programming concepts will show how the different components will interact. This is vital information in order to evaluate requirements for the mobile access.

Programming model

There is a wide range of programming models on the mobile environment from standard-based platforms, such as Java and Linux, to vendor-specific applications like .Net or native programming. It contains important information on how to integrate mobile access services into the overall solution. It is also an indicator of future application requirements and communication methods.

Communication protocols

In the context of the application architecture and programming model different communication protocols can be used. It is essential to understand what protocols the applications will use.

Application security

Application security will identify what type of security mechanism will be used in the application. Mobile applications may provide their own security mechanism, which will be combined or enhanced with security mechanisms supported by the mobile access service. Single sign-on capabilities will improve user experience and security, but they require further integration.

Target devices

The target device of the mobile application will indicate which communication and security function will be supported. This topic is related to the following section

15.1.5 Device environment

A growing variety of client devices, which differ in input, output, processing and communications capabilities, will make the implementation of a central connection and access infrastructure difficult. Communications capabilities are diverse, often supporting several communication channels including cellular data (GPRS, HSCSD), WiFi, Bluetooth, infrared, and cradled connections, and each has different bandwidth and latency characteristics. Also, the supported security and encryption mechanism can differ widely, for instance, in the ability to run a user agent.

Device characteristics for mobile access to consider are:

- Device types
- Communication profile
- Security mechanism
- Device operating system and programming model

Device types

There are a huge number of mobile devices in the market and new models emerging every day. They differ in user interface, communication capabilities, operating environment, technology options, and more. A clear classification of those devices is getting more difficult since typical functions are mixed up in many cases. Typical device types are:

- Mobile phones: Devices with a voice interface and numeric keypad only.
- smartphones: smartphones extend mobile phones with a bigger display and user interface, and provide a more powerful platform and browser access; also, some basic PDA functionality like agenda, address book, and text editors.
- PDA devices that focus more on PC type of application environment and user experience. Functionally rich PDAs are able to be an alternative to laptop and desktop PCs in some business environments. A more comfortable user interface allows different input methods such as handwriting recognition, pointing device and device extensions such as external keyboard, memory, network adapters, and so on.
- Tablet and laptop computers that offer a full user experience.

Communication profile

The range of network capabilities of mobile devices is enormous, ranging from cradle through wireless to multi-modal communication. The wide variety of devices makes it difficult to classify devices. In fact, the border between the “classic” PC versus PDA communication capabilities is getting smaller and smaller. Figure 15-6 shows a device communication environment of a PDA with mobile communication capabilities. The communication ability differs on each type of device. Understanding the device communication environment will be a indicator for present and future requirements for the mobile access service. It provides an quick overview of communication technologies for a specific device. A mobile device can access the mobile access way over different network and connection types.

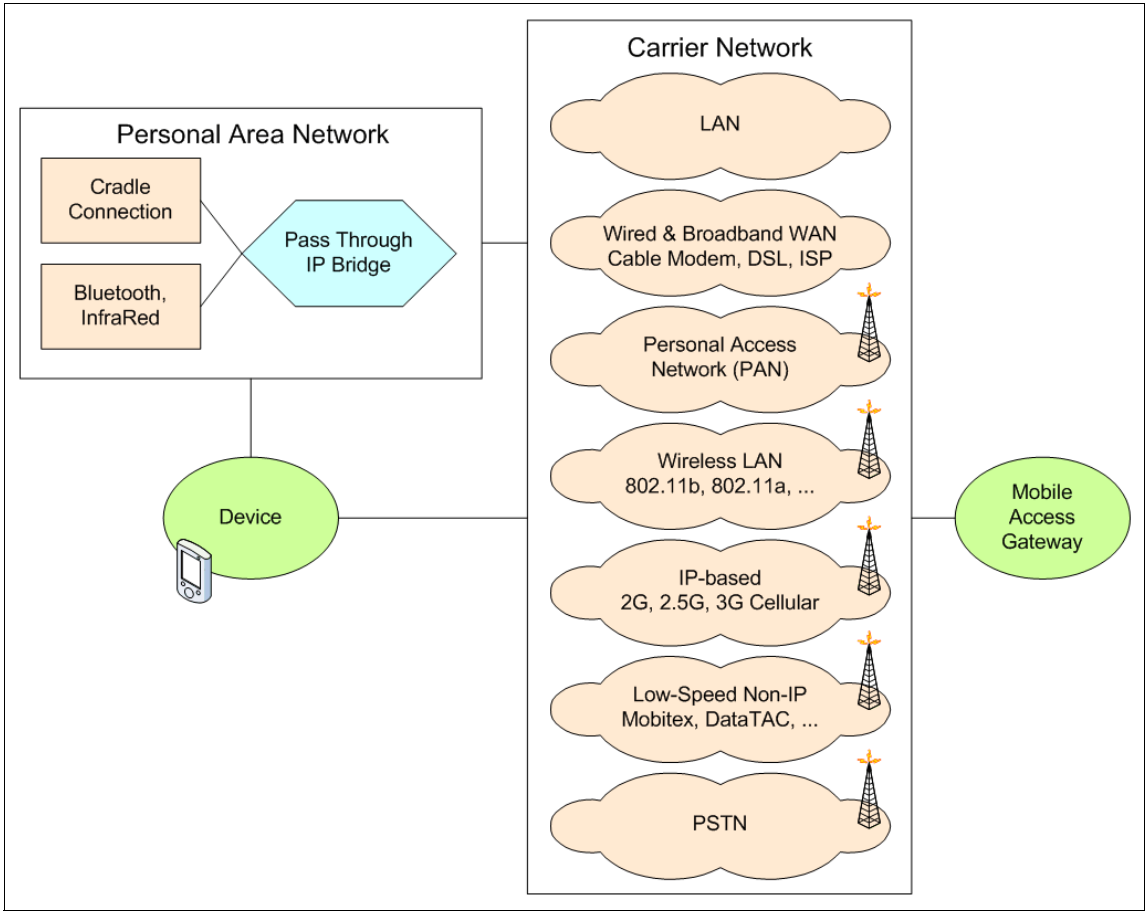


Figure 15-6 Device communication profile

As an example, a PDA can connect to the network using a cradle, Bluetooth, or infrared connection. A passtrough or IP bridge will handle the connection to the network.

Security mechanism

Loss, theft, and other security issues prevent many companies from deploying mobile solutions. Mobile devices are attractive targets to many attackers because the ability to hold and access important business information increased with new technologies and device capabilities. The security mechanism of the device must support the corporate security model for a mobile workforce environment with additional policies, processes, and technologies.

There are two security domains on the device environment: Device and transport security. Device security offers physical device safety and the data protection on the device, while transport security will protect data transmission. Supported security mechanisms on mobile devices can differ widely and must be evaluated to determine the supported security level on the device and decide whether a device will satisfy the corporate security policy and standards.

Device operating system and programming model

The operating system of a mobile device varies based on the device type and manufacturer. Common operating systems are Palm OS, Symbian, Linux, Windows Mobile (WM), RIM OS, and other. A programming model runs on top of the operating system such as Java (J2ME), .Net, and native C and C++ programming.

15.2 Non-functional requirements for mobile access

Non-functional requirements address functions that influence the underlying system architecture. These functions are essential for providing a robust and scalable environment for the mobile access services mentioned in the previous sections. Non-functional requirements are key factors in building a mobile access architecture and the runtime environment.

This section discusses some non-functional requirements important to a mobile access system.

- Availability - High availability minimizes the risk of an outage, and increases the availability of the mobile access system. A system outage can be costly in terms of lost productivity, lost revenue, and lost customers.
- Backup and recovery - It is essential that disaster recovery and backup/restore procedures be properly planned and implemented to meet the high demand for security and safety of the system.

- Capacity estimates and planning - Capacity estimates and deployment planning are important tasks in order to provide good user experience and robust connectivity services. Understanding the application type, network types, troughput, devices, and number of users and actors are key factors.
- Performance - Performance planning and measurements of the mobile access system must follow business requirements and service level agreements. Areas to consider are response time, data troughput, and system load, for example.
- Quality of service - With ever-increasing bandwidth demands being placed on the network, prioritizing traffic is a growing concern. Quality of service (QoS) offers a practical solution for ensuring the guaranteed delivery of critical data.
- Configuration management - Configuration management will gather, organize, and locate configuration information about the mobile access system. Configuration management allows you to have up-to-date information about the system configuration.
- Environment - Technical, physical, social, and organizational environment.
- Extensibility/flexibility - The ability to extend the mobile access system with new services. New emerging technologies and business requirements demand a maximum of flexibility. This goes hand in hand with the topic standards below.
- Standards - Standards are essential in many aspects of the mobile access system. They play a critical role in ensuring safety and quality of the Connecticut services. Standards enable compatibility of functions and communication modules. Standards are used in obligatory regulatory compliance requirements for governments and health care in many countries. Examples are:
  - International Common Criteria Common Methodology for Information Technology Security Evaluation (ISO 15408).
  - Federal Information Processing Standard (FIPS): the US and Canadian government standard for encryption certification.
- Maintainability - The ease of use to maintain a critical system such as a mobile access service is important. Good maintainability will be a key factor for a robust system.
- Reliability - A long mean-time between failures, availability, and correctness are important parameters.
- Scalability - Since business goals and user needs will change over time, scalability addresses the ability to react in order to reduce cost and effort.
- Security - In order to provide secure access to mobile devices, the mobile access system itself must be secure. Secure operating system and secure network access to the mobile access are essential.

- Accounting - Collect accounting and billing information for cost calculation.

15.3 Architectural decisions

Architectural decision records underlying decisions and principles that give the mobile access architecture its characteristics and consistency. An architectural decision discusses different approaches and pollster decisions by considering different options.

Table 15-2 Sample architectural decision

Subject area	Enterprise connectivity - mobile access service
Architectural decision	Mobile access service will enable mobile and remote workers to access corporate applications and information.
Problem statement	The mobile access service should integrate with existing infrastructure and back-end systems. The question is how to best provide connectivity to those systems for different user groups and different applications and information.
Assumptions	Interactions with back-end systems will be a mix of synchronous and asynchronous communications in different connectivity modes like online and offline.
Motivation	Because of the degree of application integration needed, a decision must be made on enterprise connectivity to avoid inconsistent and unmanageable connectivity.
Alternatives	1. Point-to-Point Connectivity - This alternative connects the mobile client directly and individually to each necessary enterprise system and data, using the protocol best suited for that system. 2. Existing Virtual Privat Network (VPN) - This alternative would use an existing VPN used for remote access for PC and mobile devices. 3. Reverse Proxy - A Reverse Proxy can act as a gateway accessible from the internet using an existing network link to the Internet.

Table 15-3 Mobile access services

Subject area	Enterprise connectivity - Buy versus build
Architectural decision	Custom development of access components and services (such as connectivity, load balancing, monitoring, logging, and so on) that are typically available from outside vendors will not be considered.

Subject area	Enterprise connectivity - Buy versus build
Problem statement	Developing a mobile access service will not be considered because of the complexity.
Motivation	Custom-developed middleware and infrastructure components are time-consuming to develop and maintain. Moreover, these components require a great deal of testing and documentation in order to be understood by a team of developers. Finally, developing these custom components takes away valuable time from the task of building business components and logic that can provide an enterprise with an edge over its competition.
Alternatives	Evaluate different products.

15.4 ITSO Railways sample application

This section focuses on designing a mobile access gateway service for the ITSO Railways sample following the information and guidelines described in Chapter 14, “Timetable information by Voice” on page 313. The design will consider the following areas:

- Mobile environment
- Security environment
- Network environment
- Application environment
- Device environment

15.4.1 Mobile environment

The Mobile environment of ITSO Railways has a traditional mobile workforce in place with different user groups and connectivity modes and applications.

- Conductors: Conductors are using PDA-based models in online and offline mode. The devices are usually synchronized at pickup time and after a working day when returning to the depot. The amount of data to be synchronized is highest on the initial sync. The devices are also shared among the different conductors and personalized after pickup. It is planned to leverage wireless networks in various train stations and public spots to synchronize data between the device and the backend server.
- Technical train personal: Technical train personal are responsible for critical technical equipment. In the case of a critical event, the service technican is notified and informed. Technical train personal are using PDA-based devices with a built-in communication module (mobile phone extension). The device

must be robust for usage in a rough environment and sustain rain and dust. The interface must be designed to provide an ease-of-use experience even when working with a handicap like glows. The technical train personal carries a mobile phone as a backup device during their work. Most of the time the technican will be online using any available network.

- Service personal: Service personal are using their devices for maintenance and reporting tasks. It helps the service personal to manage work schedules, orders, and maintenance reports, as well as communication between the team. The device must have an easy-to-use interface, and will communicate mainly with WLAN in train stations and over a public carrier network (GSM, GPRS) in urgent situations.
- Customers: In a first phase the customer should be able to use train schedules, booking and reservation information online and some services like personal schedules and travel information in offline mode for some supported devices. There are plans to have more customer services like notification, news letters, gold member, and location-aware services in the near future. It is the Railways' philosophy to provide an open communication platform to support as many devices as possible.
- Railway management: The Railway management will use business devices such as smartphones or PDA-based devices for mobile offices and to access critical business applications when abroad. The manager will use cradle communication over an existing laptop connection and public carrier networks to access the corporate network.

Table 15-4 Mobile environment summary

User group	Devices	Connection modes	Interaction type	Applications
Conductors	PDA	Online 20% Offline	Data	Train schedules, ticket selling, and reservations system
Technical personal	PDA with communica-tion module, mobile phone, or pager	Online 50% Offline 50%	Data Voice Multimodal	Maintenance, reporting, work schedules, incident notifications
Service personal	PDA with communica-tion module	Online 10% Offline 90%	Data Voice	Maintenance, reporting, work schedules
Management	smartphone PDA	Online 5% Offline 95%	Data Voice	Mobile office (PIM, e-mail) business applications

User group	Devices	Connection modes	Interaction type	Applications
Customers	Any	Online 60% Offline 40%	Data Voice	Train schedules, ticket booking and reservation system, news and notification services

15.4.2 Security environment

The security environment provides the required security services for all business scenarios of the ITSO Railways.

An assessment of the security services by user groups and access information following a security classification guideline are the first steps. In our scenario we have defined the user groups: Conductors, Technical Train Personal, Service Personal, Management, and Customer accessing different applications (assets) described in 15.4.1, “Mobile environment” on page 376. It is assumed that the ITSO Railways has a complete IT security environment in place with a base asset classification, as mentioned below:

- Highly confidential: Highly confidential data like customer name, address, credit card numbers, ticket information, etc., which are not only protected by corporate law, but also by government law.
- Confidential: Confidential data is data that is protected by corporate law.
- Private: Private data is data protected by corporate law, but with less impact to the business.
- Public: Unclassified data.

Table 15-5 on page 379 shows a sample asset classification table that indicates the proposed security mechanism for each asset. We assume that all assets will need some basic security services such as Identification and Authentication, Authorization, Privacy, and Confidentiality and Availability.

Conductors will store customer data on their device, such as credit card numbers, reservations, and a penalty list. This information must be protected from unauthorized access, theft, changes, and loss.

Beside the base services, protecting the data from being changed or lost is the important security service for the technical train personal.

Service personal will have mostly non-critical information on their devices, except their personal data and work reports.

ITSO management will use their devices for mobile office and some business applications. Most of this information is considered highly confidential, since that information can result in remarkable loss or damage for the company.

Customer information consists of public information (train schedules) and personal information.

Table 15-5 Sample asset classification table

User group	Assets	Security classification	Security services
Conductors	Train schedules, ticket selling	Highly confidential	Strong authentication, privacy, and basic security services
Technical train personal	Maintenance system, resource planning, notifiation	Confidential	High integrity and high availability, and basic security services
Service personal	Service maintenance system, resource planning, notifiation	Confidential	Basic security services
Management	Personal information and e-mail, business information	Confidential	Strong authentication and basic security services
Customer	Information and booking system	Private	Basic security services

15.4.3 Network environment

The network environment of the ITSO Railways consists primarily of IP-based internal and external networks. An existing internal Ethernet LAN with private WLAN networks and an Internet DMZ architecture.

The figure below shows a high-level architecture of the ITSO Railways network. It consists of an internal and external network separated by a security architecture with firewalls and access control symbolized by a firewall in the diagram. The mobile access service will enhance this architecture by a mobile gateway to allow access to the corporate network for a mobile user.

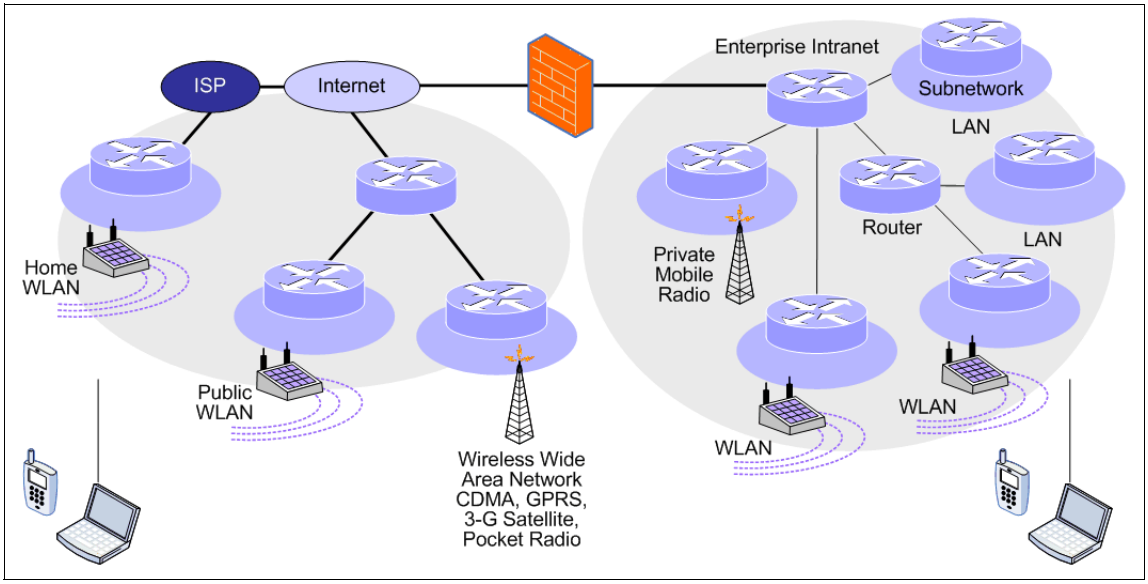


Figure 15-7 ITSO Railways network

The services are:

- Messaging services: The ITSO Railways generates SMS messages for notification services and customer services. There are further plans to set up a WAP gateway to support WAP push messaging and MMS for the technical workforce.
- Roaming: Since the ITSO Railways has many different networks and user groups travelling between the different network boundaries, roaming is considered a core service of a mobile access service.
- Virtual Private Network (VPN): Most of the ITSO Railways user groups access confidential and private information using internal and external networks. A mobile VPN will provide necessary transport security and is considered a core service.
- Data compression: Since the ITSO user groups communicate not only over high bandwidth but also low bandwidth networks, data compression can improve response time and reduce connection cost. Data compression is considered an added-value service and a recommended mobile access service.

Table 15-6 Mobile Access Network Service

User groups	Mobile access services
Conductors	VPN, roaming, data compression

User groups	Mobile access services
Technical train personal	VPN, roaming, data compression
Service personal	VPN, roaming, data compression
Management	VPN, roaming, data compression
Customers	Roaming

15.4.4 Application environment

The application environment describes the application architecture, the programming model, communication protocols, application security, and the target device.

In the ITSO example we have different use cases described. See Chapter 5, “ITSO Railway sample overview” on page 87, for more information about the use cases, application overview, and communication information.

Table 15-7 shows an overview of the ITSO Railways applications in context with the communication, target device, application security, and programming model. This table outlines the ITSO scenarios relevant for the mobile access with additional information such as a communication profile, target devices, application security, and programming model.

Table 15-7 Scenario table

ITSO scenarios	Description	Communication	Target device	Appl. security	Prog. model
Mobile Inventory management	Forms-based applications on mobile devices	Protocol: http/ https Mode: Online, offline	PDA (WM, Palm), smartphones (Symbian)	Delegation to J2EE security profile	J2EE
PIM/Emily synchronization	Synchronization of PIM/e-mail information on mobile devices	Protocol: http/https Mode: Online, offline	PDA and (WM, Palm), smartphones (Symbian), mobile phones (SyncML)	Delegation to J2EE security profile	J2EE, J2ME
Intelligent Notifiation	Alerts to maintenance workers	SMS, Instant Messaging, e-mail	PDA(WM, Palm, Symbian). smartphones. mobile phones	Delegation to J2EE security profile	J2EE, J2ME

ITSO scenarios	Description	Communication	Target device	Appl. security	Prog. model
Device Management	Maintenance of mobile devices	Protocol: http/https Mode: Online	PDA (WM, Palm), smartphones (Symbian)	Delegation to J2EE security profile	J2EE, J2ME

15.4.5 Device environment

The ITSO Railways wants to use state-of-the-art devices that fit the best for the different user groups. Since the device environment is changing rapidly, ITSO Railways wants to be able to support current and future needs. It plans to use devices described in the table below.

Table 15-8 ITSO device matrix

User group	Device type	Network capabilities	Security mechanism	Operating system
Conductors	PDA-based devices with customized user interface	Cradle, Bluetooth, WLAN	Power-on password, SmartCard	WM, Palm
Technical train personal	PDA-based devices with customized user interface	Cradle, Bluetooth, WLAN, mobile communication module (GSM, GPRS)	Power-on, SmartCard	WM, Palm
Service personal	Standard PDA-based devices with standard user interface	Cradle, Bluetooth, WLAN	Power-On, SmartCard	WM, Palm
Management	Standard PDA, smartphones, mobile phones	Cradle, Bluetooth, WLAN, public	Power-ON, SIM	WM, Palm, Symbian
Customers	Some supported	Some supported	Some supported	Some supported

15.4.6 Non-functional requirements

This section outlines the non-functional requirements of the ITSO mobile access service. Non-functional requirements are shown in the table below.

Table 15-9 Non-functional requirements

ITSO non-functional requirement	Description
Availability	High availability will ensure access to business-critical information.
Backup and recovery	Existing and planned backup and recovery processes will minimize outage time.
Capacity Estimates and planning	Expected amount of data, connections, and concurrent users.
Performance	The mobile access service must provide necessary performance for all user groups. Horizontal and vertical load balancing and clustering will distribute the load to different systems and resources.
Quality of service	Critical and high-priority classified connections.
Configuration management	Configuration changes must be stored and maintained for recovery and audit purpose.
Extensibility/flexibility	The mobile access service must be scalable in a matter of functionality and the ability to integrate new services.
Standards	ITOS Railways needs a flexible environment in order to deploy new services quickly and protect the investment.
Maintainability	ITSO Railways has limited resources to maintain the system. The mobile access service must support easy maintainability.
Reliability	The mobile access service must provide a robust platform with a minimum of failure.
Scalability	The solution must provide horizontal and vertical scalability.
Security	A mobile access service will bridge an outside network to an internal network. Therefore security is an essential requirement. The mobile access service itself must run in a secure environment and should be able to adopt the corporate security standard.

ITSO non-functional requirement	Description
Systems Management	The mobile access gateway must be able to be integrated in the existing system management environment of the ITSO Railways.
Accounting	The ability to extract accounting and billing information will provide the base of financial ratings.

15.5 Mapping ITSO Sample application requirements

This section outlines how IBM WebSphere Everyplace Connection Manager (WECM) fits to ITSO Railways mobile access service requirements.

WebSphere Everyplace Connection Manager provides a standard TCP/IP communications interface to a variety of wireless, dial-up, and local-area networks (LAN) with data optimization and security. The components that make up WebSphere Everyplace Connection Manager are designed to run on multi-vendor hardware and operating system platforms. The main functions are Mobile Virtual Private Network (VPN), Seamless network roaming, and a wide spectrum of communication protocols for both Internet Protocol (IP) and non-IP networks data optimization functionality to reduce transmission costs and lower connection fees. The three major components of WebSphere Everyplace Connection Manager are:

- Connection Manager
  - Messaging services: Enables a Web application server to send messages to messaging clients, such as a pager or a phone, using a variety of wireless networks. Messaging services include support for short message service (SMS) delivery, mobile-originated message delivery. When installed with the WAP proxy, messaging services also includes support for unconfirmed Wireless Application Protocol (WAP) push delivery.
  - Mobile access services: Creates an optimized and secure IP tunnel for communication to the Mobility Client software on your computer. The Mobility Client can use a wireless or wired connection to the mobile access services that connects to your company's private intranet to the Internet.
  - WAP proxy: Performs a protocol conversion between Hyper Text Transport Protocol (HTTP) and Wireless Application Protocol (WAP) protocols to link WAP clients with Web-based browser services or TCP application services.

- Gatekeeper: An easy-to-use administrative interface that enables you to define and manage wireless resources.
- Mobility Client software: This software runs locally on user devices and provides a full-function interface to communicate with the Connection Manager. Once Mobility Client authenticates to WebSphere Everyplace Connection Manager, a Virtual Private Network (VPN) is established, and the device securely joins the enterprise intranet.

15.5.1 Requirement mapping

Mapping of the ITSO Railways requirements to WebSphere Everyplace Connection Manager functions will consider four areas: Security, network, device, and application.

The tables below show how required ITSO Railways functions are mapped to WebSphere Everyplace Connection Manager functions.

Table 15-10 Mapping of security requirements to WECM function

Mobile access services	Service description	WECM mobile access service	Details
Security	Identification and authentication	Basic and strong authentication	- Single-party key distribution protocol. - Two-party key distribution protocol. - Diffie-Hellman key agreement algorithm.
	Authorization	Connection authorization	A secure sockets layer (SSL) or Wireless Transport Layer Security (WTLS) connection can be authorized.
	Confidentiality and privacy	Communication link confidentiality	Data encryption prevents unauthorized access. The data is transformed into encrypted data using the session key that is exchanged during the authentication process.
	Integrity	Transport integrity	Integrity of the encapsulated data traffic is maintained.

Mobile access services	Service description	WECM mobile access service	Details
	Availability	Clustering	The Connection Manager can be configured to be a principal or subordinate node in a cluster. In this manner, the Connection Manager distributes and services communication requests and provides load-balancing efficiency.

Table 15-11 Mapping of Network requirements to WECM function

Mobile access services	Service description	WECM mobile access service	Details
Network	VPN	Mobile access service.	Wireless applications using IP or non-IP packet-oriented connection, such as Mobitex or DataTAC, are supported by mobile access services.
	Roaming	Seamless cross-network roaming.	Seamless cross-network roaming; session context.
	Data compression	Data compression TCP optimization.	Reduces the amount of data transmitted with data compression and protocol optimization.
	Network support (IP, WLAN, PSTN, Cellular networks)	Connection Manager supports a variety of wireless and dial-up network technologies.	Connection support for IP and non-IP networks.
	Messaging Services (Short Message Services, SMS, WAP)	Messaging Services.	Connection Manager supports SMS, e-mail, and WAP push.
	Network Address Translation (NAT)	NAT and Port Address Translation (PAT).	NAT/PAT lets the Connection Manager act as an agent between a public network and a private network.

The ITSO Railway network architecture with WebSphere Everyplace Connection Manager is shown in the next diagram.

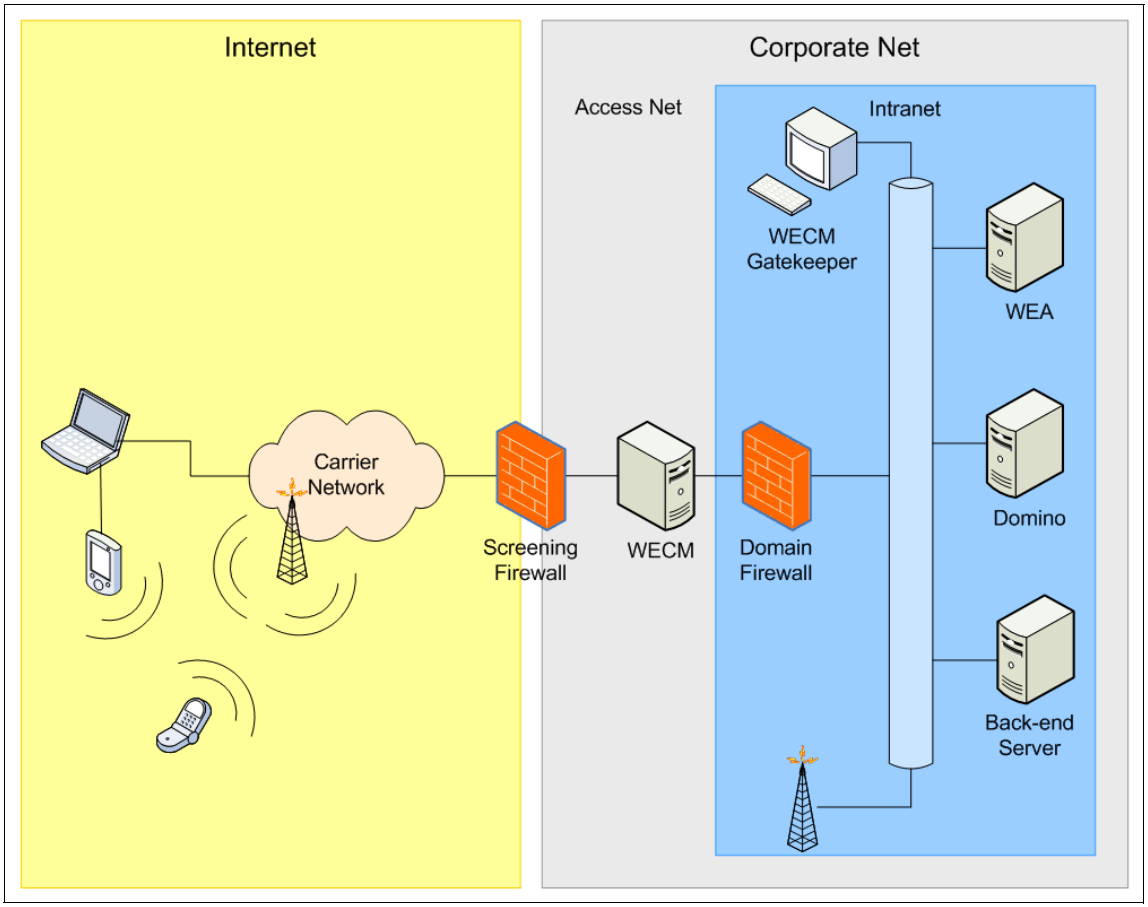


Figure 15-8 WebSphere Everyplace Connection Manager placement at the ITSO network

WebSphere Everyplace Connection Manager will be placed in the DMZ “Access Net” behind the screening Internet firewall and right before the domain firewall. The WebSphere Everyplace Connection Manager gateway will also use filters to secure the access from the mobile clients. The management console gatekeeper will be placed in the internal network and hold the configuration and logs.

Table 15-12 Mapping of application requirements to WECM function

Mobile access services	Service description	Communication	WECM mobile access service
Application	Forms-based applications on mobile devices	Protocol: http/ https Mode: Online, offline	Mobile access services
	Synchronization of PIM/e-mail information on mobile devices	Protocol: http/https Mode: Online, offline	Mobile access service
	Alerts to maintenance workers	SMS, Instant Messaging, e-mail	SMS, mobile access services, e-mail
	Maintenance of mobile devices	Protocol: http/https Mode: Online	Mobile access services

Table 15-13 Mapping of device requirements to WECM function

Mobile access services	Service description	Communication	WECM mobile access service (device support)
Devices	PDA-based devices with customized user interface	Bluetooth, WLAN, mobile communication module (GSM, GPRS)	WM Palm Symbian Linux Windows
	Standard PDA-based devices with standard user interface	Bluetooth, WLAN	
	Standard PDA, smartphones, mobile phones	Bluetooth, WLAN, Public	
	Some supported	Some supported	

Table 15-14 Mapping of non-functional requirements to WECM function

ITSO Railways non-functional requirement	WebSphere Everyplace Connection Manager gateway function
Backup and Recovery	WebSphere Everyplace Connection Manager stores configuration in the files system (config files, database, and directory).
Performance	WebSphere Everyplace Connection Manager support multiprocessor machines, and multiple Connection Managers can be configured as a cluster to distribute workload.
Configuration management	Gatekeeper management console. WebSphere Everyplace Connection Manager stores configuration in the files system, database, and directory.
Extensibility/flexibility	A comprehensive programing reference and toolkit allows the extension of connection services.
Standards	WebSphere Everyplace Connection Manager runs on standard Unix systems and supports standard networks. WebSphere Everyplace Connection Manager is being FIPS 140 certified on secure platforms.
Maintainability	The Gatekeeper is an easy-to-use administrative interface that enables the definition and management of wireless resources, the registration of users and mobile devices, the specification of logging and tracing controls, and the performance of many other administrative tasks.
Reliability	WebSphere Everyplace Connection Manager runs on reliable Unix platforms.
Scalability	WebSphere Everyplace Connection Manager cluster support enables scalability. WebSphere Everyplace Connection Manager runs on open platforms and provides functional scalability.
Security	WebSphere Everyplace Connection Manager supports a high security environment and can provide standard-based security.
Systems management	WebSphere Everyplace Connection Manager can leverage existing system management standards like SNMP, and supports systems management solutions.

ITSO Railways non-functional requirement	WebSphere Everyplace Connection Manager gateway function
Accounting	WebSphere Everyplace Connection Manager can collect accounting information.

16

Maintaining mobile devices

The falling prices of mobile devices and increase in connectivity options in the pervasive market are leading to a rising penetration of these devices. Enterprises will want to take advantage of this trend by providing ways to give access to enterprise data using these devices.

Maintenance of mobile devices, such as PDAs, mobile phones, or even notebooks, is gaining importance in the enterprise. But the management of the mobile devices that are low on computing power and not constantly connected is challenging. Methods for device management known to the server and workstation environment must be extended to mobile devices. At the same time, management systems should be able to enforce certain security guidelines.

This chapter describes one possible approach for a mobile device management solution. Requirements and use cases from the ITSO Railway example (Chapter 5, “ITSO Railway sample overview” on page 87) are used throughout the chapter.

16.1 Overview

This chapter is a summary of requirements extracted from ITSO Railway's inputs for a mobile device management system.

The following diagram is the Runtime pattern we implemented in this particular scenario.

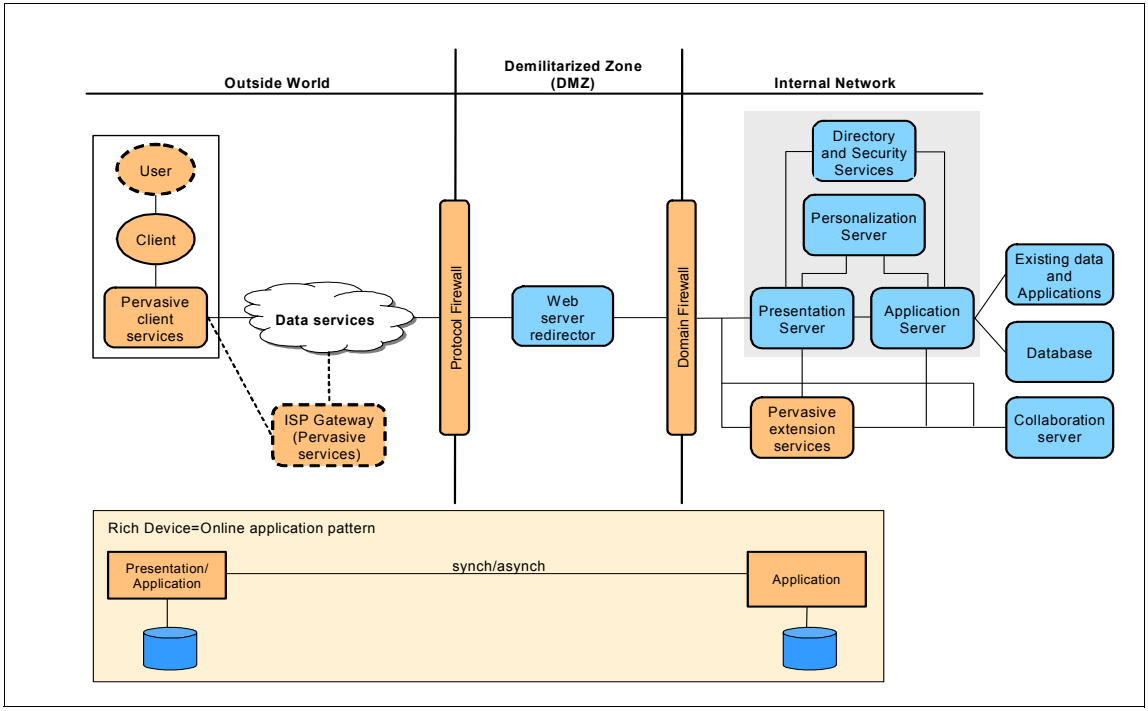


Figure 16-1 Runtime pattern for the device management scenario

The Product mapping for the scenario can be seen in the following diagram.

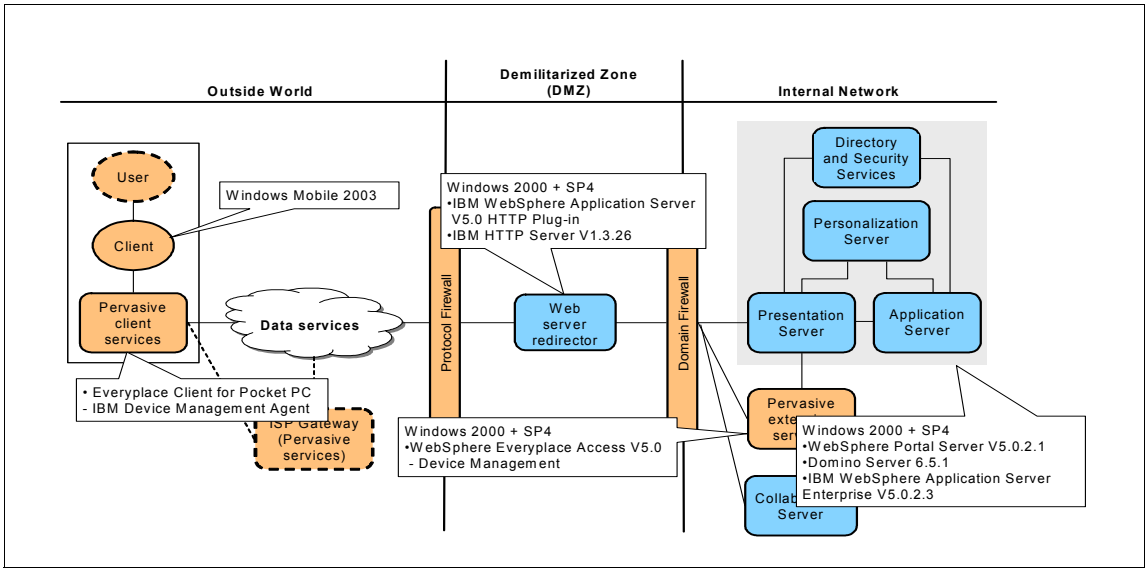


Figure 16-2 Rich Device=Online::Product mapping=Device Management, Windows 2000

16.1.1 Customer requirements

The fictive ITSO Railway company was created to explain general approaches for mobilizing applications. Examples of customer wants and needs are described and listed in Chapter 5, “ITSO Railway sample overview” on page 87.

The following business context was identified for a device management system (see Figure 16-3 on page 396). Since ITSO Railways has introduced new mobile solutions to their employees (for example, the new mobile inventory system from Chapter 11, “Mobile Inventory Management with offline forms” on page 211, and the new ticketing application from Chapter 13, “Using Workplace Client Technology, Micro Edition” on page 283), a management system for the new mobile devices is required. The main functions that must be covered by the new mobile device management solution are:

- The installation and removal of software
- Device software inventory collection
- Remote device configuration

See the requirements from ITSO Railway in 5.10, “Maintain the mobile devices” on page 97. A user interface for the administrator of the new device management system is also desired in order to be able to create appropriate device management jobs from a single administration point.

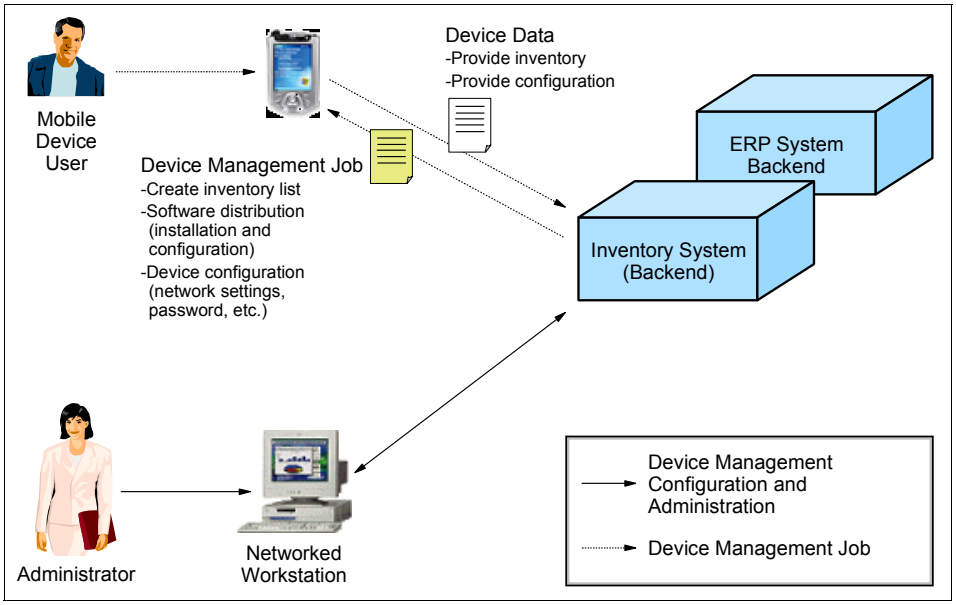


Figure 16-3 Business context for ITSO Railway device management example

The following sections list the captured functional and non-functional requirements to define the baseline according to which the business system must be designed. A sample use case model with the most important use case is also created. The deployed device management solution must adhere to these use cases.

16.1.2 Functional requirements and use case model

The functional requirements of our ITSO Railway device management system example are extracted from the customer wants and needs. They provide the main input for the use case model.

We simplified the use case model for the ITSO Railway device management example and reduced the use case description to the following constructs:

- Actors (name, description, status, superclass, subclass, and associations)
- Use cases (number, subject area, business event, name, overview, preconditions, description, associations, inputs, outputs, usability index, and notes)
- Communication Association diagram

The presented project approach for the ITSO Railway device management system example is mainly based on the IBM Global Services Method recommendation (also see 11.1, “Overview” on page 212).

Actors

The following actors are of interest for the new device management solution:

- Mobile device user
- Device management agent trigger
- Device management agent
- Administrator

Figure 16-4 shows the actors and their interaction for the ITSO Railway example. Detailed descriptions to these use case actors can be found in Table 16-1 on page 398 through Table 16-4 on page 399.

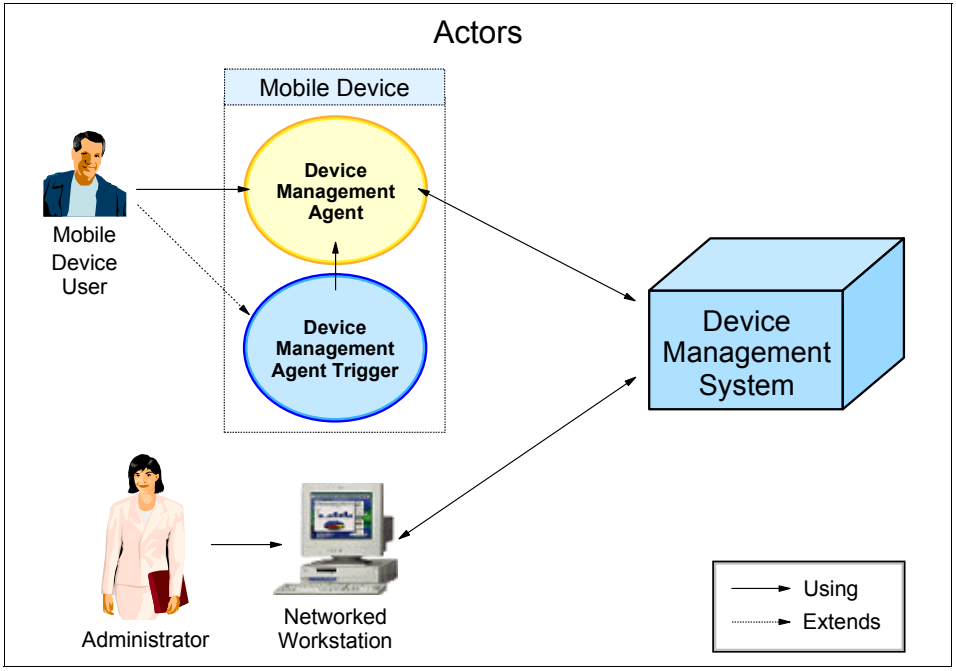


Figure 16-4 ITSO Railway device management system - Use case actors

Table 16-1 Use case actor mobile device user

Actors name	**Mobile device user**
Brief description	The mobile device user is the person in the company who works with a mobile device in online and offline mode. More then one application can run on the device, and are used for different business purposes (for example, e-mail, train supply recording, ticketing system). The mobile device user can initiate DMS job queries (by starting the device management agent).
Status	Primary.
Relationship	
Inheritance	Subclass: None.
	Superclass: Device agent trigger.
Association to use cases	Initiate job query.

Table 16-2 Use case actor device management agent trigger

Actors name	**Device agent trigger**
Brief description	The device agent trigger is a local program that can automatically start the device management agent. This trigger can be controlled locally on the device (for example, by the WEA client, by a timer program) or remotely (for example, through a SMS).
Status	Secondary.
Relationship	
Inheritance	Subclass: None.
	Superclass: Mobile device user.
Association to use cases	Initiate job query.

Table 16-3 Use case actor device management agent

Actors name	**Device management agent**
Brief description	The device management agent is located on the mobile device and is the corresponding client program to the device management server. The agent queries the server for assigned jobs and executes them appropriately.
Status	Primary.
Relationship	
Inheritance	Subclass: None.
	Superclass: None.
Association to use cases	Enroll device, run job.

Table 16-4 Use case actor administrator

Actors name	**Administrator**
Brief description	The administrator of the inventory system maintains the system, deploys the Supply Consumption Record application, and makes it available to the delivery person's PDA.
Status	Primary.
Relationship	
Inheritance	Subclass: None.
	Superclass: None.
Association to use cases	Create software package, submit and assign job.

Use cases

Use cases describe functional requirements. They are used as inputs for the system and application design. Furthermore, use cases describe the potential uses of the solution delivered to the customer. The final solution test runs utilize the use cases the customer and the system provider agreed to at project start.

The following use cases were considered in the ITSO Railway device management system (DMS) example:

1. Create software package.
2. Submit and assign job.

3. Enroll device.
4. Initiate job query.
5. Run job.

Device management use cases can be very complex. Because of this, the Run Job use case covers the following use cases in this chapter:

- Software distribution
- Inventory collection
- Bootstrap (initial installation and configuration of device)
- Device configuration
- Application configuration

Table 16-5 on page 402 to Table 16-9 on page 407 list these use cases in detail.

Use case diagram

Actors and use cases communicate to execute the interaction described in the use case. This communication is modeled as a communication-association diagram shown in Figure 16-5 on page 401. The direction of the arrow shows the direction in which the communication is initiated. In addition, Figure 16-5 on page 401 includes an application workflow example for the ITSO Railway software distribution example use case.

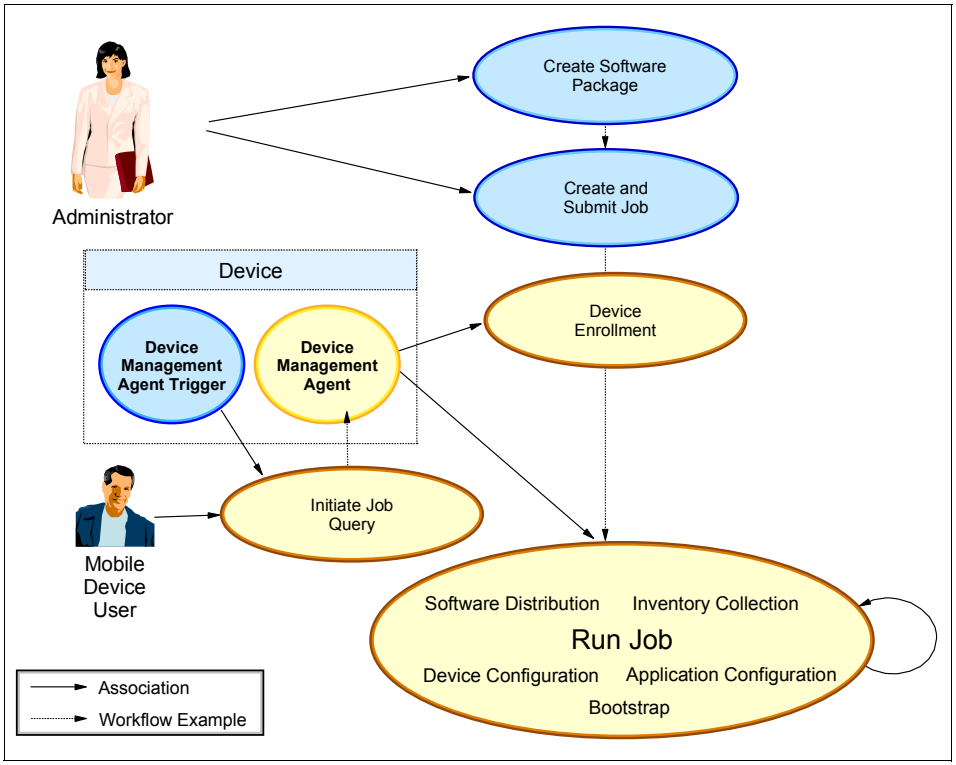


Figure 16-5 Use case association diagram for ITSO Railway device management solution

The Administrator creates the software package for distribution. Appropriate dependencies between the new software and existing versions and support programs are considered. It might be necessary to include additional software in the package to resolve these dependencies. Now the Administrator can create the device management job to distribute the created software package (software distribution job). He makes this job available to the appropriate user group or device group.

The device management agent (DM Agent) and the device management agent trigger (DM Agent Trigger) run on the mobile device. The mobile device user can interact with the device management agent directly in order to initiate a server contact (initiate job query). The user must provide a user name and password, which the agent must provide to the server at the beginning of each server connection.

The DM Agent Trigger is a program that can automatically trigger the DM Agent to connect to the server. This trigger process could be kicked off periodically by a

local timer program or through other software such as the WebSphere Everyplace Client. In addition, the DM Agent Trigger could also be started by an external process (for example, a server component). Other communication channels could also be used for triggering, such as SMS messaging (if the mobile device is capable of receiving SMS).

The first contact with the server during which the mobile device is registered with the device management server is called enrollment. After this, the device management agent can contact the server and query it for available jobs. If jobs are available for the device, they are either immediately executed or deferred. This is configurable during the job configuration time.

Log records and/or user messages must be provided to be able to track job execution. If a job fails, the reason must be determined. The job must be re-submitted for a new execution on the effected device.

The following tables describe the five major ITSO Railway mobile device management use cases.

Table 16-5 Use case 1: Create software package

Use case #1 **Subject area**	**Create software package** **Administration**
Business event	New software for mobile devices is available (additional software, new releases, fixes, etc.).
Actors	Administrator.
Use case overview	Administrator packages software for distribution through device management server.
Preconditions	Software to be distributed is available for target device platform.
Termination outcome	**Condition effecting Termination Outcome**
1) Software package ready for distribution	- Authentication to DMS server is successful. - Software package to be distributed is completely provided (including software prerequisites for mobile device). - Installation order of software pieces is considered.
2) Software not ready for distribution	- Administrator could not log on to DMS server. - Software package is incomplete (software prerequisites for mobile device are not available). - Installation order of software pieces is not considered.

Use case description	Administrator logs on to DMS server. He creates a software package by creating necessary package information and installation files (using a wizard). Software to be distributed and necessary perquisite software is packaged together.
Use case association	Used by Create and Submit Job (for software distribution) use case.
Inputs summary	Authentication data, software and perquisite software to be distributed.
Output summary	Software bundled ready for distribution.
Usability index	Administrator console has to provide support through package wizard.
Use case notes	Use case is only necessary for software distribution jobs.

Table 16-6 Use case 2: Create and submit job

Use case #2 **Subject area**	**Create and submit job** **Administration**
Business event	Information about the mobile device is needed (for inventory), new software must be delivered to mobile device or mobile device configuration must be changed.
Actors	Administrator.
Use case overview	Administrator creates device management job on device management server and assigns it to target devices and/or users.
Preconditions	DMS is up and running.
Termination outcome	**Condition effecting termination outcome**
1) DMS Job created successfully	- Authentication successful. - Plug-in for target device type available (target operating system and DMS technology supported). - Software package for software distribution job is available. - Device configuration information is available for configuration job.
2) DMS job not created	- Authentication is not successful. - Plug-in for target device type not available. - Package for software distribution job is not available. - No device configuration information available for configuration job.

Use case description	Administrator logs on to the DMS administration user interface. He creates a device management job using a wizard. Administrator assigns the job to target device class, user, or user group. He also configures the job schedule. Depending on the job type additional configuration steps are necessary: - Software distribution job: Integrate software package. - Configure device job: Integrate device configuration. - Bootstrap job: Integrate software and device configuration.
Use case association	- Used by Run Job use case. - Needs Create Software Package use case for software distribution job.
Inputs summary	- Authentication data. - Software package for software distribution job. - Device configuration data for device configuration job.
Output summary	Created DMS job.
Usability index	Administrative tasks must be supported through wizards.
Use case notes	In order to simplify the ITSO Railway example the following use case summarizes these more specific use cases: - Software distribution - Inventory collection - Device and application configuration - Bootstrap

Table 16-7 Use case 3: Enroll device

Use case #3 **Subject area**	**Enroll device** **Administration**
Business event	Mobile device user receives a new device (PDA) and needs to configure it.
Actors	Device management agent.
Use case overview	Mobile device enrolls with the DMS server.
Preconditions	DMS client software (device management agent) was pre-installed on mobile device.
Termination outcome	**Condition effecting termination outcome**

1) Device enrolled	- TCP/IP connection from device to DMS server is present (device cradled to network attached service station) and configured. - Authentication with DMS server successful (user belongs to DMS user group).
2) Device not enrolled	- TCP/IP connection from device to DMS server is not present and/or not configured. - Authentication with DMS server not successful.
Use case description	Mobile device user receives a new PDA, which is preloaded with the device management agent. The user places the PDA in the network-attached cradle and starts the device management agent. He enters user name and password and connects to the DMS server (TCP/IP connection is present). Device management agent enrolls with DMS server. DMS jobs, which were created for newly enrolled devices and match the user name, device class, and/or user group are immediately performed.
Use case association	- Necessary for all Run Job use cases (apart from Bootstrap use case). - Runs Initiate Job Query use case after first contact automatically.
Inputs summary	Authentication data.
Output summary	Device enrolled with DMS server and ready for receiving DMS jobs.
Usability index	Device management agent must be preloaded or installed through Bootstrap job. Network connection through cradle should be ready for user before.
Use Case Notes	Device enrollment is necessary before device jobs can be run on the device.

Table 16-8 Use case 4: Initiate job query

Use case #4 **Subject area**	**Initiate job query** **Manage device**
Business event	New job is available.
Actors	Mobile device user (manual procedure), device management agent trigger (automatic procedure)
Use case overview	Device management agent is started to query DMS server for assigned jobs.

Preconditions	Device was enrolled with DMS server and is configured for DMS server access. Device management agent trigger is available for automatic procedure.
Termination outcome	**Condition effecting termination outcome**
1) Server query was successful, job is available	- TCP/IP connection to DMS server is available. - User or trigger started device management agent and connects to DMS server successfully. - Authentication with DMS server was successful. - One or more jobs for device were assigned and are available.
2) Server query was successful, no job available	- TCP/IP connection to DMS server is available. - User or trigger started device management agent and connects to DMS server successfully. - Authentication with DMS server was successful. - No jobs for device is available.
3) Server query not successful	- TCP/IP connection to DMS server is not available. - User and/or trigger cannot start device management agent and cannot connect to DMS server. - Authentication with DMS server fails.
Use case description	The mobile device user is informed about newly available DMS jobs for his device or the device management agent trigger was kicked off by another process (for example, locally by a timer or remote through a SMS, see server initiated action). The device management agent is started (either by the user or the Agent Trigger) and queries the DMS server for new jobs. If new jobs for the particular device and user are available, the user can be informed and the device management agent can start executing them.
Use case association	Necessary for all Run Job use cases (apart from Bootstrap use case).
Inputs summary	
Output summary	DMS jobs can be executed.
Usability index	The device management agent must be triggered for server contact. This can be done manually by the mobile device user (for example, during each synchronization process) or automatically by a Agent Trigger (for example, local timer).
Use case notes	Agent Trigger needs to be able to run in background of device.

Table 16-9 Use case 5: Run job

Use case #5 **Subject area**	**Run job** **Manage device**
Business event	New job is available.
Actors	Device management agent.
Use case overview	DMS job is executed on mobile device.
Preconditions	Device management agent has successfully queried DMS server and job for mobile device is available.
Termination outcome	**Condition effecting termination outcome**
1) Job executed successfully	- TCP/IP connection from device to DMS server is present (device cradled to network attached service station), configured and available during job download. - Prerequisites for DMS job are met or can be resolved.
2) Job not executed	- TCP/IP connection from device to DMS server is not present, lost during execution and/or not configured. - Prerequisites for job execution are not met and can not be resolved either.
Use case description	After the Initiate Job Query use case runs successfully and detects DMS jobs for the device, these jobs are executed. After successful completion of the job, a message is created for the user and the administrator (message window or log). In case of temporary failures, the job execution (network problem) can be resumed (check-point-restart procedure). In case of not resolvable problems, the job will be canceled and not executed. An error message will be created (for the mobile device user and the administrator).
Use case association	Is initiated by Initiate Job Query use case.
Inputs summary	
Output summary	DMS job dependent data transmission.
Usability index	This is an automatic process kicked off by Initiate Job Query use case.

Use case notes	Different DMS jobs are summarized in this use case: - Software Distribution - Inventory Collection - Bootstrap (software distribution and configuration) - Device Configuration - Application Configuration

Regarding the ITSO Railway scenario and considering the scope of this book, the example software distribution use case described in 5.8.2, “Example application scenario” on page 95, was chosen for detailed consideration. In this use case, a remote display control tool (http://msdn.microsoft.com/mobility/downloads/tools/default.aspx) will be installed on the ITSO delivery peoples PDAs (see Chapter 16, “Maintaining mobile devices” on page 393). This allows the ITSO Railway IT support to remotely connect to the cradled PDA in order to see and control the user screen. It offers the capability to remotely investigate the PDA in case of problems and explain user interactions to a delivery person directly on the screen of the PDA (Similar tools exist for workstations and are called Remote Desktop Control).

The necessary software distribution job for the remote display application is created assuming that the target Pocket PC 2002 devices were already enrolled with the WebSphere Everyplace Access device management server. When the PDA is connected and synchronized, the remote display software will be installed to the assigned devices. The following use cases will be implemented in the following chapters:

- Create software package (for remote display)
- Create and submit job (for remote display)
- Initiate Job query
- Run job (software distribution job for remote display)

See 16.4, “Deployment and runtime configuration” on page 418.

16.1.3 Non-functional requirements

We are looking at these non-functional requirements from a more generalized view for our simplified ITSO Railway device management example. Many of those requirements are addressed through the choice of the device management server included in WebSphere Everyplace Access. Since the device management server runs on WebSphere Application Server, requirements like scalability, availability, security and data integrity are covered.

The non-functional requirements we are concentrating on in our example are not complete. From our perspective, we focused on the most important ones for the ITSO Railway requirements listed in 5.10, “Maintain the mobile devices” on page 97.

Performance

Performance is strongly influenced by the chosen device and the available network connection. The device type (operating system and processor performance) determines how quickly and efficient device management jobs can be executed. The speed of the network connection between device management agent and server affects the time needed for data transmission (e.g. time needed for software download).

Table 16-10 shows an example of demanded job execution times of a device management system under the assumption that the mobile device is cradled and connected to DMS server through a LAN connection.

Table 16-10 Target end-to-end job execution time

Target end-to-end execution time (device cradled and connected through LAN connection)		
Frequency of usage	Low performance device (smartphone)	High performance device (high end PDA)
Data complexity		
Inventory collection	5 minutes	3 minutes
Software package (~1MByte size)	5 minutes	3 minutes
Bootstrap (up to 10 MByte)	10 minutes	5 minutes

Availability

Availability is frequently an important Service Level Requirement. The table below lists an example specification of the desired availability. Availability requirements typically vary by use case and component. Each row represents a collection of components with common availability requirements. In our example all 3 listed components have the same availability requirements.

Table 16-11 Availability requirements example for ITSO Railway device management

	Availability requirement	**Impact of not met**
Component view		

	Availability requirement	Impact of not met
- Device management agent on handheld device (PDA) - Device management server - Network connection between agent and server	Fully available at the end of each shift of the delivery staff (7 days a week, minimum 6:00-7:00, 14:00-15:00, 22:00-23:00)	- Medium: Software inventory collection and minor software updates can not be performed - High: Configuration changes must be manually applied by mobile device user - Critical: Software updates and fixes necessary for mobile device users work

Maintainability

All application components (device management agent, server and network) must be independent of each other from a development and maintenance point of view. The following components can be designed, developed, tested and maintained independently of each other:

- Middleware components (device management agent and server)
- Network

Security

The device management system must prevent unauthorized access and job execution. The mobile device user must provide user name and password with which the device management agent can contact the server. On the DMS server side, jobs are especially assigned to users, user groups and/or device classes. Unauthenticated access can be prevented (only enrolled and enrolling devices are managed).

Manageability

Manageability mainly concentrates on the DMS system management. It contains the way the device management system (jobs, software packages, etc.) is managed from an administrator's point of view.

System constraints

System constraints are mainly defined by the used handheld device (performance), the available network connection and the size of software packages to be distributed.

A chosen PDA defines capabilities for its management. Specifications including CPU performance, memory size and operating system/execution environment all

affect this. The CPU performance effects the job execution speed on the device. The operating system and the execution environment of the device management agent (for example, OSGi) limit the remotely available configuration options for the device.

The available network speed determines the amount of time needed to download software packages and to upload inventory information.

The following constraints apply to an example of a Window Mobile 2003 PDA.

Table 16-12 Constraints defined by our chosen PDA

Constraint	Value
Processor	Intel XScale, 400 MHz
Memory	64 MB RAM, 32 MB flash ROM
Operating System (execution environment of device management agent)	Windows Mobile Edition 2003
Cradled and network connected	10 MBit/s Ethernet

System usability

The device management agent must be able to run without user interaction once the user name and password are entered (or be automatically triggered by other remote or local applications).

The device management system administration must provide graphical user interfaces to create device management jobs (these are wizard-like). There must also be options to track, log and trace device management activities.

Data integrity

The integrity of transmitted data between the device management agent and the server must be assured (for example, software packages, inventory information). The WebSphere Everyplace Access device management server and agent take care of this aspect.

Very important for mobile devices is an option to allow device management functions work over unreliable network connections. A function must be available, which allows an interrupted job to resume exactly there where it was stopped before.

We will look at these non-functional requirements and how they are fulfilled in the following chapters in more detail.

16.1.4 Solution approach

The ITSO Railway customer requirements above were the input for the example solution described in the following chapters (see 16.1.2, “Functional requirements and use case model” on page 396, as well as 16.1.3, “Non-functional requirements” on page 408). The following approach was taken to get to a device management solution for software distribution and inventory collection:

- Identification of the problem which has to be solved by considering the requirements and use cases to derive an architectural overview.
- Choice of suitable technology and Runtime pattern to create a component model for the solution.
- Identification of existing out-of-the-box solutions for mobile device management.
- Deployment of solution. Installation and configuration on the pervasive server as well as on the handheld device must be performed. This includes setup of user access to the device management system as well as creation of a software package and corresponding software distribution job for the mobile device.
- Maintenance and operation of the system.

The system deployment on the server side is described in the following chapters. The software package used for distribution is the application developed in Chapter 13, “Using Workplace Client Technology, Micro Edition” on page 283.

16.2 Architectural overview

Regarding the ITSO Railway customer requirements (see 16.1, “Overview” on page 394) the mobile device management solution must be able to provide software distribution, inventory collection, device and application configuration. The system administrator must be able to create software packages which can be distributed and remotely deployed to the mobile devices. He must also be able to force consistent device configurations for security and usage reasons (for example, enforce power-on password on device). It must be possible to limit device management jobs to a specific device class, user and/or user group.

The device management system must prevent unauthorized access to the system. It must also provide stable and secure data transmission to ensure data integrity (job resume capability in case of interrupts).

This input leads to the Pervasive runtime pattern for Device Management pattern, which was introduced earlier in this book (refer to 3.2.4, “Rich Device=Online::Runtime pattern” on page 43).

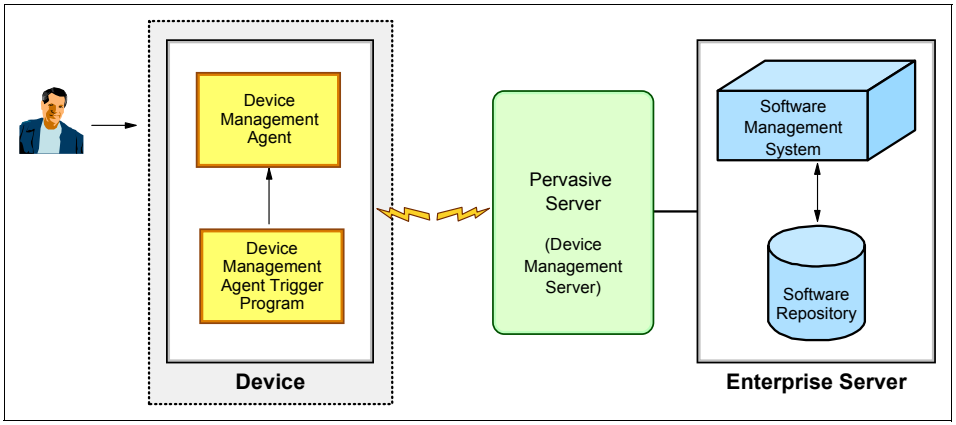


Figure 16-6 Architectural overview diagram for the ITSO Railway mobile device management solution

The high-level solution architecture in Figure 16-6 was derived from the Runtime pattern in Figure 3-4 on page 44. This device management system architecture contains two main parts (2-tier approach). The enterprise server including the Software Management system is optional and can be integrated if it exists (3-tier approach):

- Part 1: Device

  Part 1 instantiates the client including the pervasive client services node of the applied Runtime pattern in Figure 3-5 on page 45. The Device is a mobile computing device (for example, PDA) which runs the appropriate business applications (for example, e-mail, calendar, ticketing system application, etc.). It contains two important functions, the device management agent (DM Agent) and the device management agent trigger. The DM Agent is middleware counterpart to the pervasive server (device management server) to perform the device management jobs. Since the mobile device is only intermittently connected and available, the server can only be contacted by the DM Agent for job queries when a connection is available. The DM Agent Trigger is an optional component which could detect a connection or even establish it and can automatically trigger the DM Agent to contact the DM server in order to ask for available jobs. An example would be a smartphone, where a server side extension could contact the DM Agent Trigger through a SMS to trigger the DM Agent for a server query.

- Part 2: Pervasive server

The pervasive server instantiates the directory and security services, personalization server, presentation server and application server nodes including the pervasive extension services node of the Runtime pattern in Figure 3-5 on page 45. In the device management case, the Pervasive extension services node would be the device management server plug-in which extents the existing infrastructure with mobile device management capabilities.

It can optionally use existing Software Management Systems for license tracking, software repository, etc.

- Part 3: Enterprise server

  The enterprise server stands for existing software management systems in an enterprise (license tracking, software repository). It is not necessary to have this component to meet the ITSO Railway requirements. The pervasive server can fulfill all the mobile device management tasks without the existence of the enterprise server. The enterprise server is listed here to underline the capability of the pervasive server to utilize existing systems.

16.3 System design overview

In this section, some general terms and function in the field of device management are explained using WebSphere Everyplace Device Manger (WEDM) as example. Then an architectural overview for the ITSO Railway device management example is developed using the WEDM server in WebSphere Everyplace Access.

16.3.1 General device management considerations

In this chapter general device management terms and functions are explained using the example of IBM WebSphere Everyplace Device Manager (WEDM).

In order to have a common understanding of often used terms in the field of device management, the terms *Device Manger*, *Device*, and *Job* are shortly explained.

Device management terms

- *Devices* are personal digital assistants (PDAs), handheld PCs, PCs, subnotebooks, cellular phones, set-top boxes, in-vehicle information systems, and other devices for pervasive computing.
- *Jobs* are device management actions carried out by the Device Manager. Jobs may be applied to (or targeted to) a single device, a single user, a group of users or a group of devices with common characteristics (for example,

same operating system). These common characteristics can be the device owner or owner group or by some attributes of the device inventory, or a combination of that.

An assigned job is run when the device connects to the Device Manager server. The server maintains a job status history for all devices. Certain device types allow the server to notify the device about an existing job.

- *Software* is a registered software package or OSGi (Open Service Gateway Initiative) bundle. A software package is the set of meta package properties, application properties and application package files for all the software to be distributed to a device. An OSGi bundle contains all the Java interfaces and classes, a manifest file and resource files packaged into a JAR file.

The Device Manager database stores URLs for locating software packages and OSGi bundles and stores other software information. The Device Manager console provides windows for manipulating that information.

- *Device Manager* is a powerful application that helps enterprises and service providers to manage mobile devices efficiently. Device Manager is used to track devices and their configuration status in a database. With this information, Device Manger can perform management tasks depending on the device capabilities. Examples are device and application configuration, inventory collection, software distribution and initial provisioning. Device Manager can use existing user repositories or its own subscription manager component to control access to provided user interfaces for administrators and device users.

  Parts of WEDM are imbedded in several IBM products, for example in WebSphere Everyplace Access.

Introduction to WebSphere Everyplace Device Manager

WEDM is built on a Web application server model. The WEDM server is a set of J2EE servlets, running on WebSphere Application Server. Device management data storage is in a relational database, such as DB2 or Oracle. The Device Manager server requires an agent on the managed device (device management agent). The agent can be a Device Manager supplied agent, such as Pocket PC, Windows 32-bit, and PalmOS, or can be supplied by a device manufacturer.

Device agents communicate with the WEDM server using HTTP or HTTPS. The protocol running on top of HTTP is either a proprietary protocol, which is used with Pocket PC and PalmOS devices, or a standard protocol, such as the OMA DM protocol used with OSGi-compliant devices. With HTTP and HTTPS communications between devices and WEDM, requests and responses can pass through various network elements, such as firewalls.

OMA DM is a specification created by the Open Mobile Alliance (OMA) organization for device management of wireless devices. It is a standardization that allows Device Manager to write one protocol engine to encode and decode the messages passed between Device Manager and the OMA DM device agent. WEDM provides an OMA DM Management Server that is an OMA DM 1.1.2 implementation. Available management functions for SyncML DM devices include inventory collection, device configuration, and running custom built SyncML DM command jobs.

WEDM also supports software bundle management for OSGi enabled devices. OSGi is a specification that defines a run-time environment in which independently managed applications co-exist in a single virtual machine. Operations that can be performed on OSGi bundles include installing, starting, stopping, updating, and removing bundles.

WEDM is particularly suited for wireless or wired providers who want to use open standards to support their business-to-consumer subscription service. Device Manager uses the device management open standards from Open Mobile Alliance to help providers solve service-related problems and reduce the total cost of customer support.

WEDM extends the customer service and distribution capabilities of the provider's administrators, and supplies APIs for integrating value-added applications that help improve the operating environment for delivering service.

Furthermore, WEDM defines a plug-in architecture for the Device Manager server. This allows specialized support for different kinds of devices (for example, PalmOS PDA or subnotebooks running Pocket PC). Any number of device types could be supported through custom developed plug-ins.

Administrators can use graphical user interface called the Device Manager console or especially developed portlets for the WebSphere Portal Server as provided by WebSphere Everyplace Access. From this console, an administrators can manage:

- Jobs
- Device classes
- Devices
- Software

WebSphere Everyplace Device Manager components

Figure 16-7 on page 417 shows components the WebSphere Everyplace Device Manger. The core Device Management Server components are applications and user interfaces through which the capabilities of the server may be exercised, and the information stored within its database may be accessed.

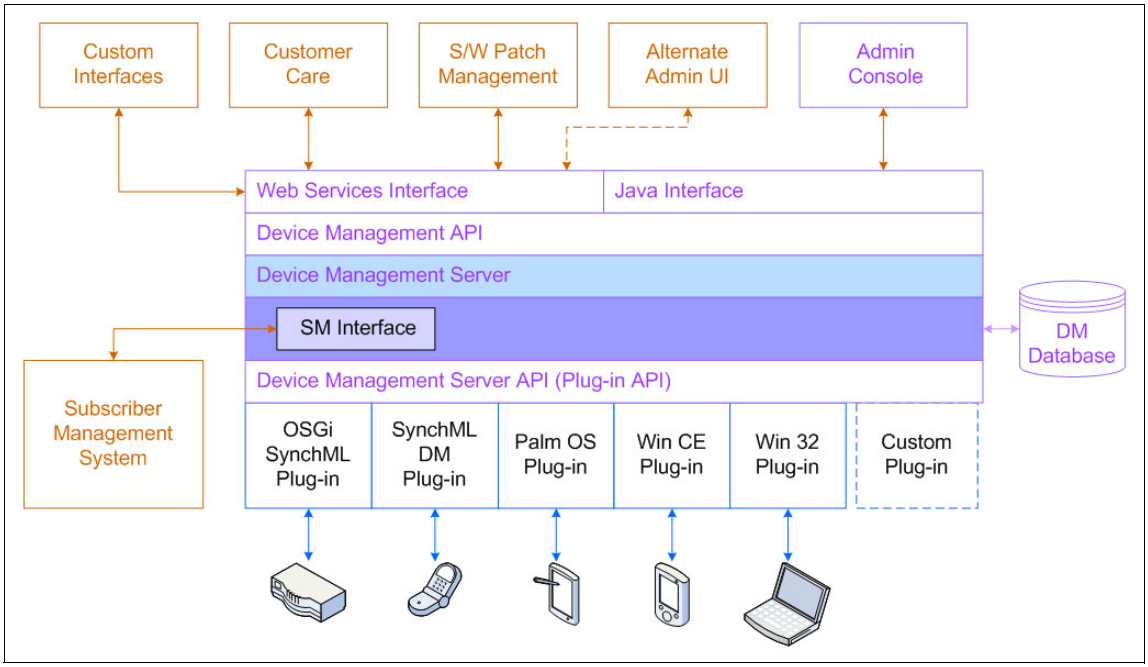


Figure 16-7 WebSphere Everyplace device management architecture

Below the server lies a series of device plug-ins, each of which provides the specific function needed to handle the class of device it is designed for.

To the left and right sides of the server are, respectively, a subscriber management system to which the Device Manager issues requests for authentication and authorization for actions which it is about to execute, and the device management database.

External applications such as subscriber management, customer care, etc. are shown in a different color; these are not included with WEDM. API information, including Web Services Description Language (WSDL) and JavaDoc is provided to support the development of applications that interface to Device Manager. WebSphere Everyplace Access has parts of WEDM integrated so that an out-of-the-box solution is available.

Please, refer to WebSphere Everyplace InfoCenter and the WebSphere Everyplace Device Manger InfoCenter for further information.

16.3.2 ITSO Railway device management solution outline

The following solution outline was developed under the consideration of the ITSO Railway customer requirements for a mobile device management solution, the Pervasive runtime pattern for device management and the device management capabilities of WebSphere Everyplace access.

16.4 Deployment and runtime configuration

This chapter explains the deployment and configuration of the ITSO Railway example for device management software distribution job using WebSphere Everyplace Access.

16.4.1 Overview

There are two options for creating software distribution jobs with WEA version 5.0:

- Using the device management portlet
- Using the device management Console

Since the first option is available through the WebSphere Everyplace Access administration user interface (portal page), the following steps taken for the ITSO Railway example are explained using the device management portlet.

These steps are necessary:

1. Create the software package for distribution.

   An archive (zipped) file that contains the software for distribution and specific job instructions must be created first.

   Then it can be uploaded to the device management portlet on the WebSphere Everyplace Access server.

2. Upload the software package to the device management portlet on the WebSphere Everyplace Access server and assign target users, user groups or devices.

Job processing begins automatically after step 2 to all authenticated users.

16.4.2 Creation of the software package

The first step to create a software distribution job is the creation of the software package which contains the software and specific installation and configuration instruction commands.

For this, an archive (zipped) file is composed of a *WSEPackage.xml file* (with the job instructions), package and meta package definition files, and software packages for distribution. The software, such as a CAB file for WinCE class devices, or a PRC or PDB file for Palm OS class devices, is ready for installation on the device according to the instructions in the *WSEPackage.xml* file.

The following steps outlines the process for creating software packages for the WEA device management portlet:

1. Gather the software file or files to be distributed, and store them in a temporary folder.
2. Create one or more package definition files.
3. Create a meta package definition file.
4. Create a *WSEPackage.xml* file.
5. Create an archive (zipped) file that contains the following files: *WSEPackage.xml*; meta package definition file; package definition file; and the software to be installed, such as CAB (for Pocket PC) or PRC (for Palm OS) files.

Gathering software files for distribution

For the ITSO Railway software distribution example use cases from 16.1.2, "Functional requirements and use case model" on page 396, the *remotedisp*[1] software must be distributed to the supported devices (PocketPC 2002). First, the files for the *remotedisp* application are stored the files in a temporary directory such as *C:\temp\remotedisp* (Figure 16-8).

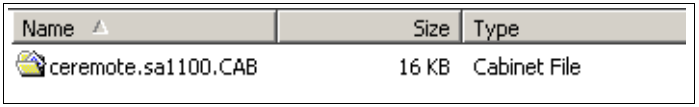


Figure 16-8 Files for Remote Display for Pocket PC 2002 application from Microsoft Corporation

Creating the package definition files

Next the package definition and meta package definition files must be created.

The Package definition file:

- Is a plain text file that identifies the software and its properties in the software package. The software package will be distributed or removed from the supported device after the archive (zipped) file is uploaded to DMS. There must be one package definition file for each software package.

[1] License free tool from Microsoft Corporation

- Points to all the components of the software, such as graphics files, database files, compiled files.
- Defines the properties for distributing or removing the software package, properties such as running a setup file after the files are downloaded; allocating disk space; restarting the device before removing software.

Please refer to the WebSphere Everyplace Access InfoCenter for more details. The package definition file *remotedisp.pkg* created for the *remotedisp* application is listed in Example 16-1.

Example 16-1 Package definition file remotedisp.pkg for the ITSO Railway software distribution job example

```
#*DFP-v1.00 DMS Filepack (version v1.00)
#*Keyword section
after_prog_path=\windows\wceload.exe ceremote.sa1100.CAB
need_space=16000
%
#*Files and directories
ceremote.sa1100.CAB d=/Program Files/rdisp
%
#*Nested file packages
%
#*Excluded files
%
#*Extra section options
```

Explanation of Example 16-1:

- `#*DFP-v1.00 DMS Filepack (version v1.00)` is the required header line.
- `after_prog_path` specifies the full path of an executable file to be run after the software packages are distributed. wceload.exe installs our remote display application (packaged in the ceremote.sa1100.CAB file).
- `need_space` is the size which the file needs on the target.
- `ceremote.sa1100.CAB d=/Program Files/rdisp` contains the names of the files and directories to be distributed, created, or removed. Each file entry or directory is listed on a separate line. For software distribution and software removal jobs, the listed files and directories are sent to the device. If a directory does not exist on the device, it is created.
- `[setProperty]` specifies each job parameter or Windows registry keyword. For each line, use the format: `jobParameter=value` or `registry Keyword=value` for each registry keyword, the location where the change will be made.

Creating the meta package definition file

A meta package definition file lists each software package by its package definition file name (see *remotedisp.pkg* created before). Individual packages can be created for software distribution jobs or software removal jobs. The meta package properties include the name of the software package, a version number for the software package, user selection options for the software package, and other properties.

A meta package definition file defines:

- Properties that affect the entire software package (meta package properties)
- Properties for each software package listed in the meta package definition file (application properties)

The meta package definition file is created as a plain text file, prefacing the package name with *meta_* by convention. Meta package properties are always listed first, followed by the stanzas listing application properties, in the *meta_<productname>.pkg* file.

Example 16-2 Meta package definition file meta_remotedisp.pkg for ITSO Railway software distribution job example

```
PackageUserSelection=yes/delay/reject
PackageName=RemoteDisplay
PackageVersion=1.0
PackageDescription=Install remote display client on PPC2002

[Application1]
ApplicationUrl=remotedisp.pkg
ApplicationSelectionDisable=yes
ApplicationName=WindowsCEAgent
ApplicationVersion=2.03
ApplicationDescription=Remote Display Windows CE
```

Explanation of Example 16-2:

- `PackageUserSelection` enables user selection or automatic downloading of the software distribution job. Options: `yes`, `yes/delay`, `yes/reject`, `yes/delay/reject` or `no`. A user can select to download all the software packages now (`yes`), delay the choice to download the software packages (`delay`), reject the download of this software distribution job (`reject`), or have no choice, so that all of the packages in the software package are downloaded (`no`). If delay or reject are not included, the corresponding buttons are inactive in the User Selection window of the supported device. We gave the ITSO mobile device user multiple options Example 16-2 for demonstration purposes.
- `PackageNamespecifies` the package name.

- `PackageVersion` specifies the package version.
- `PackageDescription` provides a description of the package.
- `[Application1]`
- `ApplicationUrl=remotedisp.pkg` specifies the name of our previously created package definition file that is included in the.zip file, not a fully-qualified URL.
- `ApplicationSelectionDisable` indicates whether or not a user can highlight that software package in a User Selection window or if it is automatically highlighted. The value `yes` in Example 16-2 on page 421 highlights the software package automatically.
- `ApplicationName` specifies the application name.
- `ApplicationVersion` specifies the application version.
- `ApplicationDescription` describes the application.

Note: Our meta package definition file *meta_remotedisp.pkg* example contains only one software package. You can include more than one software package in a software distribution job.

Creating the WSEPackage.xml file

After the creation of the software package, the package definition file, and the meta package file, the *WSEPackage.xml* must be build. This file must be named *WSEPackage.xml* and must be located at the top level of the archive structure. The device management portlet uses this file to create one or more jobs that specify how to distribute the software package in the archive file after it is uploaded to Device Manager.

Figure 16-9 on page 423 and Example 16-3 on page 423 show the ITSO Railway sample *WSEPackage.xml* file for the *remotedisp* application that directs Device Manager to install the *remotedisp* software package to the chosen PPC 2002 device.

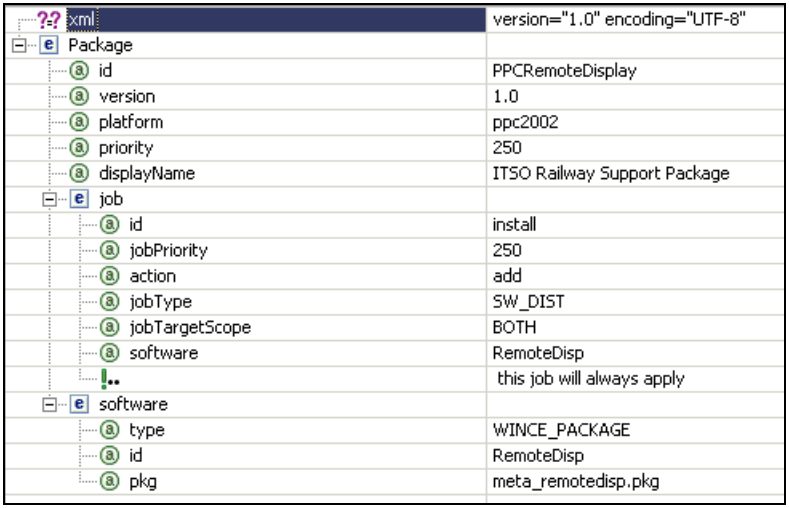


Figure 16-9 WSEPackage.xml file for the ITSO Railway software distribution job example (XML editor view)

Note the following keys in the *WSEPackage.xml* code:

- *Package* identifies the software product release and specifies its properties: release version, platform, priority, locale, class name, relative path and file name to the class directory, and display name.
- *job* assigns an action, scope, priority, and name to the software package.
- *software* specifies the file name of the software package's package definition file. The software attribute must match one of the software elements in this WSEPackage.xml file.

Example 16-3 WSEPackage.xml file for ITSO Railway software distribution job example

```
<?xml version="1.0" encoding="UTF-8"?>
<Package id="PPCRemoteDisplay"
      version="1.0"
      platform="ppc2002"
      priority="250"
      displayName="ITSO Railway Support Package">
   <job id="install"
      jobPriority="250"
      action="add"
      jobType="SW_DIST"
      jobTargetScope="BOTH"
      software="RemoteDisp" >
   <!-- this job will always apply-->
   </job>
```

```
<software type="WINCE_PACKAGE" id="RemoteDisp" pkg="meta_remotedisp.pkg"/>
</Package>
```

Explanation of the WSEPackage.xml file:

- Package
  - `PackageId`= The unique identifier for the package. Only one ID can be assigned per group. This is the name of the top level directory under which all the files will be stored (required).
  - `version`= The software package release.
  - `platform`= Must match a platform defined in the DMS portlets.
  - `priority`= A numeric value between 1 and 999, which can be overridden by the priority field in the <job> tag.
- Job
  - `id`= The unique identifier for the job (required).
  - `jobPriority`= Use only to override the default priority of the package.
  - `action=add` or `remove`. Default is *add*.
  - `jobType=SW_DIST` or `SW_REMOVAL`. Default is *SW_DIST*.
  - `jobTargetScope=new`, `existing`, or `both`. Default is *both*.
  - `software=RemoteDisp`, where RemoteDisp is the matching ID attribute of a software entry within the XML file.
  - `queryExpression condition`= Default is *AND*.
  - `queryClause attribute`= see WEA documentation referenced below.
- Software
  - `type`= A pre-determined type.
  - `id=RemoteDisp` - The name of the software package previously defined under `job` in this file.
  - `pkg=meta_remotedisp.pkg` - The name of the meta definition package previously created.

Find other keywords and options (for example, for conditional installation) in the WebSphere Everyplace Access InfoCenter.

Creating the archive file remotedisp.zip

Now, the archive file containing the device management job package is built by compressing all the previously created files into a zip archive called *remotedisp.zip* (including the software package file, package definition file, meta package definition file and *WSEPackage.xml* file; see Figure 16-10 on page 425).

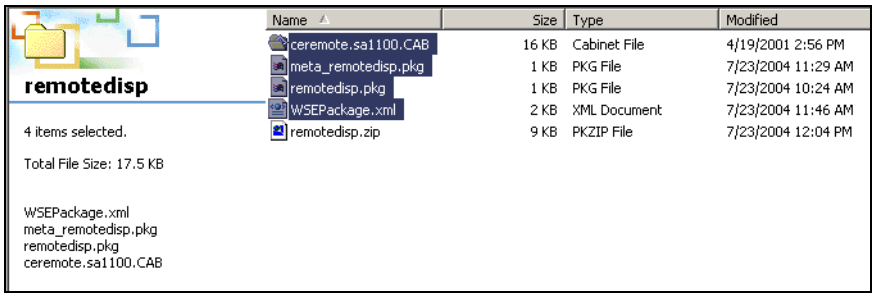


Figure 16-10 Files to be compressed into remotedisp.zip file for ITSO Railway software distribution job example

Now the software package is ready for distribution. A software distribution job must be created and assigned to the target users, user groups or devices.

16.4.3 Creation and assignment of the software distribution job

With the device management portlet, you can distribute WebSphere Everyplace Client updates and custom software packages to supported devices quickly and efficiently. The device management portlet helps you distribute Everyplace Client updates (fix packs and interim releases) to supported devices in your enterprise environment. The device management portlet also enables you to distribute custom software packages or remove existing software.

The last steps in our ITSO Railway software distribution example (Remote Display) are:

1. The upload of the software package *remotedisp.zip* (archive) created in 16.4.2, “Creation of the software package” on page 418.
2. The assignment of the automatically created job to our target user group.

Regarding to the ITSO Railway example, we decided to assign the software package to the offlineformsusers group (contains the ITSO Railway delivery stuff with the new PDA, see 11.5.1, “Configuration for offline forms-based applications” on page 253).

The device management portlet is used for uploading the *remotedisp.zip* software package. The zipped file is stored in a directory named uniquely to avoid naming conflicts.

Note that if it is attempted to upload an archive with a duplicate package ID, the upload will fail. The previous version of the package must be removed in order to upload a replacement for it.

The following steps are necessary in the WebSphere Everyplace administration portal:

- Log in to the WebSphere Everyplace Access portal using an administrator ID.
- Select the Administration page group, find the WebSphere Everyplace section, expand **Device Management** and click **Software Packages** (see Figure 16-11).

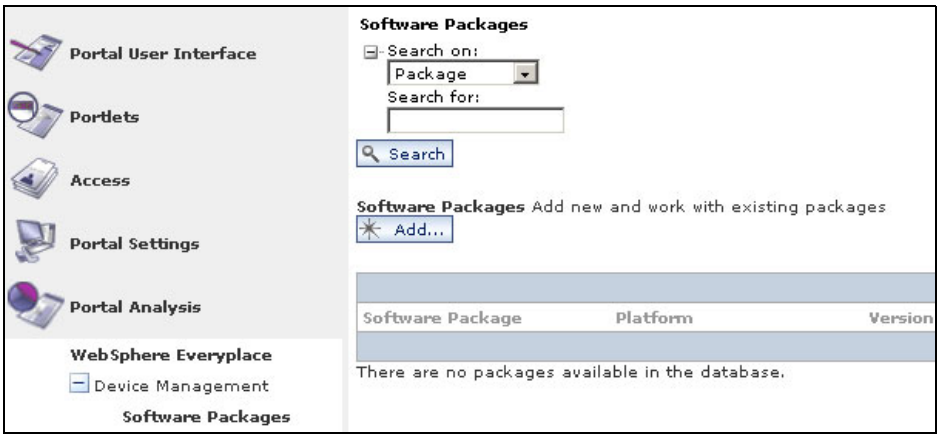


Figure 16-11 Installation of software package using WebSphere Everyplace administration potal

- Click **Add** and specify the file location of the package (*remotedisp.zip*).
- Click **OK** to add the new *remotedisp* package. The Software packages view should display the newly added package.
- Assign the software package to users and/or user groups (see Figure 16-12 on page 427):
  - Click the Manage Recipients icon next to the newly-added package.
  - If you want to assign this package to all users, click **Assign** to all users, then click **OK**. The package is now assigned to all users.
  - Click **Assign to specific user groups**, then click **OK**.
  - Select the **offlineformsusers** group and click **OK**. The Remote Display software package is now assigned to the ITSO Railway delivery stuff users group which was equipped with a new PPC 2002 PDA.Click **OK** again.

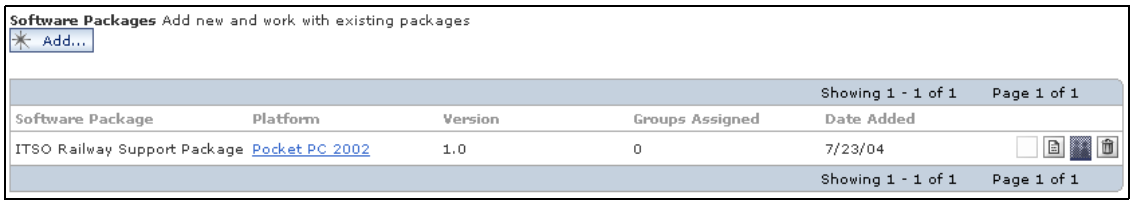


Figure 16-12 Assigning target users and/or user group to software package

After the package is assigned, the software distribution job with id=install and action=add from the WSEPackage.xml file (see Figure 16-9 on page 423) is automatically created and submitted to the selected offlineformsusers group.

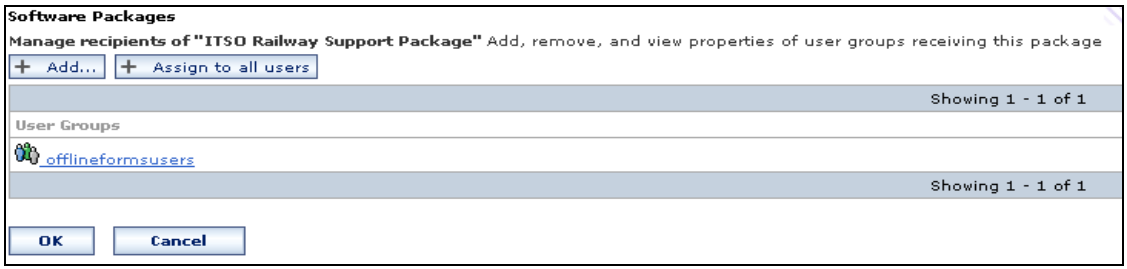


Figure 16-13 offlineformsusers group is assigned to the device management job

16.4.4 Running the software distribution job

To complete this process, users must initiate the software distribution job on the PDA. For this, the ITSO Railway delivery people start the device management agent through the WebSphere Everyplace Client interface which they use for the synchronization of their *Supply Consumption Record* as well (see 11.5.2, “Using the application” on page 258). They log on to the WEA client (see Figure 16-14 on page 428) and perform the following steps:

1. Start Everyplace Client on the Pocket PC 2002 device: Click **Start**, then click **Everyplace Client**.
2. Log in to Everyplace Client.
3. Navigate to the page that has the device management icon.
4. Tap the device management icon and click **Get Updates** to begin the install process on the device (see Figure 16-14 on page 428).

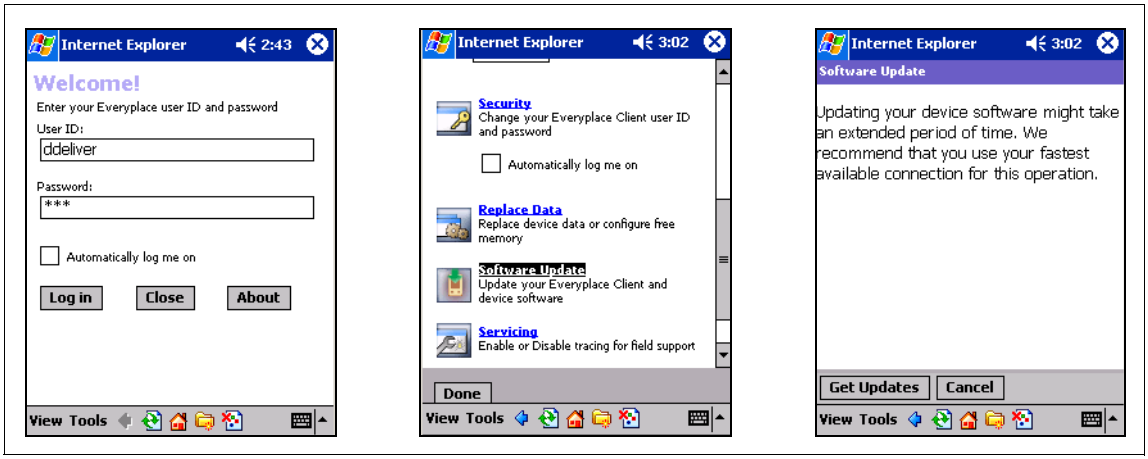


Figure 16-14 Start device management job on WebSphere Everyplace Client for PPC 2002

5. The device management agent is queries the server for the jobs (see Figure 16-15). Since we created the ITSO Railway Support job for the installation of the remotedisp software package, the job is picked up and executed. The user will have the choice whether the installation of the *RemoteDisplay* job should be carried out immediately, be deferred or be rejected (see Figure 16-15). Users should see that the job is processed (scheduled, invoked, software downloaded, files from the package extracted and installed, Figure 16-15). The installation is complete when the title bar on the screen no longer displays IBM Device Agent.

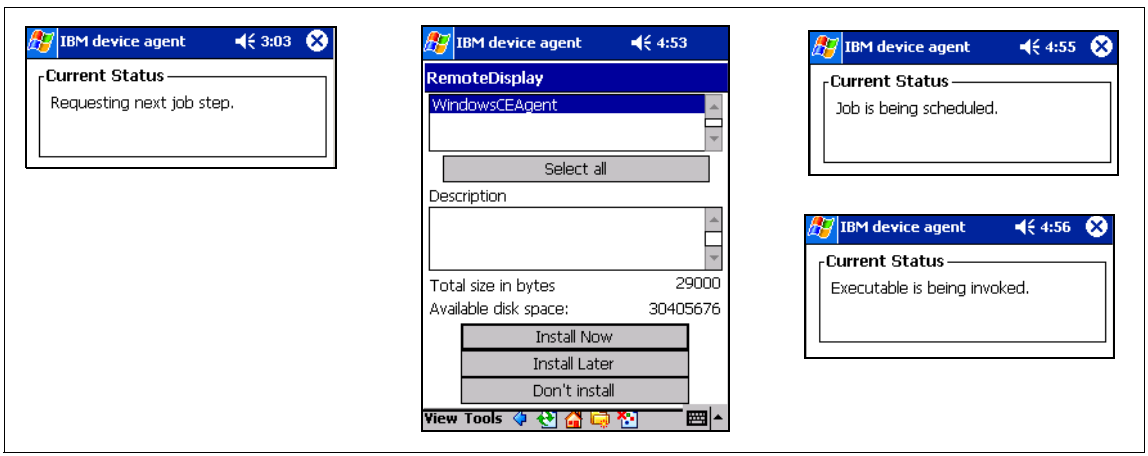


Figure 16-15 Software distribution job progress on the PPC2002 device

Note that the browser screen displaying the device management icon may close once installation is complete. This is normal. Also, it is recommended that users wait a few seconds after the install completes before using the newly installed applications.

Monitor job progress

The administrator can use the Device Manager console to monitor device management and Device Manager related tasks. This console is a separate user interface especially designed for device management. Figure 16-16 lists scheduled jobs in the device management console. Our ITSO Railway software distribution job is also included.

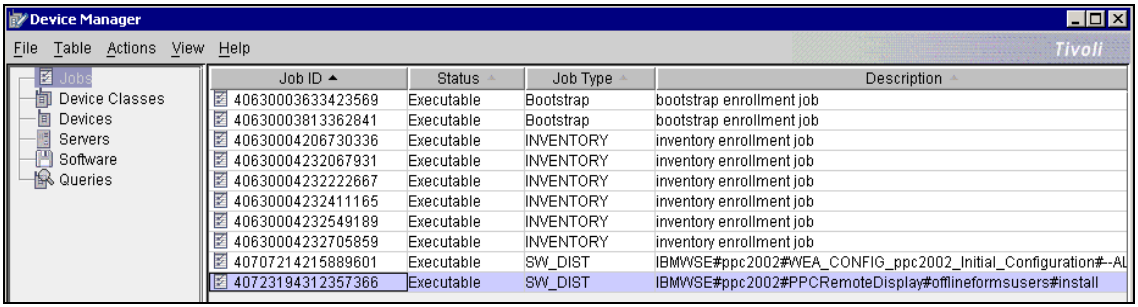


Figure 16-16 Device management console: Assigned job list

Job progress information can be gathered through right-clicking the highlighted job and choosing View Job Progress Summary. Figure 16-17 on page 430 shows that the ITSO Railway software distribution job was scheduled and one enrolled target device was identified.

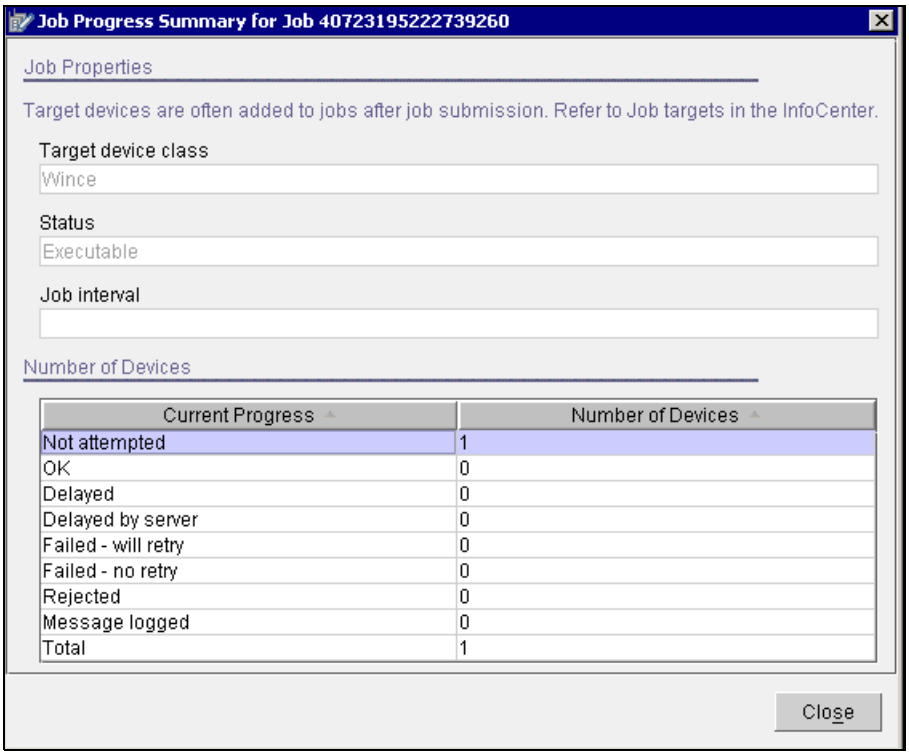


Figure 16-17 Device management console: Job progress summary, job assigned

After the ITSO Railway Support job was carried out on the device, the job status is updated and can be checked using the device management console as described above. Figure 16-18 on page 431 returns a successful completion of the *remotedisp* software distribution job.

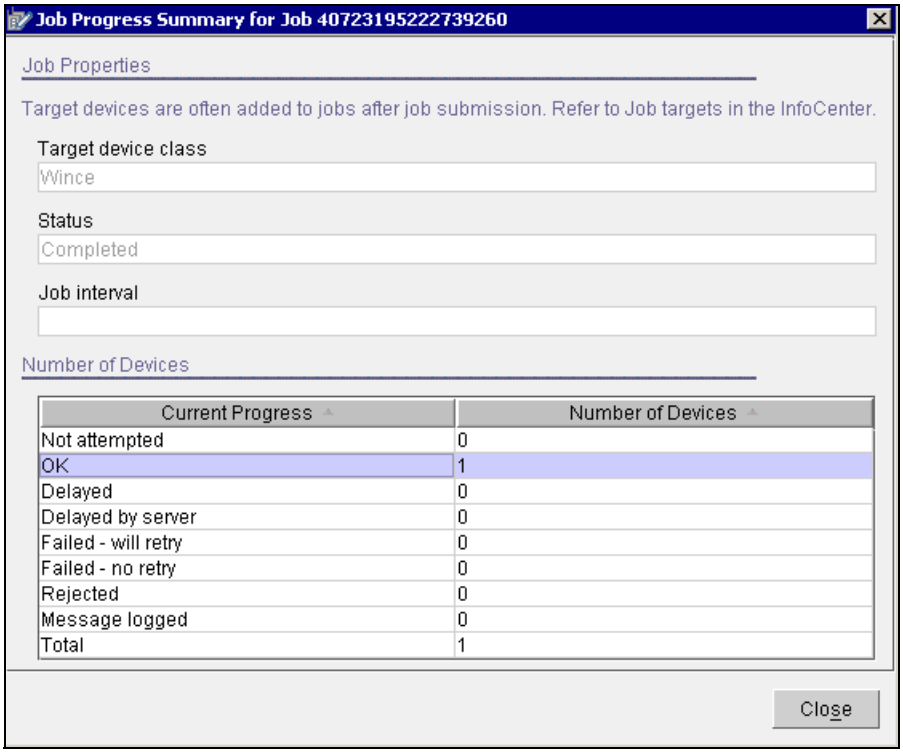


Figure 16-18 Device management console: Job progress summary, job successfully carried out

Now the remote display tool for the ITSO Railway delivery people is installed on their PDA. When the PDA is cradled and network connected, the delivery person can get support directly on the PDAs screen when the remote display application is started.

Part 4

Appendixes

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/SG246315

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, SG246315.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
SG246315.zip	Zipped code samples

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	20 MB minimum
Operating System:	Windows 2000 Server with the latest updates
Processor:	1.5G Hz or higher
Memory:	Intel Pentium 4

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 439. Note that some of the documents referenced here may be available in softcopy only.

- *Mobile Applications with IBM WebSphere Everyplace Access Design and Development*, SG24-6259
- *Patterns: Pervasive Portals Patterns for e-business Series*, SG24-6876
- *IBM WebSphere Portal V5: A Guide for Portlet Application Development*, SG24-6076
- *WebSphere Everyplace Access Version 4.3 Handbook for Developers*, SG24-7015
- *A Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management*, SG24-5309

Additional publications

- *Patterns for e-business: A Strategy for Reuse* by Jonathan Adams, Srinivas Koushik, Guru Vasudeva, and George Galambos

Online resources

These Web sites and URLs are also relevant as further information sources:

- Patterns for e-business Web site

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

- WebSphere Application Server Web site

  `http://www.ibm.com/software/webservers/appserv/enterprise`

- Open Mobile Alliance Web site

  http://www.openmobilealliance.org
- Wap forum Web site

  http://www.wapforum.org
- XForms Web site

  http://www.w3.org/TR/xforms
- Microsoft Table PC Web site

  http://www.microsoft.com/windowsxp/tabletpc
- Microsoft mobile application development tools Web site

  http://msdn.microsoft.com/mobility/prodtechinfo/devtools/netcf
- cHTML Web site

  http://www.w3.org/TR/1998/NOTE-compactHTML-19980209
- W3's validator Web site

  http://validator.w3.org
- VoiceXML Web site

  http://www.voicexml.org
- Microsoft mobile solutions Web site

  http://www.microsoft.com/mobile
- OSGI Web site

  http://www.osgi.org
- Palm source Web site

  http://www.palmsource.com
- Symbian Web site

  http://www.symbian.com
- Sun J2ME Web site

  http://java.sun.com/j2me
- Qualcom BREW Web site

  http://brew.qualcomm.com
- Sun Java JMS Web site

  http://java.sun.com/jms
- IEEE 802.x Web site

  http://www.ieee802.org

- Bluetooth Web site

  http://www.bluetooth.org
- developerWorks article

  http://www-128.ibm.com/developerworks/subscription/descfiles/vtws51wx.h
- WebSphere Studio Device Developer Web site

  http://www.ibm.com/software/pervasive/products/wsdd/index.shtml
- Everyplace Toolkit Web site

  http://www-306.ibm.com/software/pervasive/everyplace_toolkit
- Microsoft Developer Netowrk Windows Mobile Web site

  http://msdn.microsoft.com/windowsmobile
- W3 schools WAP Web site

  http://www.w3schools.com/wap
- Palm One Web site

  http://www.palmone.com
- WebSphere MQ Everyplace Web site

  http://www.ibm.com/software/wmqe

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Patterns: Pervasive and Rich Device Access Solutions

Patterns: Pervasive and Rich Device Access Solutions

Using Patterns for e-business to design pervasive solutions

WebSphere Everyplace Access and related products

Sample applications for various scenarios

Patterns for e-business are a group of proven, reusable assets that can be used to increase the speed of developing and deploying e-business applications. This IBM Redbook focuses on the architecture and implementation of pervasive solutions using currently available IBM products.

Part 1, “Pervasive solution patterns” on page 1, uses the Patterns for e-business to describe the Application and Runtime patterns applicable to pervasive solutions. IBM product mappings are applied to these patterns.

Part 2, “Guidelines” on page 111, describes the technologies key to building pervasive solutions. It also describes the IBM development tools for building these solutions.

Part 3, “Scenario implementations” on page 149, consists of eight pervasive scenarios. Each scenario implements an IBM product mapping, and provides design, development, and runtime guidelines for building these scenarios.

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

SG24-6315-00 ISBN 0738492175