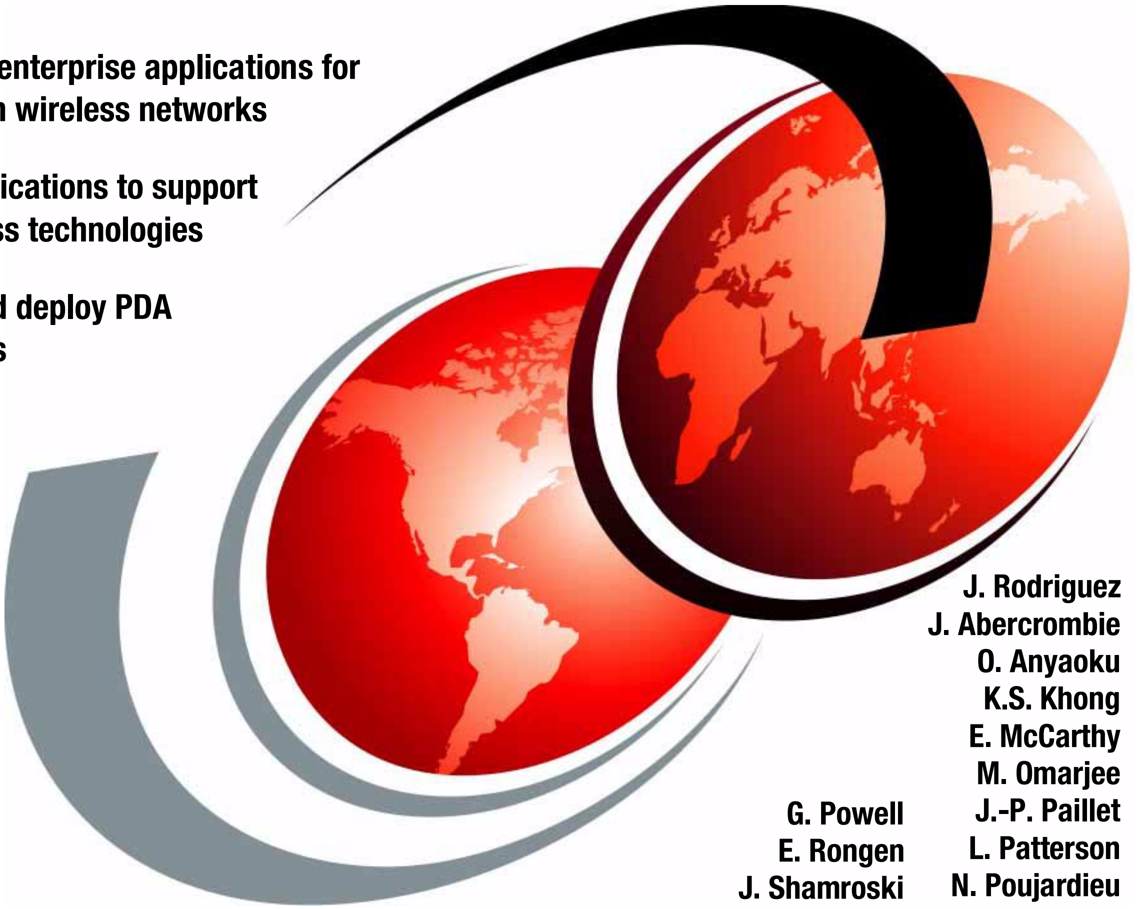
IBM

IBM WebSphere Everyplace Server Service Provider and Enable Offerings: Enterprise Wireless Applications

Adapt your enterprise applications for access from wireless networks

Extend applications to support new wireless technologies

Develop and deploy PDA applications



J. Rodriguez
J. Abercrombie
O. Anyaoku
K.S. Khong
E. McCarthy
M. Omarjee
J.-P. Paillet
L. Patterson
N. Poujardieu
G. Powell
E. Rongen
J. Shamroski

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM WebSphere Everyplace Server Service Provider and Enable Offerings: Enterprise Wireless Applications

February 2002

SG24-6519-00

Take Note! Before using this information and the product it supports, be sure to read the general information in "Special notices" on page 795.

First Edition (February 2002)

This edition applies to Version 2 Release 1 of IBM WebSphere Everyplace Server Service Provider Offering and Version 1 Release 1 of IBM WebSphere Server Enable Offering for multiplatforms.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Contents

Preface

This redbook can help you to adapt and extend new and existing enterprise applications so that they can be accessed from wireless devices, such as WAP phones and PDAs, using the IBM WebSphere Everyplace Server (WES) Service Provider Offering (SPO) and Enable Offering (EO). The information provided in this redbook targets Business-to-Employee (B2E) enterprise applications, but most of the scenarios presented apply to Business-to-Consumer (B2C) applications as well.

In this redbook, you will find step-by-step examples and scenarios showing ways to rapidly integrate your enterprise applications into an IBM WebSphere Everyplace Server environment and therefore also make them available from wireless devices by implementing new and enhanced capabilities incorporated in the current releases of IBM WebSphere Everyplace Server offerings, such as transcoding, annotators for text clipping, stylesheets and the wireless gateway.

Once your enterprise applications are available from wireless devices, you will be ready to deploy new state-of-the-art technologies, such as Push messages, Location-Based Services, Intelligent Notifications Services and Voice applications. You will find numerous scenarios describing recommended ways to develop applications using the APIs provided by the IBM WebSphere Everyplace Server components to support these services.

Although IBM WebSphere Everyplace Server offerings do not provide a voice server, we have also included guidelines to developing Voice XML applications using transcoding capabilities provided by IBM WebSphere Everyplace Server. This redbook includes scenarios using IBM Mobile Connect and Synchronization Manager with a sample DB2 Everyplace application built with the Mobile Application Builder. Transaction messaging applications are also described, using MQSeries Everyplace, a component of IBM WebSphere Everyplace Server, to provide a once-only assured delivery of messages.

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

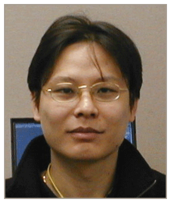
The team that wrote this redbook

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

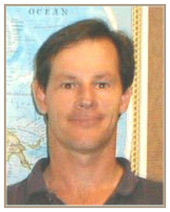
J. Rodriguez is a Consulting IT professional and Project Leader at the IBM ITSO Center, Raleigh. He received his M.S. degree in Computer Science from Iowa State University. He writes extensively and teaches IBM classes worldwide on such topics as networking, Web technologies, and data security. Before joining the IBM ITSO, he worked at the IBM laboratory in the Research Triangle Park (North Carolina, USA) as a designer and developer of networking products.

J. Abercrombie has 28 years of experience in the IT industry, 13 of which have been with IBM. Over this period of time, he has specialized in RS/6000, Networking and Pervasive Computing (Mobile e-Business). He is currently part of the Hursley International Solution Centre (ISC) for Pervasive Computing and is based in Edinburgh, Scotland.

O. Anyaoku is an Application Developer working with Enterprise Java Technologies at Lekki Technologies, based in New York. He has worked specifically for two years building and supporting applications developed on the IBM WebSphere platform. He holds a degree in Computer Systems Engineering from the University of Massachusetts, Amherst.



K.S. Khong is an IT Specialist in IBM Emerging Technology Centre, Singapore. He has five years of experience in banking and financial application development. His areas of expertise include C++, Java, J2EE, XML and object-oriented design patterns. He holds a B.Eng degree in Electrical Engineering and a Master of Technology degree in Software Engineering from the National University of Singapore.

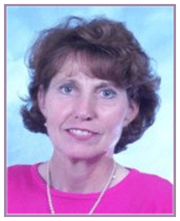


E. McCarthy is a Middleware Support Specialist with IBM GSA in Australia. He has over ten years of experience in the Information Technology field. He holds a B.S. degree from the University of Queensland. His areas of expertise include CICS and MQSeries.

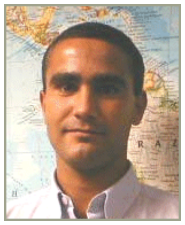
M. Omarjee is an IT specialist with IBM Business Innovation Services in Johannesburg, South Africa. He started with IBM as an applications software developer in e-business and Web-oriented solutions. His current area of expertise is centered around Web technologies such as Java, markup languages, and related object-oriented technologies. He holds a National Diploma in Information Technology from the Technikon Witwatersrand of South Africa.

J.-P. Paillet is a Consulting IT Architect in the London Solutions Practice, part of IBM UK Software Services. He has 30 years of experience in various aspects of computing. His most recent work has been consulting for projects using IBM WebSphere products, particularly design and risk management. His areas of expertise include methodology, object technology and linguistic systems.

L. Patterson is an IT Consultant within the IBM iSeries Custom Technology Center (CTC), IBM Rochester, Minnesota. She is currently with the WebSphere Everyplace Server Solution Enablement Team. Her expertise includes Java, Enterprise JavaBeans, XML and related technologies, and development education packages on Java, XML and the San Francisco product.

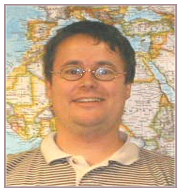


N. Poujardieu is with the IBM EMEA Mobile e-Business Advanced Technical Support team in France. He holds an M.S. degree in Pure and Analytic Mathematics from the French Scientific University of Paris. In his current job, he demonstrates the performance of IBM WebSphere Everyplace Server components. He has experience in projects involving the Wireless Gateway running over GPRS.



G. Powell is an IBM Senior I/T Specialist for IBM Global Services (USA),where he implements custom e-commerce solutions for the e-Commerce Solutions practice. He holds a degree in Computer Science; his areas of expertise include object-oriented software development, Web design and development, and solutions integration.

E. Rongen is an IBM Senior IT Specialist in Uithoorn (Netherlands). He holds a PhD in Physics and gained his IT experience working with 4GL languages such as VisualAge Generator. He now works with Web development and Pervasive Computing and is part of several teams in Java and IBM WebSphere projects specializing in emerging technologies and enabling legacy applications for e-business.



J. Shamroski is an IT Architect in the IBM Service Delivery Center West practice. He has over nine years of experience in various aspects of Enterprise Systems Management (ESM) as an ESM Architect in Columbus, Ohio. His areas of expertise include mobile data integration, development, implementation methodology, Enterprise Management architecture and application design using Tivoli products.

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

Cecilia Bardy, Margaret Ticknor, Jeanne Tucker
International Technical Support Organization, Raleigh Center

Christophe Olivier
International Technical Support Organization, Austin Center

Ivan Heninger, Mohit Jain, Joan Boone, Alan Booth, George Hall
IBM Research Triangle Park, North Carolina, USA

Barry Aldred
IBM United Kingdom

Special notice

This publication is intended to help application developers design and implement WAP and PDA wireless applications by adapting new and existing enterprise applications and extending these applications to use new available wireless technologies such as push messaging, intelligent notifications and location-based services. The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM WebSphere Everyplace Server Service Provider Offering Version 2.1 and IBM WebSphere Everyplace Server Enable Offering Version 1.1. See the PUBLICATIONS section of the IBM Programming Announcement for these products for more information about what publications are considered to be product documentation.

IBM trademarks

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

e (logo)®
IBM ®
AIX®
AS/400®
AS/400e™
Balance®
CICS®
Database 2™
DB2®
DirectTalk®
Encina®
Everyplace™
IBM®
IMS™
iSeries™
Lotus®
Lotus Notes®
Notes®
Sametime®
Domino™
Redbooks™
Redbooks Logo
Lotus Sametime™
MQSeries®
OpenEdition®
OS/390®
PAL®
Perform™
Planet Tivoli®
RS/6000®
S/390®
SecureWay®
SP™
SP1®
SP2®
SupportPac™
ThinkPad®
ViaVoice®
VisualAge®
WebSphere®

Comments welcome

Your comments are important to us!

We want our IBM 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 the address on page ii.

Part 1

Introduction

In this part of the redbook, we provide an overview of the IBM WebSphere Everyplace Server (WES) offerings, specifically, the IBM WebSphere Everyplace Server Service Provider Offering (SPO) Version 2.1 and the IBM WebSphere Everyplace Server Enable Offering Version 1.1.

We also include a description of application capabilities when running within an IBM WebSphere Everyplace Server environment; available options are discussed, ranging from an application specifically developed for pervasive use to the more common case of existing applications adapted to pervasive usage requirements. The information provided in this redbook mainly targets Business-to-Employee (B2E) enterprise applications. However, most of the topics presented can also be applied to Business-to-Consumer (B2C) application development.

1 Overview

This chapter presents an overview of IBM WebSphere Everyplace Server offerings. Starting with a functional analysis of Web-based applications, we examine how their functionality can be extended in several directions, to reach the ideal mobile computing situation: “any time, anywhere, any device.”

We then review how this functional requirement is implemented in IBM WebSphere Everyplace Server, detailing the role of each component product and its relationships with the other components.

We compare the structure of two different offerings within the IBM WebSphere Everyplace Server family: the full-featured *Service Provider Offering* (SPO) and the more enterprise-oriented *Enable Offering* (EO). The entry level offering (*IBM WebSphere Everyplace Server Access*) is not covered here.

We conclude by examining the implications of this survey on the development of mobile applications using the IBM WebSphere Everyplace Server offerings.

1.1 A functional approach to understanding IBM WebSphere Everyplace Server

IBM WebSphere Everyplace Server embodies a new step in the evolution of Web-based computing. In order to understand its structure and operation, it is useful to start with a functional review of Web-based computing in general. It goes without saying that a review contained within a few paragraphs is not historically complete or accurate. This is not intended to be a history of Web-based computing. Rather, it should provide a “20/20 hindsight” retrospective on the design of Web-oriented systems, ignoring the twists and turns of technical development to concentrate on emerging technical trends.

1.1.1 Starting with TCP

The past two decades have seen a remarkable convergence of business networks on the TCP/IP model, which was developed for academic, research and military purposes. In the information that follows, we assume an IP network, with TCP as the host-to-host (or connection) layer.

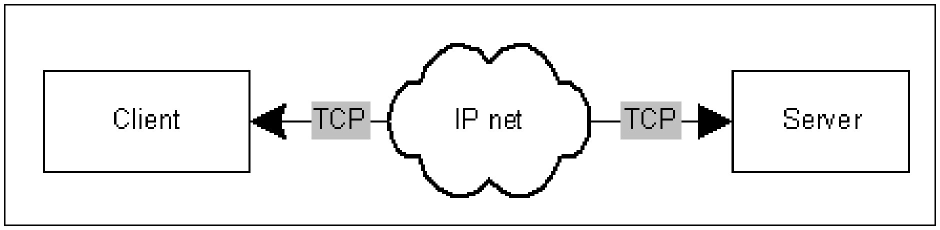


Figure 1-1 Client-server over TCP

In this context, a client-server application must manage all the functions above the connection layer; in other words, it must manage session, presentation and application issues. This implies a client component and a server component sharing some form of (private or public) protocol.

1.1.2 Hypertext: HTTP

The introduction of the Hypertext Transmission Protocol, and the associated Hypertext markup language, made possible the explosion of the World Wide Web.

This was accomplished by commoditizing the Presentation and Application layers on the client side, in the form of HTTP/HTML browsers, at least to the extent of serving hypertext (forms, rich text documents, and limited types of GUI widgets). As a consequence, the supply or certain types of services on the Web (for example, the publishing of technical or scientific papers) no longer required sophisticated programming skills, and could most often be undertaken by the owners of the information themselves. In short order, HTML editors appeared to make the task even easier.

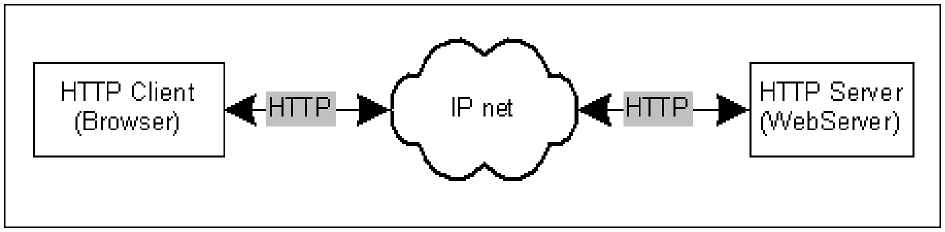


Figure 1-2 Hypertext traffic

Client-side scripting expanded the presentation capabilities, and some add-on solutions appeared for session maintenance. As the number of MIME types expanded, the Browser client could invoke a variety of applications to process the files sent by the server. The server side remained confined to serving such files.

1.1.3 Adding applications

The Common Gateway Interface (CGI) opened up server-side capabilities. Now, properly designed programs could be invoked from the client using appropriate URLs (usually coded into an HTML document).

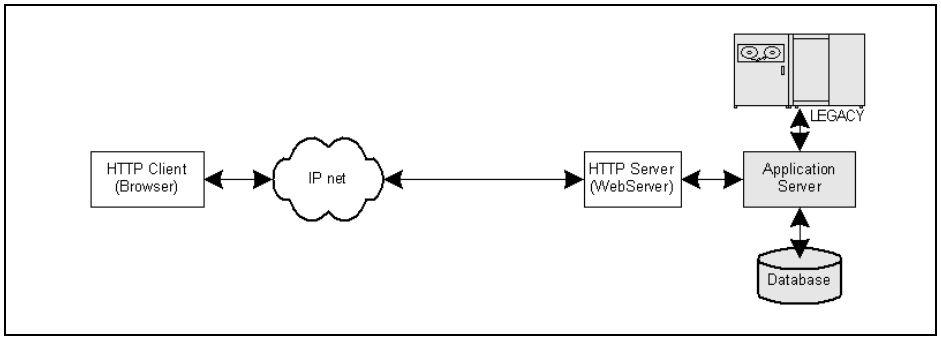


Figure 1-3 Application services over HTTP

The operation of these programs implies another functional unit, the application server, co-located with the Web Server or not.

Further, many server-side programs make use of information extracted from databases and other existing systems. Some form of standard approach to database interaction is desirable.

Among other efforts, we note the enormous contribution of the Java initiatives to the standardization of Web services construction and delivery (both client-side, with applets and the applet security model, and server-side, with servlets, standard database access methods, and later models, including access to a variety of legacy systems).

1.1.4 Controlling access

As the capability and flexibility of the Web computing model increases, serious concerns arise regarding the control of access and protection against vandalism and malicious intent.

Initially, each application designer had to provide protection as part of the application layer. For a moderately complex Web site with diverse applications, this obviously leads to a multiplication of identification, authentication and authorization actions, and impacts the user-friendliness of the site. More importantly, the absence of coordination may leave serious gaps in the security of the site.

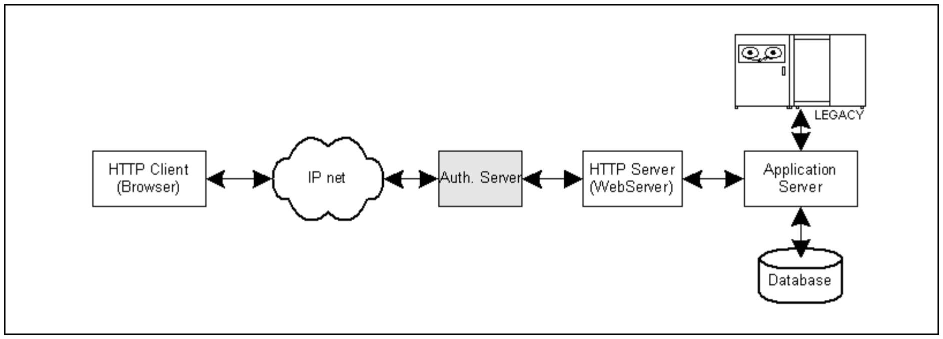


Figure 1-4 Access control in an HTTP context

This leads to the functional concept of *control at the edge of the site*. In Figure 1-4, this functional role is represented by the Auth. Server (short for authentication and authorization), a name given to a component of the previous version of IBM WebSphere Everyplace Suite.

Control at the edge means, among other things, that when a user is authenticated by the server at the edge, that user should not have to present credentials separately for any application or resource controlled by that server. This *single sign-on* feature enhances the user experience, but also contributes to improved security: if users need to remember only one user ID/password pair, they are less likely to write it down in an all too accessible place, thus eliminating the most important cause of security leaks.

1.1.5 Enablement

Edge management starts with perceived security needs, but this function can extend to cover a lot of ground:

- The URLs seen by the browser on the client machine may contain much information about the organization of the site. It has become common practice to hide this information by means of a *reverse proxy*, which presents the user with a uniform facade for the site. Such management of server resources can be further extended with *response caching* and *load balancing*.
- Since HTTP is stateless, the support of *sessions* has to be provided by the application (as for instance in the Servlet model). The sharing of session information between applications is difficult and, if applications run on different servers, it may be impossible. Maintaining session information at the edge allows applications to share this information as needed.
- Information about users is limited to their identity, and possibly access permission to an application. Each application has to maintain its own set of *detailed permissions* (this information can also be maintained by the application server). Maintaining this information in a central directory and using it at the edge both limits the opportunities for unauthorized access to the resources of the site and lightens the functional load of the applications or the application servers.
- The management of security information at the level of individual users is cumbersome, and prone to error. A *policy server* makes it possible to administer this information at a higher level of granularity, which makes sense in terms of business and otherwise.

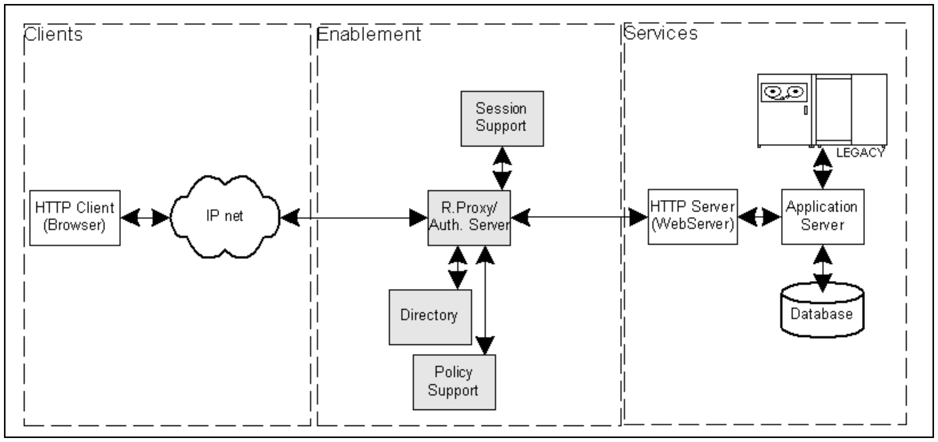


Figure 1-5 Generic enablement

It becomes necessary to expand the structure of the previous diagrams and place a new cluster between the client functions and the services. As a group, these functions provide the following benefits:

- Contribute to facilitating the relationship between client and service.
- Transcend the characteristics of individual applications.
- Can very often be factored out of the applications.

To summarize these features, we label them *Enablement functions.*

In the following discussion, we will focus on these Enablement functions (with a few Client or Service asides), and show how they represent the essential of the IBM WebSphere Everyplace Server value proposition.

1.1.6 Managing users

The facilities offered by the central Directory and the Policy Support are quite useful in steady-state operation. But some form of tool is needed to keep track of background information and use it to produce the data for Directory and Policy. For instance, in a commercial installation, various forms of packaged offerings associate access privileges with billing rates and other data irrelevant to the operation of the Web site as such.

This function would include means for customer care agents to manage the status of their customers, and to handle higher-level entities such as deals and commercial packages. It may also, to some extent, allow customers with the appropriate authorization to perform their own management. In this respect, User Manager assumes some features of an application, with a View and a Controller interacting with the configuration model: hence the arrow connecting it with the Reverse Proxy/Auth. Server function in Figure 1-6.

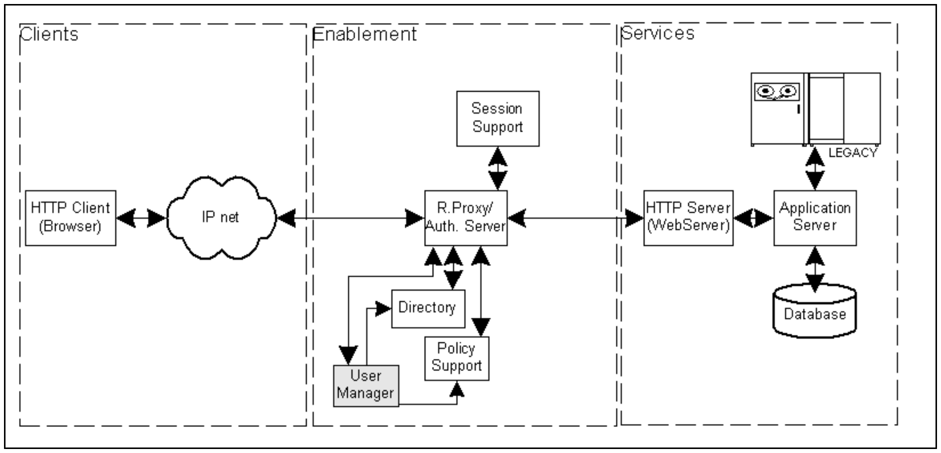


Figure 1-6   User management

1.1.7 Clients beyond IP

So far in our discussion, we have not made any assumptions beyond the accepted boundaries of Web operation. Indeed, all the functions discussed are available using standard Web service offerings.

On the other hand, the past few years have seen an explosive proliferation of mobile devices which do not share the standards of traditional IP networks. The demand for Web access from such devices is high, and some form of communication adaptation is needed.

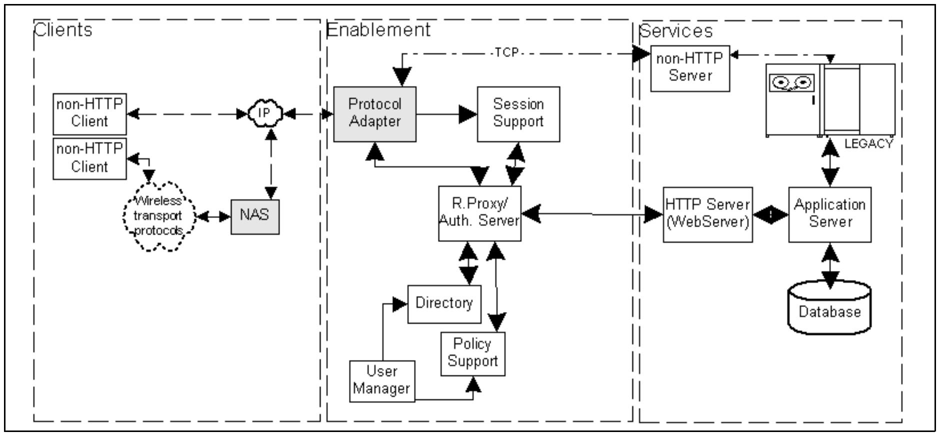


Figure 1-7 Extending the reach beyond IP networks

In the past, several solutions have been proposed to handle the case of devices with no fixed address which nevertheless support the Internet protocol (or some other well-established standard). Presumably, such solutions can be extended to mobile devices, provided some means is found to make them appear as IP devices to the rest of the network.

Figure 1-7 shows the functions needed to accomplish this objective. On the one hand, some Network Access Service (NAS) must be able to present the non-IP devices to the IP network. On the other hand, the higher-level protocols must be adapted to the TCP world. If the given application uses HTTP, the traffic can then be passed on to be handled by the Reverse Proxy/Auth. Server in the usual way. Otherwise, it will behave like any TCP-based application.

1.1.8 Managing devices

The introduction of a variety of devices which do not share the usual (or expected) features of standard computers raises the issue that the application must be aware of the device capabilities. HTTP provides for user-agent information to be carried in a header. However, many details can be subsumed under the user-agent label. Either the application must maintain this information, or it can be managed centrally by a Device Manager.

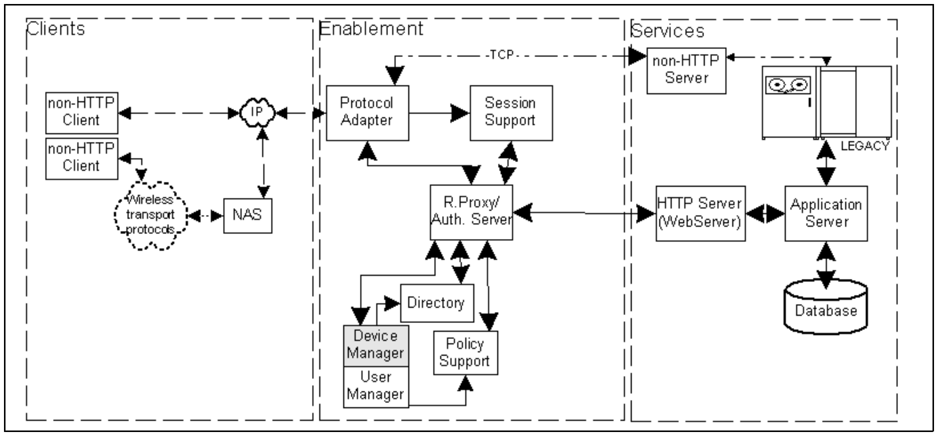


Figure 1-8 Device management

This function is similar to the User Manager in that it behaves in part like an application, accessible by customer care agents or by the customers themselves.

1.1.9 Adapting content

The previous section noted that the application must be aware of the device capabilities in order to supply its content adequately. It should come as no surprise that such adaptation might profitably be factored out of the applications and provided as an Enablement function. The needed adaptation comes in two “flavors”:

- The device may require a *new format*, that is, the device may support a presentation protocol other than HTML. Also, the application may be designed to supply its content in device-neutral XML, which then must be mapped to a particular rendering.
- The device may require a *new medium*, that is, the output device may support some form other than written text and pictures. The example of choice at this time is the standard telephone, which supports Voice.

We consider a single example of each adaptation below.

New format

With the exception of the decades-old adaptation of mainframe formats (*screen scraping*), the most widespread example of format adaptation at this time is the use of WML on WAP devices. With the exception of WAP simulators running on computers, WAP devices also require protocol translation from IP to WAP, as shown in Section 1.1.7, “Clients beyond IP” on page 9.

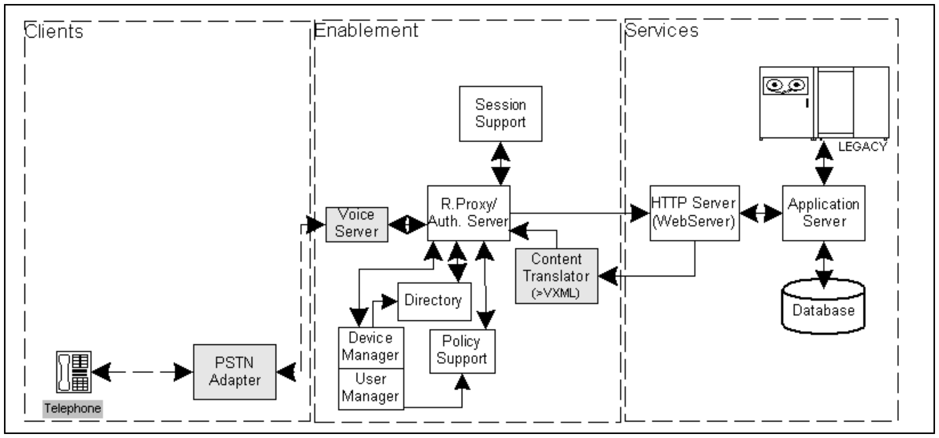


Figure 1-9 Adapting format from HTML to a new markup language: WML

In addition to translating from HTML (or XML) to WML, the Content Translator must typically cope with the limited real estate available in most WAP/WML devices. Accordingly, a *clipping* function accompanies the translation. There are limits to this *application-independent* enablement, however. Quite often, the application designer has relied on special features of HTML browsers (which are far from fully standardized) and a special effort is needed to make the clipped content usable by the mobile user. This may require redesigning some of the application, and recoding some pages.

New medium

These factors become even more important when considering the use of another medium, such as voice over a telephone. To begin with, two extra functions are required:

- Mapping Voice over IP (VoIP) to Voice XML (VXML).
- Adapting the transport from the Public Switched Telephone Network (PSTN) to the IP network. This function is similar to the NAS function presented earlier, with the additional requirement to interface with PSTN.

  Note: PSTN is a connection-oriented, circuit-switched network.

Voice over IP is transported over a connectionless network, where packets travel independent routes: their sequence is not guaranteed and must be re-established at the destination.

With these two functions in place, the content adapter can ensure the mapping from HTML or XML to VXML.

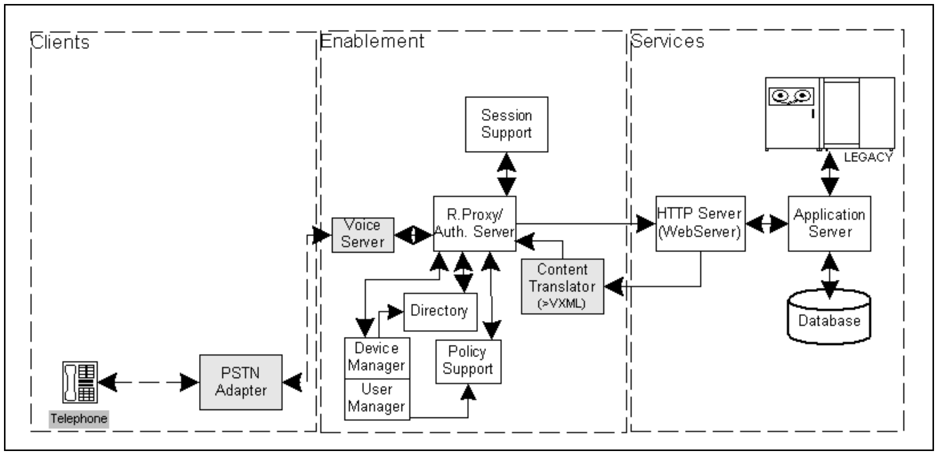


Figure 1-10 Adapting to a new medium: Voice

The designer must take into account the limitations of speech as a medium of communication: decoding spoken language into text is a difficult task, heavily dependent on context. Such context must be available in the design of the application.

Note: Part of this objective can be achieved by structuring the *voice forms* to limit the responses of the user. For details on this, please refer to Redpiece SG24-6259, *Designing and Developing Mobile Applications using WebSphere Everyplace Access V1R1*.

For this reason, few existing applications can be directly adapted for Voice service. A redesign effort is usually required.

1.1.10 Catering to pervasive contexts

Once access to the Web is provided for mobile devices, new computing contexts are made available, in which computing equipment is integrated in every facet of everyday life. The label *pervasive computing* has been applied to this situation.

From the standpoint of applications, there is a great variety of new needs to satisfy, and the more obvious category covers the needs which depend on the current (“real-time”) location of the device.

Within HTTP

Enabling standard HTTP applications to take location into account requires several functional components.

There must be some service capable of mapping the device identifier to the current location (this function is normally provided by the mobile communications carrier, for reasons of confidentiality).

There must also be a component which intercepts the incoming request for service and processes it, first checking various permissions and then requesting the location information from the location service.

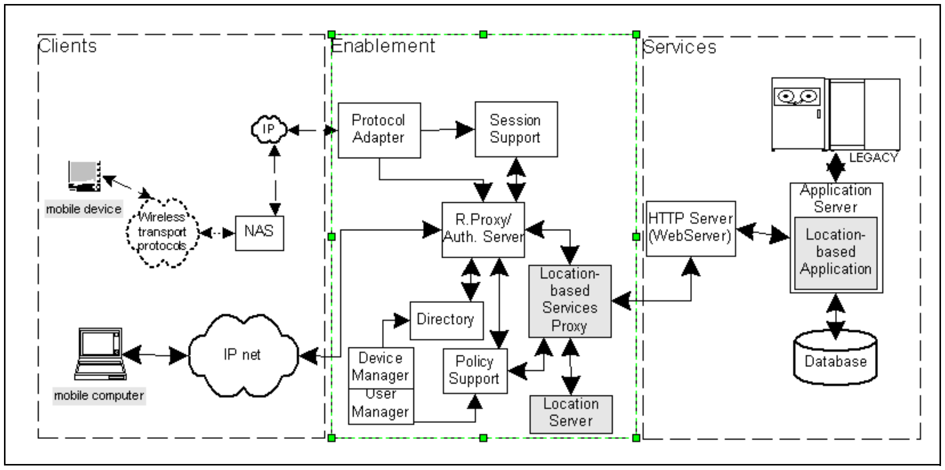


Figure 1-11 Applications specific to mobile usage

The application itself must be designed to make use of the location information if it exists, and to take suitable action otherwise.

Finally, the User Manager must make it possible for potential users to manage location-related permissions.

Beyond HTTP

Web access for mobile devices need not be limited to HTTP applications.

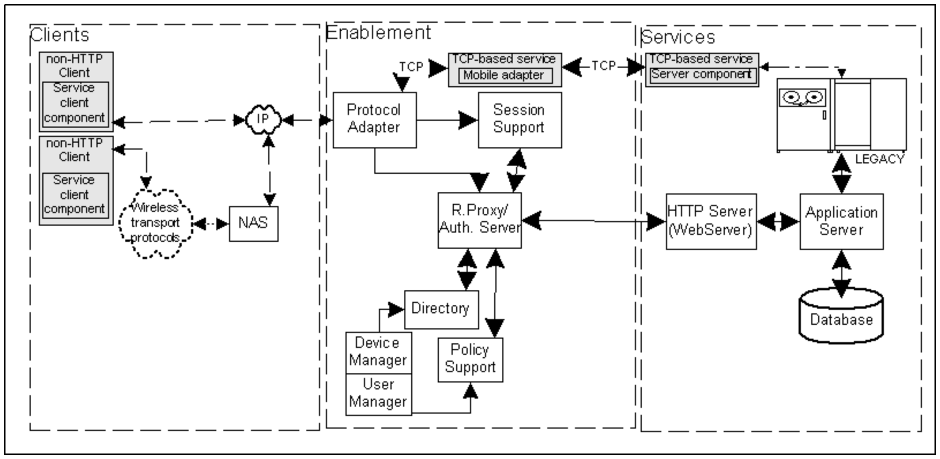


Figure 1-12 Using new capabilities outside HTTP

In the case of non-HTTP services, the standard facilities discussed so far are not available, and any application must consist of three components:

- The service itself, based on a public or private protocol.
- The client adapter, providing the client with any protocol handling needs.
- The enabler, which supports communication between the server and the client. This function is often implemented as an add-on (plug-in) to the mobile service (wireless) gateway.

1.1.11 Synthesis

Figure 1-13 summarizes the previous discussion: the numbers in circles map each function to the products to be discussed in the next section.

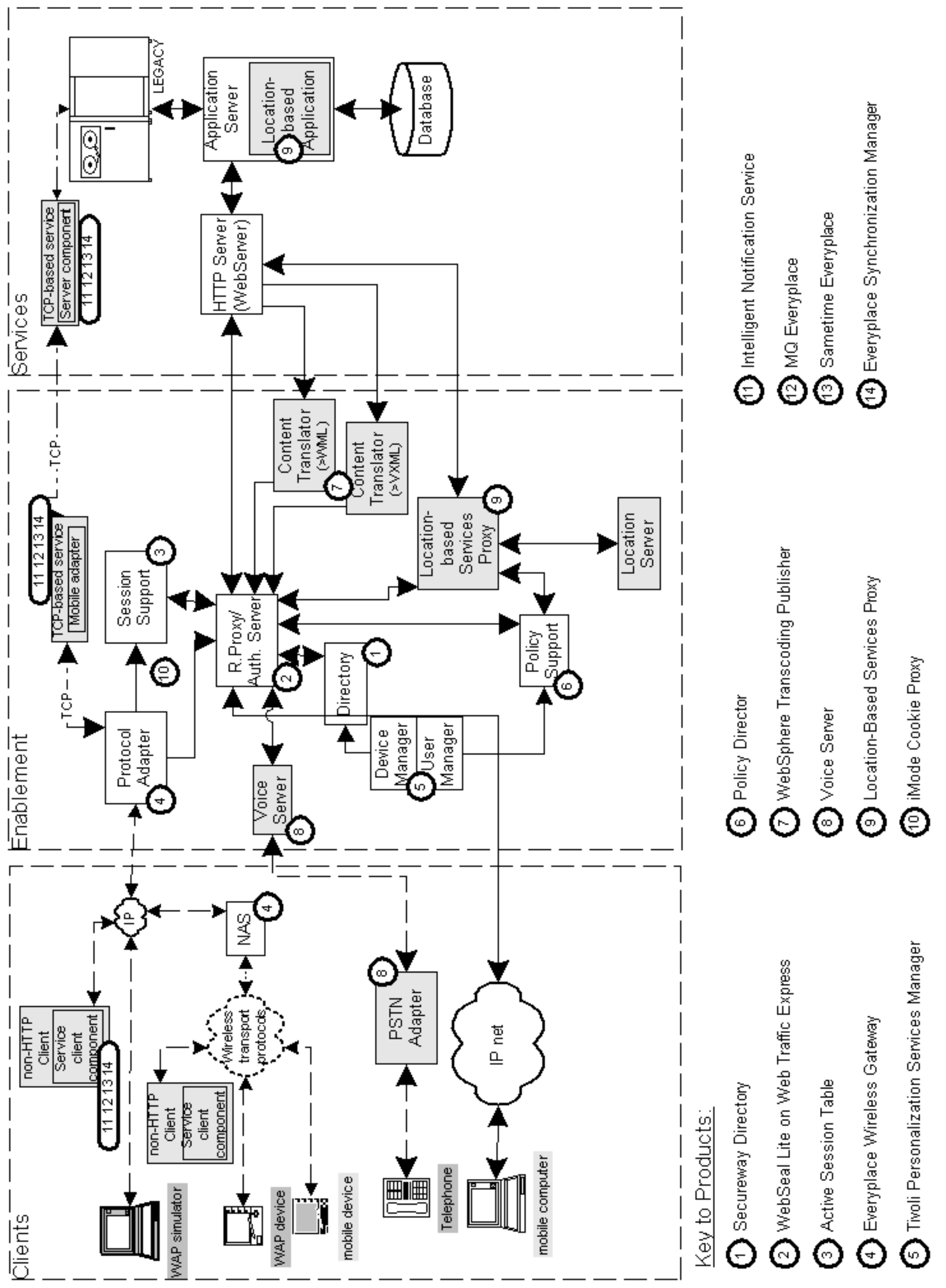


Figure 1-13 Mapping functions to products

1.2 Mapping functions to products

In this section, we take up each of the components of IBM WebSphere Everyplace Server (Service Provider Offering) and review it under five headings.

- *What is it?* describes the product underlying the component.
- *Role in IBM WebSphere Everyplace Server* explains how it fulfills its functions, and how it interacts with other components.
- *Installation, configuration and administration* gives an overview of setting up the component, its status in IBM WebSphere Everyplace Server (whether it is optional or not) and any issues or prerequisites involved.
- *Quality of service* reviews questions of scalability and reliability as they apply to the component.
- *Application considerations* notes whether the component offers any application APIs, or requires particular attention when dealing with certain kinds of applications.

Note: Much of the information offered in this section is given in greater detail in the WebSphere InfoCenter, which is available as part of the IBM WebSphere Everyplace Server Service Provider Offering.

1.2.1 SecureWay Directory

What is it?

SecureWay Directory is also available as a stand-alone product. It is a Lightweight Directory Access Protocol (LDAP) directory server that runs as a daemon. It is based on a client/server model that provides client access to an LDAP server. SecureWay Directory provides an easy way to maintain directory information in a central location for storage, updating, retrieval, and exchange.

Role in IBM WebSphere Everyplace Server

IBM WebSphere Everyplace Server installation and configuration

The LDAP server holds all configuration information for IBM WebSphere Everyplace Server, and supports the Setup Manager (see 1.2.15, “IBM WebSphere Everyplace Server Setup Manager” on page 45) and the Suite Manager (1.2.16, “IBM WebSphere Everyplace Server Suite Manager” on page 46).

IBM WebSphere Everyplace Server operation

Most components of IBM WebSphere Everyplace Server depend on the LDAP directory for their operation, as shown in Figure 1-14.

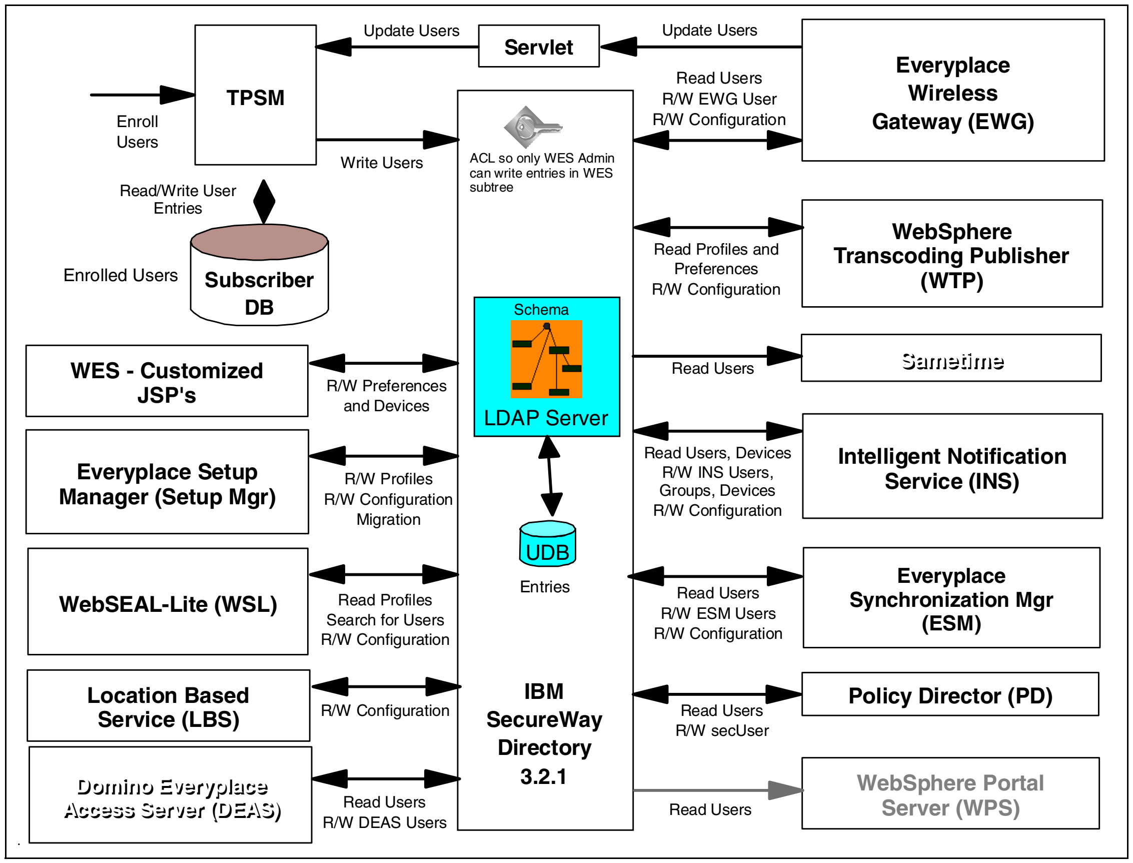


Figure 1-14 LDAP coordinates IBM WebSphere Everyplace Server components

Installation, configuration and administration

For installation and configuration, see the Setup Manager. As the LDAP directory is essential in managing the configuration information, this is the first component installed by the Setup Manager. When using the Setup Manager to install later components, you are prompted to provide the location of the SecureWay Directory. The SecureWay Directory uses a private DB2 database.

For administration, refer to the Everyplace Suite Manager. SecureWay Directory also has two management tools of its own: the *Directory Management Tool* (DMT) and the Web-based *Web Administration console*. Refer to the SecureWay Directory documentation for further information on configuration and administration.

Quality of service

Scalability

The SecureWay Directory uses DB2 for storage. As a consequence, the scaling facilities of DB2 can be used to increase directory size limits and performance. The SecureWay Directory documentation covers this subject in detail, starting with the IBM WebSphere Everyplace Server InfoCenter.

Design issues

IBM WebSphere Everyplace Server requires specific information (about users in particular). When planning the reuse of LDAP-based information (for instance, Enterprise user information), careful consideration must be given to migration or harmonization. It is always possible to maintain an independent LDAP installation for the sole use of IBM WebSphere Everyplace Server, provided some way is found to synchronize the common information between the two directories.

Application considerations

The SecureWay Directory Client SDK APIs to locate LDAP servers that are published in DNS, client-side caching for the Java-based JNDI interface, as well as other JNDI enhancements.

1.2.2 WebSEAL-Lite (WSL) on Web Traffic Express (WTE)

What is it?

Beyond its role in IBM WebSphere Everyplace Server, Web Traffic Express is also available as a component of IBM WebSphere Edge Server. The other component (Load Balancer) is beyond the scope of this survey.

Web Traffic Express is a caching proxy. It also acts as a caching server and content filter, reducing the time needed to retrieve information from the Internet and filtering Internet data.

WebSEAL-Lite is an authentication and authorization plug-in for Web Traffic Express, which provides a central authentication point for all users who wish to access resources within a secured domain. It combines the caching Web proxy of Web Traffic Express with the authorization engine of Tivoli SecureWay Policy Director to deliver protected resources to authorized users.

Role in IBM WebSphere Everyplace Server

WebSEAL-Lite is the central point of user authentication for the IBM WebSphere Everyplace Server domain. It authenticates users defined to the Everyplace Server domain when they attempt to access Everyplace Server services.

Exception: Components of IBM WebSphere Everyplace Server which do not use HTTP transport must manage their own security.

Note: Users accessing the IBM WebSphere Everyplace Server domain through the Wireless Gateway are normally authenticated by the Gateway. However, any further management of security for these users is handled by WebSEAL-Lite.

When a user requests service, WebSEAL-Lite looks for the user in its own cache. If not found there, the user is looked up in the Active Session Table (see next section), where it may have been entered by the Wireless Gateway. If such an entry exists, it is trusted by WebSEAL-Lite. If not, a challenge is issued.

At least one instance of WebSEAL-Lite is required in the Everyplace Server domain to enable integration of most Everyplace Server components. It is the point of entry to the Everyplace Server domain for devices that do not connect through Everyplace Wireless Gateway, and is the next, non-firewall hop for connections through Everyplace Wireless Gateway. WebSEAL-Lite also allows you to use gateways other than Everyplace Wireless Gateway, if desired.

WebSEAL-Lite can be configured in one of two modes:

- Authentication proxy:
  - User authentication based on HTTP Authenticate headers.
  - No other origin server (content or application server) in the Everyplace Server domain may do its own user authentication.
  - Users authenticated through the authentication proxy may not access content outside of the Everyplace Server domain.
- Transparent authentication proxy:
  - User authentication based on HTTP Proxy-Authenticate headers.
  - Origin servers (content or application servers) in the Everyplace Server domain may do their own user authentication.
  - The transparent authentication proxy allows users to access material outside the Everyplace Server domain.

WebSEAL-Lite allows for single user sign-on (with a user ID and password) for all services within the IBM WebSphere Everyplace Server domain. With this feature, user authentication only needs to be performed once to access services requiring a user ID and password. Authentication will still be needed for services outside the IBM WebSphere Everyplace Server domain. Single sign-on is implemented using the LTPA (Lightweight Third Party Authentication) protocol supported by IBM WebSphere Application Server.

WebSEAL-Lite can use the same key file as was created for WebSEAL-Lite in IBM WebSphere Application Server, with a simple editing change. Conversely, IBM WebSphere Application Server can use a key file created for WebSEAL-Lite, with the reverse editing change.

For details, refer to the IBM WebSphere Everyplace Server InfoCenter.

Policy Director implements finer levels of access control rights to Web applications. This can be done by integrating Policy Director with WebSEAL-Lite (see Tivoli SecureWay Policy Director). During installation, the Setup Manager prompts to determine whether WebSEAL-Lite should be configured *with* or *without* Policy Director. For the *with* option to succeed, Policy Director must have been installed previously (Setup Manager does not check whether this is the case). For the *without* option, Setup Manager installs a stub library to replace the Policy Director client library.

Installation, configuration and administration

WebSEAL-Lite runs as a plug-in to Web Traffic Express (Caching Proxy). The Caching Proxy is a prerequisite for WebSEAL-Lite and must be installed on the same machine as WebSEAL-Lite.

WebSEAL-Lite is installed as a subcomponent of IBM WebSphere Edge Server Caching Proxy in Setup Manager. For installation and configuration see Setup Manager.

For administration details, please the see Everyplace Suite Manager.

Quality of service

Scalability

Being deployed as a plug-in to Web Traffic Express, WebSEAL-Lite offers the same scalability as WTE.

Application considerations

There are no application-level interfaces.

1.2.3 Active Session Table (AST)

What is it?

Everyplace Active Session Table (AST) replaces the cache function formerly provided by a component of Tivoli Internet Services Manager (TISM), which is part of Tivoli Personalized Services Manager.

Role in IBM WebSphere Everyplace Server

Everyplace Active Session Table provides a high speed specialized cache for information about users that are currently connected to the IBM WebSphere Everyplace Server domain.

The AST servers are used by the Everyplace Wireless Gateways and WebSEAL-Lite servers. These servers must be able to communicate with the AST servers.

Installation, configuration and administration

Installation of Everyplace Active Session Table is handled through IBM WebSphere Everyplace Setup Manager. Two AST servers should be installed on two different machines in the IBM WebSphere Everyplace Server domain. The first AST server is the primary server and the second is the backup.

Under normal circumstances, the Active Session Table server configuration is performed by the Setup Manager at installation time using default values.

Subsequent updates can then be made to the configuration values from the Everyplace Suite Manager. If, for some reason, LDAP is not running, it is possible to configure the Active Session Table server outside of the Everyplace Suite Manager environment. Consult the IBM WebSphere Everyplace Server InfoCenter for details on this procedure.

Normally, the Active Session Table (AST) is managed from Everyplace Suite Manager, but the IBM WebSphere Everyplace Server InfoCenter details other management methods.

Everyplace Active Session Table properties

The AST properties file contains configuration parameters for the Active Session Table server. Normally, the AST server gets its properties from the LDAP directory and the AST.properties file is used only to specify how the LDAP directory is to be accessed (the properties in the LDAP directory may be changed through Everyplace Suite Manager). But if the LDAP directory is unavailable, then the AST.properties file can be used to configure all aspects of the AST server's operations.

Quality of service

Design issues

Before installing and configuring AST V2.1, you will need to estimate disk storage for the code, the cache data, and the log data. Minimum recommended values are:

- Program files: 0.5 MB
- Cache data: 400 bytes per active user
- Log data: 10 MB

You must also provide sufficient real memory for the AST server's indices. Allow 130 bytes for each active user entry, plus five megabytes for other information. Do not allow the real memory of the system on which the AST is installed to become so constrained that the AST's indices are paged out of real memory. If this occurs, performance will be impacted.

Note: The maximum number of active user entries is defined in the LDAP directory and may be changed using Everyplace Suite Manager.

Application considerations

There are no application-level interfaces.

1.2.4 Everyplace Wireless Gateway

What is it?

IBM Everyplace Wireless Gateway provides a communications platform that enables Internet Protocol and Wireless Access Protocol (WAP) applications to run in a wireless and wired environment.

A number of modules provide very wide protocol coverage:

- ARDIS, Dataradio, DataTAC Support
- Mobitex, Modacom, Motorola PMR
- Dial (GSM, AMPS, PSTN, ISDN)
- IP LAN (CDPD, GPRS, etc., as well as wired environments)
- SMS (Short Message Service)
- PAP (Push Access Protocol)
- SMTP (Simple Mail Transfer Protocol)
- SNPP (Simple Network Paging Protocol)

Wireless Client

The Wireless Client is an optional interface which supports communication through a Wireless Gateway. It wraps network-specific details inside the interface layer and allows IP applications on a mobile computer to run over a wireless network. For example, a radio network would not require any specialized communication protocols for use by a mobile device.

Wireless Client is not installed on IBM WebSphere Everyplace Server machines, but rather on the client device. Wireless Gateway provides mobile devices containing the Wireless Client with access to host and network resources through radio and dial-up networks. It can encrypt, compress, and minimize the data that passes through the wireless link, thereby increasing the speed of messaging.

New features

IBM Everyplace Wireless Gateway Version 2.1 offers some significant new features and improvements:

Wireless Client:

- Supported PC 2000 and Pocket PC platforms with Windows CE
- Supported on Windows Me
- Supported on PalmOS
- Includes DNS caching

WAP:

- *TCP application* is a new WAP service resource to transport WAP application data streams that do not use a browser
- Cookie support for unauthenticated WAP clients
- Elliptic curve cryptography for WAP clients

Integration:

- Integration with Tivoli SecureWay Policy Director
- Support for AIX Version 5
- Netscape Directory Server Version 4.1x for configuration storage
- Mobitex and DataTAC support for Messaging Service and Push APIs

Quality of service:

- Performance enhancements
- Messaging Gateway support for message cancellation and extended quality of service (QoS) functions

Management support:

– Migration Guide (Version 5.1/ Version 1.1.2+ to Version 2.1)
– Account and billing support for RADIUS-based accounting servers
– Per-seat license management to track the number of connected users

Role in IBM WebSphere Everyplace Server

IBM Everyplace Wireless Gateway provides a generalized entry point into an Everyplace Server domain, both for wireless devices and for wired computers (although it is not required in this case, as standard computers on an IP wired network can reach the WebSEAL-Lite/Web Traffic Express server directly).

Devices reaching IBM WebSphere Everyplace Server through the Wireless Gateway are identified and authenticated by the Wireless Gateway, and the information is made available to WebSEAL-Lite through the Active Session Table.

The Everyplace Wireless Gateway offers the option of using RADIUS (Remote Access Dial-In User Services) authentication. The RADIUS server is not part of IBM WebSphere Everyplace Server.

Installation, configuration and administration

For installation and configuration details, please see the Setup Manager.

This component uses its own administration console, *Wireless Gatekeeper*, to administer and configure Wireless Gateway.

Wireless Gatekeeper is a Java-based administration tool. It enables an administrator to configure wired and wireless access protocol (WAP) gateways, add users and mobile devices, define and group wireless resources, and assign administrators to wireless resources.

Refer to the IBM Everyplace Wireless Gateway documentation for further administration information.

Quality of service

Scalability

Everyplace Wireless Gateway uses clusters of gateways to balance load and improve availability. Machines can be dynamically added to and removed from a cluster.

Application considerations

The Messaging Gateway offers a Java API to support Push Access Protocol (PAP) applications.

1.2.5 Tivoli Personalized Services Manager (TPSM)

What is it?

Tivoli Personalized Services Manager (TPSM) is also available as a stand-alone product. It enables service providers to manage subscribers and devices centrally. Management includes enrolling subscribers and devices, providing self care and customer care, maintaining and billing subscriber accounts, and submitting jobs such as software distribution to devices, among others. The following subcomponents make up TPSM:

- System Management
  - Setup of groups, domains, membership plans and deals.
  - Access to subscriber profiles.
- Device Manager:
  - Management of mobile devices (PDAs, subnotebooks, smart phones, etc.).
  - Identification, configuration, and distribution of software to any supported device.
- Enrollment Server:
  - Subscriber and device enrollment engine for ISPs, including a customizable set of screens.
  - Distributes TPSM features to every subscriber.
- Database Integration:
  - Enables the creation of either a DB2 or an Oracle database.
  - Required for any installation of TPSM.
- Customer Care:
  - Enables representatives to open new or *child* accounts, and deactivate or reactivate accounts.
  - Enables representatives to view and update personal information, service plans, payment methods, and e-mail settings.
- Self Care:
  - Enables subscribers to modify the portal pages on their mobile devices.
  - Subscribers can also modify some profile data (address, telephone, billing plan, payment method, premium content).
- Portal Toolkit:
  - Support for development of portal pages.
  - Can delegate authentication to WebSEAL-Lite.

Role in IBM WebSphere Everyplace Server

Function	Role in IBM WebSphere Everyplace Server
RADIUS server	Used: optional, not installed by Setup Manager
Enrollment application	Used
Manager consoles (ISM and DM)	Used
Customer Care Application	Used
Self Care Application	Not used: replaced by IBM WebSphere Everyplace Server User Preferences GUI
Web content hosting server	Not used outside of IBM WebSphere Everyplace Server scope
Integration Toolkit	Not used outside of IBM WebSphere Everyplace Server scope
Portal Toolkit	Not used outside of IBM WebSphere Everyplace Server scope
Device Manager server	Used
Tivoli SecureWay Policy Director Integration	Used: customized in IBM WebSphere Everyplace Server
LDAP Gateway	Used: customized in IBM WebSphere Everyplace Server
Reporting	Not used: not tested by IBM WebSphere Everyplace Server test
Provisioning Toolkit	Not Used: can be set up for specific projects
Service Delivery Platform servlet	Used
Authentication	Used: configured to respect IBM WebSphere Everyplace Server Authentication

In the IBM WebSphere Everyplace Server context, TPSM has an additional subcomponent:

- Everyplace Server Enabler (also known as the *LDAP bridge*):

  This allows TPSM to manage its subscriber database in SecureWay Directory. Whenever TPSM is used to modify any user or device characteristics, the changes must be provisioned into the Directory in order to be available to the rest of IBM WebSphere Everyplace Server.

TPSM provides IBM WebSphere Everyplace Server with three kinds of capabilities:

1. Commercial management, such as package deals, billing information, and payment management.
2. Configuration management, such as portal resources, customer care and self-care.
3. Directory provisioning, that is, translating the relevant consequences of the first two items into the format of the Directory for use by the rest of IBM WebSphere Everyplace Server.

Strictly speaking, only the third type of capability is required by the rest of IBM WebSphere Everyplace Server. If a means were found of independently entering such information in the proper form, TPSM would not need to be installed at all.

On the other hand, TPSM offers unique services to any installation which requires commercial management and/or flexibility in dealing with users and devices.

Installation, configuration and administration

For installation and configuration details, see the Setup Manager.

For administration details, see the Everyplace Suite Manager.

As a stand-alone product, TPSM also includes a number of administration tools. Refer to the Tivoli Personalized Services Manager documentation for further information.

Quality of service

Scalability

The Tivoli Personalized Services Manager engine (transaction processing and event traffic) can be distributed and load-balanced across multiple systems to meet desired performance criteria.

TPSM also includes a suite of applications based on Tomcat or the IBM WebSphere Application Server. Using the latter, all IBM WebSphere Application Server scaling and load balancing options are available.

Design issues

TPSM is delivered with a default set of user interfaces for its suite of applications. In any specific installation, one important task will be to customize these user interfaces.

In terms of deployment, TPSM requires large amounts of storage and processing power. These must be taken into account when designing the physical model of the proposed system.

Finally, the primary function of TPSM is the management of users and devices, for which it offers flexibility and responsiveness. While this is indispensable for service providers or large enterprise installations, it is not crucial for installations where the user and device population is fixed or varies slowly. What is needed in such cases is a reliable means of provisioning the LDAP directory in the relatively rare cases where a change is needed.

Application considerations

The user interfaces are based on JSPs and fully customizable.

The Integration Toolkit, or iTk, is the development kit, provided with TISM, which supports functional extensions and communication with other systems. It communicates with the TPSM database via JDBC, and is composed of four subunits:

- Core iTk
  - Transaction management - multiple database accesses are managed as a single business transaction.
  - Database management - database connections and queries are separated from application logic.
  - Trace and debug - utility classes aid in problem determination.
  - Validation - supports flexible rules for input fields.
  - Exception handling - provides a standard way of handling errors.
- Subscriber Management iTk: information about customers and offerings
  - Business and consumer subscribers
  - Accounts Realms Deals
  - Billing information
  - Methods of payment
- Provisioning iTk: notification of database changes
- Billing iTk: interface to external billing systems
  - Retrieval of deals, accounts, methods of payment, etc.
  - Updates to subscriber data

1.2.6 Policy Director

What is it?

Tivoli SecureWay Policy Director is also available as a stand-alone product.

It is a policy management tool for e-business and distributed applications, addressing the challenges of e-business security: growing complexity and difficulty in implementing security policies across platforms.

Policy Director offers the following functions:

- Access control to Web objects
- Centralized and extensible security (authentication and authorization) for Web and TCP/IP applications
- Support for replication and load balancing
- One-time authentication to access multiple Web resources
- Public Key Infrastructure (PKI) support

Role in IBM WebSphere Everyplace Server

It is clear from the list above that Policy Director shows a certain amount of overlap with other components of IBM WebSphere Everyplace Server. Many of its functions can be supplied, to some degree, by other components. Accordingly, installation of Policy Director is not required for every IBM WebSphere Everyplace Server domain.

On the other hand, Policy Director implements finer levels of access control rights to Web applications. Some components, such as Location Based Services, need access to the unique services of Policy Director (for example, LBS needs to determine whether a given application is registered as "location-based" and whether a given user has authorized access to his/her location information).

In order to ensure cooperation between overlapping components, IBM WebSphere Everyplace Server provides appropriate interfaces between Policy Director and WebSEAL-Lite, TPSM, and SecureWay Directory.

- All resources in the IBM WebSphere Everyplace Server domain are registered in the Policy Director Object space under the WebSEAL_Lite entry. Resources accessible through the WTE server are entered directly, whereas those accessible by proxy are recorded under the subentries *forward* and *reverse*. The configuration also holds the login methods and login support resources (forms).
- TPSM includes a *TISM-PD bridge,* which supports the following administrative functions:
  - Using Policy Director (PD) groups to protect access to URLs
  - Creating groups in the Policy Director GUI
  - Associating PD groups with realms
  - Associating PD groups with deals
  - Automatic creation of PD entry in appropriate PD groups for a subscriber when he/she enrolls in a realm for a deal

- Policy Director reads user records from the SecureWay Directory, where they have been entered by the TPSM Everyplace Server Enabler (LDAP bridge) or other methods.

Installation, configuration and administration

Tivoli SecureWay Policy Director installation and uninstallation are *not integrated* with the IBM WebSphere Everyplace Server installation and uninstallation programs in this release. Tivoli SecureWay Policy Director is *not configurable* from IBM WebSphere Everyplace Suite Manager.

Refer to the documentation provided with Tivoli SecureWay Policy Director for installation, configuration and administration information.

Quality of service

Scalability/reliability

Policy Director supports fault-tolerance and load balancing by deploying multiple replicas in a given installation.

Application considerations

As part of IBM WebSphere Everyplace Server, Policy Director is not intended to be the target of applications. However, the client APIs are available, as well as authentication and authorization exits for extension and customization.

1.2.7 IBM WebSphere Transcoding Publisher

What is it?

The Transcoding Publisher is available also as a stand-alone product. It adapts Web content based on destination device characteristics and network service level.

As a stand-alone product, it can be deployed as a proxy or a reverse proxy, or as a filter in IBM WebSphere Application Server, or used as a Java Bean library. You can enhance the performance of the Transcoding Publisher by also installing Web Traffic Express (Edge Server Caching Proxy) downstream from it. Web Traffic Express stores transcoded material, removing the need to retranscode Web pages each time they are retrieved.

Version 3.5 of Transcoding Publisher includes a number of new capabilities:

- Deployment:
  - WebSphere Transcoding Publisher 3.5 runs as a filter in WebSphere Application Server Version 3.5 (the previous version was not WebSphere Application Server 3.x-compatible).
  - WebSphere Transcoding Publisher 3.5 supports new device types (i-mode), output types (HDML), and image types (WBMP).
  - The Administrative Console can manage *server models*, that is, server configuration data held in a central directory for use by several transcoding servers.
  - When using a central directory to store configuration data, an Administration Console can perform centralized administration without a server being installed.
  - WebSphere Transcoding Publisher 3.5 can run as a reverse proxy on behalf of one or several Web servers.
- Development:
  - Annotators: external files or HTML document tags to mark up portions of documents to include or exclude.
  - Parameterized and internationalized stylesheets.
  - A new data gathering tool for troubleshooting.

The recently released Version 4.0 includes the following new capabilities:

- New functionality:
  - The VoiceXML transcoder converts HTML to VoiceXML for voice applications.
  - The Machine Translation transcoder works with WebSphere Translation Server to translate content into different languages.
  - The Palm transcoder converts HTML to PalmOS HTML for display on Palm VII devices.

  Note: Compatibility with WebSphere Application Server 4.0 will be available in an upcoming service release.
- Deployment:
  - The WebSphere Transcoding Publisher 4.0 server can be deployed as a *plug-in* in WebSphere Edge Server Caching Proxy. This configuration replaces the *proxy with external cache* configuration.
  - WebSphere Transcoding Publisher 4.0 configuration information can be imported/exported to XML files.

  - Configuration information can be manipulated outside the Administration Console.
  - WebSphere Transcoding Publisher 4.0 can incorporate user preferences when operating in an environment where that information is available (for example, WebSphere Everyplace Server).
  - Request Viewer can be used remotely to monitor a remote (proxy or reverse proxy) WebSphere Transcoding Publisher 4.0 server.
  - JavaHelp in The Administration Console and other interfaces.
  - Improved performance.
- Development:
  - External Annotation Editor: facilitates the creation/editing of external annotators.
  - Extended functionality of annotators.
  - Stylesheet Editor: facilitates the creation/editing of XSL stylesheets.
  - You can search within your stylesheet selectors to find the resources you want to work with.
  - You can specify, within an XML document, stylesheets to be applied.
  - Improved user interfaces.

Role in IBM WebSphere Everyplace Server

Within the WebSphere Everyplace Server domain, WebSphere Transcoding Publisher (WTP) is intended to be deployed *as a proxy* (forward or reverse). It is *not* intended to be used as a servlet or a JavaBean.

The WebSphere Transcoding Publisher version delivered in this release of WebSphere Everyplace Server Version 3.5. However, WebSphere Transcoding Publisher Version 4.0 will be included in WebSphere Everyplace Server follow on release.

Note: The samples scenarios presented in this redbook have been developed using the improved facilities of WebSphere Transcoding Publisher Version 4.0

In addition, starting with Version 4.0, WebSphere Transcoding Publisher can also be installed as a plug-in to Web Traffic Express (WTE), thus further simplifying its deployment, and automatically integrating the caching function with transcoding.

The new features available with WebSphere Transcoding Publisher Version 4.0 greatly contribute to the integration of WebSphere Transcoding Publisher as a major content enabler:

- Plug-in configuration improves efficiency and simplifies deployment.
- User management in LDAP supports greater integration of user preference support.
- Portable configuration via XML facilitates scaling to multiple servers.
- VoiceXML capability further integrates Voice Server and voice applications into IBM WebSphere Everyplace Server.
- WebSphere Transcoding Publisher can be used in conjunction with Web Traffic Express (WTE) to support caching of transcoded pages.

Note: In addition, WTE can also be used with the WAP gateway in Everyplace Wireless Gateway (EWG) to support caching of binary WML content, this provides a marked improvement in performance for WAP delivery.

Installation, configuration and administration

For installation and configuration see *Setup Manager*.

For administration see *Everyplace Suite Manager*.

As a stand-alone product, WebSphere Transcoding Publisher is equipped with an administration console. Any administrative can be carried out using the console, including importing and exporting the configuration in XML form.

Quality of Service

Scalability

With WebSphere Transcoding Publisher 4.0 configured as a plug-in to Web Traffic Express, incorporating several WebSphere Transcoding Publisher servers in a WebSphere Everyplace Server installation amounts to configuring multiple Web Traffic Express proxies. This is a well-understood WebSphere Edge Server deployment task.

Configuration can be managed centrally from a single Administration Console. The Request Viewer can be used remotely to monitor WebSphere Transcoding Publisher traffic and for troubleshooting.

Design Issues

WebSphere Transcoding Publisher can be deployed in several configurations (proxy, reverse proxy, plug-in). It is important to give close consideration to these possibilities when designing the physical model of the installation. WebSphere Transcoding Publisher's small footprint also contributes to its versatility.

Application Considerations

A number of development tools are available to support WebSphere Transcoding Publisher development: they are explained in detail in Chapters 4-6.

WebSphere Transcoding Publisher can also be deployed as a servlet (filter) in the WebSphere Application Server, or as a Java Bean Library in support of independent applications. As these options are not supported in the WebSphere Everyplace Server context, they will not be covered in this redbook.

1.2.8 Voice Server

What is it?

WebSphere Voice Server is a stand-alone product. Web developers can design and develop applications that can be voice-enabled by utilizing a Voice over IP (VoIP) network infrastructure. No new Web development skills are required to create voice applications (although the design of Web Browser interactions requires a thorough understanding of voice as a delivery medium).

WebSphere Voice Server is based on industry standards (VoiceXML to interact with the Web content and H.323 for Voice over IP). It can interface with several VoIP gateways (tested with Cisco Gateways:1750, 2600, 5300).

WebSphere Voice Server supports U.S. English, French, German, and U.K. English; it is available on Windows NT.

Development is supported by the IBM WebSphere Voice Server SDK, available on Windows NT (forthcoming version for Windows 2000). The SDK is described in detail in Chapter 11, “Location-Based Services (LBS)” on page 435.

Role in IBM WebSphere Everyplace Server

WebSphere Voice Server is not part of the standard IBM WebSphere Everyplace Server offerings. However, it can be smoothly integrated with WebSphere Everyplace Server to support delivery of applications over telephone lines.

Authentication can be handled by WebSEAL-Lite, provided that the passwords chosen are DTMF-compatible.

Installation, configuration and administration

The installation and configuration procedure of WebSphere Voice Server is detailed in the *IBM WebSphere Voice Server Version 1.5 Administrator’s Guide* which accompanies the product.

It consists of the following steps:

1. Configure the telephony environment:

   to be executed by a VoIP gateway expert.

2. Install the Voice Server software.
3. Start VoiceXML browsers.
4. Verify the installation.
5. Run the applications.

Quality of Service

Scalability

WebSphere Voice Servers can be deployed simultaneously on several machines. The load balancing is provided by the VoIP gateway(s).

Design Issues

The *IBM WebSphere Voice Server Software Developers Kit (SDK) Programmer's Guide* which accompanies the product gives detailed guidance for the design of a Voice Browser interface.

Application Considerations

As an autonomous product, WebSphere Voice Server SDK supports the whole development cycle for the voice handling part of an application. It includes tutorial matter on best practice in VXML development. If the application consists solely of a voice interface (in VXML) connecting to back-end functionality, nothing more is needed. In the context of WebSphere Everyplace Server, it is to be expected that the Voice Server will be used as part of an *enabler* for applications. In this case, the VXML documents would be supplied by a WebSphere Transcoding Publisher, rather than being directly generated by the back-end application.

1.2.9 Location-Based Services (LBS) Proxy

What is it?

Location-Based Services Proxy is shipped with WebSphere Everyplace Server: it installs as a plug-in of Web Traffic Express (Edge Server Caching Proxy).

Location-Based Services Proxy provides user location information to applications, which can use this information to deliver appropriate content. For example, an application can provide a user with a list of hotels in the area where he/she is currently traveling.

Role in IBM WebSphere Everyplace Server

For location information to be used by an application, the administrator must enable the application as a location-based application (in Policy Director) and the user must allow the application to use their location information (for instance using TPSM Self-care).

Location-Based Services uses a location server (currently Signal Soft Local.info Server) to provide user location information (in a subset of GML, Geographical Markup Language, defined by the OpenGIS Consortium). Location-Based Services obtains the following information from SignalSoft:

- Latitude
- Longitude
- Uncertainty of the position estimate
- County
- City
- State/province
- ZIP/postal code
- Country
- Nearest Intersection

Location-based applications depend on the Location-based Proxy. It, in turn, depends on the Location Server and the Policy Director. Both of these must be installed separately and configured for Location-Based Services to work correctly.

Installation, configuration and administration

For installation and configuration, see the Setup Manager.

For administration, see the Everyplace Suite Manager.

There are several considerations specific to installing and configuring Location-Based Services. Policy Director is a prerequisite for the installation of the Location Server. As WebSEAL-Lite must be configured at installation to work with Policy Director, it is necessary to plan ahead if Location-Based application *deployment* is intended.

The following must be installed and configured:

– The Signal Soft Local.info Server including the customized "WhereAmI" service.
– WebSEAL-Lite
– WebSphere Everyplace Suite Wireless Gateway

- Policy Director Server 3.7.1
- Policy Director Client 3.7.1

After installation, the following configuration tasks are required:

- Setting up user preferences (in TPSM).
- Configuring location based applications in Policy Director.
- Optionally, preparing Location Based Services Application Registration File.
- Optionally, configuring Location Based Services in Suite Manager.
- Preparing Location Based Services Application Registration File.

Quality of Service

Scalability

Being deployed as a plug-in to Web Traffic Express, WebSEAL-Lite offers the same scalability as WTE.

Application Considerations

Development of the location-based application is based on a standard API which extracts the GML information from the header supplied by the Location-Based Proxy.

Several possibilities are available for development:

- Do not install the Proxy. In the absence of the header, the API will look up information in a file.
- Do not install the Proxy, but equip the development environment to insert the required GML headers in the request.
- Install the Proxy, but not the Location Server. The installation package provides a SIgnalSoft emulator.
- Install both the Proxy and the Location Server.

1.2.10 i-Mode Cookie Proxy

What is it?

Everyplace Cookie Proxy serves to enable users to use i-Mode phones.

Role in IBM WebSphere Everyplace Server

Everyplace Cookie Proxy is only available when WebSphere Everyplace Server is installed on Japanese locale machines. This proxy is provided on AIX only. Messages and other information displayed in the interface are in Japanese only (English is not supported).

Installation, configuration and administration

See the Setup Manager. Installation is only available when installing on machines with a Japanese locale.

1.2.11 Intelligent Notification Services

What is it?

Everyplace Intelligent Notification Services is shipped as part of WebSphere Everyplace Server.

It delivers messages to users of pervasive devices based on the users' preferences and subscriptions. For example, subscribers can tell Everyplace Intelligent Notification Services to notify them when news articles with "pervasive computing" in the headline are published by a content provider. Subscribers can also specify which devices to send a notification to based on the urgency of the message (for example, if the message is marked FYI, send it via e-mail. If the message is marked urgent, send it via Sametime instant messaging).

Role in IBM WebSphere Everyplace Server

Intelligent Notification provides services for publishing content, subscribing to content, and sending simple notifications to other Everyplace Server users.

It supports:

- Lotus Sametime
- Wireless Application Protocol (WAP)
- Short Message Service (SMS)
- SMTP e-mail

Installation, configuration and administration

Everyplace Intelligent Notification Services is installed by Everyplace Setup Manager when selected from the Everyplace Setup Manager. Setup Manager installs all of the prerequisite software for Everyplace Intelligent Notification Services.

During the installation, Setup Manager prompts for configuration information. Everyplace Intelligent Notification Services requires at least 7MB of disk space, not counting the requirements for the private WebSphere Application Server and DB2.

The following programs are installed (privately) by Setup Manager when Intelligent Notification is installed. They are not licensed for deployment of other applications.

- HTTP server
- WebSphere Application Server Advanced Edition
- SecureWay Directory
- DB2 UDB Enterprise Edition

Optionally, Everyplace Wireless Gateway is installed to supply WAP, SMS, and e-mail gateways.

A content provider may also be needed to implement a subscription solution if it is not supplied from local resources. The content provider provides a client application that is connected to an Internet data feed. The content adapter interfaces with this application to retrieve data to publish to the iQueue Server.

Everyplace Intelligent Notification Services requires at least 7 MB of permanent disk space. Additional disk space may be needed to store XML content files from the data feeds.

INS may be configured for several functions:

- Authentication for secured subscription management.
- e-mail notification.
- SMTP e-mail notification.
- Wireless Gateway Push e-mail notification.

Detailed instructions are available in the WebSphere Everyplace Server InfoCenter.

Quality of Service

Scalability and Design Issues

- INS consists of three components (Gryphon, IQ Server and UND), each running in its own JVM and communicating over sockets. If the traffic justifies it, several installations of INS could be deployed in the same IBM WebSphere Everyplace Server domain. However, the load distribution among them would be a matter of application design (for example, assigning different data feeds to different installations).
- Developing an INS-based application can be intricate. Such an application has value if the *intelligence* feature of INS is required. For uses which require simple notification, and do not require user subscription, using the Push API in a direct application can be more efficient.

Application considerations

Customization of the provided samples is required to tailor the services to your installation's needs. For example:

- ▸ Content adapter samples illustrate how to publish content from various sources.
- ▸ Subscription samples show how to subscribe to and process content.
- ▸ Gateway adapter samples are also available for modification to support new gateway interfaces.

The samples for this product require JSP 1.1 support. Ensure that a browser with support for JSP 1.1 is available for viewing the sample front-ends.

Customization is an important part of setting up the Everyplace Intelligent Notification Services. Customers must customize several code components in order to create a notification solution that works for their particular business needs. Everyplace Intelligent Notification Services provides a default or sample implementation of these components.

The following parts of Everyplace Intelligent Notification Services can be customized:

- – Subscription triggers: forms, servlets, and trigger handlers
- – Content Adapters
- – Gateway Adapters
- – Transcoders
- – Transcoder Style Sheets
- – Content-serving servlet.

1.2.12 MQSeries Everyplace (MQE)

What is it?

MQSeries Everyplace is also available as a stand-alone product.

It provides assured messaging capability between devices and any MQSeries family platform. It extends secure messaging to include dependable communications with mobile workers. It connects laptops, servers, PDAs, phones, and unattended devices, such as sensors, to MQSeries networks. This enables users to perform business functions through their mobile devices.

MQSeries Everyplace consists of Java and C components enabling solution developers to create an MQSeries Everyplace gateway and client on a variety of devices and platforms.

Role in IBM WebSphere Everyplace Server

MQSeries Everyplace can function independently, using the facilities of the Wireless Gateway only to support wireless communication. It has its own security management and transport protocol.

MQSeries Everyplace can also use the facilities of IBM WebSphere Everyplace Server to communicate with a servlet which incorporates the MQe class library. In this HTTP mode, an application can overcome issues of firewall configuration (for example, port selection), and authenticate itself through the WebSEAL-Lite mechanism by placing the user ID and password in the HTTP header.

Installation, configuration and administration

For installation and configuration, see the Setup Manager documentation provided with this component. The native C client version of MQSeries Everyplace is not installed with Everyplace Suite. See the Web site for more information on how to get it. Chapter 14, “Transaction messaging” on page 575 presents the details of configuration and administration.

Quality of Service

Scalability/Design Issues

A MQSeries Everyplace application on the server side runs in a JVM and listens on a specific port. Only one Queue Manager can run in a given JVM. In order to scale the application, several JVMs must be started, each listening on a different port. The client-side application must be configured to address one of the ports. In other words, load balancing must be managed by the user.

Application considerations

For further discussion of design and development, see Chapter 13, “Data synchronization for enterprise applications” on page 517.

1.2.13 Sametime Everyplace

What is it?

Sametime Everyplace is also available as a stand-alone product.

It extends the capabilities of Sametime to WAP-enabled devices such as mobile phones. It allows you to chat with other Sametime users from your mobile phone, whether they are using mobile devices, or whether they are using Sametime Connect from their desktop. Sametime Everyplace allows you to:

a. Create Contact lists and search for users.

b. View status information to see whether a person is a mobile user and to see who is online.

c. Use the Chat function for live instant messaging over WAP.

d. Invite multiple Sametime users to join a group chat.

e. Notify, via Short Message Service (SMS), mobile users who are logged on to Sametime, but not currently using Sametime.

f. Send e-mail to users who are not online.

Although some of these functions (for example, *a.*, *c.* or *f*) may be awkward to use on limited devices, they are available as needed.

Sametime Everyplace must be installed on a Domino server in your organization. You access Sametime Everyplace with the WAP browser on your mobile device. A bookmark points to the server where Sametime Everyplace resides. You then log into Sametime using your user name and your Internet password.

You can chat with any person or group listed in your Contact list, if they are active in Sametime. Anyone in your organization's address book can be included in your Contact list. If a mobile user is logged on to Sametime but not currently using Sametime, you can send the user a notification that you are trying to reach him/her. If the user is not logged into Sametime or is not a Sametime user, a mobile user can send an e-mail to that person's mail database.

A User Profile is created the first time you log into Sametime Everyplace from your mobile device. You are prompted for your telephone number, the language to be used on the mobile device and to add a person or group to your Contact list. Once you have entered this information, you can chat with anyone listed on your Contact list. You can accept the defaults set in your User Profile or you can modify your User Profile from your mobile device or from a Web browser. It may be easier to use a Web browser than to use the keys and micro display on the mobile device. To access your User Profile from a Web browser, you provide a URL pointing to the Domino server containing Sametime Everyplace.

Role in IBM WebSphere Everyplace Server

The Domino Server providing the Sametime Everyplace service uses information from the Directory. The user’s identity is obtained and managed by the Sametime server.

Installation, configuration and administration

Sametime Everyplace installation and uninstallation are *not integrated* with the WebSphere Everyplace Server installation and uninstallation programs in this release.

Sametime Everyplace is *not configurable* from WebSphere Everyplace Suite Manager.

Refer to the documentation provided with Sametime Everyplace for installation, configuration and administration information.

1.2.14 Everyplace Synchronization Manager (ESM)

What is it?

Everyplace Synchronization Manager enables handheld computing devices to link remotely to desktop applications. Mobile users can easily synchronize data with Microsoft Exchange, Lotus Notes, DB2 or any ODBC compliant database, such as Oracle or Sybase. The mobile device can synchronize using modem, wireless communications, Internet, intranet, LAN or WAN. Mobile users can be authenticated through existing Microsoft Exchange or Lotus Notes user data or through a list of users held internally in Everyplace Synchronization Manager. Data can be encrypted for secure transmission. Mobile devices can be automatically backed up or restored and applications can be remotely installed on these devices.

Everyplace Synchronization Manager contains the following subcomponents:

- Everyplace Synchronization Manager Service

  Handles the request from the mobile device, manages security, and performs all the data transfers between the mobile and the enterprise data sources. Runs on a Unix server.

- Exchange Connector

  Enables Synchronization Manager to synchronize with Microsoft Exchange Server. Runs on Windows NT 4.0 or 2000.

- Notes Connector

  Enables Synchronization Manager to synchronize with Lotus Notes. Runs on a UNIX server.

- Everyplace Synchronization Manager Admin

  Enables the administrator to set up or modify the synchronization performed by the Synchronization.

- Manager service

  Uses wizards or intuitive forms. Runs on a UNIX server.

- Everyplace Synchronization Proxy

  Mobile devices may synchronize either directly (through dial-up or packet network) to the Synchronization Manager Service or indirectly with a serial cable to a desktop PC which then connects to the Synchronization Manager Service. Everyplace Synchronization Proxy must be installed and running on the desktop PC to synchronize through cable. Runs on Windows.

- Everyplace Synchronization Client

  Enables the mobile device to synchronize with enterprise data sources through the Synchronization Manager Service. Clients are packaged with the Windows install library.

Role in IBM WebSphere Everyplace Server

Currently the Everyplace Synchronization Manager is not fully integrated with IBM WebSphere Everyplace Server. While it uses some components of IBM WebSphere Everyplace Server (for example, Everyplace Wireless Gateway), and is installed by the Setup Manager, it continues to manage its own security. It is a typical non-HTTP pervasive application.

Installation, configuration and administration

For installation and configuration, see the Setup Manager. This component has its own administration console. Refer to the Everyplace Synchronization Manager documentation for further configuration and administration information.

1.2.15 IBM WebSphere Everyplace Server Setup Manager

Everyplace Server contains many individual components and requires several additional server products. Installing so many components individually is not practical, let alone installing them into a distributed setting. Intuitively, the Everyplace Server installation is designed to be a centralized installation of all components.

Many of the components within Everyplace Server exist independently as products with their own installation process. The centralized Everyplace Server installation still utilizes the server software installation and upgrade mechanism already in place for the server components. The server installation is a coordinated effort. It determines what common configuration parameters can be factored out and supplied only once, but used by multiple components.

In addition, as components are installed and configured on specific systems, the Everyplace Server installation process captures configuration parameters and stores the information in LDAP for subsequent use later in the installation process. This way, the parameters entered during installation can be saved for other Everyplace Server component installations.

Setup Manager installs and configures the WebSphere Everyplace Server environment for you. When installing individual components, it is important that you follow the WebSphere Everyplace Server installation and uninstallation instructions provided in the WebSphere Everyplace Server InfoCenter, and *not* the installation instructions provided with the component's documentation.

Setup Manager can perform the following tasks:

- Set up the Everyplace Server environment.
- Allow limited migration from previous releases to version 2.1.
- Help you save time on subsequent installations of Everyplace Server by pre-defining component information.
- Provide a mechanism to validate that the installation and configuration completed successfully.

1.2.16 IBM WebSphere Everyplace Server Suite Manager

Everyplace Suite Manager allows you to administer components. With it you can start and stop servers, edit properties, and view logs.

Everyplace Suite Manager provides a centralized method for launching the administration consoles of the installed WebSphere Everyplace Server components. In addition, Suite Manager obtains information regarding the installed components and the servers where they are installed. From the Suite Manager console, you can make changes to configuration data that is stored in SecureWay Directory (LDAP).

Before you can start Everyplace Suite Manager, SecureWay Directory must be installed and running in the WebSphere Everyplace Server domain.

Installation, configuration and administration

- For installation on Unix (AIX or Solaris), see the Setup Manager.

  After installation, you can start the console by clicking the **console** icon, which was created during the installation process, or from a command line with the Administration Console user ID (or the root user ID), by running the script `IBM WebSphere Everyplace Serverconsole.sh`.

  Everyplace Suite Manager ensures that all of the conditions for use, such as user privilege and prerequisite software, are correct. If the conditions are met, a list of the installed Everyplace Server components is displayed.

- For installation on Windows 2000:

  a. Insert WebSphere Everyplace Server Disc 1 into your CD-ROM drive.

  b. Run.... <cdrom drive>:\eps\win32\SuiteMgr.exe.

  c. Click **OK** and follow the installation program.

  After installation, you can start the console from the Start menu:

  Start-> Programs -> WebSphere Everyplace Suite Manager -> WebSphere Everyplace Suite Manager.

or from a command line:

```
installLocation\SuiteManager.cmd
```

Note: The default install location is C:\Program Files\SuiteManager.

1.3 The IBM WebSphere Everyplace Server Offerings

The WebSphere Everyplace Server Family includes three offerings:

1. WebSphere Everyplace Access Offering for Multiplatforms Version 1.1 (WebSphere Everyplace Access V1.1). This is an entry-level product, focused on Voice enablement of Web applications.
2. WebSphere Everyplace Server Enable Offering for Multiplatforms Version 1.1 (Everyplace Server Enable V1.1, IBM WebSphere Everyplace Server EO). This focuses on supporting enterprise enablement (B2E), leveraging existing infrastructure (especially Directory services and Authentication).
3. WebSphere Everyplace Server, Service Provider Offering for Multiplatforms Version 2.1 (Everyplace Server Service Provider V2.1, IBM WebSphere Everyplace Server SPO). This offering replaces IBM WebSphere Everyplace Suite Version 1.1. It is aimed at large enterprises and Service Providers. The IBM WebSphere Everyplace Serverv.1.1 feature set has been expanded to include additional advanced features.

1.3.1 Using the IBM WebSphere Everyplace Server Offerings

The new release of WebSphere Everyplace Server endeavors to present specific offerings adapted to emerging enablement trends and needs. IBM WebSphere Everyplace Server SPO continues the general make-up of the previous releases, improving integration and administration tools, and adding new technologies. IBM WebSphere Everyplace Server Enable Offering is a new, tailored combination of components intended to give enterprises the means to enable their information systems for a pervasive computing context.

Feature comparison

Figure 1-15 shows the composition of the two offerings side by side.

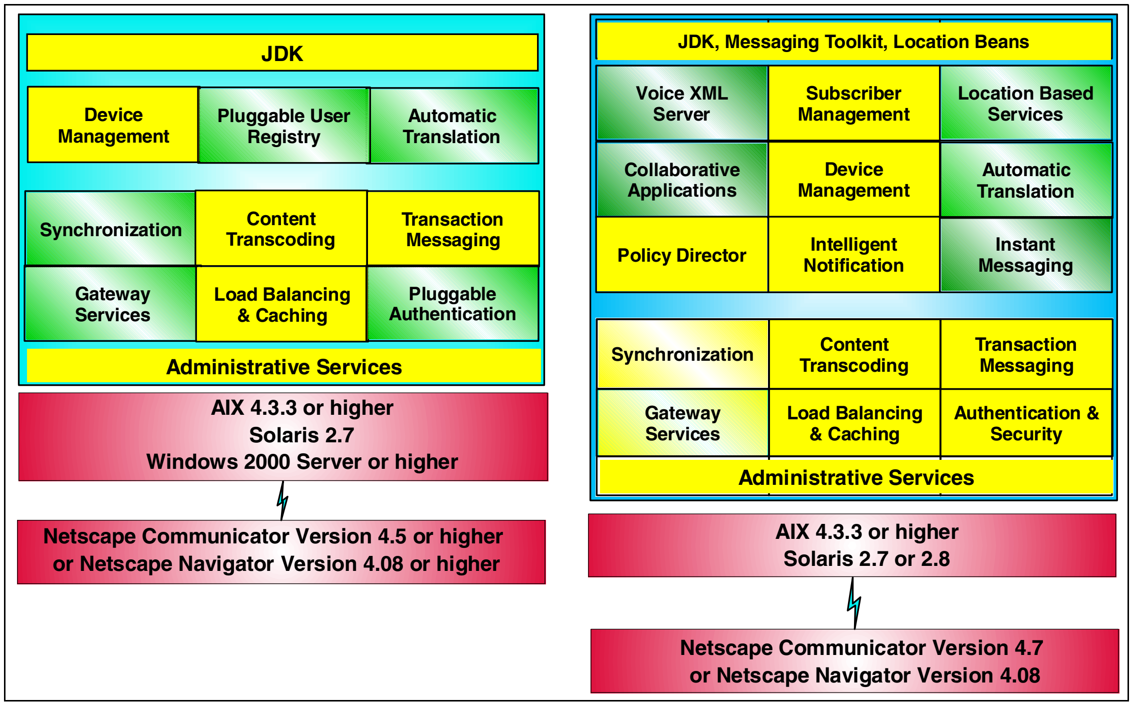


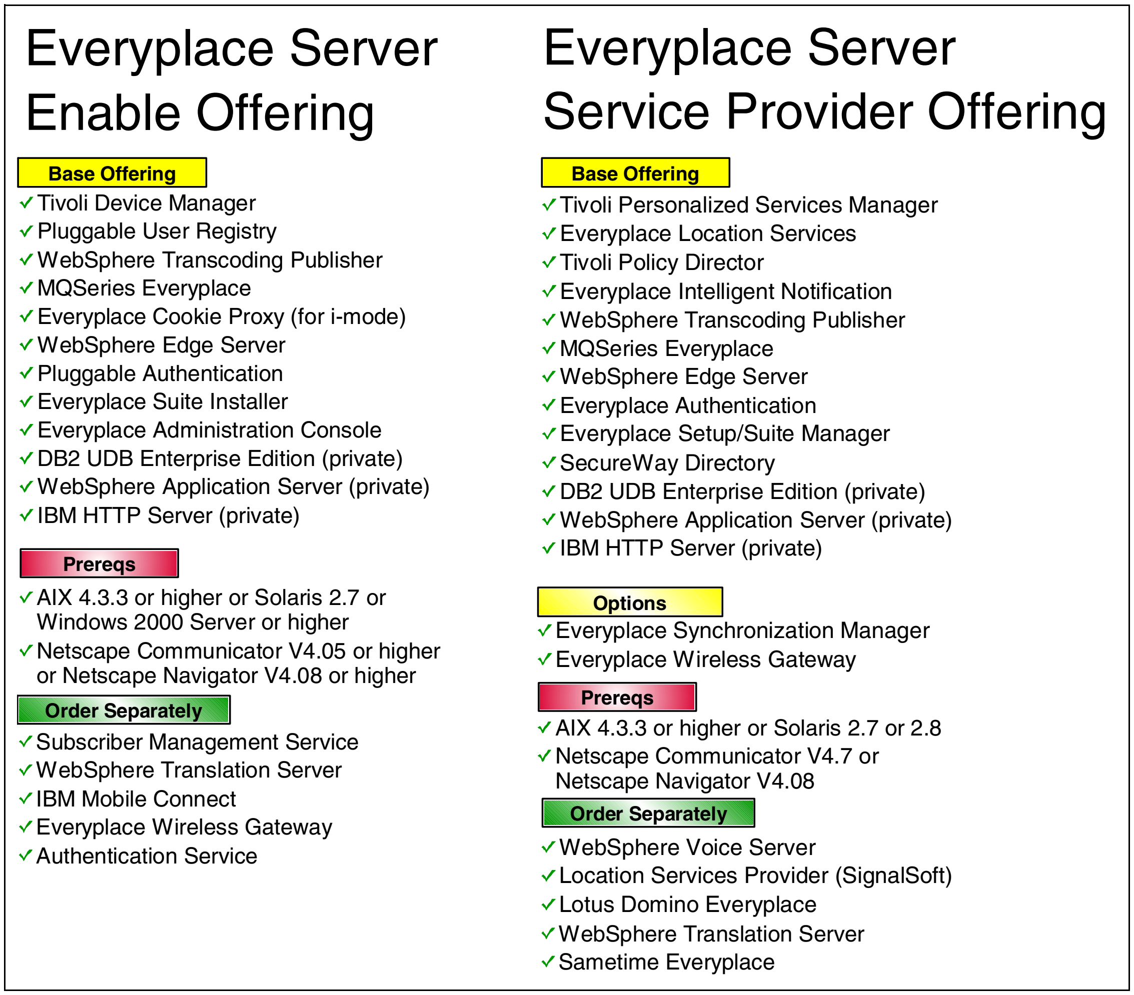
Figure 1-15 WebSphere Everyplace Server Enable Offering versus Service Provider Offering

Licensing

The licensing arrangements further extend the flexibility of the offerings. Several licensing keys allow different configurations to be managed by the same Setup Manager. For instance, the Everyplace Wireless Gateway is only available with specific WebSphere Everyplace Server license keys.

See the InfoCenter for information on license keys. Several components of IBM WebSphere Everyplace Server make use of DB2 and/or WebSphere Application Server as part of their internal operation. Such supporting subcomponents need not be licensed separately for their use as part of a IBM WebSphere Everyplace Server component (denoted as *private* in product descriptions), but may not be used for other purposes.

Figure 1-16 illustrates the available Service Provider and Enable base offerings, including prerequisites and options.



Everyplace Server Enable Offering

Base Offering

- ✓ Tivoli Device Manager
- ✓ Pluggable User Registry
- ✓ WebSphere Transcoding Publisher
- ✓ MQSeries Everyplace
- ✓ Everyplace Cookie Proxy (for i-mode)
- ✓ WebSphere Edge Server
- ✓ Pluggable Authentication
- ✓ Everyplace Suite Installer
- ✓ Everyplace Administration Console
- ✓ DB2 UDB Enterprise Edition (private)
- ✓ WebSphere Application Server (private)
- ✓ IBM HTTP Server (private)

Prereqs

- ✓ AIX 4.3.3 or higher or Solaris 2.7 or Windows 2000 Server or higher
- ✓ Netscape Communicator V4.05 or higher or Netscape Navigator V4.08 or higher

Order Separately

- ✓ Subscriber Management Service
- ✓ WebSphere Translation Server
- ✓ IBM Mobile Connect
- ✓ Everyplace Wireless Gateway
- ✓ Authentication Service

Everyplace Server Service Provider Offering

Base Offering

- ✓ Tivoli Personalized Services Manager
- ✓ Everyplace Location Services
- ✓ Tivoli Policy Director
- ✓ Everyplace Intelligent Notification
- ✓ WebSphere Transcoding Publisher
- ✓ MQSeries Everyplace
- ✓ WebSphere Edge Server
- ✓ Everyplace Authentication
- ✓ Everyplace Setup/Suite Manager
- ✓ SecureWay Directory
- ✓ DB2 UDB Enterprise Edition (private)
- ✓ WebSphere Application Server (private)
- ✓ IBM HTTP Server (private)

Options

- ✓ Everyplace Synchronization Manager
- ✓ Everyplace Wireless Gateway

Prereqs

- ✓ AIX 4.3.3 or higher or Solaris 2.7 or 2.8
- ✓ Netscape Communicator V4.7 or Netscape Navigator V4.08

Order Separately

- ✓ WebSphere Voice Server
- ✓ Location Services Provider (SignalSoft)
- ✓ Lotus Domino Everyplace
- ✓ WebSphere Translation Server
- ✓ Sametime Everyplace

Figure 1-16 Everyplace Server Offerings components and options

1.3.2 IBM WebSphere Everyplace Server Enable Offering

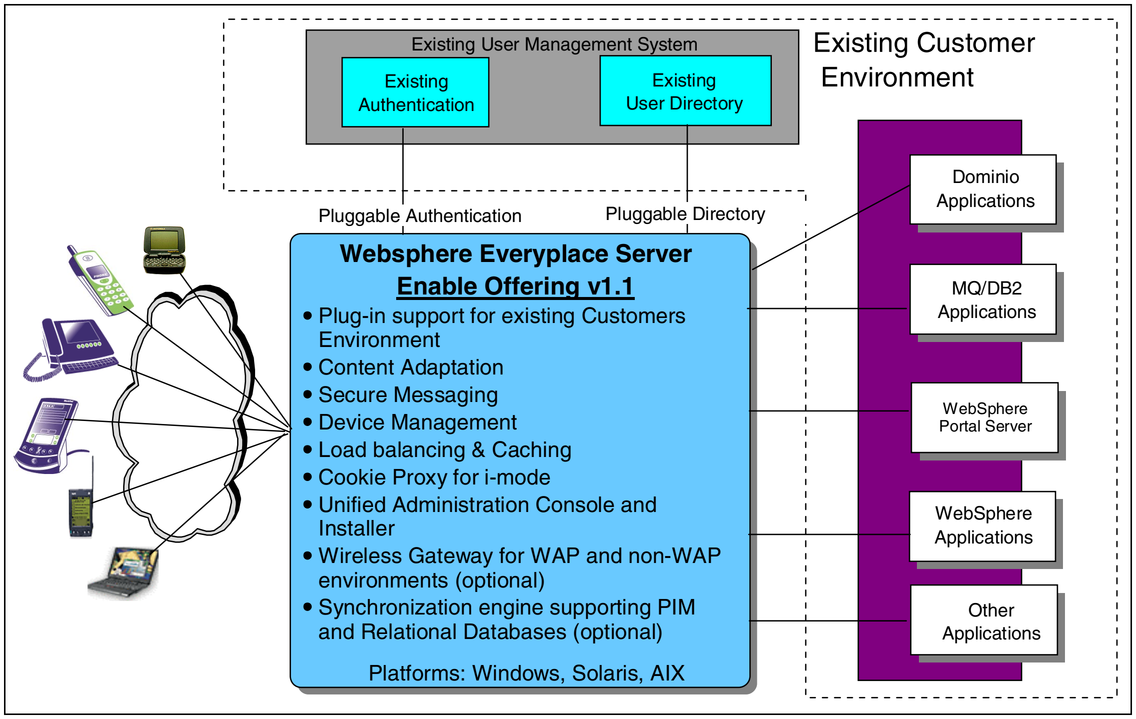
WebSphere Everyplace Server Enable Offering (IBM WebSphere Everyplace Server EO) is a new product which recognizes that many enterprises need a more targeted package which will fit flexibly in their enterprise infrastructure and provide enablement functions to their existing systems.

What to do with IBM WebSphere Everyplace Server - EO

The primary purpose of IBM WebSphere Everyplace Server EO, therefore, is to enable enterprises to extend the reach of their internal systems over mobile communications.

Everyplace Server Enable V1.1 enables enterprises to introduce their business into the wireless environment while preserving and extending existing e-business investments, with:

- Secure communications (wireless and wireline)
- Synchronization between servers and pervasive devices
- Management of pervasive devices
- Content adaptation to devices, networks and users
- Transaction messaging for secure and guaranteed delivery
- Integration with the enterprise's current environment
- Support for established LDAP and Authentication servers
- Client support:
  - Windows CE devices
  - IBM ThinkPads and compatible laptops
  - Palm OS devices
  - IBM Workpads
  - WAP phones
  - i-mode phones



Existing User Management System
Existing Authentication
Existing User Directory
Existing Customer Environment
Pluggable Authentication
Pluggable Directory
Websphere Everyplace Server
Enable Offering v1.1
• Plug-in support for existing Customers Environment
• Content Adaptation
• Secure Messaging
• Device Management
• Load balancing & Caching
• Cookie Proxy for i-mode
• Unified Administration Console and Installer
• Wireless Gateway for WAP and non-WAP environments (optional)
• Synchronization engine supporting PIM and Relational Databases (optional)
Platforms: Windows, Solaris, AIX
Dominio Applications
MQ/DB2 Applications
WebSphere Portal Server
WebSphere Applications
Other Applications

Figure 1-17 Going Mobile with IBM WebSphere Everyplace Server Enable Offering

Figure 1-18 shows a typical layout of an enterprise IBM WebSphere Everyplace Server EO deployment. Most components are installed in the enterprise secure zone (Trusted Network). Coordination with existing enterprise components is supported by EJBs.

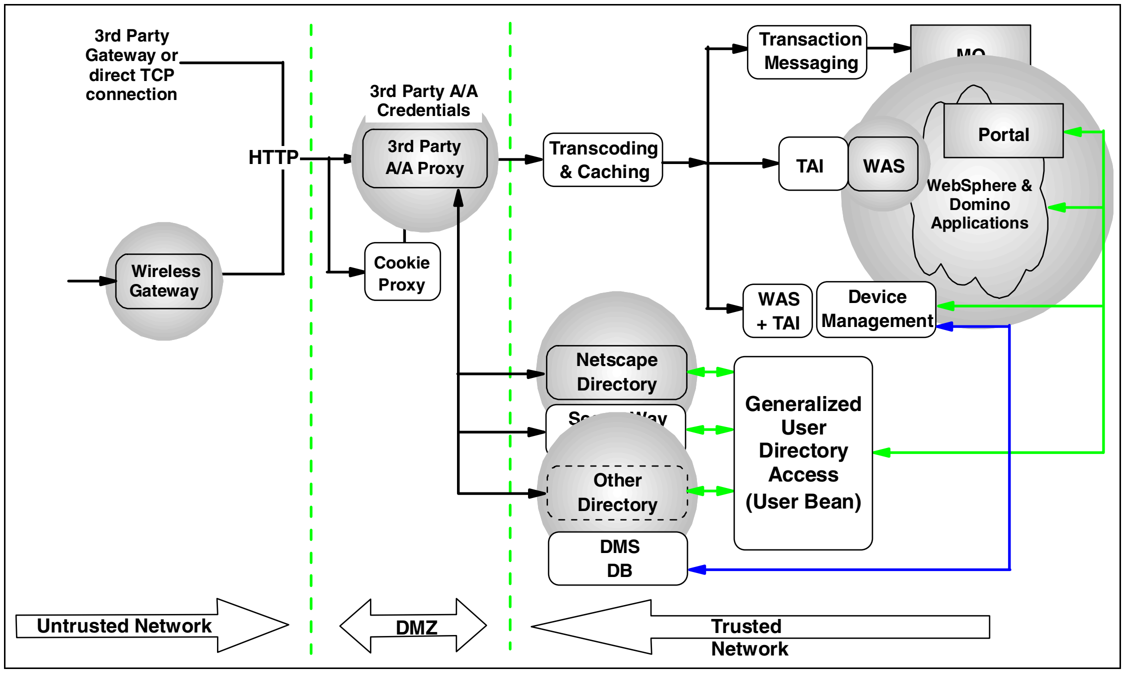


Figure 1-18 Enterprise deployment of Enable Offering

1.3.3 IBM WebSphere Everyplace Server Service Provider Offering

WebSphere Everyplace Server Service Provider Offering (IBM WebSphere Everyplace Server SPO) continues the line of development started with WebSphere Everyplace Suite Version 1.1.x. Figure 1-19 on page 53 summarizes the composition of IBM WebSphere Everyplace Server SPO, its typical physical layout and the relationships among its components, as reviewed in detail in Section 1.2, “Mapping functions to products” on page 17.

When IBM WebSphere Everyplace Server - SPO is required

The Service Provider Offering is needed when deploying new kinds of services, as distinct from, or in addition to, extending the reach of existing ones. New functionality includes intelligent notification, location-based functions, messaging and synchronizing.

SPO also offers the full range of subscriber management and personalization. These capabilities are not normally required for enterprise installations, where applications are primarily B2E, or collaborative.

IBM WebSphere Everyplace Server, Service Provider Offering addresses the requirements of the following industry segments:

– Wireless Service Provider
– Large Financial enterprise
– Large Retail enterprises
– Large Travel and Transportation enterprises

The base offering provides all the components required to work with the customer's existing connectivity software. If such facilities are not already deployed, the customer can add Everyplace Wireless Gateway or Everyplace Synchronization Manager. Figure 1-19 illustrates a typical IBM WebSphere Everyplace Server Service Provider Offering deployment.

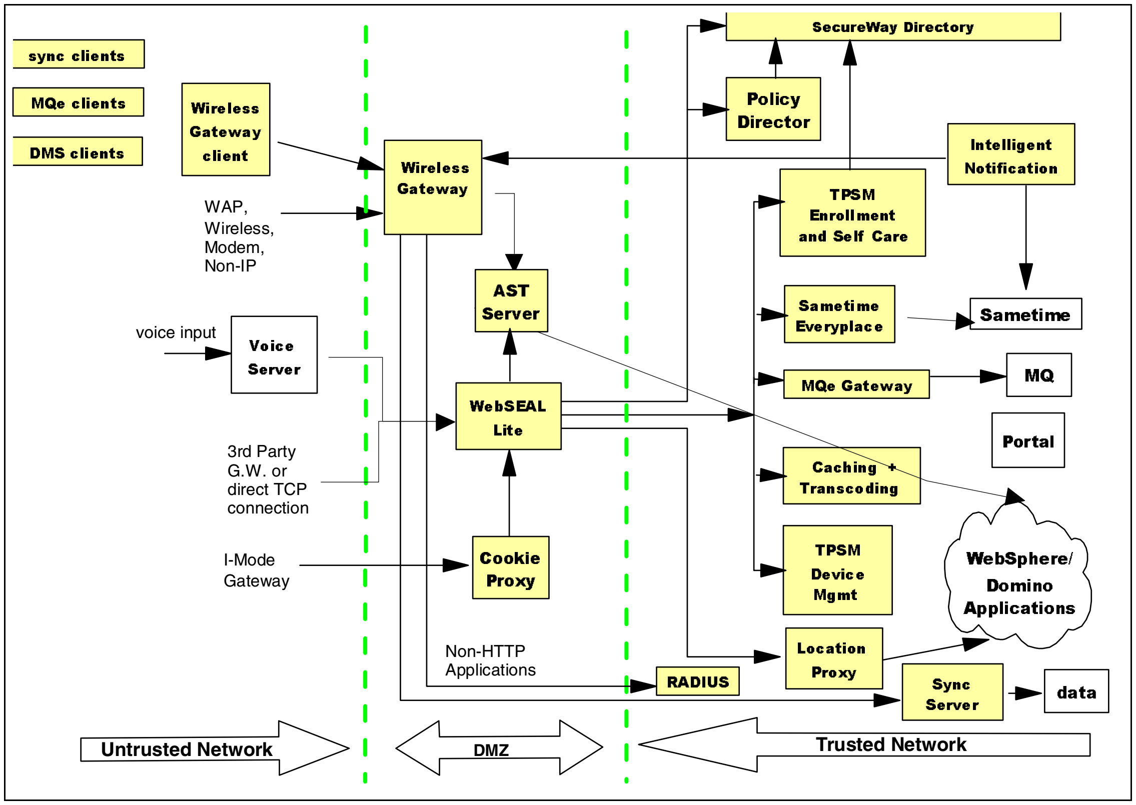


Figure 1-19 Deploying the Service Provider Offering

2

Application architecture

This chapter presents a description of the insertion of an application in the WebSphere Everyplace Server environment. Available options are discussed, ranging from an application specifically developed for pervasive use to the more common case of existing applications being adapted to pervasive usage requirements. In many cases, including that of the sample application described in this redbook, a combination of approaches will be needed for best results.

In the case of adaptation, consideration must be given to several functions of WebSphere Everyplace Server, including user and device information, security and policy issues, content adaptation, protocol management and quality of service.

2.1 Planning techniques

The previous chapter outlined a view of WebSphere Everyplace Server as an implementation of enablement functions, that is, of all functions which can be factored out of individual Web applications and separately implemented in a robust and scalable way.

This perspective enables us to position WebSphere Everyplace Server with respect to the process of planning, designing and implementing such applications.

We will begin by reviewing the various *computing modes* encountered in pervasive applications. We will then proceed to consider the use of patterns in system planning, and finally introduce the decision tree as a way of documenting available architectural and design choices.

2.1.1 Modes of pervasive computing

A computing mode is a schema of use of computing resources by actors (users or other computing systems). It is characterized primarily by the topology of communication (that is, the type and timing of communications used). In the context of relevance to WebSphere Everyplace Server, we can distinguish four main modes.

Synchronous mode

In the synchronous mode, the participants are available simultaneously, and their interaction takes place within one communication session. One example is that of a user accessing a standard Web site through a browser (assuming no interruption of the traffic through failure; if such an interruption occurs, the interaction has to start afresh).

Asynchronous mode

In the asynchronous mode, some mechanism is available to deliver the content, submitted by one participant A at a given time, to another participant B at another time. A common example is a mail system. Characteristically, participant B has to initiate the retrieval of the content. A generalization of this action, called *synchronization*, occurs when the asynchronous participants are separate components of a common application, for example, when a user synchronizes the PIM on his/her PDA with another instance running on his/her desktop.

Push mode
Instant messaging systems, for instance, are *instant* because the originator of the message can ensure that the receiving device (assuming it is in operation) receives notification of a relevant event (in this case the need for a correspondent to transmit information). Generally speaking, notification systems allow the originator to *push* the event to the destination.

Voice mode
The use of voice as the communication channel constrains the topology in a special way. In terms of timing, such systems can be synchronous, asynchronous push, or any combination of these. What distinguishes them is the unique set of capabilities and limitations of the voice channel. For this reason, they are grouped together in their own mode.

The idiosyncratic qualities of voice applications have already been hinted at in "Adapting content" on page 11. A detailed treatment can be found in Redpiece SG24-6259, *Designing and Developing Mobile Applications using WebSphere Everyplace Access V1R1*. See in particular Chapter 3 (on speech technology in general) and Section 8.4 (on the issues of designing for voice).

2.1.2 Patterns

"Patterns are supposed to be sewn together to solve a problem." (Alan Shalloway)

The word *patterns* was introduced into software engineering parlance by the famous "Gang of Four" book in 1995 (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns*,1995, Addison-Wesley). The authors took inspiration from the work of an architect, and it only seemed fair that the idea should be applied to software architecture. In fact, illustrating patterns in design was much more straightforward, because informed readers could map the patterns to their own implementations in code, and learn the technique of abstraction in that manner.

A pattern is a pairing of a software engineering problem with a proposed schema of solution. It is stated in a standard way, including the conflicting requirements and constraints making up the problem, and the elements of the solution, with the ways in which they answer these requirements and constraints.

A pattern is an abstraction of actual (best) practice. Like mathematical abstractions, it offers the possibility of transferring solution approaches and ideas from one domain to another once the common structures of problems have been recognized. Good engineers have long practiced this conceptual discipline, but the virtue of the "GoF" book is that the mental gymnastics involved was now formulated explicitly, and thereby externalized.

Immediately after the publication of the GoF book, experts in various parts of software engineering (Architecture, Analysis, Deployment,etc.) started formulating their understanding using the pattern idiom. Pattern language is now an established part of software engineering, together with the opposite *anti-patterns*, in which the idiom is reversed to express warning about bad practice.

Together, patterns and anti-patterns embody a way of exchanging ideas in a concise way. They provide a standard set of references for application development, promote precise understanding within Business Analysis, Architecture, Design, Implementation and Deployment, help to decrease development time, and simplify documentation and maintenance.

With the pattern movement firmly established, hindsight showed that many good engineering practices were in fact patterns in all but description: Layered Architecture models (for example, OSI-ISO), Pipe and Filter Architecture (Unix), Model-View_Controller (Xerox Parc/Smalltalk), Blackboard (AI systems), Publish-Subscribe, Message brokers, Proxys, etc.

This harvesting of previous data continues in the area of Business Analysis and BPR: many business practitioners have an intuitive knowledge of what works and this knowledge is being cast into the pattern idiom. The application types identified for e-Business (B2E, B2C, B2B) are broad generalizations of such patterns. On a smaller scale, J2EE practice identifies presentation, business, and integration patterns.

In the same vein, we now proceed to identify an emerging pattern which is implicit in the architecture and design of most pervasive applications.

Application patterns for mobile use

There are so far few, if any, mobile applications the structure of which cannot be seen as an extension of a standard application (for example, one designed with no mobile use in mind). We may be confident that such applications will appear, and they will exhibit their own patterns.

Currently, patterns for mobile business applications can be described as extensions of the well-known e-business applications (B2C, B2B, B2E, and so on).

Architecture pattern

The pattern statement runs as follows(a reworking of the sequence presented in “A functional approach to understanding IBM WebSphere Everyplace Server” on page 4):

- Problem statement:
  - A standard Web application assumes HTTP transport, with standard capabilities in the client (HTTP browser) and standard connection characteristics.
  - New categories of client devices are appearing, which do not share these assumed facilities.
  - It is imperative (for social or economic reasons) that business take into account and serve these new devices, and the new needs arising from their presence.
  - Re-engineering the established base of applications is not feasible (because of prohibitive cost and moving target of constantly evolving technology).
- Solution:
  - Reuse existing application base.
  - Factor out (as much as possible) all functional discrepancies between the standard model of Web applications and the variety of new functional requirements.
  - Implement the resolution of each of these discrepancies as individual modules which can be combined to enable the old application for the new contexts.

Design pattern

To implement this idea, we must find a way of inserting this enabler in the application flow.

Described in the Model-View-Controller pattern, a Web application consists of a controller/view cluster relating to a model. For instance, in a servlet-based implementation, the servlets embody the Controller and the pages, static or JSP-generated, embody the View. These are tightly coupled, while the model is embodied in Data Access Beans or EJBs which tend to be quite separate, and indeed serve more than one application.

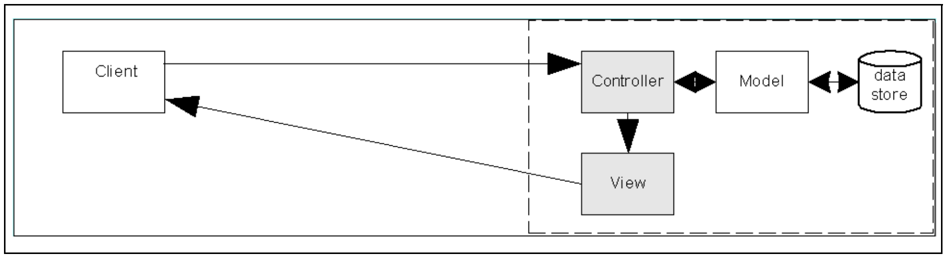


Figure 2-1 A non-pervasive application

The most common extension to the pervasive context consists of adding one or more new component(s) to modify the application's Controller and View. The model part of the application is, on the whole, independent of the mobile, or pervasive conditions of use.

A simple example of this approach is implemented by WebSphere Transcoding Publisher: as a proxy or servlet it intercepts the request, possibly modifying or adding some headers. It then intercepts the response, edits or clips the content, rewrites URLs as needed, and translates the markup notation. Incidentally, we note that WebSphere Transcoding Publisher operates by implementing a specialization, another pattern, Web Intermediaries, which can be described as extending to the Web the well-established Pipe-and-Filter pattern introduced by Unix.

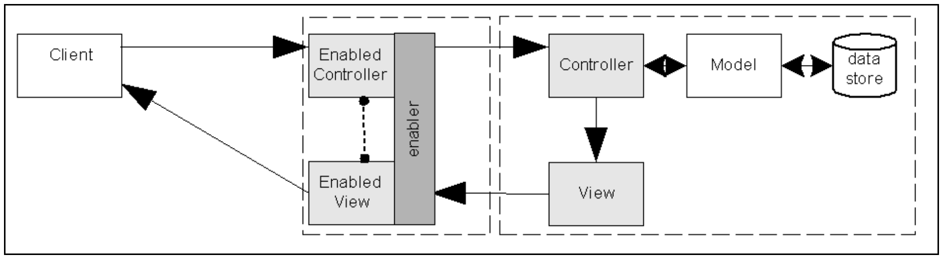


Figure 2-2 Application for pervasive use

In Figure 2-2, the dotted line connecting Enabled Controller with Enabled View is intended to suggest that these two elements of the design are rather more tightly coupled than the Controller and the View. Typically the Controller drives the View (for example, by selecting JSPs according to the current state). On the other hand, the dependency between enabled elements is bidirectional. Examples will appear in the scenarios covered in the following chapters, for instance, a servlet may have to set HTTP headers to supply information needed by an annotator.

While the most obvious application of this idea is in enabling content for different presentation formats or media, we find it also applies elsewhere. As discussed in Section 1.1, “A functional approach to understanding IBM WebSphere Everyplace Server” on page 4, the introduction of pervasive devices, which do not follow the standard HTTP over IP paradigm, requires enabling transport for different protocols, security services and content packaging (mostly rendering, but also structure).

Patterns for e-business

The IBM methodology has been making use of the concept and techniques of patterns (Business, Architecture, Design patterns) for the past several years. It has introduced a variant of the concept of patterns discussed above, proposing the flourishing planning technique known as Patterns for e-business: these have emerged as a powerful tool for the very demanding tasks of fast e-business development.

Patterns for e-business are not exactly patterns in the original sense. Original patterns focus on the dialectic of problem-solving, whereas patterns for e-business are more like static solution templates.

However, these patterns for e-business must not be considered in isolation. They are intended to be part of an engineering process (the IBM method). A great deal of the dialectic is contained in the process itself, with the templates serving as sign-posts along the way.

This process is presented in a practical way on the following site:

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

which offers the dynamic (process) dimension not available in this book. We urge you to experiment with it.

On the subject of mobile applications specifically, an excellent source of ideas and detailed information is available as Redbook SG24-6259, *Designing and Developing Mobile Applications using WebSphere Everyplace Access V1R1*.

Since the topic is thoroughly covered in that publication, we encourage you to make full use of it, and shall be content here to summarize some salient points.

Figure 2-3 summarizes both the structure and the process.

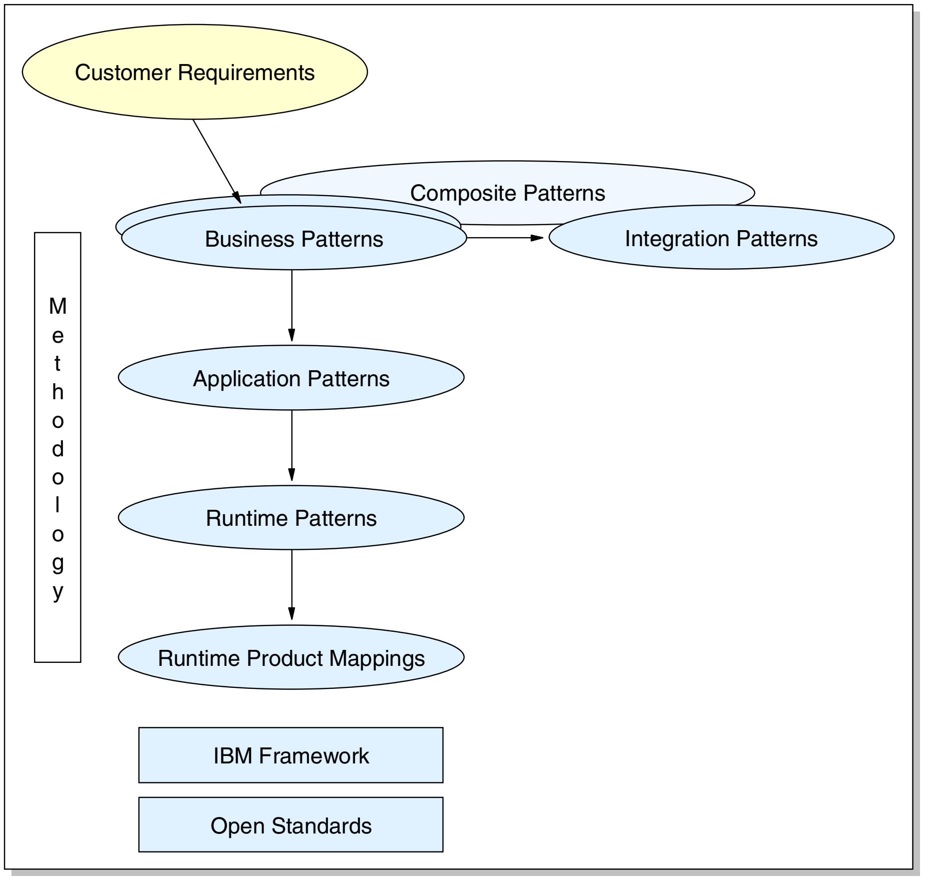


Figure 2-3 Patterns for e-business

The Patterns for e-business approach consists in a sequence of choices (guided by certain principles and constraints akin to the Gof patterns) over several layers of relevance. Each choice constrains the subsequent ones, as indicated by the arrows. The overall process is part of, and enacts, a specific project methodology.

The decision layers are as follows:

– Business patterns: identify the interactions between users, businesses and data.

– Integration patterns: tie together multiple Business patterns when a single Business pattern is insufficient to provide a solution.

- Composite patterns: represent commonly occurring combinations of Business patterns and Integration patterns.
- Application patterns: describe interactions of components and data within a business solution.
- Runtime patterns: define the logical middleware structure supporting an Application pattern.

Two sets of best practice recommendations complement these decision layers:

- Runtime product mappings: identify tested, optimal implementations for each Runtime pattern.
- Best practice guidelines: for design, development, deployment and management.

Patterns for e-business in pervasive computing

We have shown how WebSphere Everyplace Server can be seen as a toolkit for the construction and deployment of Enabler modules implementing the Enabler architecture pattern.

Patterns for e-business cover this type of functionality, and the decision process associated with it, in the Access Integration pattern. Its use is described and discussed in detail in SG24-6267, *Access Integration Pattern Using WebSphere Portal Server*; you will find there a list of the following services making up the Access Integration Patterns:

- Presentation
- Personalization
- Security and Administration
- Pervasive Device Support

Presentation covers what we called Content Enablers, *Security and Administration* corresponds to Security Enablers, and *Pervasive Device Support* matches Transport Enablers. Personalization makes use of both Security and Content enablement.

Although primarily concerned with the Portal Server, the discussion is applicable here as well, particularly for:

- Application Patterns (see Figure 2-4 on page 64):
  - Pervasive Device Access (A)
  - Single sign-on (B)
  - Personalized delivery (C)

  Pervasive Device Access and Single sign-on show the exact topology of the Enabler pattern, with an intermediate tier assuming the required functions.

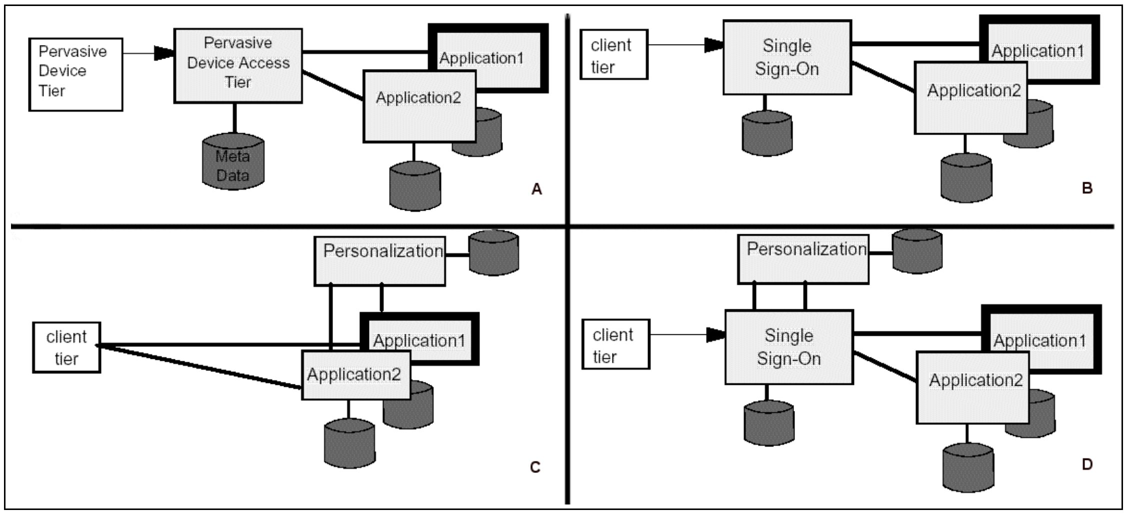


Figure 2-4 Application patterns for pervasive computing

Personalized delivery shows a different topology: here it has been thought that personalization should be different for each distinct application, so the personalization module is not placed as an intermediary between application and presentation, but rather as an adjunct to the applications themselves. This can be seen as similar to the approach described below (see “Design decision tree” on page 65) as the Direct rendering choice.

Another approach is possible, however (D). If sign-on is placed at the edge (becoming *single sign-on*), then the acquisition of user profiles can be coupled with it, so that the information needed for personalization is factored out of each application. In that case, Personalization becomes another variant of the Enabler pattern.

– Runtime patterns

  Runtime patterns are more numerous, as they must cover the diversity of implementations for each of the enablements. They also must take into account the topology of deployment of these components. This is commonly represented by the well-known three-part diagram showing the trusted network, the trusted network and the intermediate DMZ. "The IBM WebSphere Everyplace Server Offerings" on page 47 shows two instances of this diagram, one for the Enable offering and one for the Service Provider offering.

2.1.3 Design decision tree

Adding separate enablers to an existing setup is only the most immediate implementation of the Enabler idea. One may choose instead to rewrite the application to incorporate new, pervasive-adapted parts.

The design decision tree embodies a variant of the classical "make or buy" decision-making process. When planning an application or a whole system, the first step is to examine available materials or ready components, and decide whether to make use of existing resources or create new ones.

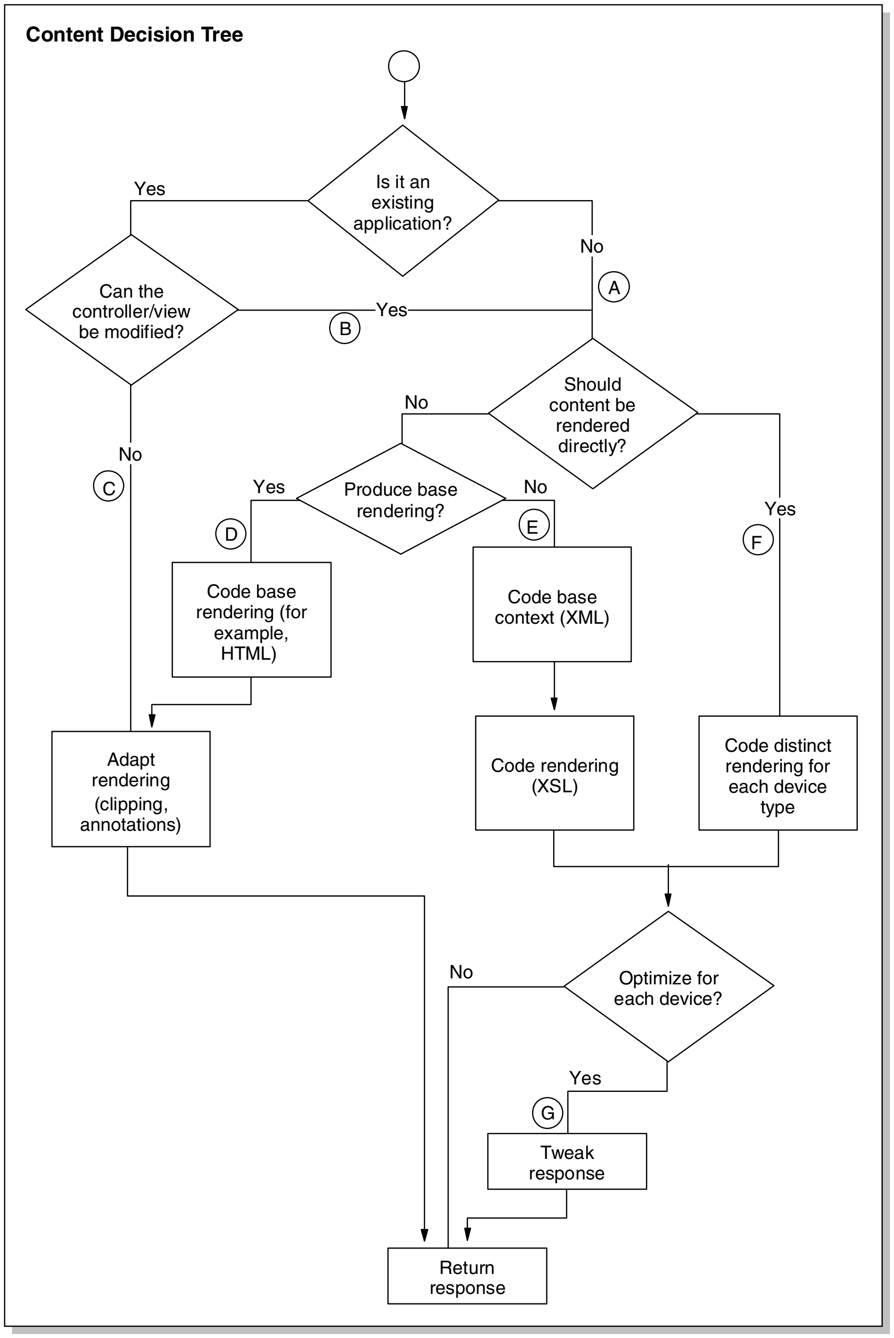

Content Decision Tree
Is it an existing application?
Yes
No
A
Can the controller/view be modified?
Yes
B
Should content be rendered directly?
No
No
C
Produce base rendering?
Yes
No
Yes
D
E
F
Code base rendering (for example, HTML)
Code base context (XML)
Adapt rendering (clipping, annotations)
Code rendering (XSL)
Code distinct rendering for each device type
Optimize for each device?
No
Yes
G
Tweak response
Return response

Figure 2-5 Content decision tree

The decision tree shown in Figure 2-5 on page 66 applies to the processing of content within the application. We assume, as discussed in the previous section, that the back-end is the subject of another decision, inasmuch as its design is not affected by the decision to “go pervasive”.

If the application is a new one (A), or if we have access to the front-end code (B) (that is, for instance, the servlets and JSPs making up the Controller and the View), we have complete freedom of design. On the other hand, if we do not have the possibility of modifying the code (C), we must proceed by adapting the rendering produced by the existing application. This may also be the case if we decide (D) to target a specific kind of device (for example, HTML browsers) and to derive the other renderings from the basic rendering, using transcoding.

We may choose instead (E) to produce rendering-neutral content, using XML, and then map the content using distinct XSL stylesheets for each device type. There is also the option (F) of coding the renderings directly, separately for each device type. Finally, for the devices within a single device class, we may want (G) to take advantage of individual characteristics to optimize the rendering in each (tweak response).

A decision tree could be constructed to document the design decisions on Security discussed in “Adapting security in an existing application” on page 75.

2.2 Centralized services offered by WebSphere Everyplace Server

Considered from the point of view of pervasive application design and development, WebSphere Everyplace Server can be treated as a toolkit supporting the implementation of various variants of Access Integration.

2.2.1 Synchronous Mode Applications

Transport Enabler

Everyplace Wireless Gateway

Delivery to mobile devices caters to the transport protocols specific to each device type. The Everyplace Wireless Gateway provides this service transparently, making the devices appear as IP devices to the rest of the system.

Security Enablers

Everyplace Wireless Gateway

The Everyplace Wireless Gateway offers:

- Authentication:
  - Optional RADIUS authentication for WAP clients.
  - WAP Gateway authentication for users registered in the LDAP directory.
- Encryption:
  - Wireless Transport Layer Security (WTLS) between the Gateway and the WAP client.
  - Secure Sockets Layer (SSL) between the Gateway and Web servers.

WebSEAL-Lite

WebSEAL-Lite is the central security manager for WebSphere Everyplace Server. For each request, the user identity is first looked up against the WSL cache, then (if not found) against the Active Session Table to retrieve those users identified by the Wireless Gateway. If the user is still unknown, a challenge is issued.

When WebSEAL-Lite operates as authentication proxy, it holds a monopoly over authentication in the WebSphere Everyplace Server domain. Any application or application server must relinquish authentication to it. This may have implications for the adaptation of existing applications.

WebSEAL-Lite with Policy Director

If the enterprise already uses Policy Director to manage its access policies, it makes sense to consider interfacing WebSphere Everyplace Server to it.

Content Enablers

WebSphere Transcoding Publisher

Content adaptation in all its forms (Header manipulation, Clipping and Editing, Annotation, Markup Language translation, and caching sensitive to content adaptation) is the sole function of WebSphere Transcoding Publisher.

Location-Based Services Proxy

The Location-Based Services Proxy maps identity headers to geographic information headers.

2.2.2 Asynchronous mode applications

Asynchronous mode application types supported by WebSphere Everyplace Server include:

- MQSeries Everyplace
- Everyplace Synchronization Manager

Transport Enablers

Everyplace Wireless Gateway

MQSeries Everyplace and Everyplace Synchronization Manager use the Wireless Client to connect to the WebSphere Everyplace Server environment.

Security Enablers

WebSEAL-Lite

MQSeries Everyplace, when operating over HTTP, can use the standard WebSphere Everyplace Server security mechanisms.

Other

MQSeries Everyplace (over TCP) and Everyplace Synchronization Manager have their own security provisions.

Content Enablers

MQSeries message formats and contents are managed by the applications.

Everyplace Synchronization Manager content is dependent on the applications being synchronized.

2.2.3 Push mode applications

Push mode application types supported by WebSphere Everyplace Server include:

- Sametime Everyplace (STEP): as a stand-alone product, STEP manages its own push facility.

  Note: Sametime Everyplace is offered in WebSphere Everyplace Server version 2.1.x as a Technology Preview. No integration is currently available.

- Intelligent Notification Services: the last stage of the process involves the Unified Notification Dispatcher, which wraps the Push API.
- Direct use of the Push API.

Transport Enablers

Everyplace Wireless Gateway

The Everyplace Wireless Gateway offers the following Push Notification support:

- SMS
- Network Operator interfaces (UCP, MPP, SMTP, SNPP)
- WAP 1.2 Push
  - WAP Push Proxy Gateway on the initiator side
  - WAP bearers and SMS bearers on the receiver side (Mobile clients)

The Everyplace Wireless Gateway supports mobile push initiators.

Security Enablers

The facilities described under Synchronous Mode are also available in Push mode.

Content Enablers

There are no content Enablers for the Push mode in the current release. Push initiators must format their messages and specify the protocol used to the Push Proxy Gateway.

2.2.4 Voice mode applications

As observed earlier, Voice Mode is distinguished not by its topology (symmetric/asymmetric, session/no session), but by the special character of its channel. For this reason, it has special requirements for transport and for content structuring.

Transport Enabler: Voice Server/VoIP Gateway

The Voice Server is responsible for the conversion between VXML/HTTP and Voice over IP (VoIP).

The VoIP Gateway (for example, Direct Talk or Cisco) is responsible for the conversion between the Switched Telephone Network and VoIP.

Security Enabler: WebSEAL-Lite

Security is supported by WebSEAL-Lite, with the constraint that the user ID and password must be DTMF-compatible.

Content Enabler: WebSphere Transcoding Publisher

WebSphere Transcoding Publisher ensures translation from HTML to VXML or (using stylesheets) from XML to VXML.

2.3 Planning a new application in WebSphere Everyplace Server context

2.3.1 Deployment considerations

WebSphere Everyplace Server is a complex product, or set of products, which can be deployed in many different configurations. These are dealt with in a companion redbook, *Everyplace Server: Guide for Architects and Integrators,* SG24-6189.

Regardless of the specific configuration, two sets of issues arise:

Application versus enablement

Although the distinction between enablement and application is conceptually clear, their implementations sometimes overlap, and their deployment almost inevitably does.

This is because enabler components are architected around some of the same products as applications, and themselves often include special-purpose applications. For instance:

- TPSM uses WebSphere Application Server (WAS), which of course is at the core of most Web applications, and itself uses DB2 and an LDAP Server.
- Most WebSphere Everyplace Server components depend on LDAP, both for their administration and for their operation. Some applications, in turn, may access some of the same data in LDAP.
- Intelligent Notification requires DB2 and WebSphere Application Server, and it is not always clear where the boundary lies between enablement infrastructure and application.

It will therefore be very important to take a detailed inventory of the functions to be assumed by these essential components, and to design their physical deployment with great care, as they may become determinant in the performance of the overall system.

Plug-in deployment

WebSEAL-Lite (WSL), WebSphere Transcoding Publisher (WTP), and, to a lesser degree, Location-Based Service (LBS), are core components of enablement. It is likely that they will be required together in a given deployment. As all three are deployed as plug-ins to Web Traffic Express (WTE), an option in the case of WebSphere Transcoding Publisher, the question arises of their relative positioning. It seems, though this has not been fully confirmed, that they cannot be plug-ins to the same WTE process or machine. If they were, it is not clear how their order of precedence could be established.

The simplest approach to deploying them is to sequence them as follows: WebSEAL-Lite->WebSphere Transcoding PUblisher->Location-Based Services, from the outside of the site inwards. This arrangement allows WSL to perform its security functions. Requests identified as requiring transcoding can be routed to WebSphere Transcoding Publisher. From either WSL or WebSphere Transcoding Publisher, location-dependent requests can then go to LBS. Those requests which do not need WebSphere Transcoding Publisher or LBS can bypass them.

How the routing and/or bypass is configured also depends on the forward proxy/reverse proxy configuration of each component.

2.3.2 Synchronous mode

When planning a new synchronous application to be used with WebSphere Everyplace Server, we have complete freedom regarding the handling of presentation. The choice is much reduced for the other modes, in the current state of the offerings. As integration progresses, the centralized services will free asynchronous and push applications from many of the common concerns (transport, security, content format).

Content/delivery options

In the content decision tree, we can choose paths D, E, or F.

- Separate content from rendering (E):

  The most generic solution is to produce the content in the form of XML documents, using one or several DTDs. Any number of XSL stylesheets can then be designed, and applied by WebSphere Transcoding Publisher, to suit variable conditions of delivery.

- Design a base rendering for the major delivery (D):

  If the site is to be used primarily in one delivery mode (for example, if most of the users access through an HTML browser), it makes sense to produce well-formed HTML directly, to save on rendering costs. Other delivery modes can be served using annotations, clipping, or both.

- Generate multiple versions(F):

  It is possible to combine the previous two options by having a generic version for uncommon delivery modes, and dedicated versions for heavy usage (possibly more than one mode). The extreme case would be for the application to generate a distinct document for each delivery mode, bypassing the benefits of WebSphere Everyplace Server adaptation altogether.

Security considerations

Because WebSphere Everyplace Server offers a complete security solution, we may want to delegate all security management to the WebSphere Everyplace Server services.

The options available are:

- Use default security configuration (WebSEAL-Lite with SecureWay Directory).
- Use Policy Director for authorization.

  This option must be chosen if the application includes Location-Based Services, or if we wish to impose a fine-grained authorization control.
- Use RADIUS authentication.

  This option is available from the Wireless Gateway.

2.3.3 Asynchronous mode

Planning an asynchronous application with Sametime Everyplace or Everyplace Synchronization Manager does not currently require taking any special account of the WebSphere Everyplace Server context.

In the case of MQSeries Everyplace, the HTTP mode integrates with the standard path through WebSEAL-Lite. A dedicated redbook on MQSeries Everyplace is in preparation.

2.4 Adapting an existing application

Existing Web applications to be adapted to pervasive operation are of the synchronous type. Content Adaptation is handled by WebSphere Transcoding Publisher.

2.4.1 Transcoding challenges

The major difficulties in adapting content in an existing application arise from the artistic creativity of Web designers. Existing Web page design tools offer a rich array of possibilities, which translate behind the scenes into technical complexity in the page code.

- Frames

  Frames are not supported in most pervasive devices. This means that the navigation of the application must be altered to visit the relevant frames in sequence as separate pages, and to ignore the purely decorative frames.

  If you do not have read access to the page source, you may have an additional difficulty. Browsers will typically display only the source for the frameset, while the content to be processed is to be found in the daughter frames. One way to solve this problem is to use the WebSphere Transcoding Publisher trace in High mode, so that the input page is recorded in full.

- Client-side scripting

  JavaScript does not work in WML browsers. WMLScript does not have all the capabilities of JavaScript, and, at any rate, translating one into the other is a difficult task. When the behavior of a page is controlled by client-side scripting, the solution may be to change the navigation by inserting intermediate pages, which can often be coded statically in WML.

- Composition by tables

  Tables are almost universally used to effect the layout of large HTML displays. Unfortunately, this layout does not carry over to WML browsers, even if the device implements table support. Most often, the best approach is to linearize the layout, if necessary using multiple cards.

- Navigation by images

  Images are generally difficult to handle, because of bandwidth in transmission and real estate in display. WebSphere Transcoding Publisher offers the option of skipping images, or replacing them with links. One particular case is that of navigation links rendered as images: this is not supported in WML, and must be replaced with plain links.

- Decorations

  Some illustrations are instrumental on a Web page, but most are purely decorative. Common sense must be applied in dealing with those.

- Large content items

  WebSphere Transcoding Publisher has a fragmentation engine which can divide content into manageable chunks for delivery to a device (maximum chunk size is a property of the device). The fragmentation engine can be defeated by oversized paragraphs. Furthermore, users of mobile devices tend to want access to essential information. For these reasons, it is often useful to reduce the text of some items to a simple summary.

- Conditional response pages

  Annotators are applied to pages under conditions specified at registration. However, conditions are formulated in terms of URL and headers only: it is not possible to specify a condition based on the content of the page. When accessing a servlet which responds with different JSPs or static pages in different conditions, the request URL is the same, and cannot be used to distinguish among the responses. If you are in a position to modify the servlet code, you can make it set a header for this purpose. Otherwise, you cannot use an annotator, and must write a clipper which can make a distinction based on the content.

2.4.2 Adapting security in an existing application

In an integrated environment, there are major advantages to a centralized approach to security (better control, single sign-on, deployment of more sophisticated security solutions, etc.).

When incorporating existing applications into such an environment, there may be conflicts between security implementations. Below are outlined three possible approaches. For a detailed coverage of security issues in WebSphere Everyplace Server deployment, consult the IBM redbook *WebSphere Everyplace Server A Guide for Architects and Systems Integrators*, SG24-6189.

Keeping security in the application

If the application has a good security implementation, you may want to keep it, at least for a first phase. WebSEAL-Lite can be deployed in Transparent Authentication Proxy mode, which allows your application server to maintain its own authentication.

Migrating security to centralized service
Alternately, you may have a sophisticated authorization service deployed in WebSphere Everyplace Server (for example, using Policy Director). WebSEAL-Lite can work with it. In this case, you will want to provision all authentication and authorization centrally, and strip it out of your existing application.

Sharing security
Finally, WebSphere Everyplace Server and the application can share security responsibilities. This would happen if your authorization decisions are complex and require application-level logic. For this situation, WebSEAL-Lite supports LTPA. The Application server will retain its authorization function, based on central authentication using LTPA.

3

Enterprise sample applications

This chapter presents the sample applications chosen to illustrate the various capabilities of WebSphere Everyplace Server, as well as the techniques and tools available for WebSphere Everyplace Server-oriented development.

We start with an existing sample B2E application, and extend it in two directions:

- Making the application available to diverse clients in several contexts.
- Adding functionality not readily supported outside of a pervasive computing environment.

3.1 Web Application models for business

Web-based business applications fall into several categories according to the nature of the interactions they support, and of the participants in those interactions.

The first distinction separates applications supporting the interactions of two or more businesses (B2B) from those involving individuals.

The B2B application model is based on the Extended enterprise pattern. In the past, it used EDI, founded on two party agreements. With the spreading of XML, data exchanges between businesses may become standardized to a greater extent. At present, there does not seem to be much incentive for adopting a pervasive approach in this type of application.

Among the applications involving individuals, we distinguish between applications addressing individuals with a specific tie to the business (B2E) and those addressing the general public (B2C).

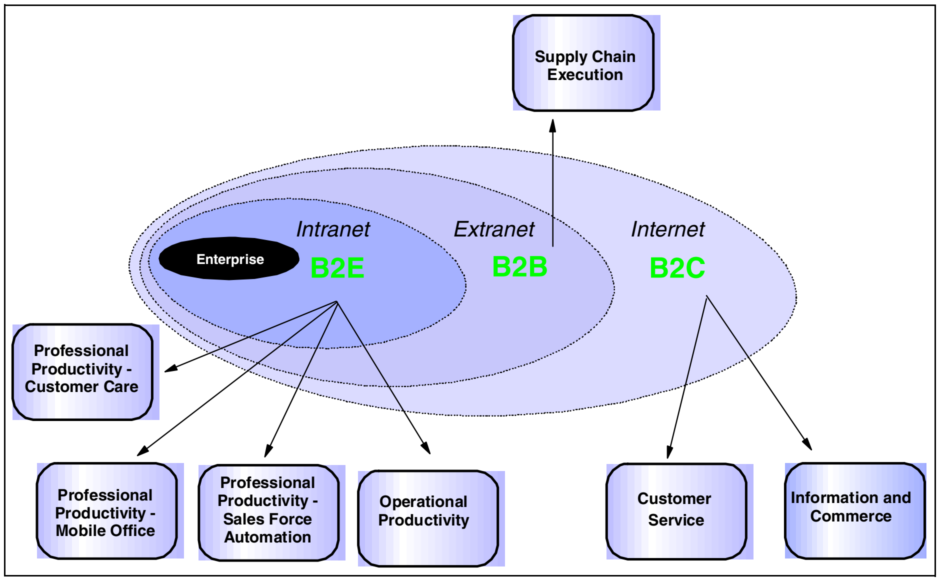

Supply Chain Execution
Intranet
Extranet
Internet
Enterprise
B2E
B2B
B2C
Professional Productivity - Customer Care
Professional Productivity - Mobile Office
Professional Productivity - Sales Force Automation
Operational Productivity
Customer Service
Information and Commerce

Figure 3-1 Business application models

The label B2C illustrates the consideration that persons not specifically related to a business will most likely address the business’s Web-face as consumers of the business’s goods or services. Typically, such applications will be based on the Self-service business pattern. Also, since the application is generally unrestricted, it must cater to a wide variety of client devices and personal preferences. As long as the delivery is restricted to the IP network, and the content to HTTP and basic MIME types, thoughtful programming of the presentation (well-formed HTML, avoidance of browser idiosyncrasies) can deal with the variety. However, migrating such applications to the pervasive context may present very real difficulties. Usually, only small subsets are made available to mobile devices, and only on WAP devices. Standardization efforts now under way will probably bring mobile browsers to a level of compatibility sufficient for a widening of the B2C audience. Another channel which holds promise for B2C applications is Voice, with carefully designed user interfaces.

When dealing with the B2E situation, on the other hand, the task of "going pervasive" is much facilitated. The main characteristic of this class of application is that the body of users has a well-defined relationship to the business (for example, its employees, hence the acronym, but also others, such as privileged customers).

Consequently, it is possible to circumscribe the variability of client devices, user preferences and conditions of use. This can be achieved:

- Either statistically, for example, by stating

  "*A market survey has established that target customers use a certain type of PDA.*"

- Or by administrative decision: for example, by stating

  "*All our sales force shall be equipped with subnotebook XYZ.*"

  or

  "*We will offer a free WAP phone with our new Premier Banking contract.*"

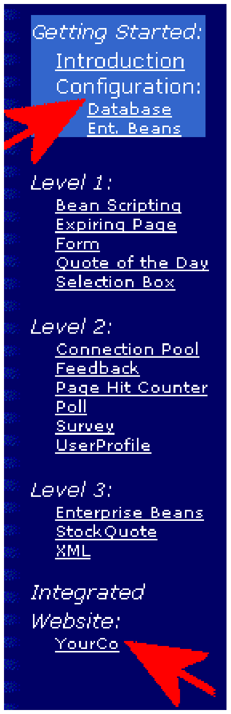
It is also possible to control the constituency of users, by requiring a specific enrollment procedure and establishing detailed classes of users. Therefore, better security can be achieved. This is fortunate, since such applications typically require access to more sensitive business data.

Because of the above considerations, it is expected that pervasive services will spread much more rapidly in the B2E market, at least for a while. This redbook is focused on a B2E application, not least because it allows us to demonstrate a wider variety of WebSphere Everyplace Server capabilities.

3.2 The sample B2E application: YourCo

The YourCo application is found in the samples packaged with WebSphere Application Server. There are some slight variations in the code of the samples delivered with each upgrade of WebSphere Application Server 3.5, which makes some components incompatible between upgrades. The explanations and examples in this Redbook are based on the sample packaged with WebSphere Application Server 3.5.4. However, the code itself can run on WebSphere Application Server 3.5.2 and above.

3.2.1 Installing and Running YourCo



If you have not installed WebSphere Application Server Version 3.5.4 and wish to try out the examples given in this redbook, you probably would like to install the WebSphere Application Server sample programs at this time. The YourCo application contains both the code of the application itself and the support material under <WAS_HOME>/WSsamples (where <WAS_HOME> is the installation path for WebSphere Application Server).

The configuration procedure is explained in detail at the sample site `<WAS_HOME>/WSsamples/index.html`, as shown in the figure to the left. Read the directions carefully. The top arrow points to the two configuration items: Database and EJBs. The Timeout part of YourCo is the only one which uses EJBs. If you do not plan to exercise Timeout, you can skip the Enterprise Beans Configuration tasks.

The bottom arrow shows how to start YourCo. The link points to a useful introduction page. The application itself starts from the URL:

```
http://<hostname>/WebSphereSamples/YourCo/index.html,
```

where `/WebSphereSamples/` is defined as an alias in httpd.conf.

3.2.2 Map of YourCo

YourCo is a typical (although somewhat simple) B2E application. It includes some unrestricted areas (such as News and Employee Directory) and an area restricted to registered employees, who can personalize their entry page and access several tools.

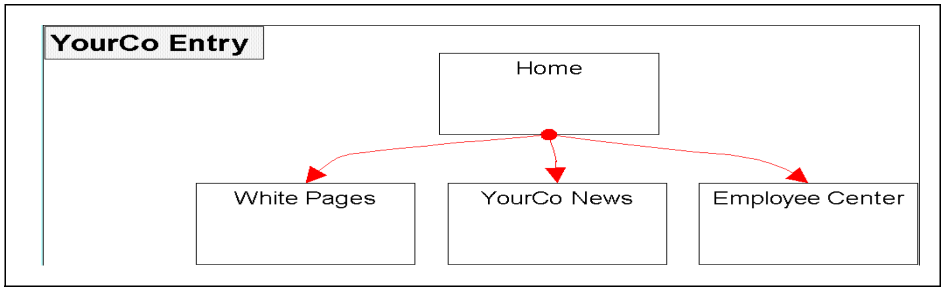


Figure 3-2 YourCo - entry level

The News area is time-dependent (in a simplistic way, which could be improved using a sophisticated newsfeed). The Employee Directory offers several kinds of searches: by Employee name, by Job type and by Department, it also provides you with a general browser.

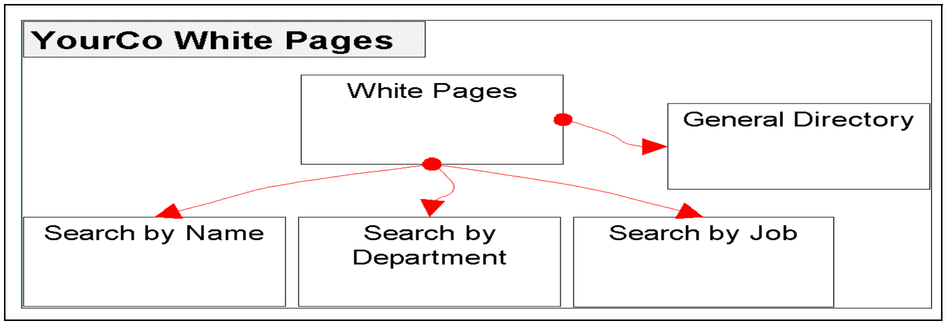


Figure 3-3 YourCo - the directory

The Employee Center requires authentication, for two reasons. First, some data available to employees may be confidential. Second, the services are personalized in some ways, and therefore depend on the identity of the user (in a WebSphere Everyplace Server context, this information could have been acquired at the edge by WebSEAL-Lite or the Wireless Gateway).

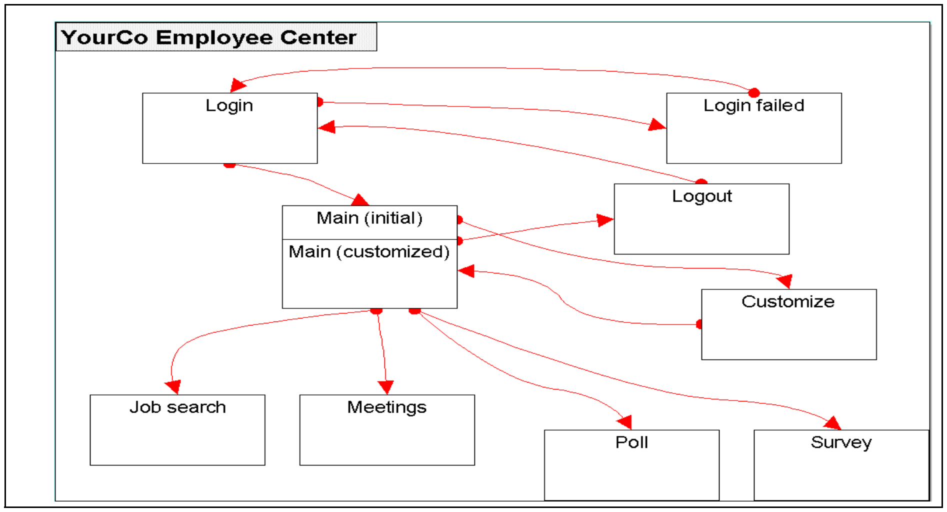


Figure 3-4 YourCo - the Employee Center (with Login)

The Employee Center offers:

- Company-wide items:
  - Poll
  - Survey
- Self-care items:
  - Customization of the front page
  - Management of leave (timeout)
- Work utilities:
  - Intra-company job advertisement and search
  - Scheduling of meeting rooms

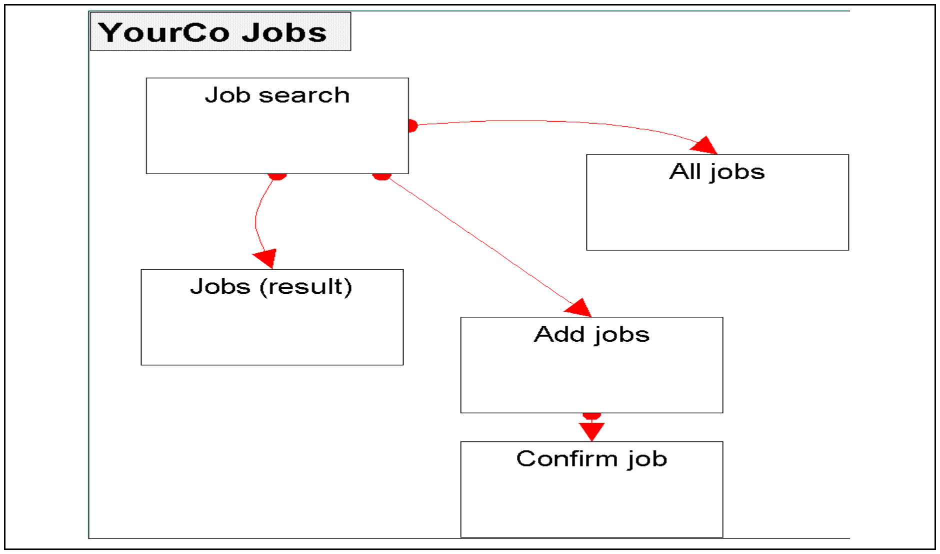


Figure 3-5 YourCo - Job search

In *Job search*, individuals can look up vacancies within the company, either by browsing the whole list of vacancies or by specifying a job profile. Managers have the extra option of adding a vacancy to the register.

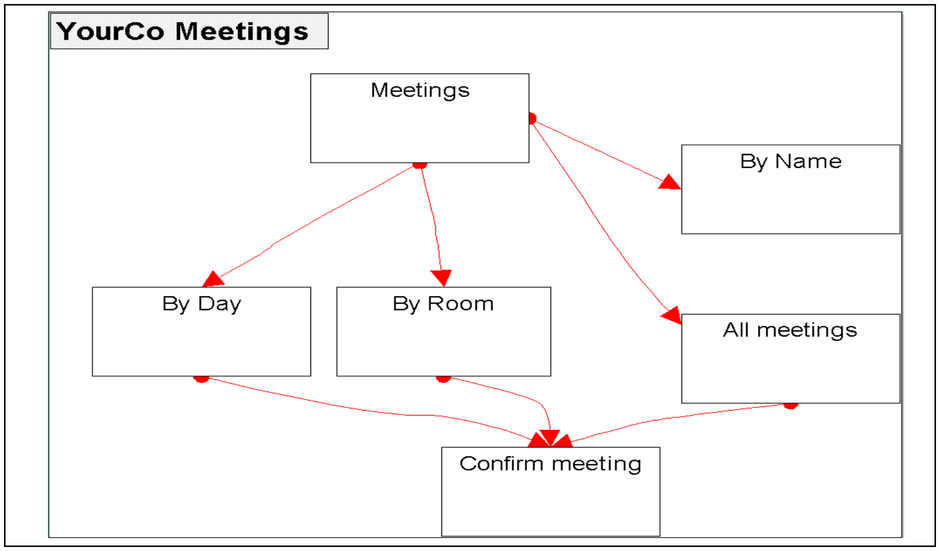


Figure 3-6 YourCo - Reserving a meeting room

The Leave management facility displays leave days available to an employee in three categories (Vacation, Personal leave and Sick leave), and allows the employee to transfer day credits from one category to another. It also displays the history of such transfers.

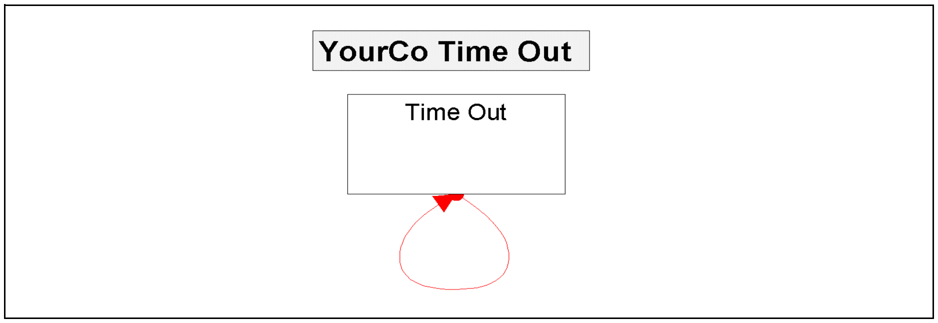


Figure 3-7 YourCo - Managing Leave time

3.2.3 Notes on implementation

The simple map of the navigation between YourCo application pages is not very informative, for the following reasons:

- The directory and the employee center make use of a frameset:
  - The directory shows:
    - Search criteria in the top frame, together with a set of general navigation links (outer level).
    - Search results in the bottom frame.
  - The Employee Center shows:
    - Content in the top frame, together with a set of general navigation links (outer level).
    - Inner level navigation (within the restricted area) in the bottom frame.
- The search criteria frame in the directory uses JavaScript to manage the state of the frame, and each state leads to a different state transition.

In order to understand and design a mobile adaptation of the site, we must re-map the content and navigation to make these inner workings explicit.

3.3 Adapting YourCo to the pervasive environment

The first approach to adapting YourCo is to plan a WML presentation of the same capabilities and content. Although it would not be practical to implement such an application entirely, planning it helps to review the challenges and the value proposition of each application area.

3.3.1 Revised map for YourCo on WML

The HTML site navigation shows redundancy. Links to various areas are repeated in some pages (for example in the text of the Employee Center page and in the inner navigation bar at the bottom). When designing the navigation for the WML site, we aim at reducing alternatives, and providing simple and unambiguous paths.

For details on WML structure and capabilities, see the WAP Forum document WAP-191-WML, *Wireless Application Protocol, Wireless Markup Language Specification Version 1.3*, available on the WAP Forum Web site (`http://www.wapforum.org/what/technical_1_2_1.htm`).

The basic unit of WML management is the *card*. A WML browser displays one card at a time, and links determine transitions between cards.

Default links *next* and *prev* are available. Other links are defined by *card ID*.

On most cards, we provide a *Home* link to go back to Card 0.

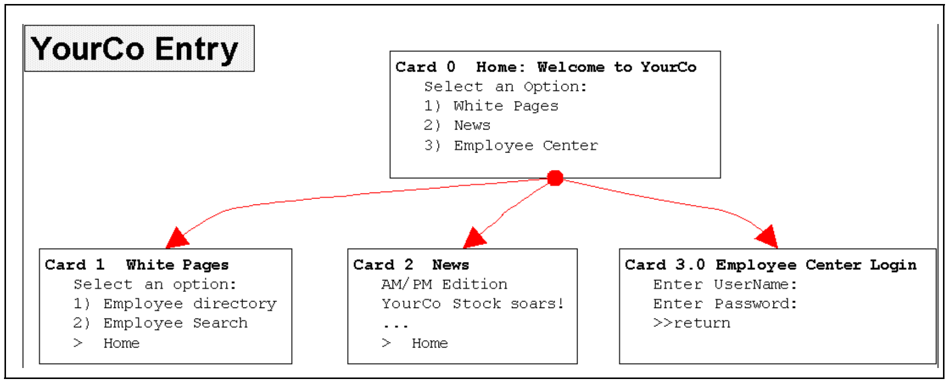


Figure 3-8 YourCo WML - Top level

Card 0 offers a choice among the three areas of the YourCo site.

Card 2 (News) has no navigation, except inasmuch as the content may have to be split into several cards, linked by *next* and *prev* links (when using WebSphere Transcoding Publisher, this is performed automatically by the Fragmentation engine).

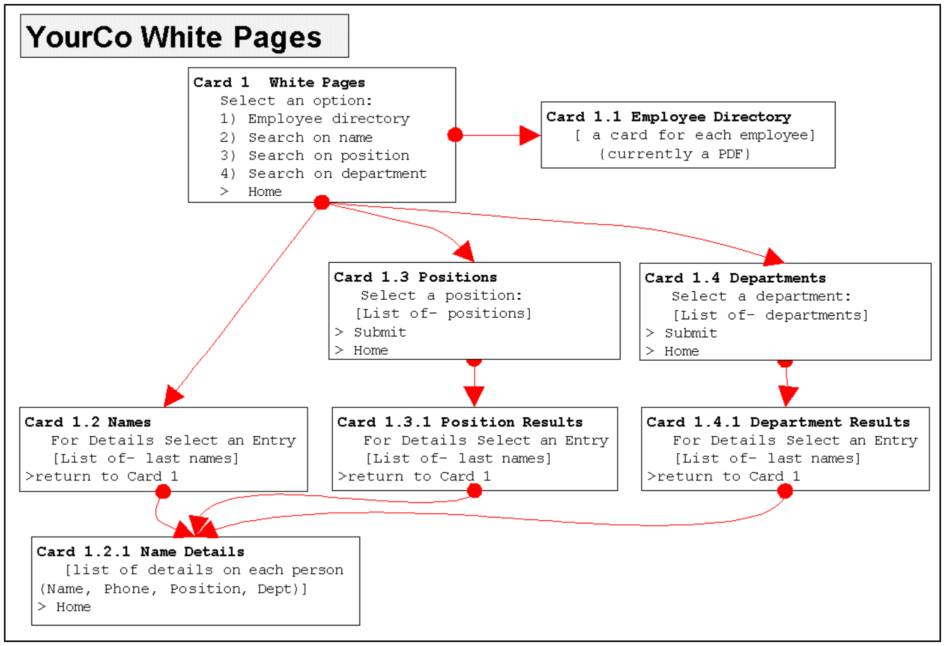


Figure 3-9 YourCo WML - The directory

Card 1 opens the Employee Directory (White Pages). In the HTML version, the various search methods appear in the top frame, and the choice among them (with consequent changes in the state of the frame) is handled by JavaScript.

In WML, we have to replace this arrangement with a sequence of cards:

- Card 1 handles the choice of methods.
- Cards 1.3 and 1.4 handle the selection of position and/or department.

  (All of the above was handled in one HTML frame)
- Cards 1.2, 1.3.1 and 1.4.1 show lists of selected names according to the search criteria.
- Finally, the user can see the details for one name in Card 1.2.1.

  Because going back is always an option (using the default link *prev*), we only need to provide shortcuts to either the start of the White Pages or the Home page.

Because of these changes in navigation control, it would be easier to code Card 1 as a static WML page, with links to the several servlets. The responses of the servlets could then be transcoded.

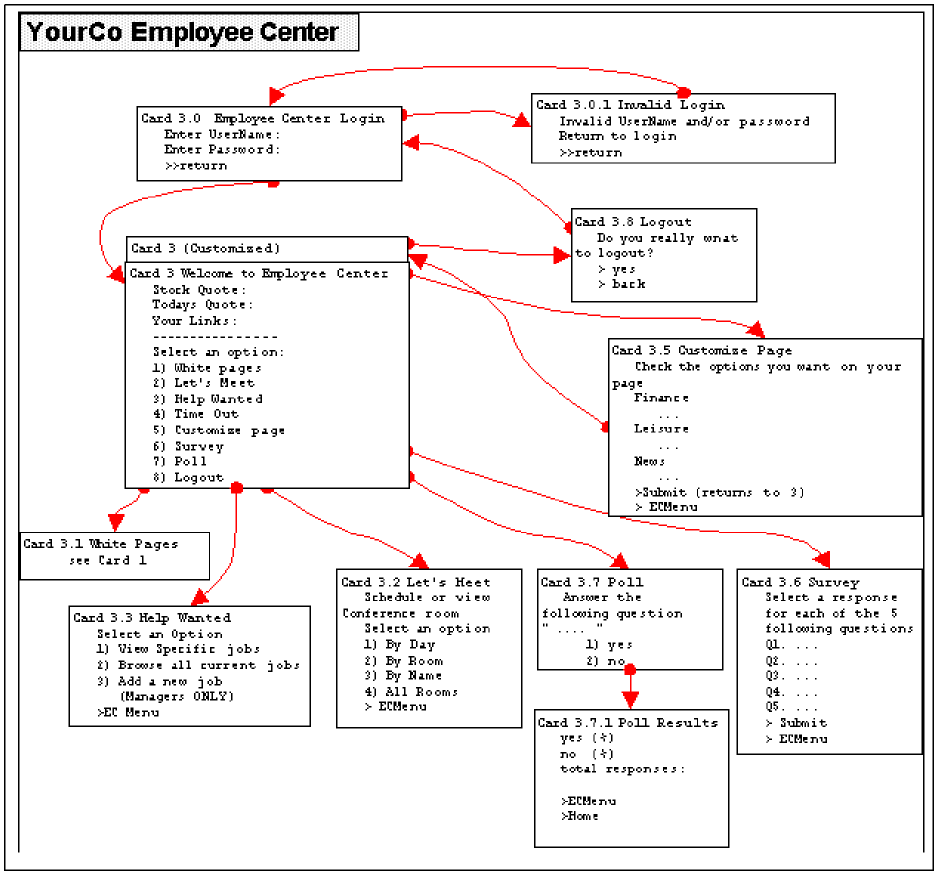


Figure 3-10 YourCo WML - The Employee Center (with Login)

The Employee Center navigation is, at the beginning, similar to the HTML version: the Login challenge is issued (Card 3.0), the credentials are either rejected (Card 3.0.1) or accepted (Card 3). However, Card 3 shows only the customized elements of the HTML, plus a list of links to several functions, some of which appeared on the same page in HTML. THe Poll function is split into a question card and a results card, and the Survey results are not shown at all (we could show them, but the detail is so cumbersome that few would ever access it on a WML browser).

The Customize Card (3.5) is very large, and would most likely be fragmented, but we do not need to design this: it is provided automatically by WebSphere Transcoding Publisher.

The White Pages option leads back to Card 1.

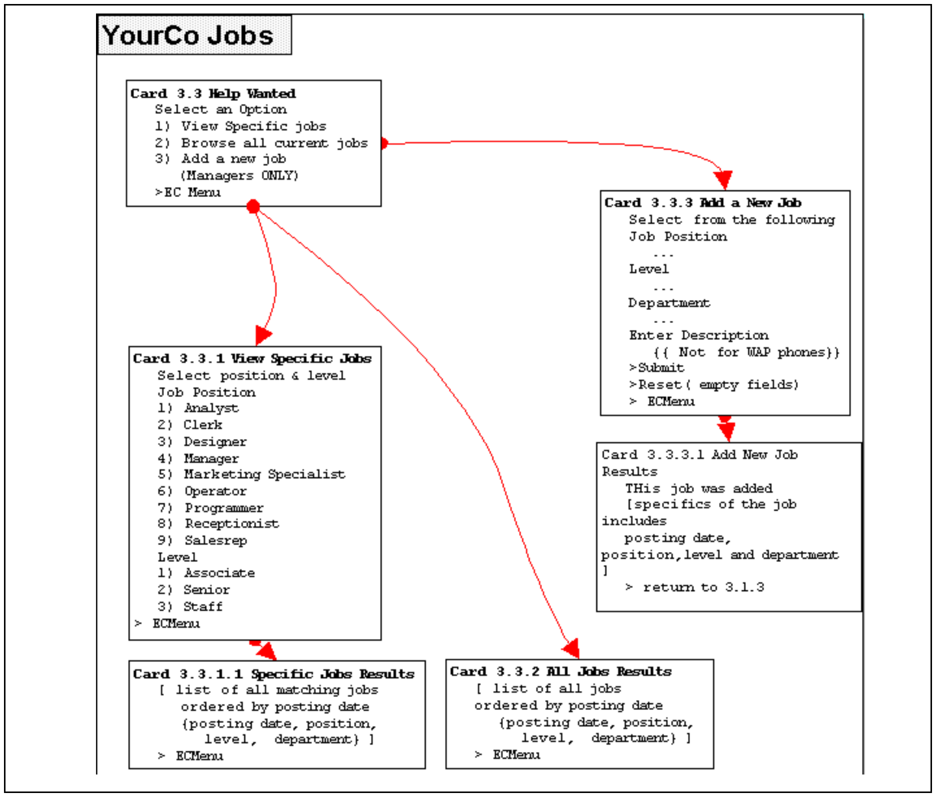


Figure 3-11 YourCo WML - Job search

The entry card for the Job search (Card 3.3) only offers a choice between three activities. Card 3.3.1 collects the search criteria. Cards 3.3.1.1 and 3.3.2 display results.

Card 3.3.3 is accessible only to managers: it collects information about a new job to be advertised; Card 3.3.3.1 confirms the new posting.

This section of the WML application differs little in structure from the HTML version, except for the splitting of the first page into two cards (3.3 and 3.3.1).

The same situation occurs in the Meetings section: the HTML version has all the search criteria in one page. The WML version first offers a choice of look-up types (Card 3.2). Then each look-up type is processed as in the HTML version.

This approach may require the coding of the static elements (*By Day* and *By Room*) as static WML (Cards 3.2, 3.2.1 and 3.2.2).

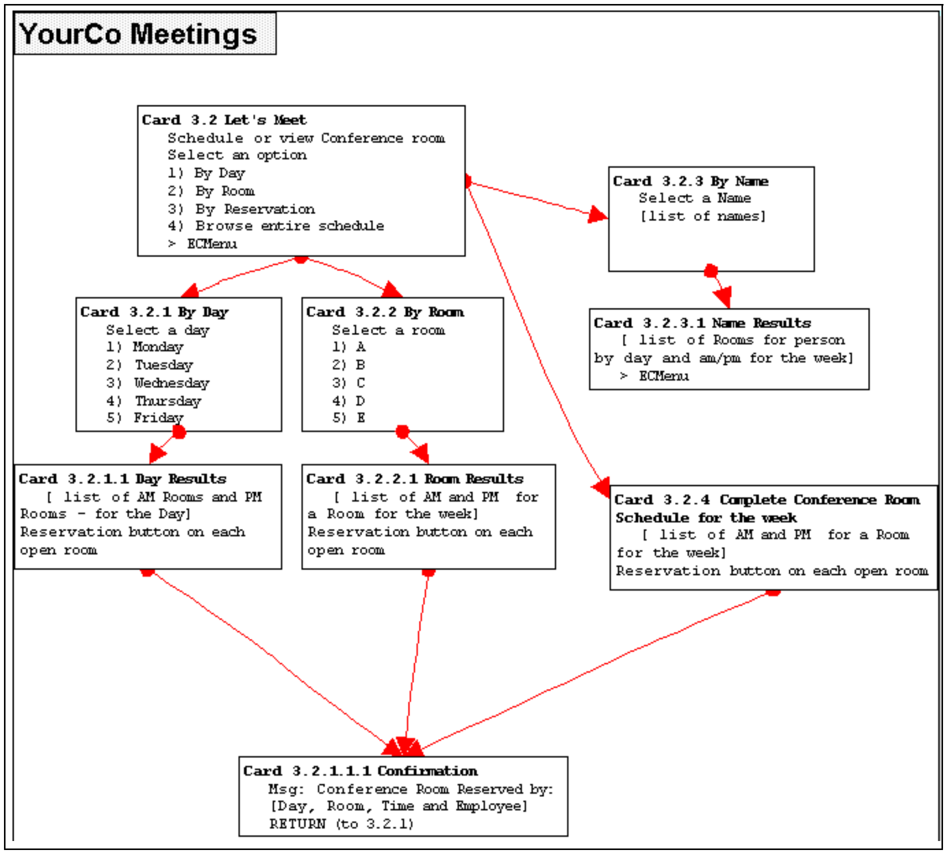
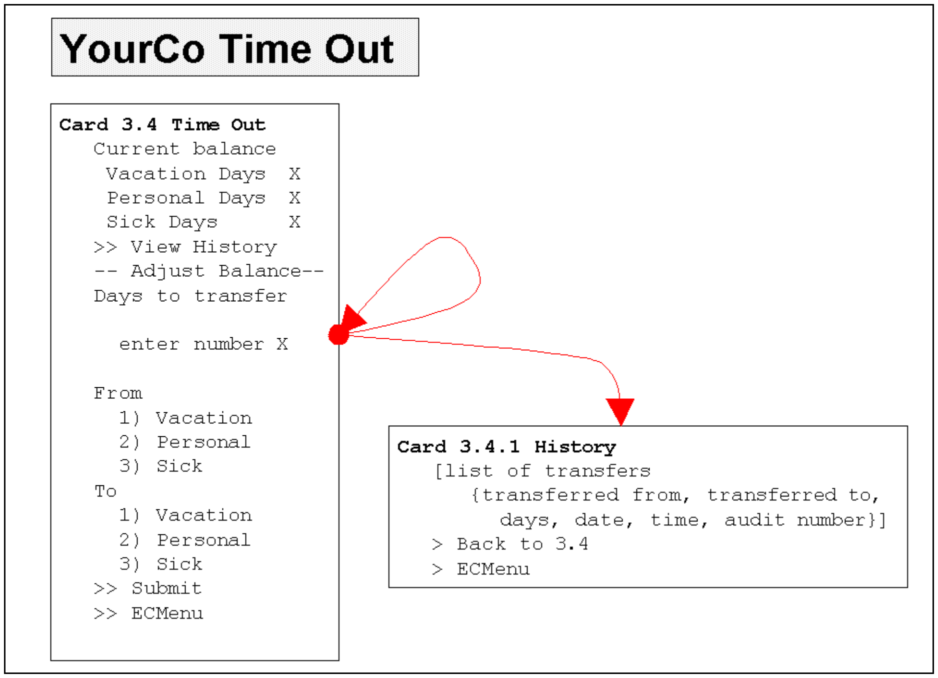

YourCo Meetings
Card 3.2 Let's Meet
Schedule or view Conference room
Select an option
1) By Day
2) By Room
3) By Reservation
4) Browse entire schedule
> ECMenu
Card 3.2.3 By Name
Select a Name
[list of names]
Card 3.2.1 By Day
Select a day
1) Monday
2) Tuesday
3) Wednesday
4) Thursday
5) Friday
Card 3.2.2 By Room
Select a room
1) A
2) B
3) C
4) D
5) E
Card 3.2.3.1 Name Results
[list of Rooms for person by day and am/pm for the week]
> ECMenu
Card 3.2.1.1 Day Results
[list of AM Rooms and PM Rooms - for the Day]
Reservation button on each open room
Card 3.2.2.1 Room Results
[list of AM and PM for a Room for the week]
Reservation button on each open room
Card 3.2.4 Complete Conference Room Schedule for the week
[list of AM and PM for a Room for the week]
Reservation button on each open room
Card 3.2.1.1.1 Confirmation
Msg: Conference Room Reserved by:
[Day, Room, Time and Employee]
RETURN (to 3.2.1)

Figure 3-12 YourCo WML - Reserving a meeting room

The Leave management section is straightforward. The history portion has been separated out, since few mobile users would wish to cope with a long and possibly fragmented sequence of records. It is still available on a link (Card 3.4.1). The main card (3.4) shows the current balance, and collects the data for a transfer.

Once submitted, the transfer is processed and returns the same card, with the balance adjusted.



```
Card 3.4 Time Out
   Current balance
    Vacation Days  X
    Personal Days  X
    Sick Days      X
   >> View History
   -- Adjust Balance--
   Days to transfer

     enter number X

   From
     1) Vacation
     2) Personal
     3) Sick
   To
     1) Vacation
     2) Personal
     3) Sick
   >> Submit
   >> ECMenu
```

```
Card 3.4.1 History
   [list of transfers
      {transferred from, transferred to,
        days, date, time, audit number}]
   > Back to 3.4
   > ECMenu
```

Figure 3-13 YourCo WML - Managing leave time

3.3.2 Selecting functionality for mobile use

When selecting which functions to adapt or introduce for pervasive usage, we take into account:

1. The specific needs of mobile users:
   a. Immediate value versus deferred access (for example, news versus statistics).
   b. Speed of access (for example, short navigation paths.)
   c. Topicality versus background (for example, if we want just the phone number).
2. The characteristics of mobile devices:
   a. Reduced message capacity (a few KB for WAP, a maximum of 160 bytes for SMS).
   b. Reduced display capacity.
   c. Constraints on input (pronounced for phones, less difficult for PDAs).

d. Constraints on MIME processing.

e. Linearity (one window, scrolling display, sequence of cards, or speech sequence only in the case of voice).

These considerations apply to the sections of YourCo as follows:

Table 3-1 Functionality for mobile use

Application section	1a	1b	1c	2a	2b	2c	2d	2e	Adaptation
► Directory									If transcoded, will require modification to bypass client-side state changes.[X]
– Entire directory:	-	-	-	-		+	-		Not in mobile use
– Search by Job/Dept	-	-	-	-					Add special function (Locate Expert) [A]
– Search by Name	+	+	+			-			Modify to allow wildcards (e.g., AN*) [Y]
► News	+	+	+	-	-	+			Use annotators [B] Use Voice [C] Add notification [D]
► Employee Center									Issues of migrating security to the edge.
– Customize	-	-*	-*	-	-	-			Not in mobile use
– Poll	-		-		-				Not in mobile use
– Survey	-		-	-	-	-		-	Not in mobile use
– Meetings	+#	+	+#						Add notification [E] Could be adapted to mobile (2nd priority)
– Jobs									
• Lookup	-	-	-		-				Not in Mobile use
• Add	+		+			-			Could be added for PDAs
– Leave									
• Transfer	+		+						Could be adapted to mobile (2nd priority)
• History	-	-	-						Not in mobile use

Notes:

- (+/-) A minus (-) sign indicates a *con*, a plus (+) sign a *pro*, an empty cell is neutral.
- (*) In *Customize*, it makes little sense to offer the function to mobile users on their mobile devices. On the other hand, the function, used on HTML browsers, enables mobile users to improve the overall score of the application, particularly on 1b. and 1c.

- (#) in *Meetings*, there may be some value to granting mobile access to room reservations, but only if the participants to the meeting could *also* be notified on their mobile devices.
- (A) A general search for a given job may not be very useful. But finding a nearby expert with a given specialty would be (see “Meeting invitation: extending YourCo” on page 97 and “Locate an Expert by telephone: VXML and Voice Server” on page 102).
- (B) Annotators can serve to reduce the volume of decorations and text, and serve only the news.
- (C) Voice is particularly adapted to delivering a news story (as in News bulletins on the radio).
- (D) Intelligent Notification can announce selected news items according to the users’ selections. The news item can be delivered directly or stored for later retrieval.
- (E) As noted above (*), the meeting function would be more useful if the participants could be notified on their mobile device. This requires extending the current function to select employees and passing the result to the Intelligent Notification Service or to an application based on the Push API.
- (X) Because of the current variety of devices, client-side state management is impractical. The opening page of the White Pages section must be replaced by a set of pages (one to offer the choice of search method, three to support the three methods offered).
- (Y) Using wildcards would reduce the amount of input required from users of small devices such as WAP phones. This would imply some redesign (input field to replace a drop-down menu).

In this redbook, we do not implement second priority items.

3.3.3 Implementing the adaptations

Restructuring

Restructuring is required to avoid client-side state management. Clearly, this is possible only if the developer has access to the code of the original application. We show how this can be done for the particular case of the White Pages section.

Figure 3-14 on page 93 shows the possible state transitions within the existing Search page. We wish to replace this single page with a set of single-state pages.

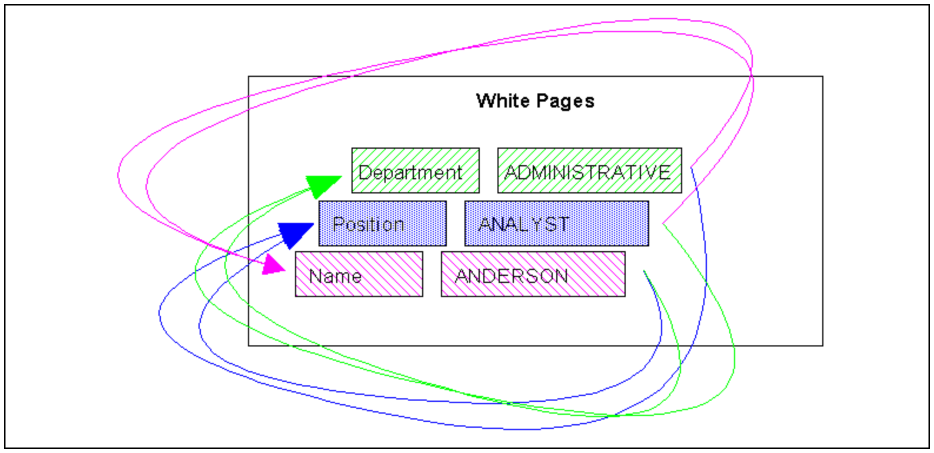


Figure 3-14 Search: multiple-state page

These single-state pages (*SearchChoice, SearchByName, SearchByJob, SearchByDept*) will offer equivalent state transitions, except that these transitions will be mediated by *SearchChoice*.

)

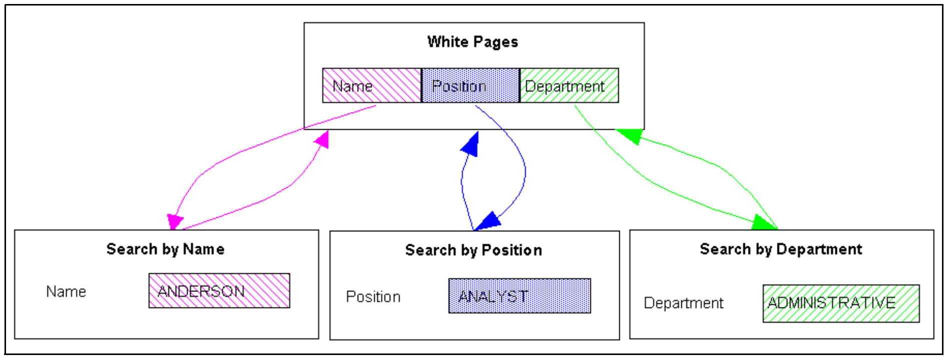


Figure 3-15 Search: single-state pages

Figure 3-16 on page 94 shows the four elements of functionality contributed by the script:

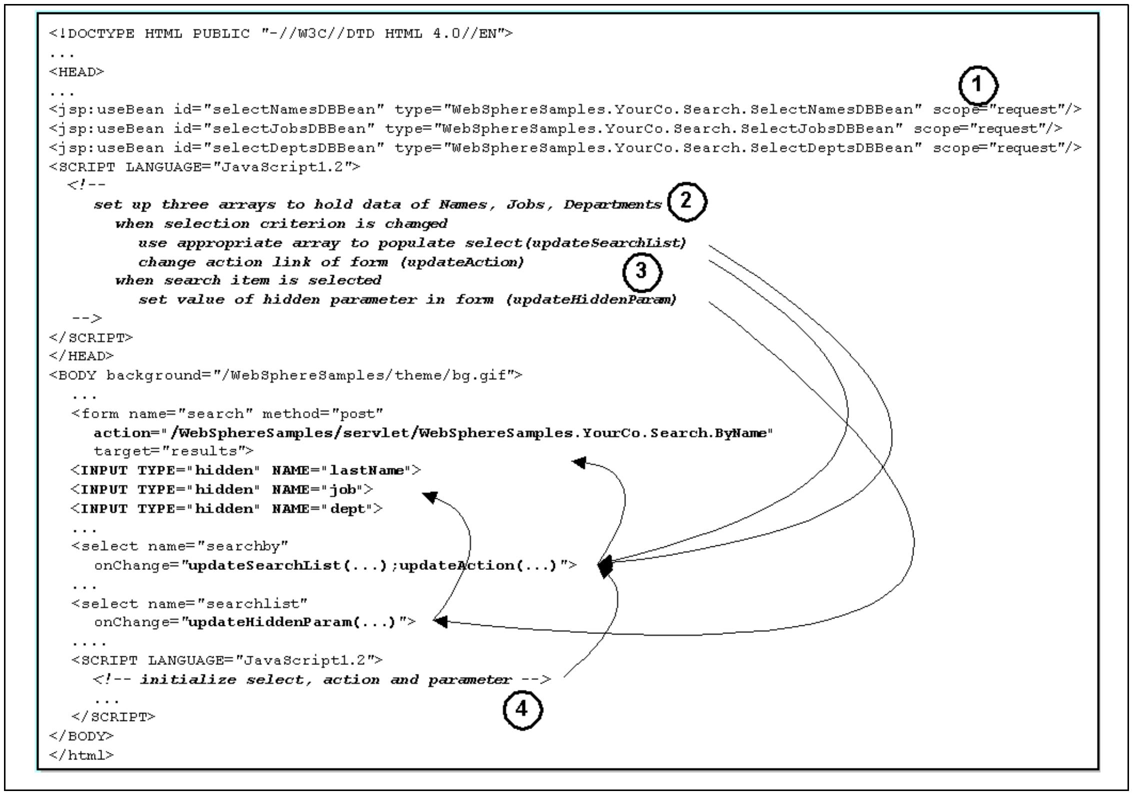
1. Declare three data beans (for *name*, *job* and *department* lists).
2. Collect data provided by the beans into three arrays accessible to the client side.

3. Define three functions to be invoked when the two select structures change state.
4. Initialize the two select structures and the hidden parameters which hold the final state.

The particular case of the search.jsp page does not exhaust the possibilities of client-side state manipulation. It is, however, typical of the most common situations. A state transition analysis usually shows a feasible approach to mapping the states on the server side and restructuring the application.

Once this is done, there remains the task of implementing this new structure. Rather than starting from nothing, it is usually possible to rework the existing page(s).

SearchChoice must be constructed separately, as a static page. The other three can be adapted from the existing search.jsp, by distributing to each a part of the functionality provided by the script in search.jsp.

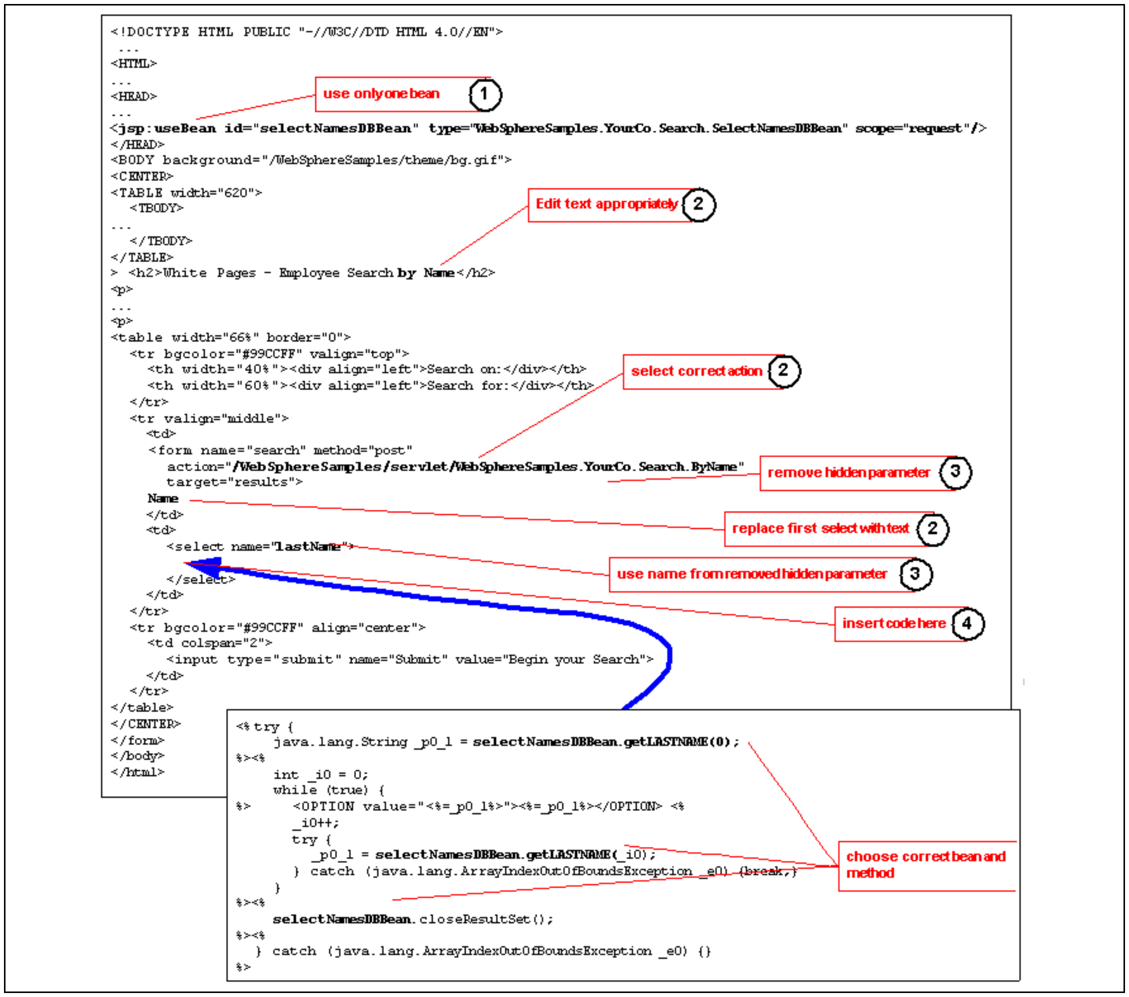


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
...
<HEAD>
...
<jsp:useBean id="selectNamesDBBean" type="WebSphereSamples.YourCo.Search.SelectNamesDBBean" scope="request"/>
<jsp:useBean id="selectJobsDBBean" type="WebSphereSamples.YourCo.Search.SelectJobsDBBean" scope="request"/>
<jsp:useBean id="selectDeptsDBBean" type="WebSphereSamples.YourCo.Search.SelectDeptsDBBean" scope="request"/>
<SCRIPT LANGUAGE="JavaScript1.2">
  <!--
    set up three arrays to hold data of Names, Jobs, Departments
      when selection criterion is changed
        use appropriate array to populate select(updateSearchList)
        change action link of form (updateAction)
      when search item is selected
        set value of hidden parameter in form (updateHiddenParam)
  -->
</SCRIPT>
</HEAD>
<BODY background="/WebSphereSamples/theme/bg.gif">
  ...
  <form name="search" method="post"
    action="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Search.ByName"
    target="results">
  <INPUT TYPE="hidden" NAME="lastName">
  <INPUT TYPE="hidden" NAME="job">
  <INPUT TYPE="hidden" NAME="dept">
  ...
  <select name="searchby"
    onChange="updateSearchList(...);updateAction(...)">
  ...
  <select name="searchlist"
    onChange="updateHiddenParam(...)">
  ....
  <SCRIPT LANGUAGE="JavaScript1.2">
    <!-- initialize select, action and parameter -->
    ...
  </SCRIPT>
</BODY>
</html>
```

Figure 3-16 Search: what the script does

Figure 3-17 shows how to distribute this functionality:

1. Each *SearchByX* page uses only one bean, and does not need to fill an array, since the work is done on the server side.
2. Static editing provides the particulars of each page (title, caption for the drop-down menu, action for the form).
3. The hidden parameters (one for each state) are no longer needed, but their name must now be given to the select construct, in order to ensure continuity with the rest of the processing.
4. The code collecting the data into arrays can now be inserted - with appropriate modifications - to prepare the option list for each select construct on the server side.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
 ...
<HTML>
...
<HEAD>
...
<jsp:useBean id="selectNamesDBBean" type="WebSphereSamples.YourCo.Search.SelectNamesDBBean" scope="request"/>
</HEAD>
<BODY background="/WebSphereSamples/theme/bg.gif">
<CENTER>
<TABLE width="620">
  <TBODY>
...
  </TBODY>
</TABLE>
> <h2>White Pages - Employee Search by Name</h2>
<p>
...
<p>
<table width="66%" border="0">
  <tr bgcolor="#99CCFF" valign="top">
    <th width="40%"><div align="left">Search on:</div></th>
    <th width="60%"><div align="left">Search for:</div></th>
  </tr>
  <tr valign="middle">
    <td>
    <form name="search" method="post"
      action="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Search.ByName"
      target="results">
    Name
    </td>
    <td>
      <select name="lastName">

      </select>
    </td>
  </tr>
  <tr bgcolor="#99CCFF" align="center">
    <td colspan="2">
      <input type="submit" name="Submit" value="Begin your Search">
    </td>
  </tr>
</table>
</CENTER>
</form>
</body>
</html>
```

```
<% try {
     java.lang.String _p0_1 = selectNamesDBBean.getLASTNAME(0);
%><%
     int _i0 = 0;
     while (true) {
%>     <OPTION value="<%=_p0_1%>"><%=_p0_1%></OPTION> <%
       _i0++;
       try {
         _p0_1 = selectNamesDBBean.getLASTNAME(_i0);
       } catch (java.lang.ArrayIndexOutOfBoundsException _e0) {break;}
     }
%><%
     selectNamesDBBean.closeResultSet();
%><%
   } catch (java.lang.ArrayIndexOutOfBoundsException _e0) {}
%>
```

use only one bean 1
Edit text appropriately 2
select correct action 2
remove hidden parameter 3
replace first select with text 2
use name from removed hidden parameter 3
insert code here 4
choose correct bean and method

Figure 3-17 Search: splitting off the states

Extending

Akin to restructuring in its need to access the original application code is the extension of functionality. We show examples of this in “Meeting invitation: extending YourCo” on page 97, “Locate an Expert: a location-based application” on page 98 and “News: adding an XML feed” on page 100.

Content adaptation

Content Adaptation is the function of the WebSphere Transcoding Publisher. This task can be accomplished using three kinds of tools, depending on the objective. Chapters 4 to 6 provide worked examples, as follows:

Annotators:

Annotators consist of special markup elements which direct WebSphere Transcoding Publisher in keeping, removing or replacing structure or text in an HTML document. They are used in processing the *entry page*, the first page of *News*, and the *Locate an Expert* extension (described in “Locate an Expert: a location-based application” on page 98, below).They could also be applied to several of the second priority or unused items.

Transcoders:

Transcoders must be used when the task exceeds the capabilities of annotators. They consist of Java classes registered as plug-ins. WebSphere Transcoding Publisher will invoke them as directed by the preferences specified at registration.

XSL stylesheets:

WebSphere Transcoding Publisher can apply an XSL stylesheet to an input XML document. The stylesheet is selected on the basis of preferences specified at registration. Preferences can apply to the DTD of the document, the request header, and the preference profile of the target device. In this redbook, they are applied to *news items* (described in Section 3.4.5, “Meeting notification: Intelligent Notification with triggers” on page 99, and Section 3.4.6, “News: adding an XML feed” on page 100).

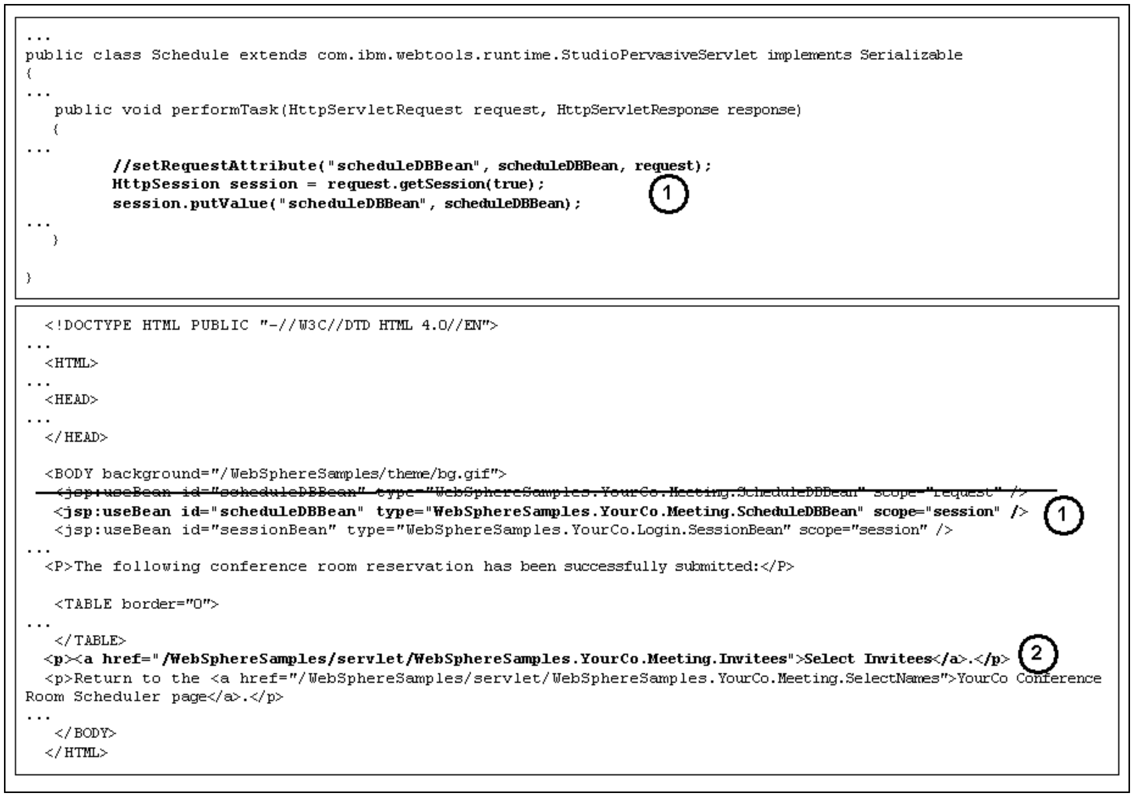
3.4 Adding pervasive functions to YourCo

WebSphere Everyplace Server supports a variety of application resources. In order to exercise those, we extend the functionality of YourCo in several directions. For a general map of the original YourCo, modifications and additions, see 3.5, “YourCo directory” on page 102.

3.4.1 Meeting invitation: extending YourCo

The Meeting function in YourCo is incomplete, in that it only reserves a conference room for a meeting. Other items are required for meeting management. We add a simple extension, which allows for selection of the participants from the list of all employees. The resulting selection can be directed to various services for scheduling, notification, document distribution, etc.

This extension requires two changes to *ScheduleResults.jsp*, and one to *Schedule.java*.



```
...
public class Schedule extends com.ibm.webtools.runtime.StudioPervasiveServlet implements Serializable
{
...
   public void performTask(HttpServletRequest request, HttpServletResponse response)
   {
...
         //setRequestAttribute("scheduleDBBean", scheduleDBBean, request);
         HttpSession session = request.getSession(true);
         session.putValue("scheduleDBBean", scheduleDBBean);          1
...
   }

}
```

```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
...
  <HTML>
...
  <HEAD>
...
  </HEAD>

  <BODY background="/WebSphereSamples/theme/bg.gif">
   ~~<jsp:useBean id="scheduleDBBean" type="WebSphereSamples.YourCo.Meeting.ScheduleDBBean" scope="request" />~~
   <jsp:useBean id="scheduleDBBean" type="WebSphereSamples.YourCo.Meeting.ScheduleDBBean" scope="session" />  1
   <jsp:useBean id="sessionBean" type="WebSphereSamples.YourCo.Login.SessionBean" scope="session" />
...
  <P>The following conference room reservation has been successfully submitted:</P>

   <TABLE border="0">
...
   </TABLE>
  <p><a href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Meeting.Invitees">Select Invitees</a>.</p>  2
  <p>Return to the <a href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Meeting.SelectNames">YourCo Conference
Room Scheduler page</a>.</p>
...
   </BODY>
  </HTML>
```

Figure 3-18 Inserting link to invitations

1. The scope of the ScheduleDBBean must now be Session, so that the reservation will be available to notify the invitees. This affects both the servlet and the JSP.
2. An extra link must be inserted, to invoke the Invitees.java servlet.

The *I*nvitees.java servlet uses the InviteesDBBean.java to collect information about employees, and Invitees.jsp presents a list so the user can select whom to invite. The resulting form invokes the downstream functionality (for example, notifying the invitees).

3.4.2 Locate an Expert: a location-based application

As mentioned earlier (seeTable 3-1 on page 91), the Directory search by job or department is somewhat unwieldy when used on a mobile device. In order to meet criteria 1a to 1c, we choose to offer a more restricted, but more immediately useful function. *Locate an Expert* queries the user for a type of expertise (we use the Job data as an approximation), and uses location information to find the nearest expert in that category. This combines the static data from the YourCo database with dynamic information at the user's current location.

This extension is available on static machines as well. For such cases, the location information can be derived from the company's network configuration information (based on the machine initiating the request) or from the employee record in the YourCo database (this requires Login to acquire the employee's identity).

In order to make this extension available, we add a link to the Welcome page of YourCo. This link points to a frameset containing the LocateInput.html static page. If we wished, we could instead use a servlet to acquire the list of expert specialties from the database. In turn, LocateInput.html invokes the Location-based servlet FindExpert.java.

Details of this application are given in Chapter 11, "Location-Based Services (LBS)" on page 435.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<title>YourCo Main Site</title>
...
</head>
<body background="/theme/bg.gif">
...
     <TD width="120" valign="middle" align="left">
       <A href="/WebSphereSamples/YourCo/main.html" target="_top">
          <FONT size="-1">Employee Center</FONT>
       </A>
     </TD>
     <TD width="5"></TD>
     <TD><IMG src="/WebSphereSamples/theme/button.gif" width="20" height="20" border="0"></TD>
     <TD valign="middle" align="left" width="120">
       <A href="/WebSphereSamples/YourCo/Locate/frameset.html" target="_top">    1
          <FONT size="-1">Locate an Expert</FONT>
       </A>
     </TD>
   </TR>
...
</BODY>
</HTML>
```

Figure 3-19 Inserting a link to Locate

3.4.3 Meeting notification: using the Push facility

A simple notification service can be implemented by using the Push API directly. Details are given in Chapter 9, “Push messaging applications” on page 323.

3.4.4 Meeting notification: using the Intelligent Notification Services

A more sophisticated notification function can be implemented using the INS. This application can use different delivery channels according to the priority status of the message. Details are given in Chapter 10, “Intelligent Notification Services (INS)” on page 359.

3.4.5 Meeting notification: Intelligent Notification with triggers

The Intelligent Notification Services can filter incoming data according to users’ registered preferences and notify the users for each event. Users have the choice of displaying the whole item directly, or seeing only a reference (URL and item ID) for asynchronous access. We use the Meeting Schedules to demonstrate these capabilities. Details are given in Chapter 10, “Intelligent Notification Services (INS)” on page 359.

3.4.6 News: adding an XML feed

The YourCo News provides company news based on date and time of day. It could also provide access to external newsfeeds for YourCo employees. We add a demonstration of this as a button on the YourCo News page, which leads to news about IBM Everyplace. The external news feed is provided in XML, which must be processed by WebSphere Transcoding Publisher for use by various devices.

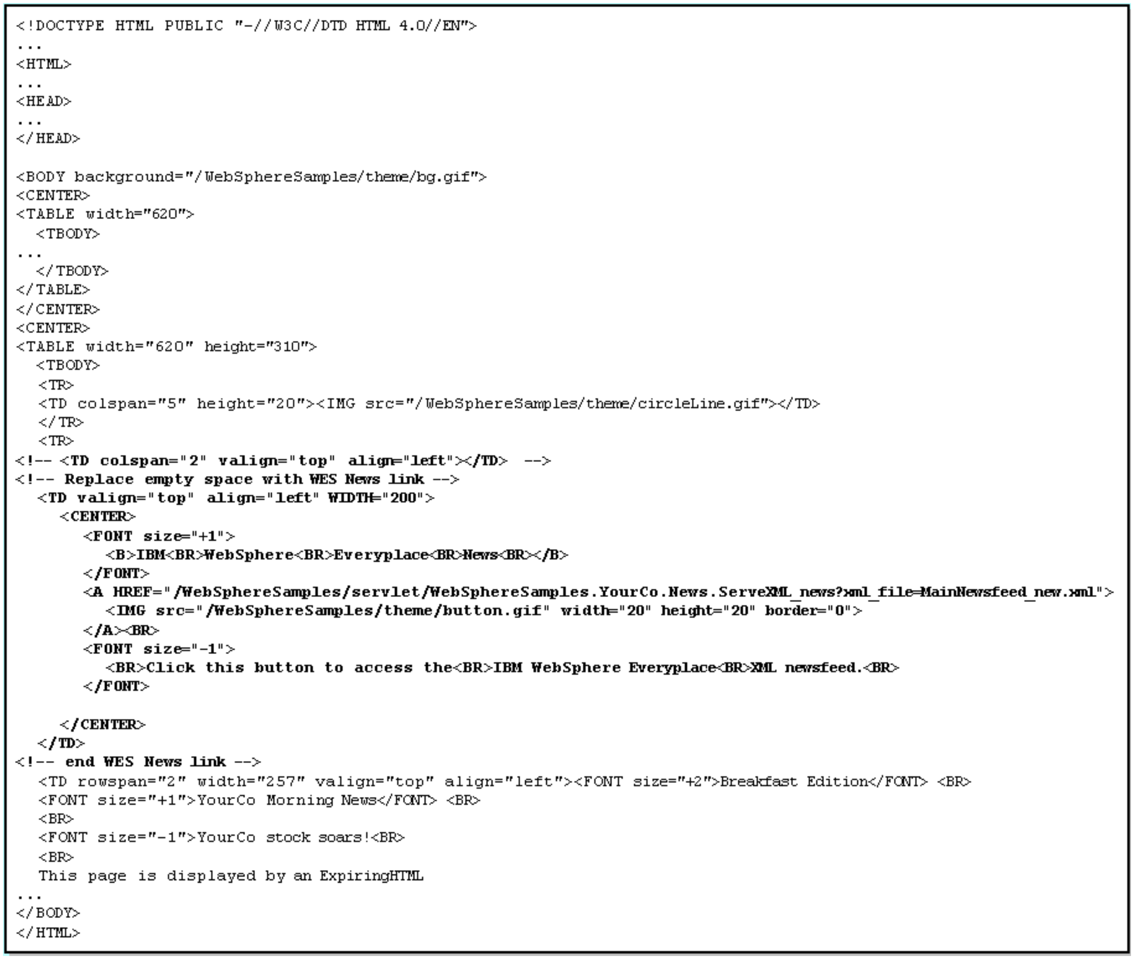
The relevant pages are the two JSPs found in the following directory:

```
<WAS_HOME>\hosts\default_host\WSsamples_app\web\YourCo\ExpHTMLServlet.
```

As they are almost identical, we show the modification once (see Figure 3-20 on page 101).

The ServeXML_news.java servlet serves XML documents. Unless the user has an XML-aware browser, it will be necessary to process the document, typically by applying an XSL stylesheet to it in the WebSphere Transcoding Publisher.

Note: Chapter 6, “Using stylesheets” on page 211 deals with the use of XSL stylesheets in detail.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
...
<HTML>
...
<HEAD>
...
</HEAD>

<BODY background="/WebSphereSamples/theme/bg.gif">
<CENTER>
<TABLE width="620">
  <TBODY>
...
  </TBODY>
</TABLE>
</CENTER>
<CENTER>
<TABLE width="620" height="310">
  <TBODY>
  <TR>
  <TD colspan="5" height="20"><IMG src="/WebSphereSamples/theme/circleLine.gif"></TD>
  </TR>
  <TR>
<!-- <TD colspan="2" valign="top" align="left"></TD>  -->
<!-- Replace empty space with WES News link -->
  <TD valign="top" align="left" WIDTH="200">
    <CENTER>
      <FONT size="+1">
        <B>IBM<BR>WebSphere<BR>Everyplace<BR>News<BR></B>
      </FONT>
      <A HREF="/WebSphereSamples/servlet/WebSphereSamples.YourCo.News.ServeXML_news?xml_file=MainNewsfeed_new.xml">
        <IMG src="/WebSphereSamples/theme/button.gif" width="20" height="20" border="0">
      </A><BR>
      <FONT size="-1">
        <BR>Click this button to access the<BR>IBM WebSphere Everyplace<BR>XML newsfeed.<BR>
      </FONT>

    </CENTER>
  </TD>
<!-- end WES News link -->
  <TD rowspan="2" width="257" valign="top" align="left"><FONT size="+2">Breakfast Edition</FONT> <BR>
  <FONT size="+1">YourCo Morning News</FONT> <BR>
  <BR>
  <FONT size="-1">YourCo stock soars!<BR>
  <BR>
  This page is displayed by an ExpiringHTML
...
</BODY>
</HTML>
```

Figure 3-20 Inserting WebSphere Everyplace Server News

3.4.7 News on the telephone: VXML

News delivered from an XML newsfeed lend themselves particularly well to voice delivery. WebSphere Transcoding Publisher applies its default Voice Transcoder to convert the input XML news item to VXML. This process is demonstrated in Chapter 6, “Using stylesheets” on page 211.

The VXML document is then routed through the Voice Server, from which it travels over IP either to a telephone network adapter or directly to a computer (VoIP client required).

The voice application presented here uses the Voice Server SDK on Windows NT. There is no need to deploy a Voice Server or a VoIP router. Details are presented in Chapter 11, “Location-Based Services (LBS)” on page 435.

3.4.8 Locate an Expert by telephone: VXML and Voice Server

The location-based application can also be reached using an ordinary telephone. The location service provides a mapping of telephone numbers to location, and the Voice Browser prompts the user for the desired expertise. Details are presented in Chapter 11, “Location-Based Services (LBS)” on page 435.

3.4.9 Accessing the Leave Bank through MQSeries Everyplace

MQSeries Everyplace, though not restricted to mobile or restricted capacity devices, nevertheless proves particularly useful with those.

We imagine a manager needing remote access to company information from a PDA, and illustrate this function with a simple access to the Employee Leave database. Details appear in Chapter 13, “Data synchronization for enterprise applications” on page 517.

3.4.10 Synchronizing remote applications

Chapter 12, “Voice-enabled applications” on page 493 demonstrates the use of WebSphere Everyplace Server to support remote synchronization of Office tools (MS Outlook, MS Exchange, Lotus Notes).

3.5 YourCo directory

To assist you in finding your way through the diversity of materials, the following pages present a map, or directory, of the augmented YourCo application. Each folder on the left hand side is linked to three boxes listing its files, whether unchanged from the original YourCo, modified for the purposes of this redbook, or altogether new.

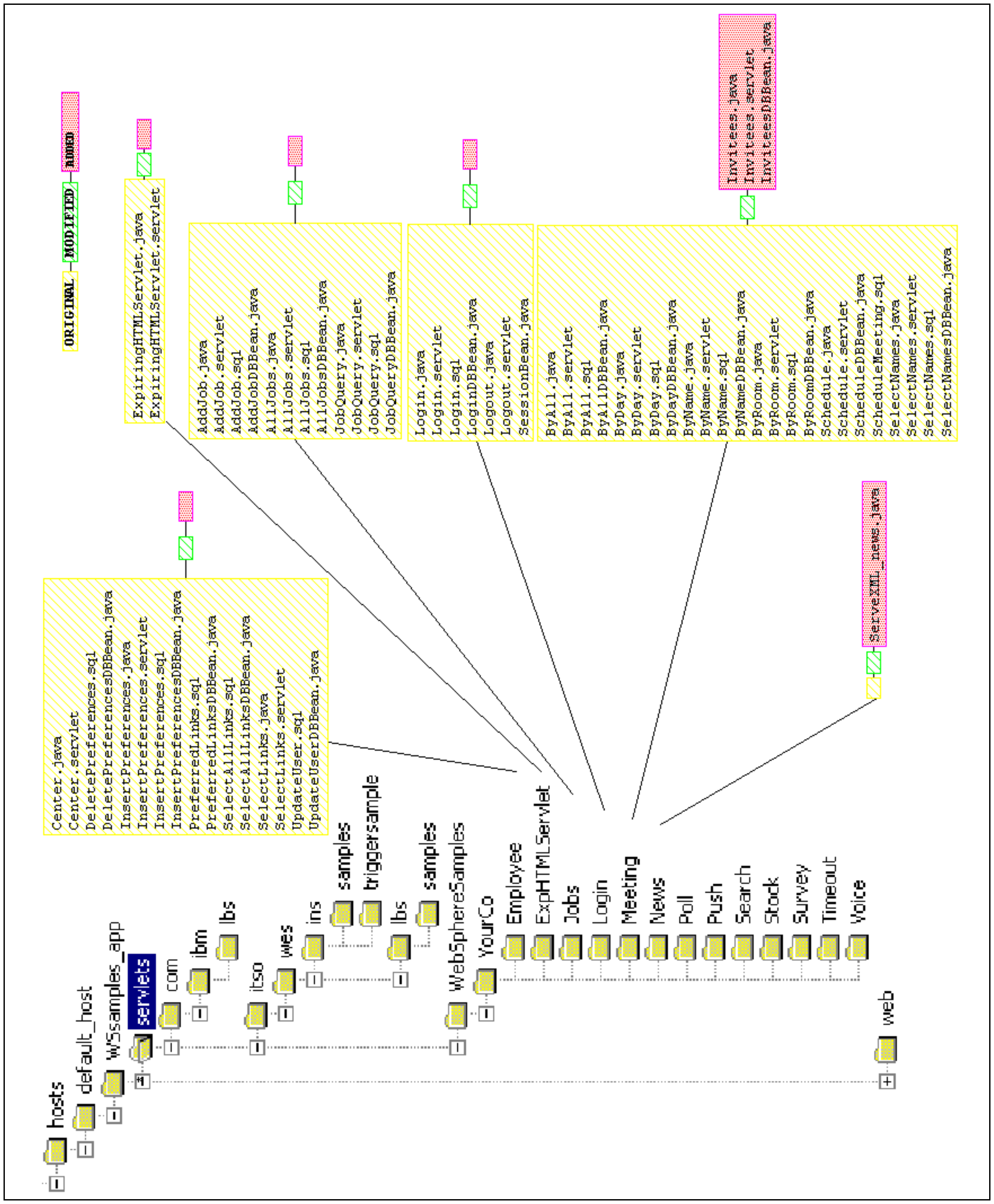


Figure 3-21 YourCo servlets - 1

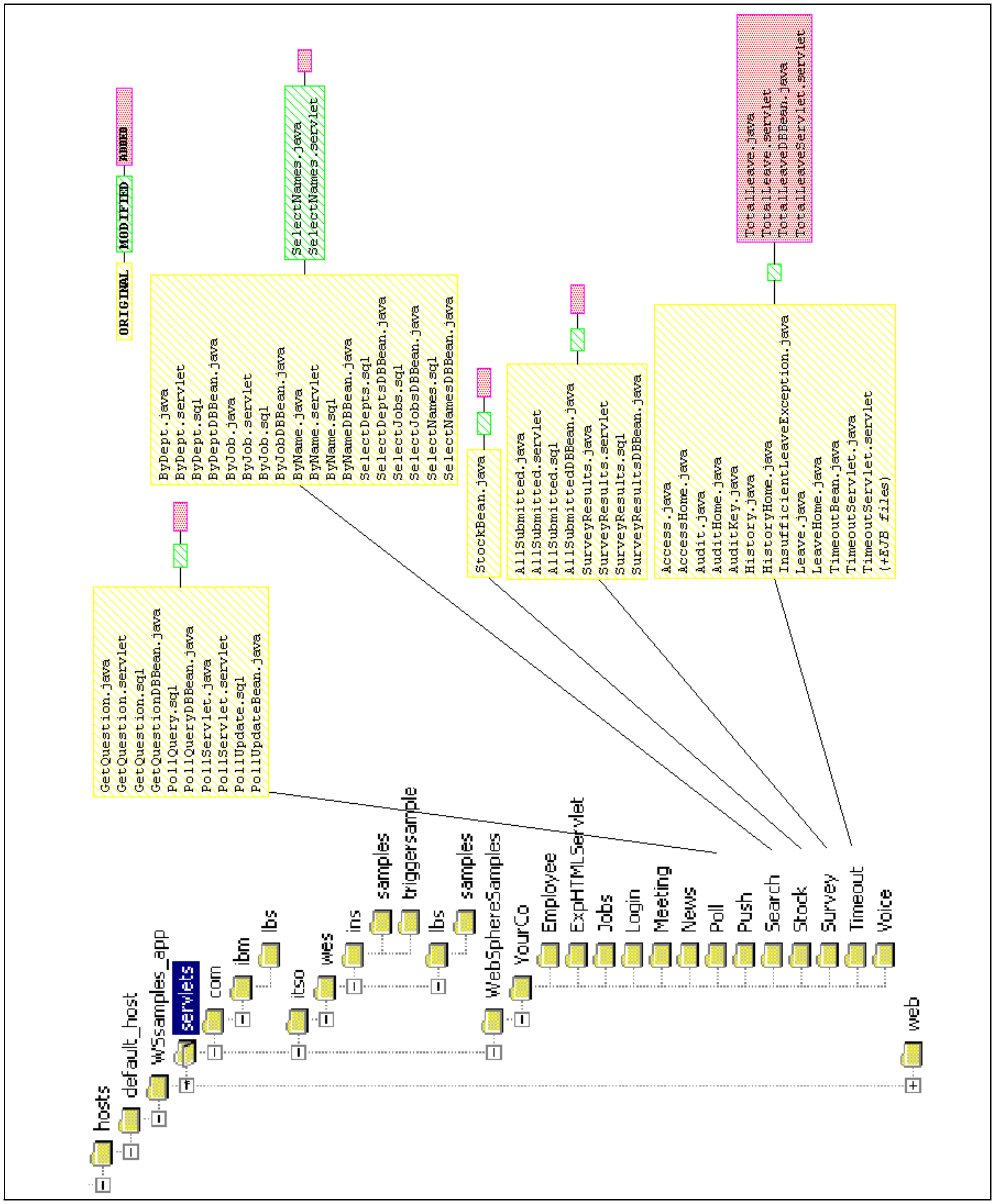


Figure 3-22 YourCo servlets-2

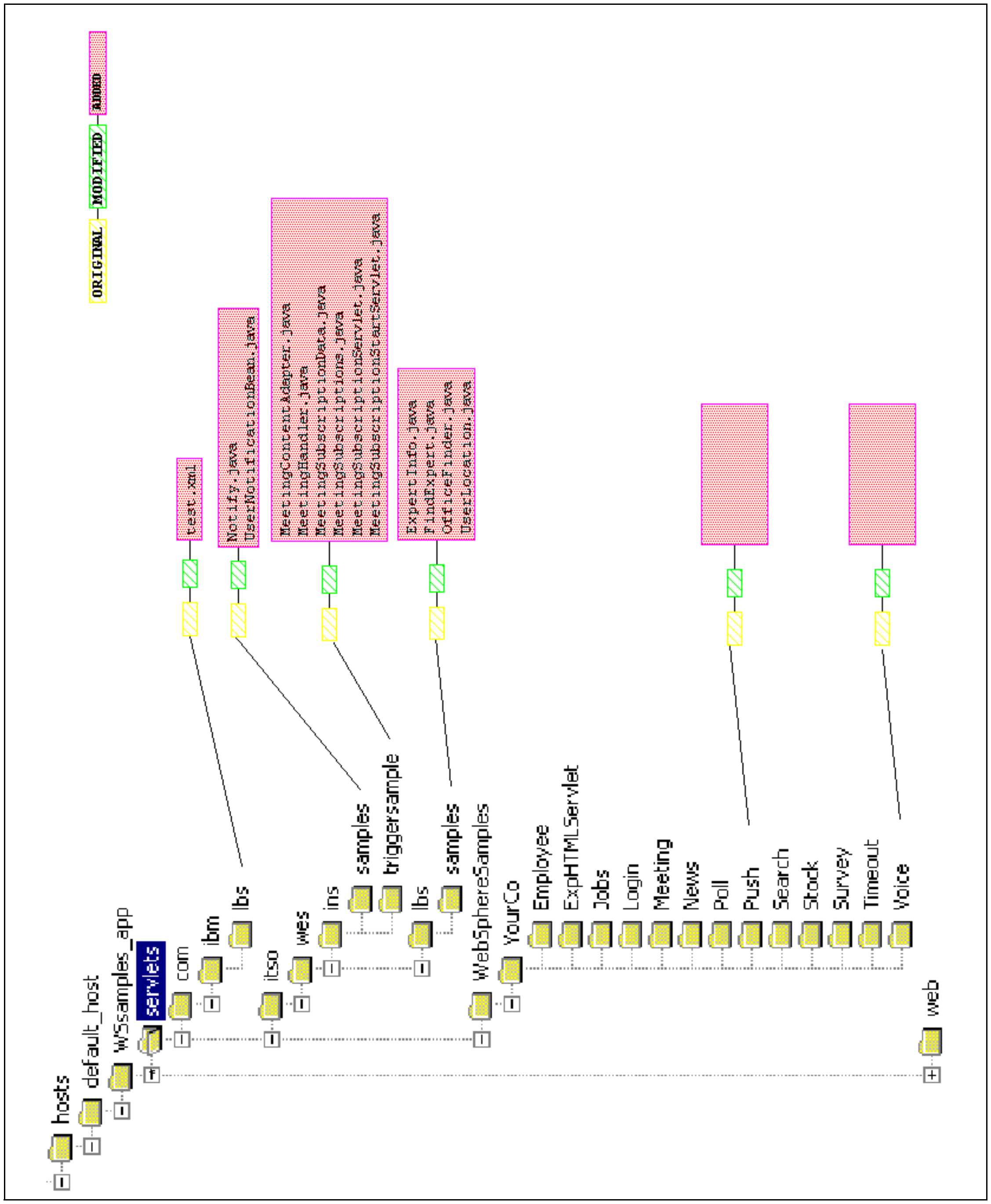


Figure 3-23 YourCo servlets-3

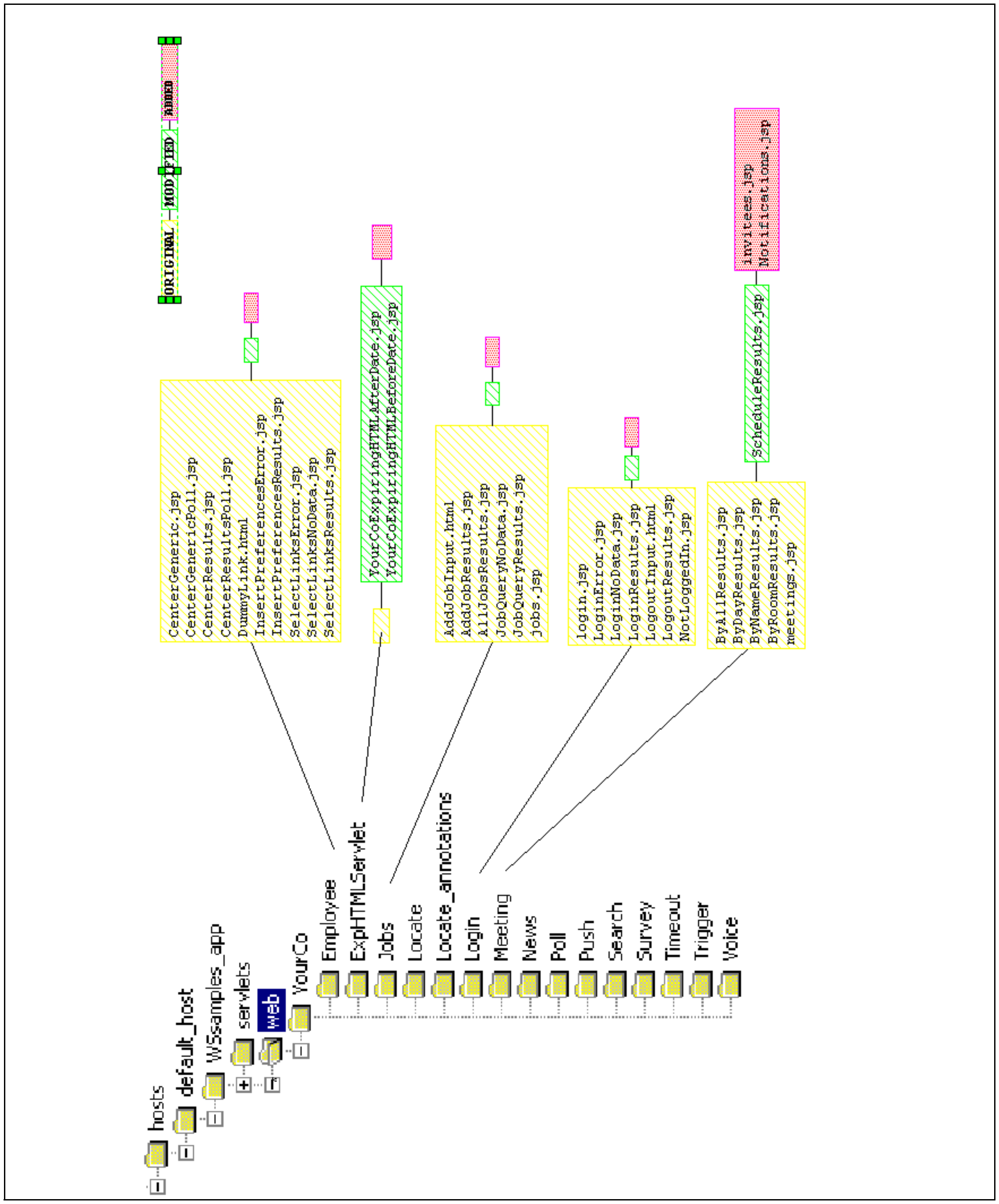


Figure 3-24   YourC0_web-1

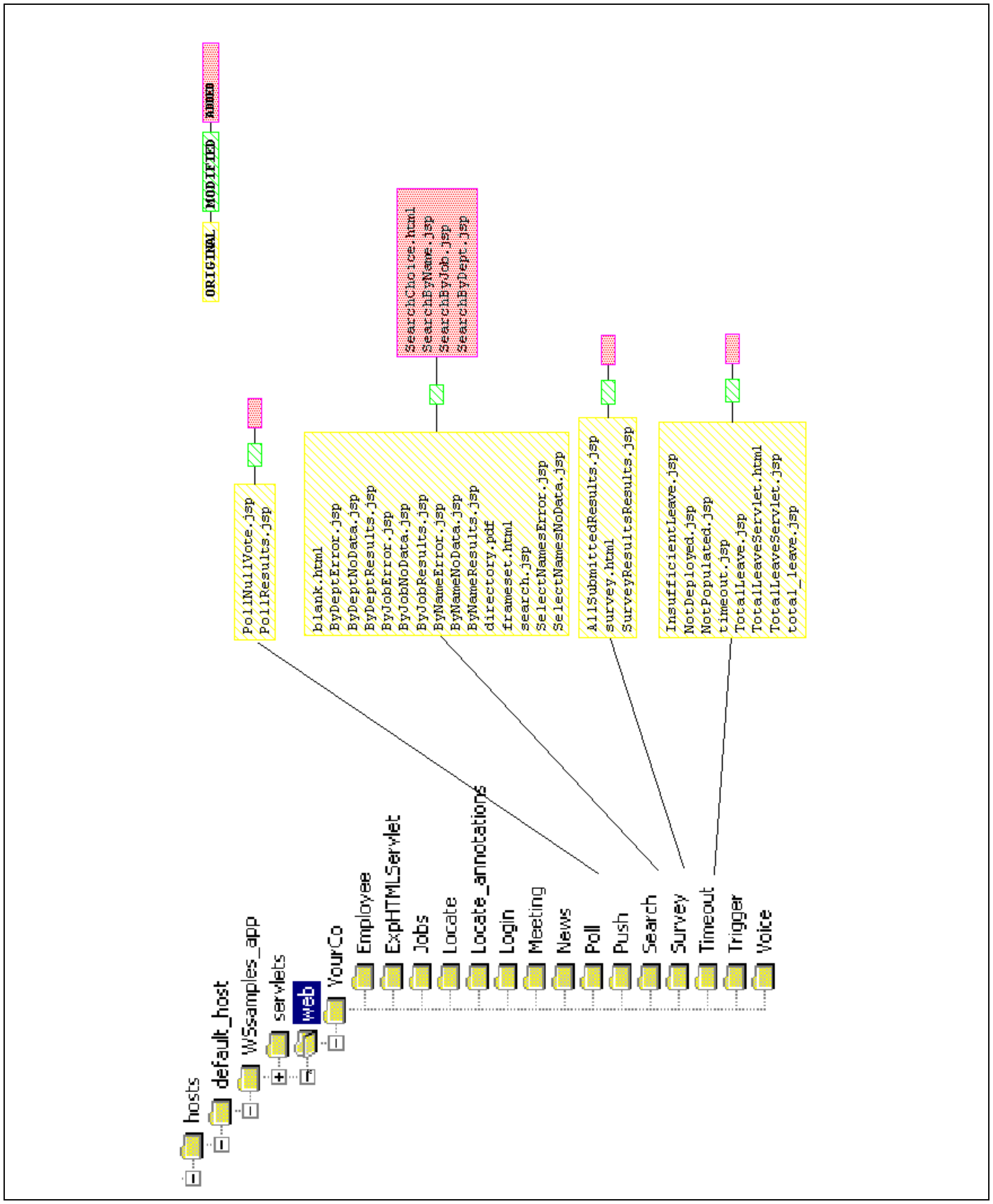


Figure 3-25 YourC0_web-2

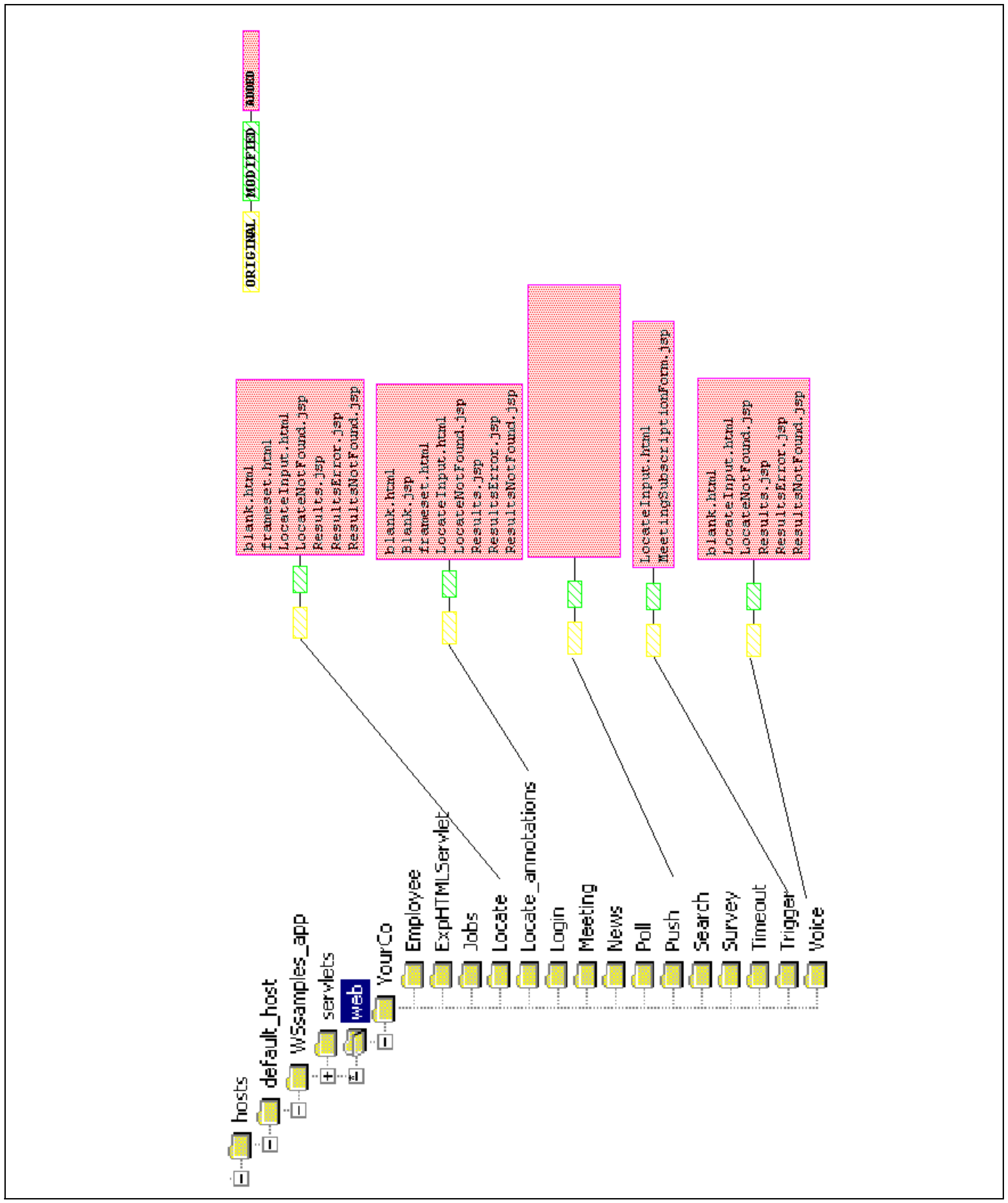


Figure 3-26 YourC0_web-3

3.6 Sample lab configuration

Most of the scenarios described in this redbook have been integrated in a WebSphere Application Server environment shown in Figure 3-27. For specific information about a particular function or service provided by WebSphere Everyplace Server, refer to the proper chapter in this redbook.

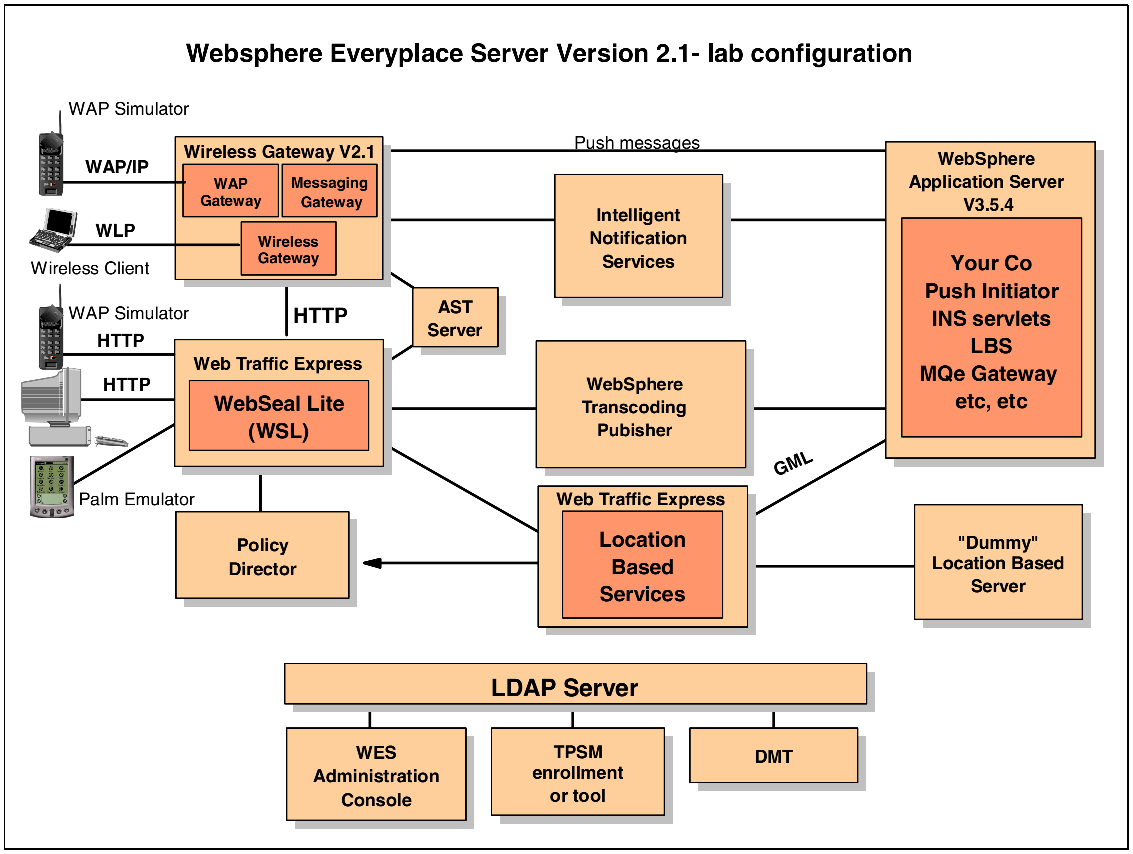


Figure 3-27 Lab configuration

Part 2

Adapting new and existing applications

In this part of the redbook, we describe ways to rapidly integrate your enterprise applications into a WebSphere Everyplace Server (WES) environment and therefore make them also available from wireless devices such as WAP phones, by implementing new and enhanced capabilities incorporated in the current releases of WebSphere Everyplace Server offerings, such as transcoding, annotators for text clipping, stylesheets and the Wireless Gateway. You will find step-by-step examples and scenarios to deploy and adapt your enterprise applications in a WebSphere Everyplace Server environment.

4

Transcoding application content

Transcoding is a technology that enables you to extend your Web-based applications to wireless and pervasive devices, while at the same time minimizing the conversion effort. IBM brings the transcoding technology to the market with the IBM WebSphere Transcoding Publisher product . This chapter describes some of the new features included in IBM WebSphere Transcoding Publisher Version 4.0 which will help you to adapt your applications so they can be accessed from wireless devices such as WAP phones.

In this chapter, you will find the following information:

- Overview of the fundamentals of WebSphere Transcoding Publisher.
- Discussion of the new features within WebSphere Transcoding Publisher Version 4.0.
- Discussion of the various features and capabilities of WebSphere Transcoding Publisher.
- A sample scenario showing the use of WebSphere Transcoding Publisher to display Web information on a WAP simulator (without any customization).

Note: IBM WebSphere Transcoding Publisher is a component of IBM WebSphere Everyplace Server Service Provider Offering (for AIX and Solaris). It is also a component of IBM WebSphere Everyplace Server Enable Offering (Windows 2000, AIX and Solaris).

4.1 Overview

This chapter introduces you to the IBM WebSphere Transcoding Publisher Version 4.0 (referred to as WTP). WebSphere Transcoding Publisher helps you to easily extend your existing Web content to the wireless and pervasive computing environment (these devices are known as PvC devices). WebSphere Transcoding Publisher helps you accomplish this without requiring you to redesign or reformat your existing Web site content. WebSphere Transcoding Publisher makes it easy to extend your existing Web content to pervasive computing devices because it handles the following issues:

- Different PvC devices have different size screens, therefore the content needs to be scaled for that device.
- Different PvC devices require that the content be represented in different formats. For example, the formats include:
  - Wireless Markup Language (WML).
  - Handheld Device Markup Language (HDML).
  - VoiceXML.
  - Compact HTML or cHTML (used by i-mode devices).
- Web content sometimes needs to be simplified, for instance by removing objects or features not supported on the PvC device; Web content may also be customized, for example by converting tables to lists.
- Networks have different bandwidth constraints, requiring the content to be fragmented before transmission to the device.
- Users may have different preferences (new function in WebSphere Transcoding Publisher Version 4.0) which includes restrictions of the types of content sent their devices.
- XML content needs to be formatted for desktop browsers and PvC devices.
- There may need to be image manipulation for particular screen sizes or file sizes; WebSphere Transcoding Publisher can also deal with color choice constraints, image elimination or converting the image to a link.
- There may need to be text translation from one language into another.

By dealing with these issues, WebSphere Transcoding Publisher makes the transformation of existing Web content to a PvC device content a fast and easy process. WebSphere Transcoding Publisher also provides a single point for transforming content for rendering on various devices. Using WebSphere Transcoding Publisher minimizes the expense of redesigning content, moving or modifying existing data representations and modifying applications in order to extend your existing Web content to support the mobile user.

WebSphere Transcoding Publisher accomplishes these content transformations by using the transcoding technology, which has the ability to make Web content available to a variety of devices, including wireless, wired and PvC devices. Transcoding provides the target device with content tailored to its capabilities and condensed to meet the network bandwidth constraints. Each WebSphere Transcoding Publisher transcoder has specific abilities for transforming or manipulating a given input type to a modified or a different output type.

WebSphere Transcoding Publisher consists of a set of integrated components which provide a flexible, extendable foundation to support all your content transformation needs. Figure 4-1 on page 116 depicts the general WebSphere Transcoding Publisher componentry, which includes (starting from the bottom of the figure):

- Administration - services provided by the Administration Console for configuring and managing transcoding server models, administering profiles, resources and settings, and working with log and trace facilities.
- Transcoding framework - provides common rules and services for a wide variety of transcoders and also provides capabilities so you can create custom transcoders to fit your particular needs and plug them into WebSphere Transcoding Publisher.
- Developer Toolkit - various tools which enable the developer to work with and extend WebSphere Transcoding Publisher facilities.
- Profiles - contains information about various types of transformation preferences.
- Transformation plug-ins - WebSphere Transcoding Publisher provides various transcoders to transform content. These transcoders provide various image and markup transformations.
- Third party transcoders - transcoders provided by you can be plugged into the transcoding framework.

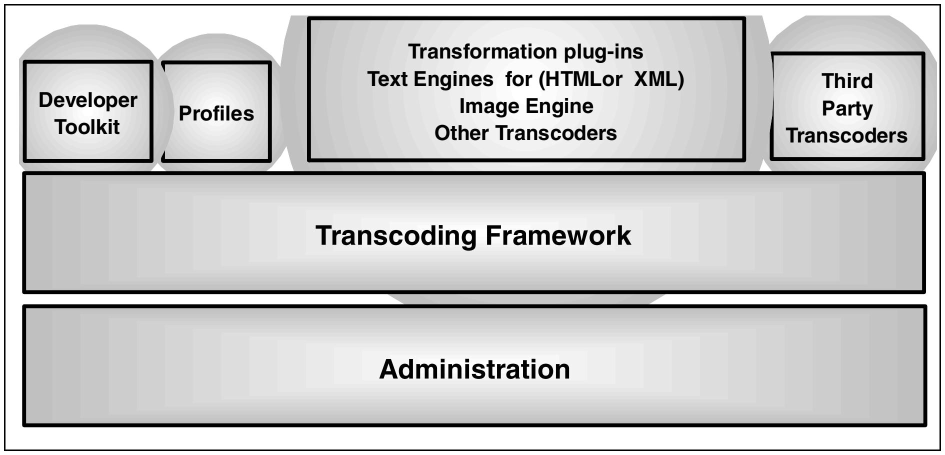


Figure 4-1 WebSphere Transcoding Publisher structure

From Figure 4-1, you can see that WebSphere Transcoding Publisher consists of a set of services, tools and capabilities that are accessible to the developer, the administrator, and the various transcoders. For example, transcoders can access profile information in order to generate the appropriate response for a particular device. WebSphere Transcoding Publisher also provides documentation, including a *Developer's Guide*, an *Administration Guide*, JavaDoc for the APIs and samples which show how to use various aspects of WebSphere Transcoding Publisher.

You may be wondering how WebSphere Transcoding Publisher accomplishes the transformation of content to fit the needs of the target device. Figure 4-2 on page 117 depicts a simplified flow of an HTTP request through WebSphere Transcoding Publisher into the Web application and the HTTP response flow from the Web application through WebSphere Transcoding Publisher and back to the device. In this case, WebSphere Transcoding Publisher is acting as a proxy between the network and the Web service.

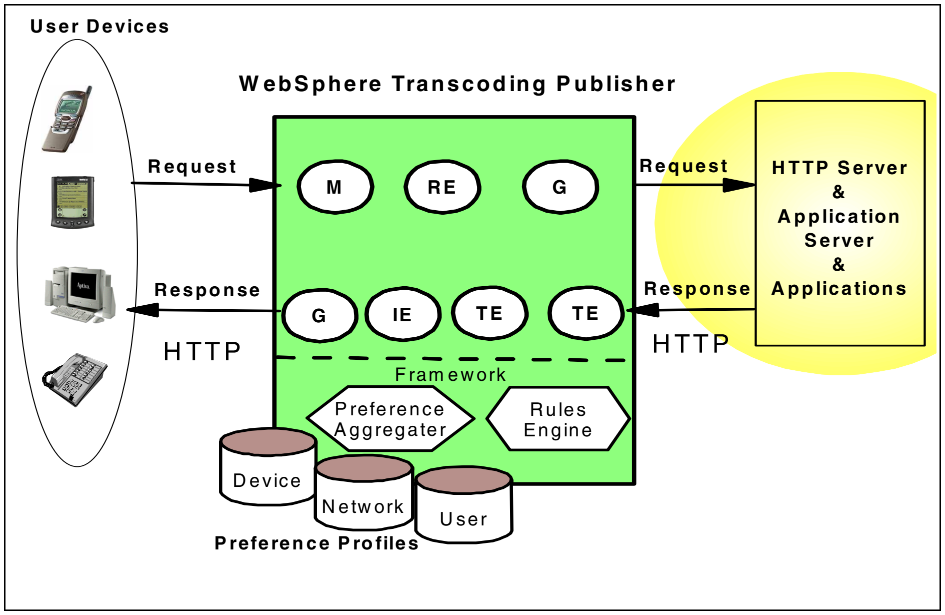


Figure 4-2   WebSphere Transcoding Publisher flow

On the left of Figure 4-2 on page 117 is a subset of the possible devices that can interact with the application environment. WebSphere Transcoding Publisher (in the middle) intercepts the HTTP requests, passing those requests through Request Editors (RE) and a particular Generator (G) before forwarding the request to the application environment (on the far right).

The HTTP header, which is input to WebSphere Transcoding Publisher, contains the requesting device, user ID, and network information. With this information, WebSphere Transcoding Publisher transforms the content for the device characteristics, the network capacity, and the user preferences (if a user preference exists for this user).

Once the application environment has formulated an HTTP response, that response is passed to WebSphere Transcoding Publisher. WebSphere Transcoding Publisher uses information in the HTTP header (information such as the requesting device, the user ID and the network) to determine how to transform the content for the target. WebSphere Transcoding Publisher accomplishes this by executing the appropriate Text Editors (TE) and Image Editors (IE). The appropriate Text Editors and Image Editors are determined based on the HTTP header information, Profiles, Rules and the Content type(s). The Text Editor(s) transform the content into the form needed by the target. The Image Editors manipulate the images as needed for the target.

Each of the transformation components within WebSphere Transcoding Publisher provides a unique service:

- Monitors - collect information about the HTTP request.
- Request Editors - can modify the HTTP request fields and possibly redirect the request by changing the URL.
- Generators - (a single generator will execute per request) can convert a request into a response.
- Text Editors - can customize the HTTP responses content. Multiple Text Editors may execute to transform the content.
- Image Editors - manipulate the images to fit the needs of the device.

The framework provides the common services used by each request and response. The rules engine and the preference aggregator are two of those services. The rules are processed by the rules engine based on information from the request and response. The preference aggregator resolves contradictory values in the associated preference profiles (device, network and user). For more details on the WebSphere Transcoding Publisher Architecture, read *IBM Transcoding Publisher Architecture* at `http://www.ibm.com/software/webservers/transcoding/library.html`.

4.2 What's new in WebSphere Transcoding Publisher Version 4.0

WebSphere Transcoding Publisher 4.0 provides many new features. The new features have been grouped into categories and are listed below:

- New transcoders/customizations:
  - The VoiceXML transcoder converts HTML to VoiceXML for voice applications.
  - The Machine Translation transcoder works with WebSphere Translation Server to translate content into different languages.
  - The Palm transcoder converts HTML to PalmOS HTML for display on Palm VII devices.
  - There is support for Cocoon specifications within the XML documents. That is the media tag which refers to the source device type making the request.
  - WebSphere Transcoding Publisher 4.0 provides support of user preference profiles for determining transformation processing. The

WebSphere Personalization interface is supported as a means to obtain user preference information.

- New deployment models:
  - The WebSphere Transcoding Publisher server can be deployed as a plug-in WebSphere Edge Server Caching Proxy, IBM's high-performance caching proxy. This configuration replaces the proxy with an external cache. This replaces the Network proxy.
- New and enhanced tools:
  - The External Annotation Editor makes it easier to create external annotation files to customize the information returned to small devices. Several other improvements extend the functions you can perform with annotations.
  - The Steeliest Editor makes it easier to create your own stylesheets to convert XML documents. You can specify, within an XML document, the names of one or more stylesheets to be applied to that document.
  - The Request Viewer now executes as an independent tool to monitor a remote WebSphere Transcoding Publisher server that is deployed as a proxy or reverse proxy.
- Enhanced administration:
  - The Administration Console and the server are installed as separate components.
  - You can import and export WebSphere Transcoding Publisher configuration information to XML (configuration file) to copy configurations from one machine to another. This allows you to perform mass distribution of the configuration to multiple servers. Also, you can modify the XML configuration file as needed.
  - You can search within your stylesheet selectors to find the resources you want to work with.
  - WebSphere Transcoding Publisher can incorporate user preferences when it is operating in an environment where users are identified and user preference information is available, such as the WebSphere Everyplace(TM) Suite.
  - The Administration Console and other interfaces use JavaHelp, which enables you to search for the information you need.

- General enhancements:
  - WebSphere Transcoding Publisher adds a variety of improvements to the usability and accessibility of the user interfaces and to the performance of the WebSphere Transcoding Publisher server.
  - JavaBeans have been enhanced to share a common preference bundle by passing in the HTTPPreferenceAggregatorBean.

4.2.1 Models

The transcoding server can be deployed in your network structure in a variety of ways. The server models are:

- Stand-lone proxy
- Reverse proxy
- IBM WebSphere Edge Server Caching Proxy
- WebSphere Application Server (WAS) filter
- JavaBean component(s) within your application

Let us take a moment to look at these deployment models.

Stand-alone proxy - this runs WebSphere Transcoding Publisher as a single service (outside the firewall) that tailors content coming from many different Web servers. The stand-alone proxy intercepts HTTP requests and responses and performs the appropriate content changes and transformation as they flow between the user and the Web server. The model is suitable for users who have the ability to configure a proxy device in their browsers (some wireless devices do not provide this capability). This is a good choice when caching; HTTP 1.1 or SSL endpoint support are not needed.

Reverse proxy - WebSphere Transcoding Publisher behaves like a Web server, in that it forwards user requests for content to the servers it can access. WebSphere Transcoding Publisher transforms the response formulated by the server before forwarding the (transformed) response to the client. To ensure that subsequent requests pass through WebSphere Transcoding Publisher, WebSphere Transcoding Publisher performs URL rewriting. When WebSphere Transcoding Publisher runs as a reverse proxy, no special configuration is required by the user.

IBM WebSphere Edge Server caching proxy - here, WebSphere Transcoding Publisher 4.0 is integrated into the Edge Server (ES) where WebSphere Transcoding Publisher runs as a WebSphere Edge Server plug-in. WebSphere Edge Server provides the caching support for WebSphere Transcoding Publisher, thereby allowing transcoded results to be cached for fast access.

WebSphere Transcoding Publisher takes advantage to WebSphere Edge Server caching to minimize redundant transcoding of frequently accessed content. This approach provides the benefits of combining the HTTP 1.1 protocol, caching support in the Edge Server (enhanced performance for cacheable content) and WebSphere Transcoding Publisher's transcoding capabilities.

WebSphere Application Server (WAS) filter - running WebSphere Transcoding Publisher as a WebSphere Application Server filter allows for transcoding the content at the source (the WebSphere Application Server application). In this model, WebSphere Transcoding Publisher is configured as a servlet, so that it can be incorporated into WebSphere Application Server. When the servlet is invoked, the content transformation occurs. WebSphere Transcoding Publisher servlets operate within the security context of the application server, allowing transcoded information to later be encrypted before being sent to the client.

JavaBean components - using the WebSphere Transcoding Publisher transcoders as JavaBeans allows close integration of specific transcoders with applications. Specific transcoders can be separated from the framework and run independently as JavaBeans. This means that other server programs, such as servlets, independent content-providing programs or JavaServer Pages (JSP) can invoke single transcoders (represented as JavaBeans) directly. The following WebSphere Transcoding Publisher transcoders can be provided as JavaBeans:

- ImageTranscoderBean - converts images
- HtmlReducerBean - simplifies HTML documents
- ImodeBean - converts HTML to c-HTML
- HdmlBean - converts HTML to HDML
- WmlBean - converts HTML to WML
- XmlHandlerBean - converts HTML to XML
- HtmlDomBean - creates a DOM from HTML

When using the WebSphere Transcoding Publisher JavaBeans, you must provide a way to detect the user device type and invoke the appropriate WebSphere Transcoding Publisher JavaBean(s).

More details on the stand-alone proxy, reverse proxy, WebSphere Application Server filter and JavaBean component deployment models are found in the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5,* SG24-6233.

Using WebSphere Transcoding Publisher as an Edge Server plug-in is discussed in the *WebSphere Transcoding Publisher Developer Guide* found at `www.ibm.com/software/web servers/transcoding/library.html.`

4.2.2 Resources

WebSphere Transcoding Publisher has four kinds of resources; each one plays a unique role in content transformation. The Administration console allows you to work with these resources. The four WebSphere Transcoding Publisher resources are:

- Preference profiles
- Annotators
- Transcoders
- XML stylesheets

All these resources can be included in the XML configuration data and are viewable using the Administration Console.

Preference profiles

IBM WebSphere Transcoding Publisher makes use of various profiles which enable customized transformation of content to meet the needs of the device, the network and user preferences. There are three types of preference profiles available for use with WebSphere Transcoding Publisher 4.0, which are:

- Device profiles - these contain specific characteristics for each device, including Windows CE devices, Palm devices, Wireless Application Protocol (WAP) phones, i-mode phones, HDML phones, traditional browsers (Netscape and Internet Explorer), XML-capable desktop browsers, and a default device.
- Network profiles - these contain specific characteristics for each network; this includes wireless networks, dial networks, and a default network.
- User profile (new with WebSphere Transcoding Publisher 4.0) - this includes group or user preferences (utilizing the WebSphere Personalization capabilities, which are part of WebSphere Everyplace Server). A user preference could specify that images not be displayed on their device even though the device is capable of displaying them.

WebSphere Transcoding Publisher uses these profiles as input for the content transformation process. When processing an HTTP response, WebSphere Transcoding Publisher matches the device, use ID and realm name in the HTTP header against the WebSphere Transcoding Publisher preference profile information. With the selected (match) information, WebSphere Transcoding Publisher performs the appropriate transformation(s) to generate the formatted content. If no specific profiles match the information in the HTTP header, default profiles are used.

The profile matching occurs in this order:

1. Specific user profile
2. Specific network profile
3. Specific device profile
4. Default user
5. Default network
6. Default device

If more than one profile matches (user, network, and/or device), various transcoders will be applied to the content based on the combined profiles. WebSphere Transcoding Publisher provides profiles for several common PvC devices and for several network types. Default profiles are also provided by WebSphere Transcoding Publisher. The profiles are listed in the Resource window of the Administration Console. By expanding the resource tree, you can view the details of a particular profile. The user profiles are not available in the Administration Console.

Annotators

Annotators provide conditions (represented in the annotation language) for preprocessing a document within WebSphere Transcoding Publisher to better suit that document for transcoding. For example, only a subset of a particular document may be desired for a user device. Annotation instructions specify that the content within a particular document must be kept, removed or replaced with other content. Each annotator is associated with a specific document and contains detailed knowledge of the structure and content of that document.

Annotators are selected to process a particular document based on the URL contained in HTTP or the location of the document. Other HTTP field values, including device, user ID and network information, along with the preference profiles, can be used in annotator selection.

Annotators are used when the automatic transcoding does not produce the desired results. For example, transcoders may leave too much content for the user device because of excess content being processed by the transcoder. By selecting the desired content, an annotator simplifies the processing of the transcoder and ensures that the desired results are produced.

WebSphere Transcoding Publisher supports two types of annotators:

- Internal annotators - these are imbedded XML documents (annotations) within the HTML document. The annotations are represented as HTML comments. WebSphere Studio Version 4 can create internal annotations.

- External annotators - these are separate (annotator) files containing annotation conditions. These files are registered within WebSphere Transcoding Publisher using the Administration Console. The external annotators are created using the new WebSphere Transcoding Publisher 4.0 External Annotator tool.

Annotators are discussed in more detail in Chapter 5, “Text clipping” on page 145.

Transcoders

A transcoder is a program which transforms (modifies) the format and content of a document. WebSphere Transcoding Publisher provides various transcoders. A transcoder is selected to process a document based on criteria specified for that transcoder when it was created. The transcoders provided by WebSphere Transcoding Publisher include:

- Text editors - can translate an HTML or XML document to another markup language (for example WML or HDML) or can reformat content (for example, a table may be reformatted into a list).
- Image editors - change the image type (for example a JPEG image can be converted into a GIF or WBMP image) or change the image characteristics (for example the image size, coloring, scaling and quality).
- Fragmentation editor - breaks transformed documents into small units which are acceptable to the network and the target device.
- Special purpose transcoders - include the machine translation transcoder used with WebSphere Translation Server to translate content from one language to another.
- Reverse Proxy Cookie transcoder - solves the problem created by Web sites using the Set-Cookie HTTP objects, specifying a domain for one or more responses.
- HTML DOM Generator transcoder - creates a Document Object Model (DOM) from HTML documents. This enables you to create and apply text clippers to the DOM to create new documents containing a subset of the original document.

When a document goes through the transformation process, more than one transcoder may be executed to produce the desired results. Details on the WebSphere Transcoding Publisher transcoders are provided in the WebSphere Transcoding Publisher *Developer’s Guide* found at `www.ibm.com/software/web servers/transcoding/library.html`.

Stylesheets

eXtensible Markup Language (XML) has become the de facto standard for representing business data outside the convention data store. Business data does not lose its meaning when encoded in XML, because each tag is specific to a data element. Unlike HTML, XML does not contain any presentation tagging needed to display content within many browsers. eXtensible Stylesheet Language (XSL) provides the means to transform an XML document to other markup languages, including presentation markup languages like HTML, WML, and so on.

XSL is rules-based language, consisting of template rules which specify patterns for selecting specific content in an XML document. Each template rule defined within a stylesheet contains a rule body with the actions to be performed on the matching content. For example, the rule body can contain statements to retag or reformat the associated content.

WebSphere Transcoding Publisher not only provides the fundamental means to apply stylesheets to XML documents, but also supports stylesheet parameterization, internationalization, stylesheet encoding and caching.

Refer to Chapter 6, “Using stylesheets” on page 211 for details on stylesheet handling within WebSphere Transcoding Publisher.

4.2.3 Installation

WebSphere Transcoding Publisher Version 4.0 installation and configuration is very similar to the WebSphere Transcoding Publisher Version 3.5 installation and configuration; for details, see the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5*, SG24-6233.

During installation, WebSphere Transcoding Publisher Version 4.0 checks for previous releases of WebSphere Transcoding Publisher on the target system. If a previous release exist, a new window will appear in the dialog, identifying this condition. You can select a button to either go back, proceed or cancel this installation. If you proceed, a migration option is offered. The migration options are:

- `Yes, please save my configuration data and use it with the new version.`
- `Yes, please save my configuration data but do not use it with the new version.`
- `No, I do not want to save my configuration data.`

If you proceed, you will be asked for the file name of the migration file and the directory it is to be located in. You are given the choice to either save or cancel the operation.

In WebSphere Transcoding Publisher 3.5, a response file was created by invoking `instwin.bat` with the `-r` option. The response file can be used later to install on other machines using the same settings. WebSphere Transcoding Publisher 4.0 advises creating a response file using the Summary window in the install process by clicking the **Create response file** button. The subsequent window asks for the name and location of the response file. After installation is complete, the response file can be replayed using `instwin.bat` with the `-p` option.

The XSL Stylesheet Editor is installed as part of the WebSphere Transcoding Publisher 4.0 install process.

4.2.4 Administration and configuration

The Administration Console is the primary tool for administering your WebSphere Transcoding Publisher environment. The Administration Console adapts to your particular server configuration and the location of the configuration data. The Administration Console allows:

- Working with transcoder server models - WebSphere Transcoding Publisher supports various server models discussed in 4.1.2.
- Working with resources - WebSphere Transcoding Publisher resources are preference profiles, annotators, stylesheets and transcoders. You can:
  - Enable and disable resources
  - Add resources
  - Organize resources into folders
  - Modify preference profiles, annotators and stylesheets selectors
- Working with settings - modifying settings for central directory, firewall, notification, reverse proxy, Web application, and network port, as well as modifying request viewer hosts.
- Working with logging and tracing.
- Importing and exporting configurations.
- Refreshing the transcoding server.
- Starting and stopping WebSphere Transcoding Publisher.

The transcoding Server Setup wizard is part of the Administration Console. As stated earlier, the transcoding server can be configured to run as a reverse proxy and as a WebSphere Edge Server plug-in. The transcoding Server Setup wizard was changed to add these new server models. Other changes to the Server Setup wizard include:

- The caching configuration was removed (when WebSphere Transcoding Publisher is running as an Edge Server plug-in); caching is now handled by the Edge Server.
- Firewall and reverse proxy information is no longer required (when WebSphere Transcoding Publisher is running as an Edge Server plug-in or a WebSphere Application Server filter).

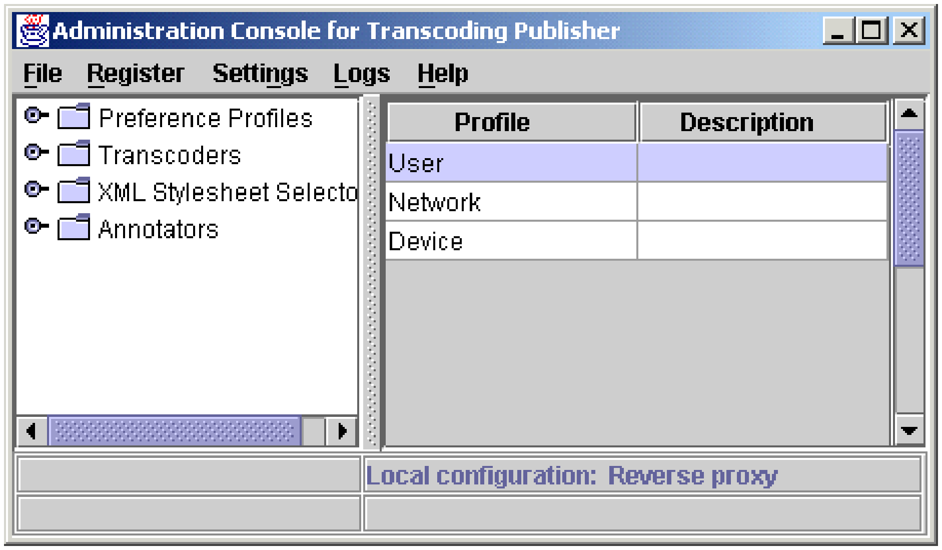
The Administration Console has a graphical interface. The main window has a toolbar with pull-down menus, two panes and a status area at the bottom. The two panes within the window are side by side with:

- The left pane containing a tree view of your resources, allowing those resource to be edited.
- The right pane containing specifics about the selected resource (from the tree view).

The status area (at the bottom of the window) is divided into four quadrants:

1. Upper left quadrant - displays the status messages
2. Upper right quadrant - displays the current server model.
3. Lower left quadrant - displays the approximate number of pending changes.
4. Lower right quadrant - (used when WebSphere Transcoding Publisher is using LDAP for persistent storage and there is a local server) displays the name of the server model for the local server.

The Administration Console is detailed in the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5,* SG24-6233, and in the *WebSphere Transcoding Publisher Administration Guide*.


Administration Console for Transcoding Publisher
File Register Settings Logs Help
Preference Profiles
Transcoders
XML Stylesheet Selecto
Annotators

Profile	Description
User	
Network	
Device	

Local configuration: Reverse proxy

Figure 4-3 Administration Console

4.2.5 XML configuration

WebSphere Transcoding Publisher now provides support for an XML-based configuration file containing your server configuration. Using a single XML configuration file to represent the WebSphere Transcoding Publisher configuration information simplifies configuration administration by:

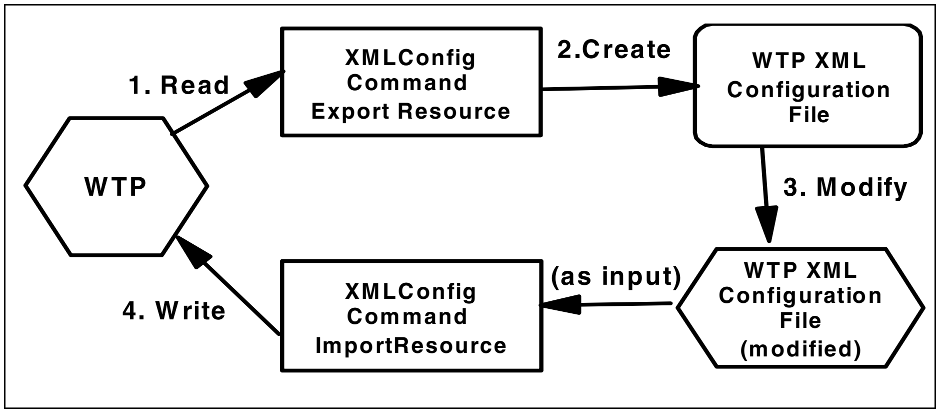
- Allowing for automated and batch updates to the WebSphere Transcoding Publisher configuration.
- Allowing remote deployment of WebSphere Transcoding Publisher server configuration.
- Simplifying service with a single configuration file.
- Using a single simple mechanism for migrating configurations across releases and across installs of a release.
- Providing an easy way to backup/restore WebSphere Transcoding Publisher configurations.
- Providing a report for registered resource.

Both the Administration Console and the install process support the XML configuration file.

The Administration Console's menu bar file option now contains Import, Export, and Export All options pertaining to the XML configuration file. The XML configuration file includes all resources but does not support secondary resources, such as stylesheets (.xsl files), annotators (.ann files) and Java plug-ins (.jar files). *The secondary files must be handled manually.*

The configuration file can be modified outside of WebSphere Transcoding Publisher, which allows adjustments to the configuration as needed. Figure 4-4 shows the XML configuration flow consisting of the following steps:

1. Use the `ExportResources` command to read the existing WebSphere Transcoding Publisher configuration.
2. A WebSphere Transcoding Publisher XML configuration file is created in the specified directory.
3. The WebSphere Transcoding Publisher XML configuration file can be modified as needed.
4. Use the `ImportResources` command to write the modified WebSphere Transcoding Publisher XML configuration into WebSphere Transcoding Publisher.



WTP
1. Read
XMLConfig Command Export Resource
2.Create
WTP XML Configuration File
3. Modify
WTP XML Configuration File (modified)
(as input)
XMLConfig Command ImportResource
4. Write

Figure 4-4 XML configuration flow

4.3 Tools

A suite of tools is provided with WebSphere Transcoding Publisher; these are classified into two categories: development and run-time tools. The development tools support the creation of annotators, stylesheets and profiles. The run-time tools support debugging and problem determination.

The tools are:

- Profile Builder
- External Annotation Editor
- Stylesheet Editor
- Transform Tool
- Request Viewer
- Administration Console

4.3.1 Profile Builder

The Profile Builder helps you create new preference profiles or modify existing profiles. WebSphere Transcoding Publisher comes with an number of preference profiles for the commonly used devices and networks. In the case these WebSphere Transcoding Publisher profiles do not meet your needs, you can create new profiles with this tool.

Profile Builder is like a wizard in that it steps you through the process of either:

- Creating a completely new profile,
- Creating a new profile from an existing profile, or
- Editing an existing profile.

You can create a profile for either a device or a network. As the Profile Builder steps you through the profile create/edit process, it provides descriptive text about that step. It is important to understand the details of a particular profile entry in order to create it correctly.

For details on the profile builder, see the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5,* SG24-6233, and the *WebSphere Transcoding Publisher Developer Guide* found at `www.ibm.com/software/webservers/transcoding/library.html`.

4.3.2 External Annotation Editor

The External Annotation Editor (EAE) enables you to create annotation files which can be used at run-time to simplify or modify an HTML document. Annotation files can be used to remove, replace, keep or modify the contents of an HTML document. Creating annotations files removes the need to write Java code to refine the document content being transformed. Using annotations can also make the overall transcoding process easier, because unnecessary content can be eliminated before transcoding occurs.

The EAE is a graphical tool; the main window is composed of six major areas:

1. Menu bar - provides access to application options (Files, Edits, Views, Annotation and Help); menu options can be disabled when the action is not available.
2. Standard toolbar - provides access to the common application actions and specific annotation tasks like Remove, Keep and Replace.
3. XPath toolbar - enables you to specify an XPath expression that is applied to the active HTML document (this toolbar is available when the active HTML document is in outline mode only).
4. HTML viewer - displays the active HTML document (the document currently loaded into the viewer); the document can be displayed in either Outline or Source mode. A document is loaded when the HTML document is opened.
5. Annotator viewer - displays the annotations you created for the HTML document displayed in the HTML viewer. These can be newly created annotations (created in this session) or an existing annotation file.
6. Attribute panel - displays attributes associated with the currently selected HTML document or annotation file. Attributes displayed in this area can be edited.

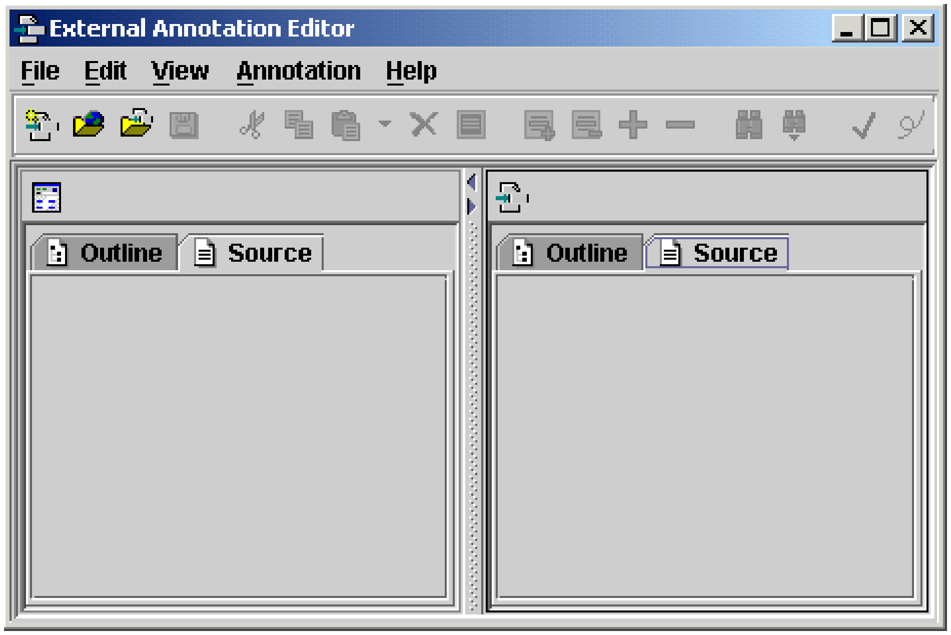


Figure 4-5 External Annotation Editor

Annotations are created by selecting specific elements within the HTML document and designating how that element is to be processed by applying a Keep, Remove, Replace or Add content action. Once the action has been selected, a wizard walks you through the details of completing the annotation statement, by asking where the annotation is to occur: either before or after the selected element. Once the annotation statements have been created, they can be saved in an annotation file. Each annotation file is registered with the transcoding server using the Administration tool. We will look at this tool in more detail in Chapter 5, “Text clipping” on page 145.

4.3.3 Stylesheet Editor

The XSL Stylesheet Editor enables you to create XSL stylesheets to transform an XML document into a document using other markup languages. Currently, the stylesheets created by the Stylesheet Editor transform XML to XHTML; other markup languages are forthcoming. The editor provides a WYSIWYG experience and is targeted at the novice XSL stylesheet creator. The tool does provide shortcuts for the experience user. The editor is a graphical tool and its main window is composed of six major areas:

1. Menu bar - provides access to the application options for Files, Edits, Views, Annotation and Help; menu options can be disabled when the action is not available.
2. Toolbar - provides access to the common application actions and specific stylesheet tasks.
3. Stylesheet display - provides views of the stylesheet in either Tree, Rule or Text format.
4. Output display - provides views of the input XML document transformed into XHTML with the stylesheet template rules applied. The XHTML can be viewed in Tree, Design or Text format.
5. Input XML document - provide views of the input XML document. The XML document may be viewed in either Tree or Text format.
6. Projects - provide a tree view of the projects defined.

The XSL Stylesheet editor is shown below.

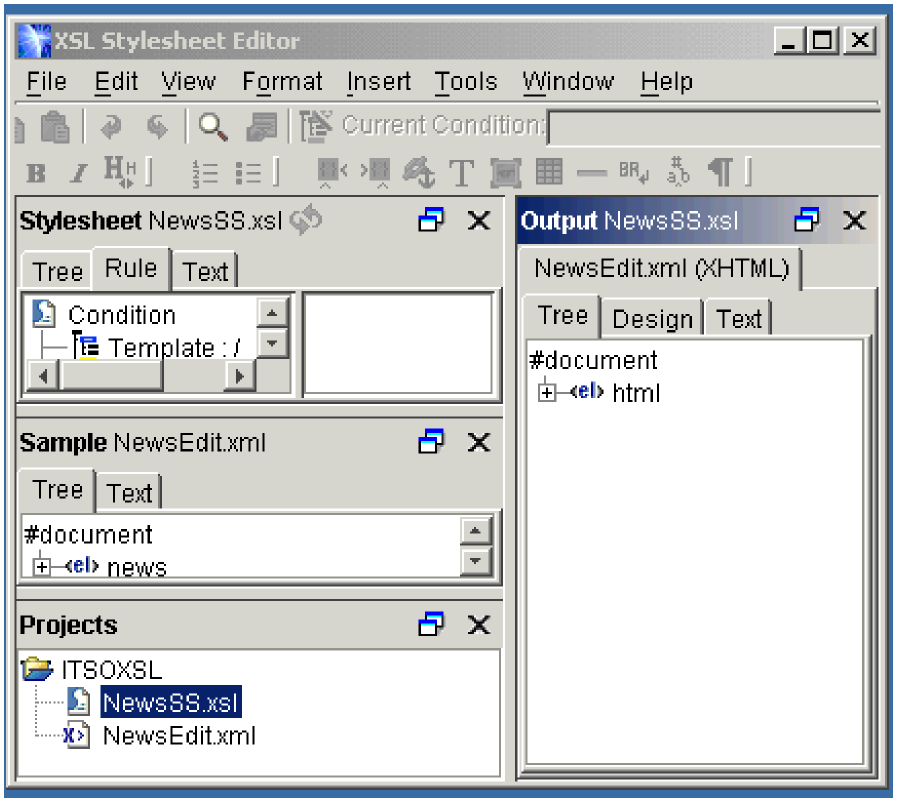


Figure 4-6 XSL Stylesheet Editor

The Stylesheet Editor is discussed in Chapter 6, “Using stylesheets” on page 211.

4.3.4 Transform Tool

The Transform Tool is a developer tool that allows you to preview the effect of applying various transcoding operations to a given document. The incoming document and the results are viewable side by side to see what changes have occurred. For example, you can see the original image in the left pane and the resulting reduced image in the right pane.The Transform Tool uses the preference profiles as the basis for performing the transformation operations.

With WebSphere Transcoding Publisher 4.0, the Transform Tool takes the user preferences as well as the other profiles into consideration.

The Transform Tool is graphical and the window consists of five main parts, which are:

1. Menu bar - provides access to the application option for file selection You can open either a image, HTML or XML document. The results can be saved as well.
2. Toolbar - provides access to the common file selection options graphically, as well as to the transcode action.
3. Pull-down menus - three pull-down menus allow you to specify the target device, target network and user preferences.
4. Input document - provides a view of the input document.
5. Output Document - provides views of the resulting output document.

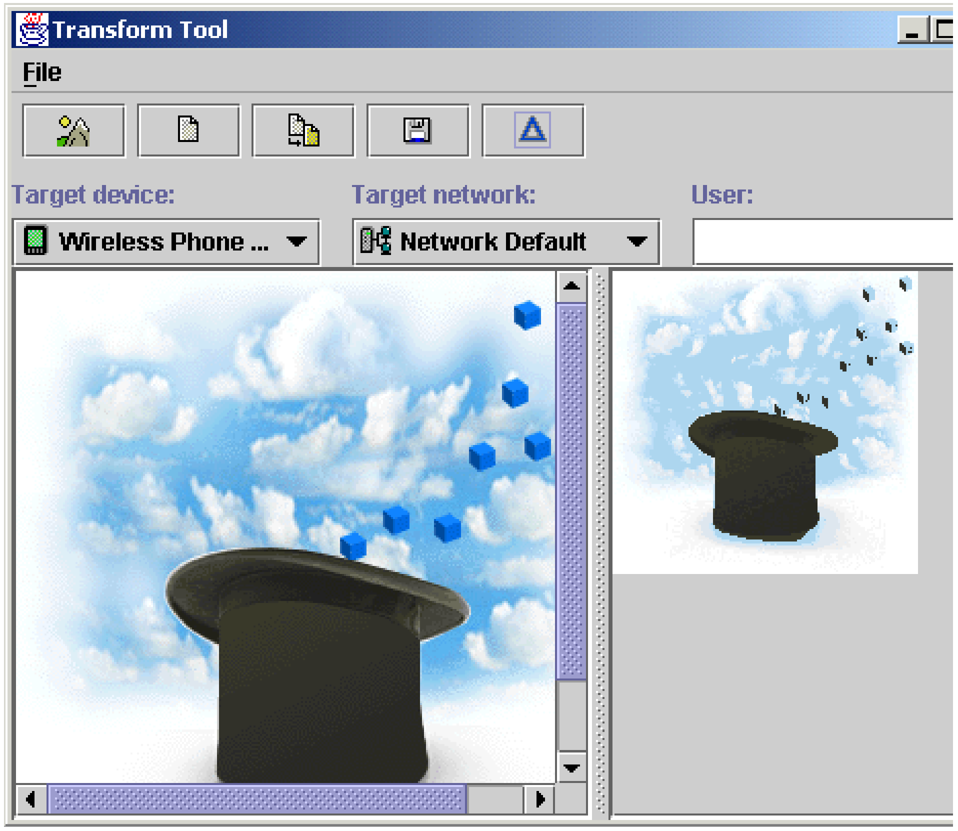


Figure 4-7 Transform Tool

For more details on the Transform Tool, see the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5,* SG24-6233 and the *WebSphere Transcoding Publisher Developer Guide,* found at `www.ibm.com/software/webservers/transcoding/library.html`.

4.3.5 Request Viewer

Request Viewer is a visual tool for monitoring the operation of the transcoding server when running in stand-alone mode. Request Viewer is particularly useful as a debugging tool, because it enables you to monitor the flow of requests through the server and to observe which transcoders are triggered and when. The request viewer displays the HTTP header and the content of each request as it is manipulated by the transcoders.

The main display for Request Viewer consists of these parts:

1. Menu bar - provides access to the application options for Files, Actions, Requests and Help; menu options can be disabled when the action is not available.
2. View options - two tabs specify view selections which are either Server Configuration or Request Processing views. When either of these views is selected, the associated content in the left main panel and the right main panel changes accordingly. The Server configuration view provides the transcoding server information, including the transcoders registered, preference profiles, and so on. The Request Processing view shows the flow of the request through the transcoding server.
3. The Server Configuration view subsets the window into three parts:
    a. Server Configuration - a list view of the contents of this configuration (including sublayers and plug-ins) in the top left pane.
    b. Server Configuration Details - displays the details of an item within the configuration. Content is displayed in this pane when an item is selected in the Server Configuration tree view. This is found in the top right pane, across from the Server Configuration.
    c. Output Messages - displays messages generated at execution time; this is found across the bottom of the window, below the Server Config and Server Config details panes.
4. The Request Processing view subsets the window into three parts:
    a. Request Processing - a tree view of the flow of a request through WebSphere Transcoding Publisher (in the top left pane).
    b. Transaction Header - displays the contents of the HTTP header. The content is displayed in this pane when an item is selected in the Request

Processing tree view. This is found in the right top pane, across from Request processing.

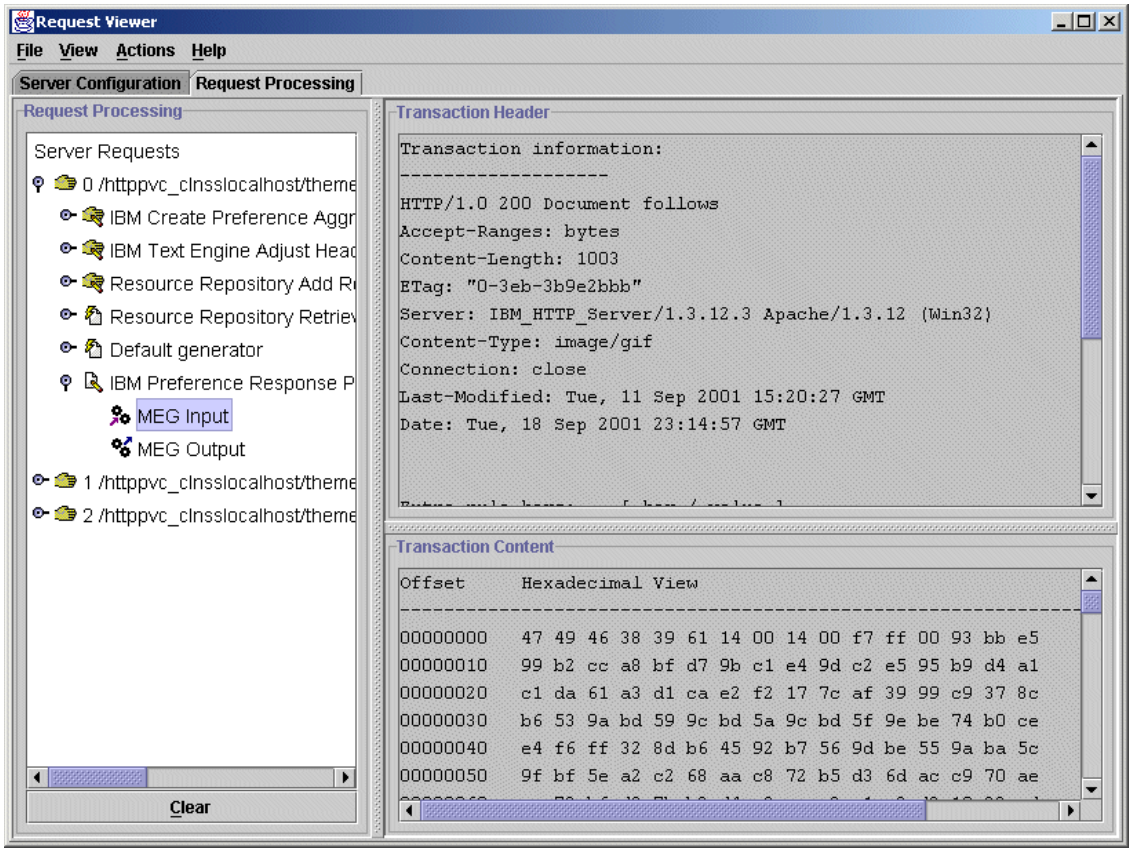
c. Transaction Content - displays a particular transcoder's input or output (below the Trx Header and in the left pane).

Remember also that when running the Request Viewer, you should not run the WebSphere Transcoding Publisher service, since both use the same ports. A sample Request Viewer request processing view is shown in Figure 4-8.

Start the Request Viewer tool by entering the following command from the IBMTrans directory:

```
RunTranscoding -g
```

Note: It is highly recommended that you use the Request Viewer tool to develop applications using transcoding. However, you should be aware that this tool is only available when running as a stand-alone reverse or forward proxy.


Request Viewer
File View Actions Help
Server Configuration
Request Processing
Request Processing
Server Requests
0 /httppvc_clnsslocalhost/theme
IBM Create Preference Aggr
IBM Text Engine Adjust Head
Resource Repository Add R
Resource Repository Retriev
Default generator
IBM Preference Response P
MEG Input
MEG Output
1 /httppvc_clnsslocalhost/theme
2 /httppvc_clnsslocalhost/theme
Clear
Transaction Header
Transaction Content

```
Transaction information:
--------------------
HTTP/1.0 200 Document follows
Accept-Ranges: bytes
Content-Length: 1003
ETag: "0-3eb-3b9e2bbb"
Server: IBM_HTTP_Server/1.3.12.3 Apache/1.3.12 (Win32)
Content-Type: image/gif
Connection: close
Last-Modified: Tue, 11 Sep 2001 15:20:27 GMT
Date: Tue, 18 Sep 2001 23:14:57 GMT
```

```
Offset     Hexadecimal View
--------------------------------------------------------------
00000000   47 49 46 38 39 61 14 00 14 00 f7 ff 00 93 bb e5
00000010   99 b2 cc a8 bf d7 9b c1 e4 9d c2 e5 95 b9 d4 a1
00000020   c1 da 61 a3 d1 ca e2 f2 17 7c af 39 99 c9 37 8c
00000030   b6 53 9a bd 59 9c bd 5a 9c bd 5f 9e be 74 b0 ce
00000040   e4 f6 ff 32 8d b6 45 92 b7 56 9d be 55 9a ba 5c
00000050   9f bf 5e a2 c2 68 aa c8 72 b5 d3 6d ac c9 70 ae
```

Figure 4-8   Request Viewer Request Processing view

For more details on the Request Viewer, see the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5,* SG24-6233, and the *WebSphere Transcoding Publisher Developer Guide* found at `www.ibm.com/software/webservers/transcoding/library.html`.

4.3.6 Device simulators

Many of the leading PvC devices manufacturers have created device simulators to allow developers to test their applications. These simulators provide a means to ensure that the application is generating device-usable content. However, testing with a real device should be part of the final testing process to ensure that the implementation works properly.

Wireless Application Protocol (WAP) is a specification for receiving content on phones. The Nokia Toolkit WAP simulator (Nokia Toolkit 2.1) has a built-in WAP browser. The toolkit is available at `http:www.nokia.com/corporate/wap/sdk.html`; you must register before downloading the toolkit. This is one of the simulators we will be using.

Openwave has implemented its own version of the WAP specification in the UP-browser. The simulator is available at `http://developer.openwave.com`.

compact HTML(cHTML) is used on i-mode devices. Wapprofit.com has created an i-mode simulator, which is available at the Wapprofit site at `http://www.wapprofit.com`.

Palm OS is used for handheld devices like PDAs. Palm OS provides many advanced features and functions beyond the capabilities of a browser. The Palm OS simulator is available at `http://www.palm.com/devzone/prose/seed.html`.

EPOC: while EPOC OS is relatively new, it runs on a variety of devices like the Nokia communicator and the PSION handhelds. The Ericsson R380 simulator looks like other WAP phone simulators, but it is running the EPOC OS.

Microsoft Windows CE is Microsoft's entry into the handheld device market. The Windows CE simulator runs on Windows NT and brings the Windows CE OS to the desktop.

4.4 Problem determination

WebSphere Transcoding Publisher provides two tools to help in problem determination; these are the tracing and logging facilities. Tracing levels are set in the Administration Console under the Menu bar's Logs entry.

The tracing facility allows you to see three message levels: low, medium, and high. Tracing provides detailed information for troubleshooting problems. Tracing is normally run at the low level. Tracing at higher levels is usually requested by service personnel.

The logging facility provides you with various messages. The messages types record different information:

- Informational - normal events.
- Warning - indicates possible problems.
- Error - indicates a definite problem requiring administrator attention.

Normally, tracing will be turned on at the low level and message login will be turned on. This enables you to monitor the behavior of WebSphere Transcoding Publisher without affecting its performance.

The trace files can be found in the*/IBMTrans/log directory; they are:

- LocalRegistryTrace.log - contains information about problems with the LocalRegistry, for example, problems with dynamic updates.
- TranscoderTrace#.log - helps identify all general product problems.
- The cmdmagictrace#.log - contains all information on the import/export configuration.

Note: For problems with WebSphere Transcoding Publisher as a WTE plug-in, check the WTE directory logs.

The Administration console can be launched using the **`AdminConsole`** command with a **`-d`** option that logs debug messages to the */IBMTrans/log/acDebug.log file. The types of problems that can be traced for the Administration Console are values not being saved as AdminConsole exits. Another source for Administration Console errors is the ConsoleTrace.log.

These tools are very useful when trying to perform WebSphere Transcoding Publisher-related application debugging.

- TracecoderTrace#.log - contains detailed information about the HTTP request and response processing.
- Using the Request Viewer - with this, you can look at all the transcoders inputs and output, plus the HTTP header content. Also, if you started the Request Viewer using the **`runtranscoder -g`** command, the DOS prompt provides stack trace information.
- Transform Tool - allows you to view the results of transformations to verify that you are getting what you expected before you use the transcoding server.

The RASCollect tool can be used to pull together all the information about the product state, trace, log, etc., into one file that can easily be sent to IBM service when necessary.

4.5 VoiceXML

WebSphere Transcoding Publisher 4.0 provides a new transcoder to transform HTML into VoiceXML. VoiceXML is an XML dialect which is used by voice browsers to transform the VoiceXML document into speech. WebSphere Transcoding Publisher supports the VoiceXML 1.0 Standards used by a variety of voice servers. VoiceXMl provides consumers the ability to access Web content through Interactive Voice Response (IVR) systems. Existing VoiceXML applications can call for HTML pages, which WebSphere Transcoding Publisher can change to usable VXML code, and provide a path back to the requesting application. Also, Using VoiceXML allows the users to have an interactive voice experience with the Web. In essence, you can hear textual content, choose from a list of links for the voice browser to follow, and prompt for or respond to voice input.

For more information about Voice XML in IBM WebSphere Transcoding Publisher, see Chapter 12, “Voice-enabled applications” on page 493.

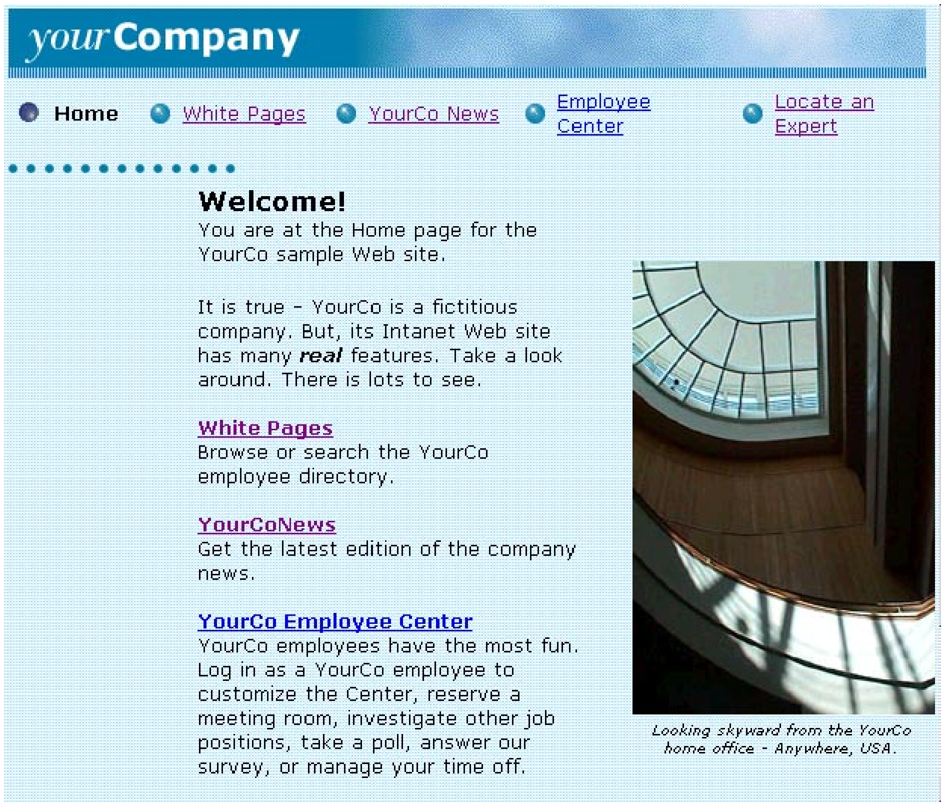
4.6 Fragmentation

Sometimes the content being sent to various handheld devices is not in small enough units for the device to handle. WebSphere Transcoding Publisher provides a transcoder (Fragmentation Transcoder) to break these oversized units into acceptable sizes for the target device. These smaller units are dynamically linked together so that as they are viewed on the device, the subsequent units can be retrieved from WebSphere Transcoding Publisher. *Fragmentation* is the term used to describe this activity, which is performed by the Fragmentation Transcoder.

For information about WML Fragmentation, see Chapter 7, “WML fragmentation considerations” on page 259.

4.7 Sample scenario

In Chapter 3, “Enterprise sample applications” on page 77, you were given a complete overview of the Your Company (YourCo) sample application that we are using within this book as the base application. In this scenario, we are going to see how WebSphere Transcoding Publisher transcodes the YourCo application’s main page. The YourCo application’s main page is shown in Figure 4-9 on page 141.



Welcome!
You are at the Home page for the YourCo sample Web site.

It is true - YourCo is a fictitious company. But, its Intanet Web site has many ***real*** features. Take a look around. There is lots to see.

White Pages
Browse or search the YourCo employee directory.

YourCoNews
Get the latest edition of the company news.

YourCo Employee Center
YourCo employees have the most fun. Log in as a YourCo employee to customize the Center, reserve a meeting room, investigate other job positions, take a poll, answer our survey, or manage your time off.

Looking skyward from the YourCo home office - Anywhere, USA.

Figure 4-9 YourCo main page

As you can see, this is not a exceptionally busy main page, but it is still much more than most pervasive devices want to handle.

4.7.1 The environment

The system environment consists of two machines. One is a Windows machine which contains the IBM HTTP Server, WebSphere Application Server Version 3.5.4, the B2E YourCompany application, and DB2 7.1 with the YourCompany data base installed. The other machine contains WebSphere Transcoding Publisher Version 4.0, all the WebSphere Transcoding Publisher tools and the WAP simulator.

For convenience, the transcoding server has been configured as a reverse proxy. The development environment is shown below in Figure 4-10.

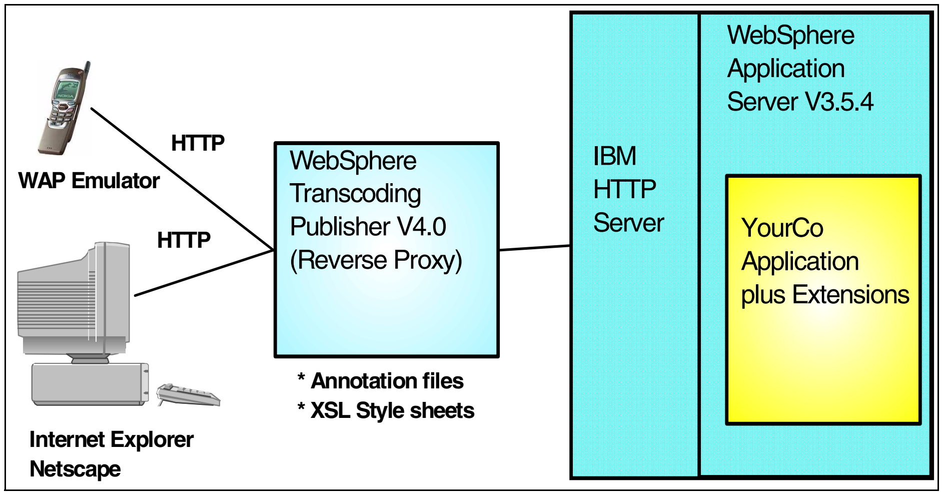

WAP Emulator
HTTP
HTTP
Internet Explorer
Netscape
WebSphere Transcoding Publisher V4.0 (Reverse Proxy)
* Annotation files
* XSL Style sheets
IBM HTTP Server
WebSphere Application Server V3.5.4
YourCo Application plus Extensions

Figure 4-10 The development environment

We added two extensions to the YourCo application for our scenario:

- The WebSphere Everyplace News application, to provide us with an XML document source
- The Locate Expert application, which is a Location based application.

4.7.2 Transcoding results

With the environment in place and using the Request Viewer to start the WebSphere Transcoding Publisher environment, from the WAP simulator we access the YourCo application through WebSphere Transcoding Publisher configured as a reverse proxy. WebSphere Transcoding Publisher (without customization) generates the content shown in Figure 4-11 below. As you can see, WebSphere Transcoding Publisher transformed the page, but there is excess content, making it hard to use the application. Using WebSphere Transcoding Publisher customization tools and techniques (discussed in the following chapters), you will learn how to customize the content for the target device.

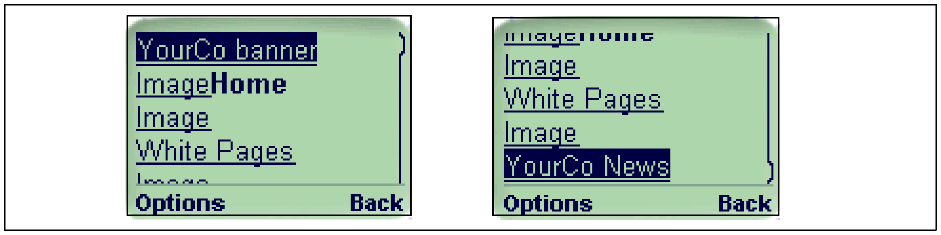


Figure 4-11 A portion of the menu bar

Figure 4-12 illustrates a portion of the Welcome window as provided by WebSphere Transcoding Publisher without any text clipping done for HTML content and without applying any stylesheets for XML content.

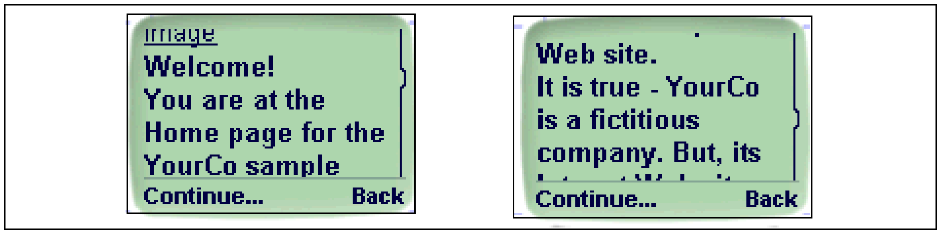
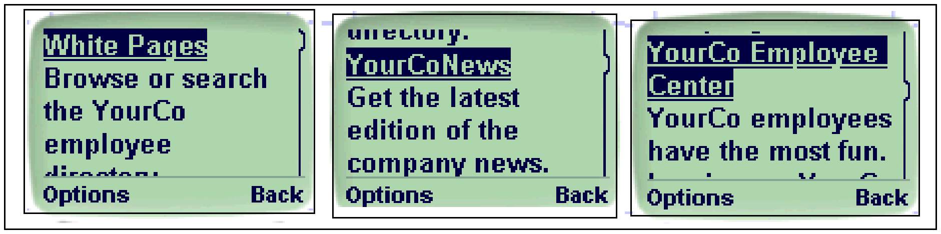


Image
Welcome!
You are at the
Home page for the
YourCo sample
Continue... Back

Web site.
It is true - YourCo
is a fictitious
company. But, its
Continue... Back

Figure 4-12 A portion of the Welcome window

In a similar way, a portion of the application detail options are shown in Figure 4-13.



White Pages
Browse or search
the YourCo
employee
Options Back

directory.
YourCoNews
Get the latest
edition of the
company news.
Options Back

YourCo Employee
Center
YourCo employees
have the most fun.
Options Back

Figure 4-13 A portion of the detail options

As illustrated in Figure 4-14, similar results are obtained for WML text provided by WebSphere Transcoding Publisher; it is obvious that the actual content needs to be customized for the WAP phone's small screen, using text clipping techniques. This process will be explained in Chapter 5, "Text clipping" on page 145.

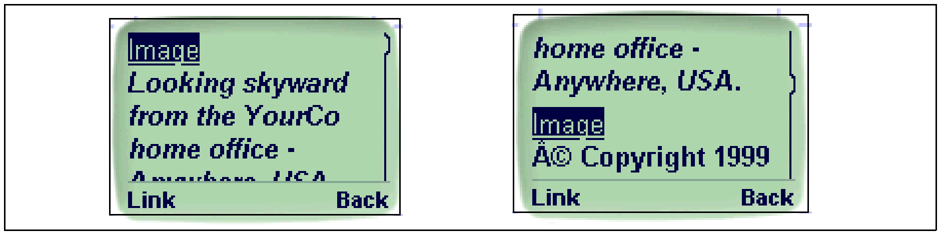


Image
Looking skyward
from the YourCo
home office -

Link Back

home office -
Anywhere, USA.
Image
Â© Copyright 1999

Link Back

Figure 4-14 The page footer

5

Text clipping

In this chapter, we discuss document clipping on IBM WebSphere Transcoding Publisher Version 4.0. Document clipping is used to simplify browser-based HTML pages which are intended for display on wireless and pervasive computing devices. The capability to perform document clipping was introduced in the Transcoding Publisher Version 3.5 with the annotation function. Prior versions of WebSphere Transcoding Publisher provided the ability to perform text clipping by creating custom transcoders using the Java language and Transcoder APIs. WebSphere Transcoding Publisher Version 4.0 extends the annotation capability by adding a new tool: the External Annotation Editor.

This chapter includes the following:

- An overview of document clipping
- Discussion of annotation
- Details on the new External Annotation Editor
- Examples of using WebSphere Studio Version 4.0 to create internal annotations
- An example of text clipping with Java
- An example of using the new Import and Export functions

5.1 Overview

In the previous chapter, we discussed why transcoding is needed to extend browser-based HTML content to Pervasive Computing (PvC) devices. However, as demonstrated in that chapter's sample application, using transcoding without any further instructions does not produce the customized content often needed for PvC devices. Text clipping and annotation allow you to refine the HTML content and strip that content of any extraneous information which is not vital to the user of the PvC device.

Document clipping is necessary because of the pervasive computing environment characteristics, which include:

- Limited screen real estate - PvC devices have very small screens which limit the amount of information displayed at any one time.
- Limited storage capacity - this prevents the devices from being able to store large pages.
- Network bandwidth limitations - the amount of information that can be sent to a PvC device is much less than is available to connected devices.

Document clipping is more than just a matter of removing content from an HTML page; it requires you to take into consideration the existing Web content, the content needs of the user and the interaction flow, which includes moving both forward and backward. Along with these issues, the following items must be considered:

- Review and understand the existing content.
- Understand what the PvC user needs most; what information is necessary for them to accomplish the task or activity they are trying to perform?
- What are the characteristics and considerations of the device?
- What are the limitations of the network?
- What is a meaningful way to organize the application content?
- What is the skill level of the user using the application and the device?

WebSphere Transcoding Publisher transcoding and document clipping techniques enable you to provide the types of customized content the users of PvC devices need and deserve.

There are two types of document clipping available within WebSphere Transcoding Publisher:

1. Annotations - additional instructions to pre-process an HTML page, making the page better suited for transcoding.
2. Text clippers - custom transcoders (written in Java code) that manipulate an HTML page either in its text form or as a Document Object Model (DOM).

The annotations and text clippers are customized to work with a particular HTML page and are usually intended to render specific content on specific devices. They do not support generalized transformation techniques like the WebSphere Transcoding Publisher text editors.

5.2 Annotation overview

Annotators provide a means to perform document clipping without understanding or creating text clippers, which is custom Java code. Text clipping was the primary clipping technique before annotations were defined. An annotation file consists of specific instructions used by an annotation engine to clip and customize an associated HTML document. The annotation instructions are a specific XML dialect defined for annotating (clipping) documents. The annotation instructions are in two forms:

1. External annotations - the annotation instructions are contained in an .ann file. Instructions provide both location information (the XPath to the associated HTML tag or content) and the action to be performed on that HTML tag or content. The annotation file applies to a specific URL or set of URLs for the associated HTML file.
2. Internal annotations - the annotation instructions embedded as comments within the HTML document. The annotation instructions provide the actions to be performed on the associated HTML tag or content.

External and internal annotations are processed by the WebSphere Transcoding Publisher annotation engine. The annotation engine outputs a clipped version of the HTML document. This clipped HTML document is then processed by the text editor(s). The annotation engine executes only once prior to the WebSphere Transcoding Publisher text editors, as shown in Figure 5-1.

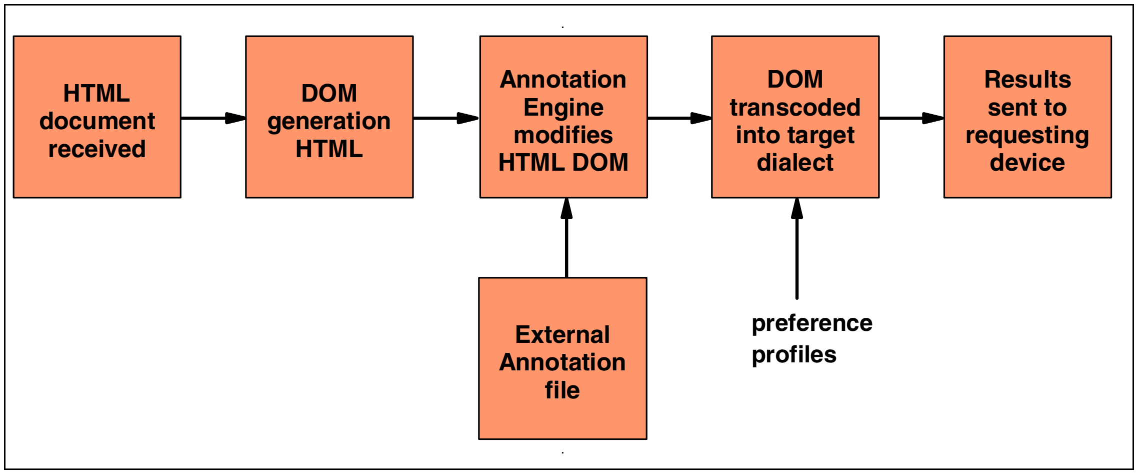

HTML document received
DOM generation HTML
Annotation Engine modifies HTML DOM
DOM transcoded into target dialect
Results sent to requesting device
External Annotation file
preference profiles

Figure 5-1   HTML document with Annotation processing flow

Annotation provides a quick and easy means of performing:

- – Content selection and attribute setting
- – Image and form replacement
- – Form or table reduction and reformatting

These activities are accomplished by taking advantage of a concept known as *clipping state*. Clipping state allows you to instruct the annotation engine (using the annotation instructions) on how the associated detail content is to be handled, that is, whether to keep or remove the content. Figure 5-2 on page 149 shows an example HTML file, clipping actions (keep or remove) and the results of these actions.

```
          Input HTML file
<HTML>                                          Annotations
  <H1>Document Clipping</H1>...............................</remove>
    <H2>External Annotation</H2>............................</keep>
      <P>Wow we have this new
            Clipping technique.</P>
    <H2>Internal Annotation</H2>
      <P>Lets look adding internal annotations.........</remove>
            using NotePad as the editor.
      <P>WebSphere Studio makes.........................</keep>
            internal clipping easy</P>
</HTML>
```

```
          Resulting HTML file
 <HTML>
    <H2>External Annotation</H2>
      <P>Wow we have this new
            Clipping technique.</P>
    <H2>Internal Annotation</H2>
      <P>WebSphere Studio makes
             internal clipping easy</P>
 </HTML>
```

Figure 5-2 Sample annotation activity

Let us look at a list of the new annotation instructions in WebSphere Transcoding Publisher 4.0:

– Insert markup - inserts rendered markup into a page by using a CDATA section inside the element.

– Conditional annotation - annotation(s) are applied based on device or network preferences.

– Fragmentation split points - tells the fragmentation engine where to break a deck or a card. Note that the fragmentation engine has the final say on where fragmentation occurs.

– Set preferences - allows dynamic setting of the preferences for this particular HTML document.

– Machine translation enhancements - overrides the default values for translation to achieve better translation results.

5.3 External annotation

External annotation allows you to create a separate file (.ann file) which contains the annotation instructions to perform document clipping on a particular HTML document. This approach is useful when the annotation author does not own the HTML document. The external annotation file consists of both the annotation markup action and the exact location (the XPath) of the associated HTML document tag or content to which the annotation instruction applies. Because the annotations are in a separate file, the XPath expression is needed to indicate to the annotation engine against which HTML element or content the annotation instruction is directed.

Let us take a moment to understand the basics of XPath before looking more closely at external annotations. XPath (a W3C specification) provides a syntax for defining the specific parts of an HTML or XML document. An HTML or XML document is hierarchica, with a root that contains all other elements; each element can contain other elements. There is no limit to the levels of nesting that can occur. Each element within this hierarchy (tree) has a unique path which defines it. Ultimately, XPath is an expression language for these hierarchical paths. Below, we show an HTML document and the associated XPath expressions (with the syntax as shown in an external annotation file).

Note: The HTML[1]/BODY[1] XPath expression is part of all elements within the BODY element. We did not include it because it appears on all sub elements of BODY.

```
           Input HTML file                    XPath expression
<HTML>............................................./HTML[1]
 <BODY>............................................/HTML[1]/BODY[1]
   <H1>Document Clipping </H1>....................................../H1[1]
     <H2>External Annotation</H2>..................................../H2[1]
     <P>Wow we have this new Clipping............................../P[1]
            technique. We can:
       <UL>............................................................................./UL[1]
        <LI>Make the world a better place by</LI>................/UL[1]/LI[1]
          <OL>......................................................................./UL[1]/LI[1]/OL[1]
            <LI>Making web content viewable on.................../UL[1]/LI[1]/OL[1]/LI[1]
                  handheld devices
            <LI>Opening the world to mobile workers............/UL[1]/LI[1]/OL[1]/LI[2]
            </OL>
        <LI>Solve world hunger</LI>................................../UL[1]/LI[2]
       </UL>
     </P>
     <P>External annotations are fun to create.</P>......../P[2]
 </BODY>
</HTML>

    Note: all XPath expressions within <BODY> have the same starting path of /HTML[1]/BODY[1]
```

Figure 5-3 XPath example

An external annotation file can be created either with an editor such as Notepad, or by using the External Annotation Editor. Our examples of external annotations are created using the External Annotation Editor.

5.3.1 The external annotation language

Before we look at the External Annotation Editor, let us look at the external annotation language. An annotation file is an XML documen, so it starts with the XML version statement and is followed by the root of the document, the element <annot version= 2.0>.

Each annotation instruction is defined in a <description> element. A <description> element consists of:

– condition=text - defines a condition that must be true for the annotation to be applied. This attribute is optional.
– take-effect=before or after - indicates whether the annotation is to occur before or after the target node.
– target=XPath expression - the target node as defined by the XPath expression.
– action element - the action to be performed.

There are various annotation action elements including <comment>, <contentlanguage>, <insertattribute>, <insertcomment>, <inserthtml>, <insertmarkup>,<keep>, <remove>, <replace>, <replacewithhtml>, <setpreference>, <splitpoint>, <table>, <wtsubject>.

Following is an example <description> element (annotation instruction) which removes the first Heading level 1 <h1> within an HTML document. Any HTML tags or content following this statement will be removed until either a Table tag is encountered or another annotation instruction is processed.

Example 5-1 External annotation description element

```
<description take-effect="before" target="/HTML[1]/BODY[1]/H1[1]"><remove/>
    </description>
```

In the above example, the `take-effect=”before”` means that this annotation will take effect before the <H1> tag is processed.

5.3.2 Using the External Annotation Editor

The External Annotation Editor is new within WebSphere Transcoding Publisher 4.0 and minimizes the education needed to create annotation files. Prior to using the tool, you would need to understand a variety of techniques (including XML syntax, XPath and the annotation language) before you could create an annotation file. With this tool, you can easily create annotations by selecting the HTML content you want to work with, selecting the action you want taken and stepping through the associated wizard to complete the creation of that annotation instruction. By using this process, the appropriate annotation instructions, including the XPath expression, are created. The Annotation Editor displays the HTML document and the corresponding annotation file side by side. Both the HTML document and the annotation file can be viewed in outline and source mode.

The other features of the annotation editor include:

– One step drag and drop - allows you to indicate individual nodes or regions of the HTML document that should be kept or removed.
– Wizards (smart guides) - once an action is selected, the associated wizard leads you through building the particular annotation instruction details.
– Synchronized DOM and source view - both the HTML and annotation document are provided in a DOM view or source view. This allows flipping between the outline (DOM view) and the source view and retaining the selections you have made between views.
– Full support for the annotation language version 1.0 constructs- this includes keep, remove, replace, insert comment, insert attribute, and distribute table headers.
– Support for most of the annotation language version 2.0 constructs - this includes insert HTML, replace HTML, insert rendered markup and splitpoints.

Getting Started with the Annotation Editor

Start the annotation editor from the Windows task bar, by selecting **Start -> Programs -> IBM Transcoding Publisher -> Toolkit -> Annotation Editor.** The annotation editor is shown in Figure 5-4 on page 154.

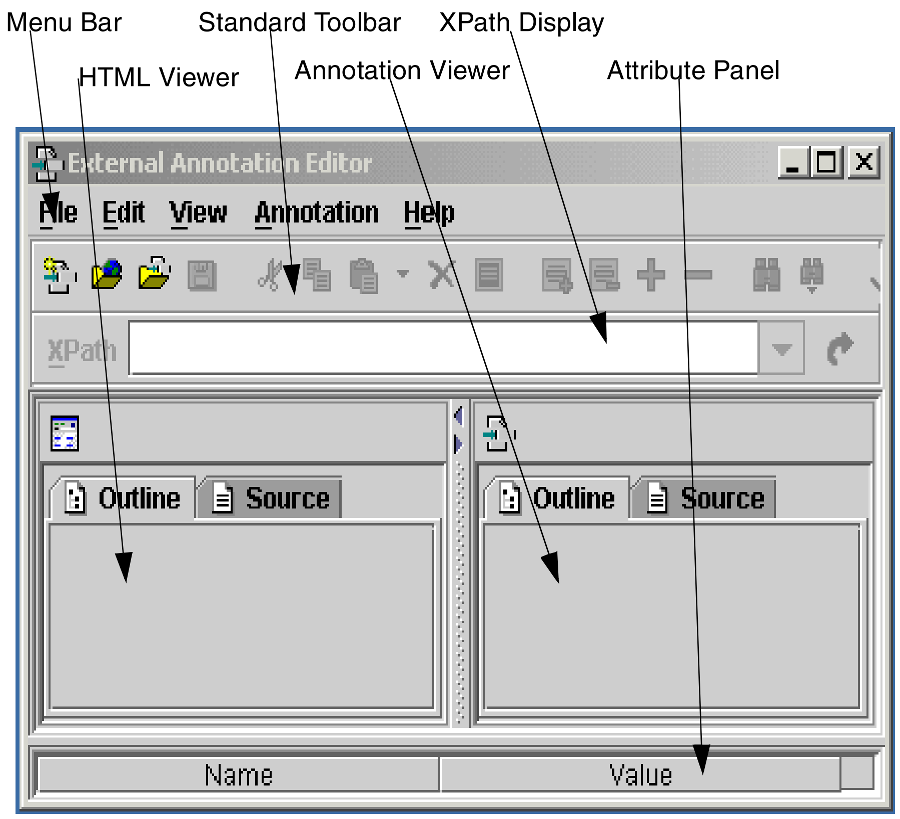


Figure 5-4   External Annotation Editor

The Annotation Editor window is composed of six major areas, which are:

1. Menu bar - provides access to general actions like file handling, edit commands, view options, annotation constructs, and help.
2. Standard tool bar - provides icons which represent commonly used edit commands and annotation constructs.
3. XPath display - an entry field allowing you to specify the XPath expression which is applied to the document to locate specific elements within the document. This is enabled when the HTML outline view is visible.
4. HTML Viewer - provides either a Outline or Source view of the HTML document being worked on.

5. Annotation Viewer - provides either a Outline or Source view of the annotation file being created.
6. Attribute Panel - is an optional display area which contains any attributes associated with a selected HTML element and allows you to modify the attribute values by clicking the value and typing in the new value.

Opening files in the Annotation Editor

You can open an HTML file using the menu's **File -> Open HTML** option or using the **Open HTML** icon from the toolbar. The Open HTML option allows you to either specify a URL for the HTML document or the location of the local document. Only one HTML document may be open at a time.

You can open an existing annotation file using the menu's **File -> Open Annotator** option or using the **Open Annotator** icon from the toolbar. To create a new annotation file, select the **New** menu option or use the **New Annotator** icon from the toolbar. You can either browse to get the directory location for the existing annotation file or specify the name of the new file.

Navigating Documents

As stated earlier, you can see the documents in either Outline or Source view. You must use the Outline view to edit the HTML file, but the Source view allows you to see the detail HTML structure and content. You can select (highlight) elements in the Source view and switch to the Outline view to edit that same element.

In order to expand or collapse the Outline views of the HTML document or the annotation file, you can use either the menu bar View options or select tool bar icons. The outline expand or collapse options are:

- Expand all elements
- Collapse all elements
- Expand a single element
- Collapse a single element

Searching within documents

The annotator editor has a search mechanism, which allows you to search for text in the current HTML document. From the menu select **Edit -> Find** or select the **Find** icon from the toolbar, then

1. Enter the search string.
2. Select the find options and the search direction (either forward or backward).
3. Click the **Find** button to start the search.
4. The first occurrence of the search string is highlighted in the HTML document

Creating annotations

Document clipping is the process of identifying the specific element(s) within an HTML document that need to be either kept, extracted or tailored to meet the needs of the client device. There are two key concepts to understand:

1. Clipping region - the area within the HTML document which has an associated annotation instruction. The area may consist of one or more elements of the HTML document.
2. Clipping state - the action to be taken by the annotation engine in that area of the HTML document. The two key clipping states are Keep and Remove.

Annotation instructions are created using the following steps:

1. select the HTML element(s) that are the target of the annotation instruction.
   - To select an element. either use the arrow keys or use mouse highlighting.
   - To select multiple elements, either hold down the Control or Shift key or click and drag the mouse to create a selection net.
2. Determine the clipping state.
   - Select the **Annotate** option from the menu bar, then select the desired clipping state, or
   - Select the clipping state from the toolbar.
3. A wizard (smart guide) will lead you through the process of filling in the annotation instruction. As an example, let us assume that you selected the **Keep** clip state.
   a. Specify where the Keep is to start, either **before** the specified HTML element or **after** the specified HTML element.
   b. Specify the Keep state, either **all** or **selected**; let us assume that you selected the **all** option.
   c. Click the **Finish** button.
4. Now the annotation instruction is created. Similar actions are taken for all clipping activity. Now you can either return to step 1 to continue the clipping process or move to step 5 to complete the activity.
5. To save your annotation file, from the menu bar select **File ->Save**.

In addition to the two clipping states (Keep and Remove), there are other annotation instructions. The general instruction types are:

- Replace content - replaces any element with either plain text or HTML.
- Insert comments - inserts comments in the output HTML document.
- Insert attribute(s)- inserts additional attributes in an existing HTML element.

- Insert HTML fragments - inserts HTML fragments into the HTML document.
- Insert rendered markup fragments - inserts rendered markup into the HTML document.
- Define splitpoints (fragmentation directives) - identifies where fragmentation should occur.
- Distribute table headers - identifies tables with headers which, if converted to lists, should have the table headers distributed over each row.

To create these instructions, use basically the same process as defined above; however, the wizards may have questions or options specific to that instruction.

The current version of the External Annotation Editor does *not* support these annotation instructions:

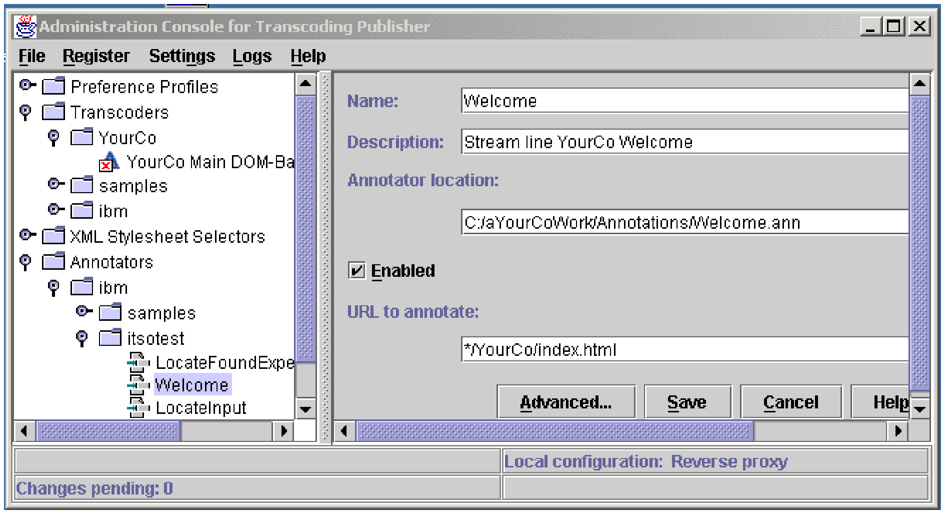
- Set preferences - modifies or extends the preference profile properties.
- Column - for clipping table columns.
- Row - for clipping table rows.
- Content language and wts subject - provides information for machine translation.
- Condition - sets conditions to control annotation.

These instructions will be supported in a follow on release or an update.

5.3.3 External annotation file administration

The WebSphere Transcoding Publisher administration console is the tool used to access all the resources with WebSphere Transcoding Publisher. For more details on the administration console, look at the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5*, SG24-6233.

Among other functions, the administration console is used to register your annotation files and to identify the conditions which determine when the annotation file is applied to the HTML page. A registered annotation file is shown in Figure 5-5 on page 158.



Name: Welcome

Description: Stream line YourCo Welcome

Annotator location: C:/aYourCoWork/Annotations/Welcome.ann

☑ Enabled

URL to annotate: */YourCo/index.html

Local configuration: Reverse proxy

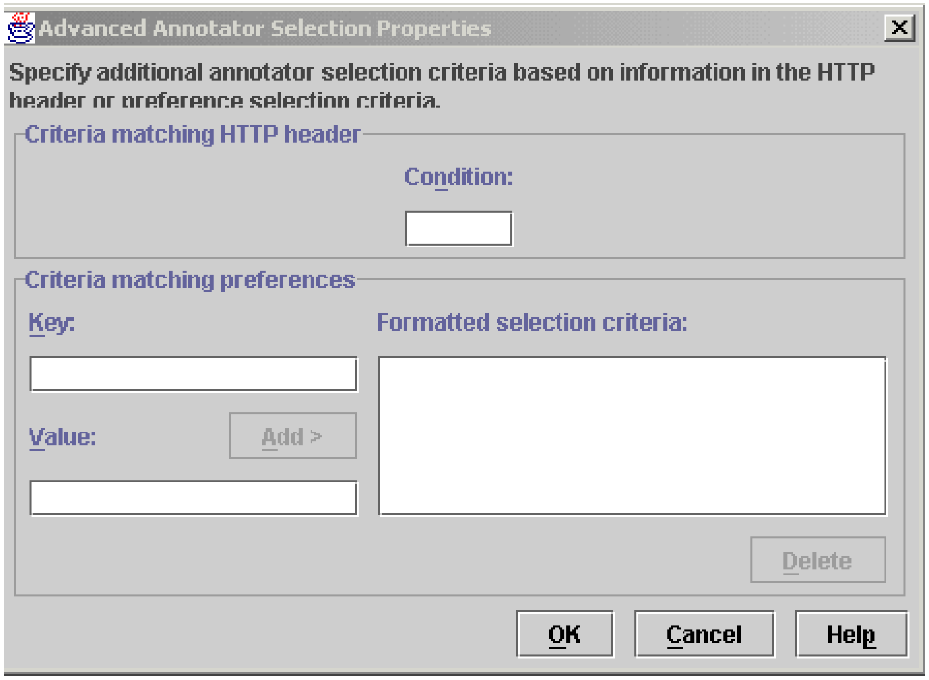
Changes pending: 0

Figure 5-5 Administration console with a registered annotation file

When you select an annotator in the Annotator subtree, the console displays all the basic settings relevant to it in the right pane. The information displayed includes:

- Name/ Description - the name and description of the annotator used within WebSphere Transcoding Publisher. Note that this is the annotator registration name (not to be confused with the actual annotation file name).
- Annotator location - the path (file path or URL) to the location of the annotation file.
- URL to annotate - the URL for the HTML file to be annotated.

One of the buttons at the bottom of this screen is the Advanced button, which invokes the Advanced Annotator Selection Properties window, shown below. This display allows you more selection criteria, by specifying **Criteria** within the HTTP header and/or **Criteria** within the preference profiles.


Specify additional annotator selection criteria based on information in the HTTP header or preference selection criteria.

Advanced Annotator Selection Properties
Criteria matching HTTP header
Condition:
Criteria matching preferences
Key:
Formatted selection criteria:
Value:
Add >
Delete
OK
Cancel
Help

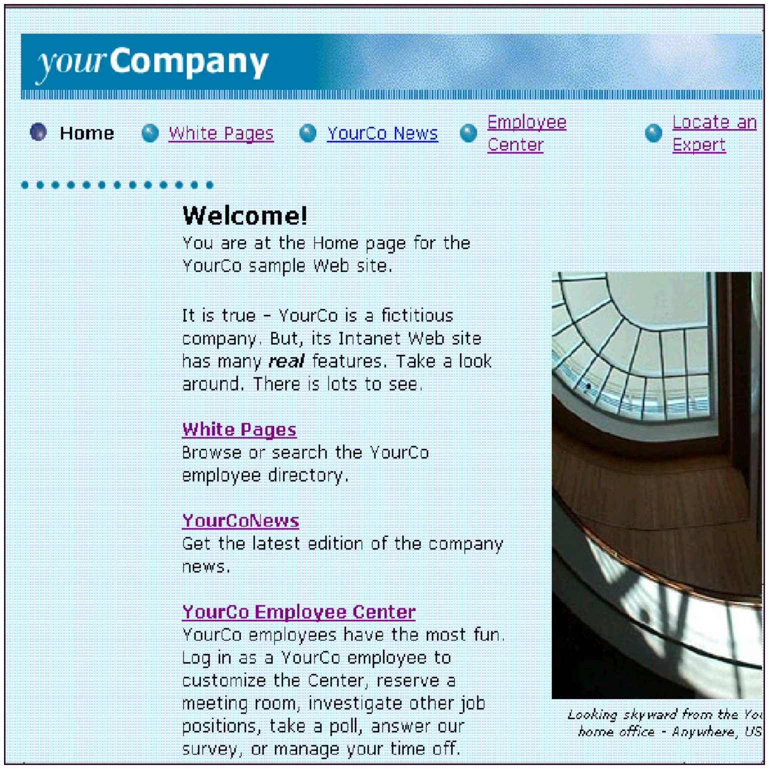
Figure 5-6 Advanced Annotator selection criteria

The selection criteria matching the HTTP header can be a complex formula using the AND or OR conjunctions as well as parentheses. It can read all the header attributes of the HTTP request. The criteria matching preferences requires key/value pairs (see Figure 5-6) that will match the key/value pair in one of the profiles (device, network, user).

5.3.4 Sample scenario: Locate Expert

We are using the Locate Expert application from the Your Company (YourCo) Web site. The Locate Expert application is a link off the YourCo home page. The objective of the Locate Expert application is to allow the user to locate the person with the selected expertise in the closest proximity to the requester. Starting at the home page, we will now walk through the Locate Expert application, as it appears on the Web.

On the home page, you can select the Locate An Expert link , as shown in Figure 5-7.



Welcome!
You are at the Home page for the YourCo sample Web site.

It is true - YourCo is a fictitious company. But, its Intanet Web site has many *real* features. Take a look around. There is lots to see.

White Pages
Browse or search the YourCo employee directory.

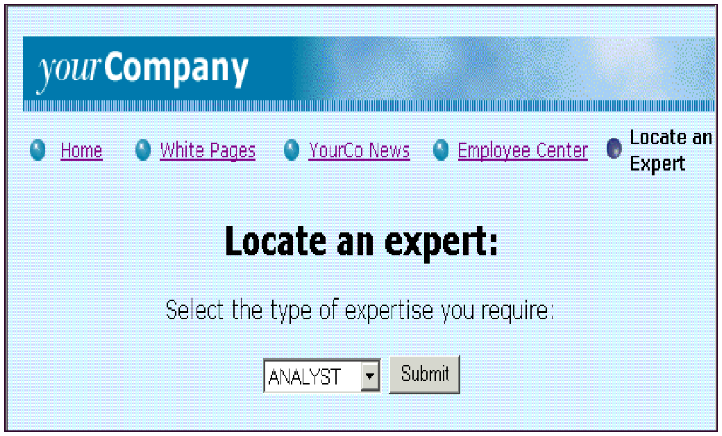
YourCoNews
Get the latest edition of the company news.

YourCo Employee Center
YourCo employees have the most fun. Log in as a YourCo employee to customize the Center, reserve a meeting room, investigate other job positions, take a poll, answer our survey, or manage your time off.

Looking skyward from the Yo home office - Anywhere, US

Figure 5-7 YourCo main page

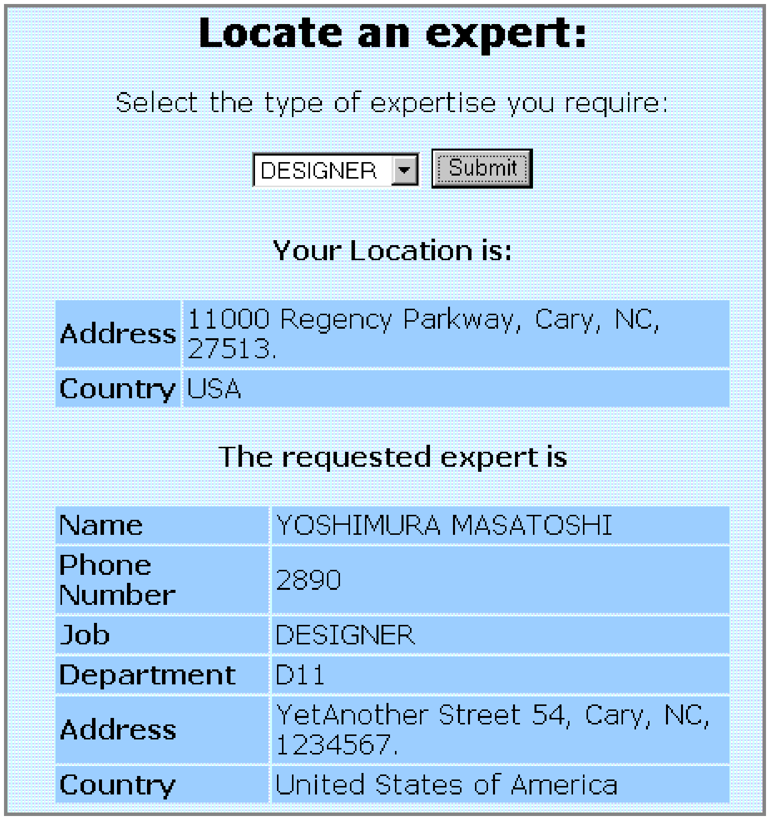
Next, you select the expertise from the list (see Figure 5-8) and click the **Submit** button. If an expert exists close to you, the expert's details, including name, phone number and other information appear in a table at the bottom of the screen.



Locate an expert:

Select the type of expertise you require:

Figure 5-8 Locate Expert Select


Locate an expert:

Select the type of expertise you require:

Your Location is:

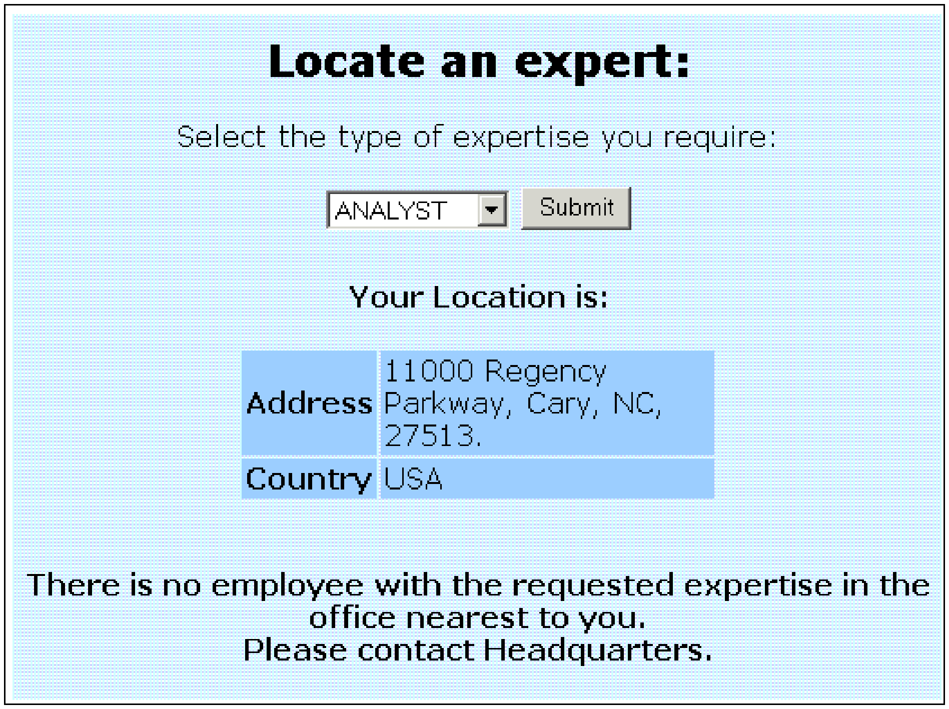
Address	11000 Regency Parkway, Cary, NC, 27513.
Country	USA

The requested expert is

Name	YOSHIMURA MASATOSHI
Phone Number	2890
Job	DESIGNER
Department	D11
Address	YetAnother Street 54, Cary, NC, 1234567.
Country	United States of America

Figure 5-9 Locate Expert - Expert Found

If an expert does not exist, the lower portion of thewindow displays a message, as shown in Figure 5-10.


Locate an expert:

Select the type of expertise you require:

Your Location is:

Address	11000 Regency Parkway, Cary, NC, 27513.
Country	USA

There is no employee with the requested expertise in the office nearest to you.
Please contact Headquarters.

Figure 5-10 Locate Expert - None found

The environment

We are using the same environment identified in Chapter 4, “Transcoding application content” on page 113, which consists of two machines; both machines use the Windows 2000 operating system. The development system has the IBM WebSphere Transcoding Publisher Version 4.0 and the Nokia Toolkit 2.1 (as the target PvC device) loaded. WebSphere Transcoding Publisher is configured as a reverse proxy. Because WebSphere Transcoding Publisher is running as a reverse proxy, no configuration of the Nokia toolkit is needed.

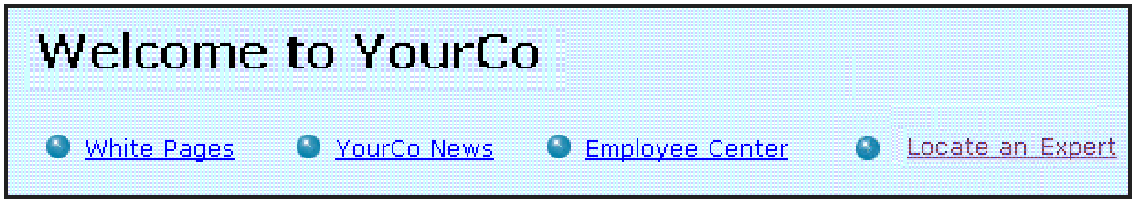
The other machine has IBM WebSphere Application Server Version 3.5.4, IBM HTTP Server Version 1.0 and DB2 Version 7.1 running. The YourCo application has been loaded, which creates tables in the database and includes application components within the application server.

First step: simplify

The first step in extending our Locate Expert application to the WAP device is to simplify it. This allows us to think through the target user's content needs and to test the application flow before creating annotations.

Our intended user is an experienced user of the YourCo Web site. Therefore, the wireless application will contain minimal instructions on how to use the application. Dealing with an experienced user greatly simplified the screen content but we still wanted to make the application as intuitive as possible.

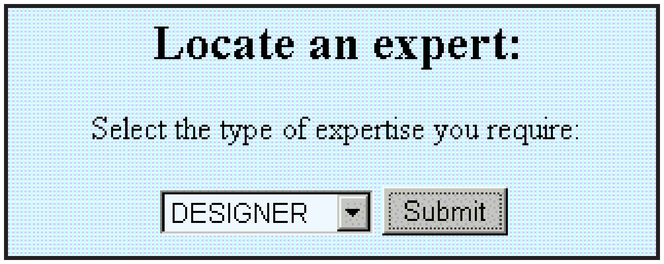
Let us look at the Locate Expert application with the simplified windows. We start with the YourCo home page, as shown in Figure 5-11.


Welcome to YourCo

White Pages YourCo News Employee Center Locate an Expert

Figure 5-11 Stripped YourCo home page

After selecting the Locate Expert Link, you will see the Locate Expert selection page, as shown in Figure 5-12.

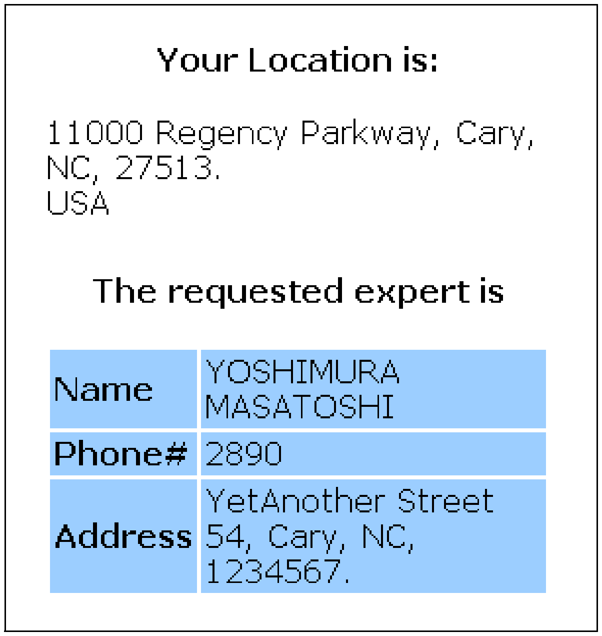

Locate an expert:

Select the type of expertise you require:

DESIGNER Submit

Figure 5-12 Stripped Locate Expert - selection

For the `Expert Found` display, we removed any unnecessary content to shrink the display as much as possible. Note we are showing in the display only the results portion of the screen.



Your Location is:

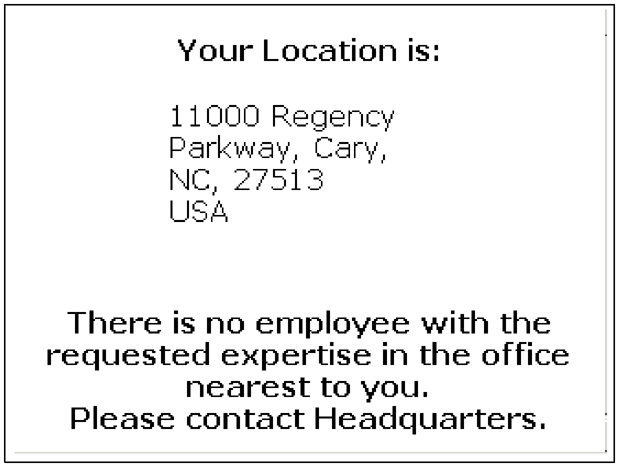
11000 Regency Parkway, Cary, NC, 27513.
USA

The requested expert is

Name	YOSHIMURA MASATOSHI
Phone#	2890
Address	YetAnother Street 54, Cary, NC, 1234567.

Figure 5-13 Stripped Locate Expert - Expert Found

If an expert does not exist, the message displayed is shown as in the figure below. Note that only the response portion of the screen is shown.



Your Location is:

11000 Regency
Parkway, Cary,
NC, 27513
USA

**There is no employee with the requested expertise in the office nearest to you.
Please contact Headquarters.**

Figure 5-14 Stripped Locate Expert - No expert found

First annotation - YourCo home page

Our first annotation activity was to simplify the YourCo home page. As seen from the simplification process, we want a simple YourCompany banner with a list of the various applications.

The YourCo home page index.html file consists primarily of three tables:

– Table 1 - contains two rows: a row with the YourCo banner and a row with another table containing the application menu bar. Our changes are:
  - Replace the banner with the text `Welcome to YourCo`
  - Remove the images from the application links

– Table 2 - contains the application index links with a description for each of the applications. We are going to remove the entire table.

– Table 3 - contains the site copyright and contact information. We are going to remove the entire table.

The HTML document for the YourCo home page is shown in the example below. We are not showing the contents of document head tables 2 and 3 because we will be removing them.

Example 5-2 YourCo home page HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
</head>
<body background="/theme/bg.gif">
<CENTER>
<TABLE width="620">
  <TBODY>
    <TR>
      <TD colspan="10"><IMG src="/theme/topBanner.gif" border="0" alt="YourCo 
banner" align="bottom" width="607" height="46"></TD>
      </TR>
    <TR>
      <TD colspan="10">
      <TABLE>
        <TBODY>
          <TR>
            <TD valign="middle" align="left" width="25"><IMG 
src="/theme/buttonDWN.gif" width="20" height="20" border="0"></TD>
            <TD valign="middle" align="left" width="50">
<FONT size="-1"><B>Home</B></FONT></TD>
            <TD width="5"></TD>
            <TD><IMG src="/theme/button.gif" width="20" height="20" 
border="0"></TD>
```

```
            <TD valign="middle" align="left" width="100"><A
href="/WebSphereSamples/YourCo/Search/frameset.html" target="_top"><FONT
size="-1">White Pages</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/theme/button.gif" width="20" height="20"
border="0"></TD>
            <TD valign="middle" align="left" width="100"><A
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.ExpHTMLServlet.Expiring
HTMLServlet/TheExpiringHTMLServlet" target="_top"><FONT size="-1">YourCo
News</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/theme/button.gif" width="20" height="20"
border="0"></TD>
            <TD width="120" valign="middle" align="left"><A
href="/WebSphereSamples/YourCo/main.html" target="_top"><FONT
size="-1">Employee Center</FONT></A></TD>
          </TR>
        </TBODY>
      </TABLE>
      </TD>
    </TR>
    </TBODY>
</TABLE>
</CENTER>
<CENTER>
<TABLE width="620" height="310"></TABLE>
</CENTER>
<CENTER>
<TABLE width="620" height="50">
</TABLE>
  <p>&nbsp;</p>
</CENTER>
</BODY>
</HTML>
```

Using the Annotation Editor, we created the annotation instructions to modify the HTML document. Let us look at the annotation instruction created. First, within table 1 we replace table row 1 with a new site welcome message.

Example 5-3 Annotation - replacing table row 1

```
<description target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[1]">
        <replacewithhtml>
<![CDATA[<P><B>Welcome to YourCo<B><P></BR>]]></replacewithhtml>
    </description>
```

Still within table 1, we keep table row two, which contains a table with the application navigation bar, but eliminate the image. Because this is a new table, the annotation engine assumes it will be kept. This is shown in Example 5-4 on page 167.

Example 5-4 Removeing the image

```
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[2]/TD[1]/TABLE[1]">
        <remove tag="IMG"/>
    </description>
```

We also removed the home entry (image and text) in the navigation bar.

Example 5-5 Annotation - removing entries

```
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[2]/TD[1]/TABLE[1]/TBODY
[1]/TR[1]/TD[1]">
        <remove/>
    </description>
<description take-effect="after"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[2]/TD[1]/TABLE[1]/TBODY
[1]/TR[1]/TD[2]">
        <remove/>
    </description>
```

After removing these rows, we keep the rest of the this table.

Example 5-6 Annotation - keep the rest of the entries in the navigation bar

```
<description take-effect="after"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[2]/TD[1]/TABLE[1]/TBODY
[1]/TR[1]/TD[4]">
        <keep/>
    </description>
```

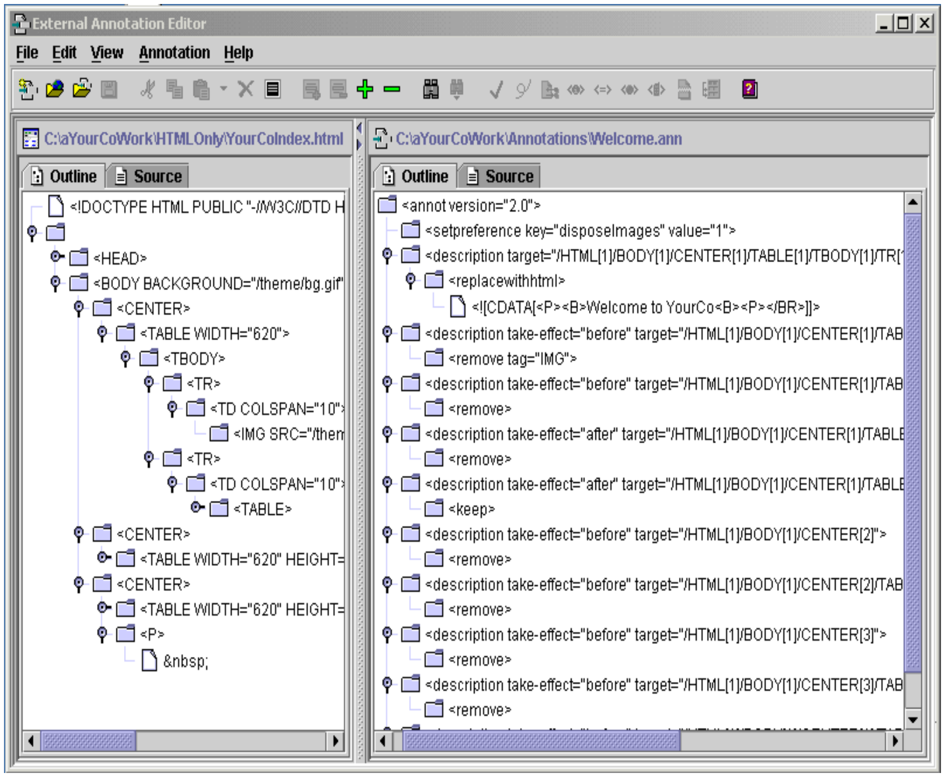
Next, we remove the center tags in table two and table three with these instructions:

Example 5-7 Annotation - removing center tags

```
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[2]">
<remove/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[2]/TABLE[1]"><remove/>
    </description>
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[3]"><remove/>
    </description>
```

```
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[3]/TABLE[1]">
<remove/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[2]/TD[1]/TABLE[1]/TBODY
[1]/TR[1]/TD[4]">
<keep/>
    </description>
```

Before we look at the results in the Nokia Emulator, let us look at the External Annotation Editor to see both the HTML file and the Annotation file. We removed the display of the XPath and Attributes by using the menu bar **View->Show** options. We did not expand the tables or all of the annotation instructions completely, but this gives you an idea of what the screen looks like.

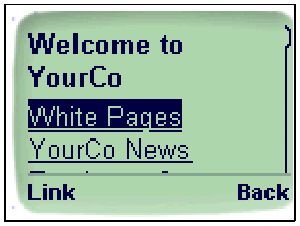

External Annotation Editor
File Edit View Annotation Help
C:\aYourCoWork\HTMLOnly\YourCoIndex.html
Outline Source
C:\aYourCoWork\Annotations\Welcome.ann
Outline Source

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD H
<HEAD>
<BODY BACKGROUND="/theme/bg.gif"
<CENTER>
<TABLE WIDTH="620">
<TBODY>
<TR>
<TD COLSPAN="10"
<IMG SRC="/them
<TR>
<TD COLSPAN="10"
<TABLE>
<CENTER>
<TABLE WIDTH="620" HEIGHT=
<CENTER>
<TABLE WIDTH="620" HEIGHT=
<P>
&nbsp;
```

```
<annot version="2.0">
<setpreference key="disposeImages" value="1">
<description target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[
<replacewithhtml>
<![CDATA[<P><B>Welcome to YourCo<B><P></BR>]]>
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/TAB
<remove tag="IMG">
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/TAB
<remove>
<description take-effect="after" target="/HTML[1]/BODY[1]/CENTER[1]/TABLE
<remove>
<description take-effect="after" target="/HTML[1]/BODY[1]/CENTER[1]/TABLE
<keep>
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[2]">
<remove>
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[2]/TAB
<remove>
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[3]">
<remove>
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[3]/TAB
<remove>
```

Figure 5-15 External Annotation Editor with files

The results of annotating and transcoding the YourCo main page as viewed on a WAP simulator, look like those shown in Figure 5-16 on page 169.


Welcome to YourCo
White Pages
YourCo News
Link
Back

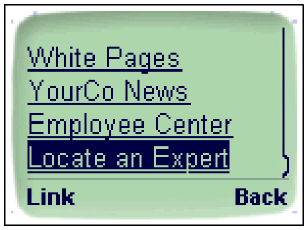

White Pages
YourCo News
Employee Center
Locate an Expert
Link
Back

Figure 5-16 YourCo Welcome, annotated

Between the YourCo main page and the selected application is frameset.html, which contains two links: one to the Locate an Expert application and the other to a blank.html file. We did not annotate this file, so when executing the application you must select **link 1** and click the link to get to the Locate an Expert application.

Creating annotations - Locate Expert select

The HTML document used to display the Locate Expert select contains the following:

– Navigation table - contains the links to home and other applications within YourCo. This will be removed.

– Heading Level 2 - contains the text `Locate an expert:` , which will remain unchanged.

– Paragraph - contains directions to selecting an expert; this is replaced with the text `Select expert`.

– Form- contains a select element with options and an input element. The form is kept but the input element is removed.

The specific HTML we are working on looks like that shown in Example 5-8.

Example 5-8 Locate an Expert select HTML

```
<p>Select the type of expertise you require:<p>
<form name="locate" method="post"
action="http://9.24.105.152:80/httppvc_clnss9.24.104.13/WebSphereSamples/servle
t/itso.wes.lbs.samples.FindExpert" target="results">
        <select name="expertise">
<OPTION>ANALYST
<OPTION>CLERK
<OPTION>DESIGNER
<OPTION>FIELDREP
<OPTION>MANAGER
<OPTION>OPERATOR
<OPTION>PRES
```

```
<OPTION>SALESREP
   </select>
   <INPUT TYPE="submit" NAME="Submit" ID="Submit" VALUE="Submit">
</form>
```

The annotation file we created is shown below.

Example 5-9 Annotation - Locate Expert Select

```
<?xml version='1.0' ?>
<annot version="2.0">
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]">
<remove/>
    </description>
<description target="/HTML[1]/BODY[1]/CENTER[1]/P[1]">
<replacewithhtml><![CDATA[<P>Select expert type:]]></replacewithhtml>
    </description>
<description take-effect="before" target="/HTML[1]/BODY[1]/CENTER[1]/FORM[1]">
        <keep/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/FORM[1]/SELECT[1]"><keep/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/FORM[1]/INPUT[1]"><remove/>
    </description>
</annot>
```

Because the annotation file is an XML file, the first statement is the XML version statement. The next statement is the <annot> element, which is the root of the document. The description instructions shown in the above example perform the following activities:

1. Remove the navigation table and heading level 2 (these are not shown in the HTML but are part of the HTML for each display in the Locate Expert application).
2. Replace the instruction `Select the type of expertise you require:` with the text `Select expert type:`
3. Keep the form, which during the transformation process is turned into a WML <do> with a <go> that includes the URL.
4. Keep the forms select statement.
5. Remove the input statement.

The Locate Expert selection viewed from an Nokia WAP emulator looks like this:

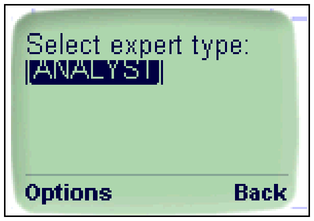


Figure 5-17   Locate Expert select on Nokia

Creating annotations - Locate Expert: No Expert Found

On the occasion when no expert is found, the application returns a message to the requester as an HTML file indicating this condition. The No Expert Found HTML file is shown below:

Example 5-10   Locate Expert - No Expert Found HTML

```
<BODY background="/WebSphereSamples/theme/bg.gif">
    <CENTER>
       <h4>Your Location is:</h4>
       <table width="50%" border="0" cellspacing="2" cellpadding="2">
         <tr align="left" bgcolor="#99CCFF">
           <td><b>Address</b></td><td>11000 Regency Parkway, Cary, NC, 
27513.</td>
         </tr>
         <tr align="left" bgcolor="#99CCFF">
           <td><b>Country</b></td><td>USA</td>
         </tr>
       </table>
       <br><br>
       <h4>There is no employee with the requested expertise in the office 
nearest to you.<br>
           Please contact Headquarters.</h4>
    </CENTER>
  </BODY>
```

The example has the following:

– Heading level 4 - with the text `Your Location is:`, which is kept.

– Table - contains the requester’s location. The first column of the table is removed.

– Heading level 4 - contains the Not Found message. This is replaced by a short message: `No expert is near you. For assistance call 919-333-7878.`

As you can see, the annotation instructions created for this HTML document consist mostly of Keep and Remove instructions, so we will not discuss them. The new type of instruction created is for the last Heading Level 4 (the Not Found message). Here we used a replacewithhtml instruction (the instruction is highlighted).

Example 5-11   Annotation - Locate Expert Not Found

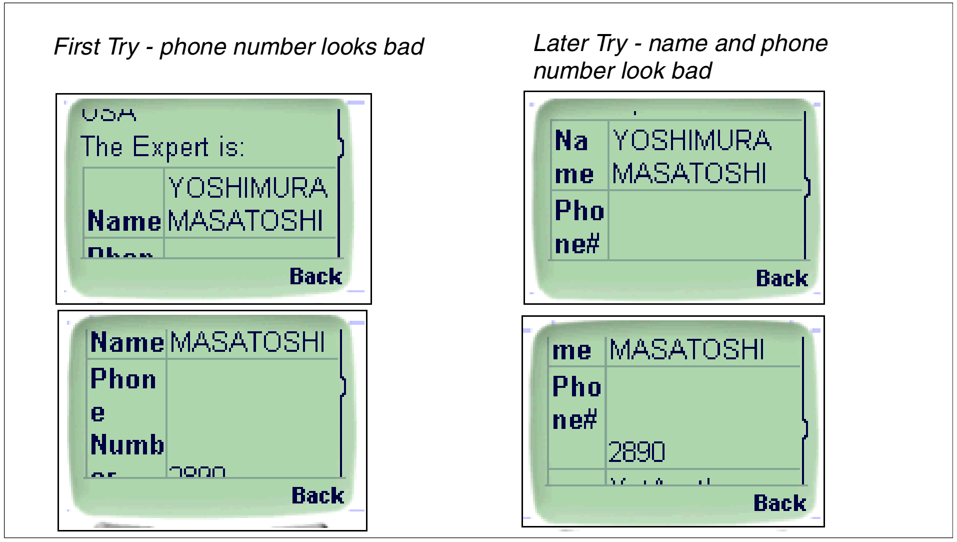
```
<?xml version='1.0' ?>
<annot version="2.0" >
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[1]/TD[1]"><remove/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[1]/TD[2]"><keep/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[2]/TD[1]"><remove/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[2]/TD[2]"><keep/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/BR[1]"><remove/>
    </description>
<description target="/HTML[1]/BODY[1]/CENTER[1]/H4[2]">
<replacewithhtml><![CDATA[<P><B>No expert is near you.</B><BR/>For assistance
call<BR/><B>919-333-7878</B></P>]]></replacewithhtml>
    </description>
</annot>
```

The replacewithhtml wizard allows you to put both HTML tags and text in the text entry window. The description instruction wraps the text and HTML within a CDATA statement.

The *WebSphere Transcoding Publisher Developer's Guide* discusses the <table>, <column> and <row> annotation instructions to clip rows and columns from tables. The annotation editor creates Keep and Remove statements (as shown above) instead. These instructions (table, column and row) were created as shorthand for developers creating annotation files manually. Since the annotation editor generates the XPath as part of the <description> element, these statements (table, column and row) are not part of the annotation editor.

Creating annotations - Locate Expert: Expert Found

The challenge to working with the Expert Found information table in the Locate Expert application was to make the table format nicely on the Nokia emulator. To see the full table, refer to Figure 5-9 on page 161 for the desktop browser view of the HTML document. The various device browsers and emulators decide how they will display tables. So regardless of what we do in the annotation instructions, we cannot make the table look nice, as you can see in Figure 5-18.


First Try - phone number looks bad
Later Try - name and phone number look bad
Back
Back
Back
Back

USA
The Expert is:

Name	YOSHIMURA MASATOSHI

Name	MASATOSHI
Phon e Numb	

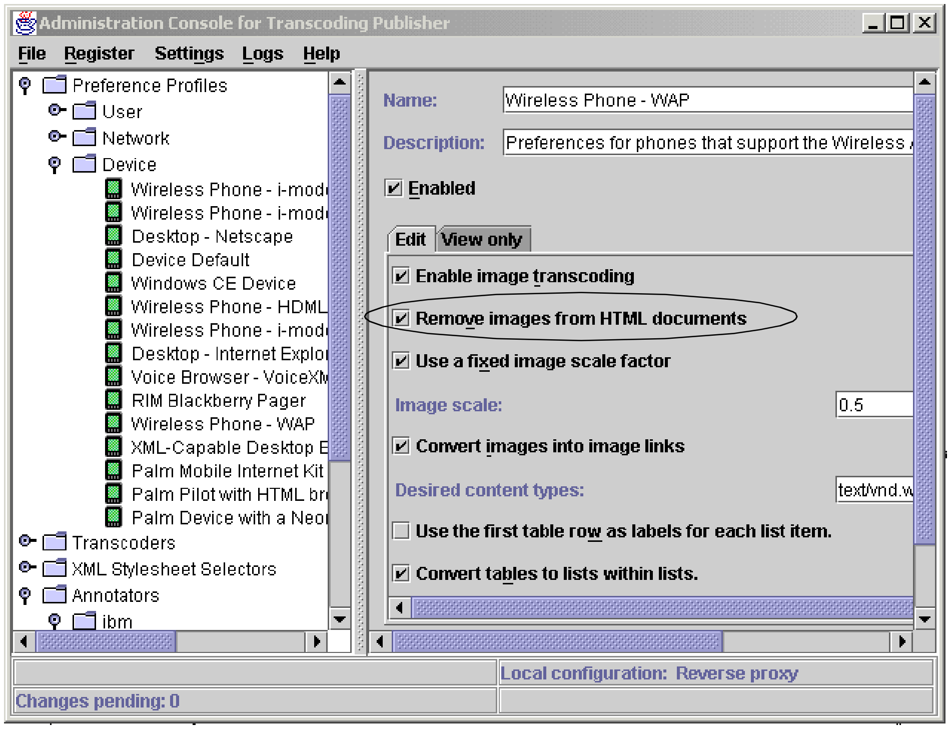
Na me	YOSHIMURA MASATOSHI
Pho ne#	

me	MASATOSHI
Pho ne#	2890

Figure 5-18 Locat Expert - Expert Found using tables

We decided that it was best to replace the table with a list. This is accomplished by changing the device preferences for the Wireless Phone WAP device. The Administration console is used to complete this activity, using the following steps:

1. Open the preference profile folder.
2. Open the device profile folder.
3. Select the Wireless Phone WAP profile. The Wireless Phone WAP profile is displayed (as shown in Figure 5-19).
4. Select **Convert Tables to Lists** within the Lists checkbox.
5. Click the **Save** button.
6. Refresh the server.


Administration Console for Transcoding Publisher

File Register Settings Logs Help

- Preference Profiles
  - User
  - Network
  - Device
    - Wireless Phone - i-mod
    - Wireless Phone - i-mod
    - Desktop - Netscape
    - Device Default
    - Windows CE Device
    - Wireless Phone - HDML
    - Wireless Phone - i-mod
    - Desktop - Internet Explo
    - Voice Browser - VoiceX
    - RIM Blackberry Pager
    - Wireless Phone - WAP
    - XML-Capable Desktop B
    - Palm Mobile Internet Kit
    - Palm Pilot with HTML br
    - Palm Device with a Neo
- Transcoders
- XML Stylesheet Selectors
- Annotators
  - ibm

Name: Wireless Phone - WAP

Description: Preferences for phones that support the Wireless

[✓] Enabled

Edit | View only

[✓] Enable image transcoding

[✓] Remove images from HTML documents

[✓] Use a fixed image scale factor

Image scale: 0.5

[✓] Convert images into image links

Desired content types: text/vnd.w

[ ] Use the first table row as labels for each list item.

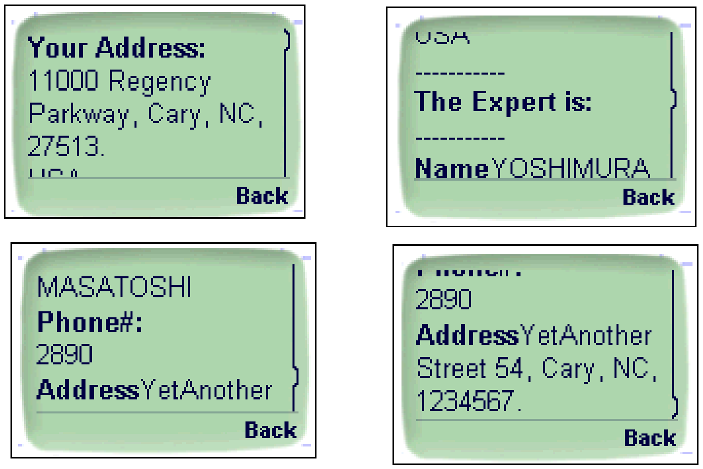
[✓] Convert tables to lists within lists.

Local configuration: Reverse proxy

Changes pending: 0

Figure 5-19 Administration Console - Converting tables for a WAP device

You must be aware that making this change to the device profile is a universal change. This change is now in effect for all HTML files targeted to this device. Figure 5-20 shows the results of making this profile change.


Your Address:
11000 Regency
Parkway, Cary, NC,
27513.
Back

The Expert is:

NameYOSHIMURA
Back

MASATOSHI
Phone#:
2890
AddressYetAnother
Back

2890
AddressYetAnother
Street 54, Cary, NC,
1234567.
Back

Figure 5-20 Locate Expert -Expert Found: Nokia with Device preference change

The annotation file for the Expert Found is in the following example

Example 5-12 Locate Expert Found annotation file

```
<?xml version='1.0' ?>
<annot version="2.0">
<description target="/HTML[1]/BODY[1]/CENTER[1]/H4[1]">
<replacewithhtml><![CDATA[<p><b>Your Address:</b></p>]]></replacewithhtml>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[1]/TD[1]"><remove/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[1]/TD[2]"><keep/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[2]/TD[1]"><remove/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[1]/TBODY[1]/TR[2]/TD[2]"><keep/>
    </description>
<description target="/HTML[1]/BODY[1]/CENTER[1]/H4[2]"><replacewithhtml>
<![CDATA[<b>-----------<br/>The Expert
is:<br/>-----------<br/></b>]]></replacewithhtml>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[1]"><keep/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[2]"><keep/>
```

```
    </description>
<description target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[2]/TD[1]">
        <replacewithhtml><![CDATA[<b>Phone#:</br></b>]]></replacewithhtml>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[3]"><remove/>
    </description><description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[4]"><remove/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[5]"><keep/>
    </description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/CENTER[1]/TABLE[2]/TBODY[1]/TR[6]"><remove/>
    </description>
</annot>
```

Putting the application together

When we tested the individual annotation files, we enabled only the annotation files needed for that particular path. To test the complete dialog, we enabled both the Expert Found and Expert Not Found annotation files. Both of these annotation files use the same URL which is `*/itso.wes.lbs.samples.FindExpert`. So, regardless of the actual content (Found or Not Found), the same annotation file was being used by the annotation engine, causing erroneous displays for one of the paths.

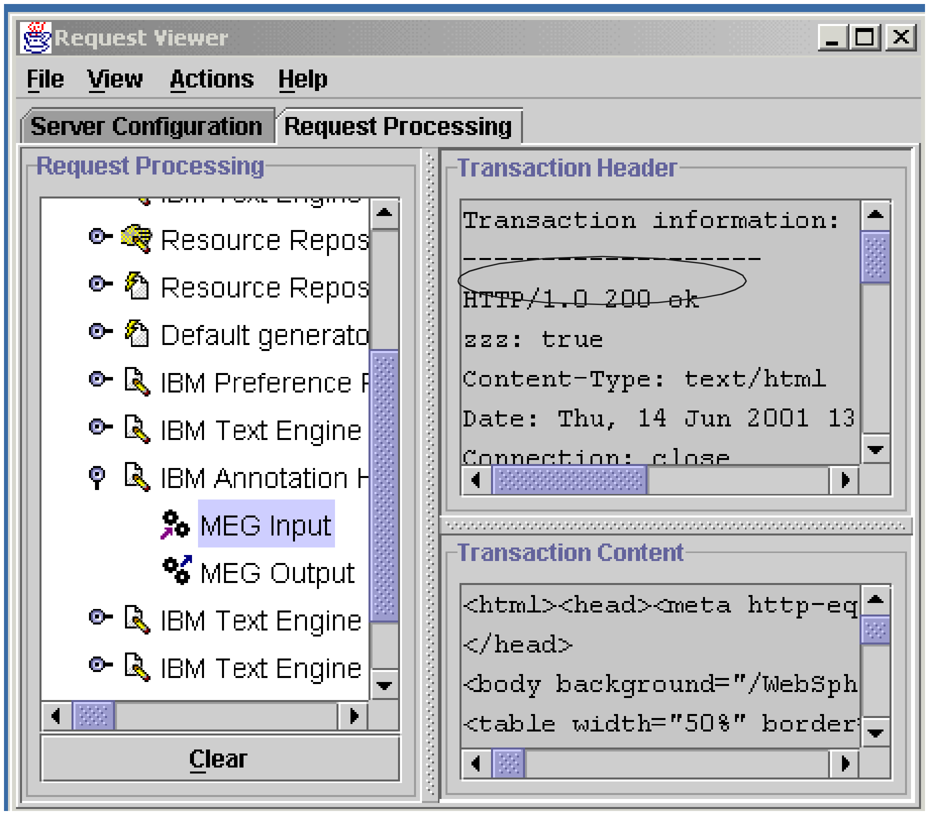
The annotation engine needed further direction on which annotation file to use for either the Found or Not Found HTML file. Annotation file selection can depend on fields in the HTTP header or on preferences that are in effect for the HTML document being processed by the annotation engine. In both instances, we were using the same device, and our solution was to modify the HTTP header. Our JSP developer added a unique field to the HTTP header. The field name was *zzz* and the field value settings were:

– true - an expert was found

– false - an expert was *not* found

– error - there was an error

To verify the HTTP header change, we executed the Locate Expert on the emulator and viewed the HTTP header in the Request Viewer’s Transaction Header panel, shown in Figure 5-21.

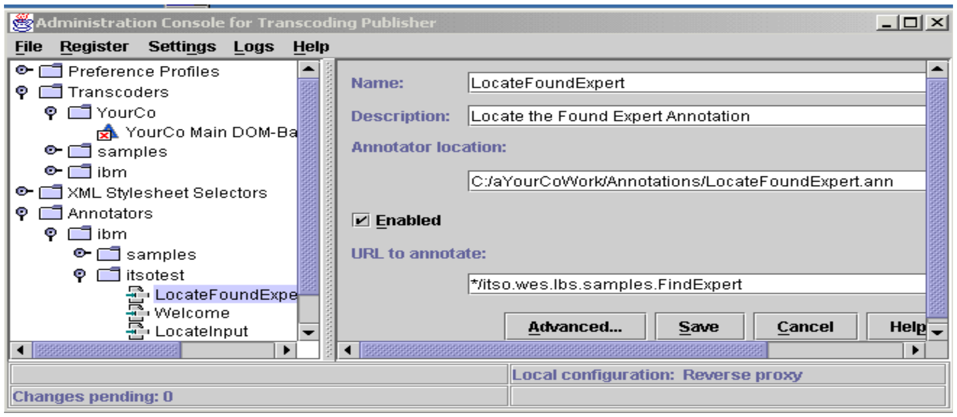


```
Transaction information:
--------------------
HTTP/1.0 200 ok
zzz: true
Content-Type: text/html
Date: Thu, 14 Jun 2001 13
Connection: close
```

```
<html><head><meta http-eq
</head>
<body background="/WebSph
<table width="50%" border
```

Figure 5-21 Request Viewer - Transaction Header panel

Next, we registered this change for both annotation files using the Administration Console. The annotation file registration panel (within the Administration Console) has an Advanced Annotation Selection Properties window which allows you to add special constraints for the annotation file. The Administration Console with the Annotation Registration panel is shown in Figure 5-22.


Administration Console for Transcoding Publisher

File Register Settings Logs Help

- Preference Profiles
- Transcoders
  - YourCo
    - YourCo Main DOM-Ba
  - samples
  - ibm
- XML Stylesheet Selectors
- Annotators
  - ibm
    - samples
    - itsotest
      - LocateFoundExpe
      - Welcome
      - LocateInput

Name: LocateFoundExpert

Description: Locate the Found Expert Annotation

Annotator location: C:/aYourCoWork/Annotations/LocateFoundExpert.ann

☑ Enabled

URL to annotate: */itso.wes.lbs.samples.FindExpert

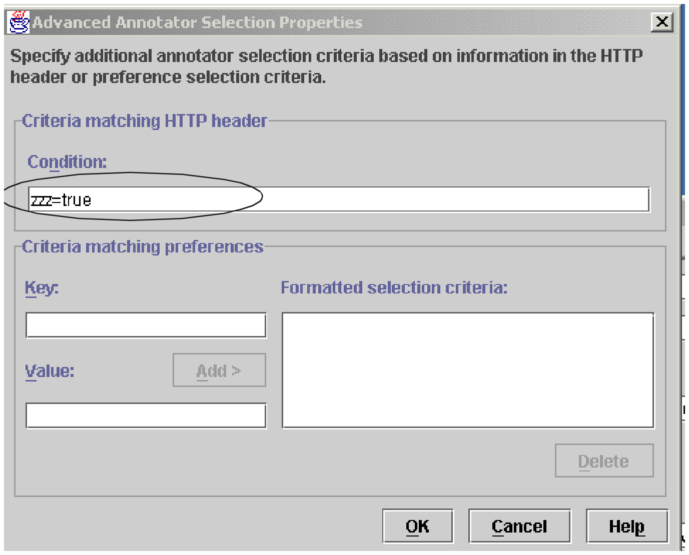
Advanced... Save Cancel Help

Local configuration: Reverse proxy

Changes pending: 0

Figure 5-22 Administration Console Register LocateFoundExpert.ann

By clicking the **Advanced** button on the registration screen, the Advanced Annotation Selection Properties screen is displayed. The `Compare` condition is entered in the condition entry field of the Criteria matching HTTP header box, as shown below.


Advanced Annotator Selection Properties

Specify additional annotator selection criteria based on information in the HTTP header or preference selection criteria.

Criteria matching HTTP header

Condition: zzz=true

Criteria matching preferences

Key:

Value:

Formatted selection criteria:

Add > Delete

OK Cancel Help

Figure 5-23 Administration Console Advanced Selection Properties Panel

The condition was set for the following annotation files:

- LocateFoundExpert.ann - zzz= true
- LocateNoOneCondition.ann - zzz=false

With this change implemented, the annotation engine is able to determine the correct annotation file to use with the HTML document. The emulator is receiving the correctly annotated content.

One final challenge

As a final test, we selected the Locate Expert application in Internet Explorer and, much to our surprise, the annotations were still in effect there also. We needed to constrain the annotations further. We performed the following steps to determine the User_Agent field set by the emulator;

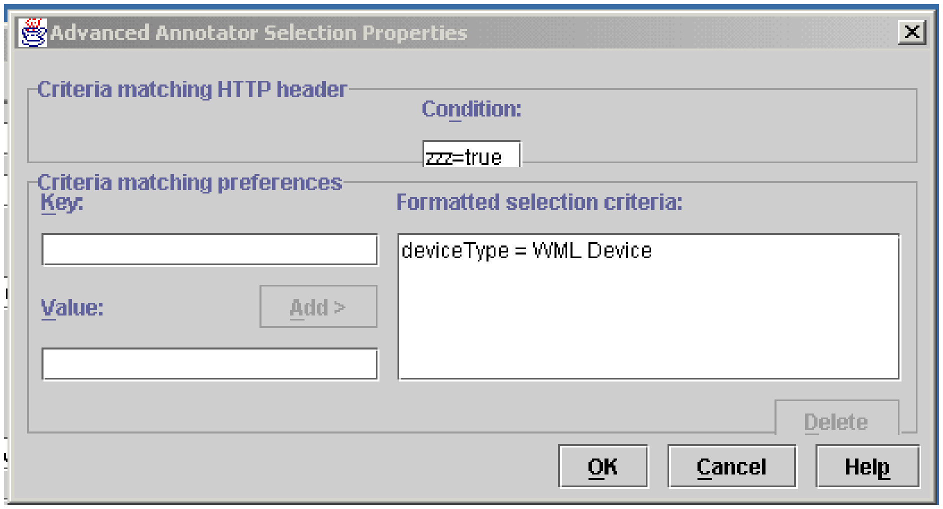
1. Set the Trace level to **High** in the Administration Console; select **Logs -> Trace -> Properties** to get the Transcoding Publisher Trace Properties window and select the **Trace Categories Level High** radio button. Click the **OK** button.
2. Run the Locate Expert application on the emulator; this will update the log file.
3. Make a copy of the trace file, which is found at */IBMTrans/Log.
4. Look in the trace file (copy) for the HTTP with the User_Agent value; the HTTP header found in the log is shown in Example 5-13.

Example 5-13 HTTP header information from log file

```
2001.09.13 11:26:49.523 com.ibm.wbi.protocol.http.sublayer.HttpBackend
sendRequest Megs worker #5
  Final request header:
GET /WebSphereSamples/YourCo/Locate/frameset.html HTTP/1.0
Accept-Language: en
Cookie: sesessionid=SP5BTC4GOHZV2MGQYHV535Y
Accept-Charset: UTF-8, ISO-8859-1, ISO-10646-UCS-2
Connection: Keep-Alive
User-Agent: Nokia-WAP-Toolkit/2.1null
Date: Thu, 13 Sep 2001 15:26:49 GMT
Accept: */*
Host: 9.24.104.13
```

5. We found the User_Agent =Nokia-WAP-Toolkit/2.1 in the log file.

With the User_Agent information, we could determine the device profile that applied, which was the WML Device, and selected the deviceType=WML Device as our condition. We changed the Annotation file registration information to reflect the deviceType as the constraint. This information was entered for each annotation file in the Advanced Annotation Selection Properties window(s), as shown next.


Advanced Annotator Selection Properties
Criteria matching HTTP header
Condition:
Criteria matching preferences
Key:
Formatted selection criteria:
Value:
Add >
Delete
OK
Cancel
Help

zzz=true

deviceType = WML Device

Figure 5-24 Advanced Annotator Selection Properties - deviceType

With the server refreshed, we tested our change, which worked effectively to allow:

– The Internet Explorer browser to display the complete HTML content for the application.

– The Emulator to display the annotated content for the application.

5.3.5 Testing an annotation file

Once the annotation file has been created, it must be tested to ensure that you get the expected results. Various WebSphere Transcoding Publisher tools were used in the test process:

– Request Viewer - used to watch the flow of the requests and responses through the transcoding server. The Request Viewer is a helpful debugging tool. When we encountered problems with edited (using Notepad) annotation files, the DOS prompt helped us identify the error.

> **Note:** To start Request Viewer, we used a DOS prompt, set the class path to the request view location */IBMTrans and entered the runtranscoding `-g` command. This allowed us to use the prior version of the Request Viewer, which does not expect RMI. This matched our local development environment.

– Administration Console - used to register the annotation files, modify preference profiles, and refresh the server, so that the server recognized any changes we made.

> **Note:** Each time an annotation file is registered, the annotation registration is altered, any other modifications occur via the Administration Console. The server must be refreshed to have these changes take effect.

– Internet Explorer - used to view the original Web application.

– Nokia Toolkit 2.1 - used to view our annotation results.

– External Annotation Editor - used to create and adjust the annotation files.

5.3.6 Hints and tips

Along the way, we learned various things that may be of use to you in your development:

– When determining the URL to use when registering (with the Administration Console), an annotation file may not be obvious because the HTML is generated by JSPs or servlets. We found that the Nokia Toolkit workbench displays the HTML URL for the content being displayed on the Blueprint emulator.

– When using JSPs, capturing the HTML to be used by the Annotation Editor is not always an obvious process. We found that you can get the HTML for the IE browser by clicking the content with the right mouse button and selecting **Display Source** from the pop-up menu. This source can then be saved to a file. Another approach is to change the trace level to **error** (in the Administration Console); the response stream (going back to the browser) is included in the log file. We copied the HTML from the log file to an HTML file.

- The device preference settings are stored (in our case) in Program Files/IBMTrans/etc/preferences/device/*.prop. The name-value pairs contained in the properties can be used to set conditions.
- If you save changes to an annotation file using an editor such as Notepad and that file is registered, you do not need to refresh the server. The next time the annotation engine runs against that file, the changes are processed.
- When trying to refine the output to the device, use the Nokia Toolkit workbench to edit, save and run the changes. This way, you have a better idea of the type of annotations that must be created to produce this output.

5.4 Internal annotation

Internal annotation allows you to include the annotation instructions as comments within the HTML document to which they apply. Creating internal annotations is useful when the annotation author has access to or owns the HTML document. The HTML document can be changed either by editing the HTML document directly or by using the WebSphere Studio Version 3.5.3 or higher. The Page Designer within Studio is used to create the annotations.

5.4.1 WebSphere Studio Page Designer

Page Designer offers annotation support to give you greater control over how pervasive computing devices will display your HTML. You can use Page Designer to insert annotation tags directly into the HTML file. These annotation instructions indicate what and how the HTML elements should be displayed when your pages are served to the PvC devices. When the annotated HTML pages are requested from the client device, WebSphere Transcoding Publisher tailors the content based upon your annotation instructions and the characteristics of that particular client device.

Page Designer has many of the capabilities found in the External Annotation Editor and includes the basic annotation instruction currently defined. Page Designer supports the following annotation options:

- Remove or keep elements - for either individual elements or regions of the HTML document
- Replace elements with text - to replace elements with straight text (without any HTML included). If you include HTML tags in the text, they will be part of the text displayed.
- Remove table columns or rows - to select which table rows and/or columns to remove. You can remove multiple rows and columns in any combination.
- Propagate table labels - to allow column headings to be propagated as labels with the row content when rendering tables as lists.

WebSphere Studio allows adding annotation instructions to the HTML or JSP files, but it does not have a run-time environment for executing the annotation instructions. WebSphere Transcoding Publisher is required to execute the annotation instruction embedded in your HTML or JSP files.

Loading HTML and JSP

The YourCo Locate Expert was not created in WebSphere Studio. In order to get the Locate Expert HTML and JSPs into Studio, we first created a project called WTPforITSO within Studio. Within the WTPforITSO project is a Resource file containing a WTPforITSO folder. We created blank .jsp and .html files with the names of the HTML and JSPs in Locate Expert and copied the code into these files. Here are the steps we followed:

1. Right-click the folder.
2. Select **Insert -> File**.
3. In the Insert File window, select either **Blank.htm** or **Blank.jsp** (whichever is appropriate for the file type you want to copy).
4. In the File Name entry field, change the File name from Blank to the appropriate name.
5. Click the **OK** button.
6. From NotePad, copy the original source and paste it into the file just created.
7. Save the new file.

We repeated these steps for all the HTML and JSP files.

Using Page Designer to annotate

Page Designer allows you to set the default clipping state for the pages within the project. The default clipping state determines whether you plan to keep or remove most of the elements with the document(s). So if the majority of your activity on the files will be to keep content, set the default clipping state to Keep, otherwise set it to Remove. To set the default clipping state, follow these steps:

1. Double-click an HTML or JSP file; this causes Page Designer to be invoked.
2. Click the **Normal** tab in the Page Designer window.
3. From the Menu bar select **Page -> Page Properties**.
4. In the Page Properties window, select the **Annotation** tab.
5. Select either the **Keep** or **Remove** radio button and click **OK**.

Remove or keep elements

To remove or keep elements in the HTML file or JSP:

1. Open that file within Page Designer and select the **Normal** tab, which allows you to see the page representation.
2. Select a particular element or portion of the document that you want to annotate.
3. From the main menu select **Edit -> Annotate** or right-click and from the pop-up menu select **Annotate**.
4. Select the action you want to apply (Remove or Keep).

If the action you select is Remove, that element(s) appears with diagonal hash marks to visually indicate that the element(s) will be clipped during transcoding. This visual clue is optional and can be disabled from the main menu by selecting **Tools -> Page Designer Options** and clicking the **Annotation** tab. Within the Annotation tab, deselect the **Show removed regions with shaded overlay** box and click the **OK** button.

Replace elements with text

You can replace any HTML element in your page with a text string.To replace an element with text:

1. Select the text element.
2. From the menu select **Edit->Annotate->Set Text Replacement**.
3. A Text Replace Window appears; type your replacement text.
4. Click **OK**.

Remove table columns or rows

You can use Page Designer to remove any table column or table row that is not essential to the page content. Rows and columns may be clipped in any combination. To remove a row or column from a table:

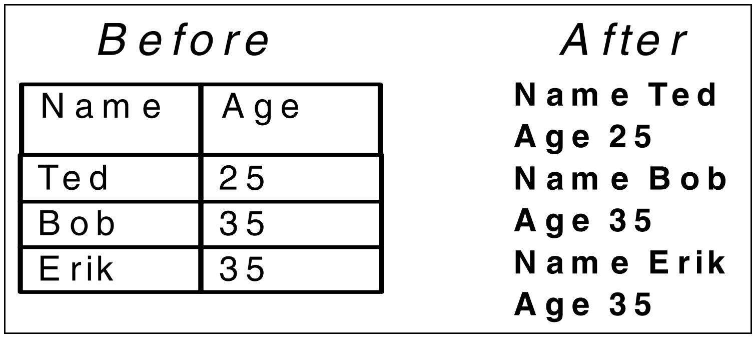
1. Select a cell in the row or column you want to remove.
2. From the main menu, select **Edit->Attributes**.
3. Select the table from the tab pull-down (if not already selected) that adds various tabs to the window.
4. From the Attributes window, select the **Annotation** tab.
5. Within the Annotation tab, select the **Remove this row** and/or the **Remove this column** checkbox.
6. Click **OK**.

Propagate table labels

WebSphere Transcoding Publisher can convert tables to lists for better viewing on a PvC device. When tables are converted to lists, each row is converted to a list and then appended to the previous row. In order to keep it clear as to which list entry goes with which heading, you can specify that the headings be propagated with the cells when they are displayed. To make this happen:

1. Select a cell within the table.
2. From the main menu, select **Edit-> Attribute**.
3. Select the table from the tab pull-down (if not already selected) that adds various tabs to the window.
4. In the Attributes window, select the **Annotation** tab.
5. In the Propagate labels pull-down menu, select **Row**.
6. Click **OK**.

To understand visually what this means, Figure 5-25 on page 186 shows the “before” table and the resulting list with the heading propagated to the rows.


Before

Name	Age
Ted	25
Bob	35
Erik	35

After

Name Ted
Age 25
Name Bob
Age 35
Name Erik
Age 35

Figure 5-25 Table propagation example

5.4.2 Sample scenario: Locate Expert

In our environment, we used WebSphere Studio Version 4.0. We are using the YourCo Locate Expert application, which was also used as the External Annotation sample scenario. In this example, we show the different types of annotations that can be performed with Studio and the resulting annotation instructions. Propagating table labels will not be shown in our sample scenario because the tables in this application do not have headings.

Remove or keep elements

In Page Designer, the Remove Elements annotation is visually interesting because the element(s) designated with Remove appear with diagonal slashes through them. This indicates that WebSphere Transcoding Publisher will delete this content. The steps to remove or keep elements are explained in “Remove or keep elements” on page 184. In Page Designer, we have the LocateInput.html file, shown in Figure 5-26 on page 187.

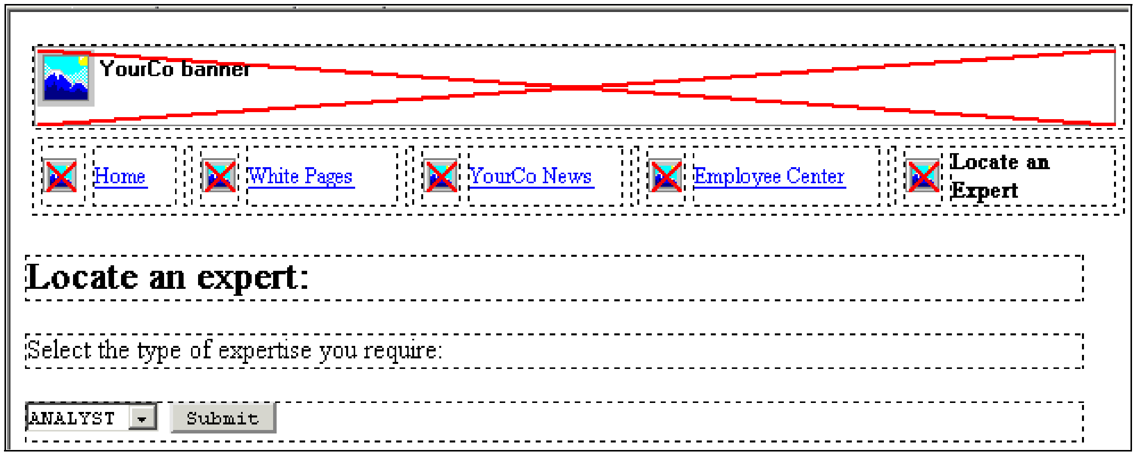


Figure 5-26 LocateInput.HTML in Page Designer

We removed the YourCo banner by selecting it (the frame around it becomes pink) and right-clicking **Select Annotation->Set Remove Region**. The HTML for the YourCo banner now has diagonal lines through it.

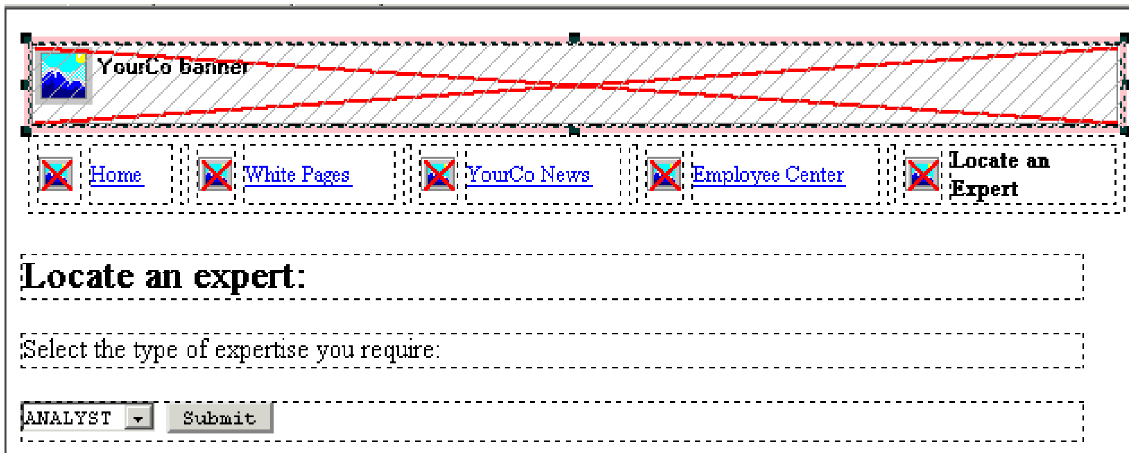


Figure 5-27 LocateInput.HTML in page designer with YourCo banner removed

The annotation instruction is placed in the HTML as a comment prior to the HTML tag; the instruction is shown below. The original statement starts with **`<IMG`**.

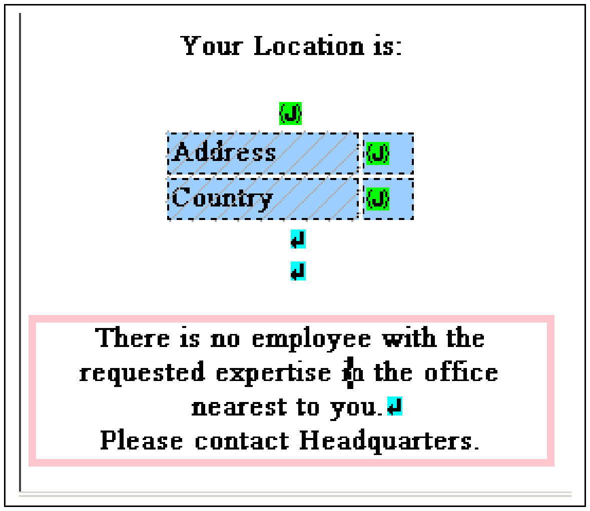
Example 5-14 Remove Annotation Instruction

```
<!--METADATA type="Annotation" startspan
<?xml version="1.0"?><annot version="1.0"><remove /></annot>--> <IMG
src="/WebSphereSamples/theme/topBanner.gif" border="0" alt="YourCo banner"
align="bottom" width="607" height="46">
<!--METADATA type="Annotation" endspan
<?xml version="1.0"?><annot version="1.0"><keep /></annot>
-->
```

The second annotation (keep) was placed in the code because the default annotation state is Keep.

Replace elements with text

We used the ResultsNotFound.jsp to show text replacement. This replaces the specified text within the JSP, along with any associated HTML tags. This instruction should be used carefully because the new content may not appear on the device as you want it to. Let us start with the Page Designer display of ResultsNotFound.jsp, where the text area has been selected.


Your Location is:

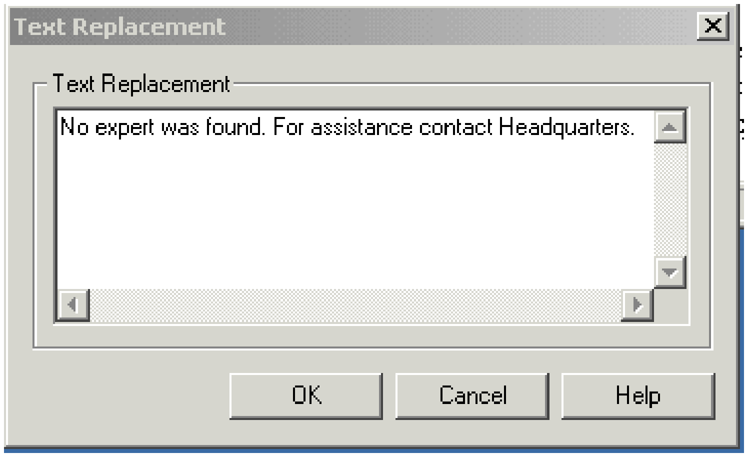
Address

Country

There is no employee with the requested expertise in the office nearest to you.
Please contact Headquarters.

Figure 5-28 ResultsNotFound.jsp in Page Designer

Next, right-click and select **Annotate->Select text replacement**. The Text entry window appears; we typed in the replacement text.



No expert was found. For assistance contact Headquarters.

Figure 5-29 Replacement text

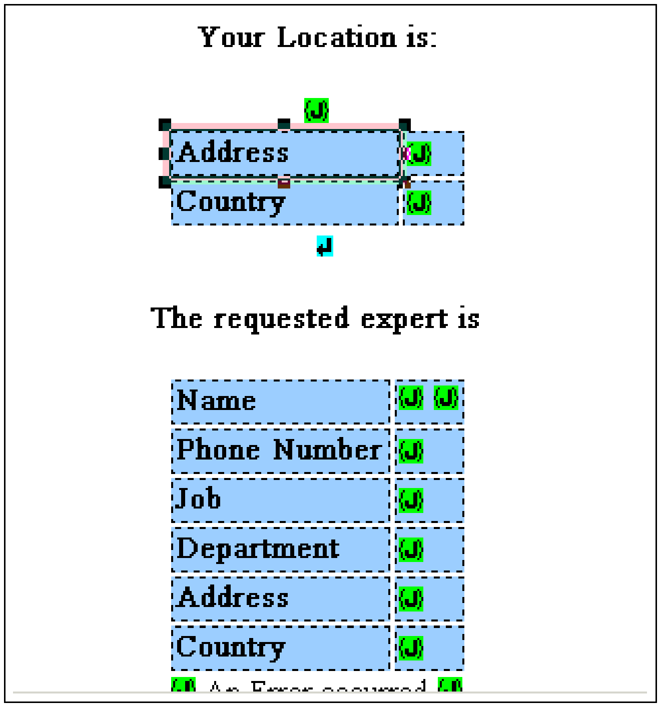
The annotation that is created is as follows; the original text is in bold and starts with **`<h4>`**.

Example 5-15 Replacement text

```
<!--METADATA type="Annotation" startspan
<?xml version="1.0"?><annot version="1.0">
<replace><text>No expert was found. For assistance contact
Headquarters.</text></replace></annot>
-->
<h4>There is no employee with the requested expertise in the office nearestto
you.<br>
Please contact Headquarters.</h4>
<!--METADATA type="Annotation" endspan-->
```

Remove table columns and rows

We used the ResultsFound.jsp to show the process of removing rows and columns from tables. Here we start with the Page Designer display of ResultsFound.jsp's two tables, where Table 1 first column and first row are selected (containing the word `Address`).


Your Location is:

Address	
Country	

The requested expert is

Name	
Phone Number	
Job	
Department	
Address	
Country	

Figure 5-30 ResultsFound.jsp before Annotation

With the table entry selected:

1. Right-click, select **Attributes** from the Attributes window .
2. Select **Annotation**.
3. Select the **Remove this column** checkbox.

This will remove the first column of this table. In the second table, we removed specific rows including Job, Department, and Country. Once a specified row was selected, the same steps as stated above were executed, except that the final step was to select the **Remove this row** checkbox. After these action were performed, the Page Designer view showed the two tables with the appropriate rows and columns marked.

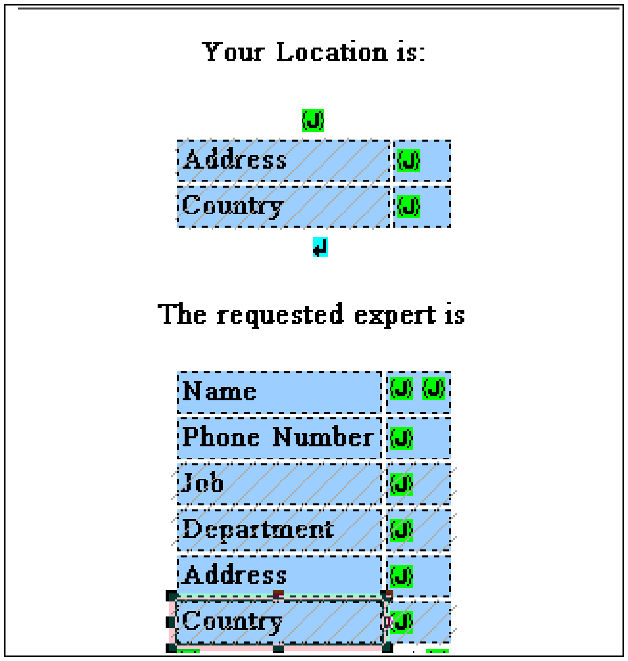


Figure 5-31   ResultsFound.jsp After Annotating Tables

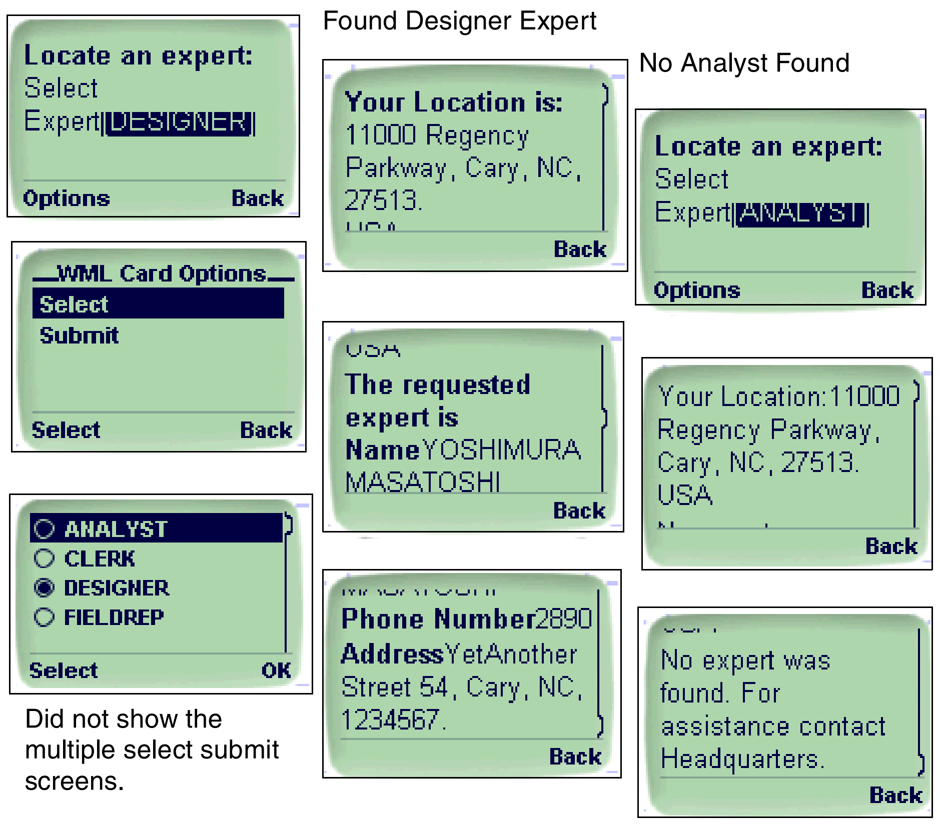
The annotation instructions created for the first table are shown in the following example.

Example 5-16   ResultsFound.jsp Table Annotations

```
<TABLE width="50%" border="0" cellspacing="2" cellpadding="2">
<tr align="left" bgcolor="#99CCFF">
<!--METADATA type="Annotation" startspan
<?xml version="1.0"?><annot version="1.0"><remove /></annot>-->
      <td><b>Address</b></td>
<!--METADATA type="Annotation" endspan
<?xml version="1.0"?><annot version="1.0"><keep /></annot>-->
<td><%= _u0_6 + ", " + _u0_7 + ", " + _u0_8 + ", " + _u0_9 + "."  %></td> </tr>
<tr align="left" bgcolor="#99CCFF">
<!--METADATA type="Annotation" startspan
<?xml version="1.0"?><annot version="1.0"><remove /></annot>-->
<td><b>Country</b></td>
<!--METADATA type="Annotation" endspan
<?xml version="1.0"?><annot version="1.0"><keep /></annot>-->
<td><%= _u0_10 %></td> </tr>
</TABLE>
```

Final results

We performed many of the same annotations with internal annotations as we performed with the external annotations, with the exception of some of the text content changes. The Locate Expert application with all the annotations applied looks as shown in Figure 5-32 on page 192.



Locate an expert:
Select
Expert[DESIGNER]

WML Card Options
Select
Submit

ANALYST
CLERK
● DESIGNER
FIELDREP

Your Location is:
11000 Regency Parkway, Cary, NC, 27513.
USA

The requested expert is
Name YOSHIMURA MASATOSHI

Phone Number 2890
Address YetAnother Street 54, Cary, NC, 1234567.

Locate an expert:
Select
Expert[ANALYST]

Your Location:11000 Regency Parkway, Cary, NC, 27513.
USA

No expert was found. For assistance contact Headquarters.

Figure 5-32 WAP device - Results of internal annotation

Notice how the text wraps on the very first screen (Locate an Expert). This is because we used the Replace text annotation to shorten the instruction to the user and lost the associated HTML tags.

5.5 Text Clipping with Java

Another way to perform document clipping is to create your own custom transcoders, known as *text clippers*. Text clipper are written in Java, allowing you to manipulate the target document (the target document can be HTML or any other presentation markup language). You can write a text clipper to work with the target document in either of these forms:

– A Document Object Model (DOM) representation of the target document

– The target document in text form

Operating on the text document is challenging because you must write code to handle the content and to handle and manage the element nodes as well. Working with the text document increases the complexity of your code.

To simplify development, WebSphere Transcoding Publisher has added capabilities enabling you to work with the target document. These WebSphere Transcoding Publisher capabilities are found in the following classes:

– TextClipper - a (Response) editor provides functions for clipping text from either the DOM or the text document.

– DomUtilities - this provides various utilities for working with the DOM, for example, finding specific nodes within the DOM and manipulating them.

Developing text clippers using the DOM allows you to take advantage of these classes, their capabilities and other XML Parser functions. However, you must evaluate the approach that best meets your needs and develop the appropriate text clipper.

You can implement the text clipper to either of these APIs:

– Web Intermediaries (WBI) APIs - allowing you to create a WebSphere Transcoding Publisher plug-in (known as a MEG).

– Java Servlet Version 2.1 API - allowing you to create an independent transcoder (known as a MEGlet).

Creating a MEGlet provides you with portability across platforms and environments and takes advantage of the Web server. However, creating a MEG allows you to take advantage of other transcoders, which aids in manipulating the document and helps provide greater control over the processing. You must determine how and where you want your text clipper to run and how it should function in your environment. For more details on these approaches and other considerations, look at the *WebSphere Transcoding Publisher Developers Guide*, which can be found at `www.ibm.com/software/webservers/transcoding/library.html`.

5.5.1 Sample scenario - YourCo main page text clipper

We created a text clipper to clip the YourCo main page. Our text clipper works on the HTML page once it has been converted into WML. Our clipper uses the WML DOM as input; we also know that all the images have been removed. Our text clipper is a WBI plug-in and performs the following processes against the WML DOM:

– Find the menu bar Home entry and change the text `Home` into the text `YourCo! Welcome!`

– Keep the anchors in the menu bar (White Page, YourCo News, Employee Center, and Locate an Expert)

– Remove almost everything else in the document (everything between the text `Welcome!` and the <do> element at the end)

To make it easier, we copied an existing WebSphere Transcoding Publisher text clipping example (IBMStockClipperDom.java), named it MainClipDom.java and made the necessary modifications.

Creating MainClipDom

Our directory structure for the clipper consists of:

– The root - aYourCoTextClip which contains the .java file, the .make file, and our .bat file.

– The subdirectory YourCo, which contains the property file.

Each of these components will be discussed, but first let us look at the source code. Our imports areas shown in Example 5-17:

Example 5-17 Import statements

```
import java.util.Enumeration;
import java.io.IOException;
import java.lang.String;

import org.w3c.dom.Node;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import com.ibm.wbi.Plugin;
import com.ibm.wbi.PluginError;
import com.ibm.wbi.RequestEvent;
import com.ibm.wbi.SystemContext;
import com.ibm.wbi.MegContext;
import com.ibm.wbi.MegInputStream;
import com.ibm.wbi.MegWriter;
import com.ibm.wbi.RequestRejectedException;

import com.ibm.transform.TranscoderConstants;
import com.ibm.transform.textengine.DomMegObject;
import com.ibm.transform.textengine.mutator.DOMUtilities;
import com.ibm.transform.textengine.mutator.DOMMutator;
import com.ibm.transform.textengine.mutator.MutatorContext;
import com.ibm.transform.textengine.mutator.TextClipper;
import com.ibm.transform.textengine.mutator.wml.WMLPrinter;
```

The MainClipDom.java file contains two classes:

– MainClipDom - which extends Plug-in and creates an instance of the second class (MainClipDomEdit).

– MainClipDomEdit - which extends TextClipper and accesses and manipulates the DOM.

The code for MainClipDom.java is as follows:

Example 5-18   MainClipDom.java

```
public class MainClipDom extends Plug-in
{
    public void enable()
    {
       MainClipDomEditor meg = new MainClipDomEditor();
       try {
           addMeg(meg);
       }
       catch (PluginError pe) {
           pe.printStackTrace();
       }
    }
}
```

Let us take a look at the MainClipDomEdit.java class construct and its instance variables. As you can see, there are various variables which are used for matching against the DOM content, and a method getPropertiesName() which returns the associated property file.

Example 5-19   MainClipDomEdit construct

```
class MainClipDomEditor extends TextClipper
{
  /** * The content type used for our input is WML*/
  private static final String WML_CONTENT_TYPE = "text/vnd.wap.wml";
private static final String SETUP_PROPERTIES = "plugins/YourCo/MainClipDom";
  /***  Key WML tags used for matching within the DOM */
```

```
private static final String PARAGRAPH_ELEMENT_TAG_NAME = "P"; //* PARAGRAPH
private static final String DO_ELEMENT_TAG_NAME = "DO"; //*DO element
private static final String ANCHOR_ELEMENT_TAG_NAME = "A"; //* anchor
private static final String BOLD_ELEMENT_TAG_NAME = "B"; //* bold
private static final String BREAK_ELEMENT_TAG_NAME = "BR"; //* break
private static final String IMAGE_ELEMENT_TAG_NAME = "IMG"; //*image
/*****
* getPropertiesName()
******/
public String getPropertiesName()
  {
     return SETUP_PROPERTIES;
  }
```

The DOM processing occurs in the handleRequest(), which is called by WBI when the request being processed matches the conditions associated with this clipper. The conditions must be defined in the properties file. This first part of the method performs setup and verification activities.

Example 5-20 handleRequest() part 1

```
public void handleRequest (RequestEvent reRequest)
    throws RequestRejectedException, IOException
  {
System.out.println("MainClipDomEditor handling request");
//---------------------------------------------------------------
     // Get the MegContext
     //---------------------------------------------------------------
     MegContext megContext = reRequest.getMegContext();
//----------------------------------------------------------------
     // The MIME-type that we should generate.
     // This is required in order to do any processing.
     //----------------------------------------------------------------
     Enumeration contentTypes = getPreferredContentTypes(reRequest);
     if (!contentTypes.hasMoreElements())
     {
       //-------------------------------------------------------------
       // Must have an output type so we know what to generate
       //-------------------------------------------------------------
       System.err.println("No output content-type specified.  " +
                          "Check the preferences files.");
forwardErrorPage(reRequest, 417, "TC_MISSING_TARGET_CONTENT_TYPE",
                        WML_CONTENT_TYPE);
```

```
          return;
        }

        DomMegObject domMegObject = null;

        String sOutputPage = ""; //default to empty page

        String contentType="";
```

The next portion of handleRequest() method:

- Determines the contentType,
- Obtains the paragraph within the card element of the DOM, and
- Finds the text `Home` within the DOM.

Example 5-21   handleRequest() part 2

```
if(contentTypes.hasMoreElements())
     {
contentType = contentTypes.nextElement().toString();
        if(contentType.equalsIgnoreCase("text/vnd.wap.wml"))
        {
           Document doc = (Document) getTranscodedDOM(reRequest);
           if(doc == null)
           {
              throw new RequestRejectedException();
           }
           Node pNode = DOMUtilities.findNodeOfType(doc,
PARAGRAPH_ELEMENT_TAG_NAME);
           if(pNode != null)
           {
              Node startNode =
DOMUtilities.findChildWithNodeMatchingPattern(pNode,

"*Home*", true);
              if(startNode == null)
              {
                System.out.println("** could not find Home **");
              }
```

Next, the code looks for the text `Welcome!` because the text and everything associated with it up to the <do> tag is removed, as shown in Example 5-22.

Example 5-22   handleRequest() part 3

```
if(startNode != null) {
                 System.out.println("*** startNode *** " +
startNode.getNodeName());
```

```
                 //----------------------------------------------
                 // Find parent node
                 //----------------------------------------------
                  Node parent = startNode.getParentNode();
                  System.out.println("*** parent of startNode *** " +
parent.getNodeName());
                 //------------------------------------------
                 // We need to skip over everything until the Welcome!
                 //------------------------------------------
                 boolean found = false;
                 Node ignoreNode =
DOMUtilities.findChildWithNodeMatchingPattern(pNode,

"*Welcome!*", true);
                  //----------------------------------------
                 // Now just remove everything up to the
                 // DO element node at the end.
                 //----------------------------------------
                 while(
                        (ignoreNode != null) &&

!ignoreNode.getNodeName().equalsIgnoreCase(DO_ELEMENT_TAG_NAME)
                      )
                 {
                    Node nextSibling = ignoreNode.getNextSibling();
                    parent.removeChild(ignoreNode);
                    ignoreNode = nextSibling;
                 }
                 //-------------------------------------------------
                 // Now remove any remaining image nodes.  These
                 // may exist because disposeImages=false ** this is just in
case
                 //-------------------------------------------------
                 Node imageNode = DOMUtilities.findNodeOfType(pNode,
IMAGE_ELEMENT_TAG_NAME);
                 while(imageNode != null)
                 {
                    imageNode.getParentNode().removeChild(imageNode);
                    imageNode = DOMUtilities.findNodeOfType(pNode,
IMAGE_ELEMENT_TAG_NAME);
                 }
```

Now, the handleRequest() method searches through the remaining tree for the text node contains `Home` and, once that is found, replaces it with the text `YourCo! Welcome!` This is a recursive routine that looks at the nodes and their children, trying to find the text node containing `Home`.

Example 5-23 handleRequest() part 4

```
if (startNode.hasChildNodes()){
NodeList children =  startNode.getChildNodes();
                   NodeList babes;
                   System.out.println("*** # of Children *** " +
children.getLength());
                   int k = 0; int i = 0;
                    for(i=0; k<children.getLength(); i++){
                     if (children.item(i).hasChildNodes()) {
                      babes = children.item(i).getChildNodes();
                      for(k=0; k<babes.getLength(); k++){
                       System.out.println("babes / type " +
babes.item(k).getNodeName() + "/" +
                            babes.item(k).getNodeType());
                       if (babes.item(k).getNodeType() ==
org.w3c.dom.Node.TEXT_NODE){
                        System.out.println("*** OLD text node value *** " +
babes.item(k).getNodeValue());
                        if (babes.item(k).getNodeValue().equals("Home")){
                              babes.item(k).setNodeValue("YourCo! Welcome!");
                         }
                        System.out.println("*** New home value *** " +
babes.item(k).getNodeValue());
                       }//* end if
                      } //* end for k
                     } //* end if
                    } //* end for i
                  } //* end if has childNodes
```

The final portion of the handleRequest() method performs the following:

- Generates the DomMegObject from the modified DOM,
- If no DomMegObject can be created, uses the input RequestEvent, and
- If neither of these include any content (null), generates an error, otherwise the appropriate object is written out.

Example 5-24 handleRequest() final part

```
WMLPrinter printer = new WMLPrinter();
                 domMegObject = new DomMegObject(doc, contentType, printer);
              }
              else
              {
                 System.err.println("Did not find node matching \"HOME \"");
              }
           }
        }
     }
     //---------------------------------------------------------------
     // If we don't have a DOM object generated, let's just get the
     // input page and use that as the output
     //-----------------------------------------------------------------
     if(domMegObject == null)
     {
        //-----------------------------------------------------------------
        // Grab the input page since using the DOM had a problem.
        //-----------------------------------------------------------------
        sOutputPage = getInputPage(reRequest);
     }
if ((domMegObject == null) && (sOutputPage.length() == 0))
{
        //--------------------------------------------------------------
        // Forward an error page since we got no output.  The created
        // page should have a translated page indicating the
        // generated page was empty
        //--------------------------------------------------------------
        forwardErrorPage(reRequest, 417, "TC_EMPTY_PAGE", WML_CONTENT_TYPE);
} else {
reduceHeader(reRequest);
if(domMegObject != null) {
           writeOutput(reRequest, domMegObject);
        } else {
           writeOutput(reRequest, sOutputPage);
        }
}
System.out.println("MainClipDomEditor done handling request");
  }
}//* end class
```

Compiling MainClipDom

In order to compile MainClipDom.java, we had to modify our .bat file to include JDK 1.3 and the following .jar files:

- xerces.jar
- xalan.jar
- log.jar
- htmltemplate.jar
- servlet.jar
- bsf.jar
- wtpcommon.jar
- wtpserver.jar
- wtpadmin.jar

Creating our property file

A text clipper will have one or more property files associated with it. We created a single property file containing both the plug-in registration information and the execution conditions. Our conditions indicate that MainClipDom will run when the URL contains `*/YourCo/index.html` (regardless of case) and when the content type is text/vnd.wap.wml. Our property file is as shown in Example 5-25.

Example 5-25   Property file

```
#Properties of MainClipDom
Class=MainClipDom
Description=Performs text clipping on YourCo Main page transcoded to WML
DescriptiveName=YourCo Main DOM-Based WML Text Clipper
Major=1
Minor=0
Name=YourCo Main DOM-Based WML Text Clipper
Condition=(url~*/YourCo/index.html) & (content-type=text/vnd.wap.wml)
Priority=3
```

Creating a .jar file

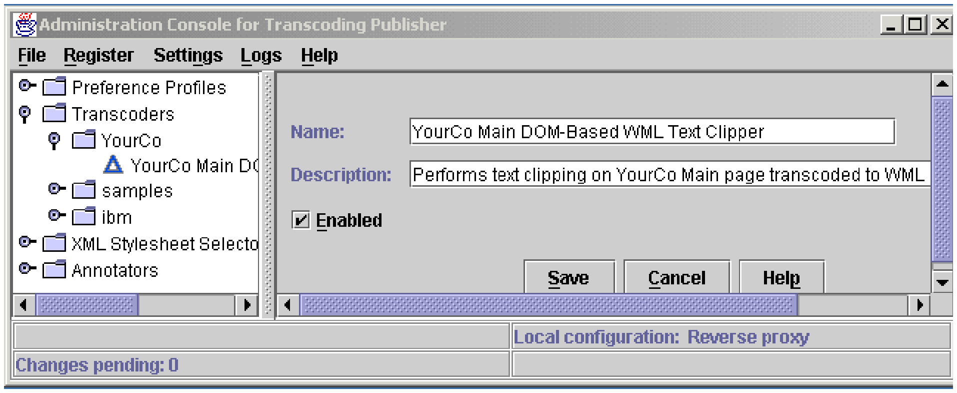
The Administration Console registers a new transcoder from a .jar file containing the class(es) and the property file. We used a makefile to create our. jar file; the makefile is shown below.

Example 5-26   Makefile

```
@REM Make the Plugin Jar file for the DOM-Based Text Clipper
jar -cf MainClipDom.jar MainClipDom.class MainClipDomEditor.class
YourCo\MainClipDom.prop
```

Registering the transcoder

We used the Administration Console to register our transcoder. The **Register->Transcoder** option started the wizard to register the transcoder. The wizard asked the location of the .jar file, and with this information was able to preload the other wizard panels. The MainClipDom registration is shown in Figure 5-33.



Name: YourCo Main DOM-Based WML Text Clipper

Description: Performs text clipping on YourCo Main page transcoded to WML

☑ Enabled

Local configuration: Reverse proxy

Changes pending: 0

Figure 5-33   Registration of the MainClipDom .jar file

After the transcoder was registered and the server was refreshed, we used the Nokia emulator to view our clipper; the final results are shown in Figure 5-34.

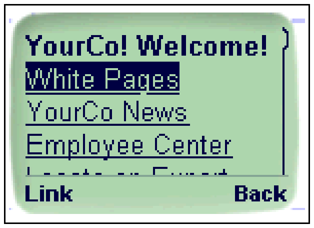


Figure 5-34 Text clipping results

5.6 Exporting and importing configuration data

WebSphere Transcoding Publisher version 4.0 has added Import and Export functions which use a WebSphere Transcoding Publisher-independent XML configuration file. This capability helps you manage your WebSphere Transcoding Publisher server environment very efficiently and effectively. WebSphere Transcoding Publisher has commands that enable you to:

- Capture and preserve key aspects of your existing configuration when you migrate to a new version of WebSphere Transcoding Publisher.
- Use the exported configuration file as a base for a new configuration for situations like these.
- Change the existing configuration because of directory path changes without going through the Administration Console.
- Add various style sheets and/or annotators to your configuration without registering them individually using the Administration Console.
- Replicate your configuration across other instances of WebSphere Transcoding Publisher.
- Import a Cocoon properties file, so you can use WebSphere Transcoding Publisher with a Cocoon-based application.
- Define your networks RMI registry server, which is used to notify WebSphere Transcoding Publisher servers of changes to server models.

The new XML configuration is an XML dialect, which contains the WebSphere Transcoding Publisher configuration information. The XML configuration file has information about the Setup resources; including local settings information and the WebSphere Transcoding Publisher resources. The information about the resources can be modified, deleted or added to fit your configuration needs and to make your configuration changes easier. The XML configuration contains the following types of setting and resources:

- Settings (basic WebSphere Transcoding Publisher settings; Firewall, Proxy Port, Reverse Proxy and Server setup)
- Annotators
- Preference profiles (Device, Network and User)
- Plug-ins (transcoders)
- Stylesheets

Note: The WebSphere Transcoding Publisher settings do not include any WebSphere Application Server settings.

It is important to understand that the XML configuration does not contain the resources themselves, but rather the registration information. The stylesheets (.xsl files), annotators (.ann files) and custom transcoders (.jar files) must be copied to the target machine and into the appropriate directories. The preference profiles are copied with the XML configuration, so you do not need to deal with them separately.

Copying the resource files should be done before you import the configuration, otherwise you will get warning errors for each item in the XML configuration file that does not already exist in the specified locations.

There are two ways in which the configuration information can be captured:

- Back up the resources in WebSphere Transcoding Publisher-dependent XML format, which is created directly from WebSphere Transcoding Publisher definitions without changes to the structure or names.
- Export the resources in WebSphere Transcoding Publisher-independent XML format, a very readable XML format that contains the same information as the WebSphere Transcoding Publisher-dependent format, but has more expressive element names.

You can use the Administration Console to import and export configuration information (using the XML configuration file). The XML configuration file allows you to copy information from one server to another. You can add or modify resources specifics contained within the XML configuration file without registering those resources using the Administration Console beforehand.

The Import and Export functions available through the Administration Console, within the File menu option, are:

- Export - creates a file containing an XML representation of the selected WebSphere Transcoding Publisher resources. You can select which resource types or specific resources to include.
- Export All - creates a file containing an XML representation of all the WebSphere Transcoding Publisher resources defined for this server or server model, plus the local settings defined for the server.
- Import - imports a file containing an XML representation of a set of WebSphere Transcoding Publisher resources. The XML configuration file will be validated as the file is read.

The backup and restore functions available through the Administration Console within the Settings menu option are:

- Backup - creates a WebSphere Transcoding Publishe- dependent format file with all the configuration information for the local server or the open server model. This information includes all WebSphere Transcoding Publisher resources and settings. You could use this for regularly scheduled server backups.
- Restore - restores a server or server model configuration that was created using the Backup function.

You can also issue the following commands outside the Administration Console:

- ExportResource - exports WebSphere Transcoding Publisher resources to the WebSphere Transcoding Publisher (independent format) XML configuration file or, if the -Node argument is included in the command, to a WebSphere Transcoding Publisher (dependent format) XML configuration.
- ImportResource - imports WebSphere Transcoding Publisher resources from the WebSphere Transcoding Publisher (independent format) XML configuration file or, if the -Node argument is included in the command, from a WebSphere Transcoding Publisher (dependent format) XML file.
- BackupResources - exports WebSphere Transcoding Publisher resources to the WebSphere Transcoding Publisher (dependent format) XML file.
- BackupConfig - exports WebSphere Transcoding Publisher configuration information to a WebSphere Transcoding Publisher (dependent format) XML file.
- RestoreResources - imports WebSphere Transcoding Publisher resources from the XML file created by a BackupResources Command.

- RestoreConfig - imports WebSphere Transcoding Publisher resources from the XML file created by a BackupConfig command.
- ImportCocoon - imports a Cocoon properties file to support Cocoon functionality.

These commands have various arguments, which are:

- The -File (filename) - specifies the name of the file to export to; the default is WebSphere Transcoding PublisherResouces.xml.
- The -Append - (for export commands only) appends new resources to the specified file; the default is No.
- The -ResourceType - (for export and import commands only) identifies the resources by the specified resource type (style sheet, annotator, plug-in, preference profile). The default is All Types.
- The -Node - (for export and import commands only) the definitions of the specified resources by node name. This requires the path of the resource.
- The -Comment - (for export and backup commands only) adds the specified text to the file as a comment.
- The -Encoding - encoding for the XML configuration file. The default is UTF-8.
- The -Help - displays help for the command. The default is False.
- The -Debug - writes messages to the console as records are written to the configuration file. The default is False.

It is possible to use the XML configuration file as a migration tool, but we will not be discussing this here. That topic is covered in the *WebSphere Transcoding Publisher Developer Guide* under the heading *Using XML configuration as a migration tool,* found at `www.ibm.com/software/webservers/transcoding/library.html`.

The XML configuration file allows you to manage your WebSphere Transcoding Publisher configuration very effectively. To modify the XML configuration file, you must edit it using the Export function. Then, import the modified configuration to the target WebSphere Transcoding Publisher servers. To minimize the copying or movement of the resources themselves, place them on a Web server instead of on the machine with the WebSphere Transcoding Publisher instance. When using a Web server, the resources can be accessed by a WebSphere Transcoding Publisher server using HTTP.

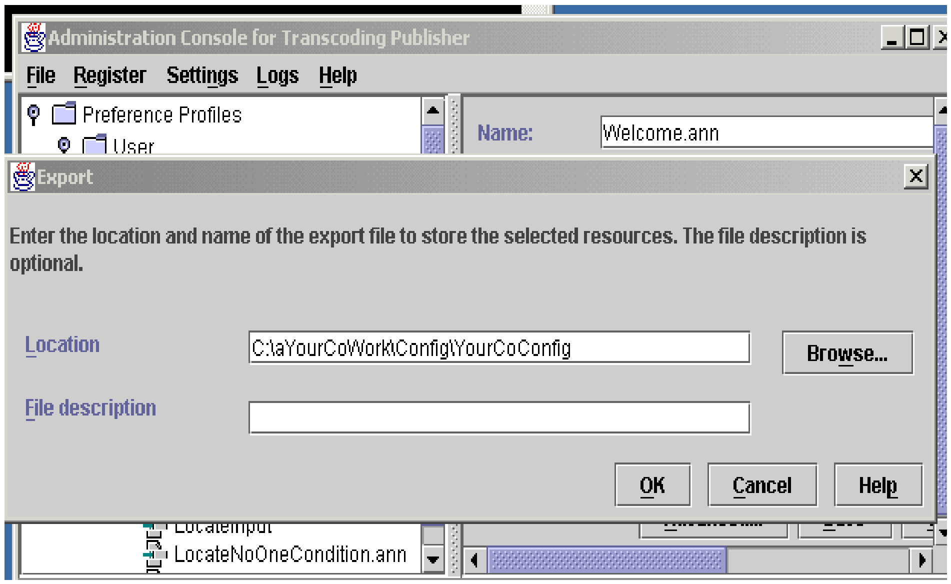
5.6.1 Sample scenario: export and import configuration

In this scenario, we are setting up another machine with our WebSphere Transcoding Publisher development environment configuration and we want to change the directory location of the annotators. We are using:

- The Administration Console (on the development machine) to export the XML configuration,
- Notepad to modify the configuration, and
- The Administration Console (on the target machine) to import the configuration.

Prior to exporting the configuration, we saved the annotation files (.ann), created a new directory on the target machine and copied the annotation files into the specified directory. Next, we performed the following steps:

1. Using the Administration Console (on the development machine), we selected the **Export All** option from the File entry in the menu bar. We specified the directory in which to store the configuration file and its name, as shown in Figure 5-35.



Enter the location and name of the export file to store the selected resources. The file description is optional.

Location: C:\aYourCoWork\Config\YourCoConfig

File description:

Figure 5-35 Exporting the configuration

2. By using the **Export All** option, the XML configuration file included the preference profile settings, so our changes to the Wireless Phone - WAP device profile (Remove images and Convert tables to rows) were included in

the configuration. The portion of the XML configuration file containing our annotators information is shown in Example 5-27.

Example 5-27 XML Configuration - an annotation

```
<Annotator><Folder>ibm/itsotest</Folder>
<SelectorName>Welcome</SelectorName>
<Name>Welcome</Name>
<Description>Stream line YourCo Welcome</Description>
<URL>*/YourCo/index.html</URL>
<Location>C:/aYourCoWork/Annotations/Welcome.ann</Location>
<Enable>true</Enable>
</Annotator>
<Annotator><Folder>ibm/itsotest</Folder>
<SelectorName>LocateInput</SelectorName>
<Name>LocateInput</Name>
<Description>This ann file is for the Locate Input html within YourCo</Description>
<URL>*/LocateInput.html</URL>
<Location>C:/aYourCoWork/Annotations/LocateInput.ann</Location>
<Keys><Key Name="User_Agent">Nokia-WAP-Toolkit/2.1</Key>
</Keys>
<Enable>true</Enable>
</Annotator>
<Annotator><Folder>ibm/itsotest</Folder>
<SelectorName>LocateNoOneCondition.ann</SelectorName>
<Name>LocateNoOneCondition.ann</Name>
<Description>Fix No expert clash with Found expert .ann file</Description>
<URL>*/itso.wes.lbs.samples.FindExpert</URL>
<Location>C:/aYourCoWork/Annotations/LocateNoOneCondition.ann</Location>
<Keys><Key Name="condition">zzz=false</Key>
</Keys>
<Enable>true</Enable>
</Annotator>
<Annotator><Folder>ibm/itsotest</Folder>
<SelectorName>LocateFoundExpert.ann</SelectorName>
<Name>LocateFoundExpert.ann</Name>
<Description>Locate the Found Expert Annotation</Description>
<URL>*/itso.wes.lbs.samples.FindExpert</URL>
<Location>C:/aYourCoWork/Annotations/LocateFoundExpert.ann</Location>
<Keys><Key Name="User_Agent">Nokia-WAP-Toolkit/2.1</Key>
<Key Name="condition">zzz=true</Key>
</Keys>
<Enable>true</Enable>
</Annotator>
```

3. Next, we editted information about our annotators; our edits included:
   a. Changing some of our annotator names and descriptions
   b. Changing our annotators’ directory name from aYourCoWork to YourCo

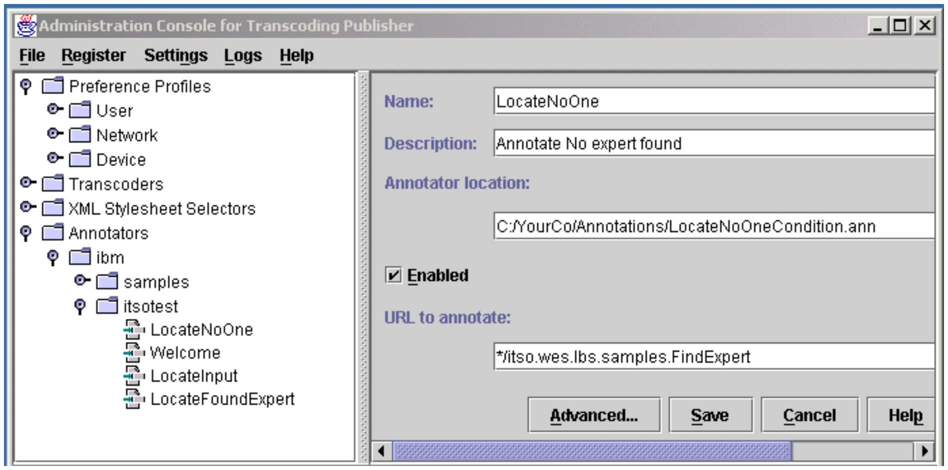
The same portion of the modified XML configuration file is shown in Example 5-28.

Example 5-28 Modified XML configuration

```
<Annotator><Folder>ibm/itsotest</Folder>
<SelectorName>Welcome</SelectorName>
<Name>Welcome</Name>
<Description>YourCo Welcome Page</Description>
<URL>*/YourCo/index.html</URL>
<Location>C:/YourCo/Annotations/Welcome.ann</Location>
<Enable>true</Enable>
</Annotator>
<Annotator><Folder>ibm/itsotest</Folder>
<SelectorName>LocateInput</SelectorName>
<Name>LocateInput</Name>
<Description>Annotate Locate Input portion of Locate Expert</Description>
<URL>*/LocateInput.html</URL>
<Location>C:/YourCo/Annotations/LocateInput.ann</Location>
<Keys><Key Name="User_Agent">Nokia-WAP-Toolkit/2.1</Key>
</Keys>
<Enable>true</Enable>
</Annotator>
<Annotator><Folder>ibm/itsotest</Folder>
<SelectorName>LocateNoOne</SelectorName>
<Name>LocateNoOne</Name>
<Description>Annotate No expert found</Description>
<URL>*/itso.wes.lbs.samples.FindExpert</URL>
<Location>C:/YourCo/Annotations/LocateNoOneCondition.ann</Location>
<Keys><Key Name="User_Agent">Nokia-WAP-Toolkit/2.1</Key>
<Key Name="condition">zzz=false</Key>
</Keys>
<Enable>true</Enable>
</Annotator>
<Annotator><Folder>ibm/itsotest</Folder>
<SelectorName>LocateFoundExpert</SelectorName>
<Name>LocateFoundExpert</Name>
<Description>Annotate found Expert</Description>
<URL>*/itso.wes.lbs.samples.FindExpert</URL>
<Location>C:/YourCo/Annotations/LocateFoundExpert.ann</Location>
<Keys><Key Name="User_Agent">Nokia-WAP-Toolkit/2.1</Key>
<Key Name="condition">zzz=true</Key>
</Keys>
<Enable>true</Enable>
</Annotator>
```

4. On the target machine, we used the Administration Console to import the modified configuration. Figure 5-36 on page 210 shows the modified annotator for LocateNoOne. Notice that the name changed from

LocateNoOneCondition.ann to LocateNoOne, the description has changed, and the annotator location (directory) has changed.



Name: LocateNoOne

Description: Annotate No expert found

Annotator location: C:/YourCo/Annotations/LocateNoOneCondition.ann

[✓] Enabled

URL to annotate: */itso.wes.lbs.samples.FindExpert

Figure 5-36 Annotator in target WebSphere Transcoding Publisher server

5. Our final activity was to start the Request Viewer and the Nokia Emulator to view the YourCo Locate Expert Application. Needless to say, everything ran smoothly.

6

Using stylesheets

In this chapter on WebSphere Transcoding Publisher Version 4.0., we discuss using eXtensible Stylesheet Language (XSL) to convert XML document content to HTML and WML. Many Web applications generate HTML for their static content and use XML documents to represent their dynamic content. WebSphere Transcoding Publisher uses XSL stylesheets to transform a XML documents content to the desired output format. This chapter shows:

- The enhancements to WebSphere Transcoding Publisher Version 4.0 for stylesheet processing
- How to register XSL stylesheets
- How to use parameters in XSL stylesheets
- How to use XSL stylesheets to internationalize the XML document content
- How to use the new XSL editor to create XSL stylesheets to generate XHTML

6.1 Overview

The eXtensible Markup Language (XML) has become the de facto standard for representing business data when that data is outside of a convention data store (like DB2). These XML documents are being used in many business to business (B2B) exchanges, often as an extension to or a replacement for EDI (Electronic Data Interchange). It is only natural that this same dynamic data, represented as XML, would be used to extend Web site information. Many applications now use XML as a means to represent the dynamic data that is available in Web applications today.

XML, by its very nature, is a data representation mechanism. Its tagging defines the associated data; this makes it useful in a B2B environment. But when XML data is sent to a browser, it just displays the data tags. XML documents are not browser friendly because a browser looks for presentation tags (that is, tags that tell the browser how to present the information) that HTML and other presentation markup languages provide. eXtensible stylesheet Language (XSL) is used to convert an XML document into other XML dialects and HTML. XSL stylesheets are used by WebSphere Transcoding Publisher to convert XML documents into HTML and other display XML dialects, such as WML.

WebSphere Transcoding Publisher uses two standards-based technologies, the Xerces-Java XML parser and the Xalan-Java stylesheet processor, as the foundation for its XML transcoding function. WebSphere Transcoding Publisher uses the Xerces-Java XML parser primarily to create DOM (Document Object Model) tree representations of the XML document and the stylesheet, so that they are easy to access and handle. These DOM trees are input to the Xalan-Java processor, which formats and transforms the XML document based on the associated stylesheet specifications. Both of these technologies are open-source software packages available from the Apache Software Foundation.

There are three major benefits when using WebSphere Transcoding Publisher to support XML documents and XSL stylesheets:

- WebSphere Transcoding Publisher helps you organize your stylesheets locally in a file system or remotely in a Web server.
- WebSphere Transcoding Publisher allows you to configure the selection criteria, such that a specific stylesheet is applied to an XML document, based on the current environment.
- Based on various dynamic conditions (HTTP header information, preference profiles, and configured selection criteria), WebSphere Transcoding Publisher selects the stylesheet to be applied on the fly.

For more details on WebSphere Transcoding Publisher and stylesheet processing, look at the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5,* SG24-6233.

6.1.1 WebSphere Transcoding Publisher Version 4.0 enhancements

WebSphere Transcoding Publisher Version 4.0 includes various improvements and additions to its existing XSL stylesheet support. These enhancements are:

- Enhance condition support for stylesheet selection.
- Support for Cocoon-style (Apache style) specification of embedded stylesheet links within an XML document.
- Support for WebSphere Transcoding Publisher condition-based specification of embedded stylesheet links within an XML document. This aids in selection of the best or right stylesheet from multiple stylesheets.

Stylesheets provide WebSphere Transcoding Publisher with instructions on how an XML document should be transformed. Stylesheets can be associated with an XML document in two ways:

- External to the XML document - these stylesheets are provided to WebSphere Transcoding Publisher by registering them using the Administration Console.
- Internal to the XML document - these stylesheets are embedded in the XML document. It is the responsibility of the XML document to identify which stylesheet(s) WebSphere Transcoding Publisher should use for XML transcoding.

WebSphere Transcoding Publisher, by default, looks for internal stylesheets only if no appropriate external (registered) stylesheets are found. You can change this default method of processing by changing the value of checkForDocumentStylesheetsFirst to `true`, in the etc/plugins/ibm/TextEngine/XMLHandler.prop file. When this variable is set to `true`, WebSphere Transcoding Publisher examines each XML document looking for the wtp-condition or xml-stylesheet processing (relates to Cocoon addition) statement, to help determine which stylesheet to use.

Now, let us look at these enhancements a little more closely.

Enhanced condition support

WebSphere Transcoding Publisher has added three new field names for use in both internal and external stylesheet conditioning; they are:

- device - the device profile name
- network - the network profile name
- user - the user profile name

For example, instead of having to know how to match the user-agent information in the HTTP header to create a stylesheet condition, you can specify the related device profile name. These field names, when used in stylesheet conditions, map to the preference profile file names, notto the translated description names. For example,

- device: WML-Device
- network: wireless

You can find the device profile names under /etc/preference/device and the network profile names under /etc/preference/network. These three field names (device, network, user) are in addition to the HTTP header fields (like user-agent) which are already supported on stylesheet condition statements.

Cocoon-style embedded stylesheet link support

The other method for internal stylesheet selection is based on a function within Cocoon which is an open source technology. This function maps stylesheets to XML documents based on the target device. The mechanism to identify the target device is the media tag within an XML document's xml-stylesheet processing instruction(s). The xml-stylesheet processing instruction gives a link to a stylesheet, for example:

Example 6-1 Cocoon media link

```
<?xml-stylesheet href=”LocateExpertWML.xsl” type=”text/xsl” media=”lynx” ?>
```

WebSphere Transcoding Publisher uses the etc/Cocoon.prop file to get ordered definitions to associate with the media tag. During initialization, WebSphere Transcoding Publisher takes these entries and makes internal structured condition rules for them.

Condition-based specification of embedded stylesheet links

Prior to WebSphere Transcoding Publisher 4.0, the XML transcoder used the xml-stylesheet element for stylesheet directives within an XML document. Now WebSphere Transcoding Publisher introduces the wtp-condition processing instruction. Several wtp-condition processing instructions can be in the prologue of an XML document, specifying the different stylesheets to use for different conditions. These instructions can use the values of any HTTP header field and any preference profile in effect. As stated in the enhancements, WebSphere Transcoding Publisher 4.0 has added three new field names (device, network, user) which can be used as wtp-conditions. Let us look at an example wtp-condition instruction:

Example 6-2 wtp-condition example

```
<?wtp-condition stylesheet=”file://theexample.xsl:
condition=”(url=*/LocateExpert)” ?>
```

With the wtp-condition processing instruction, you can:

- Match on more than just the user-agent field.
- Check on profiles, for instance, which device profile was chosen using a new device condition.
- Have AND conditions together using `&amp;` between the conditions.
- Have OR conditions using the | (vertical bar).

When an XML document is being processed, WebSphere Transcoding Publisher looks at the xml-stylesheet processing instruction and examines the media tag. WebSphere Transcoding Publisher examines all structured conditions stored in its table under that media tag and tries to find a match. If there is a match, the associated stylesheet is applied.

Specifying output content-type in the stylesheet

Previous versions of WebSphere Transcoding Publisher looked at the value of the content-type stylesheet variable to determine the output content type. Instead of using this approach, WebSphere Transcoding Publisher now uses the XSL support for specifying the output content type with the media-type attribute on the xsl:output element. Setting the output content type is only necessary if:

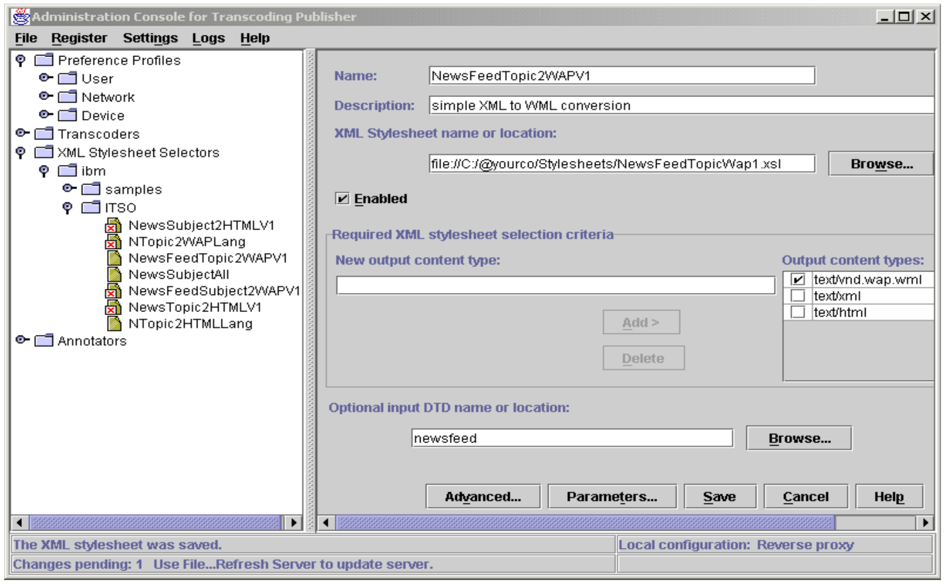
- The registered stylesheet indicates that it could generate more than one content type.
- The stylesheet is determined using an embedded stylesheet link within the XML document.

If the media type is not present, WebSphere Transcoding Publisher determines the media types.

6.1.2 XSL stylesheet administration

The WebSphere Transcoding Publisher Administration Console is the tool used to access all the resources with WebSphere Transcoding Publisher. For more details on the Administration Console, look at the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5*, SG24-6233.

Among other functions, the Administration Console is used to register your stylesheets and identify the conditions which determine when the stylesheet is applied to the XML document. A registered stylesheet is shown below.



Name: NewsFeedTopic2WAPV1
Description: simple XML to WML conversion
XML Stylesheet name or location: file://C:/@yourco/Stylesheets/NewsFeedTopicWap1.xsl
[✓] Enabled
Required XML stylesheet selection criteria
New output content type:
Output content types:
[✓] text/vnd.wap.wml
[ ] text/xml
[ ] text/html
Optional input DTD name or location: newsfeed
The XML stylesheet was saved.
Local configuration: Reverse proxy
Changes pending: 1 Use File...Refresh Server to update server.

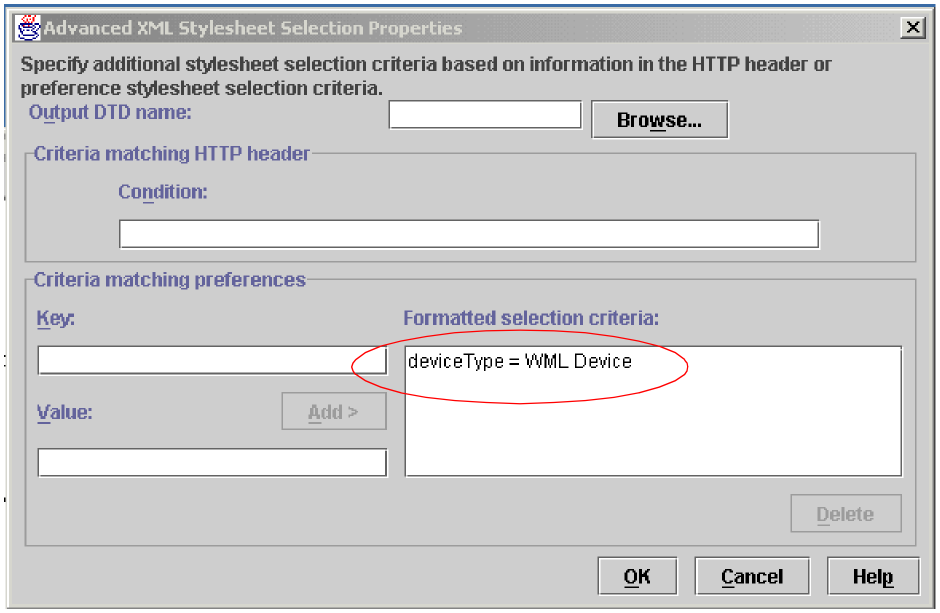
Figure 6-1 Administration Console with a registered stylesheet

When you select a stylesheet in the Stylesheet selector subtree, the console displays all the basic settings (in the right pane) relevant to it. The information displayed includes:

- Name/ Description - the name and description of the stylesheet used within WebSphere Transcoding Publisher. Note that this is the stylesheet registration name (not to be confused with the actual stylesheet file name).
- Stylesheet name or location - path (file path or URL) to where the stylesheet is located.

- Required Stylesheet selection criteria - output content type for the request. By default, it can be either HTML, XML or WML, but the administrator has the ability to add new output types by entering the corresponding MIME type.
- Optional input DTD - the selection criteria can be restricted by using this field, where the administrator enters the file path or URL of the DTD for which the stylesheet will be applied. The input DTD can be the DOCTYPE of the XML document or the PUBLIC name of the DTD.

One of the buttons at the bottom of this screen is the Advanced button, which invokes the Advanced Stylesheet Selection Properties display, shown below. This display allows you to provide more stylesheet selection criteria, by specifying an output DTD, conditions on the HTTP request header, and/or conditions on the preference profile.


Advanced XML Stylesheet Selection Properties

Specify additional stylesheet selection criteria based on information in the HTTP header or preference stylesheet selection criteria.

Output DTD name:

Criteria matching HTTP header

Condition:

Criteria matching preferences

Key:

Value:

Formatted selection criteria:

deviceType = WML Device

Figure 6-2 Advanced stylesheet selection criteria

The selection criteria matching for the HTTP header can be a complex formula using the AND or OR conjunctions as well as parentheses. WebSphere Transcoding Publisher can read all the header attributes of the HTTP request if needed for matching.

The criteria matching preferences requires key/value pairs that will match the key/value pair in one of the profiles (device, network or user profiles). For example, as shown in Figure 6-2, a key of `deviceType` and a value of `WML Device` matches to that key value pair in the WML Device profile.

Using parameters

A business may have a very large number of XSL stylesheets to meet the XML document customization requirements. Often these stylesheets are very similar in the transformations they perform. For example, a business may have a stylesheet for each output type it needs to generate from an XML document. A business usually has a wide variety of different XML documents that are transformed by an associated group of stylesheets. As the variety and variations of stylesheets increase, the business has difficulty in managing the proliferation of stylesheets.

There is a variety of methods for dealing with this proliferation of stylesheets. For example, the Administration Console helps you organize your stylesheets into folders within the XML Stylesheet Selector tree. These folders can be named according to your own meaningful scheme. Another option is to transform the XML document into a common intermediate format, HTML, and let WebSphere Transcoding Publisher transcode the document to the appropriate device format; for details, review the article on IBM developerWorks, *Spinning your XML for screens of all sizes Using HTML as an intermediate markup language,* found at `www.ibm.com/software/webserver/transcoding/library.html`. Another option, the one we are discussing here, is to use parameters within stylesheets to determine which actions within template rules apply.

Using parameters within a stylesheet can help you deal with the proliferation of stylesheets. The parameters can be defined outside the stylesheet and then passed to the stylesheet for processing. For example, you might choose to use a single stylesheet for all output types and use parameters in the stylesheet to select the type of output to be generated. Example 6-3 shows two stylesheet parameters.

Example 6-3 XSL Stylesheet parameter

```
<xsl:param name=”parm1”>
<xsl:param name=”parm2” select=”default”>
```

WebSphere Transcoding Publishers Text Engine attempts to find the value of each stylesheet parameter in preparation for processing in the following places:

- Configured into stylesheets - the stylesheet registration can define key/value pairs which are used as parameters.
- HTTP header - parameter values in the HTTP request and response header.
- Preference profiles - the preference aggregator is checked for any parameter values defined in the preference profiles.

Where the particular parameter is defined determines the availability of that parameter. For example, if you want to limit the parameter's availability to a specific stylesheet, define the parameter within the stylesheet registration. Defining the parameter in the stylesheet registration ensures the parameter is available for each execution of the stylesheet. However, to associate a parameter with a particular target device, define the parameter in the device profile.

The parameters associated with profiles and stylesheets are defined using the Administration Console. For stylesheets, the parameters are defined as key/value pairs in the stylesheet registrations Stylesheet Parameters window, found by selecting the **Parameter** button on the Administration Console. For device profiles, the parameters are defined as key/value pairs in the Advanced Preference Profile Properties window. For more details on setting parameters, look at the IBM redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5,* SG24-6233.

Parameters can be used in sophisticated ways to effect the processing of stylesheets. Parameters can be used to:

- Control inclusion of other stylesheets by adjusting the file name or URL of a stylesheet you want. By using the `xsl:choose` statement in your stylesheet, you can control which of several includes is executed, based on parameters you provide.
- Setting the output type, to group stylesheets by group name.
- Reduce the number of stylesheets needed for different devices; use the deviceType parameter to drive which template rules are executed, which stylesheets are included, based on specific differences.

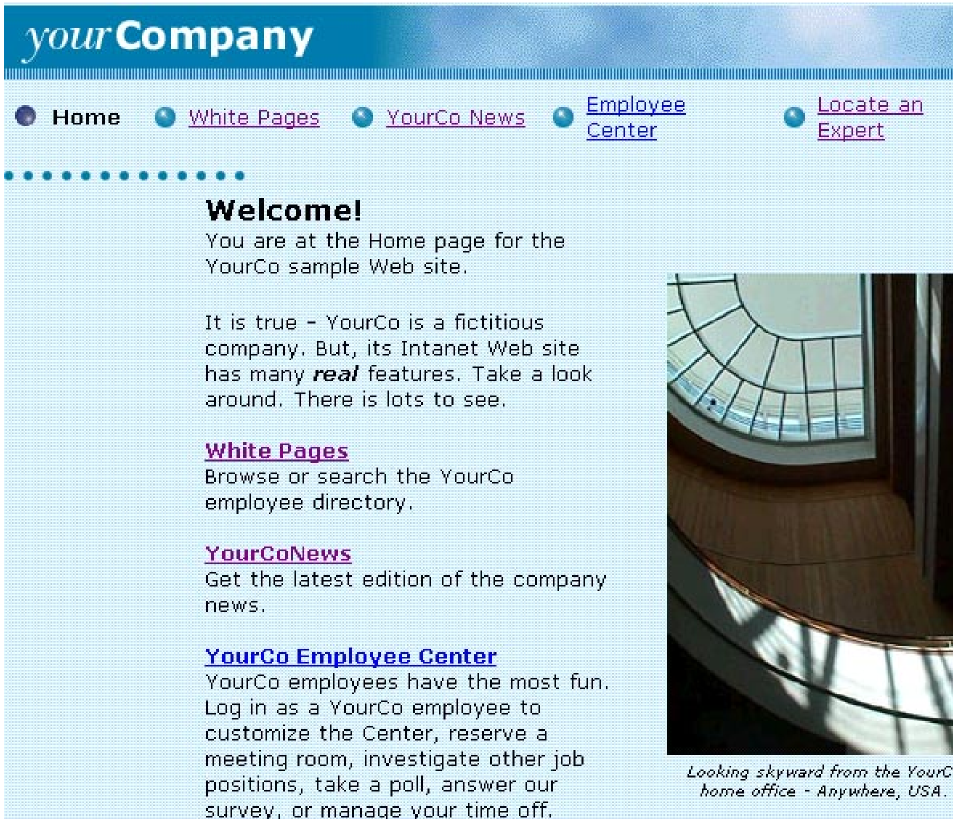
Parameters can go a long way in reducing the number of stylesheets required to support your environment. However, overuse of parameters produces complexity as well. For example, trying to use a single stylesheet to support a variety of unrelated devices (using different markup languages) could be cumbersome. However, tuning the output generated to a specific set of similar devices would be a natural use of parameters.

Sample scenario for parameterization

For this sample scenario, the WebSphere Everyplace News application within the sample YourCo Web application (used in this redbook) will be used. WebSphere Everyplace News is an extension to the YourCo New application.

The application flow (using a desktop browser and starting from the YourCo main page) is as follows:

1) Select **YourCo News** from the YourCo main menu.



Welcome!
You are at the Home page for the YourCo sample Web site.

It is true - YourCo is a fictitious company. But, its Intanet Web site has many ***real*** features. Take a look around. There is lots to see.

White Pages
Browse or search the YourCo employee directory.

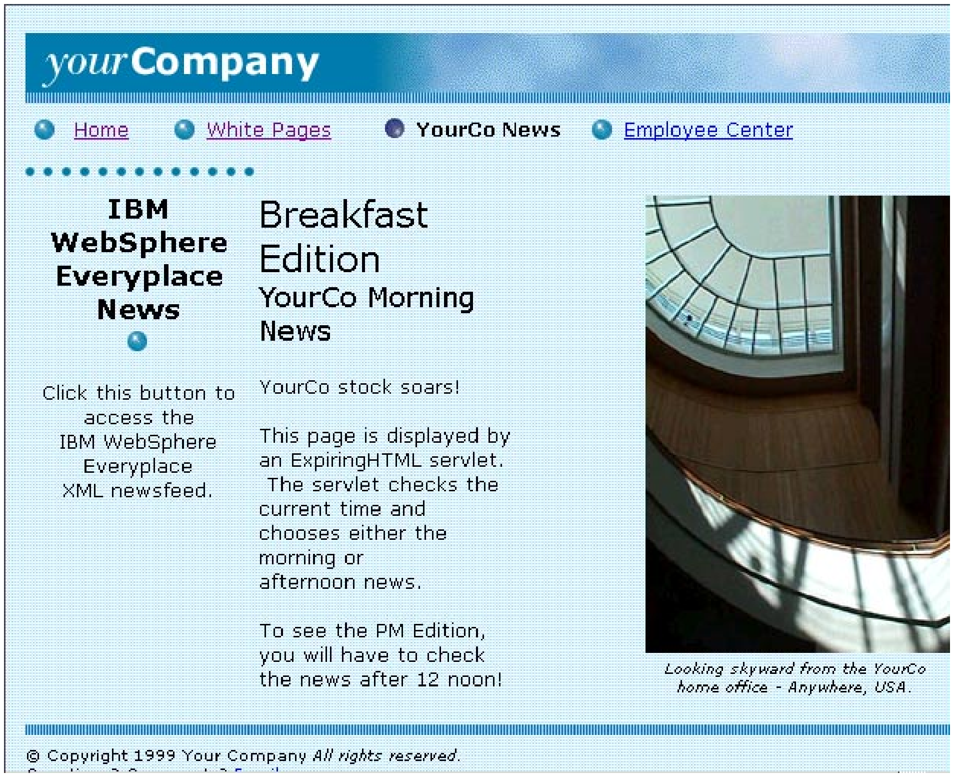
YourCoNews
Get the latest edition of the company news.

YourCo Employee Center
YourCo employees have the most fun. Log in as a YourCo employee to customize the Center, reserve a meeting room, investigate other job positions, take a poll, answer our survey, or manage your time off.

Looking skyward from the YourC home office - Anywhere, USA.

Figure 6-3 Selecting YourCo News

2) Click the **WebSphere Everyplace News** icon.



IBM WebSphere Everyplace News

Click this button to access the IBM WebSphere Everyplace XML newsfeed.

Breakfast Edition

YourCo Morning News

YourCo stock soars!

This page is displayed by an ExpiringHTML servlet. The servlet checks the current time and chooses either the morning or afternoon news.

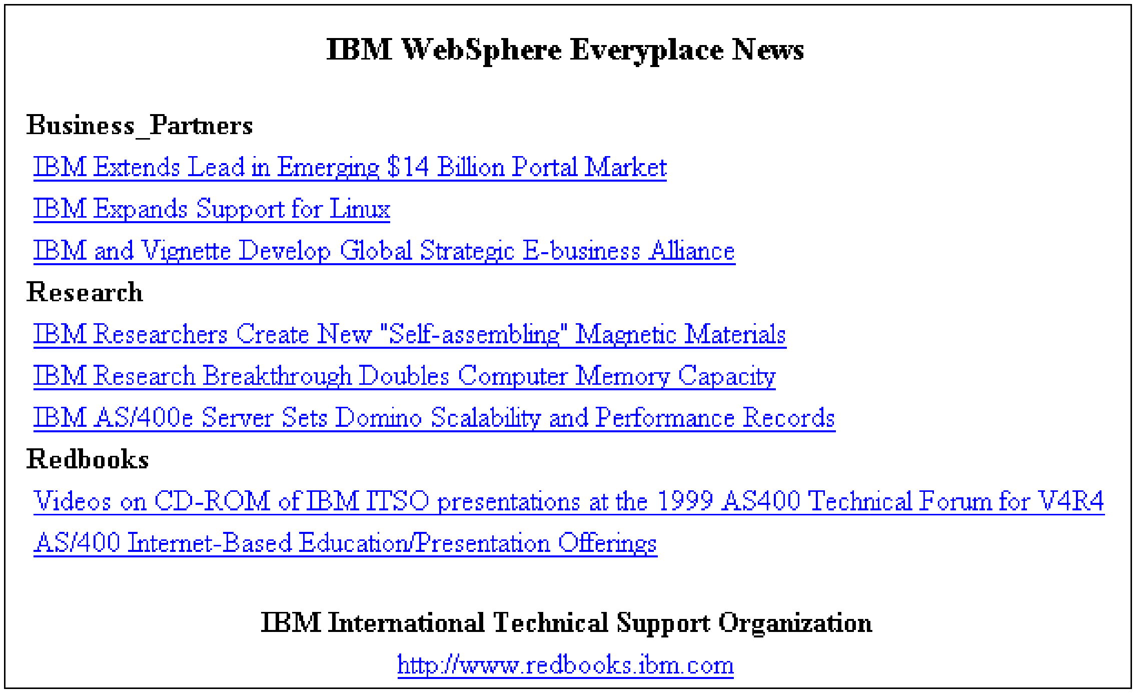
To see the PM Edition, you will have to check the news after 12 noon!

Looking skyward from the YourCo home office - Anywhere, USA.

© Copyright 1999 Your Company *All rights reserved.*

Figure 6-4 Selecting IBM WebSphere Everyplace News

3) Select an article (listed by title) from within one of the three categories (Business Partner, Research, Redbook).


IBM WebSphere Everyplace News

Business_Partners

IBM Extends Lead in Emerging $14 Billion Portal Market

IBM Expands Support for Linux

IBM and Vignette Develop Global Strategic E-business Alliance

Research

IBM Researchers Create New "Self-assembling" Magnetic Materials

IBM Research Breakthrough Doubles Computer Memory Capacity

IBM AS/400e Server Sets Domino Scalability and Performance Records

Redbooks

Videos on CD-ROM of IBM ITSO presentations at the 1999 AS400 Technical Forum for V4R4

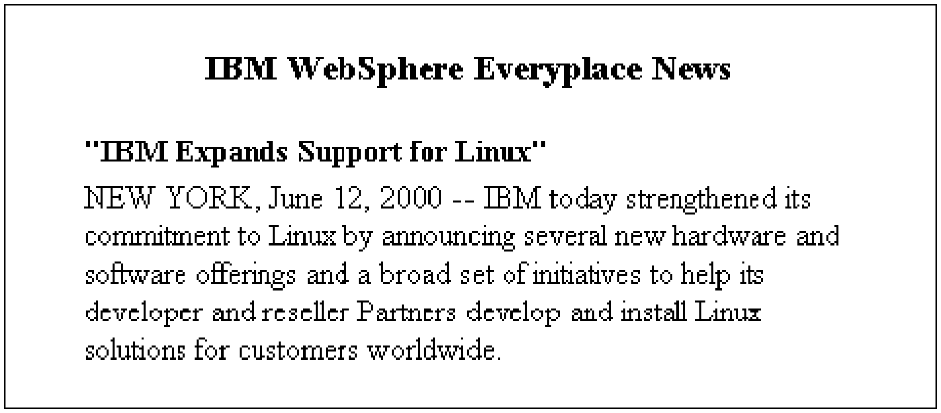
AS/400 Internet-Based Education/Presentation Offerings

IBM International Technical Support Organization

http://www.redbooks.ibm.com

Figure 6-5 Display topics and select a topic

4) The details of the article are displayed as in Figure 6-6.


IBM WebSphere Everyplace News

"IBM Expands Support for Linux"

NEW YORK, June 12, 2000 -- IBM today strengthened its commitment to Linux by announcing several new hardware and software offerings and a broad set of initiatives to help its developer and reseller Partners develop and install Linux solutions for customers worldwide.

Figure 6-6 Displaying an article

The WebSphere Everyplace News presents its data in XML documents. There are two XML document which contain the following:

- News topics - a single document containing the titles of the news article, by category.
- News article - a document of a news article containing the title and the story details.

The news topics XML document including its Document Type Definition (DTD) is shown in the following example.

Example 6-4 News feed topic XML document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE newsfeed [
<!ELEMENT newsfeed(news+)>
<!ATTLIST news category CDATA #REQUIRED>
<!ELEMENT news(item+)>
<!ELEMENT item(title)>
<!ELEMENT title (#PCDATA)>
<!ATTLIST item reference CDATA #REQUIRED>
 ]>
<newsfeed>
<news category="Business_Partners">
<item reference="PortalMarket_story">
<title>IBM Extends Lead in Emerging $14 Billion Portal Market</title>
   </item>
   <item reference="SupportLinux_story">
     <title>IBM Expands Support for Linux</title>
    </item>
   <item reference="Vignette_story" >
      <title>IBM and Vignette Develop Global Strategic E-business Alliance
</title>
   </item>
 </news>
 <news category="Research">
   <item reference="Magnetic_story">
     <title>IBM Researchers Create New "Self-assembling" Magnetic Materials
</title>
   </item>
   <item reference="Memory_story" >
    <title>IBM Research Breakthrough Doubles Computer Memory Capacity</title>
   </item>
   <item reference="Dominorecords_story">
    <title>IBM AS/400e Server Sets Domino Scalability and Performance Records
</title>
   </item>
 </news>
 <news category="Redbooks">
```

```
    <item reference="ITSOPresentations_story" >
      <title>Videos on CD-ROM of IBM ITSO presentations at the 1999 AS400 
Technical Forum for V4R4</title>

    </item>
    <item  reference="as400_story" >
      <title>AS/400 Internet-Based Education/Presentation Offerings</title>
    </item>
 </news>
</newsfeed>
```

For each news title, there is an associated news article XML document file. We included one of the news article XML document files, including its DTD (in this case PortalMarket_story.xml) in the following example.

Example 6-5 News Article XML document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE newsitem [
<!ELEMENT newsitem (subject, story)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT story (#PCDATA)>
 ]>
<newsitem>
<subject>"IBM Extends Lead in Emerging $14 Billion Portal Market"</subject>
<story>
  SAN JOSE, Calif., June 21, 2000 -- IBM today extended its lead in redefining 
the management market and introduced the IBM Enterprise Information Portal 
(EIP) Version 7, the most advanced portal solution in the industry. Now 
customers can access and organize relevant business information whatever the 
format, wherever it resides and slash development time nearly in half.
</story>
</newsitem>
```

Phase 1 - Simple Conversions

We created two sets of XSL stylesheets used to transform and format the XML documents into HTML and WML, respectively. The stylesheets to transform the XML documents into HTML are:

- A stylesheet to transform the news feed XML document into HTML
- A stylesheet to transform a news article XML document into HTML

The news feed stylesheet (NewsTopictoHTML1.xsl) is shown in Example 6-6.

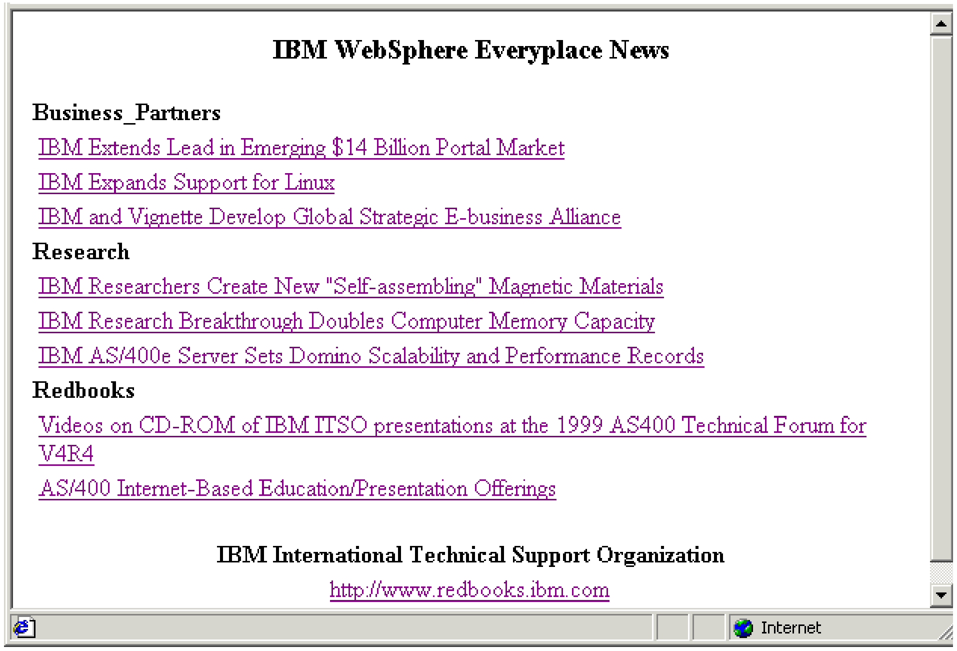
Example 6-6 News feed XML Document to HTML

```
<?xml version='1.0'?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
  xmlns="http://www.w3.org/1999/xhtml" version="1.0">
<xsl:output method="html"/>
<!-------------- match on the Document Root ---------->
<xsl:template match="newsfeed">
  <html>
  <head>
   <title>IBM WebSphere Everyplace News - By Subject and Headline</title>
  </head>
  <body bgcolor="#FFFFFF">
  <!-- header -->
  <center><h3>IBM WebSphere Everyplace News </h3>
  <table border="0">
    <xsl:apply-templates/>
  </table>
  <!-- footer -->
  <br/>
  <table border="0">
    <tr align="center"><td><b>IBM International Technical Support
Organization</b></td></tr>
    <tr align="center"><td><a
href="www.redbooks.ibm.com">http://www.redbooks.ibm.com</a></td></tr>
  </table>
  </center>
  </body>
  </html>
</xsl:template>
<!-------------- match on the news element ---------->
<xsl:template match="news">
  <tr>
    <td colspan="2">
        <b><xsl:value-of select="@category"/></b>
    </td>
  </tr>
  <xsl:apply-templates/>
</xsl:template>
<!-------------- match on the item element ---------->
<xsl:template match="item">
<tr>
    <td></td>
    <td>
      <xsl:text disable-output-escaping="yes">&lt;a
href="ServeXML_news?xml_file=</xsl:text>
<xsl:value-of select="@reference"/>
      <xsl:text disable-output-escaping="yes">.xml"&gt;</xsl:text>
         <xsl:apply-templates/>
      <xsl:text disable-output-escaping="yes">&lt;a&gt;</xsl:text>
     </td>
   </tr>
</xsl:template>
```

```
<!-------------- match on the title element ---------->
<xsl:template match="title">
       <xsl:value-of select="."/>
</xsl:template>
</xsl:stylesheet>
```

The results of applying the above stylesheet to the news feed XML document are shown below. First, you see the generated HTML displayed in a browser, and then a subset of the source HTML.


IBM WebSphere Everyplace News

Business_Partners

IBM Extends Lead in Emerging $14 Billion Portal Market

IBM Expands Support for Linux

IBM and Vignette Develop Global Strategic E-business Alliance

Research

IBM Researchers Create New "Self-assembling" Magnetic Materials

IBM Research Breakthrough Doubles Computer Memory Capacity

IBM AS/400e Server Sets Domino Scalability and Performance Records

Redbooks

Videos on CD-ROM of IBM ITSO presentations at the 1999 AS400 Technical Forum for V4R4

AS/400 Internet-Based Education/Presentation Offerings

IBM International Technical Support Organization

http://www.redbooks.ibm.com

Figure 6-7 Browser view of topics (simple conversion)

Example 6-7 Generated HTML for Topics (simple conversion)

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>IBM WebSphere Everyplace News - By Subject and Headline</title>
</head>
<body bgcolor="#FFFFFF">
<center>
<h3>IBM WebSphere Everyplace News </h3>
<table border="0">
 <tr>
<td colspan="2">
<b>Business_Partners</b>
```

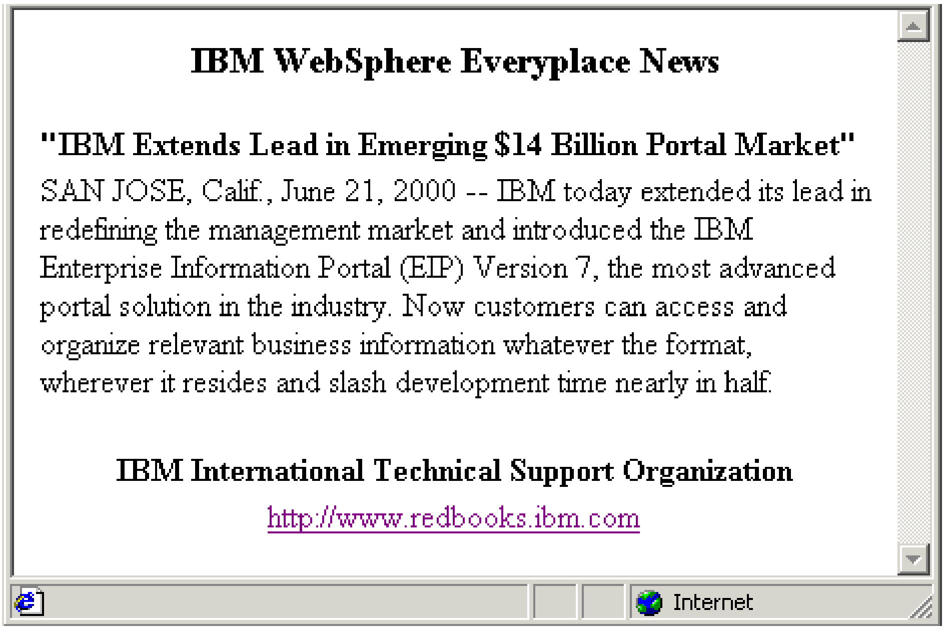
```
</td>
</tr>
   <tr>
<td/>
<td><a
href="http://9.24.105.152:80/httppvc_clnss9.24.104.13/WebSphereSamples/servlet/
ServeXML_newspvc_qxml_file=PortalMarket_story.xml">
      IBM Extends Lead in Emerging $14 Billion Portal Market
   <a></td>
</tr>
   <tr>
<td/>
<td><a href="ServeXML_news?xml_file=SupportLinux_story.xml">
     IBM Expands Support for Linux
    <a></td>
</tr>
   <tr>
<td/>
<td><a href="ServeXML_news?xml_file=Vignette_story.xml">
      IBM and Vignette Develop Global Strategic E-business Alliance
   <a></td>
</tr>
<!-------*** other categories and their entries are here *** ---------->
</table>
</center>
</body>
</html>
```

Next, we show the stylesheet used to convert the news article to HTML.

Example 6-8 News Article XML to HTML stylesheet

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.w3.org/1999/xhtml" version="1.0">
<!-- --------------- Match on the ROOT ----------->
<xsl:template match="newsitem">
  <html>
  <head>
    <title>IBM WebSphere Everyplace News - By Headline</title>
  </head>
  <body bgcolor="#FFFFFF">
  <center>
  <h3>IBM WebSphere Everyplace News</h3>
  </center>
<!-- ---------build tableiwith the article (subject and story) ----->
  <table border="0">
  <xsl:apply-templates/>
  </table>
  <!-- footer -->
  <br/>
  <center>
  <table border="0">
    <tr align="center"><td><b>IBM International Technical Support
Organization</b></td></tr>
    <tr align="center"><td><a
href="www.redbooks.ibm.com">http://www.redbooks.ibm.com</a></td></tr>
  </table>
  </center>
  </body>
  </html>
</xsl:template>
<!-- ----------- match on the subject -------- -->
<xsl:template match="subject">
  <tr>
    <td>
      <b>
       <xsl:value-of select="."/>
      </b>
    </td>
   </tr>
</xsl:template>
<!-- ---------- match on the story ----- -->
<xsl:template match="story">
  <tr>
    <td>
      <xsl:value-of select="."/>
    </td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Shown next is the browser view of a selected article (news subject) followed by the associated HTML document.


IBM WebSphere Everyplace News

"IBM Extends Lead in Emerging $14 Billion Portal Market"

SAN JOSE, Calif., June 21, 2000 -- IBM today extended its lead in redefining the management market and introduced the IBM Enterprise Information Portal (EIP) Version 7, the most advanced portal solution in the industry. Now customers can access and organize relevant business information whatever the format, wherever it resides and slash development time nearly in half.

IBM International Technical Support Organization

http://www.redbooks.ibm.com

Figure 6-8 Browser view of a Article (simple conversion)

Example 6-9 HTML for News Article (simple conversion)

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>IBM WebSphere Everyplace News - By Headline</title>
</head><body bgcolor="#FFFFFF">
<center>
<h3>IBM WebSphere Everyplace News</h3>
</center>
<table border="0">
 <tr>
  <td>
    <b>"IBM Extends Lead in Emerging $14 Billion Portal Market"</b>
  </td>
</tr>
<tr>
  <td>
    SAN JOSE, Calif., June 21, 2000 -- IBM today extended its lead in
redefining the management market and introduced the IBM Enterprise Information
Portal (EIP) Version 7, the most advanced portal solution in the industry. Now
customers can access and organize relevant business information whatever the
format, wherever it resides and slash development time nearly in half.
  </td>
```

```
</tr>
</table>
<br/>
<center>
<table border="0">
<tr align="center">
  <td>
    <b>IBM International Technical Support Organization</b>
  </td>
</tr>
<tr align="center">
  <td>
    <a
href="http://9.24.105.152:80/httppvc_clnss9.24.104.13/WebSphereSamples/servlet/
www.redbooks.ibm.com">http://www.redbooks.ibm.com</a>
  </td>
</tr>
</table>
</center>
</body>
</html>
```

We used the same approach to create stylesheets to convert the XML documents to WML:

- ► A stylesheet to convert the news topic to WML.
- ► A stylesheet to convert a news article to WML.

Let us look at the XML to WML stylesheet, starting with the news topic stylesheet.

Example 6-10   XML to WML news topic stylesheet

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="newsfeed">
  <wml>
    <card id="card1">
      <p>
        <xsl:for-each select="news">
            <strong>
<xsl:text disable-output-escaping="yes">&lt;a href="#</xsl:text>
<xsl:value-of select="@category"/>
             <xsl:text disable-output-escaping="yes">"&gt;</xsl:text>
             <xsl:value-of select="@category"/>
<xsl:text disable-output-escaping="yes">&lt;/a&gt;</xsl:text>
            </strong><br/>
        </xsl:for-each>
```

```
            <do type="prev" label="Back"><prev/></do>
        </p>
      </card>
       <xsl:apply-templates select="news"/>
    </wml>
</xsl:template>
<!---- create cards for each category with its topics ---->
<xsl:template match="news">
 <xsl:text disable-output-escaping="yes">&lt;card id="</xsl:text>
  <xsl:value-of  select="@category"/>
   <xsl:text disable-output-escaping="yes">"&gt;</xsl:text>
     <xsl:for-each select="item">
      <p>
<xsl:text disable-output-escaping="yes">&lt;a
href="ServeXML_news?xml_file=</xsl:text>
<xsl:value-of select="@reference"/>
        <xsl:text disable-output-escaping="yes">.xml"&gt;</xsl:text>
         <xsl:value-of select="title"/>
        <xsl:text disable-output-escaping="yes">&lt;/a&gt;</xsl:text>
      </p>
     </xsl:for-each>
    <do type="prev" label="Back"><prev/></do>
   <xsl:text disable-output-escaping="yes">&lt;/card&gt;</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

The resulting WML displayed with the WAP simulator is shown in Figure 6-9.

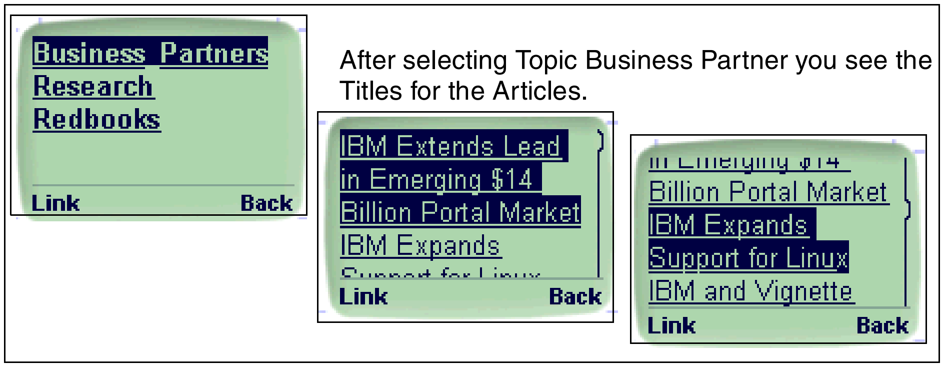


Figure 6-9   WAP display of News Topics

A portion (card1 and one category card with its titles) of the WML generated by the stylesheet is shown below.

Example 6-11   WML for the News feed

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="card1">
    <p>
      <strong><a
href="http://9.24.105.152:80/httppvc_clnssifragp-2147I43/9.24.104.13.webspheres
amples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed
_new.xml#Business_Partners">Business_Partners</a></strong>
      <br/>
      <strong><a
href="http://9.24.105.152:80/httppvc_clnssifragp-2147I43/9.24.104.13.webspheres
amples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed
_new.xml#Research">Research</a></strong>
      <br/>
      <strong><a
href="http://9.24.105.152:80/httppvc_clnssifragp-2147I43/9.24.104.13.webspheres
amples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed
_new.xml#Redbooks">Redbooks</a></strong>
      <br/>
      <do label="Back" type="prev"><prev/></do>
    </p>
  </card>

  <card id="Business_Partners">
    <p>
      <a
href="http://9.24.105.152:80/httppvc_clnss9.24.104.13/WebSphereSamples/servlet/
ServeXML_newspvc_qxml_file=PortalMarket_story.xml">IBM Extends Lead in Emerging
$$14 Billion Portal Market</a>
    </p>
    <p>
      <a
href="http://9.24.105.152:80/httppvc_clnss9.24.104.13/WebSphereSamples/servlet/
ServeXML_newspvc_qxml_file=SupportLinux_story.xml">IBM Expands Support for
Linux</a>
    </p>
    <p>
      <a
href="http://9.24.105.152:80/httppvc_clnss9.24.104.13/WebSphereSamples/servlet/
ServeXML_newspvc_qxml_file=Vignette_story.xml">IBM and Vignette Develop Global
Strategic E-business Alliance</a>
    </p>
    <do label="Back" type="prev">
```

```
      <prev/>
    </do>
  </card>
</wml>
```

Next, let us look at the stylesheet to convert a news article to WML.

Example 6-12 XML to WML news article stylesheet

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<!---------- template for each newsitem -->
<xsl:template match="newsitem">
        <wml>
          <card>
             <p>
                <xsl:apply-templates/>
                <do type="prev" label="Back"><prev/></do>
              </p>
          </card>
        </wml>
</xsl:template>
<!---------- template for each subject -->
<xsl:template match="subject">
      <strong>
        <b>
         <xsl:value-of select="."/>
        </b>
      </strong>
      <br/>
</xsl:template>
<!--------- template for story -->
<xsl:template match="story">
      <xsl:value-of select="."/>
</xsl:template>
</xsl:stylesheet>
```

Once a selection has been made, the article is displayed .

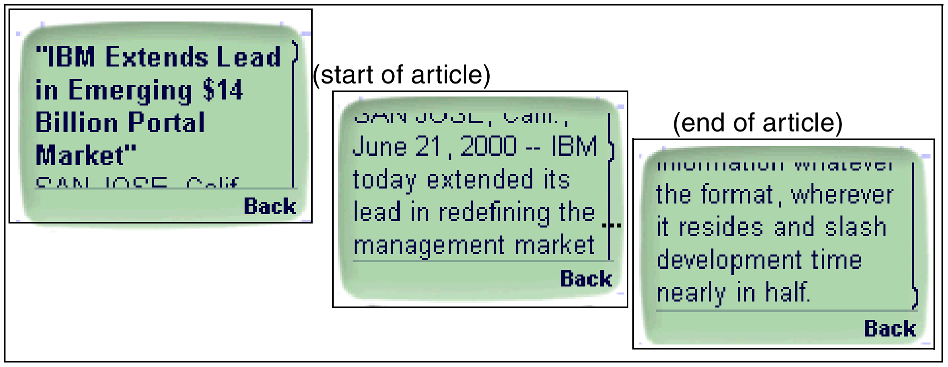

"IBM Extends Lead in Emerging $14 Billion Portal Market"
Back
(start of article)
June 21, 2000 -- IBM today extended its lead in redefining the management market
Back
(end of article)
the format, wherever it resides and slash development time nearly in half.
Back

Figure 6-10   WAP display of the selected News Subject

The WML for the news article is shown below.

Example 6-13   WML for a news article

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="cmgcfp0">
    <p>
      <strong><b>&quot;IBM Extends Lead in Emerging $$14 Billion Portal
Market&quot;</b></strong>
      <br/>
      SAN JOSE, Calif., June 21, 2000 -- IBM today extended its lead in
redefining the management market and introduced the IBM Enterprise Information
Portal (EIP) Version 7, the most advanced portal solution in the industry. Now
customers can access and organize relevant business information whatever the
format, wherever it resides and slash development time nearly in half.
      <do label="Back" type="prev"><prev/></do>
    </p>
  </card>
</wml>
```

Phase 2 - using parameters

Now we will take the simple HTML and WML stylesheets and add a parameter. We thought it would be interesting to converge the HTML and WML stylesheets for the news article into a single stylesheet and use a parameter (deviceType) to select the appropriate output type (HTML or WML).

First, we had to map the two stylesheets to converge them. Various elements mapped directly; for example, the <WML> tag mapped to the <HTML> tag. The HTML document had various additional content items which do not map to anything in the WML stylesheet and the reverse is true also. We created a generalized template containing the mapping instructions. Within this generalized stylesheet, we used two XSL mechanisms:

- The <xsl:variable> element - this allows you to create named constants. The element defines named strings. These named strings can be used by other stylesheets to fill in content dynamically.
- The <xsl:template> element with a name attribute - the name attribute's value allows the template to be called into action when needed. The body of the template contains the content to be added to the calling stylesheet.

Let us look at an example <xsl:variable> element.

Example 6-14 <xsl:variable> element

```
<xsl:variable name="startDoc">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'">wml</xsl:when>
      <xsl:otherwise>html</xsl:otherwise>
   </xsl:choose>
</xsl:variable>
```

The `<xsl:variable name= “startDoc”>` element (where the name identifies this element) consists of `<xsl:choose>` which, in this example, outputs a `wml` tag if `deviceType=‘WML Device’` is true; otherwise, an `HTML` tag is output.

An example <xsl:template> is shown below:

Example 6-15 <xsl:template> element

```
<xsl:template name="header">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'"></xsl:when>
      <xsl:otherwise>
          <center>
          <h3>IBM WebSphere Everyplace News</h3>
          </center>
      </xsl:otherwise>
   </xsl:choose>
</xsl:template>
```

The `<xsl:template name=”header”>` element (where the name identifies this element) consists of `<xsl:choose>` which, in this example, outputs nothing if `deviceType=‘WML Device’` is true; otherwise, the HTML heading is output.

Now let us look at the stylesheet which contains the generalizations, as shown below.

Example 6-16   Generalized stylesheet

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:variable name="table">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'"></xsl:when>
      <xsl:otherwise>table</xsl:otherwise>
   </xsl:choose>
</xsl:variable>

<xsl:variable name="tRow">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'">p</xsl:when>
      <xsl:otherwise>tr</xsl:otherwise>
   </xsl:choose>
</xsl:variable>

<xsl:variable name="tDef">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'"></xsl:when>
      <xsl:otherwise>td</xsl:otherwise>
   </xsl:choose>
</xsl:variable>

<xsl:variable name="break">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'">br</xsl:when>
   </xsl:choose>
</xsl:variable>

<xsl:variable name="startDoc">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'">wml</xsl:when>
      <xsl:otherwise>html</xsl:otherwise>
   </xsl:choose>
</xsl:variable>

<xsl:variable name="body">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'">card</xsl:when>
      <xsl:otherwise>body</xsl:otherwise>
   </xsl:choose>
</xsl:variable>

<xsl:template name="title">
```

```
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'"></xsl:when>
      <xsl:otherwise>
          <head>
          <title>IBM WebSphere Everyplace News - By Headline</title>
          </head>
      </xsl:otherwise>
   </xsl:choose>
</xsl:template>

<xsl:template name="header">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'"></xsl:when>
      <xsl:otherwise>
          <center>
          <h3>IBM WebSphere Everyplace News</h3>
          </center>
      </xsl:otherwise>
   </xsl:choose>
</xsl:template>
<xsl:template name="footer">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'"></xsl:when>
      <xsl:otherwise>
       <br/>
       <center>
        <table border="0">
         <tr align="center"><td><b>IBM International Technical Support
Organization</b></td></tr>
         <tr align="center"><td><a
href="www.redbooks.ibm.com">http://www.redbooks.ibm.com</a></td></tr>
        </table>
       </center>
      </xsl:otherwise>
   </xsl:choose>
</xsl:template>

<xsl:template name="do">
   <xsl:choose>
      <xsl:when test="$deviceType='WML Device'">
        <do type="prev" label="BACK"><prev/></do>
     </xsl:when>

   </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

To invoke an `<xsl:variable> found` in the generalized stylesheet, we are using the `<xsl:element name=”{$startDoc}”>` element, where the name contains the name of the `<xsl:variable>` element. An `</xsl:element>` creates the associated end element. To invoke an `<xsl:template>` in the generalized stylesheet, we are using the `<xsl:call-template name=”header”/>` element, where the name contains the name of the template. The converged news article stylesheet is shown below. The parameter is defined in this stylesheet and shared with the generalized stylesheet. The parameter element is highlighted in the example.

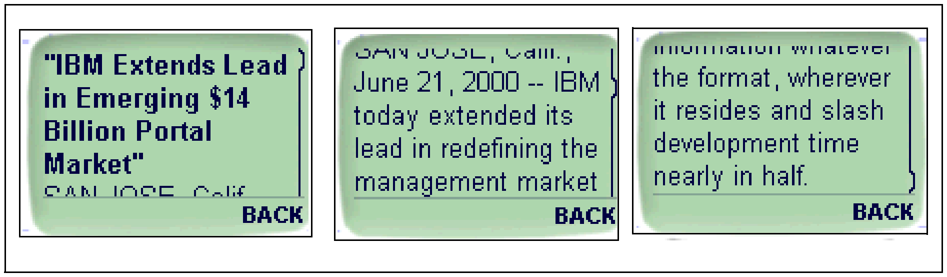
Example 6-17   Converged stylesheet

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:param name="deviceType" select="defaultValue" />
<xsl:include href="general.xsl"/>

<!-- template for each newsitem -->
<xsl:template match="newsitem">

 <xsl:element name="{$startDoc}">
      <xsl:call-template name="title"/>
  <xsl:element name="{$body}">
      <xsl:call-template name="header"/>

    <xsl:element name="{$table}">
         <xsl:apply-templates/>
    </xsl:element>
         <xsl:call-template name="do"/>
   </xsl:element>
  </xsl:element>
</xsl:template>
<!-- template for each subject -->
<xsl:template match="subject">
  <xsl:element name="{$tRow}">
    <xsl:element name="{$tDef}">
     <B>
         <xsl:value-of select="."/>
     </B>
    </xsl:element>
   </xsl:element>
</xsl:template>
<!-- template for story -->
```

```
<xsl:template match="story">
   <xsl:element name="{$tRow}">
    <xsl:element name="{$tDef}">
      <xsl:value-of select="."/>
    </xsl:element>
   </xsl:element>
</xsl:template>
</xsl:stylesheet>
```

Note: The generalized stylesheet does not need to be registered. Either explicitly specify the directory path in the include, or place it in the same directory as the using stylesheet.

The news article displayed on the browser is the same as that shown in Figure 6-7. The results on the emulator are shown below.


"IBM Extends Lead in Emerging $14 Billion Portal Market"

BACK

June 21, 2000 -- IBM today extended its lead in redefining the management market

BACK

the format, wherever it resides and slash development time nearly in half.

BACK

Figure 6-11   Emulator results

For more details on creating generalized stylesheets, look at the article *Reusable stylesheet components* found at:
`www.ibm.com/software/webserver/transcoding/library.html`.

6.1.3 Implementing internationalization

Stylesheets can be used for more than just transforming XML documents from one XML markup language to another. They can be used to generate content specific to a locale. This capability allows a single XML document's content to be internationalized. You can translate an XML document content to varying degrees:

- Target-specific content for translation, for example, content in table headers or forms.
- Extensive content translation, that is, translating all or most of the content within the document.

The translations are contained in XML files which consist of the terms (as identifiers) and their target language counter part. These XML files are used with stylesheets to perform the conversion. WebSphere Transcoding Publisher allows you to implement internationalization by either using a separate file for each language or by using a single file for all the languages you plan to support.

WebSphere Transcoding Publisher provides several sample translation files, which are located in the */IBMTrans/etc/stylesheet directory path. This directory contains translation files for specific languages, called translations_xx.xml, where xx identifies the language (for instance en = english, fr=french, etc.), and a file containing multiple languages named translations.xml. Also included in this same directory (*/IBMTrans/etc/stylesheet) are two stylesheets which retrieve the new value from the XML translation files:

- The translate.xsl file is used to access translation content when each language is in a separate file.
- The translate_sf.xsl file is used to access translation content when all languages are in a single file.

To reference the translate.xsl file, you would include this XSL statement:

Example 6-18 Including the translate.xsl file

```
<xsl:include href="translate.xsl"/>
```

To invoke the translation of a term (word), you would include the following template call:

Example 6-19 Invoking translate.xsl from your stylesheet

```
<xsl:call-template name=”translate”>
<xsl:with-param name=”word” select=”’button’“
</xsl:call-template>
```

The above XSL statement requests the return of the translation value for the <word> element with an identifier of `button`. The translate stylesheet (referenced in the above instruction) does the following:

- Determines the value of the language parameter (lang).
- Builds the file name of the XML file containing the translation for the specific language, which in this case would be the translations_en.xml (english).
- Retrieves and returns the translation for the word.

Let us look at the translations_en.xml translation file.

Example 6-20 Sample translations_en.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<translations>
  <word id="address">address</word>
  <word id="button">button</word>
  <word id="check">check</word>
  <word id="enter">enter</word>
  <word id="find">find</word>
  <word id="language">language</word>
  <word id="location">location</word>
  <word id="name">name</word>
  <word id="press">press</word>
  <word id="search">search</word>
  <word id="submit">submit</word>
  <word id="telephone">telephone</word>
</translations>
```

For the above example, your stylesheet would output the string `button` for the English language. If the target language was French, using the translate_fr.xml file with an entry `<word id=button”>bouton</word>`, the output string would be `bouton`.

Note: You are free to add your own <word> elements to the XML translation files that come with WebSphere Transcoding Publisher. Be aware that future releases of WebSphere Transcoding Publisher might add elements to these files. You must be prepared to migrate your additions to the newer versions of the files. You can also create your own separate XML files using a similar scheme to be used in a similar fashion.

Extension elements

An extension element is an instruction that is not defined by the XSLT Transformation standard, but looks and behaves like an XSLT instruction. Extension elements enable you to add new functions to the XSLT language.

WebSphere Transcoding Publisher includes two extension elements designed to simplify the XML coding required to access a translated string from a stylesheet:

- The <nls:trans> element is used when the translated strings for each language are in a separate file.
- The <nls:trans_sf> element is used when the translated strings for all languages are in a single file.

For more details on the WebSphere Transcoding Publisher extension elements, look at the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5,* SG24-6233.

Sample scenario for internationalization

We are extending the WebSphere Everyplace News application to translate from English into French. The WebSphere Everyplace News application is explained above. In this scenario, we will be adding translation to the news topics, which is shown below in English. Note that we added a title (The News) to the stylesheet to make it more interesting.

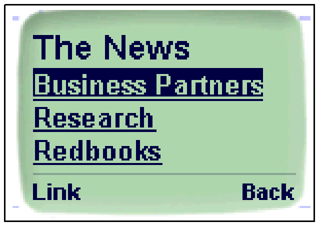


Figure 6-12 The News categories in English

The files used to perform internationalization in our sample are:

- ► The XML document (MainNewsfeed_new.xml) which was shown in Example 6-4.
- ► The registered stylesheet (NewsTopicWAPLang.xsl). We modified the stylesheet from its original form to add more words to translate.
- ► The translation stylesheet (translate.xsl) which is included in our stylesheet (NewsTopicWAPLang.xsl).
- ► Our translation files: ITSOtranslation_en.xml (English) and ITSOtranslation_fr.xml (French).
- ► The device and network preference profiles.
- ► The properties (.prop) file created by the Administration Console when we registered the stylesheet.

The steps to perform the translation are as follows:

1. WebSphere Transcoding Publisher receives the XML document to be transcoded.
2. WebSphere Transcoding Publisher reads the target device preference profiles.

3. WebSphere Transcoding Publisher determines the registered stylesheet to use for this situation. It takes into account the selection criteria, parameters and device characteristics. The NewTopicWAPLang.xsl stylesheet is selected.
4. The <xsl:include> element identifies the translate.xsl stylesheet used for translation.
5. The stylesheet registration specifies the translation files to use when this stylesheet is invoked. They are ITSOtranslation_en.xml (English) and ITSOtranslation_fr.xml (French). The translation file for the target language is selected and the translations are performed.
6. The XML document is transformed into the target mark up language, in this case WML.

Created stylesheet

The stylesheet created for translation is shown below. The translation-related instructions are highlighted.

Example 6-21 Translation stylesheet

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:lxslt="http://xml.apache.org/xslt"
  xmlns:nls="com.ibm.transform.textengine.mutator.stylesheet.ext.NLS"
  extension-element-prefixes="nls"
  version="1.0">

<xsl:include href="translate.xsl"/>
<xsl:output doctype-system="http://www.wapforum.org/DTD/wml_1.1.xml"/>

<xsl:template match="newsfeed">
  <wml>
    <card id="card1">
     <p><big><b>
      <nls:trans id="WENewHead"/>
     </b></big></p>
      <p>
        <xsl:for-each select="news">
            <strong>
<xsl:text disable-output-escaping="yes">&lt;a href="#</xsl:text>
<xsl:value-of select="@category"/>
             <xsl:text disable-output-escaping="yes">"&gt;</xsl:text>
             <xsl:variable name="varname">
               <xsl:value-of select="@category"/>
             </xsl:variable>
             <xsl:call-template name="translate">
               <xsl:with-param name="word" select="$varname"/>
             </xsl:call-template>
```

```
<xsl:text disable-output-escaping="yes">&lt;/a&gt;</xsl:text>
           </strong><br/>
        </xsl:for-each>

         <xsl:text disable-output-escaping="yes"> &lt;do type="prev"
label="</xsl:text>
          <nls:trans id="back"/>
        <xsl:text
disable-output-escaping="yes">"&gt;&lt;prev/&gt;&lt;/do&gt;</xsl:text>
      </p>
    </card>
     <xsl:apply-templates select="news"/>
  </wml>
</xsl:template>

<xsl:template match="news">
 <xsl:text disable-output-escaping="yes">&lt;card id="</xsl:text>
  <xsl:value-of  select="@category"/>
   <xsl:text disable-output-escaping="yes">"&gt;</xsl:text>
     <xsl:for-each select="item">
      <p>
        <xsl:text disable-output-escaping="yes">&lt;a
href="ServeXML_news?xml_file=</xsl:text>
   <xsl:value-of select="@reference"/>
        <xsl:text disable-output-escaping="yes">.xml"&gt;</xsl:text>
         <xsl:value-of select="title"/>
        <xsl:text disable-output-escaping="yes">&lt;/a&gt;</xsl:text>
      </p>
     </xsl:for-each>

    <xsl:text disable-output-escaping="yes"> &lt;do type="prev"
label="</xsl:text>
          <nls:trans id="back"/>
    <xsl:text
disable-output-escaping="yes">"&gt;&lt;prev/&gt;&lt;/do&gt;</xsl:text>

   <xsl:text disable-output-escaping="yes">&lt;/card&gt;</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

We used two different ways to handle the translations:

- For dynamic content coming from the XML document, as in the case of the (@category) attribute, the attribute value is captured as a variable and the variable in input to the <xsl:call-template> element (see Example 6-22).

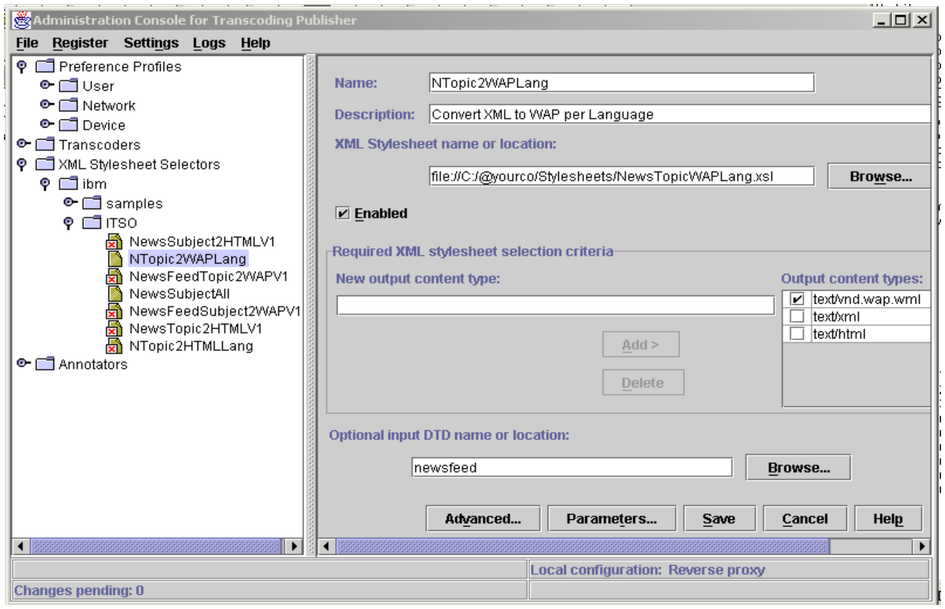
Example 6-22 Using the variable for translation

```
<xsl:variable name="varname">
          <xsl:value-of select="@category"/>
</xsl:variable>

<xsl:call-template name="translate">
          <xsl:with-param name="word" select="$varname"/>
</xsl:call-template>
```

- For static stylesheet content, we used the <nls:trans id="xxx"/> statement, causing the specified word id to be translated.

Stylesheet registration

This stylesheet is registered like all other stylesheets, except that we must specify the translation files to use, since they are different from the default translation files. The first step to registration is filling in the general registration information, as shown below.



Name: NTopic2WAPLang

Description: Convert XML to WAP per Language

XML Stylesheet name or location: file://C:/@yourco/Stylesheets/NewsTopicWAPLang.xsl

[✓] Enabled

Required XML stylesheet selection criteria

New output content type:

Output content types:
[✓] text/vnd.wap.wml
[ ] text/xml
[ ] text/html

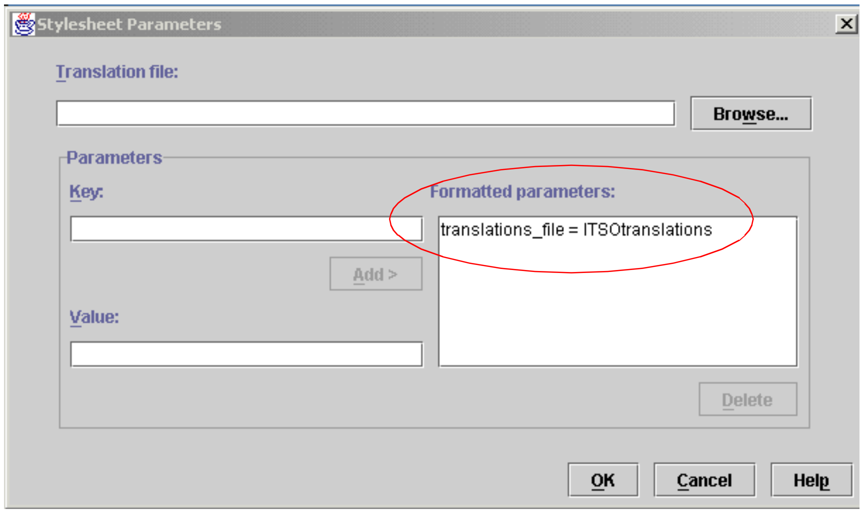
Optional input DTD name or location: newsfeed

Local configuration: Reverse proxy

Changes pending: 0

Figure 6-13 Stylesheet registration for NewsTopicWAPLang.xsl

To specify our translation files, click the **Parameters** button and set the parameter, as shown below.



translations_file = ITSOtranslations

Figure 6-14 Setting the translation file parameter

Note: Make sure that the stylesheet is enabled and that you refresh the server. You do not need to register the included stylesheets.

Translation files

The two translation files we are using are in our stylesheet directory. This is where WebSphere Transcoding Publisher looks for the translation files. If it does not find them there, it looks in its own directory, */IBMTrans/etc/stylesheets. The translation files are shown below. The english translation file is shown first.

Example 6-23 English translation file ITSOtranslation_en.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<translations>
  <word id="Business_Partners">Business Partners</word>
  <word id="Research">Research</word>
  <word id="Redbooks">Redbooks</word>
  <word id="WENewTitle">The News</word>
  <word id="WENewHead">The News</word>
  <word id="ITSO">IBM International Technical Support Organization</word>
</translations>
```

The French translation file is shown in Example 6-24.

Example 6-24 French translation file ITSOtanslation_fr.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<translations>
  <word id="Business_Partners">Partenaires Commerciaux</word>
  <word id="Research">Recherche</word>
  <word id="Redbooks">Documents ('Redbooks')</word>
  <word id="WENewHead">Manchettes</word>
  <word id="ITSO">Organisme de Support Technique International IBM</word>
  <word id="back">Precedent</word>
</translations>
```

The stylesheet can be tested in different ways:

- Run WebSphere Transcoding Publisher and request the application from a WAP phone emulator or a WAP phone.
- Run the Request Viewer (this also allows you to monitor the execution) from a WAP phone emulator.
- Use the Transform tool to perform the transformation. This tool is part of the WebSphere Transcoding Publisher development tools. For details, see the redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5,* SG24-6233.

For example, to see the various languages on the Nokia Toolkit WAP phone simulator, perform the following steps:

1. Select the **Toolkit** menu option; from the pull-down menu, select **Device Settings**.
2. In the BluePrint Device Settings window, select the **Header** tab.
3. Using the Language Preference pull-down menu, select a language.
4. Click the **OK** button.

The resulting transcoded WML is included below. First, let us look at the English version and the results on the emulator.

Example 6-25 WML for translation (English)

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="card1">
    <p>
      <big><b>The News</b></big>
    </p>
    <p>
      <strong><a
href="http://9.24.105.152:80/httppvc_clnssifragp-2114I66/9.24.104.13.websphereS
amples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed
_new.xml#Business_Partners">Business Partners</a></strong>
      <br/>
      <strong><a
href="http://9.24.105.152:80/httppvc_clnssifragp-2114I66/9.24.104.13.websphereS
amples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed
_new.xml#Research">Research</a></strong>
      <br/>
      <strong><a
href="http://9.24.105.152:80/httppvc_clnssifragp-2114I66/9.24.104.13.websphereS
amples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed
_new.xml#Redbooks">Redbooks</a></strong>
      <br/>
      <do label="" type="prev"><prev/></do>
    </p>
  </card>
<!--- Article Title cards --->
</wml>
```

The WML displayed on the WAP emulator is illustrated in Figure 6-15.

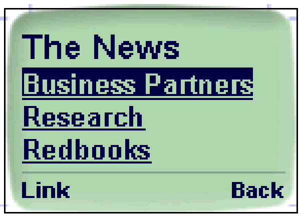

The News
Business Partners
Research
Redbooks

Figure 6-15 WML on the emulator (English)

The WML generated from the translation to French (with the translations highlighted) is shown in Example 6-26.

Example 6-26 WML for translation (French)

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card id="card1">
    <p>
      <big><b>Manchettes</b></big>
    </p>
    <p>
      <strong><a
href="http://9.24.105.152:80/httppvc_clnssifragp-1925I68/9.24.104.13.webspheres
amples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed
_new.xml#Business_Partners">Partenaires Commerciaux</a></strong>
      <br/>
      <strong><a
href="http://9.24.105.152:80/httppvc_clnssifrag-6934I68/9.24.104.13.webspheresa
mples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed_
new.xml#Research">Recherche</a></strong>
      <br/>
      <strong><a
href="http://9.24.105.152:80/httppvc_clnssifragp-1925I68/9.24.104.13.webspheres
amples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed
_new.xml#Redbooks">Documents (&apos;Redbooks&apos;)</a></strong>
      <br/>
      <do label="Precedent" type="prev"><prev/></do>
    </p>
  </card>
<!--- Article Title cards --->
</wml>
```

The WML displayed on the WAP emulator is shown in Figure 6-16.

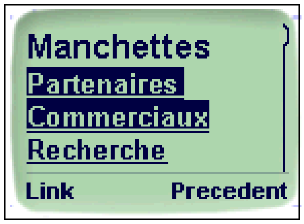


Figure 6-16 WAP Translation (French)

6.1.4 XSL Stylesheet Editor

WebSphere Transcoding Publisher 4.0 introduces the XSL Stylesheet Editor. The XSL Stylesheet Editor is a graphical tool that provides a WYSIWYG environment intended for the novice stylesheet creator. However, the editor provides escapes for the advanced user, so that the stylesheet template rules can be extended as needed.

The XSL Stylesheet Editor is included in the WebSphere Transcoding Publisher 4.0 Win32 installation process. There is no separate installation process and no option to not install it. It is installed as part of the Toolkit under the Toolkit directory in folder xslide. It can be uninstalled without uninstalling WebSphere Transcoding Publisher, but requires Java on the system for the uninstallation process to be implemented.

The editor is started using **Start -> Programs -> IBM Transcoding Publisher -> Toolkit -> Stylesheet Editor** or by issuing the `runxslide` command from a DOS prompt pointing at the directory path. The editor is a graphical tool and its main frame is composed of these major areas:

1. Menu bar - provides access to the application options, for example Files, Edits, Views, Annotation and Help. Menu options are disabled when the action is not available.
2. Toolbar - provides access to the common application actions and specific stylesheet tasks. When content is selected in the Output display, the Condition pull-down menu displays possible stylesheet template rules from which to choose.
3. Stylesheet window - provides views of the stylesheet in either Tree, Rule or Text format.
4. Output window - provides views of the input XML document transformed into XHTML with the stylesheet template rules applied. The XHTML can be viewed in Tree, Design or Text format.
5. XML sample window - provide views of the input XML document. The XML document may be viewed in either Tree or Text format.
6. Projects - provide a tree view of the projects defined in the editor. All projects are shown, and there is no concept of open or closed projects.
7. Message window - (not displayed unless the user turns it on) displays messages produced by the editor, Xerces (parser) or Xalan (processor). Help is available on some messages by clicking the **Help** icon when a message is highlighted.

The editor’s main window is shown below using the XML document NewsEdit.xml displayed and the stylesheet NewsSS.xsl.

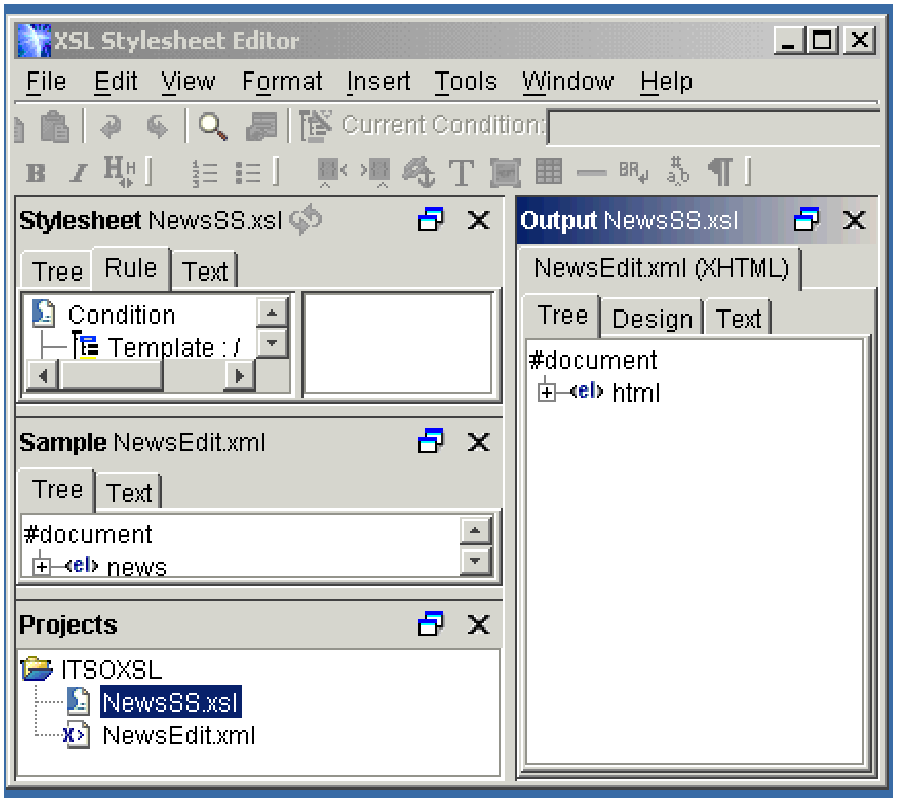


Figure 6-17 XSL StyleSheet Editor - main window

Creating a project

The editor uses the project to organize work using XML document samples and XSL stylesheets. A project contains one or more XML samples and one or more XSL stylesheets. Creating a project is the first step in working with the editor. When you start the editor, it will ask if you want to create a new project or work with an existing project. If you want to create a new project, just follow the wizard directions to get started. Another way to create a project (once the editor is running) is to **Select File -> New Project** or click the **New Project** toolbar entry; the wizard (the same wizard as used at start-up) steps you through the process by:

- Asking for the project name
- Asking you to choose the XML sample file(s) to add to the project

- Asking you to either:
  - Choose to create a new XSL stylesheet
  - Choose an existing XSL stylesheet

When you have the editor running, use the following actions to create new XML samples or to add new stylesheets:

- To add XML samples to the project, either select **File -> Add Samples** or click **Add Samples** from the toolbar. A window appears to help you locate the XML document.
- To add XSL stylesheets to the project, either select **File -> New Stylesheet** or click **New Stylesheet** from the toolbar. Enter a name and a location for the stylesheet.

XML sample window

The sample XML documents are used as the basis for creating the template rules within the stylesheet. The template rules determine the way the XML document content is structured and represented in the output document. You should select XML sample documents representative of the XML data that the associated stylesheets are intended to handle. The XML sample window provides two views:

- A tree view - a hierarchical tree view of the XML document. Both selection and highlighting are available in the view.
- A text view - a simple text view of the document. This is a read-only view and selection and highlighting are not supported in this view.

Stylesheet window

The stylesheet window displays the XSL stylesheet. The stylesheet is used to process the sample XML document, producing an output document. The stylesheet window has three views:

- A rules view - provides a Windows Explorer-like view of the stylesheet, based on rules (xsl template rules) found in the stylesheet. The left-hand pane contains the match conditions (that is, what elements in the XML data match these rules). The right-hand pane contains output generators (that is, what happens when the rule is matched). Both selection and highlighting are available in this view.
- A tree view - provides a hierarchical view of the XSL stylesheet. Both selection and highlighting are available in this view.
- A text view - a simple text view of the stylesheet. This is a read-only view, but editing in this view disables all other views until you rebuild the stylesheet. Neither selection nor highlighting are available in this view.

Output window

The output window displays the results of applying the XSL stylesheet to the XML sample. The window provides three views:

- A design view - provides a WYSIWYG environment for modifying the output document. This is a browser-like display of the output document. Changes made to the output document (in this view) are actually changes being applied to the stylesheet. Both selection and highlighting are available.
- A tree view - provides a hierarchical view of the output document. Both selection and highlighting are available in this view, which is very useful when the stylesheet is producing markup other than XHTML output.
- A text view - a simple view of the output document. This is a read-only view within which you cannot edit the output. Neither selection nor highlighting are available in this view.

You will only see a design view if the stylesheets produce XHTML. The XHTML toolbar is only available with XHTML output. The XHTML stylesheets must be part of the XHTML namespace. All other stylesheets are treated as generic XSLT stylesheets with no output design view and no XHTML toolbar.

Creating a stylesheet rule

Within a project, select the XML sample document for which you want to create a stylesheet. The content of the XML document is displayed in the output window. Rule creation consists of three main steps:

1. Select the XML content that is the target of the template rule.
2. Determine the match condition for the template rule.
3. Define the output of the rule, that is, how the selected content is affected by the execution of the rule.

There are two ways to create the template rule match condition: either use the Current Condition field or the XPath Builder. Let us explore each method.

The Current Condition field is found on the toolbar. When a selection is made in any of these views:

- The stylesheet window rule or tree view,
- The XML sample window tree view,
- The Output window design or tree view,

then the Current Condition field shows a match condition for the selection (the XML document content selected). The Current Condition field provides a drop-dow menu which contains the first 20 potential refinements of the current match condition. You can select the condition that meets your needs for this match condition.

The XPath Builder provides a more advanced condition builder. The current condition is shown by default, but it can be edited. XPath Builder provides categories of the possible match condition refinements (that is, match conditions based on attributes, parent, etc.).

Sample scenario using XSL StyleSheet Editor

Prior to starting the XSL editor, we created a new sample XML document, named NewsEdit.xml. This new XML document is shown in the example below.

Example 6-27 TITSO NewsEdit.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<news>
 <article>
   <createDate>06012001</createDate>
   <author>
     <authorName>LR Robins</authorName>
     <authorNumber>4709-331</authorNumber>
   </author>
   <details>
      <category>Technical</category>
      <title>IBM Wins Big</title>
      <subject>IBM wins with its new Super computer</subject>
   </details>
 </article>
</news>
```

When we started the editor, we were asked if we wanted to create a new project, which we did. Our new project was named ITSOXSL. We added our new XML document (NewsEdit.xml) to the project. We named the new XSL stylesheet XSLEditor.xsl. At this point, we had a skeleton XSL document for our XML document. The XML document sample was displayed in the output window.

Next, we created three template rules:

- The authorName rule, which utilizes the authorName contents.
- The createDate rule, where the createDate contents are in bold face.
- The title element rule, which makes the contents a Heading Level 2.

We selected each element and used the Current Condition to select the element as the template rule match value. This caused the creation of a skeleton template rule. From the tool bar, we selected the appropriate HTML tag, which became part of the rule body.

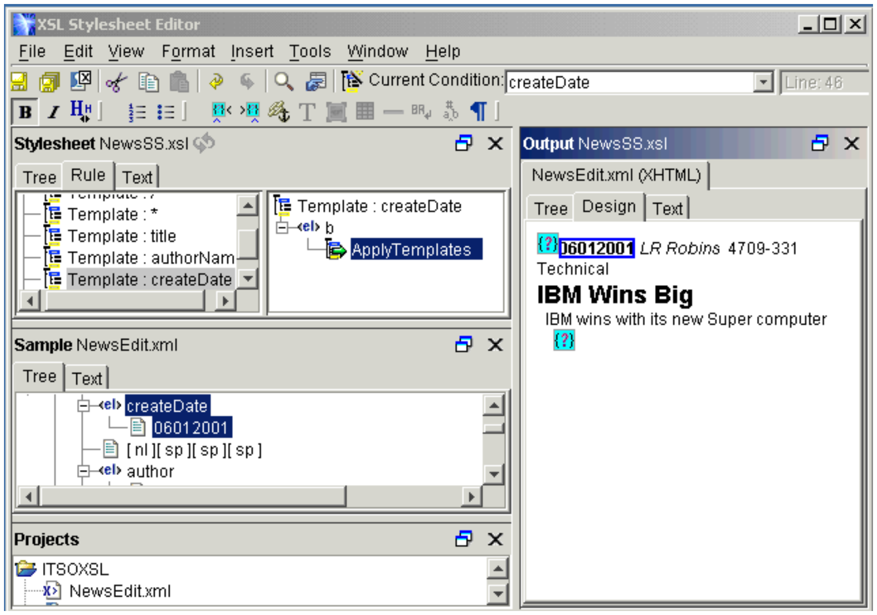


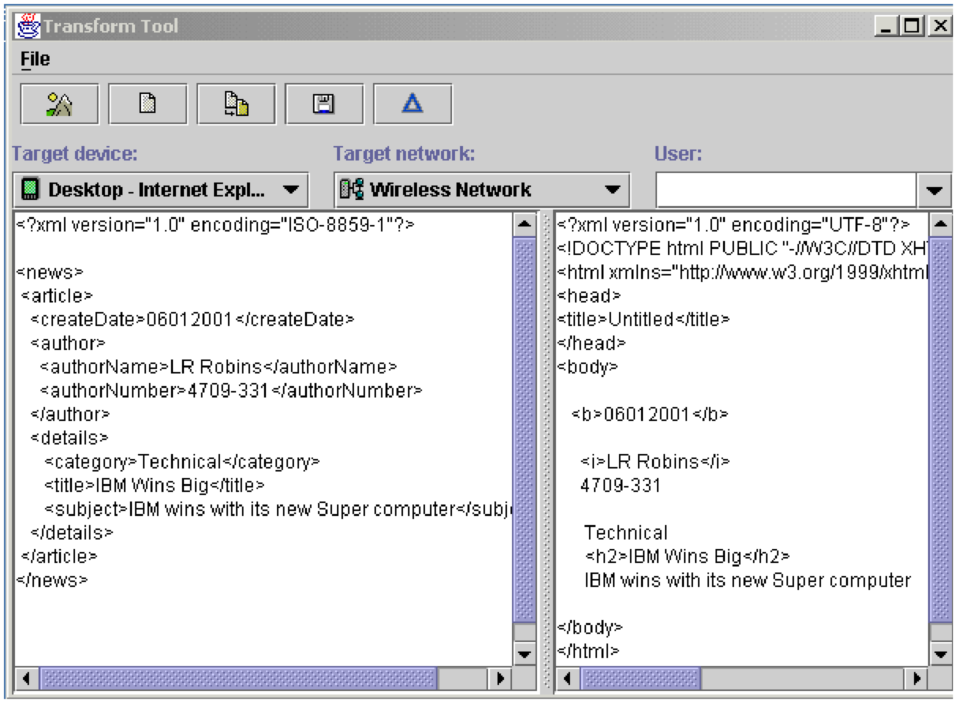
Figure 6-18 XSL Editor with template rule

The resulting XSL stylesheet rules created by the XSL editor are shown below.

Example 6-28 XSL template rules

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns="http://www.w3.org/1999/xhtml"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output
        doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
        doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
        indent="yes" method="xml"/>

  <xsl:template match="/">
    <html>
      <head>
         <title>Untitled</title>
      </head>
      <body>
      <span xsl-ide-placeholder="before"/>
      <xsl:apply-templates/>
      <span xsl-ide-placeholder="after"/>
      </body>
    </html>
  </xsl:template>

<xsl:template match="*">
      <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="title">
      <h2><xsl:apply-templates/></h2>
  </xsl:template>

  <xsl:template match="authorName">
      <i><xsl:apply-templates/></i>
  </xsl:template>

  <xsl:template match="createDate">
      <b><xsl:apply-templates/></b>
  </xsl:template>
</xsl:stylesheet>
```

We registered this stylesheet and used the associated XML document as input to the TransformTool. Shown below is the results of the transformation by the TransformTool.



```
<?xml version="1.0" encoding="ISO-8859-1"?>

<news>
 <article>
  <createDate>06012001</createDate>
  <author>
   <authorName>LR Robins</authorName>
   <authorNumber>4709-331</authorNumber>
  </author>
  <details>
   <category>Technical</category>
   <title>IBM Wins Big</title>
   <subject>IBM wins with its new Super computer</subj
  </details>
 </article>
</news>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XH
<html xmlns="http://www.w3.org/1999/xhtml
<head>
<title>Untitled</title>
</head>
<body>

  <b>06012001</b>

   <i>LR Robins</i>
   4709-331

    Technical
    <h2>IBM Wins Big</h2>
    IBM wins with its new Super computer

</body>
</html>
```

Figure 6-19 XSL Editor stylesheet results in the Transform Tool

XSL Editor limitations

This is the initial release of the XSL Stylesheet Editor. The editor has the following limitations and restrictions:

- It is only supported on Windows 2000 or Windows NT.
- It requires Java Version 1.3 JVM.
- The Output Design View only supports XHTML.
- Not all XHTML tags are supported.

7

WML fragmentation considerations

IBM WebSphere Transcoding Publisher (WTP) provides services to transform application content into a series of dynamically linked information (called *decks*), suitable for handling by client devices using markup languages such as WML, compact HTML (cHTML) and HDML phones. In this chapter, we describe the fragmentation function provided by WebSphere Transcoding Publisher and how this support must be integrated in a WebSphere Everyplace Server environment for proper application deployment.

Two sample scenarios are included to show how you will need to configure Web Traffic Express (WTE) and WebSphere Transcoding Publisher (running as a reverse proxy and as a forward proxy) when deploying WAP applications using Transcoding Publisher to generate WML application content.

7.1 Overview

Many phones have limited storage capacity (for example, 1440 bytes). However, many Web pages exceed these limits. Therefore, converting an HTML page to HDML, i-Mode (cHTML), or WML is very likely to result in a deck and/or page that exceeds the maximum storage capacity of a phone. The same problem may be encountered with native wireless content if the content generator is unaware of the specific limits of the phone being used. Exceeding the storage capacity of the phone means that the page cannot be viewed on that phone.

The IBM WebSphere Transcoding Publisher (WTP) fragmentation function makes it possible to view these over-large pages on the limited storage phones. Fragmentation solves this problem by splitting a single oversized deck and/or page into multiple smaller decks/pages, each one smaller than the maximum size limitation.

Fragmentation is performed by the transcoder (*Fragmentation Transcoder)* in the Administration Console. This transcoder is registered and enabled by default.

Note: IBM WebSphere Transcoding Publisher supports fragmentation for WML, i-Mode (cHTML), and HDML.

7.1.1 How does it work?

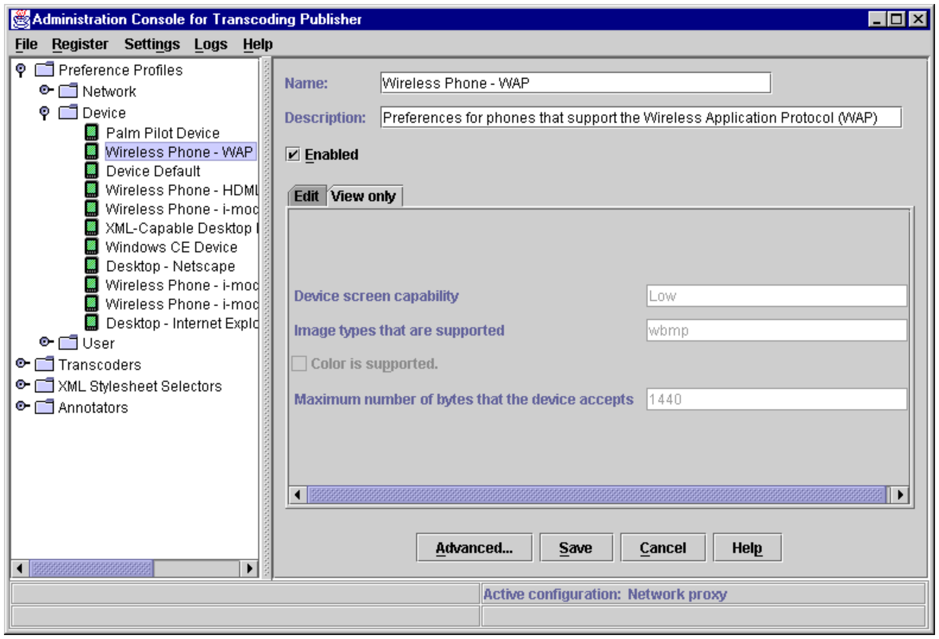
The Fragmentation Transcoder traverses the Document Object Model (DOM) tree representing the over-large card/page. The DOM is generated internally and automatically when HTML is transcoded to WML, i-Mode (cHTML) or HDML. Therefore, you do *not* need to enable the HTML Document Object Model (DOM) Generator transcoder.

If elements will fit on the new page, they are added to it and removed from the original. If elements do not fit, there are two ways to handle them:

- They can be fragmented: recursively traverse the element's subtree.
- They cannot be fragmented: finish fragmenting the new page; the original may still be too large.

The maximum size for a fragment is a property of the device. The fragmentor determines the size value from the information as follows:

- WML and HDML: there is a parameter which specifies the maximum size value in the device preference profile. The parameter is shown in Figure 7-1 in the `Maximum number of bytes that the device accepts` field. By default, you cannot change this value (it appears in the *View only* tab). However, if you create a new device profile by using the WebSphere Transcoding Publisher profile builder, you can set the value and make it configurable.



Name: Wireless Phone - WAP

Description: Preferences for phones that support the Wireless Application Protocol (WAP)

☑ Enabled

Device screen capability: Low

Image types that are supported: wbmp

☐ Color is supported.

Maximum number of bytes that the device accepts: 1440

Active configuration: Network proxy

Figure 7-1 The maximum size for a WML fragment

- i-Mode (cHTML): the maximum size value is 2048 bytes by default. There is no parameter as with WML and HDML. i-Mode phones can also have different cache sizes. The size is specified in the User-Agent field in the HTTP header. For example, User-Agent DoCoMo/1.0/N502/c8 specifies an 8 KB cache. In this case, the fragmentor adjusts fragmentation size based on this cache size. If the cache size is c8 (8 KB), the fragmentor sets the maximum size to 3000 bytes. If the cache size is c10 (10 KB), the fragmentor sets the maximum size to 4000 bytes.

In addition to splitting up the deck/page into smaller chunks, the fragmentor adds links to each of the generated pieces to allow navigation from one piece to the next and previous one. The Continue...link allows you to move to the next fragment and the Return link moves you to the previous one. The first fragment has no Return link and the last no Continue... link.

Figure 7-2 shows an example of WML fragmentation. A single oversized WML deck is fragmented into two smaller pieces. The Continue... and Return links are inserted into the fragments to allow for navigation between the fragments. Also, any intra-deck links in the original deck are fixed to point to the target in whatever deck/card in which they are placed.

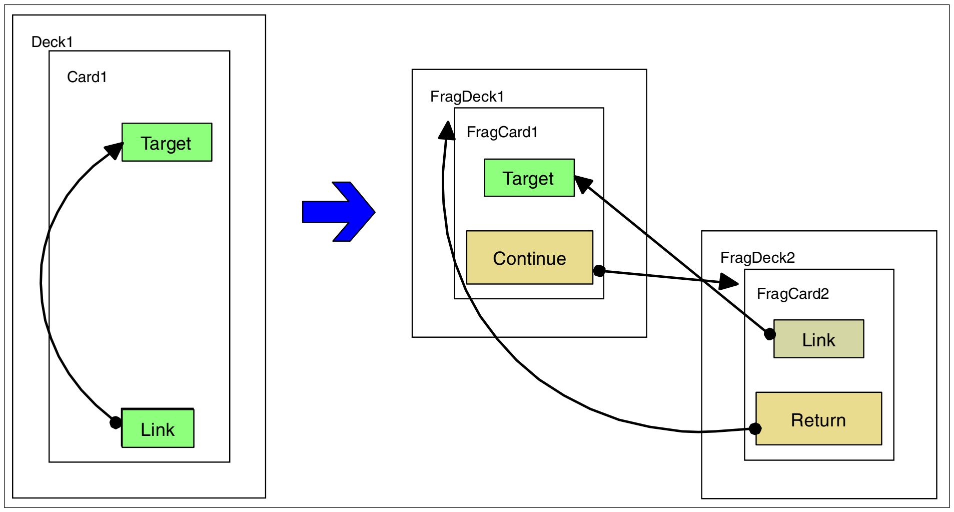
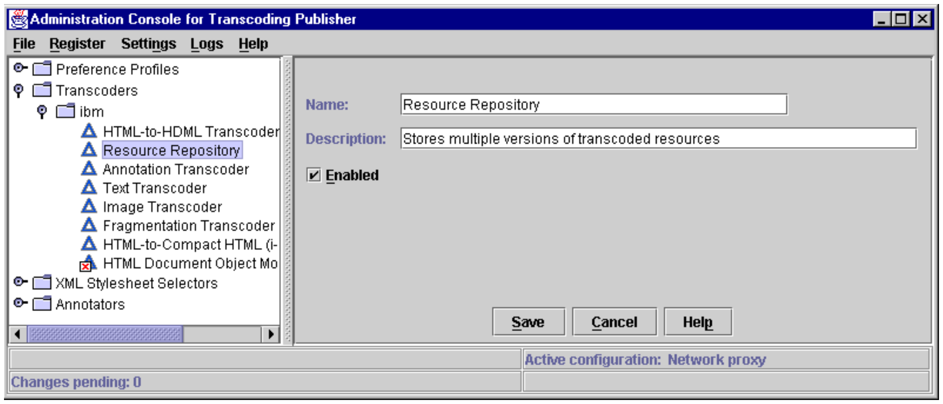


Figure 7-2 Fragmented WML deck

After fragmentation is performed, the first fragment is sent to the client as an HTTP response. The fragmentation engine stores non-first fragments in a general-purpose resource repository. The resource repository appears as a transcoder in the Administration Console. Figure 7-3 on page 263 shows the resource repository in the Administration Console. Making the resource repository general purpose will allow for reuse by other components needing a similar service in the future. But for now, only the fragmentation engine uses the resource repository. Because the fragmentor needs the resource repository to save fragments for later retrieval, the resource repository should *not* be disabled if fragmentation is being used. The Administration Console does not enforce this because it is unaware of the dependence. If the fragmentor is disabled, the resource repository should also be disabled; this will improve performance.



Name: Resource Repository
Description: Stores multiple versions of transcoded resources
✓ Enabled
Active configuration: Network proxy
Changes pending: 0

Figure 7-3 Resource repository as a transcoder

All fragments are named so that a request for any fragment will be routed back to the transcoder.

- For the proxy model, this means prefixing the URL with a special string that will trigger the resource repository generator to retrieve and return the fragment.
- For the servlet model, this means including the Web server host name and the name of the transcoding servlet so that the transcoder is invoked to handle the request.
- For the reverse proxy model, this means including the local WebSphere Transcoding Publisher server's host name so that the request will be routed back to the transcoding server.

Fragments are kept until the original document expires. If a request for a discarded fragment is received, a `Fragment expired` message is sent.

7.1.2 Fragmentable elements

Fragmentable elements are:

- Nodes (tags) with children
- Nodes (tags) that can be safely cloned with child nodes distributed among the clones ("safely" means that the resulting markup is valid and the content's meaning and/or presentation is essentially unchanged)

Note that elements with no children (for example, break elements) are not listed below as fragmentable, but a card/page may be split at one of these elements.

- WML fragmentable elements are:

  <wml>, <card>, <p>, <em>, <strong>, <i>, <b>, <u>, <big>, <small>, <table>

- i-Mode fragmentable elements are:

  <html>, <body>, <p>, <blockquote>, <blink>, <center>, <dir>, <div>, <font>, <plaintext>, <pre>, <ul>

- HDML fragmentable elements are:

  <hdml>, <display>, <choice>, <center>, <line>, <right>, <wrap>

7.1.3 Common problems

Invalid input (invalid content) will cause a request to be rejected (FragmentRejectedException). Sometimes, content cannot be fragmented into small enough pieces. This is likely to happen with the folowing elements:

- Fragmentable elements provided as a reference
- Long paragraphs with no breaks
- Large forms

7.1.4 Example

Figure 7-4 on page 265 is a simple example of WML deck fragmentation. The fragmentor traverses the tree depth-first. At each node (tag), the fragmentor calculates the size of the page represented by the nodes visited so far, plus any descendants of the current node. If this amount exceeds the maximum size, then either the tree is fragmented before the current node, or the subtree(s) of the current node is recursively considered for fragmentation.

The double line in Figure 7-4 indicates where the fragmentor determines that this tree must be fragmented.

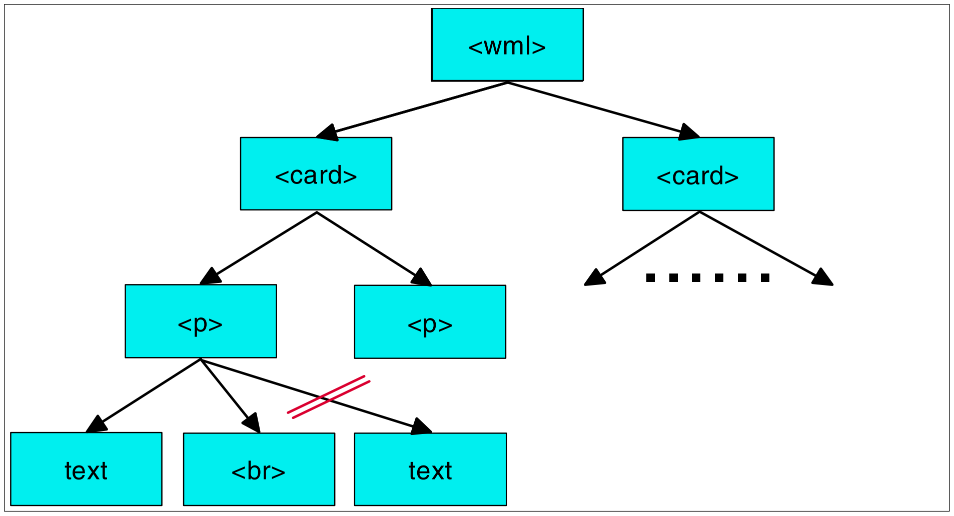


Figure 7-4 Fragmentation example

Figure 7-5 illustrates the first fragment resulting from fragmenting the WML deck as indicated in Figure 7-4.

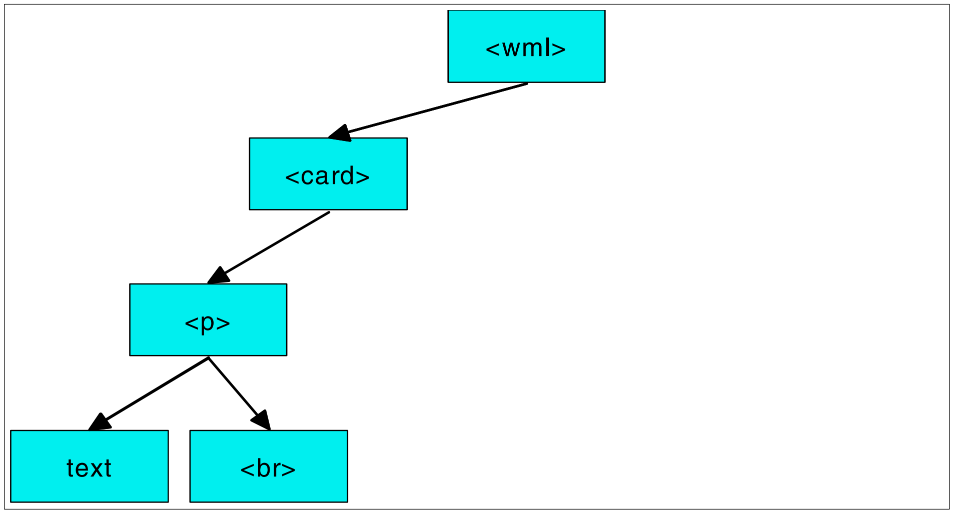


Figure 7-5 First fragment

Figure 7-6 is the second fragment. Notice that the <wml>, <card>, and <p> elements from the original card were duplicated, with children distributed or duplicated as necessary between the new and old elements. This remaining fragment may still be too large for the target device, so the fragmentor begins again at the top of this DOM to see if it needs to be fragmented.

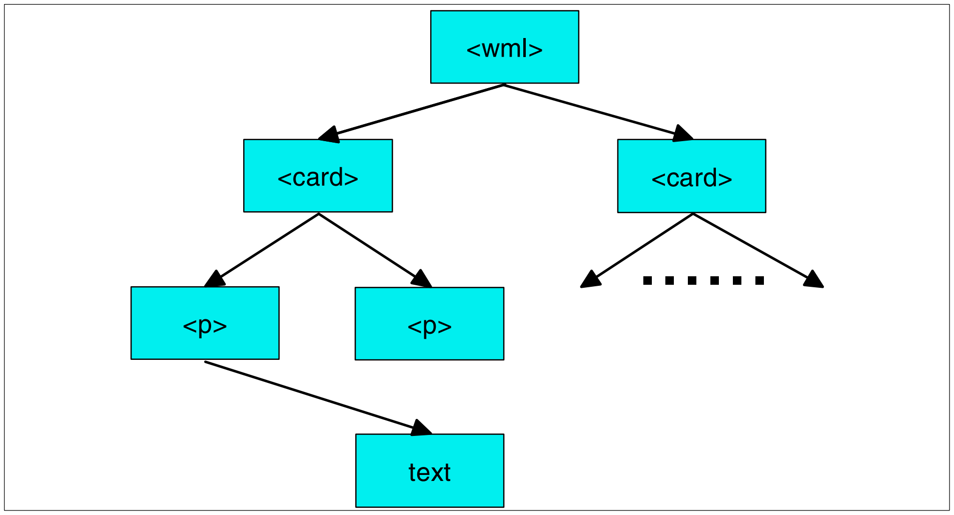


Figure 7-6 Next fragment

7.2 WML fragmentation in WebSphere Everyplace Server environment

WML fragmentation algorithms in WebSphere Transcoding Publisher (WTP) include URLs that expect the device to be using WebSphere Transcoding Publisher as a proxy server. That is, WebSphere Transcoding Publisher assumes in this case that the device is connecting directly to the WebSphere Transcoding Publisher proxy. However, when running WebSphere Transcoding Publisher behind another proxy, such as WebSEAL-Lite (WTE/WSL) in WebSphere Everyplace Server, WebSphere Transcoding Publisher fragmentation will not work properly, since the first proxy (WTE) does not understand the URLs that WebSphere Transcoding Publisher created.

Note: In a WebSphere Everyplace Server environment, WebSEAL-Lite (WSL) running as a plug-in of WTE provides user authentication function.

In this section, we include two sample scenarios to illustrate how you will configure WTE running as a reverse proxy and connected to WebSphere Transcoding Publisher running as a reverse proxy and as a forward proxy.

Note: WTE/WSL and WebSphere Transcoding Publisher must be properly configured when running applications using WML fragmentation.

7.2.1 Scenario 1: Running WebSphere Transcoding Publisher as a reverse proxy

In this section, we describe a sample scenario for WML fragmentation in a WebSphere Everyplace Server environment where both WebSEAL-Lite (a plug-in of WTE) and WebSphere Transcoding Publisher are configured as reverse proxies. The WAP client device can be connected to a WebSphere Everyplace Server domain in the following ways:

- Using HTTP to WebSEAL-Lite using a WAP simulator. This is commonly done for application development.
- Using a WAP/IP connection to the Everyplace Wireless Gateway (EWG).
- Using other WAP connections such as using PPP protocol.

In the scenario, we show you how to configure WebSphere Transcoding Publisher as a reverse proxy, WTE as a reverse proxy and WAP Gateway connected to a reverse proxy (WTE/WSL). The scenario is illustrated in Figure 7-7.

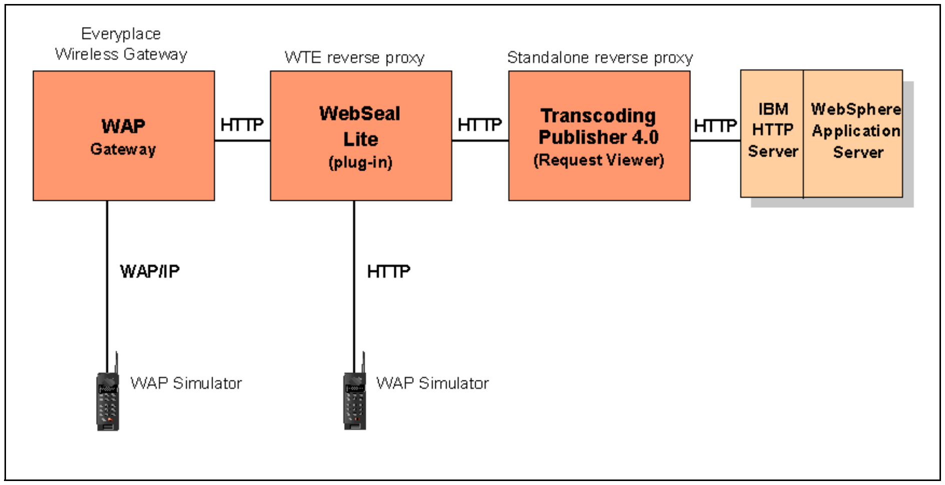


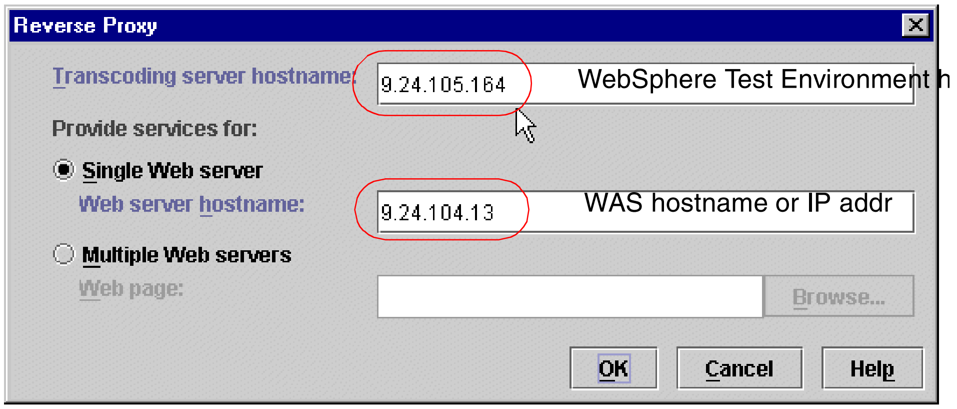
Figure 7-7 WML fragmentation with WebSphere Transcoding Publisher as a reverse proxy

In this scenario, there are basically two important issues you will need to understand:

1. The WAP device in this common WebSphere Everyplace Server environment is not directly connected to WebSphere Transcoding Publisher. Therefore, WebSphere Transcoding Publisher reverse proxy must provide the return address for the first proxy in the chain, that is, the WTE host name or IP address (see Figure 7-8).
2. Transcoding Publisher (WTP) does not let you configure the port of the first reverse proxy in the chain and will just default to its own port. Therefore, you will need to use exactly the same port number you are using in WebSphere Transcoding Publisher, for example, one of the default reverse proxy ports 80, 81 or 82.

Restriction: When WTE/WSL and WebSphere Transcoding Publisher are both configured as reverse proxies, you must use the same port number. In addition, because of this restriction, you cannot run both WTE and WebSphere Transcoding Publisher on the same machine because of port conflicts.

The WebSphere Transcoding Publisher reverse proxy configuration is illustrated in Figure 7-8. In this scenario, there is only one application server, so the *Single Web server* option is selected and its host name or IP address is also configured.


Reverse Proxy

Transcoding server hostname: 9.24.105.164 WebSphere Test Environment h

Provide services for:

(●) Single Web server

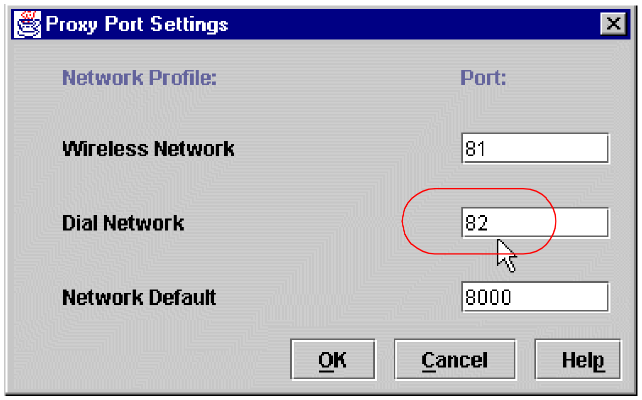
Web server hostname: 9.24.104.13 WAS hostname or IP addr

( ) Multiple Web servers

Web page:

Figure 7-8 WebSphere Transcoding Publisher reverse proxy configuration for WML fragmentation

In this scenario, WebSphere Transcoding Publisher is configured to use ports 81, 82 and 8000 (see Figure 7-9), and the sample application will use port 82 to illustrate the process.


Proxy Port Settings

Network Profile:	Port:
Wireless Network	81
Dial Network	82
Network Default	8000

Figure 7-9   WebSphere Transcoding Publisher reverse proxy ports

Since WebSphere Transcoding Publisher running as a reverse proxy is configured to listen on these ports and the application is using port 82, you will need to configure port 82 as the listening port in WTE Port directive in the ibmproxy.conf file as follows:

```
Port 82
```

In addition, you will need to specify the protocols that this proxy server will forward. For example, to forward all HTTP requests to WebSphere Transcoding Publisher reverse proxy server, you configure the Proxy directive as follows:

```
Proxy   /*         http://9.24.106.193:82/*
```

where `9.24.106.193` is the IP address (you can also use the host name) of WebSphere Transcoding Publisher reverse proxy and `82` is its listening port.

The application can now point to the WTE/WSL reverse proxy when connecting to the WebSphere Everyplace Server domain. For example:

```
<a href="http://9.24.105.164:82/servlet/HelloWorldServlet">Hello World</a>
<a
href="http://9.24.105.164:82/WebSphereSamples/servlet/WebSphereSamples.Your
Co.News.ServeXML_news?xml_file=MainNewsfeed_new.XML">YourCo Application</a>
```

where `9.24.105.164` is the IP address (or host name) of WTE/WSL reverse proxy and `82` is its listening port.

In this scenario, WTE/WSL will forward HTTP requests to Transcoding Publisher and, in turn, WebSphere Transcoding Publisher will forward the request to the application server. In a similar way, the WTE reverse proxy will forward all fragment requests to Transcoding Publisher.

For WAP connections, the Everyplace Wireless Gateway must be configured to connect to a reverse proxy (WTE/WSL). The WAP Gateway configuration is shown in Figure 7-10 on page 271; this option is not available during installation and you will need to use the Gatekeeper.

Hint: As you can see in this scenario, it is important to notice that the WTE/WSL reverse proxy IP address and listening port number are specified in two different places:

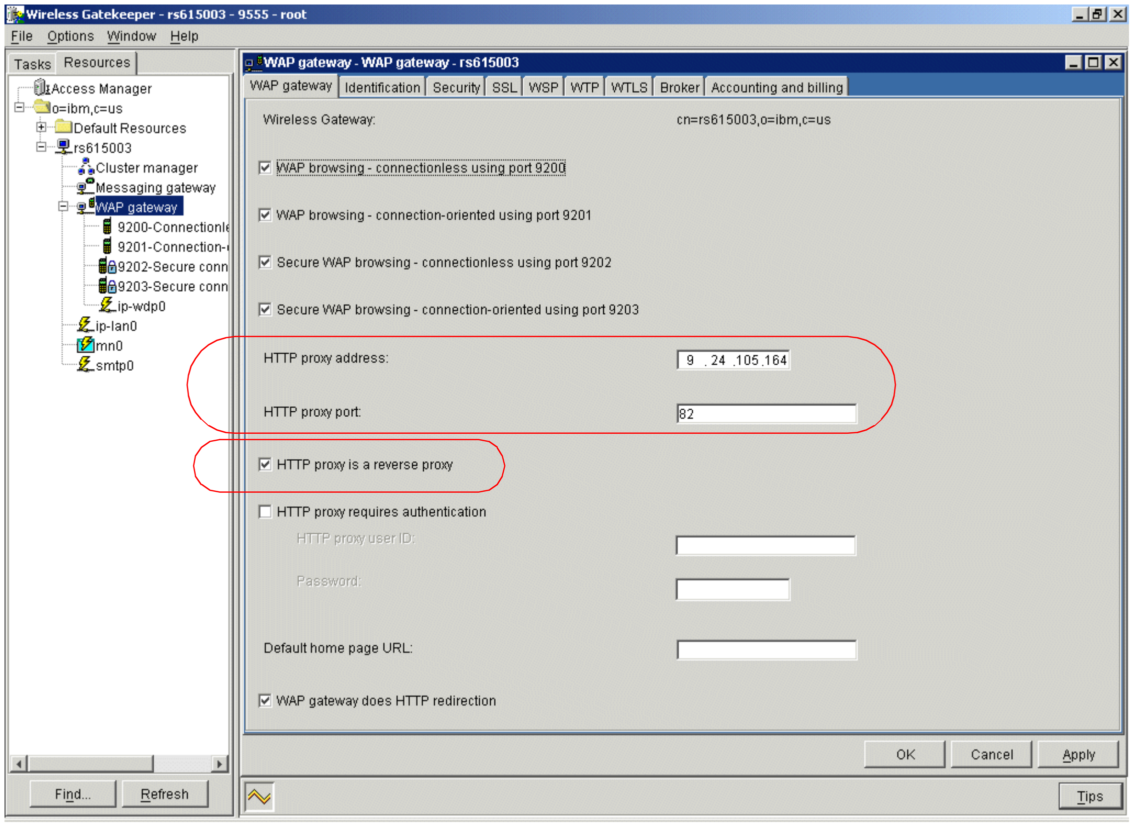
- In the application.
- In the WAP Gateway configuration.

When using the WAP Gateway connected to a reverse proxy, the values you configured in the WAP Gateway are used. This means that any values can be used in the application since they will be ignored. For example:

```
<a href="http://XXXX:YY/servlet/HelloWorldServlet">Hello World</a>
```

where `XXXX` is any host name or IP address and `YY` is any port number.

Of course, this is not true if you are connected directly to the WTE/WSL reverse proxy using the HTTP protocol. In this case, the application values will be used.



Wireless Gateway: cn=rs615003,o=ibm,c=us

- [✓] WAP browsing - connectionless using port 9200
- [✓] WAP browsing - connection-oriented using port 9201
- [✓] Secure WAP browsing - connectionless using port 9202
- [✓] Secure WAP browsing - connection-oriented using port 9203

HTTP proxy address: 9 . 24 . 105 . 164

HTTP proxy port: 82

- [✓] HTTP proxy is a reverse proxy
- [ ] HTTP proxy requires authentication

HTTP proxy user ID:

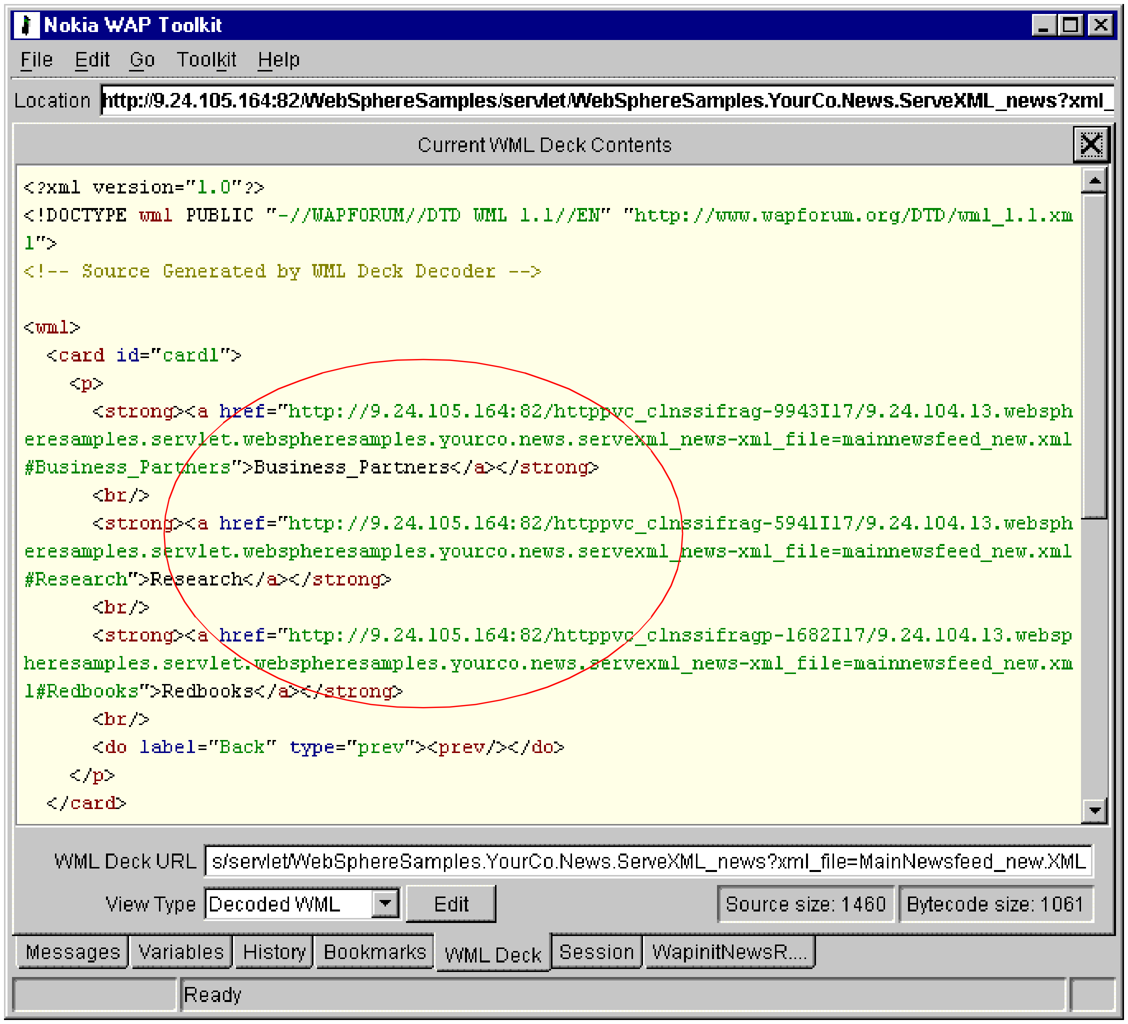
Password:

Default home page URL:

- [✓] WAP gateway does HTTP redirection

Figure 7-10 WAP Gateway configuration to connect to WTE/WSL reverse proxy

Finally, Figure 7-11 on page 272 shows a WML deck where you can see that, as expected, all references in URLs point to the WTE/WSL reverse proxy port 82.


Nokia WAP Toolkit
File Edit Go Toolkit Help
Location http://9.24.105.164:82/WebSphereSamples/servlet/WebSphereSamples.YourCo.News.ServeXML_news?xml_
Current WML Deck Contents

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xm
l">
<!-- Source Generated by WML Deck Decoder -->

<wml>
  <card id="card1">
    <p>
      <strong><a href="http://9.24.105.164:82/httppvc_clnssifrag-9943117/9.24.104.13.websph
eresamples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed_new.xml
#Business_Partners">Business_Partners</a></strong>
      <br/>
      <strong><a href="http://9.24.105.164:82/httppvc_clnssifrag-5941117/9.24.104.13.websph
eresamples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed_new.xml
#Research">Research</a></strong>
      <br/>
      <strong><a href="http://9.24.105.164:82/httppvc_clnssifragp-1682117/9.24.104.13.websp
heresamples.servlet.webspheresamples.yourco.news.servexml_news-xml_file=mainnewsfeed_new.xm
l#Redbooks">Redbooks</a></strong>
      <br/>
      <do label="Back" type="prev"><prev/></do>
    </p>
  </card>
```

WML Deck URL: s/servlet/WebSphereSamples.YourCo.News.ServeXML_news?xml_file=MainNewsfeed_new.XML

View Type: Decoded WML

Source size: 1460 Bytecode size: 1061

Edit
Messages Variables History Bookmarks WML Deck Session WapinitNewsR....
Ready

Figure 7-11 WML deck in WAP simulator

7.2.2 Scenario 2: Running WebSphere Transcoding Publisher as a forward proxy

In this section, we describe a sample scenario for WML fragmentation in a WebSphere Everyplace Server environment where WebSEAL-Lite (a plug-in of WTE) is configured as a reverse proxy and Transcoding Publisher (WTP) is configured as a network proxy (forward proxy). The scenario is illustrated in Figure 7-12.

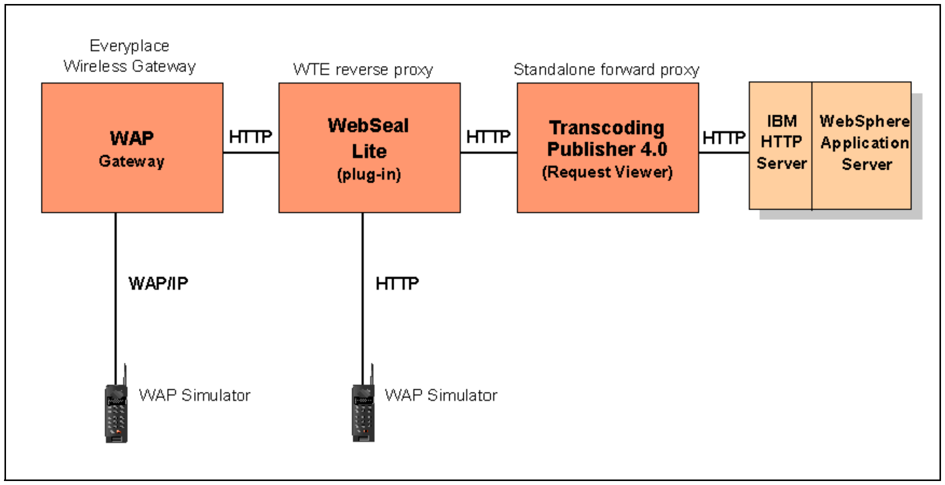
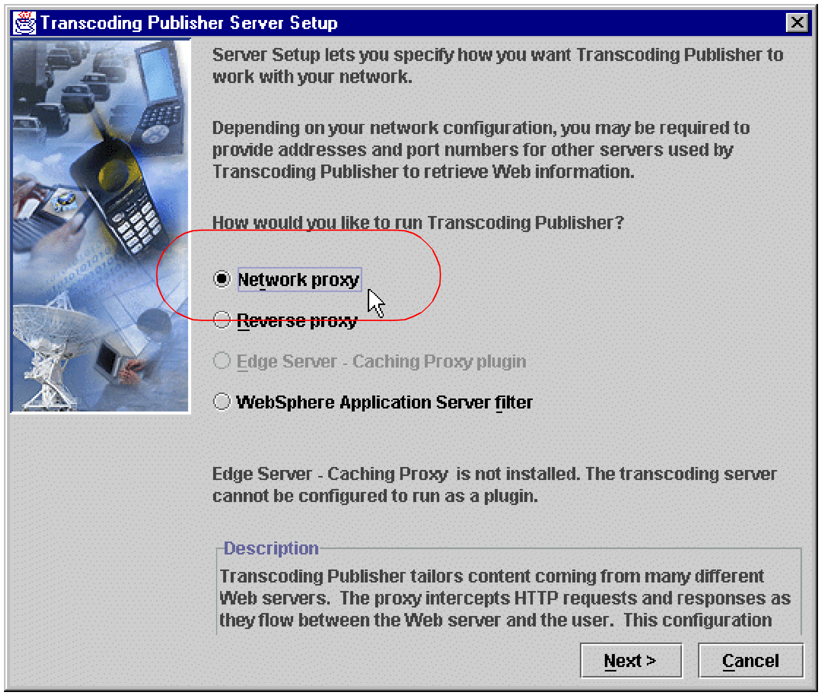


Figure 7-12 WML fragmentation with WebSphere Transcoding Publisher as forward proxy

Transcoding Publisher is configured as a forward proxy (see Figure 7-13).



Server Setup lets you specify how you want Transcoding Publisher to work with your network.

Depending on your network configuration, you may be required to provide addresses and port numbers for other servers used by Transcoding Publisher to retrieve Web information.

How would you like to run Transcoding Publisher?

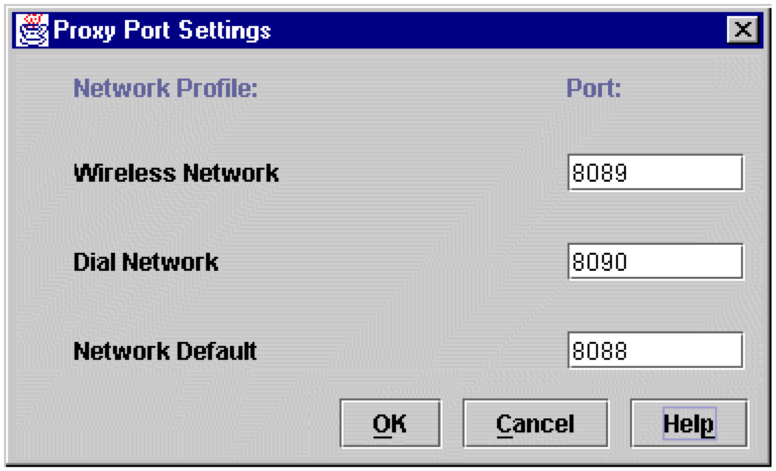
● Network proxy
○ Reverse proxy
○ Edge Server - Caching Proxy plugin
○ WebSphere Application Server filter

Edge Server - Caching Proxy is not installed. The transcoding server cannot be configured to run as a plugin.

Description
Transcoding Publisher tailors content coming from many different Web servers. The proxy intercepts HTTP requests and responses as they flow between the Web server and the user. This configuration

Figure 7-13   WebSphere Transcoding Publisher configured as a forward stand-alone proxy

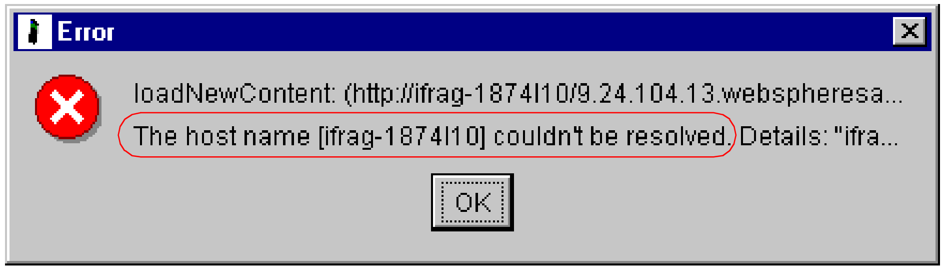
The default ports shown in Figure 7-14 will be used by WebSphere Transcoding Publisher configured as a forward proxy.


Proxy Port Settings

Network Profile:	Port:
Wireless Network	8089
Dial Network	8090
Network Default	8088

Figure 7-14 WebSphere Transcoding Publisher forward proxy ports (default values)

In this scenario, Transcoding Publisher fragmentation implements algorithms using URLs that expect the client device to be using WebSphere Transcoding Publisher as a proxy server. This is done since WebSphere Transcoding Publisher assumes that the device is connecting directly to the WebSphere Transcoding Publisher. When running WebSphere Transcoding Publisher behind another reverse proxy, this breaks down, since the first proxy (WTE/WSL) does not understand the URLs that WebSphere Transcoding Publisher created. Figure 7-15 shows the error message indicating that the host name starting with the prefix `ifrag` could not be resolved.


Error

```
loadNewContent: (http://ifrag-1874l10/9.24.104.13.webspheresa...
The host name [ifrag-1874l10] couldn't be resolved. Details: "ifra...
```

Figure 7-15 WML fragmentation error

In order to eliminate this problem, you will need to change the WebSphere Transcoding Publisher's fragment identifiers by editing the FragmentationEngineConfiguration.prop file in the WebSphere Transcoding Publisher directory:

IBMTrans->etc->plugins->ibm->FragmentationEngine

Note: When using LDAP in a WebSphere Everyplace Server environment, this needs to be done using the DMT tool (or another LDAP manipulation tool). When using DMT, select the WebSphere Transcoding Publisher server model that you want to update and navigate through the directory as follows:

> **WTP Server Model->etc->plugins->ibm->FragmentationEngine ->FragmentationEngineConfiguration**

The updated properties file (FragmentationEngineConfiguration.prop) should be updated as follows:

1. Replace the statement `FragmentSpecifier=ifrag-` with the following statement:

   ```
   FragmentSpecifier=1.1.1.1/ifrag-
   ```

2. Replace the statement `PrimarySpecifier=ifragp-` with the following statement:

   ```
   PrimarySpecifier=1.1.1.1/ifragp-
   ```

In this scenario `1.1.1.1` is any dummy IP address or host name, for example a.b.c.d, mywte, mywsl or any other value. The point here is that this value will be included in a fragment request and when the WAP Gateway in the Wireless Gateway (EWG) is configured to connect to a reverse proxy (WTE/WSL), it will replace this value with the configured WTE proxy address in the WAP Gateway.

Note: In this sample scenario, IP addresses 1.1.1.1 and 2.2.2.2 are any dummy values that will be replaced in the WAP Gateway with the configured WTE reverse proxy IP address. You can also use dummy host names for these values.

In this sample scenario, 1.1.1.1 will be replaced with 9.24.105.164 using port 80 as configured in the WAP Gateway (see Figure 7-16 on page 277). In this scenario, 9.24.105.164 is the WTE/WSL reverse proxy address.

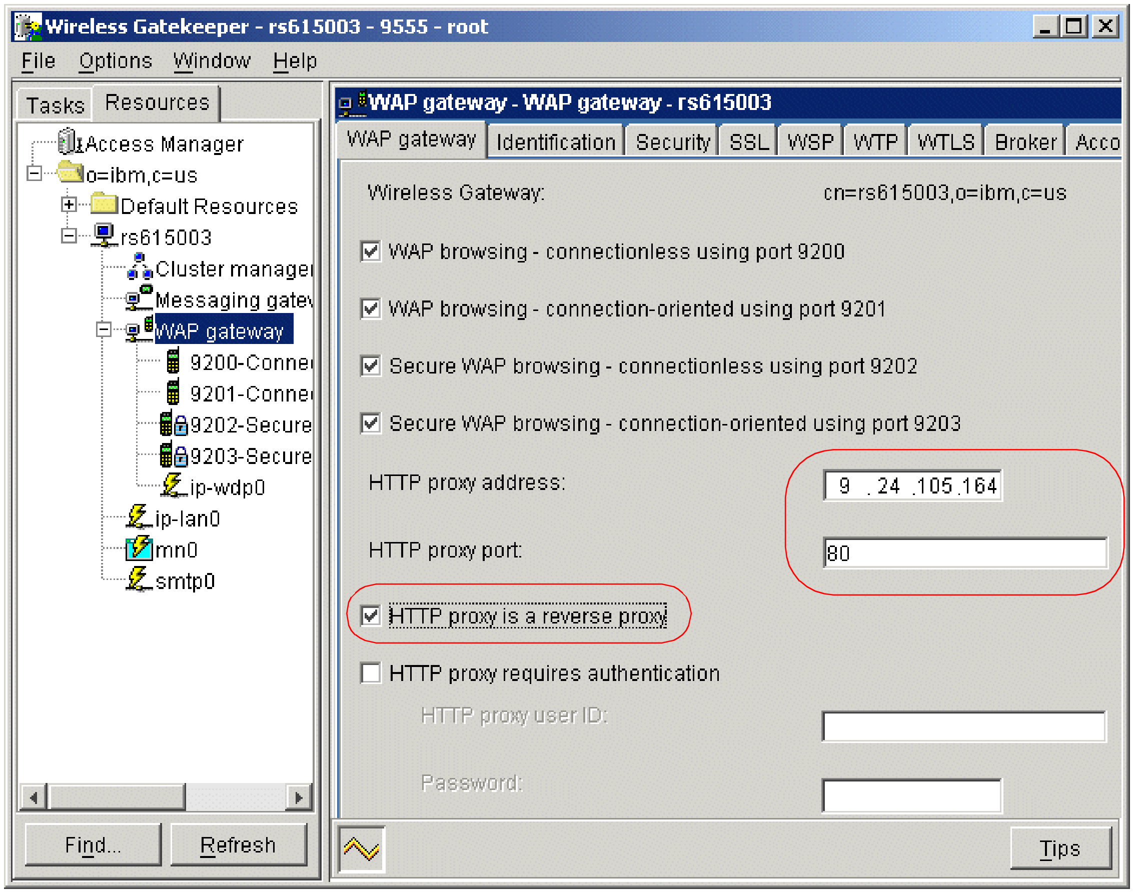
For example, the fragment request:

```
http://1.1.1.1/ifrag-1040I1/...
```

will be changed by the WAP Gateway to the following value:

```
http://9.24.105.164/ifrag-1040I1/...
```

where `9.24.105.164` is the WTE/WSL reverse proxy address. This scheme guarantees that the fragment request will reach the WTE/WSL reverse proxy.



Wireless Gateway: cn=rs615003,o=ibm,c=us

- [✓] WAP browsing - connectionless using port 9200
- [✓] WAP browsing - connection-oriented using port 9201
- [✓] Secure WAP browsing - connectionless using port 9202
- [✓] Secure WAP browsing - connection-oriented using port 9203

HTTP proxy address: 9 . 24 . 105 . 164

HTTP proxy port: 80

- [✓] HTTP proxy is a reverse proxy
- [ ] HTTP proxy requires authentication

HTTP proxy user ID:

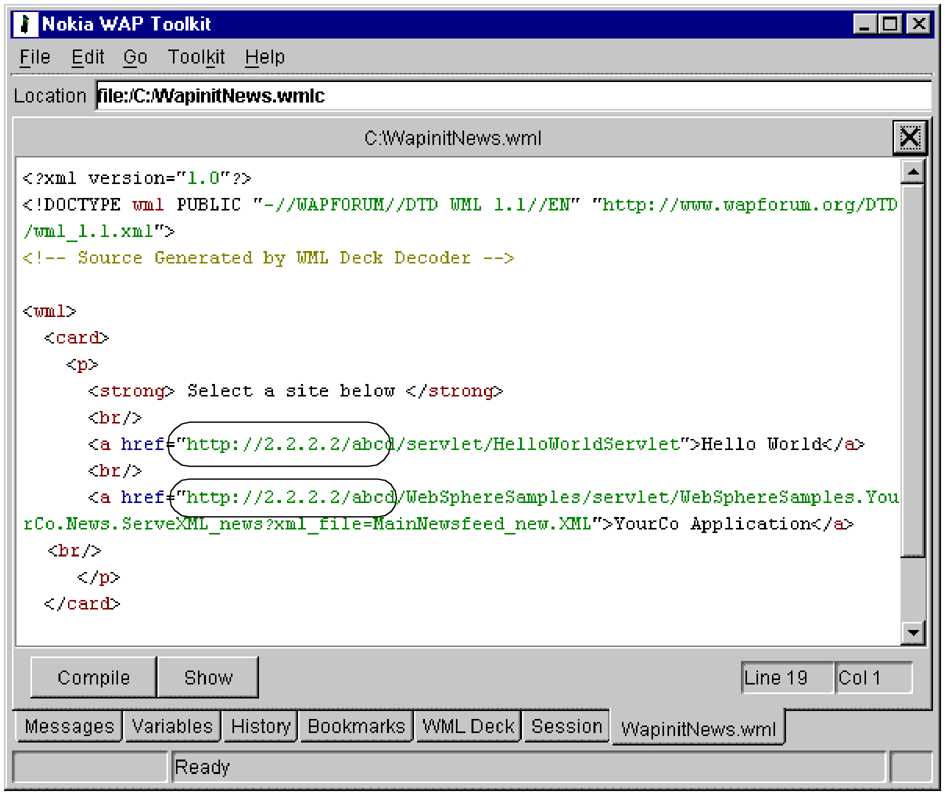
Password:

Figure 7-16 WAP Gateway with HTTP reverse proxy (WTE/WSL) configuration

In a similar way, as shown in Figure 7-17, other non-fragment requests can also use the same scheme and provide a dummy address or host name that eventually will be changed by the WAP Gateway for proper access of the WTE/WSL reverse proxy.

For example, a dummy address of value 2.2.2.2 (see Figure 7-17) used in this sample WML page will be changed by the WAP Gateway to the WTE/WSL reverse proxy IP address.

Note: For this scenario, you must select the **HTTP proxy is a reverse proxy** box (WTE/WSL).



```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD
/wml_1.1.xml">
<!-- Source Generated by WML Deck Decoder -->

<wml>
  <card>
    <p>
      <strong> Select a site below </strong>
      <br/>
      <a href="http://2.2.2.2/abcd/servlet/HelloWorldServlet">Hello World</a>
      <br/>
      <a href="http://2.2.2.2/abcd/WebSphereSamples/servlet/WebSphereSamples.You
rCo.News.ServeXML_news?xml_file=MainNewsfeed_new.XML">YourCo Application</a>
  <br/>
     </p>
  </card>
```

Figure 7-17 Sample WML page

Finally, the WTE reverse proxy is configured in the ibmproxy.conf file as follows:

1. Configure the http_proxy directive to point to WebSphere Transcoding Publisher forward proxy using a valid port. For example:

```
http_proxy      http://9.24.106.193:8088/
```

2. Configure any application required Proxy directives. For example, in this scenario, the application needs the following directives:

```
Proxy /abcd/*             http://9.24.104.13/*
Proxy /theme/*            http://9.24.104.13/theme/*
Proxy /WebSphereSamples/* http://9.24.104.13/WebSphereSamples/*
```

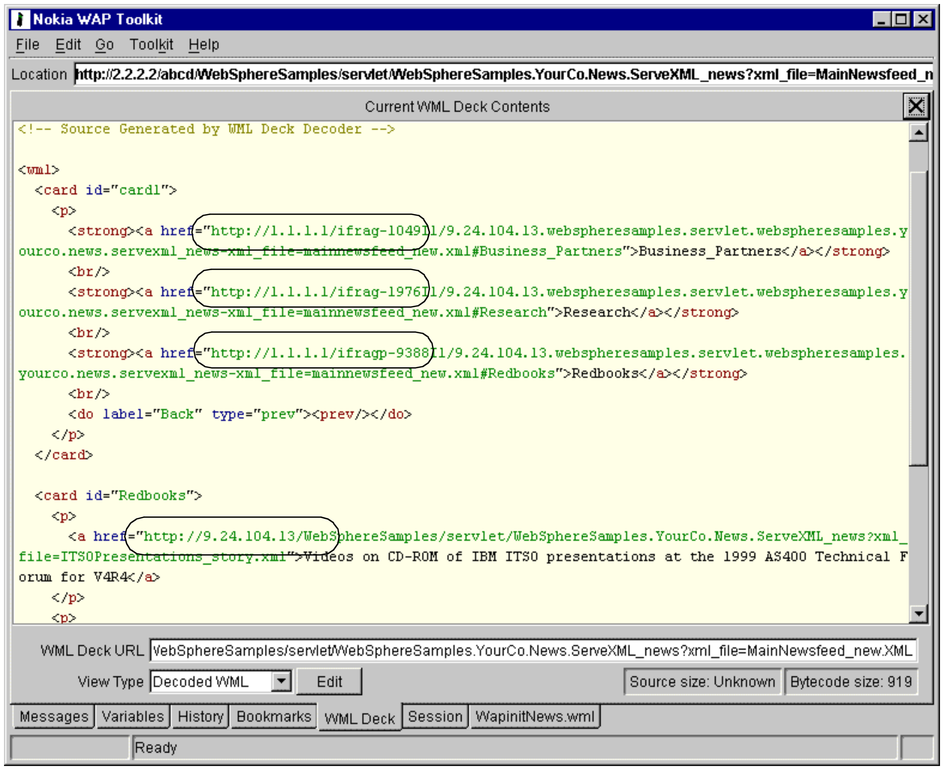
Note: This scenario targets a single application server with IP address 9.24.104.13. When using multiple application servers, you will need to configure additional Proxy directives.

3. Configure a Proxy directive for the fragmentation specifier so that it can be recognized by WebSphere Transcoding Publisher. This is because the WAP Gateway will replace the original value. For example, in this scenario:

   ```
   Proxy /ifrag*   http://1.1.1.1/ifrag*
   ```

   Note: The IP address you use should match exactly what you have previously configured in the FragmentationEngineConfiguration.prop file. In this sample scenario it is `1.1.1.1`, but any value can be used as long as it matches the value in the Proxy directive.

A sample transcoded page is shown in Figure 7-18. Notice that IP addresses `1.1.1.1` and `9.24.104.13` used in this application will be replaced by the WAP Gateway with the address of the configured WTE reverse proxy.



Location http://2.2.2.2/abcd/WebSphereSamples/servlet/WebSphereSamples.YourCo.News.ServeXML_news?xml_file=MainNewsfeed_n

```
<!-- Source Generated by WML Deck Decoder -->

<wml>
  <card id="card1">
    <p>
      <strong><a href="http://1.1.1.1/ifrag-1049I1/9.24.104.13.webspheresamples.servlet.webspheresamples.y
ourco.news.servexml_news-xml_file=mainnewsfeed_new.xml#Business_Partners">Business_Partners</a></strong>
      <br/>
      <strong><a href="http://1.1.1.1/ifrag-1976I1/9.24.104.13.webspheresamples.servlet.webspheresamples.y
ourco.news.servexml_news-xml_file=mainnewsfeed_new.xml#Research">Research</a></strong>
      <br/>
      <strong><a href="http://1.1.1.1/ifragp-9388I1/9.24.104.13.webspheresamples.servlet.webspheresamples.
yourco.news.servexml_news-xml_file=mainnewsfeed_new.xml#Redbooks">Redbooks</a></strong>
      <br/>
      <do label="Back" type="prev"><prev/></do>
    </p>
  </card>

  <card id="Redbooks">
    <p>
      <a href="http://9.24.104.13/WebSphereSamples/servlet/WebSphereSamples.YourCo.News.ServeXML_news?xml_
file=ITSOPresentations_story.xml">Videos on CD-ROM of IBM ITSO presentations at the 1999 AS400 Technical F
orum for V4R4</a>
    </p>
    <p>
```

WML Deck URL VebSphereSamples/servlet/WebSphereSamples.YourCo.News.ServeXML_news?xml_file=MainNewsfeed_new.XML

View Type Decoded WML | Edit | Source size: Unknown | Bytecode size: 919

Figure 7-18 Sample WML transcoded page

Important: The suggested way, shown in this section, to support WML fragmentation in WebSphere Everyplace Server, when WTE/WSL is configured as a reverse proxy and WebSphere Transcoding Publisher is configured as a forward proxy, relies on the fact that for HTTP and fragment requests the WAP Gateway in EWG replaces the host name or IP address with the IP address of the configured reverse proxy. Therefore, this scheme cannot be used when using a WAP simulator connected directly to WTE/WSL reverse proxy using the HTTP protocol.

8 Going wireless!

In this chapter, we describe Everyplace Wireless Gateway (EWG) issues involved in accessing the B2E sample application used in this redbook from WAP phones and other wireless devices. You will also find information about the Wireless Gateway components and the features of the EWG Version 2.1.1 release, including WAP over IP, WAP Push, wireless clients and so on.

Note: The Everyplace Wireless Gateway (EWG) is available for AIX and Sun Solaris platforms and is a component of WebSphere Everyplace Server Service Provider Offering (SPO) Version 2.1. It is also an optional component of the WebSphere Everyplace Server Enable Offering (EO) Version 1.1. In addition, EWG can also be used as a standalone product.

8.1 Overview

The Everyplace Wireless Gateway (EWG), a component of the IBM WebSphere Everyplace Server (WES), is a distributed, scalable, multipurpose UNIX communications platform. It supports optimized, secure data access by both Wireless Application Protocol (WAP) clients and non-WAP clients over a wide range of international wireless network technologies, as well as local area (LAN) and wide area (WAN) wire line networks. The Wireless Gateway integrates the WAP Version1.2.1 standard support as defined by the WAP Forum.

All the EWG features addressed in this chapter are included in Everyplace Wireless Gateway Version 2.1.1. Figure 8-1 shows the Everyplace Wireless Gateway (EWG) with supported packet networks, GPRS, cellular networks, private RF networks, dial-up telephone (PSTN) and WAP support.

Note: Everyplace Wireless Gateway is available for AIX and Solaris platforms.

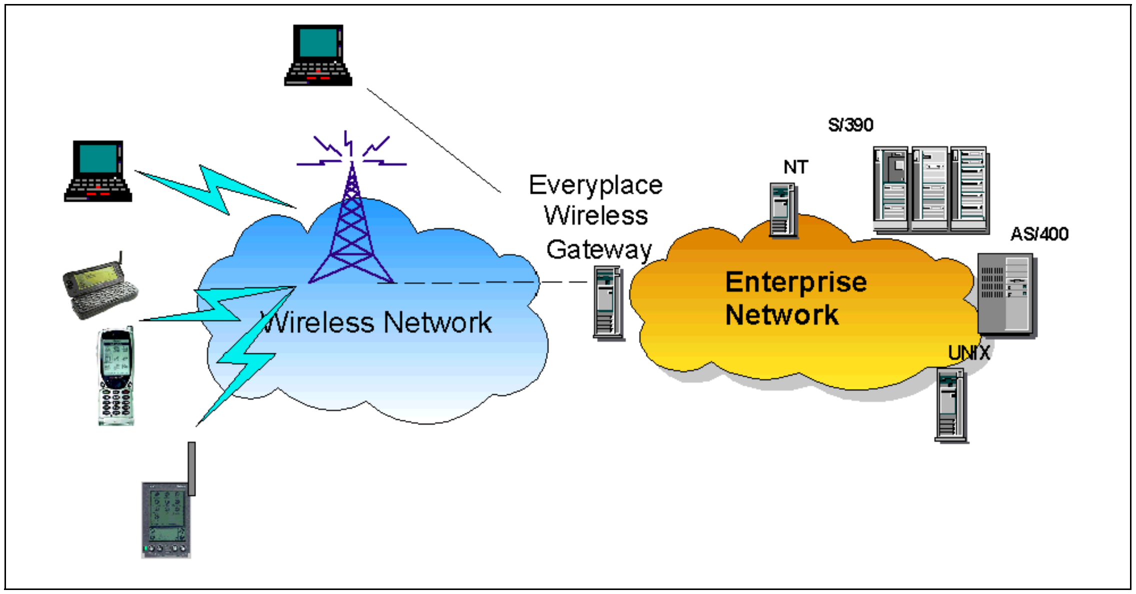


Figure 8-1 Everyplace Wireless Gateway overview

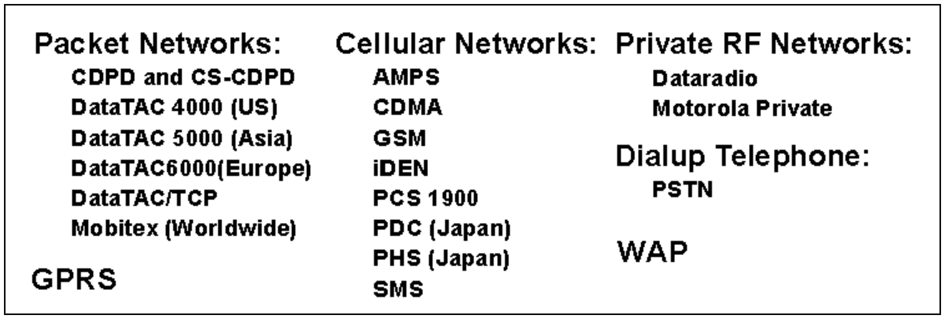
The main components of the Everyplace Wireless Gateway are:

- The Wireless Gateway, which runs on the IBM AIX and Sun Solaris operating systems, provides a standard communications interface (TCP/IP) to a variety of wireless, dial-up, and LAN networks with data optimization and security.
- The Wireless Gatekeeper, a Java-based administrator's console, provides an easy-to-use interface that enables you to configure Wireless Gateways,

define wireless resources, group the resources to control access, and assign administrators to perform operations on the resources as needed. The Wireless Gatekeeper enables one or more administrators to work with Wireless Gateways remotely.

- The Wireless Client provides an optimized and secure IP tunnel for communication with the Wireless Gateway using a variety of wireless and wireline networks.
- Persistent data storage consists of independent databases containing information about the resources comprising your wireless network. The databases are directory services using LDAP and ODBC-compliant relational databases.
- The access manager program is an AIX or Solaris daemon that manages communications among Wireless Gatekeepers, the Wireless Gateway and persistent data storage.

Figure 8-2 shows the Everyplace Wireless Gateway (EWG) supported networks.



Packet Networks:
- CDPD and CS-CDPD
- DataTAC 4000 (US)
- DataTAC 5000 (Asia)
- DataTAC6000(Europe)
- DataTAC/TCP
- Mobitex (Worldwide)

GPRS

Cellular Networks:
- AMPS
- CDMA
- GSM
- iDEN
- PCS 1900
- PDC (Japan)
- PHS (Japan)
- SMS

Private RF Networks:
- Dataradio
- Motorola Private

Dialup Telephone:
- PSTN

WAP

Figure 8-2 Everyplace Wireless Gateway supported networks

WAP phones

WAP clients, such as WAP-compliant mobile phones, can connect to the Wireless Gateway using a microbrowser. When the Wireless Gateway is configured as a WAP Gateway, it uses Wireless Session Protocol (WSP) to link the microbrowser with connection-oriented and connectionless WAP services.

WAP clients connect to the Wireless Gateway through a circuit-switched or dial IP network (GSM, TDMA, CDMA) or through an SMS network (SMPP, UCP).

Your WAP phone needs some configuration to be used with the Wireless Gateway. You have to provide your WAP phone with the EWG IP address and the home page URL of your application.

The following mobile devices have been proven to work with the WAP Wireless Gateway:

- Ericsson R320s
- Ericsson R380s
- Ericsson R380 World
- Ericsson A1228c
- Ericsson R520
- Ericsson R278d
- Mitsubishi Trium Geo
- Siemens M35i
- Motorola Timeport P7389 Tri-Band
- Motorola TalkAbout T2288
- Nokia 7110
- Nokia 7190
- Nokia 7160

Windows CE Client

The Wireless Client can be installed on the following Windows CE platforms:

- Versions: H/PC Pro 2.11, HPC 2000, PocketPC
- Processors: StrongARM, MIPS, SH3

Palm OS

Supported devices are:

- Palm devices that come with Palm OS 4.0
- Palm devices that come with Palm OS 3.5 and are upgradable to Palm OS 3.5.3

Supported networks are:

- Dial-up (for example, IBM or Palm modems)
- Direct Serial (for example, Minstrel wireless modem)

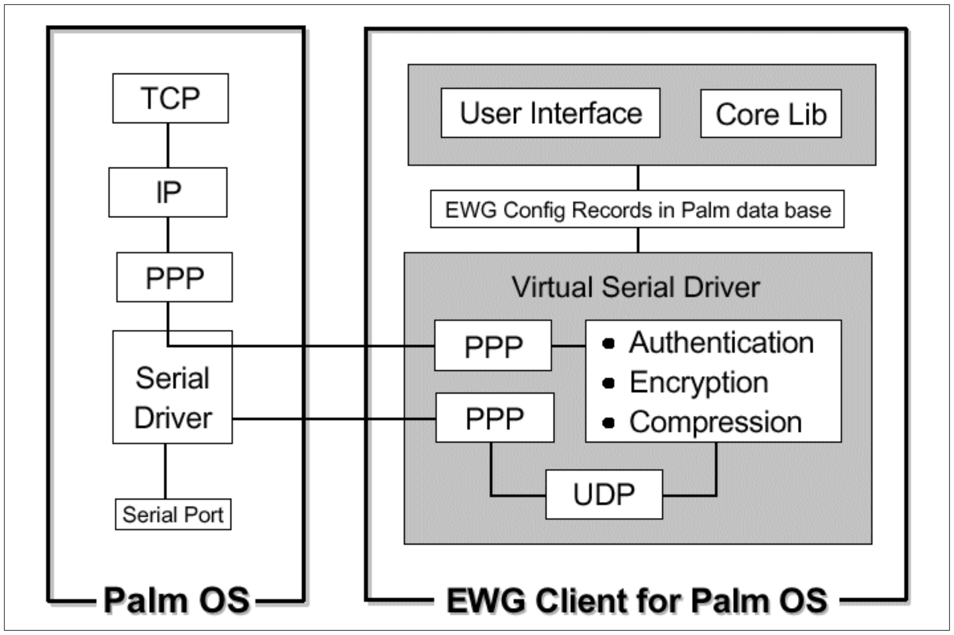


Figure 8-3 Wireless Client for Palm OS

The limitations of the Wireless Client for Palm OS are as follows:

- Palm Infrared port is not supported
- Callback is not supported
- Short hold is not supported
- Triple DES is not supported

8.1.1 Connectivity

The Wireless Gateway allows multiple simultaneous connections to multiple networks; each physical connection is represented in EWG by a separate interface called a Mobile Network Connection (MNC). In addition, EWG balances the load of messages across similar MNCs and the Network Dispatcher may also be used to scale messaging over multiple Wireless Gateways.

As a part of WebSphere Everyplace Server, EWG interacts with the other WebSphere Everyplace Server components, as illustrated in Figure 8-4.

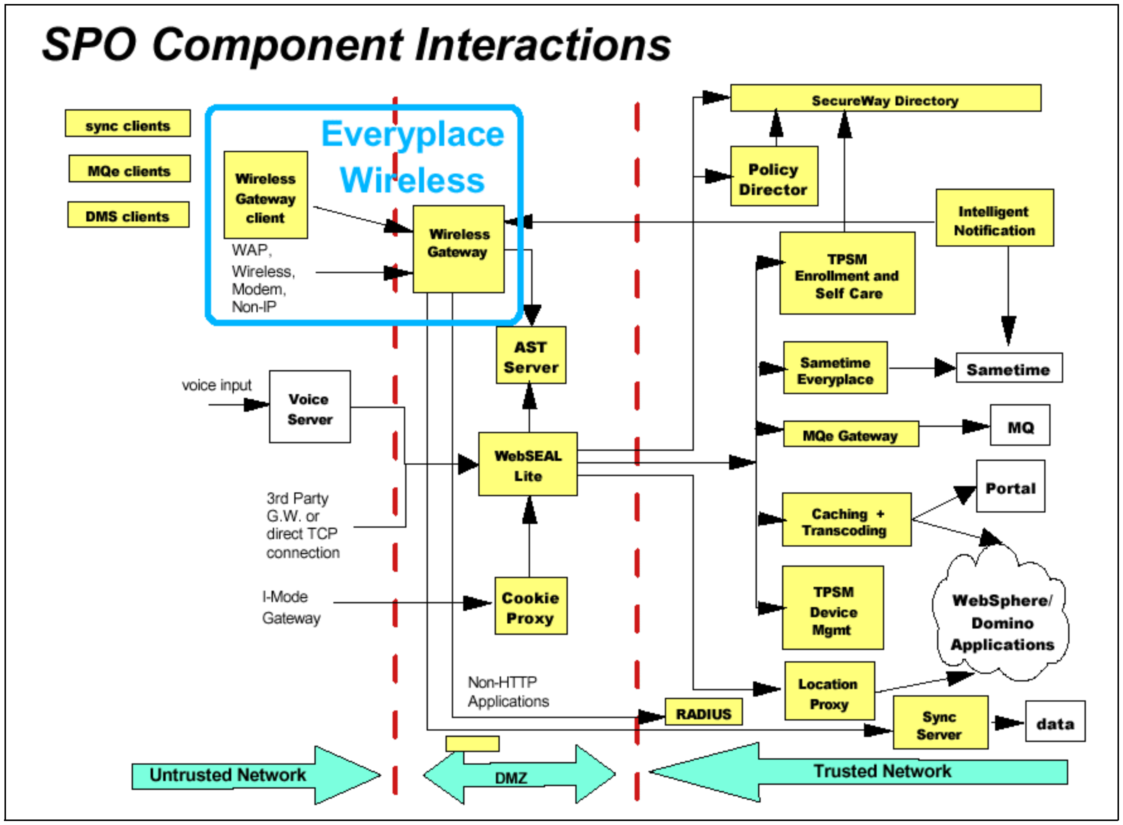


Figure 8-4 EWG interaction with the WebSphere Everyplace Server components

Some of the key features and benefits provided by Everyplace Wireless Gateway are:

- Scalability: EWG supports clustering of gateways for larger systems and backup. Thus it is possible to add gateways to handle increases in traffic without shutting the service down.

  Note: EWG supports remote gateway functionality for corporate environments.

- Messaging Gateway provides push messaging capabilities for clients such as WAP phones, SMS messaging and others. Thus end users can get information whenever and wherever they need it.
- Security: EWG provides two-way user authentication and data encryption for wireless clients using the WLP protocol to connect to the Wireless Gateway. In addition, the WAP Gateway provides support for WTLS secure connections.

- Optimization: EWG improves network response time and reduces the amount of data transmitted with data compression and protocol optimization. As a consequence, data exchange between application and user is faster and more efficient.
- WAP support: EWG allows standards-based support for devices with WAP browsers installed.
- Worldwide network technology support: EWG delivers applications to mobile users over a wide variety of wireless and wired networks.

The main new features provided by the Everyplace Wireless Gateway in this release are:

- Wireless Client support on the handheld PC 2000 and Pocket PC platforms using the Windows CE operating system.
- Wireless Client support on Palm OS.
- Wireless Client support on Windows Millenium.
- TCP application, a new WAP service resource, which makes it possible for WAP application data streams that do not use a browser to be transported to and from WAP client devices.
- Improved storage of cookies on behalf of WAP clients. You can enable session cookies for secure or connection-oriented browse services without requiring authentication.
- Integration with Tivoli SecureWay Policy Director.

8.1.2 Everyplace Wireless Gateway administration

The Everyplace Wireless Gateway (EWG) can integrate all supported networks within a single resource. There is only one Wireless Gateway installed per system. Each Wireless Gateway has configuration information that designates resources assigned to it.

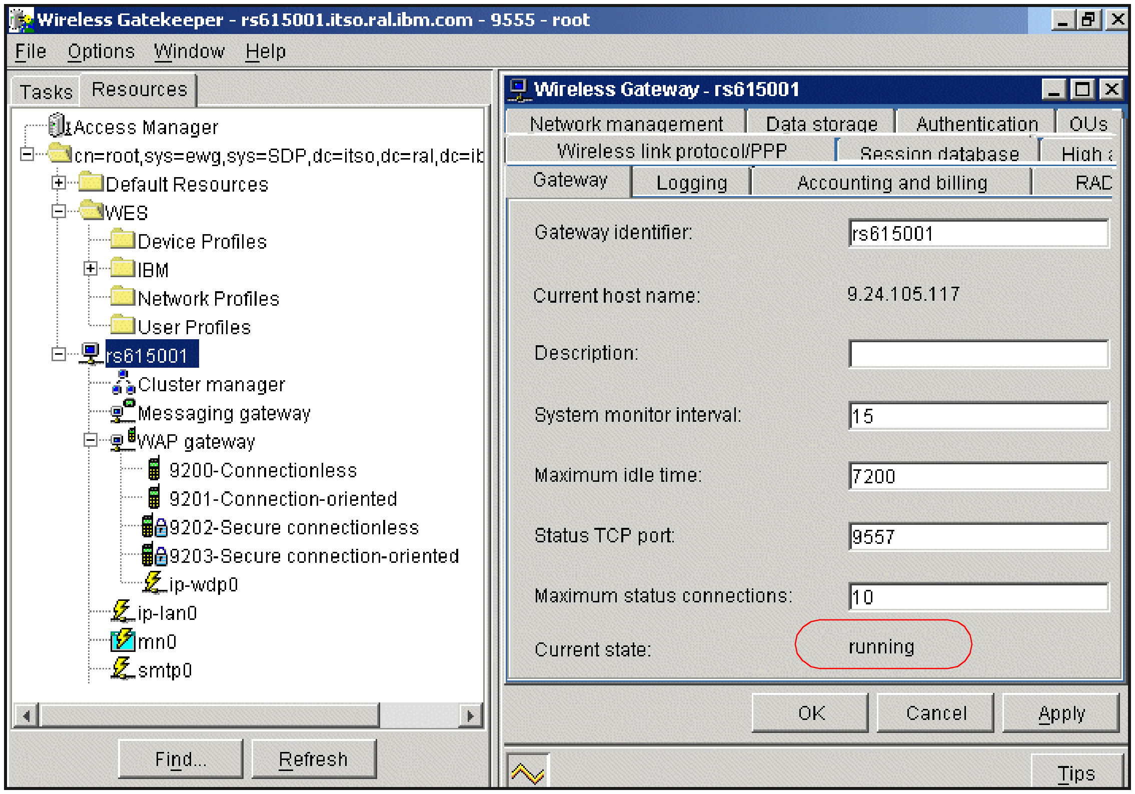
The Gatekeeper is the EWG component that allows you to configure the Wireless Gateway, register users and mobile devices, specify logging and tracing controls, and perform many other administrative tasks. The Wireless Gatekeeper interface is divided into two panes. The left pane of the interface contains two tabs, Tasks and Resources, which give you access to the information and tasks you can perform and to the resources you can manage.

The Tasks tab displays common tasks such as finding a resource, viewing logs, adding a resource, and so on.

The Gatekeeper is started by typing the following AIX command:

```
wgcfg
```

On the Resources tab, right-click the Wireless Gateway machine's name, then select **Properties**. The panel shown in Figure 8-5 appears; here you can configure EWG options and resources.



Gateway identifier: rs615001

Current host name: 9.24.105.117

Description:

System monitor interval: 15

Maximum idle time: 7200

Status TCP port: 9557

Maximum status connections: 10

Current state: running

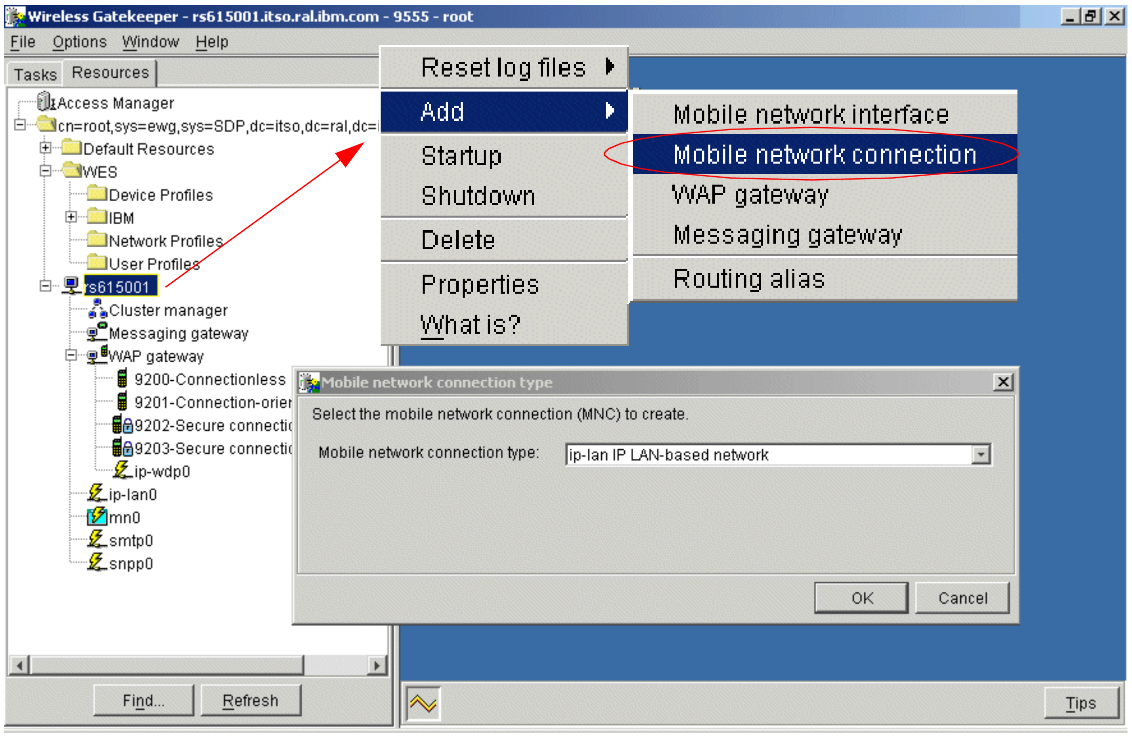
Figure 8-5 Using the Wireless Gateway Gatekeeper

Note: The Gatekeeper can also be installed on Windows workstations.

Mobile Network Connection (MNC)

A mobile network connection is a resource that is assigned to a Wireless Gateway and defines a specific type of network connection. The MNC consists of a line driver, a network protocol interpreter, and one or more physical ports. You configure one MNC for each network provider that you will use.

For example, to add a Mobile Network Connection, use the Wireless Gatekeeper and right-click the Wireless Gateway machine's name to select the option **Add a MNC**, then follow the pane's instructions. Figure 8-6 shows a sample configuration to add an IP LAN-based network.



Mobile network connection type

Select the mobile network connection (MNC) to create.

Mobile network connection type: ip-lan IP LAN-based network

OK Cancel

Figure 8-6 Adding an MNC using the Everyplace Wireless Gateway Gatekeeper

Mobile Network Interface (MNI)

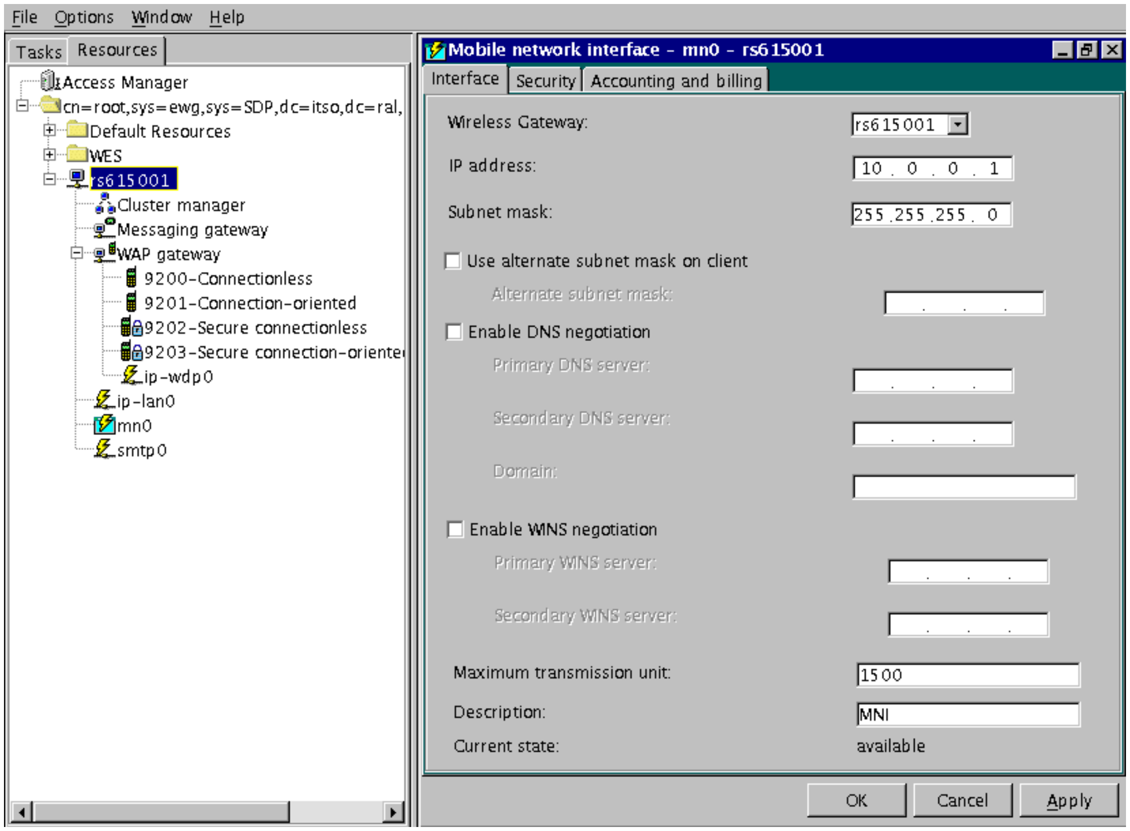
A Mobile Network Interface defines an IP subnet through which the Wireless Gateway routes IP traffic for wireless clients and for WAP clients that use a native-PPP connection. The messaging clients and WAP clients that do not use native-PPP to connect through a dial MNC do *not* use MNI resources.

A Wireless Gateway can have one MNI for all networks or multiple MNIs for different ranges of addresses. MNIs can support static addressing and DHCP (Dynamic Host Configuration Protocol) for a pool of dynamically assigned addresses.

When defining a MNI, you need to understand how to:

- Acquire a subnet using a host IP address and a subnet mask for that address.
- Make sure that reserved addresses in the subnet are not used as Wireless Client or mobile devices IP addresses (typically, the MNI reserves the first usable IP address in a subnet as its own and this address is the Wireless Gateway's point of presence on the subnet).
- Determine if you need an alternate subnet mask.
- Update your organization's routing tables to include the Wireless Gateway's IP address.
- Update your organization's routing tables to route the MNI address range to the IP address of the Wireless Gateway.
- Verify that traffic is routed securely between a Wireless Client and a Wireless Gateway.
- Optionally, set up filters, packet mappings or routing aliases for the MNI.

For example, to add a MNI, from the Wireless Gatekeeper, right-click the Wireless Gateway machine's name and select **Add a MNI**, then follow the pane's instructions.



Wireless Gateway: rs615001
IP address: 10 . 0 . 0 . 1
Subnet mask: 255 .255 .255 . 0
☐ Use alternate subnet mask on client
Alternate subnet mask:
☐ Enable DNS negotiation
Primary DNS server:
Secondary DNS server:
Domain:
☐ Enable WINS negotiation
Primary WINS server:
Secondary WINS server:
Maximum transmission unit: 1500
Description: MNI
Current state: available

Figure 8-7 Sample MNI configuration using Everyplace Wireless Gateway Gatekeeper

8.1.3 Wireless Gateway logging

Log files are very useful and allow you to check the status of wireless connections, data packets transmitted through the Wireless Gateway, etc. The wg.log file provides all this information and much more. You can view this file by typing the following AIX commands:

```
#cd /var/adm
#pg wg.log
```

Warning: If the logs are turned on in the Wireless Gatekeeper, data is provided every 15 seconds in the wg.log file. It will be necessary to remove the contents of this file when needed, in order to keep enough disk space for future traces.

In you want to remove the wg.log content, in the /var/adm directory, enter the following command:

```
#rm wg.log
```

Sample Wireless Gateway log file

```
Date: Fri Sep  7 15:21:32 2001
Server: IBM Wireless Gateway V2.1.0
Content-Type: application/xml
Content-Language: en
Content-Length: 397
<?xml version="1.0"?>
<!DOCTYPE pap PUBLIC "-//WAPFORUM//DTD PAP 1.0//EN"
         "http://www.wapforum.org/DTD/pap_1.0.dtd">
<pap product-name="IBM Wireless Gateway V2.1.0">
<push-response push-id="bAnE9GbCZj4bLTbzJBQkYg=="
sender-address="http://rs615001:13131" reply-time="2001-09-07T19:21:32Z">
<response-result code="1001" desc="Accepted for Processing"/>
</push-response>
</pap>
2954:17477 (Sep 07 01/15:21:32): ppg_handler: encoded WSP header:
2954:17477 (Sep 07 01/15:21:32):
038381ea456e636f64696e672d76657273696f6e00312e32008c99af828d8e
2954:17477 (Sep 07 01/15:21:32): push_one_msg: delivering
bAnE9GbCZj4bLTbzJBQkYg== to device addr = 9.24.106.131, remote port = 9200
2954:17477 (Sep 07 01/15:21:32): SendWAPPush: delivering packet to mnc ip-wdp0
2954:17477 (Sep 07 01/15:21:32): delivered bAnE9GbCZj4bLTbzJBQkYg== to
wappush=9.24.106.131/type=IPv4@pi.ibm.com
2954:17477 (Sep 07 01/15:21:32): update_status_info: code [1000] desc [Message
delivered] bAnE9GbCZj4bLTbzJBQkYg== wappush=9.24.106.131/type=IPv4@pi.ibm.com
2954: 1543 (Sep 07 01/15:21:32):  [BROKER Debug] api (1)->wsp (2)
H=24(localhost:9200/9.24.106.131:9200) pr=2/2 len=61
2954: 1543 (Sep 07 01/15:21:32):  [BROKER Debug] wsp (2)->wdp (6)
H=24(localhost:9200/9.24.106.131:9200) pr=5/0 len=60
2954: 1543 (Sep 07 01/15:21:32): BI-ip-wdp0: init, iostate = 12
2954: 1543 (Sep 07 01/15:21:32): BI-ip-wdp0::sendBuffer (entry)
2954: 1543 (Sep 07 01/15:21:32): ip-wdp0::deliver (entry)
2954: 1543 (Sep 07 01/15:21:32): IpWdpPort-9200::write (entry)
2954: 1543 (Sep 07 01/15:21:32): IpWdpPort-9200: delivery data to
'9.24.106.131:9200' (48)
0000:  00 06 1f 03 83 81 ea 45 6e 63 6f 64 69 6e 67 2d
0010:  76 65 72 73 69 6f 6e 00 31 2e 32 00 8c 99 af 82
0020:  8d 8e 53 61 79 20 48 69 20 74 6f 20 45 72 69 63
System Load:  0.0607 0.1373 0.1285
Active Sessions:  0  Activation Rate:  0.00
ip-wdp0:
   MNC Ports   In (pkt)  Out (pkt)    In (kb)   Out (kb) In (p/s) Out (p/s)
b/sec  Errs
       9200         5        20         0         0     0.00      0.00
0.00    0
     ip-lan0        11        11         1         0     0.00      0.00
0.00    0
       smtp0         2         0         0         0     0.00      0.00
0.00    0
WAP Server Statistics:
    Resource   In (pkt)  Out (pkt)    In (kb)   Out (kb) In (p/s) Out (p/s)
b/sec  Errs
 browse-9200         2         2         0         0     0.00      0.00
0.00    0

Port number information :
The Wireless Gateway and access manager are installed on the same system and
require a port for communication with the Wireless Gatekeeper.
The main ports used in this chapter are :
8889 : IP-LAN receive, including CDPD connections
9200 : connectionless WAP service
9201 : connection-oriented WAP service
9202 : secure connectionless WAP service
9203 : secure connection-oriented WAP service
13131 : messaging gateway
13132 : secure messaging gateway
```

8.1.4 Everyplace Wireless Gateway security

The Wireless Gateway has several ways to enforce the security of your network, your applications and their data. There are several types of security options, for example:

- Access. The process of how users at each end of a communication link know the identity of the end user is called authentication. The basic mechanism for authentication is the mutual presentation of a secret key.

  In addition to the two-party key distribution protocol used by the Wireless Gateway for wireless clients, you can also connect third-party RADIUS servers.

- Confidentiality. In most cases, the communication link may not be private, and therefore data should be encrypted before it is sent.

  Data encryption provided by the Wireless Gateway (see Figure 8-8) helps prevent unauthorized access to data by presenting data in an unintelligible form, so that the original data cannot be obtained using only a decrypting process. The data is transformed into encrypted data using the session key exchanged during the authentication process.

An SSL (Secure Sockets Layer) or WTLS (Wireless Transport Layer Security) connection with the Wireless Gateway ensures that the communication link is securely accessed, confidential and authorized.

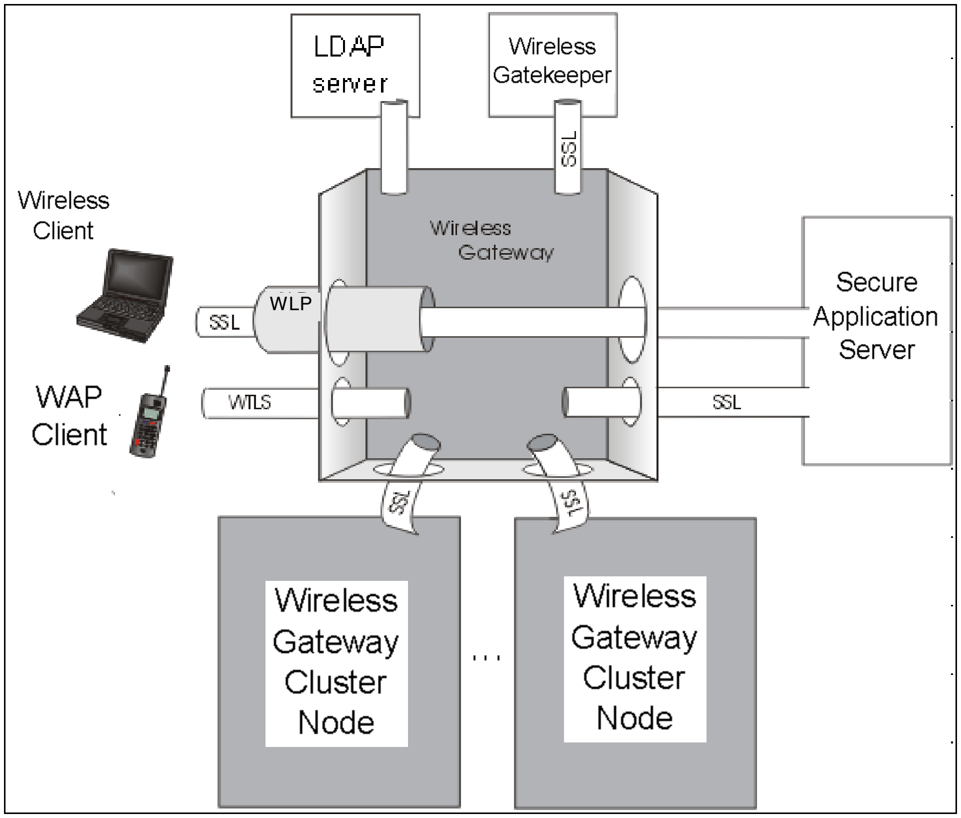


Figure 8-8 Wireless Gateway security options

Note: A managed set of public key certificates, usually issued by a Certificate Authority (CA), is required to enable SSL and WTLS communications.

- Authentication between the Wireless Gateway and Wireless Clients. Authentication is a prerequisite for the communication link between the Wireless Gateway and the Wireless Client to be encrypted. EWG uses a modified PPP (Point-to-Point Protocol) called WLP (Wireless Optimized Link Protocol) to authenticate the connection between itself and Wireless Clients. WLP uses two-party key distribution protocol in which the Wireless Gateway and Wireless Clients authenticate one another without sending a password over the air.
- Encryption between the Wireless Gateway and Wireless Clients. When you install the Wireless Client, you can choose the type of encryption used for

data transmitted between the Wireless Client and the Wireless Gateway. For example:

  – DES (Digital Encryption Standard)
  – RC5
  – Triple-DES

- Third-party authentication to the Wireless Gateway. You can also configure a Wireless Gateway to use RADIUS third-party authentication. Acting on behalf of WAP or Wireless Clients connected to it, the Wireless Gateway routes authentication requests to a RADIUS server. Because these authentication requests are in addition to WLP (Wireless Optimized Link Protocol), request and response times to authenticate clients increase when you configure the Wireless Gateway to use RADIUS authentication.
- Authorization. Authorization is the assurance that the user is authenticated and permitted access to the data. In addition, you can also configure the WAP Gateway to use WebSEAL server, a component of Tivoli SecureWay Policy Director which is now a part of WebSphere Everyplace Server, as its HTTP proxy. Policy Director provides access control management to centralize network and application security policies. WebSEAL manages access to all your Web servers. In a WebSphere Everyplace Server environment, access control management using Policy Director is handled by WebSEAL-Lite.

8.2 WAP Gateway

The Everyplace Wireless Gateway (EWG) can be configured to include a WAP Gateway. EWG then performs a protocol conversion to provide communication with WAP clients and HTTP Web servers. The WAP Gateway listens for data and messages arriving from WAP clients and translates the Wireless Session Protocol (WSP) requests into HTTP requests that are forwarded to an HTTP proxy. The response headers are converted from HTTP into WSP response headers; the content is encoded from WML and any WMLScript into binary XML (wbxml) and forwarded to the WAP client.

Figure 8-9 shows mobile devices that are WAP-compliant and connecting through bearer networks to the Wireless Gateway. The Wireless Gateway and servers shown on the right side in this figure enable secure access to the Web and enterprise applications.

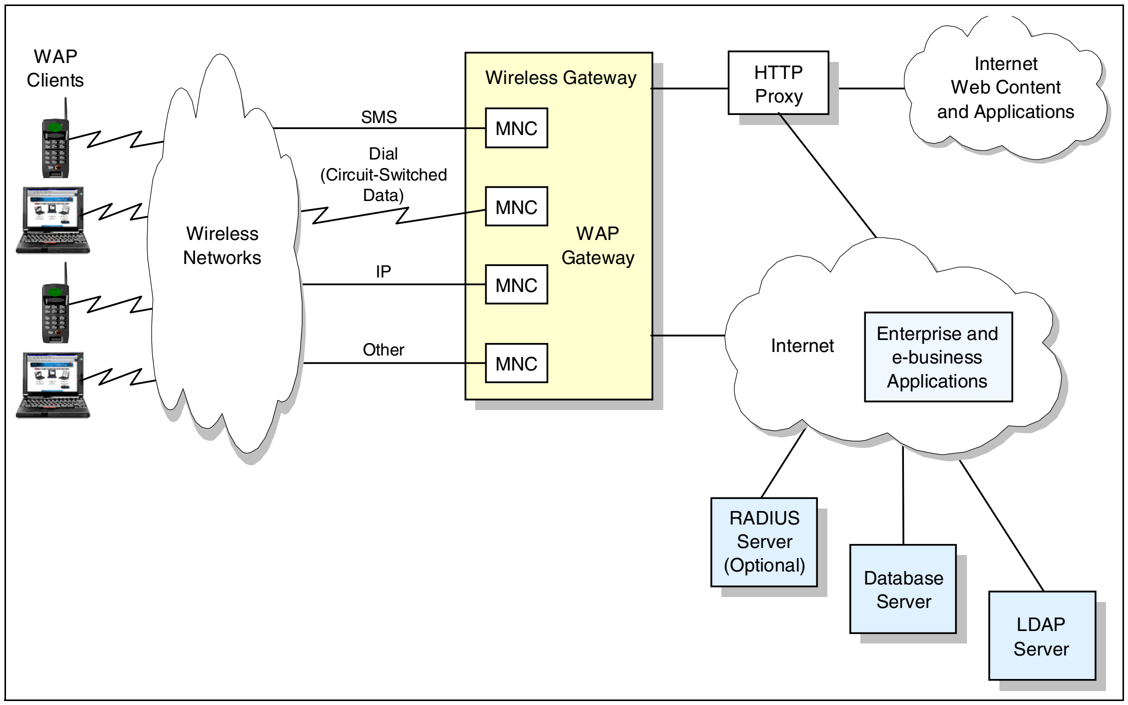


Figure 8-9 Wireless Gateway configured to include a WAP Gateway

The WAP Gateway provides the following functions:

- WAP Version 1.2.1 compliant protocol stack
- High scalability with distributed architecture:
  - Uses clusters of gateways to balance load and improve availability
  - Dynamically adds and removes machines from the cluster
  - Able to support large carrier-grade installations and corporate enterprise environments
- Administration of distributed gateways:
  - Uses the Wireless Gatekeeper
  - Enables centralized or distributed remote administration
- LDAP server support for administration and configuration data
- Functions as a WAP Push Proxy Gateway
- WAP Proxy for user cookies. WAP phones do not store HTTP cookies, so the Wireless Gateway will store these cookies on behalf of the WAP client and deliver them to the Web servers when they are requested.

- Accounting records are stored in an ODBC relational database (DB2 or Oracle).

The WAP Gateway security support includes:

- Wireless Transport Layer Security (WTLS) that secures the connection from the Gateway to the WAP client.
- Secure Sockets Layer (SSL) used to establish a secure connection from the Gateway to back-end Web servers.
- WAP user authentication:
  - Optional RADIUS protocol-based authentication of WAP clients.
  - WAP Gateway authentication for users in LDAP directory.

The Wireless Gateway allows you to optimize WAP applications:

- Binary transmission is used to compress data and therefore increase the effective data rate and lower transmission costs.
- Unnecessary protocol headers are eliminated to lower transmission costs.
- The number of messages sent between the device and server is optimized to lower transmission costs.
- It can dynamically disconnect-reconnect to lower connection fees.

WAP request flow

Figure 8-10 illustrates a typical WAP request flow.

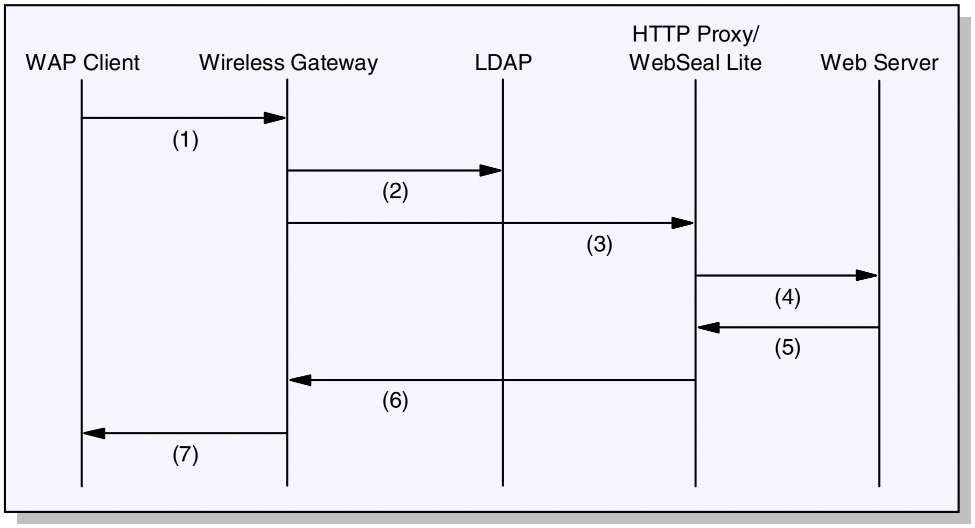


Figure 8-10 WAP request flow through EWG and WebSphere Everyplace Server components

1. The WAP Client makes a request for a Web page. EWG receives the request and checks its configuration settings and internal device cache to see if this user or device has been validated.
2. If WAP challenge authentication is enabled, the Gateway validates the user ID and password with LDAP and, optionally, a RADIUS authentication server.
3. The validated request is converted from binary WAP language to standard Web protocols. Additional HTTP request headers are added (such as cookies, WebSphere Everyplace Server integration headers, etc.). The request is forwarded to an HTTP proxy.
4. The HTTP proxy handles the request, typically forwarding it on to the content server.
5. The response is returned from the content server.
6. The HTTP proxy returns the response to the Gateway. The Gateway processes any cookies that are present and converts WML documents to binary WML. HTTP headers are encoded to binary WSP headers.
7. The response is sent back to the WAP client.

8.2.1 Configuration

Using the Wireless Gateway as a WAP Gateway requires some configuration tasks. The resources are shown on the Resources tab within the hierarchy of organizational units.

This is how you add a WAP Gateway to the Wireless Gateway:

1. Open the Gatekeeper by typing the following AIX command:

   ```
   wgcfg
   ```

2. On the Resources tab, right-click the Wireless Gateway machine’s name to which you want to add a WAP Gateway. Then click **Add** and select **WAP Gateway**.

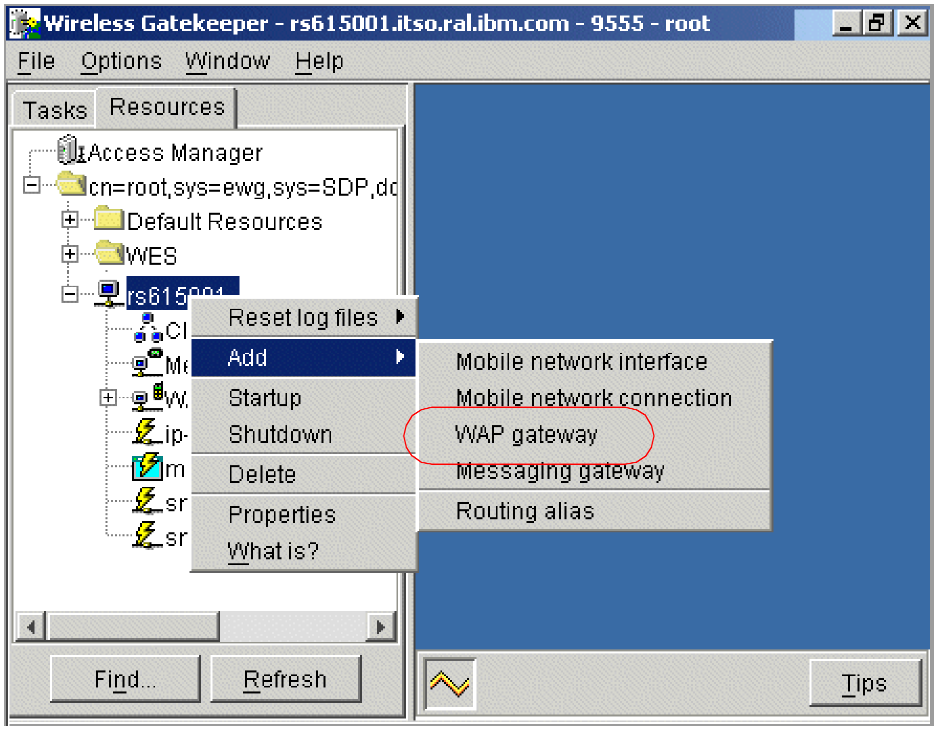
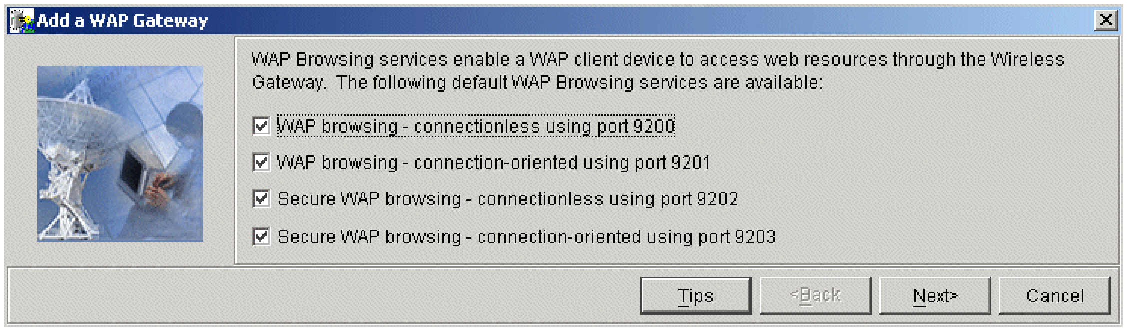


Figure 8-11 Adding a WAP Gateway

3. As illustrated in Figure 8-12, choose your WAP browsing service. The WAP browsing service allows WAP clients to access Web content. There are four options: Connectionless, Connection-oriented, Secure connectionless and Secure connection-oriented.


Add a WAP Gateway

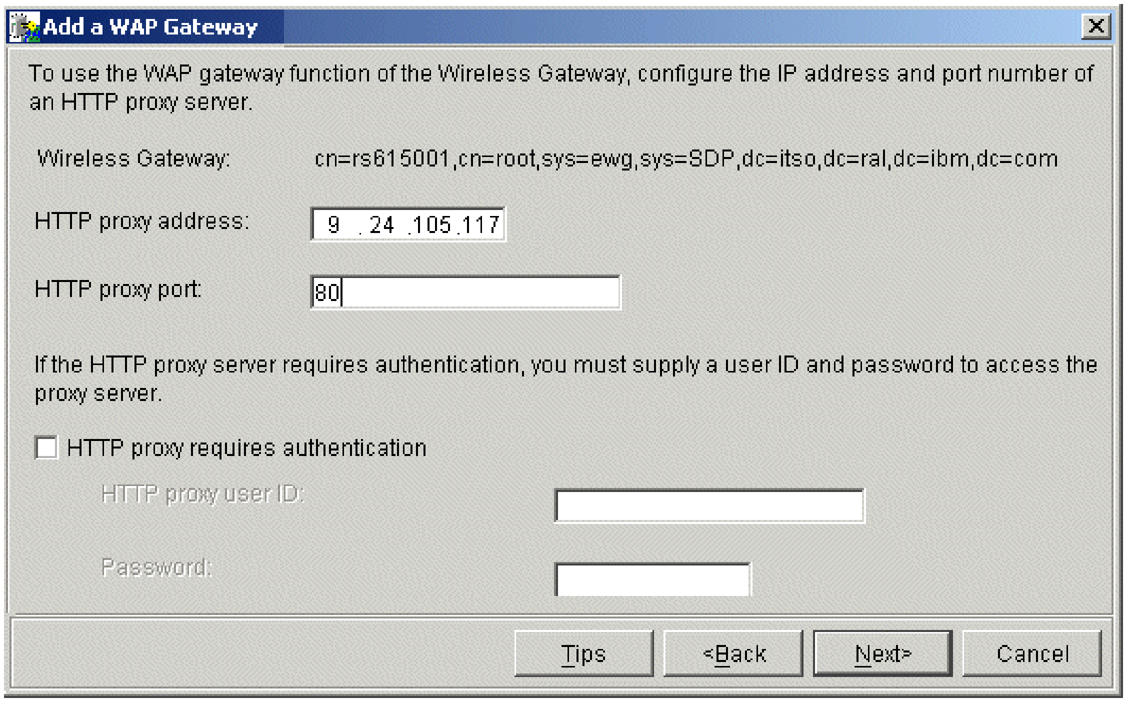
WAP Browsing services enable a WAP client device to access web resources through the Wireless Gateway. The following default WAP Browsing services are available:

- [✓] WAP browsing - connectionless using port 9200
- [✓] WAP browsing - connection-oriented using port 9201
- [✓] Secure WAP browsing - connectionless using port 9202
- [✓] Secure WAP browsing - connection-oriented using port 9203

Figure 8-12 WAP browser service configuration

4. Configure the IP address and port number of an HTTP proxy server (see Figure 8-13). You must provide this address even if you plan to connect to a reverse proxy. You will also need to configure the HTTP proxy port; the default value is port 80.

   Note: A reverse proxy is configured using the Gatekeeper.



To use the WAP gateway function of the Wireless Gateway, configure the IP address and port number of an HTTP proxy server.

Wireless Gateway: cn=rs615001,cn=root,sys=ewg,sys=SDP,dc=itso,dc=ral,dc=ibm,dc=com

HTTP proxy address: 9 .24 .105.117

HTTP proxy port: 80

If the HTTP proxy server requires authentication, you must supply a user ID and password to access the proxy server.

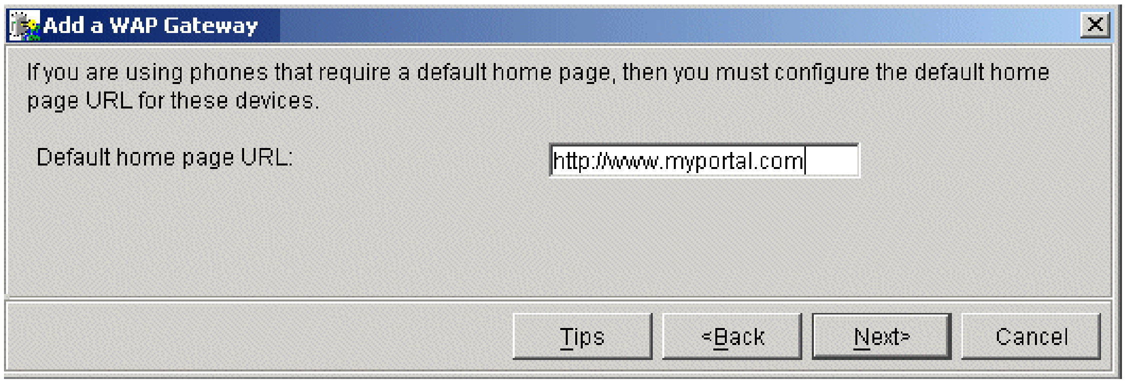
☐ HTTP proxy requires authentication

HTTP proxy user ID:

Password:

Figure 8-13 HTTP proxy information.

5. You can configure the default home page URL for your WAP phone (see Figure 8-14). Enter the URL, for example `http://www.myportal.com`, or you can leave the field blank.

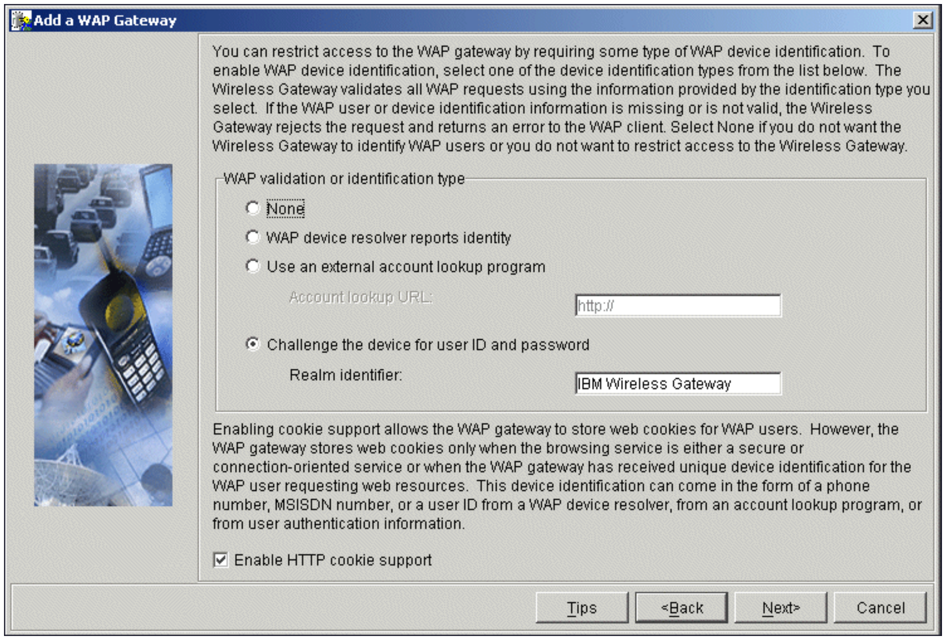


If you are using phones that require a default home page, then you must configure the default home page URL for these devices.

Default home page URL: http://www.myportal.com

Figure 8-14 Home page URL information for the WAP phone

6. If required, configure the WAP Gateway with WAP device identification (see Figure 8-15).



You can restrict access to the WAP gateway by requiring some type of WAP device identification. To enable WAP device identification, select one of the device identification types from the list below. The Wireless Gateway validates all WAP requests using the information provided by the identification type you select. If the WAP user or device identification information is missing or is not valid, the Wireless Gateway rejects the request and returns an error to the WAP client. Select None if you do not want the Wireless Gateway to identify WAP users or you do not want to restrict access to the Wireless Gateway.

WAP validation or identification type

- None
- WAP device resolver reports identity
- Use an external account lookup program
  - Account lookup URL: http://
- [✓ Challenge the device for user ID and password]
  - Realm identifier: IBM Wireless Gateway

Enabling cookie support allows the WAP gateway to store web cookies for WAP users. However, the WAP gateway stores web cookies only when the browsing service is either a secure or connection-oriented service or when the WAP gateway has received unique device identification for the WAP user requesting web resources. This device identification can come in the form of a phone number, MSISDN number, or a user ID from a WAP device resolver, from an account lookup program, or from user authentication information.

☑ Enable HTTP cookie support

Figure 8-15 User identification information and cookie support

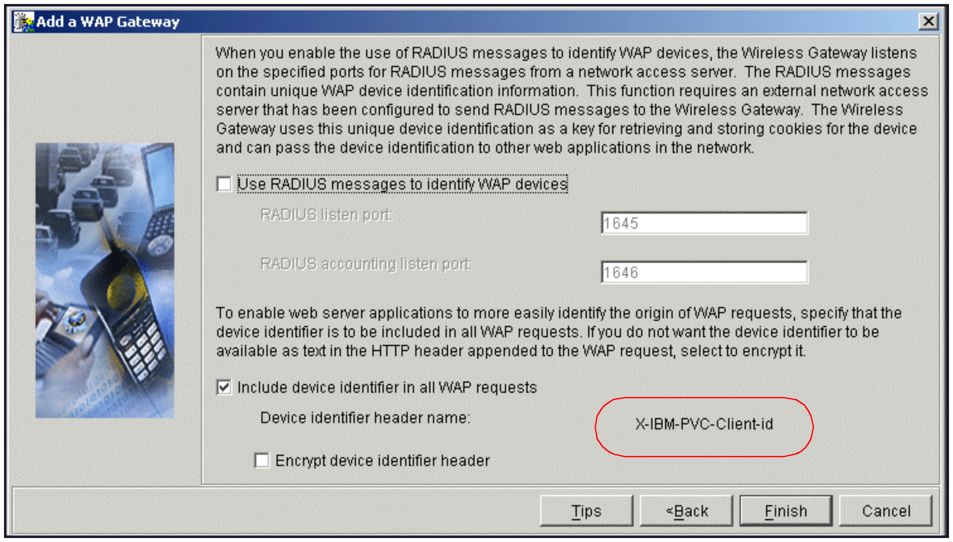
WAP cookie proxy

As shown in Figure 8-15, the enable HTTP cookie support checkbox is selected by default.

- The Wireless Gateway can proxy both session and persistent cookies for a WAP user.
- Session cookies only exist for the time that the user is connected to the WAP Gateway. Persistent cookies have an expiration date and can exist for a long time if kept in permanent storage.
- Persistent cookies are only stored if the WAP user has been identified.
- Persistent cookies are stored in DB2 (or Oracle) database.
- In version EWG Version 2.1.1, session cookies can be cached for unidentified users if they are using connection-oriented or secure devices.

RADIUS and device identifier

As shown in Figure 8-16, RADIUS messages can be used to identify WAP devices. By default, the WAP Gateway adds a device identifier in the HTTP headers from WAP requests.



When you enable the use of RADIUS messages to identify WAP devices, the Wireless Gateway listens on the specified ports for RADIUS messages from a network access server. The RADIUS messages contain unique WAP device identification information. This function requires an external network access server that has been configured to send RADIUS messages to the Wireless Gateway. The Wireless Gateway uses this unique device identification as a key for retrieving and storing cookies for the device and can pass the device identification to other web applications in the network.

☐ Use RADIUS messages to identify WAP devices

RADIUS listen port: 1645

RADIUS accounting listen port: 1646

To enable web server applications to more easily identify the origin of WAP requests, specify that the device identifier is to be included in all WAP requests. If you do not want the device identifier to be available as text in the HTTP header appended to the WAP request, select to encrypt it.

☑ Include device identifier in all WAP requests

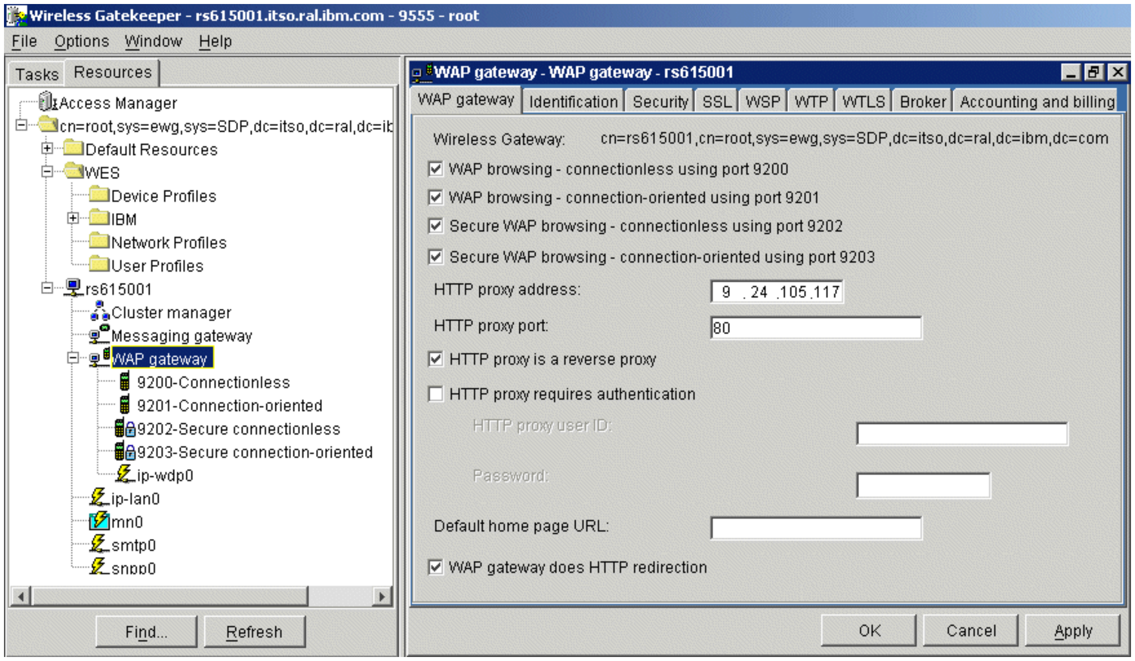
Device identifier header name: X-IBM-PVC-Client-id

☐ Encrypt device identifier header

Figure 8-16 RADIUS optional information

Post-configuration options

Use the Gatekeeper to add or change WAP Gateway configuration options. This is done by right-clicking **WAP Gateway** and selecting **Properties**. For example, configure the option to indicate that the WAP Gateway will be connected to a reverse proxy.


Wireless Gatekeeper - rs615001.itso.ral.ibm.com - 9555 - root
File Options Window Help
Tasks Resources
Access Manager
cn=root,sys=ewg,sys=SDP,dc=itso,dc=ral,dc=ib
Default Resources
WES
Device Profiles
IBM
Network Profiles
User Profiles
rs615001
Cluster manager
Messaging gateway
WAP gateway
9200-Connectionless
9201-Connection-oriented
9202-Secure connectionless
9203-Secure connection-oriented
ip-wdp0
ip-lan0
mn0
smtp0
snpp0
Find...
Refresh
WAP gateway - WAP gateway - rs615001
WAP gateway Identification Security SSL WSP WTP WTLS Broker Accounting and billing
OK Cancel Apply

Wireless Gateway: cn=rs615001,cn=root,sys=ewg,sys=SDP,dc=itso,dc=ral,dc=ibm,dc=com

- [x] WAP browsing - connectionless using port 9200
- [x] WAP browsing - connection-oriented using port 9201
- [x] Secure WAP browsing - connectionless using port 9202
- [x] Secure WAP browsing - connection-oriented using port 9203

HTTP proxy address: 9 . 24 .105.117

HTTP proxy port: 80

- [x] HTTP proxy is a reverse proxy
- [ ] HTTP proxy requires authentication

HTTP proxy user ID:

Password:

Default home page URL:

- [x] WAP gateway does HTTP redirection

Figure 8-17 WAP Gateway properties

For more information about WAP Gateway configuration and WML fragmentation issues, see Chapter 7, “WML fragmentation considerations” on page 259.

8.2.2 WAP device resolver

The WAP device resolver resource on the WAP Gateway works in conjunction with a Network Access Server (NAS) to uniquely identify WAP devices whenever the devices connect to the network. The unique identity of WAP devices is required before the Wireless Gateway can pass the device identity in WAP requests to other Web servers and proxies in the network. When you enable the WAP device resolver on the WAP Gateway, this unique identifier is passed upon request to the WAP Gateway in RADIUS authentication or RADIUS accounting messages from the NAS.

WAP device resolver request flow

A WAP client connects to a NAS which sends a RADIUS packet to the WAP Gateway. The packet includes the IP address of the WAP client and a unique identifier. The WAP client requests a Web page using WAP protocol. The WAP Gateway converts the request to HTTP, appends the identifier, sends the request to the destination and optionally sends the original request to a RADIUS server.

The type of RADIUS messages that are sent from the NAS (authentication or accounting) depends on the NAS configuration and whether any other authentication or accounting servers exist in the network. You can configure the WAP Gateway to return RADIUS responses directly back to the NAS, or as a proxy whereby the WAP Gateway forwards RADIUS messages to other servers in the network, then returns the subsequent responses to the NAS.

When you define the WAP device resolver resource on the WAP Gateway, you specify which RADIUS attribute type (this is the unique identifier sent by the NAS) you want to use to uniquely identify the WAP device. The identifier must be unique for each device and must be the same each time a particular device connects to the network.

Users who are identified by the WAP device resolver do not display as active users in the Wireless Gatekeeper.

8.2.3 Sample scenario: accessing the WAP Gateway

In a typical WAP application development environment, you will use a WAP simulator using the HTTP protocol to connect to other servers. However, at some point you will need to also run your new applications in a WebSphere Everyplace Server environment for proper integration using other WebSphere Everyplace Server components.

In this section, we show you how to access the B2E YourCo application (used throughout this redbook) from a WAP simulator using WAP/IP protocols. In this scenario, the WAP Gateway is configured to use a proxy using WTE and WebSeal-Lite (WSL) for user authentication.

The sample scenario is illustrated in Figure 8-18.

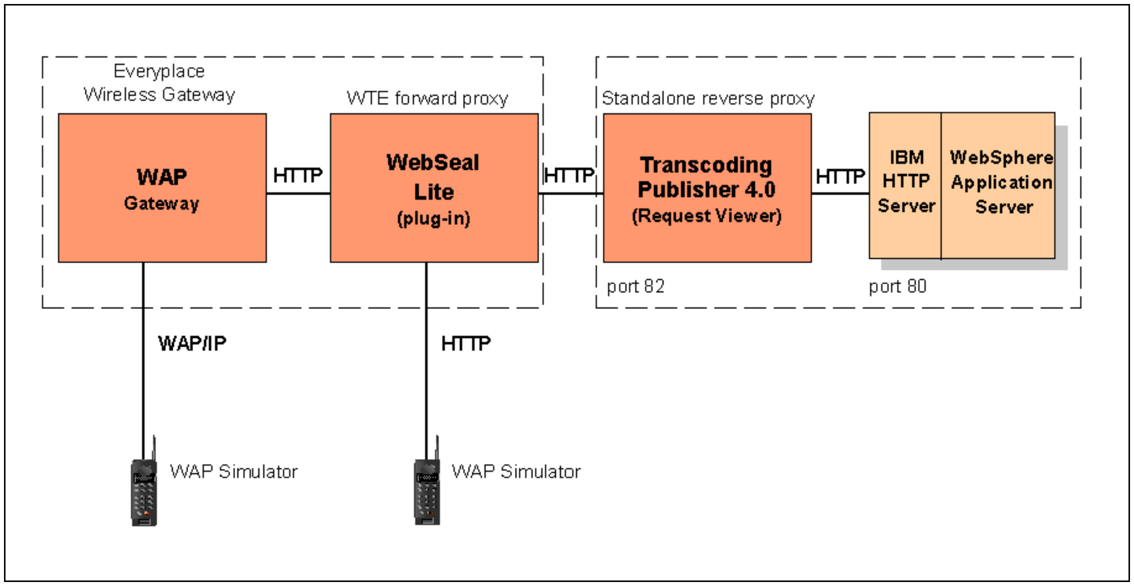
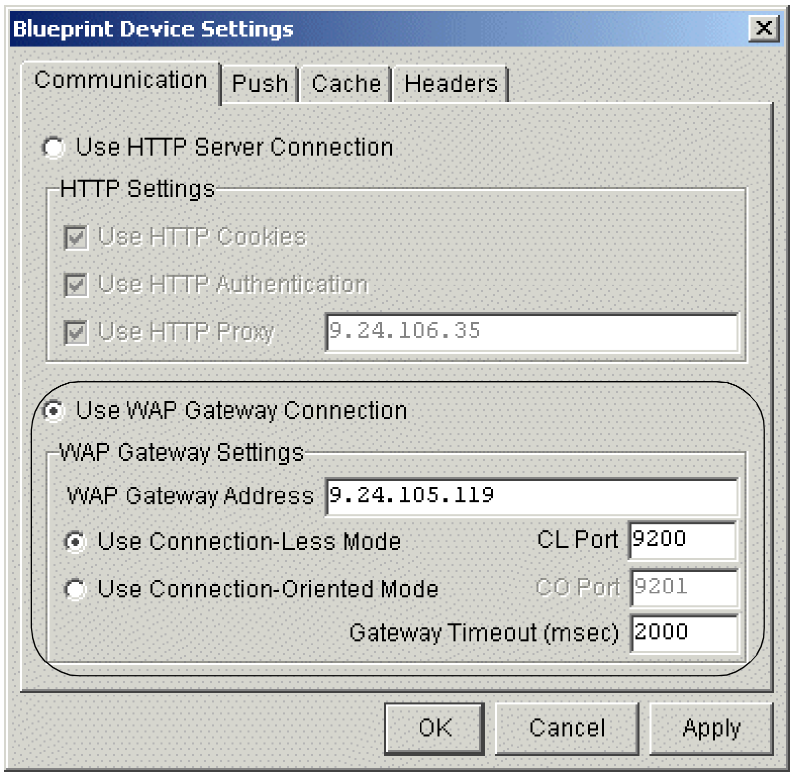


Figure 8-18 Sample scenario - accessing the WAP Gateway

For example, as shown in Figure 8-19, configure the Nokia WAP simulator to use a connectionless mode link to the WAP Gateway and using CL Port 9200. In this scenario, the connection uses WAP over IP to the WAP Gateway.

For this scenario, we recommend the following:

- For initial WML application development you will probably want to connect your WAP simulator directly to the WebSeal-Lite (WSL) machine using the HTTP protocol. In other words, we recommend that you connect to the WAP Gateway once your application has been thoroughly tested.
- Decide and understand how the WTE/WSL should run in the WebSphere Everyplace Server environment. For enterprise applications, you will most likely configure this machine as a reverse proxy.
- Since you will be using a proxy in this scenario (WTE/WSL), you will need to configure real host names or IP addresses instead of using localhost or 127.0.0.1 addresses in your application URLs and anchors.
- Review Chapter 7, “WML fragmentation considerations” on page 259 for issues regarding WAP Gateway, WTE/WSL and Transcoding to support WML fragmentation.



Blueprint Device Settings

( ) Use HTTP Server Connection

HTTP Settings

[✓] Use HTTP Cookies

[✓] Use HTTP Authentication

[✓] Use HTTP Proxy 9.24.106.35

(●) Use WAP Gateway Connection

WAP Gateway Settings

WAP Gateway Address 9.24.105.119

(●) Use Connection-Less Mode CL Port 9200

( ) Use Connection-Oriented Mode CO Port 9201

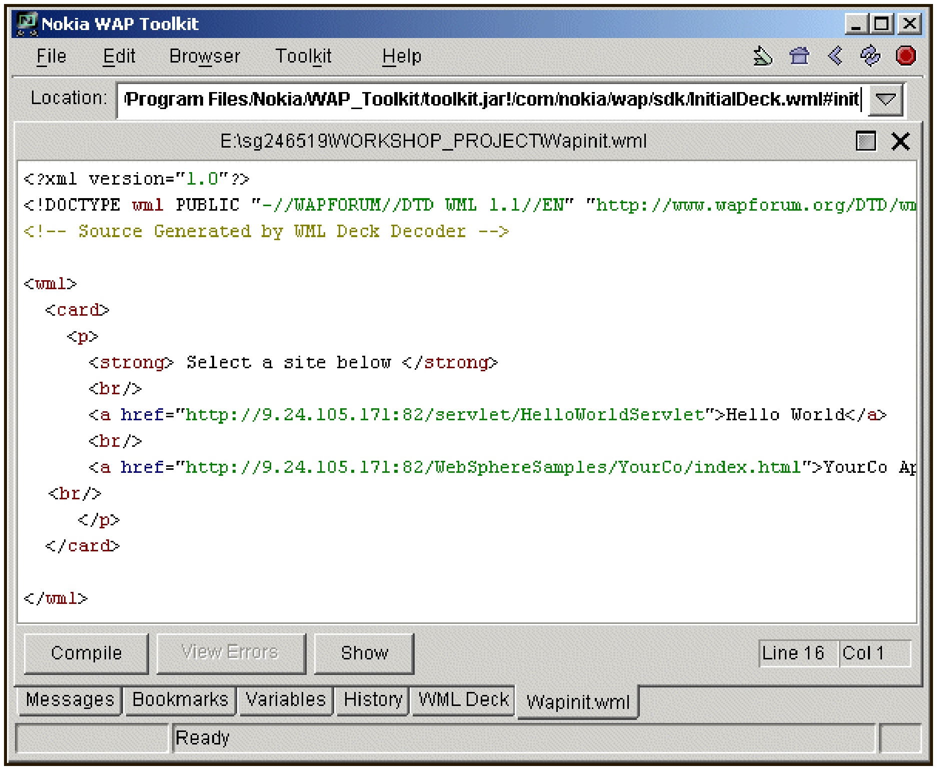
Gateway Timeout (msec) 2000

Figure 8-19 Nokia WAP simulator configuration for WAP/IP access

Figure 8-20 shows a sample WML initial page that can be used to access enterprise applications using the WAP Gateway via WAP/IP.

If you configure the option indicating that WTE/WSL is a reverse proxy, once the request reaches the WAP Gateway, the host name or IP address and port in the URL will be replaced with the address and port you configure in the HTTP proxy of the WAP Gateway.

Note: For more details about how you configure the WAP Gateway when WSL is a forward proxy or a reverse proxy, see also Chapter 7, “WML fragmentation considerations” on page 259.


Nokia WAP Toolkit
File Edit Browser Toolkit Help
Location: Program Files/Nokia/WAP_Toolkit/toolkit.jar!/com/nokia/wap/sdk/InitialDeck.wml#init
E:\sg246519\WORKSHOP_PROJECT\Wapinit.wml
Compile
View Errors
Show
Line 16
Col 1
Messages
Bookmarks
Variables
History
WML Deck
Wapinit.wml
Ready

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wm
<!-- Source Generated by WML Deck Decoder -->

<wml>
  <card>
    <p>
      <strong> Select a site below </strong>
      <br/>
      <a href="http://9.24.105.171:82/servlet/HelloWorldServlet">Hello World</a>
      <br/>
      <a href="http://9.24.105.171:82/WebSphereSamples/YourCo/index.html">YourCo Ap
  <br/>
     </p>
  </card>

</wml>
```

Figure 8-20 WAP initial page

8.3 Messaging gateway

The messaging gateway is the Wireless Gateway component that provides messaging support. By using the APIs provided with the Messaging Services and Push Toolkit, you can create applications that use the messaging gateway to send short messages to and receive short messages from a variety of mobile devices. The following clients are supported:

- WAP phones
- GSM-SMS mobile phones
- SMTP e-mail clients
- SNPP (Simple Network Pager Protocol) supported pagers and phones
- Mobitex client devices

The Messaging Gateway provides support for the following functions:

- Sending WAP push messages
- Sending non-WAP push messages
- Receiving mobile-originated messages
- Canceling push messages
- Querying the status of push messages
- Over-the-air provisioning

For more details about these functions, see also Chapter 9, “Push messaging applications” on page 323.

Short message delivery operation

A short message delivery operation starts when a message processing application or servlet uses the Wireless Gateway messaging services and Push APIs to send a message to the messaging gateway. The messaging gateway forwards the message to a short message service center (SMS-C), an SMTP server, or other network server for subsequent delivery to a client.

Figure 8-21 illustrates mobile devices receiving messages from the messaging gateway, which in turn received the information from a messaging processing application.

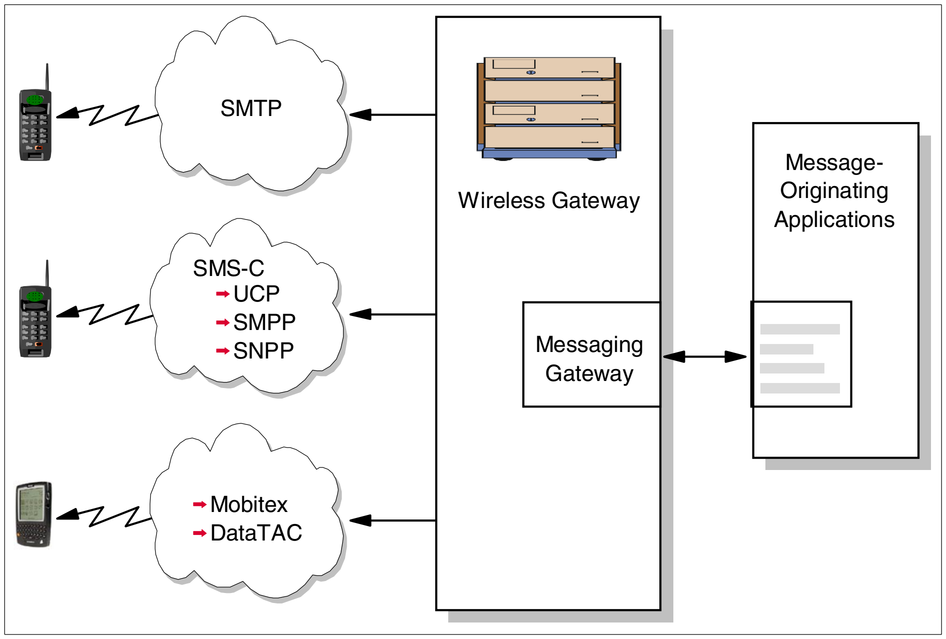


Figure 8-21 Wireless Gateway configured as a messaging gateway

WAP Push

A WAP push operation starts when a Push Initiator (PI) transmits content to the Push Proxy Gateway (PPG) using the WAP Push Access Protocol (PAP) for subsequent delivery to a client. PAP messages, which carry control information, content, and optionally, client capabilities information, are exchanged between the PI and the PPG. PAP messages are created using EWG messaging services and Push APIs.

Note: Push Proxy Gateway services in Everyplace Wireless Gateway are provided by the messaging gateway.

Figure 8-22 illustrates mobile devices receiving WAP push messages from the PPG, which in turn received the information from a push initiator application.

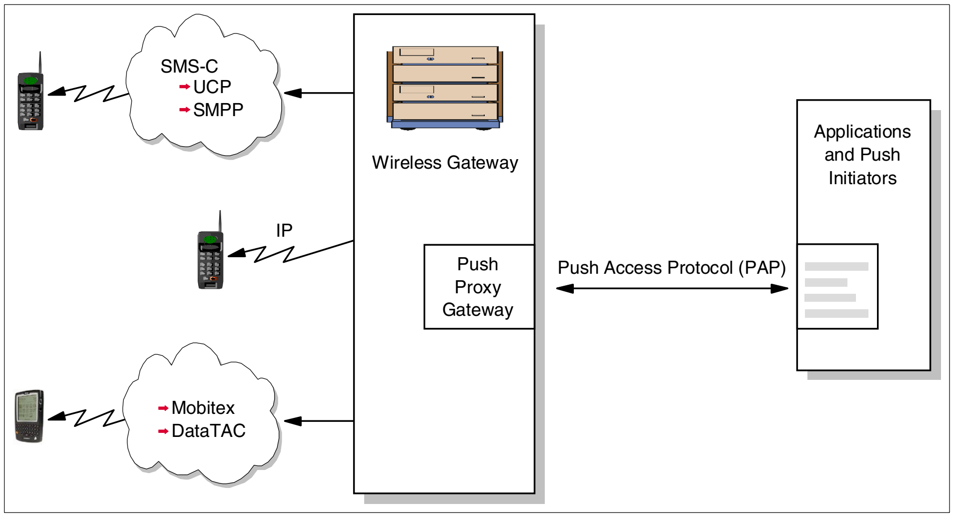


Figure 8-22 Wireless Gateway configured as a WAP push proxy gateway

Mobile-originated message operation

A mobile-originated message operation starts when a client sends a message to a network provider, for delivery to the messaging gateway. The messaging gateway uses an HTTP post operation to forward the message to an application or servlet which uses EWG and Push APIs.

Figure 8-23 illustrates mobile device sending messages to the messaging gateway, which in turn forwards the information to a message processing application or servlet.

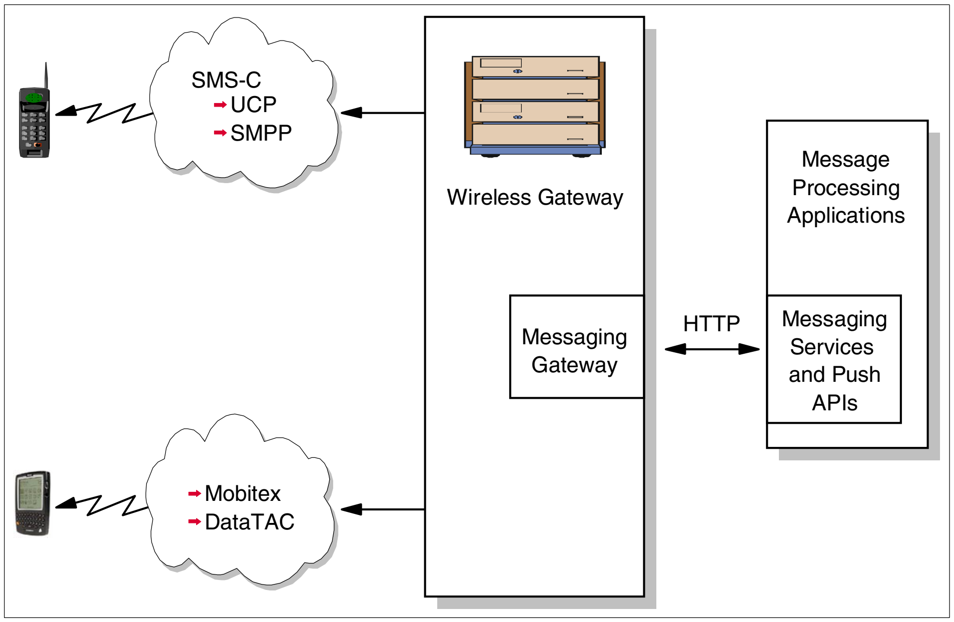

SMS-C
UCP
SMPP
Wireless Gateway
Message Processing Applications
Messaging Gateway
HTTP
Messaging Services and Push APIs
Mobitex
DataTAC

Figure 8-23 Messaging gateway accepting mobile-originated messages

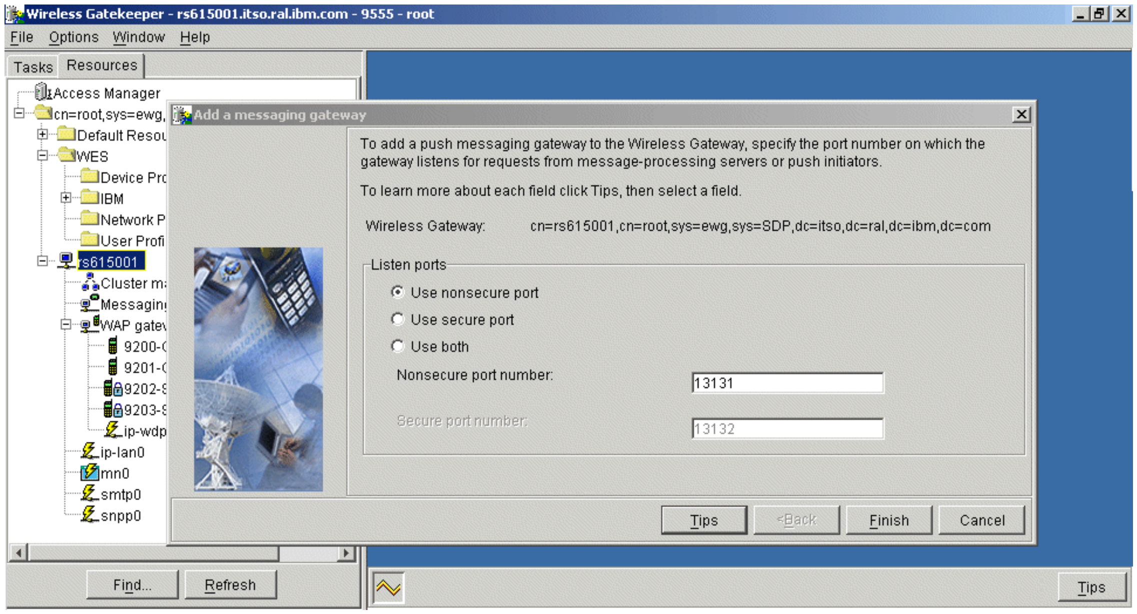
8.3.1 Configuration

In order to add a messaging gateway to the Wireless Gateway, execute the following steps:

- Open the Gatekeeper by typing the following AIX command:

  ```
  wgcfg
  ```

- Under the Resources tab, right-click the Wireless Gateway machine's name to which you want to add a messaging gateway. Then click **Add** and select **Messaging Gateway**.
- Specify the listening ports and choose a secure and/or non-secure port. The Wireless Gateway provides the port number on which the gateway listens for requests from applications using the messaging services and Push APIs. As shown in Figure 8-24, the default ports are as follows:
  – Nonsecure port: 13131
  – Secure port: 13132



To add a push messaging gateway to the Wireless Gateway, specify the port number on which the gateway listens for requests from message-processing servers or push initiators.

To learn more about each field click Tips, then select a field.

Wireless Gateway: cn=rs615001,cn=root,sys=ewg,sys=SDP,dc=itso,dc=ral,dc=ibm,dc=com

Listen ports

[✓ Use nonsecure port]
Use secure port
Use both

Nonsecure port number: 13131

Secure port number: 13132

Figure 8-24 Adding a messaging gateway

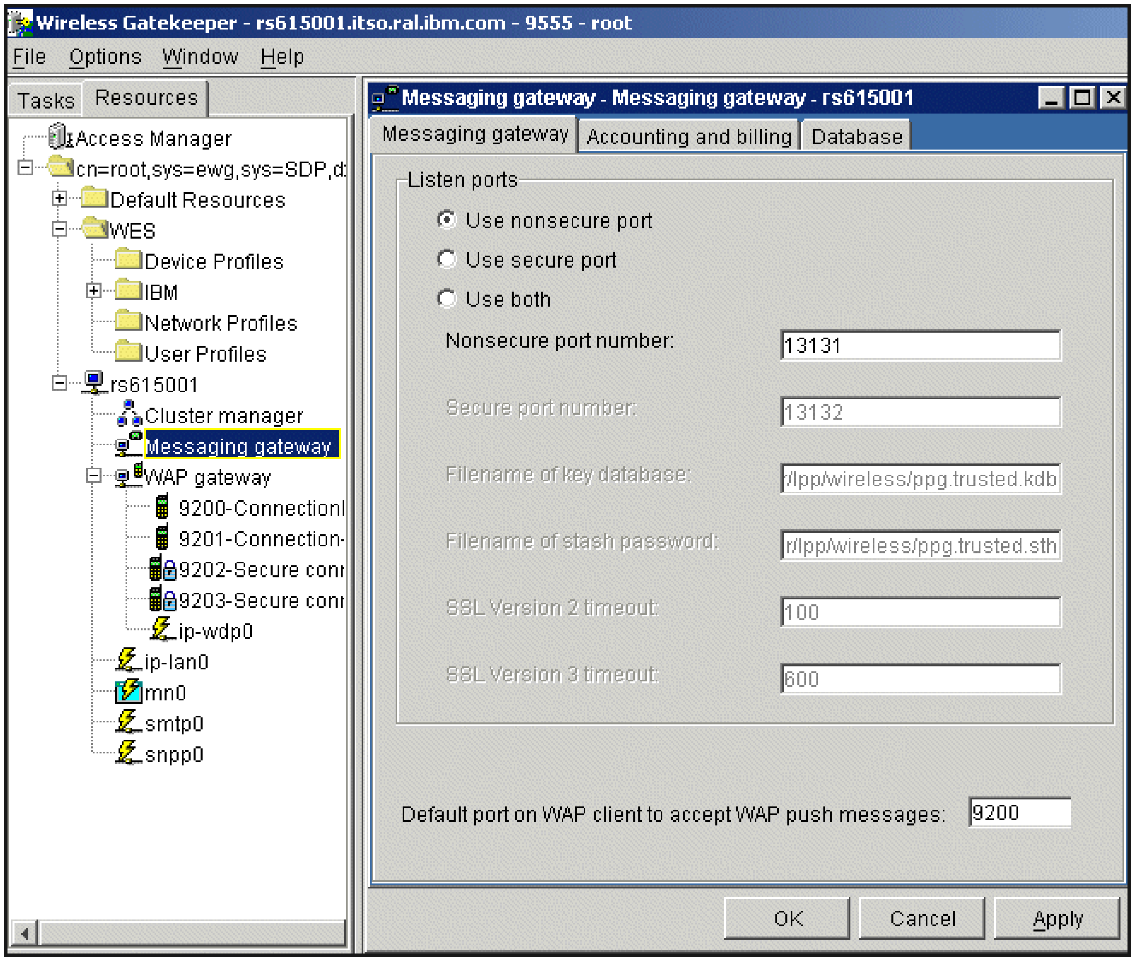
Post-configuration options

Use the Gatekeeper to add or change Messaging Gateway configuration options. This is done by right-clicking **Messaging Gateway** and selecting **Properties**. You can, for example, configure the option to direct the default port on WAP clients to accept push messages.

When you select the option to use a secure port, it means that Secure Sockets Layer (SSL) will be used between the application and the Messaging Gateway; therefore, the application must use a certificate that will be received by the Messaging Gateway in order to authenticate the sender (application). For example, if the application is a servlet running in a WebSphere Application Server environment, the Web server must be configured for SSL and must provide an X.509 certificate. In this case, the servlet is the Push Initiator (PI).

Selecting a secure port for SSL requires that you also provide extra information required, such as a secure port number (the default value is 13132), key database name, database password location and SSL version.

A sample Messaging Gateway configuration using a nonsecure port is shown in Figure 8-25.



Listen ports

(●) Use nonsecure port
( ) Use secure port
( ) Use both

Nonsecure port number: 13131

Secure port number: 13132

Filename of key database: /lpp/wireless/ppg.trusted.kdb

Filename of stash password: r/lpp/wireless/ppg.trusted.sth

SSL Version 2 timeout: 100

SSL Version 3 timeout: 600

Default port on WAP client to accept WAP push messages: 9200

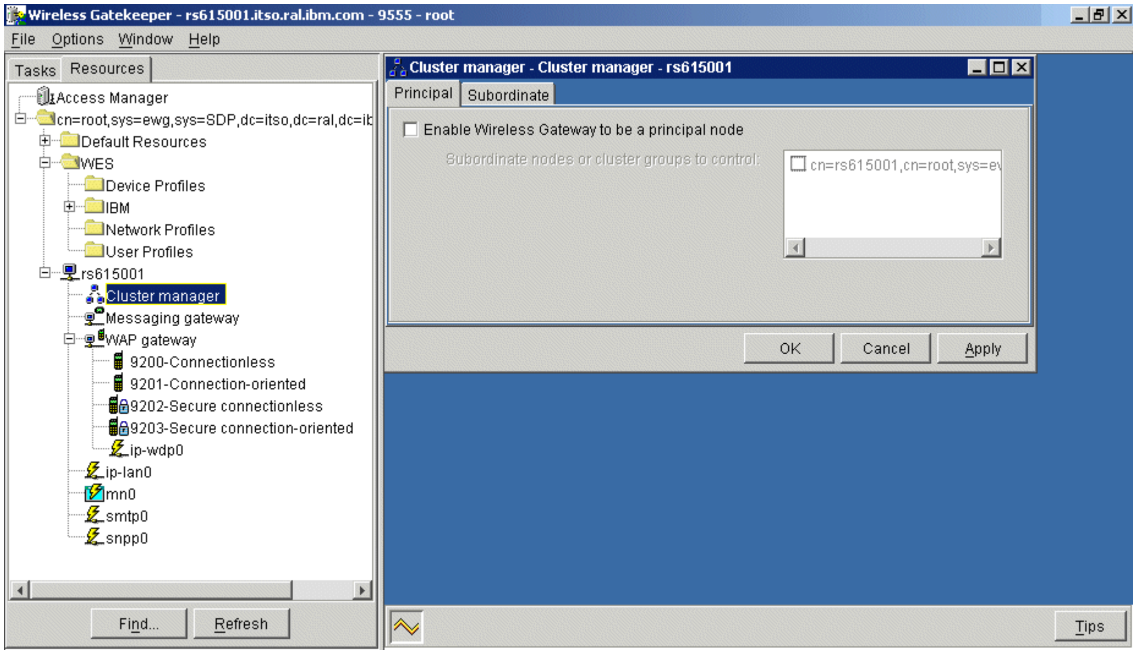
Figure 8-25 Messaging Gateway settings

8.4 Multiple Wireless Gateways cluster

Everyplace Wireless Gateway supports clustering. Multiple Wireless Gateways can be configured as a cluster to distribute the workload in a multi-node configuration. A cluster manager is a resource that performs dynamic load balancing and improves availability among Wireless Gateways.

Note: EWG clustering is beyond the scope of this redbook; if you need details about this function, please refer to the EWG documentation provided with the product.

As illustrated in Figure 8-26, you can configure the Wireless Gateway to be a principal node or a subordinate node with the Cluster manager provided by the Wireless Gatekeeper.



☐ Enable Wireless Gateway to be a principal node

Subordinate nodes or cluster groups to control: ☐ cn=rs615001,cn=root,sys=e\

Figure 8-26 Cluster manager

8.5 Wireless clients

The Wireless Client enables mobile computers to establish a secure connection to a corporate network using numerous wired and wireless bearer networks, and run TCP/IP applications over the connection, taking advantage of the data optimizations provided by the Wireless Client. The Wireless Client is middleware located below the TCP/IP stack.

Note: Wireless clients connect to the Everyplace Wireless Gateway using the Wireless Link Protocol (WLP).

The supported networks are:

- CDPD and IP-based
- Dataradio
- DataTAC and Private Mobile Radio

- Mobitex
- Norcom Satellite
- Dial

Before the Wireless Client's installation and configuration can be performed, it is necessary to:

- Get configuration information from the EWG administrator (user ID, password, IP address, etc.)
- Set up and activate a user account on EWG
- Install modem drivers

The Wireless Client's supported platforms are:

- Windows 95, 98 (SE recommended), 2000, ME, NT4.0 Service pack 4+
- Palm OS
- H/PC 2.00, H/PC Pro 2.11, HPC 2000
- Pocket PC

8.5.1 Wireless client configuration

In this redbook, wireless clients are used to run sample scenarios using the IBM Mobile Connect (or Everyplace Synchronization Manager) and MQSeries Everyplace components of the WebSphere Everyplace Server product offerings.

For example, you may want to perform the following steps to configure a wireless client device:

1. Select **Start->Programs->IBM Wireless Client->Connections**. A Wireless Connections window appears, as illustrated in Figure 8-27.

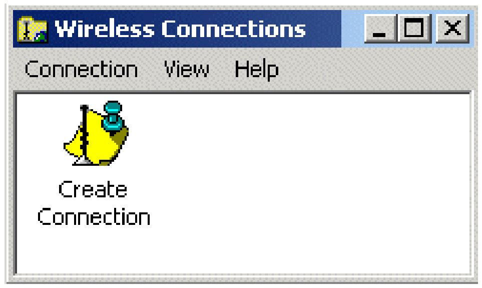
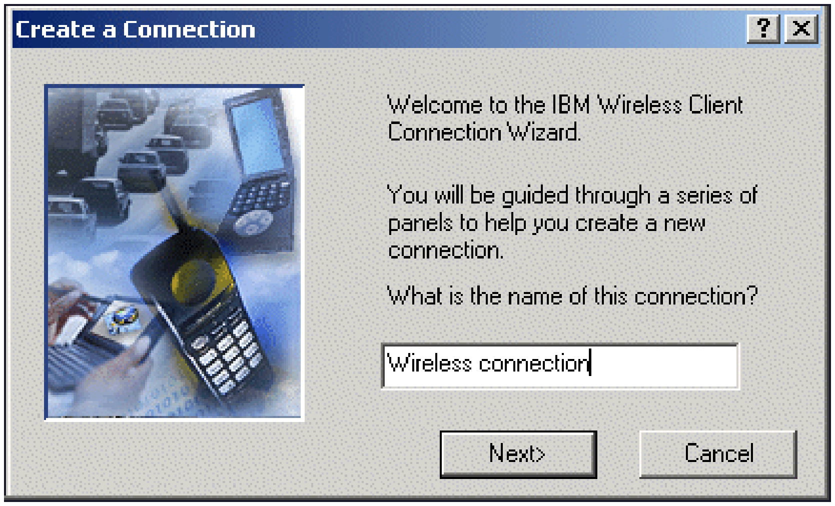


Figure 8-27 Wireless Connections window

2. Double-click the **Create Connection** icon and name the connection (see Figure 8-28).


Create a Connection

Welcome to the IBM Wireless Client Connection Wizard.

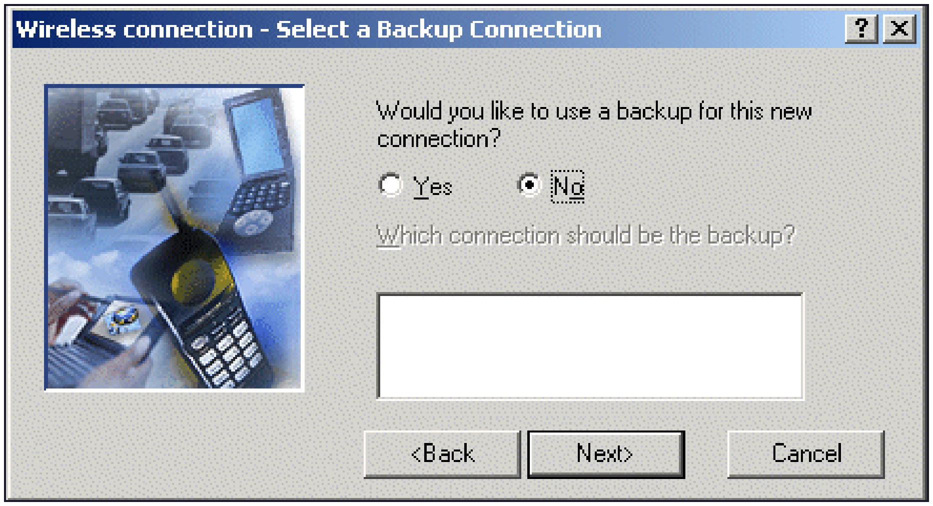
You will be guided through a series of panels to help you create a new connection.

What is the name of this connection?

Wireless connection

Figure 8-28 Create a Connection window

3. As shown in Figure 8-29, select a backup connection if you have more than one connection defined. In this sample scenario, a backup connection is not defined.


Wireless connection - Select a Backup Connection

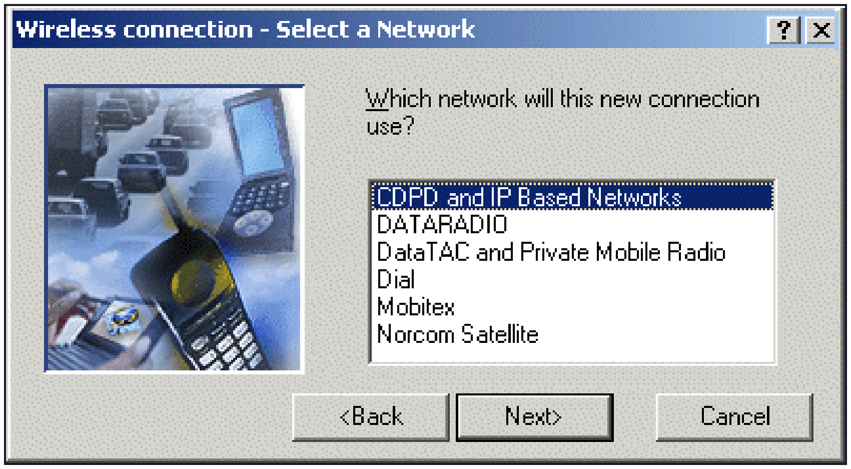
Would you like to use a backup for this new connection?

Yes [✓ No]

Which connection should be the backup?

Figure 8-29 Backup connection

4. Next, you will need to select a network type. For example, Figure 8-30 shows an IP-based network selected, since the scenarios documented in this redbook include an IP over LAN connection.

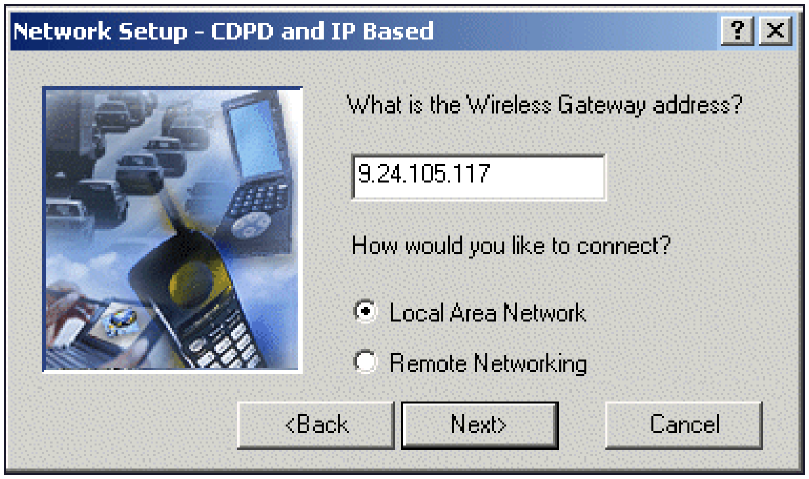


Which network will this new connection use?

CDPD and IP Based Networks
DATARADIO
DataTAC and Private Mobile Radio
Dial
Mobitex
Norcom Satellite

Figure 8-30 Selecting a network type

5. Next, you will enter the Wireless Gateway IP address and select **Local Area Network** or **Remote Networking** for this connection. As shown in Figure 8-31, the scenarios in this redbook use an IP over LAN connection.



What is the Wireless Gateway address?

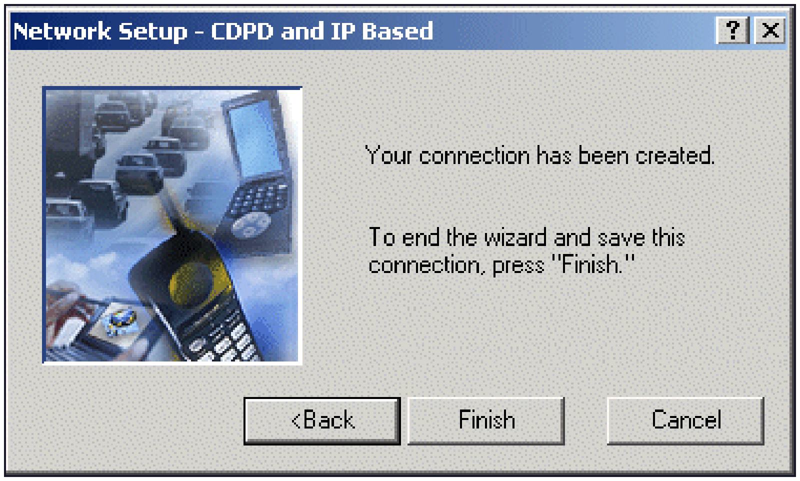
9.24.105.117

How would you like to connect?

[✓ Local Area Network]
Remote Networking

Figure 8-31 Wireless Gateway address

6. As shown in Figure 8-32, after a proper configuration has been entered, a validation window will indicate the creation of this connection.


Network Setup - CDPD and IP Based

Your connection has been created.

To end the wizard and save this connection, press "Finish."

Figure 8-32 Connection validation

7. At this time, you will need to open the connection to the Wireless Gateway by double-clicking the new icon that was created (see Figure 8-33).

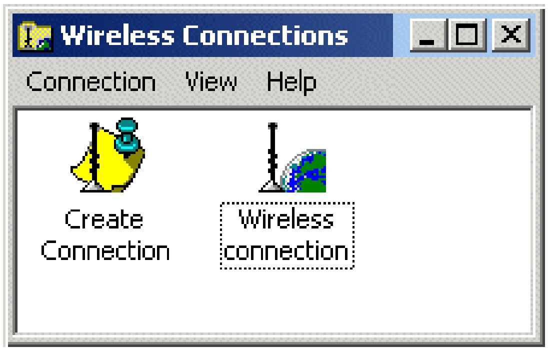
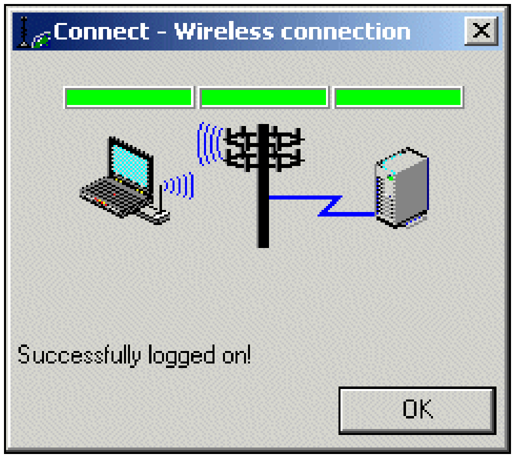


Figure 8-33 Opening the wireless connection

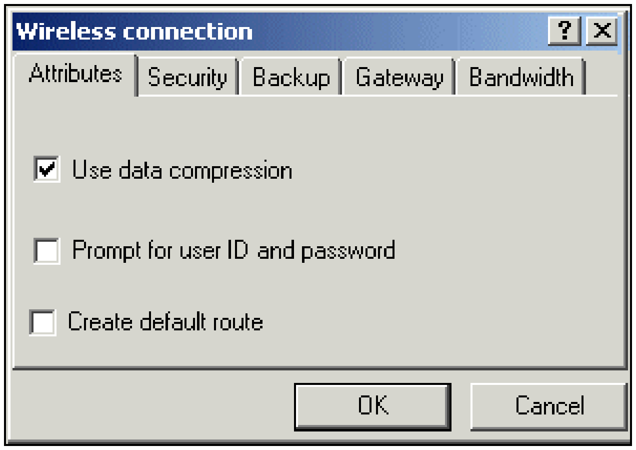
8. As illustrated in Figure 8-34, the connection can be visually monitored; when all three bars are green, it means that a successful connection to the Wireless Gateway has been established.



Successfully logged on!

Figure 8-34 Status of the wireless connection

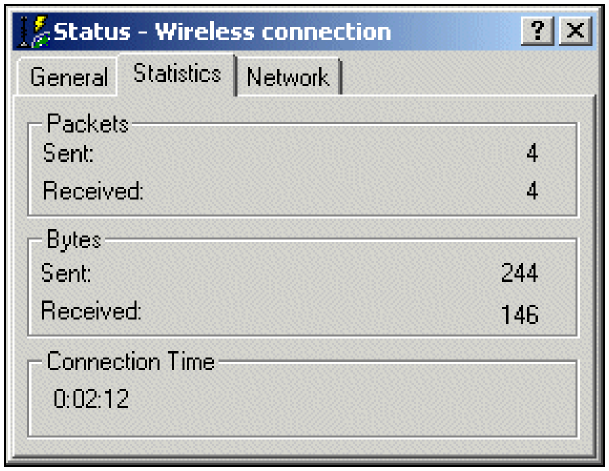
9. As shown in Figure 8-35, you can also configure your wireless connection at any time by right-clicking the wireless connection's icon and selecting the **Properties** option.



- [✓] Use data compression
- [ ] Prompt for user ID and password
- [ ] Create default route

Figure 8-35 Wireless Client connection properties

10. Also when you get a successful connection, a *new connection* icon will appear on the bottom-right of your screen. This enables you to monitor the connection status and statistics, as illustrated in Figure 8-36.



Packets
Sent: 4
Received: 4

Bytes
Sent: 244
Received: 146

Connection Time
0:02:12

Figure 8-36 Status of the wireless connection

8.5.2 Hints and tips

The following lines contain recommended tips to optimize the performance of the Wireless Gateway and client devices using the Wireless Link Protocol (WLP):

- Installing the Wireless Gateway and the Wireless Gatekeeper on different machines can improve the performance of the Wireless Gateway.
- The log files and trace files of the Wireless Gateway grow continuously and rapidly. You can check these files for tests, but for performance matters, it is recommended that you turn off all the log files in the Wireless Gatekeeper and Wireless Client.
- Data compression is provided by the Wireless Gateway and Wireless Client. Just turn it on.
- MTU size determines the maximum packet size sent from the IP stack to the Wireless Client interface. It may improve performance to modify the MTU, depending on your application data and network type.

Part 3

Extending enterprise applications

In this part of the redbook, we introduce new, state of the art technologies that you can use to extend your enterprise applications and provide end-users with better ways to access back-end enterprise data. You will find information about how to develop applications using WebSphere Everyplace Server programming interfaces (APIs) such as Push messages, Location-Based Services, Intelligent Notifications Services and Voice applications. We have included numerous scenarios describing recommended ways to develop applications using the APIs provided by the WebSphere Everyplace Server components to support these services.

Although WebSphere Everyplace Server offerings do not provide a Voice Server, we have also included guidelines to develop Voice XML applications using transcoding capabilities provided by WebSphere Everyplace Server . This redbook includes scenarios using IBM Mobile Connect and Synchronization Manager using a sample DB2 Everyplace application built with the Mobile Application Builder. Transaction messaging applications are also described using MQSeries Everyplace, a component of WebSphere Everyplace Server, to provide a once-only assured delivery of messages.

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

9

Push messaging applications

This chapter describes how to develop and execute Push messaging applications using the messaging capabilities provided by the IBM Everyplace Wireless Gateway, a component of the WebSphere Everyplace Server (WES) Service Provider Offering (SPO) and an optional component of the WebSphere Everyplace Server Enable Offering (EO).

This chapter provides a description of the messaging framework (architecture and protocols); it then explains how to implement a messaging application using the IBM Messaging and Push Toolkit SDK. Sample programs are also included.

9.1 Overview

Messaging in the context of WebSphere Everyplace Server describes the transmission of content between hosts at end points of networks through the Everyplace Wireless Gateway. The messages could be initiated by applications running either on client devices, on a middle tier or on a back-end server.

Applications built to send messages to client devices without a request are called *push messaging applications*. This is different from the more established pull applications, in which the client first sends a request to a server and then receives a response to that message. Push messaging applications could be used to provide asynchronous service, delivering to targeted devices the content that is of interest to the user.

Some events which can be used to trigger a push message are, for example:

- The time of day.
- A change in the status of a reservation.
- Data reaching a preset threshold value, at which point the user would like to take action.

To receive push messages, a user will only be required to register for the service. Services that are good candidates for push applications include stock quotes, weather reports, flight information and news, among others. Figure 9-1 illustrates a simple comparison of push and pull technologies currently used with wireless devices such as WAP phones.

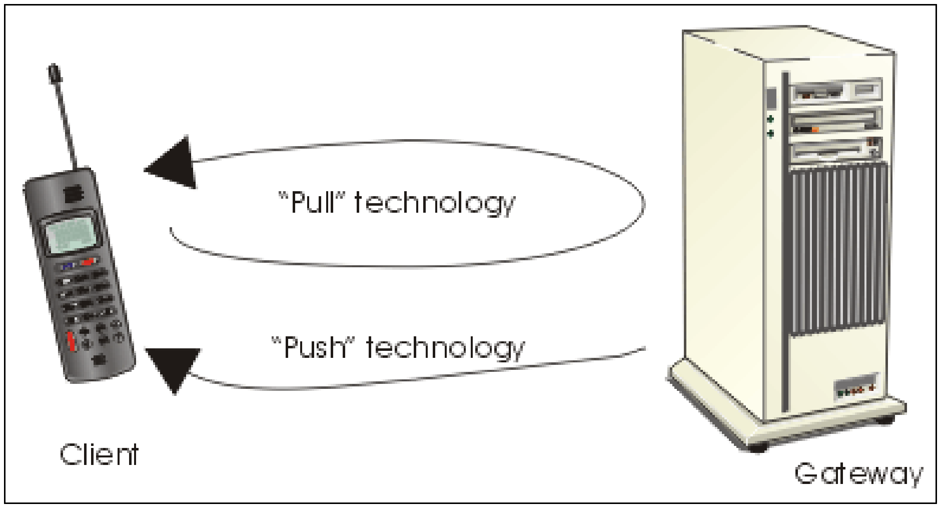


Figure 9-1 Simplified view of push versus pull operation

9.2 Architecture

In a minimal configuration, a messaging solution is made up of three logical components:

- The messaging application, also known as the Push Initiator (PI).
- The messaging gateway, also known as the Push Proxy Gateway (PPG).
- The target or client device.

The IBM Everyplace Wireless Gateway (EWG), after installation, can be configured to provide messaging services. In this configuration, it is considered a PPG as described in the WAP Forum specifications on Push.

It should be noted that to achieve scalability and improved performance, there can be multiple messaging applications and gateways on a single network. Such a messaging network could include several other components that do not participate directly in the messaging function.

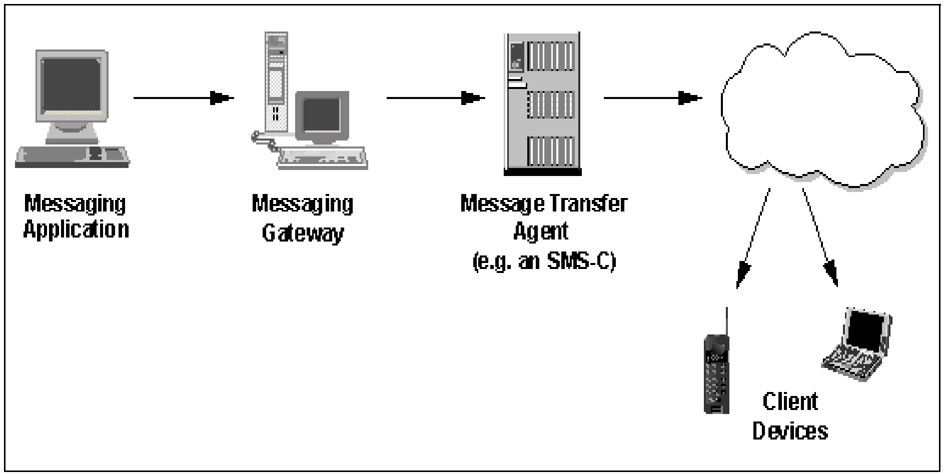


Figure 9-2 Messaging infrastructure

An IBM Network Dispatcher may be installed between the PIs and several messaging gateways to load-balance the messaging traffic between them[1].

Wireless Gateways are typically configured in clusters to distribute workload. Consequently, any messaging gateway components configured on them operate within the rules of the cluster. However, the performance of the messaging gateway does not benefit from the clustered arrangement.

[1] The IBM Network dispatcher is available as a component of the IBM WebSphere Edge Server.

The PPGs within a cluster should have the same configuration to ensure consistent service to client PIs. However, a known issue with this topology is that a network dispatcher may not maintain message affinity between a PI and a PPG. This opens the possibility that a messaging application may send status queries and cancel requests that are not delivered to the gateway which handled the original message. The messaging gateway will have no record of the push ID in the status queries and so will respond with a message status code indicating an unrecognizable push ID.

As shown in Figure 9-2, the network agent to which the messaging gateway delivers push messages is called the Message Transfer Agent (MTA). Each supported network has a different MTA.

9.2.1 Messaging application

The messaging application is responsible for initiating the message transfer of push messages. It runs on the origin server and can be triggered by preset preferences of registered users of the application. To send a message, it builds a message context, specifies the address of the messaging gateway and client device(s) that will receive the message, the content type of the message and the message itself. It is typically a component of a larger enterprise application.

A messaging application can add the push feature by calling classes in a Push API. Several Push APIs are available for use in building applications. In this redbook, we shall be using the Push API provided in the Messaging Toolkit from IBM.

The IBM Push API allows you to communicate with the messaging gateway installed in the Wireless Gateway. It enables a developer to implement the Push Framework described in the WAP specification without having to know the internal issues and details of this technology. This way, the development focus is on message application rather than standards compliance.

Figure 9-3 illustrates a typical profile of a Push Initiator (PI) application program where a push message is created and sent to a client device.

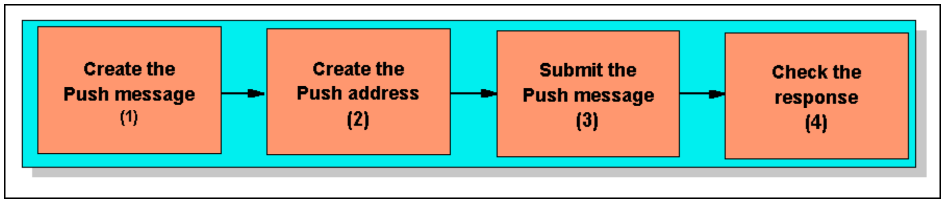

Create the Push message (1)
Create the Push address (2)
Submit the Push message (3)
Check the response (4)

Figure 9-3 Push Initiators (PIs) lifecycle

Beyond the message submission stage, a PI may also request the status of a message or cancel a previously sent message.

9.2.2 Messaging Gateway

The Messaging Gateway is a component of the Everyplace Wireless Gateway (EWG). The EWG Messaging Gateway can handle both WAP and non-WAP push requests from the messaging application (PI) on a specified port. On arriving at the gateway, the path of a request through the gateway depends on its type. WAP messages are passed to the EWG WAP services while non-WAP messages are sent to the proper Mobile Network Connection (MNC); for example, a mail message will be sent to a mail server.

Mobile Network Connection (MNC)

A Mobile Network Connection, or MNC, is a resource of the Wireless Gateway and provides the EWG with a connection to a network. For example, as illustrated in Figure 9-4, for every type of WAP network (SMS, Dial, IP and others) there is a different MNC.

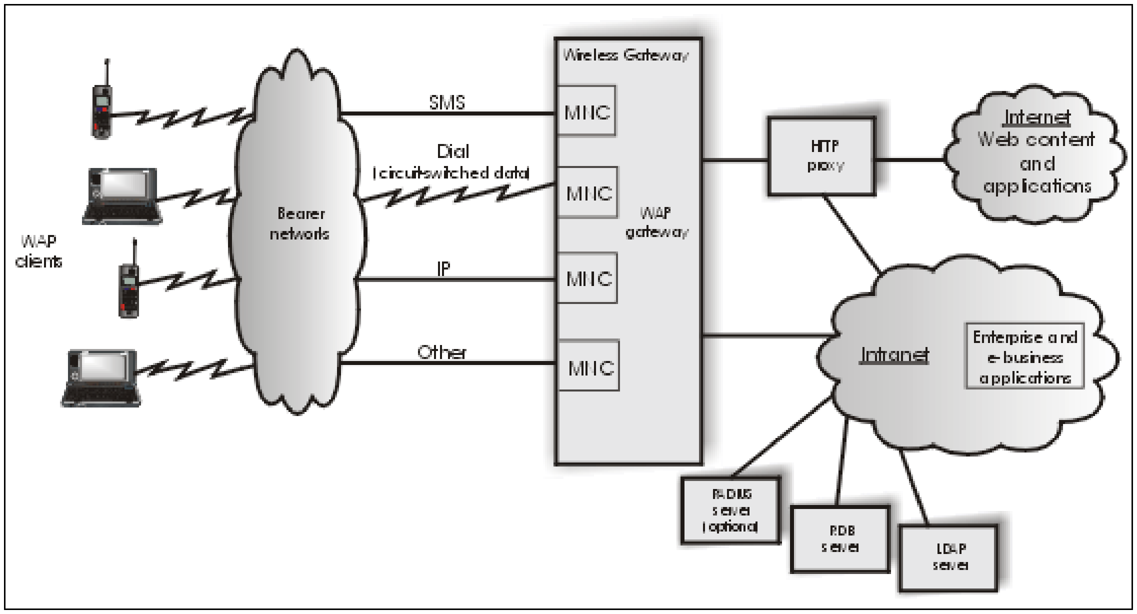


Figure 9-4 Wireless Gateway configured as a WAP gateway

Short messages and WAP push content are delivered over a variety of networks and protocols including:

- GSM SMS
- Mobitex (Cingular in the US)
- DataTAC (Motient in the US)
- Simple Network Paging Protocol or SNPP (Nextel and Skytel in the US)
- Simple Mail Transfer Protocol (SMTP)
- TCP/IP

When a messaging gateway is configured in the EWG, a port number is provided, on which the gateway listens for requests from applications (PIs) using the Messaging Services and Push APIs. The default port number is 13131.

WAP messages

Wireless Access Protocol (WAP) is a set of open protocols developed by the WAP Forum, that are followed in developing applications and services that use wireless networks. WAP messages follow the format laid out by the standard. For more information on WAP specifications, see WAP-151, WAP-164, Push Access Specification and WAP-165, Push Architectural Overview.

Within the EWG, support for WAP messaging is provided by a WAP services module. It complies with the WAP Push Proxy Gateway 1.2 specification from the WAP forum. When configured as a WAP gateway, as shown in Figure 9-5, WAP push messages are routed through this component and leave the gateway through a a WAP configured mobile network connection (MNC).

As a WAP gateway, the EWG performs a protocol conversion to provide communication with WAP clients. The data and messages arriving from applications with HTTP headers are given into WSP headers and their content encoded into binary XML (wbxml).

Supported networks include:

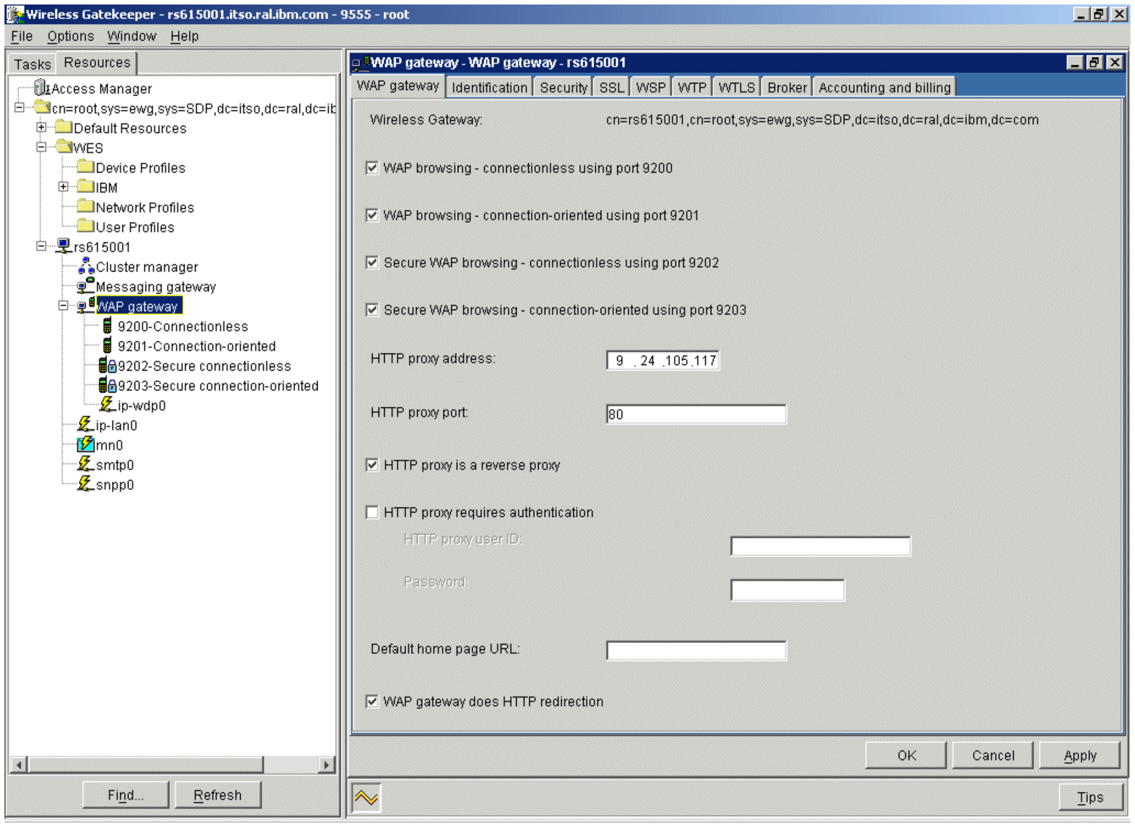
- GSM SMS (UCP and SMPP protocols)
- Mobitex
- DataTAC
- IPv4

In this configuration, the components of the messaging infrastructure map to the WAP Push specifications as follows:

- Messaging application -> Push Initiator (PI).
- EWG messaging gateway -> Push Proxy Gateway (PPG).

Messages travel between them over the Push Access Protocol (PAP).

The PAP protocol specifies the format of the messages exchanged by the PI and the PPG. It uses HTTP to travel over the Internet and can be easily adapted to use any other prevalent network protocol.


Wireless Gatekeeper - rs615001.itso.ral.ibm.com - 9555 - root
File Options Window Help
Tasks Resources
Access Manager
cn=root,sys=ewg,sys=SDP,dc=itso,dc=ral,dc=ib
Default Resources
WES
Device Profiles
IBM
Network Profiles
User Profiles
rs615001
Cluster manager
Messaging gateway
WAP gateway
9200-Connectionless
9201-Connection-oriented
9202-Secure connectionless
9203-Secure connection-oriented
ip-wdp0
ip-lan0
mn0
smtp0
snpp0
Find... Refresh
WAP gateway - WAP gateway - rs615001
WAP gateway Identification Security SSL WSP WTP WTLS Broker Accounting and billing
OK Cancel Apply
Tips

Wireless Gateway: cn=rs615001,cn=root,sys=ewg,sys=SDP,dc=itso,dc=ral,dc=ibm,dc=com

☑ WAP browsing - connectionless using port 9200

☑ WAP browsing - connection-oriented using port 9201

☑ Secure WAP browsing - connectionless using port 9202

☑ Secure WAP browsing - connection-oriented using port 9203

HTTP proxy address: 9 .24 .105.117

HTTP proxy port: 80

☑ HTTP proxy is a reverse proxy

☐ HTTP proxy requires authentication

HTTP proxy user ID:

Password:

Default home page URL:

☑ WAP gateway does HTTP redirection

Figure 9-5 Sample WAP configuration in Wireless Gateway

Push message content

For the delivery of unsolicited WAP content using the Messaging Gateway, the Everyplace Wireless Gateway supports the following document specifications in WAP Push messages:

- Service Indication (SI): used to send an unsolicited message to the WAP device. Optionally, a URI can be included to invoke an application.
- Service Loading (SL): used to send a URI to load (pull) an application.

Figure 9-6 illustrates the path of a WAP Push message starting from the application (Push Initiator) to the Messaging Gateway or Push Proxy and to the device agent for delivery.

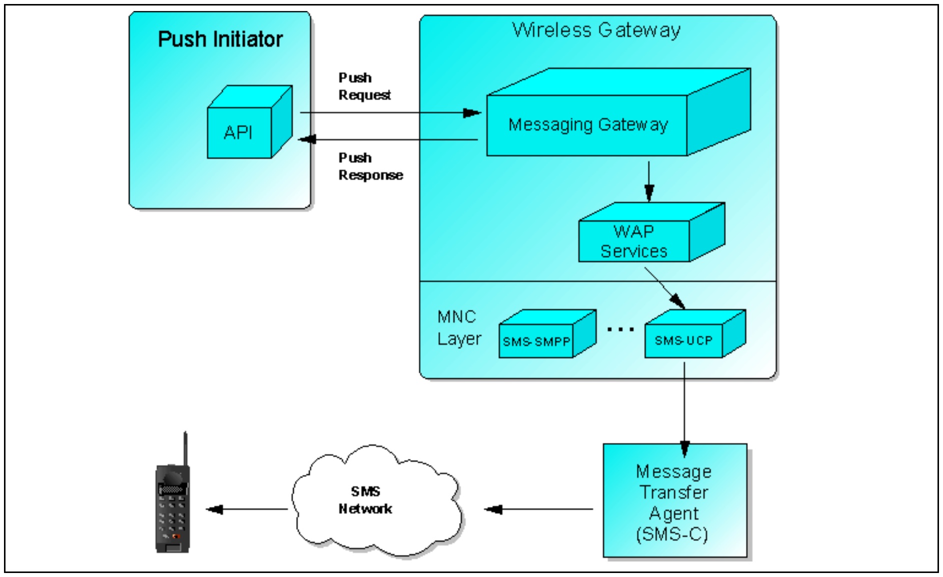


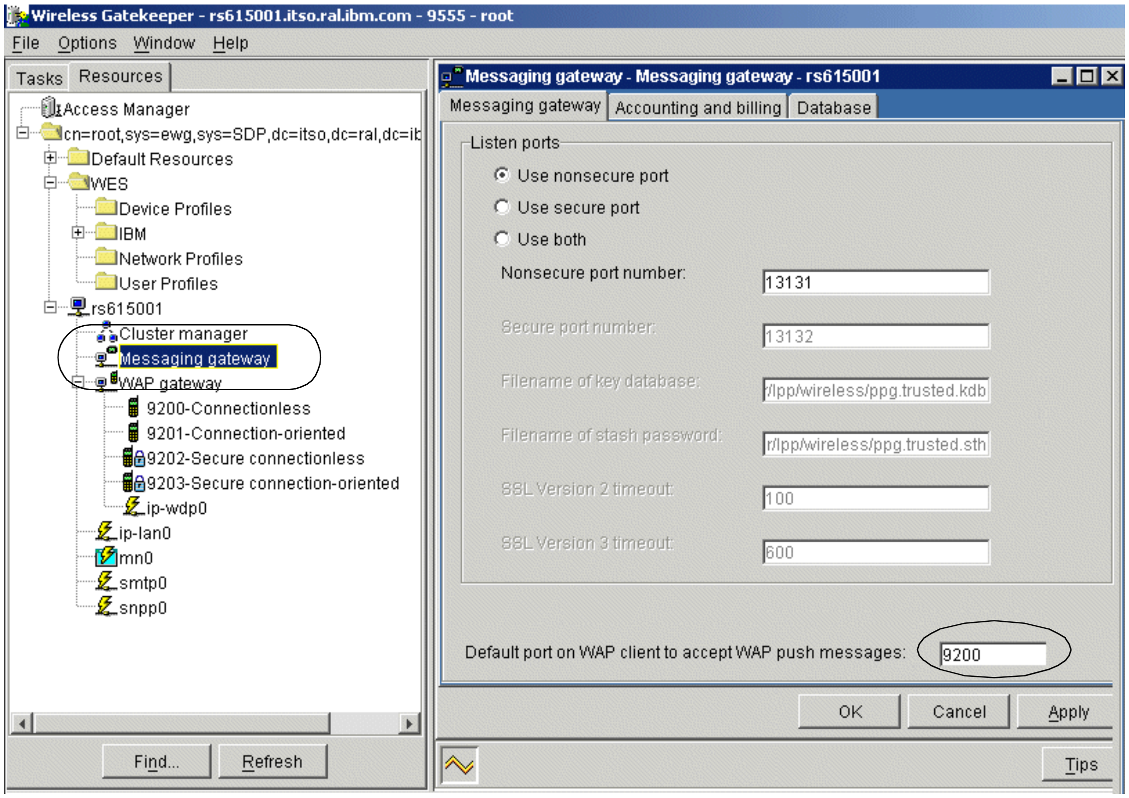
Figure 9-6 Path of a WAP Push message

Non-WAP Push

This is by far the more prevalent type of messaging traffic at this time. IBM has, by extending the WAP standards, added non-WAP capabilities to the messaging gateway so that it can interface with the more established short messaging networks. Supported networks in this release include:

- GSM SMS (UCP and SMPP Protocols)
- SNPP
- Mobitex
- DataTAC

- SMTP


Wireless Gatekeeper - rs615001.itso.ral.ibm.com - 9555 - root

File Options Window Help

Tasks | Resources

- Access Manager
- cn=root,sys=ewg,sys=SDP,dc=itso,dc=ral,dc=ib
  - Default Resources
  - WES
    - Device Profiles
    - IBM
    - Network Profiles
    - User Profiles
  - rs615001
    - Cluster manager
    - Messaging gateway
    - WAP gateway
      - 9200-Connectionless
      - 9201-Connection-oriented
      - 9202-Secure connectionless
      - 9203-Secure connection-oriented
      - ip-wdp0
    - ip-lan0
    - mn0
    - smtp0
    - snpp0

Messaging gateway - Messaging gateway - rs615001

Messaging gateway | Accounting and billing | Database

Listen ports

[✓ Use nonsecure port] Use secure port Use both

Nonsecure port number: 13131

Secure port number: 13132

Filename of key database: /lpp/wireless/ppg.trusted.kdb

Filename of stash password: r/lpp/wireless/ppg.trusted.sth

SSL Version 2 timeout: 100

SSL Version 3 timeout: 600

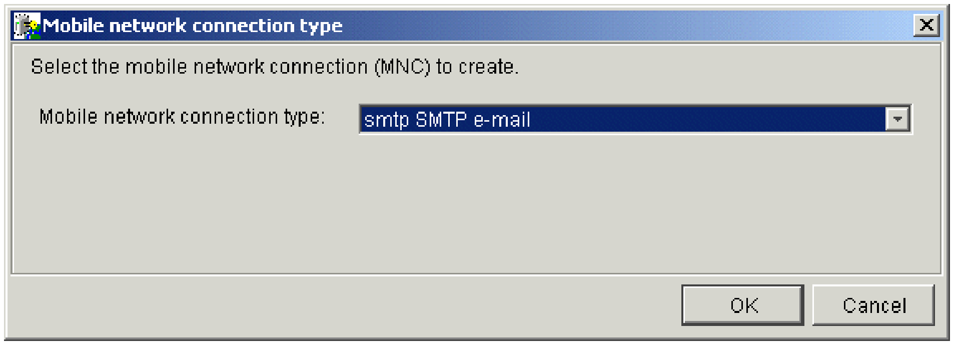
Default port on WAP client to accept WAP push messages: 9200

Figure 9-7 Messaging gateway configuration in Wireless Gateway

Non-WAP messages are routed through the messaging gateway and then directly to the MNC corresponding to the delivery network specified in the message or to the address type of the client device. They bypass the WAP services module of the EWG.

SMTP MNC configuration

To ensure delivery of an e-mail message which is received by the messaging gateway, an SMTP MNC must be configured. On the Wireless Gateway, right-click the gateway name and select **Add->Mobile Network Connection**. Select the SMTP e-mail MNC type as shown in Figure 9-8 on page 332.

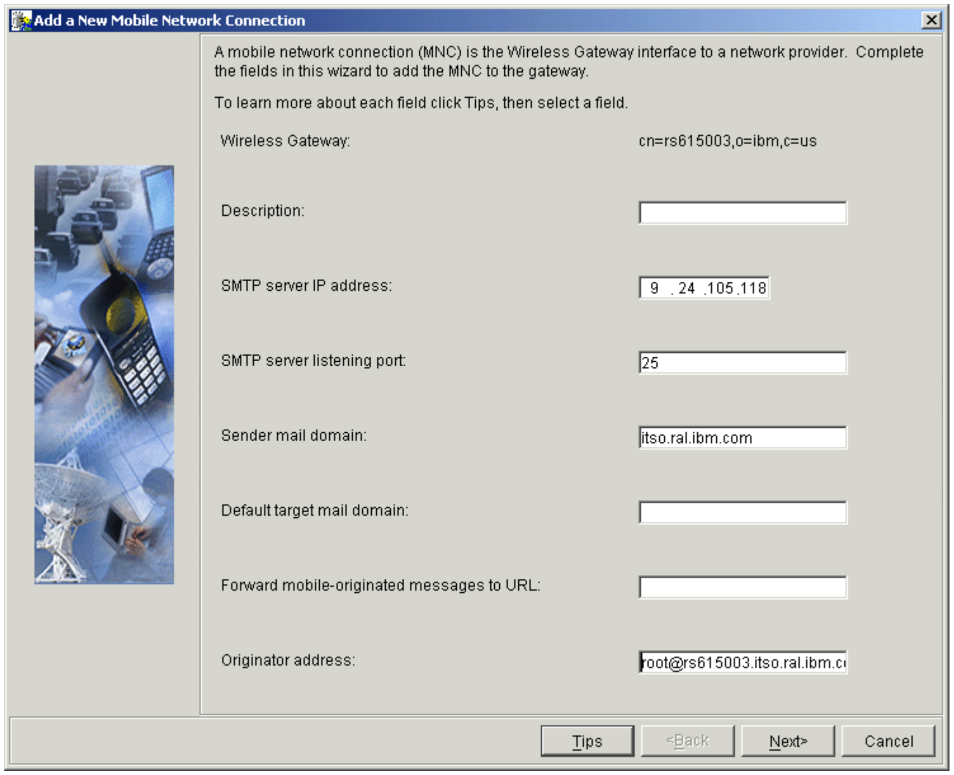


Select the mobile network connection (MNC) to create.

Mobile network connection type: smtp SMTP e-mail

Figure 9-8 SMTP Mobile Network Connection (MNC)

Complete the panel as shown below:



A mobile network connection (MNC) is the Wireless Gateway interface to a network provider. Complete the fields in this wizard to add the MNC to the gateway.

To learn more about each field click Tips, then select a field.

Wireless Gateway:	cn=rs615003,o=ibm,c=us
Description:	
SMTP server IP address:	9 .24 .105.118
SMTP server listening port:	25
Sender mail domain:	itso.ral.ibm.com
Default target mail domain:	
Forward mobile-originated messages to URL:	
Originator address:	root@rs615003.itso.ral.ibm.c

Figure 9-9 SMTP MNC configuration

Enter the IP address of the SMTP server (for example, 9.24.105.118), the sender's mail domain (for example, itso.ral.ibm.com) and the originator's address (for example, root@rs615003.itso.ral.ibm.com). This address will appear in the `From:` part of the SMTP message which is generated. Click **Next** and then **Finish**.

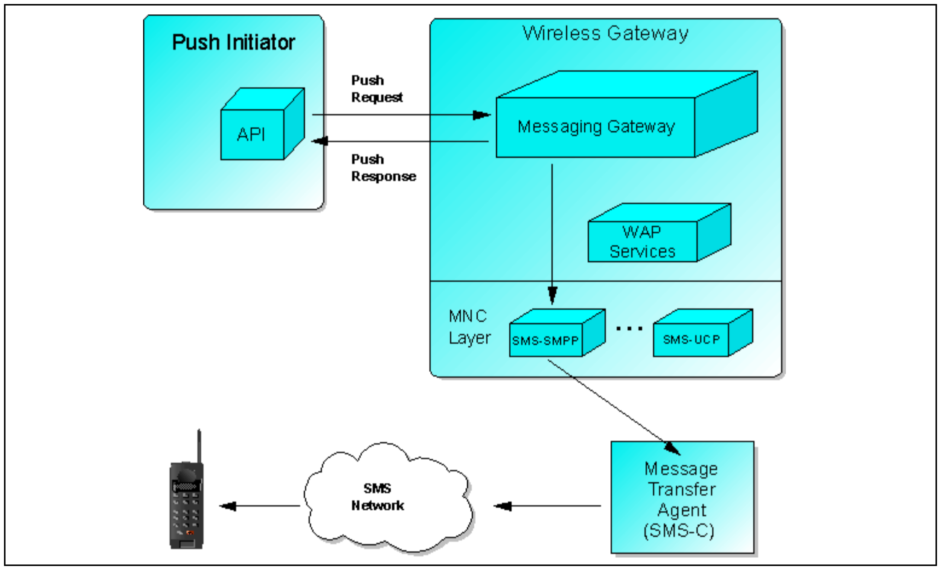


Figure 9-10 Non-WAP Push configuration

For details on how to configure the Everyplace Wireless Gateway as a messaging or a WAP gateway, see Chapter 8, "Going wireless!" on page 281.

9.2.3 Client devices

These are the portable communications devices that are the target of push messages in the case of mobile-terminated messages. Client devices run applications such as WAP browsers capable of handling message content delivered to them. Possible client devices include a WAP phone, SMTP e-mail client, a GSM-SMS mobile phone, a DataTAC end-user device, an SNPP paging device or a Mobitex end-user device.

Client devices can also play the role of message initiator, composing and sending short messages. The Wireless Gateway can receive these mobile-originated messages.

9.3 Push API

IBM has made available a Messaging Services and Push Toolkit that can be used to build applications to push short messages from an application running on a wired network through the Everyplace Wireless Gateway to client devices. An application developer using the toolkit is shielded from the details of the protocols. APIs for Java and C are provided. However, sample scenarios included in this redbook use the Java API only.

9.3.1 Obtaining the messaging toolkit

The Messaging Services and Push Toolkit can be dowloaded from:

```
http://www-3.ibm.com/pvc/tech/downloads.shtml
```

To install the Toolkit, perform the following steps:

- Download toolkit.zip into a directory of your choice.
- Unzip toolkit.zip.

Note: The Toolkit documentation files, programming library files, and sample programs are installed in the a:/ directory and subdirectories relative to where you unzip the toolkit.zip.

To compile the Java API, you must first add these file names to your CLASSPATH or specify the paths on the command line using the **`-classpath`** option:

- help/en/messagingtoolkit/Java/push.jar (classes of the Java API).
- help/en/messagingtoolkit/Java/xerces.jar (class of the XML parser).
- help/en/messagingtoolkit/Java/log.jar (classes of the logging facility).

Once you have compiled the samples (using the **`javac`** command for example), you will need to modify the push.properties file in the directory with the samples, so that the push.proxy-url value points to your messaging gateway and port (default port is 13131) where the push.properties file is located.

```
help\en\messagingtoolkit\java\doc\doc-files
```

9.3.2 Configuration

A developer using the toolkit has the option of setting the key parameters from within the application code or setting default values in a properties file called push.properties. The file can be made available to the application as a system wide property called push.properties-file[2] or it can be placed in the same *working directory* as the application where a running program using the API searches for it first by default.

The file follows the normal syntax for a Java properties file. Lines beginning with a pound (#) symbol are comments for the reader and are ignored by the interpreter. Blank lines do not matter. An actual property is defined and assigned a value in a single line. For example:

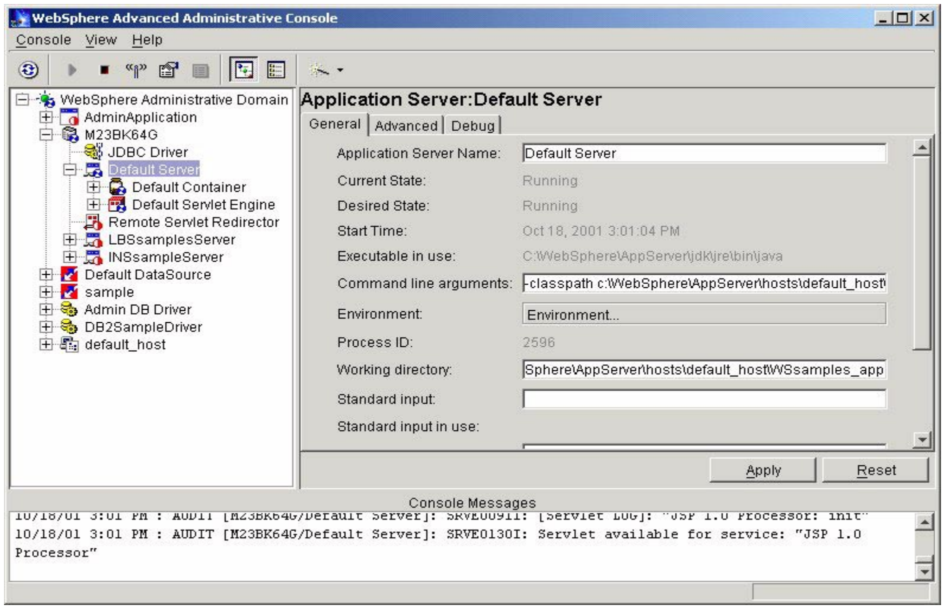
```
push.proxy-language = en
```

The file push.properties contains a parameter that tells any push program where to look for the DTD file called PAP_1.0.DTD. The parameter is:

```
push.pap-dtd-uri=./pap_1.0.dtd
```

For example, in the WebSphere environment, you are required to specify the working directory and place the push.properties file into the specified directory. The working directory of WebSphere Application Server is configured using the Administrative Console, as shown in Figure 9-11.

[2] The system property name is a macro for the fully qualified pathname of the file.



Application Server:Default Server

Application Server Name:	Default Server
Current State:	Running
Desired State:	Running
Start Time:	Oct 18, 2001 3:01:04 PM
Executable in use:	C:\WebSphere\AppServer\jdk\jre\bin\java
Command line arguments:	-classpath c:\WebSphere\AppServer\hosts\default_host\
Environment:	Environment...
Process ID:	2596
Working directory:	Sphere\AppServer\hosts\default_host\WSsamples_app
Standard input:	
Standard input in use:	

```
10/18/01 3:01 PM : AUDIT [M23BK64G/Default Server]: SRVE0091I: [Servlet LOG]: "JSP 1.0 Processor: init"
10/18/01 3:01 PM : AUDIT [M23BK64G/Default Server]: SRVE0130I: Servlet available for service: "JSP 1.0
Processor"
```

Figure 9-11 Configuring WebSphere Application Server working directory

The API documentation lists the properties of the Java APIs as well as their configuration options. Also included is a sample properties file: push.properties.

The Java API in the toolkit consists of three packages:

- com.ibm.wireless.push
- com.ibm.wireless.push.util
- com.ibm.wireless.push.samples

The core package which a developer using the Java API must import is the com.ibm.wireless.push package. It contains the classes and interfaces for sending and receiving push messages from the EWG configured as a messaging gateway.

The Toolkit Java API contains four key classes:

- Pusher
- PushMessage
- PushAddress
- PPG Responses

Class	Description
Pusher	Used to instantiate an object that holds data relative to the push environment. Its configuration data can be obtained from a default properties file or can be set within the program by passing the URL for the messaging gateway, the port number of the listening thread and the SSL context to the constructor.
PushMessage	The PI creates a message object of this type to encapsulate a push message. The object members include the control entity, the content entity and optionally, the capabilities information.
PushAddress	A data type created to hold the address of the various target devices (e-mail, pagers, mobile phones). It is the base class in a hierarchy of address classes used to specify the message type.
PPG Responses	Several classes that contain the responses to status queries sent to the messaging gateway from the PI. These classes include: PushResponse, StatusResponse, CancelReponse, CCQResponse and ResultNotification.

Table -1 Key Classes in the Java API of the Messaging Toolkit.

Example 9-1 The PushAddress class hierarchy

```
        com.ibm.wireless.push.PushAddress
           |
           + ------ com.ibm.wireless.push.IPv4PushAddress
           |
        ---+ ----- com.ibm.wireless.push.IPv6PushAddress
           |
           + ------ com.ibm.wireless.push.MANPushAddress
           |
           + ------ com.ibm.wireless.push.PLMNPushAddress
           |
           + ------ com.ibm.wireless.push.DataTACPushAddress
           |
        ---+----- com.ibm.wireless.push.USERPushAddress
           |
           + ------ com.ibm.wireless.push.SMTPPushAddress
           |
           + ------ com.ibm.wireless.push.SMPPPushAddress
        |
           + ------ com.ibm.wireless.push.PLMNPPPushAddress
```

9.3.3 IBM extensions

To give the Push Framework broader application, IBM has extended its messaging gateways support to:

- Other delivery channels
- Mobile-originated messages
- Security

Delivery channels

PAP was designed from the beginning to be adaptable to Internet protocols (as it piggybacks on them). A few such adaptations were made to enable the messaging gateway to support the following messaging technologies:

- WAP (including service over all defined WAP bearers)
- GSM-SMS over a GSM network
- SMTP
- SNPP
- Mobitex SMS
- DataTAC

The intended delivery channel of a message is specified by the address type (/TYPE=<address-type>) and optionally , an added delivery (/DELIVERY=<delivery-channel>) attribute. For example, a message address with the following address:

```
wappush=+41-79-678-12345/DELIVERY=SMS/TYPE=PLMN@pi.ibm.com
```

tells the messaging gateway to deliver over a GSM-SMS channel, rather than WAP the default channel for PLMN address types.

Only one delivery channel is recognized by the messaging gateway for each message. This eliminates a possible conflict with the quality of service attribute, which may not be applicable to all the indicated channels.

Mobile-originated messages

A client device capable of composing messages may send them over an SMS network and through the messaging gateway to a processing application (see Figure 9-12). Within the gateway, the message is posted to a designated URL where it is interpreted by an application. Each MNC within the gateway has a designated forwarding URL.

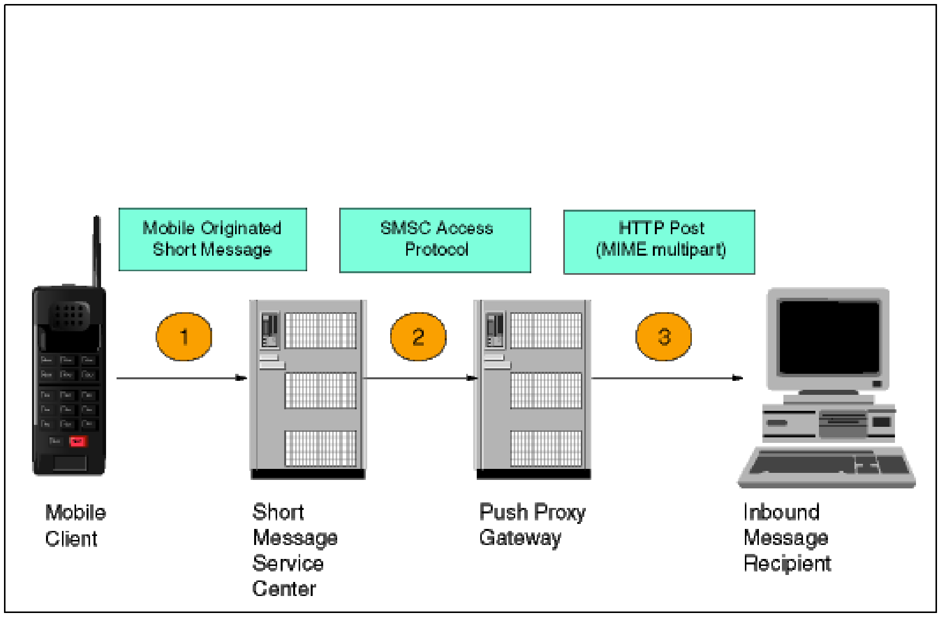

Mobile Originated Short Message
SMSC Access Protocol
HTTP Post (MIME multipart)
1
2
3
Mobile Client
Short Message Service Center
Push Proxy Gateway
Inbound Message Recipient

Figure 9-12 Receiving a mobile-originated Push

Security

In the WAP PAP 1.2 standard, there is no provision for security when messages are sent between the PI and the PPG. The EWG can support secured connections when communicating with the PI using the Secure Socket Layer (SSL) protocol with the gateway acting as the server and the PI acting as a client. The gateway can extend this secure link to any message entity arriving from or destined for a host URL-address with an HTTPS protocol component.

9.4 Message components

Push messages are made up of:

- Control Entity
- Content Entity
- Capabilities Entity (optional)[3]

Entity	Description

Control Entity	Contains attributes that specify how the gateway is to handle the message. By reading its contents, the gateway learns the following about the message:
	1. Content provider, set using
	`pushMessageObject`[a]`.setContentProvider ("providerName");`
	2. Need for progress notes, set using
	`pushMessageObject.notifyPushProgress(true); //false by default`
	3.Delivery time constraints ,set as follows:
	`pushMessageObject(DateObject.setTime() + someDelay);`
	4.Quality of service, the components of which are:
	i. Priority, set to either **low**, **medium** or **high** using
	`pushMessageObject.setDeliveryPriority(DeliveryQoS.PRIORITY_HIGH);`
	ii .Need for confirmation of delivery, which could have possible values **confirmed**, **preferconfirmed**, **unconfirmed** and **notspecified** , for example
	`pushMessageObject.setDeliveryMethod(DeliveryQoS.DELIVERY_CONFIRMED);`
	iii. Delivery network type, for example
	`pushMessageObject.setNetworkType(DeliveryQoS.NETWORK_GSM);`
	iv. Bearer type, for example
	`pushMessageObject.setBearerType(DeliveryQoS.BEARER_SMS);`

[3] Capabilities queries are currently NOT supported by the IBM Wireless Gateway.

Entity	Description
Content Entity	Consists of a header and a body entity. The header could be either: 1.Generic: Similar to the Internet message headers in common use. Fields could be set through the JAVA API, for example the cache-control field, date: `setCacheControl`, `setDate` 2. WAP: Begin with the prefix X-WAP, for example X-Wap-Application-Id 3. IBM proprietary headers. Body: the actual message content to be delivered.The content-type of the body is a function of the delivery channel; for example: GSM-SMS channels carry text/plain
Capabilities Entity	Used by the PI to inform the gateway about the assumed capabilities of the client receiving this message. The value can be set through call to the Java API; for example: `setCapability(URL url)`

a. Handle of a PushMessage object returned from the call *new PushMessage()*;

For more information on the message contents, see the Developer's Guide contained in the Messaging Toolkit download.

9.5 Secure push connections

You can also configure secure connections by using Secure Sockets Layer (SSL) when sending messages between applications and the messaging gateway using the Messaging Service and Push APIs. The SSL environment must be configured at the Push Initiator and the messaging gateway. The default port on which the messaging gateway listens for SSL requests is 13132. The messages are HTTP formatted and can be secured.

SSLight

As part of the toolkit installation, the sslight.jar file is placed in the CLASSPATH of the messaging application server.

When implementing security, the SSL context string has the syntax:

```
CLASS:class-file-name:password
```

For example:

```
CLASS:keyring.class;SecurePush
```

To enable the SSL, you may either use the Pusher class constructor, which requires an SSL-context string, or configure the system properties push.proxy-url and push.ssl-context. For example:

```
Pusher pusher = new
Pusher("http://ppg.ibm.com:54321","CLASS:keyring.class;SecurePush");
```

As an alternative, you can use this second option:

```
push.proxy-url=https://ppg.ibm.com:54321
push.ssl-context="CLASS:keyring.class;SecurePush"
```

After this, you may simply use the **`no-arg`** constructor to instantiate a Pusher object. For example:

```
Pusher pusher = new Pusher();
```

9.6 Scenario: WAP push

This example had the push initiator running on a PC workstation, pushing a WML file as the message to a WAP emulator running on another PC, using a messaging gateway.

Example 9-2 MySimplePush.java

```
import java.net.*;
import java.io.*;
import com.ibm.wireless.push.*;
public class MySimplePush {
    /** Main method.<p>
     *
     * @param arg0 the IP Address to send the message or the fully-qualified
       WAPPUSH address.
     * @param arg1 the URL of the messaging gateway
     * @param arg2 the sender.  */
    public static void main(String args[]) {

        String pushAddr = null;   // receiver IP Addresss
        String message = null;    // message text
        String ppgURL = null;     // URL of messaging gateway

        /************************************************************
         * Process input arguments
         ************************************************************/
        pushAddr = args[0];

        message =
          "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?> \n" +
          "<!DOCTYPE si PUBLIC \"-//WAPFORUM//DTD SI 1.0//EN\"\n" +
          "\"http://www.wapforum.org/DTD/si.dtd\">\n" +
          "<si>\n" +
          "<indication>\n" +
          "This is my SIMPLE PUSH message\n" +
          "</indication>\n" +
          "</si>\n";

        ppgURL = "http://" + args[1] + ":13131" ;

        /**************************************************************
         * Compose and submit message
         **************************************************************/
        Pusher pusher = null;
        PushMessage msg = null;
        IPv4PushAddress addr = null;
        PushResponse rsp = null;
        try {
            /**************************************************************
             * Create message
             **************************************************************/
            msg = new PushMessage();
            msg.setContent(message, "text/vnd.wap.wml");
            msg.setFrom(args[2]);
             /* Ask for progress notes */
            msg.notifyPushProgress(true);

            System.out.println(msg);
            System.out.println
             ("------------------------------------------------");
            System.out.println(message);

            /**************************************************************
             * Create an address to send the message to
             **************************************************************/
            addr = new IPv4PushAddress(pushAddr);
            addr.setAddress(pushAddr);

            System.out.println(addr);

            /**************************************************************
             * Submit message
             **************************************************************/
            /* Instantiate a pusher */

                pusher = new Pusher(ppgURL, null);


            /* Push the message */
            rsp = pusher.push(msg, addr);
```

```
        } catch(PushException e) {
            System.err.println(e);
            System.exit(1);
        }

        /**************************************************************
         * Check status reported
         **************************************************************/
        System.out.println(rsp);
        if (rsp.getStatusClass() != StatusCodes.SUCCESS) {
            System.out.println("Push submission failed!");
            System.exit(1);
        } else {
            System.out.println("Push submission succeeded!");
            System.exit(0);
        }
    }
    private static void usage() {
        System.out.println
("Usage:\n\tjava com.ibm.wireless.push.samples.SimplePush <address> <message>
[<ppg-URL>]");
        System.out.println
("\twhere <address> is something like \"195.153.199.30\"");
        System.out.println("\t      <message> text ");
        System.out.println
("\t      <ppg-URL> is the URL of the PPG; if this argument is not\n\t
specified, the system property \"push.proxy-url\" must be set;\n\t      e.g.,
\"push.proxy-url=http://ppg.ibm.com:12345/\"");
        System.exit(1);
    }
}
```

The sample PI program shown in Example 9-2 sends the hardcoded .wml file to the PPG. At the PPG, the message is encoded and sent to the WAP emulator running on the host machine whose IP Address is supplied in the PushAddress field. The message is received by the listening thread of the emulator program in binary XML format and decoded to obtain the original WML message.

9.7 Scenario: Pushing to an e-mail client

The Program Inititor (PI) module built to send an e-mail message is similar to the WAP Push PI. The main difference is that it uses the API classes that handle SMTP addresses and message formats.

A sample Program Initiator for e-mail push is illustrated in Example 9-3.

Example 9-3 MyMailPush.java

```
import java.net.*;
import java.io.*;

import com.ibm.wireless.push.*;
public class MyMailPush {
    /** Main method.<p>
     *
     * @param arg0 the IP Address to send the message - mail server
     * @param arg1 the IP Address of the messaging gateway
     * @param arg2 the sender.
     * @param arg3 text   */

    public static void main(String args[]) {

        String pushAddr = null;   // receiver IP Addresss
        String message = null;    // message text
        String ppgURL = null;     // URL of messaging gateway

        /**************************************************************
         * Process input arguments
         **************************************************************/
        pushAddr = args[0];
        message =args[3];     //  text
        ppgURL = "http://" + args[1] + ":13131" ;   // build Messaging GW URL

        /**************************************************************
         * Compose and submit message
         **************************************************************/
        Pusher pusher = null;
        PushMessage msg = null;
        SMTPPushAddress addr = null;
        PushResponse rsp = null;
        try {
            /**************************************************************
             * Create message
             **************************************************************/
            msg = new PushMessage();
            msg.setContent(message, "text/plain");
            msg.setFrom(args[2]);

             /* Ask for progress notes */
            msg.notifyPushProgress(true);

            System.out.println(msg);
            System.out.println
             ("-------------------------------------------------");
            System.out.println(message);
            /************************************************************
             * Create an address to send the message to
             ************************************************************/
            addr = new SMTPPushAddress(pushAddr);
            addr.addToRecipient(args[4]);   // user e-mail address
           System.out.println(addr);

            /************************************************************
             * Submit message
             ************************************************************/
            /* Instantiate a pusher */

                pusher = new Pusher(ppgURL, null);

            /* Push the message */
            rsp = pusher.push(msg, addr);
        } catch(PushException e) {
            System.err.println(e);
            System.exit(1);
        }

        /************************************************************
         * Check status reported
         ************************************************************/
        System.out.println(rsp);
        if (rsp.getStatusClass() != StatusCodes.SUCCESS) {
            System.out.println("Push submission failed!");
            System.exit(1);
        } else {
            System.out.println("Push submission succeeded!");
            System.exit(0);
        }
    }
    private static void usage() {
        System.out.println("Usage:\n\tjava
com.ibm.wireless.push.samples.SimplePush <address> <message> [<ppg-URL>]");
        System.out.println
          ("\twhere <address> is something like \"195.153.199.30\"");
        System.out.println("\t      <message> text ");
        System.out.println
("\t      <ppg-URL> is the URL of the PPG; if this argument is not\n\t
specified, the system property \"push.proxy-url\" must be set;\n\t      e.g.,
\"push.proxy-url=http://ppg.ibm.com:12345/\"");
        System.exit(1);
    }
}
```

9.8 Pushing from a servlet

Many enterprise applications are user-driven, the most common interface being the Web browser. Servlets built with the Java messaging API could be used to give pushing capabilities to an enterprise application.

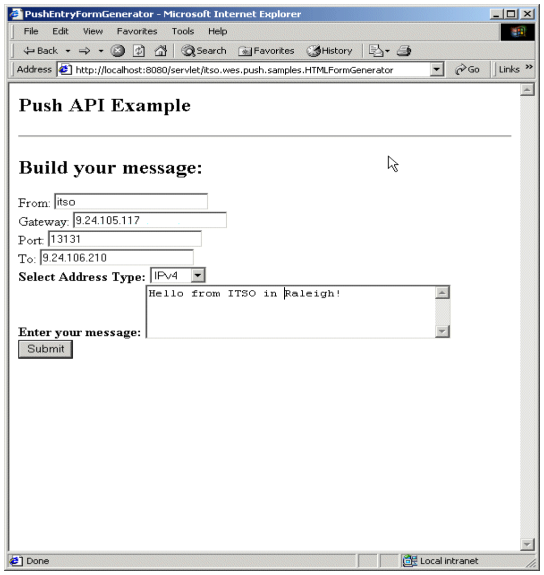
Example 9-4 below illustrates a simple method for pushing messages from a servlet. In this example, the information required to build a push context and send a message is provided in the servlet request parameter.

Example 9-4 Push messages from a servlet

```
package itso.wes.push.samples;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.wireless.push.*;

public class PushServlet extends HttpServlet {

/**
 *
 * Creation date: (9/28/2001 9:40:58 AM)
 */
public void doGet(HttpServletRequest req, HttpServletResponse res) throws 
IOException{

   performTask(req,res);

   }
/**
 * Initializes the servlet.
 */
public void init() {
   // insert code to initialize the servlet here

}
/**
 *
 * Creation date: (9/28/2001 9:40:58 AM)
 */
public void performTask(HttpServletRequest req, HttpServletResponse res) 
throws IOException{

   res.setContentType("text/plain");
        PrintWriter out = res.getWriter();
 String addresstype = null;
 String pushAddr = null;   // client address
 String message = null;    // message text
 String ppgURL = null;     // Full URL of messaging gateway
 String gateway = null;    // gateway address
 String port    = null;
 String from    = null;


 from    = req.getParameter("from");
 addresstype = req.getParameter("addresstype");
 pushAddr = req.getParameter("to");
message =  req.getParameter("Body");
gateway = req.getParameter("gateway");
port    = req.getParameter("port");

ppgURL = "http://" + gateway + ":" + port;

   Pusher pusher = null;
   PushMessage msg = null;
   PushResponse rsp = null;


   try{

   if(addresstype.equals("ipv4"))
      {
         msg = new PushMessage();
                    msg.setContent(message, "text/vnd.wap.wml");
                    msg.setFrom(from);
                    pusher = new Pusher(ppgURL, null);
         IPv4PushAddress addr = null;
         addr = new IPv4PushAddress("pi.ibm.com");
         addr.setAddress(pushAddr);
         rsp = pusher.push(msg, addr);  //Push the message

      }
   }
   catch(PushException e)
      {
         String error = "Push Exception Error";
         out.println(error);

      }
      out.println(rsp);

     if (rsp.getStatusClass() != StatusCodes.SUCCESS)
   {
```

```
               out.println("Push submission failed!");

           } else {
               out.println("Push submission succeeded!");

           }
      }
   }
```

One way to pass push information to a push servlet is by using an HTML form (see Figure 9-13 on page 350). This will have a POST action parameter set to the URL of the push servlet.


PushEntryFormGenerator - Microsoft Internet Explorer

Address http://localhost:8080/servlet/itso.wes.push.samples.HTMLFormGenerator

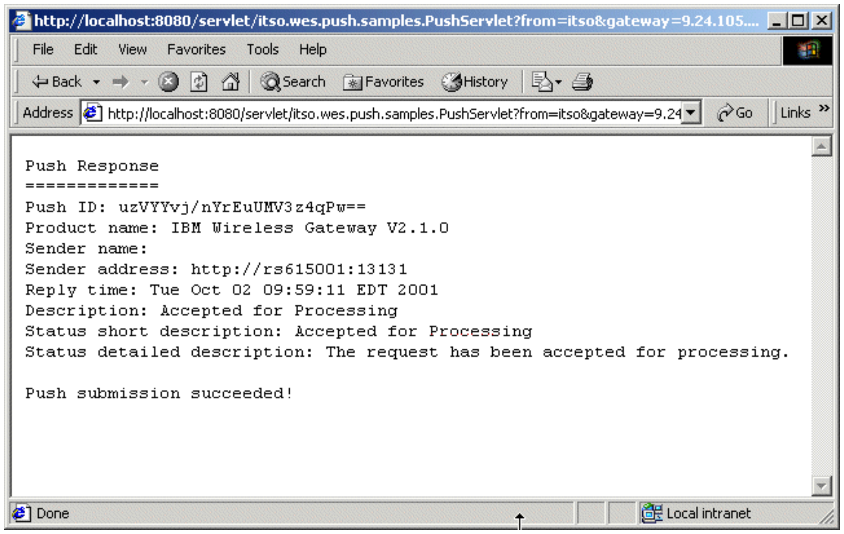
Push API Example

Build your message:

From: itso
Gateway: 9.24.105.117
Port: 13131
To: 9.24.106.210
Select Address Type: IPv4
Enter your message: Hello from ITSO in Raleigh!
Submit

Figure 9-13 Push submission HTML form

The results of the push submission are then displayed in the browser window.


http://localhost:8080/servlet/itso.wes.push.samples.PushServlet?from=itso&gateway=9.24.105....
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8080/servlet/itso.wes.push.samples.PushServlet?from=itso&gateway=9.24
Go Links
Done
Local intranet

```
Push Response
=============
Push ID: uzVYYvj/nYrEuUMV3z4qPw==
Product name: IBM Wireless Gateway V2.1.0
Sender name:
Sender address: http://rs615001:13131
Reply time: Tue Oct 02 09:59:11 EDT 2001
Description: Accepted for Processing
Status short description: Accepted for Processing
Status detailed description: The request has been accepted for processing.

Push submission succeeded!
```

Figure 9-14 Results of push response from messaging gateway

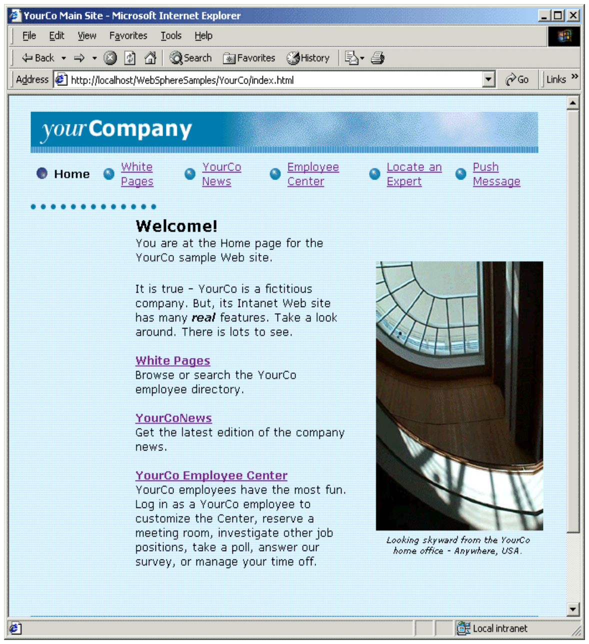
Information about developing servlets can be obtained from the redbook *Servlet and JSP Programming with IBM WebSphere Studio and VisualAge for Java,* SG24-5755.

9.9 Extending the YourCo sample application

Using the PushServlet, you can extend the YourCo sample application to provide push messaging capabilities using the Messaging Gateway of the Everyplace Wireless Gateway.

The source code is similar to the one shown in Example 9-4. A link was added on the index page of the YourCo sample application that would provide the push capabilities, as shown in Figure 9-15.

Note that in order for a servlet that uses Push Messaging APIs to be deployed in the WebSphere Application Server environment, you *must* ensure that the file push.properties is in the working directory, as described in “Configuration” on page 335. Moreover, in the push.properties file the property push.pap-dtd-uri must point to a valid pap_1.0.dtd file. For example, if push.pap-dtd.uri=./pap_1.0.dtd, it means that the pap_1.0.dtd file is located in the same directory as the file push.properties.


YourCo Main Site - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost/WebSphereSamples/YourCo/index.html
Go Links
yourCompany
Home
White Pages
YourCo News
Employee Center
Locate an Expert
Push Message
Local intranet

Welcome!
You are at the Home page for the YourCo sample Web site.

It is true - YourCo is a fictitious company. But, its Intanet Web site has many ***real*** features. Take a look around. There is lots to see.

White Pages
Browse or search the YourCo employee directory.

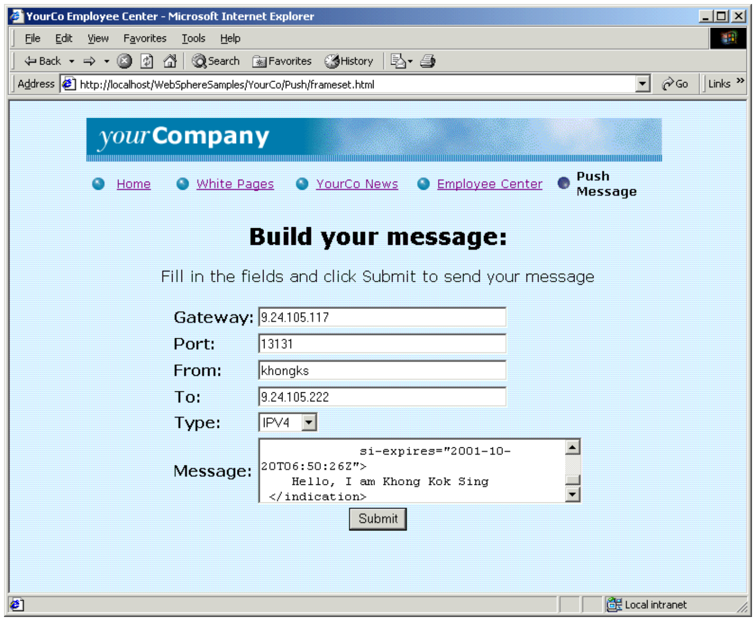
YourCoNews
Get the latest edition of the company news.

YourCo Employee Center
YourCo employees have the most fun. Log in as a YourCo employee to customize the Center, reserve a meeting room, investigate other job positions, take a poll, answer our survey, or manage your time off.

Looking skyward from the YourCo home office - Anywhere, USA.

Figure 9-15 Providing a link to Push Message APIs

We also have a page that provides fields to help you build a push message, as shown in Figure 9-16.


YourCo Employee Center - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost/WebSphereSamples/YourCo/Push/frameset.html
Go Links
yourCompany
Home White Pages YourCo News Employee Center Push Message
Local intranet

Build your message:

Fill in the fields and click Submit to send your message

Gateway: 9.24.105.117
Port: 13131
From: khongks
To: 9.24.105.222
Type: IPV4
Message:

```
          si-expires="2001-10-
20T06:50:26Z">
    Hello, I am Khong Kok Sing
 </indication>
```

Submit

Figure 9-16 Building a message

9.10 Problem determination

Typically, there are problems with the development of a messaging solution. To effectively troubleshoot these problems, each tier of the application should be analyzed separately. As a developer, the PI and its client application, as well as the client device, will be directly under your control. Accessing the gateway will require the cooperation of your administrator.

Messaging application (PI)

Build and test the PI component of the messaging application separately. Then ensure it runs correctly before integrating it with the larger messaging application. In the development version of your PI, use of the ResultNotification classes will provide you information about the status of a message submission to the gateway, *after* processing. This is quite different from the initial push response which tells the PI if the submission was accepted or not.

The status code will be one of the following:

- SUCCESS: message or response received and accepted
- CLIENT ERROR: syntax error in the request
- SERVER ERROR: valid request not fulfilled by the messaging gateway
- SERVICE FAILURE: service could not be performed
- MOBILE CLIENT ABORT: returns abort codes of mobile device to PI

Note that the results the EWG returns in response to the status queries do not have to go to the PI sending the queries. A URL may be specified at the time the notification response class is being constructed, where the gateway may send the status reports.

The following code segment handles the status code returned by the messaging gateway:

```
.....
System.out.println(resp);
if (resp.getStatusClass() != StatusCodes.SUCCESS) {
System.out.println("Push submission failed!");
System.exit(1);
} else {
System.out.println("Push submission succeeded!");
System.exit(0);
}
.....
```

The development environment should include all the required packages or your program will not run. If you are using an integrated development environment such as VisualAge for Java, the core API package com.ibm.wireless.push contained in the push.jar file must be in the project. Also, the xerces.jar, log.jar and (for security) ssl.jar files must be present.

Note: When using VisualAge for Java for development, make sure you do not have Web Traffic Express feature added to the current workspace. This avoids the problem of certain packages contained in the xerces.jar clashing with identical packages in Web Traffic Express when importing the .jar file. Alternatively, you may proceed with the import with Web Traffic Express still in the workspace, and delete the clashing packages from the .jar file flagged by the import operation.

Messaging gateway

Use the log files from the Wireless Gateway here. Contact the administrator of the Everyplace Wireless Gateway you are using as your PPG to obtain a copy of the log file, named wg.log, generated within the period the test applications were run. Scroll through the EWG log until you locate the entries corresponding to your test session (the IP address of your test machine is a good search parameter to use in searching the log file).They show the application developer the packet arrivals at the gateway within the logging interval (set by the gateway administrator). With this information, you can see if your push message:

– Reached the gateway

– Was accepted by the gateway

– Was transmitted by the gateway to the target address using the specified bearer network.

The log file is a comprehensive record of the gateway’s behavior over a given interval. Familiarization with the format of its entries provides valuable troubleshooting information.

Example 9-5 A section of the wg.log file showing the log entry for a WAP Push message

```
---------------------------------
  2954:18001 (Sep 07 01/15:21:32): Messaging GW rcvd post from client
9.24.106.131
  2954:18001 (Sep 07 01/15:21:32): received HTTP/PAP header: [192]
                                   post / http/1.1

date: Fri, 07 Sep 2001 19:33:15 UTC

content-type: multipart/related; boundary="bAnE9GbCZj4bLTbzJBQkYg==";
type="application/xml"

host: 9.24.105.117

content-length: 614

---------------------------------
  2954:18001 (Sep 07 01/15:21:32): tag: pap 1 4
  2954:18001 (Sep 07 01/15:21:32): token: 1 pap
  2954:18001 (Sep 07 01/15:21:32): tag: product-name 126 17
  2954:18001 (Sep 07 01/15:21:32): token: 126 product-name
  2954:18001 (Sep 07 01/15:21:32): token: 3000 =
  2954:18001 (Sep 07 01/15:21:32): token: 2000 Java Push API, Copyright IBM,
2000, 2001
  2954:18001 (Sep 07 01/15:21:32): tag: push-message 2 75
  2954:18001 (Sep 07 01/15:21:32): token: 2 push-message
  2954:18001 (Sep 07 01/15:21:32): tag: push-id 100 83
  2954:18001 (Sep 07 01/15:21:32): token: 100 push-id
  2954:18001 (Sep 07 01/15:21:32): token: 3000 =
  2954:18001 (Sep 07 01/15:21:32): token: 2000 bAnE9GbCZj4bLTbzJBQkYg==
  2954:18001 (Sep 07 01/15:21:32): tag: progress-notes-requested 105 135
  2954:18001 (Sep 07 01/15:21:32): token: 105 progress-notes-requested
  2954:18001 (Sep 07 01/15:21:32): token: 3000 =
  2954:18001 (Sep 07 01/15:21:32): token: 2000 false
  2954:18001 (Sep 07 01/15:21:32): tag: address 13 153
  2954:18001 (Sep 07 01/15:21:32): token: 13 address
  2954:18001 (Sep 07 01/15:21:32): tag: address-value 106 167
  2954:18001 (Sep 07 01/15:21:32): token: 106 address-value
  2954:18001 (Sep 07 01/15:21:32): token: 3000 =
  2954:18001 (Sep 07 01/15:21:32): token: 2000
wappush=9.24.106.131/type=IPv4@pi.ibm.com
  2954:18001 (Sep 07 01/15:21:32): token: 3001 />
  2954:18001 (Sep 07 01/15:21:32): tag: push-message 1002 228
  2954:18001 (Sep 07 01/15:21:32): token: 1002 push-message
  2954:18001 (Sep 07 01/15:21:32): tag: pap 1001 235
  2954:18001 (Sep 07 01/15:21:32): token: 1001 pap
  2954:18001 (Sep 07 01/15:21:32): parsed PAP header ok...
  2954:18001 (Sep 07 01/15:21:32): ppg_db_enter_push_msg:
bAnE9GbCZj4bLTbzJBQkYg==: ok
  2954:18001 (Sep 07 01/15:21:32): push_response:
                                   HTTP/1.1 202 PPG reply

Date: Fri Sep  7 19:21:32 2001

Server: IBM Wireless Gateway V2.1.0

Content-Type: application/xml

Content-Language: en

Content-Length: 397

<?xml version="1.0"?>

  <!DOCTYPE pap PUBLIC "-//WAPFORUM//DTD PAP 1.0//EN"

          "http://www.wapforum.org/DTD/pap_1.0.dtd">
```

```
<pap product-name="IBM Wireless Gateway V2.1.0">

  <push-response push-id="bAnE9GbCZj4bLTbzJBQkYg=="
sender-address="http://rs615001:13131" reply-time="2001-09-07T19:21:32Z">

    <response-result code="1001" desc="Accepted for Processing"/>

  </push-response>

</pap>

 2954:18001 (Sep 07 01/15:21:32): freeing msg --no pid-- type: 0 @[31c0c108]
thread: 4651 state: 0
  2954:18001 (Sep 07 01/15:21:32): ppg_client_read: enqueue non-delayed
push-id: bAnE9GbCZj4bLTbzJBQkYg==
  2954:17477 (Sep 07 01/15:21:32): ppg_worker: dequeued push-id:
bAnE9GbCZj4bLTbzJBQkYg==
  2954:17477 (Sep 07 01/15:21:32): ppg_msg_lock: 31c0bbb8 4445
  2954:17477 (Sep 07 01/15:21:32): ppg_handle_push_msg:
bAnE9GbCZj4bLTbzJBQkYg==
  2954:17477 (Sep 07 01/15:21:32): ppg_handler: transformed HTTP hdr:
                                   content-type: text/plain ; charset=UTF-8
content-language: en
x-wap-application-id: 2

content-length: 14
```

Client devices

Troubleshooting methods at the client level depend on the intended message targets. For this redbook, e-mail and WAP clients were used to receive push messages; a few problems to look out for are:

- In the SMTP configuration of the messaging gateway, misdirected messages do not reach the client.

  Solution: ensure that the SMTP MNC of the Wireless Gateway has an Originator Address configured (for example, root@rs615003.itso.ral.ibm.com).

- There is a problem with the format of the WML file being sent to the emulator.

  Solution: obtain an emulator that can decode the message type your application will be using. The Nokia toolkit used for this redbook will handle WAP messages only. It is a good idea to avoid making assumptions about the capabilities of the client device.

10

Intelligent Notification Services (INS)

This chapter provides an overview, description and sample scenarios of Intelligent Notification Services (INS), provided in IBM WebSphere Everyplace Server (WebSphere Everyplace Server) Service Provider Offering (SPO) Version 2.1.1.

In this chapter, you will find information about how to set up Intelligent Notification Services in a WebSphere Everyplace Server environment, how to work with user preferences, and how to develop applications using simple notifications and subscription applications. You will also find extensions to a B2E sample application (YourCo application) illustrating the use of simple notifications and subscriptions.

Note: INS is not available in WebSphere Everyplace Server Enable Offering (EO) Version 1.1.

10.1 Overview

Intelligent Notification Services (INS) is a new function provided in WebSphere Everyplace Server Version 2.1.1. It is aimed at sending notifications to pervasive users, depending on the users' preferences, which define how they want the notifications to be delivered. A user can register triggers, and INS will notify the user when the event defined in the trigger occurs. The event will be a condition of data that exists in content, scanned by INS from external content providers. INS has different ways of delivering the notifications: SMS, e-mail, Instant Messaging (SameTime) and WAP Push.

Examples of how INS can be used for notification are:

- Notify the user when IBM Stock hits 150 points.
- Notify the user when there is news on WebSphere Everyplace Server from IBM, but do not send the entire document. Just save it and sendthe user a reference so that he/she can browse it when it suits him/her.
- Send urgent messages as WAP Push to the user's cell phone, and other messages as e-mail.

INS can also be used to push messages to users directly, without triggers, to their preferred device. Figure 10-1 shows a schematic overview of INS.

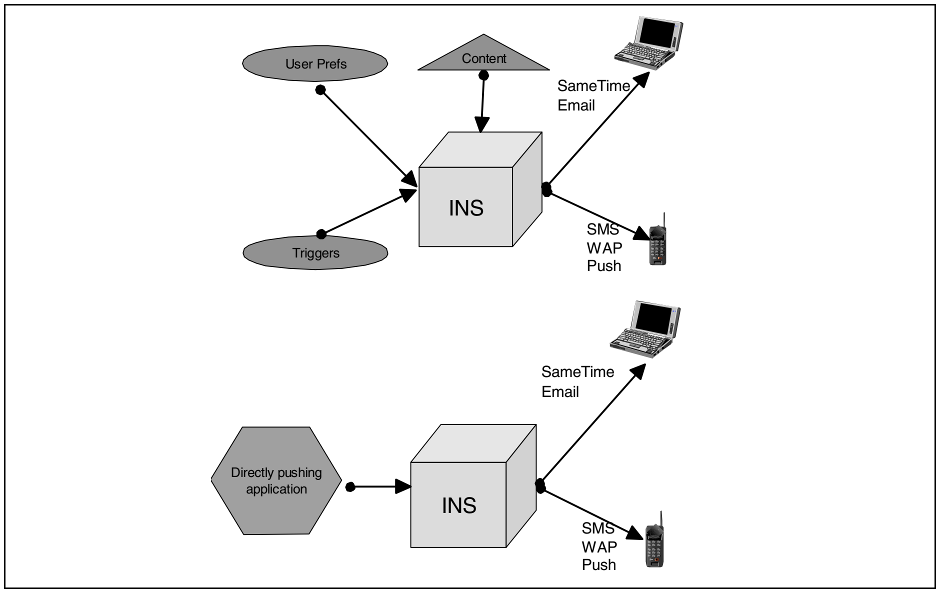


Figure 10-1 Schematic overview of the two modes of operation of INS

User preferences and triggers are maintained using a Web browser. An application that wants to directly notify users may use the INS API, which will be discussed later.

10.1.1 INS in WebSphere Everyplace Server

INS is tightly integrated in WebSphere Everyplace Server. It uses WebSphere Everyplace Server components for security (WSL), user preferences (TPSM), delivery of the notifications (Wireless Gateway) and the underlying LDAP (SWD) for storage of its preferences and user information. In Figure 10-2, a schematic overview of INS and related WebSphere Everyplace Server components is shown.

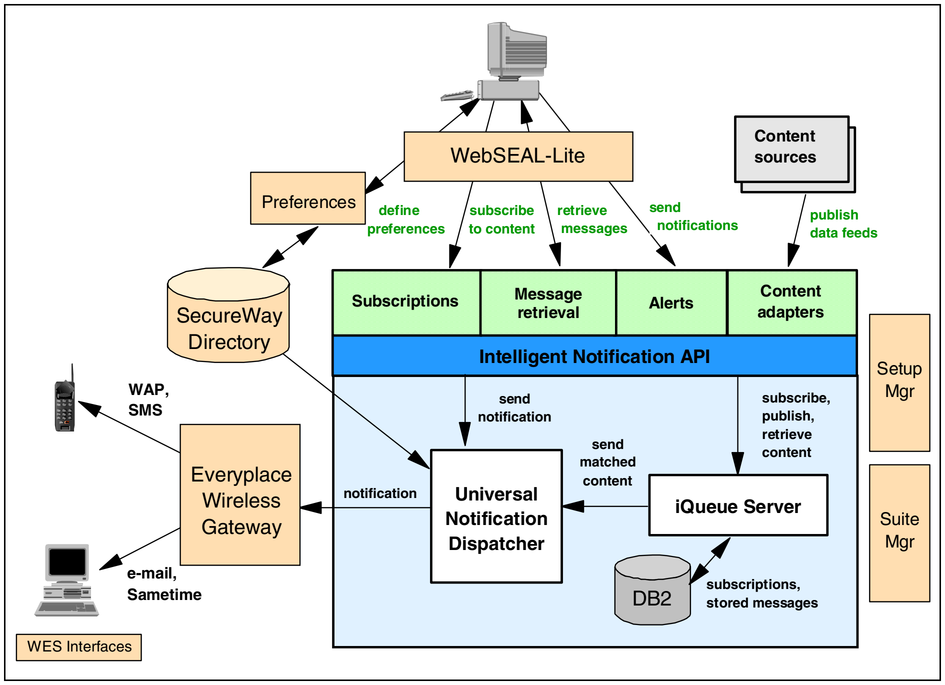


Figure 10-2 Intelligent Notification Services in WebSphere Everyplace Server

WebSEAL-Lite

WebSea-Lite is the security component of WebSphere Everyplace Server. It provides authentication and authorization facilities for access to the user preferences pages of INS, the Web pages to register their devices, preferences and the triggers.

IBM SecureWay Directory

IBM SecureWay Directory contains both the configuration information for INS and for the user preferences. There is one centralized instance of IBM Secureway Directory for the installation of WebSphere Everyplace Server.

Tivoli PSM

User preferences are managed using Tivoli Personalized Service Manager (TPSM). TPSM will update the directory with the user information, from which the INS services read that information. TPSM is presented to the end-users as a set of Web pages where the users can register and manage their devices and preferences.

EveryPlace Wireless Gateway

The EveryPlace Wireless Gateway is used to deliver notifications to the mobile devices of the users. It is used for delivering SMS messages and WAP Push messages. For Instant Messaging and e-mail, EWG is not used.

10.1.2 User preferences

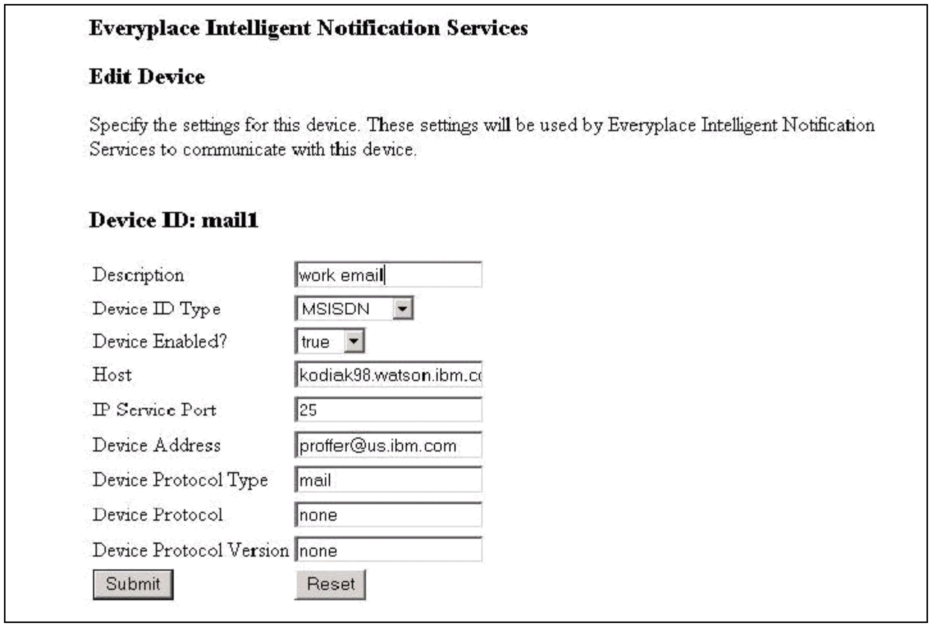
User preferences define the settings and preferences for individual users. There are three categories of preferences

User profile

The user profile contains information about a user and the associated user ID. It also contains the device profiles and group profiles that are associated with the user.

Device profile

The device profile provides information about a device the user wants to use for a notification by a specific protocol. It has information on the type of device, the protocol version, device address and information on the gateway that is used for this device. Figure 10-3 on page 363 shows what the user interface looks like when the device is registered through TPSM.


Everyplace Intelligent Notification Services

Edit Device

Specify the settings for this device. These settings will be used by Everyplace Intelligent Notification Services to communicate with this device.

Device ID: mail1

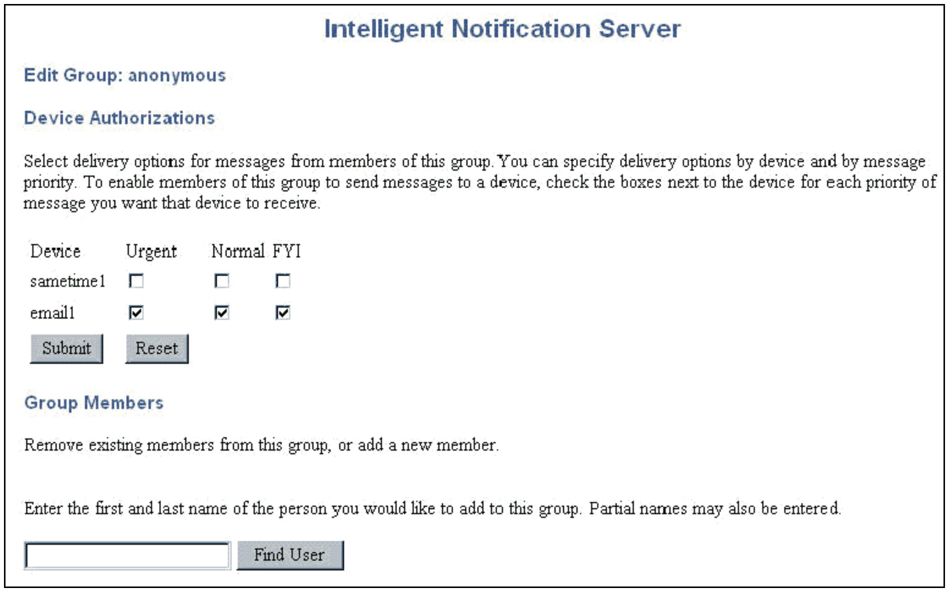
Description	work email
Device ID Type	MSISDN
Device Enabled?	true
Host	kodiak98.watson.ibm.c
IP Service Port	25
Device Address	proffer@us.ibm.com
Device Protocol Type	mail
Device Protocol	none
Device Protocol Version	none

Figure 10-3   Creating a device profile for a user for INS

Group profile

The group defines how notifications, originating from members of the group, are delivered to the user for the different priorities that are defined. These priorities are Urgent, Normal, and FYI (for your information). For each of these, a delivery mechanism can be set up, which allows delivery of notifications with different priorities to different devices (for example, urgent notifications are delivered to the mobile phone as WAP Push messages and normal notifications are delivered as e-mail).

Figure 10-4 shows how to define a group in the example. All notifications, Urgent, Normal and FYI, are sent to e-mail. There is always a group called Anonymous, which is the default group. This group is used for all users that do not belong to any other group. Note that the users are not really anonymous, since notifications can only be sent by users who are registered in the directory.


Intelligent Notification Server

Edit Group: anonymous

Device Authorizations

Select delivery options for messages from members of this group. You can specify delivery options by device and by message priority. To enable members of this group to send messages to a device, check the boxes next to the device for each priority of message you want that device to receive.

Device	Urgent	Normal	FYI
sametime1	☐	☐	☐
email1	☑	☑	☑

Group Members

Remove existing members from this group, or add a new member.

Enter the first and last name of the person you would like to add to this group. Partial names may also be entered.

Figure 10-4 Creating a group for INS users

In Section 10.3, “Users and user preferences without TPSM” on page 380, you will find how to set up users and user preferences without the use of TPSM. Instead, you will work directly with the entries that are in the Secureway Directory. This is more convenient in a development environment.

10.1.3 INS end to end flow

There are several steps involved in using INS in a production system. In this section, you will find a detailed explanation of how requests, triggers and notifications are handled by the INS system.

Figure 10-5 shows the different flows of requests, triggers and notifications in INS. Next, you will find how the data for the different functions flows through the system.

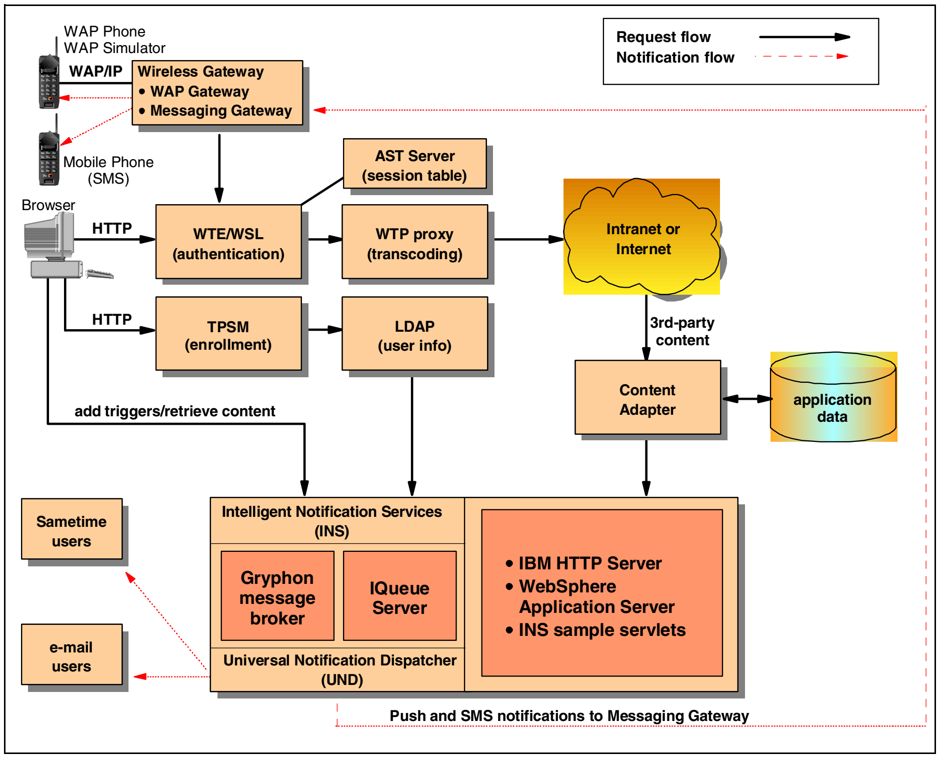


Figure 10-5 Request and notification flow in INS

Authentication

Every HTTP request (originating from a browser or a pervasive device or voice) is proxied by WTE, WebSEAL-Lite, as a plug-in of WTE will ensure that every request that flows on is authenticated. The Wireless Gateway will authenticate requests that come in from a non-HTTP origin. For these protocols, WSL will pass on the request.

Enrollment

The first step is enrollment of the user using TPSM. The user and device information that is entered is passed on to the central LDAP server for INS servers to access.

Trigger registration

A user registers a trigger through a set of JSP and servlet Web page served from the private IBM HTTP Server/WebSphere Application Server configuration that comes with INS. These triggers are stored in the private DB2 database of INS. In the examples that are shipped with INS, the servlets and JSPs are designed for access from a browser, but they could very well be designed for access from other devices.

Content provision

The content is fed in to INS by a content adapter, which uses the INS API to feed content into the system. Typically the content adapter will retrieve information from a third party content provider such as a news agency, or from a proprietary database. The Content Adapter has full freedom on how to obtain the information. It can be polling a server for information, or some triggering mechanism from the content providing server to the content adapter may be in effect. The content adapter processes the content, and feeds it into INS.

Sending a notification

When INS receives the content, it determines which users need to be notified. Next, it checks the LDAP for the user preferences, and sends the notification through the channel selected by the user. To this end, the notification is sent to the appropriate gateway adapter. This will then construct the right kind of markup and send the notification to the appropriate server. If the user has elected to save the message, the document is saved in the database (the private instance of DB2 that is installed with INS), and a link to it is sent to the user.

Retrieving the content

When the notification has arrived at the client device, the user can either read the message or follow the link to the content in the INS database. The content is then served through a servlet that resides on the INS installed WebSphere Application Server.

10.2 Setting up the system

For development of applications that use INS applications, you must install INS. There is no provision for emulating the INS servers, such that you could unit test your application without installing the INS system. This section discusses the installation and configuration of INS and the users needed for testing your application.

10.2.1 Installation

Installation of INS is performed using Setup Manager. You must follow the instructions in the WebSphere Everyplace Server InfoCenter for installation of INS.

When you select INS to be installed on a system, Setup Manager checks the prerequisites. When not available, SetupManager will also install:

- IBM Secureway Directory client
- IBM DB2
- IBM HTTP Server
- IBM WebSphere Application Server Advanced Edition
- IBM WebSphere Everyplace Server Intelligent Notification Services

Although TPSM and WebSEAL-Lite (WSL) are used in a production environment, for development these services are not necessary, and they are not discussed in this chapter. We will show later on how the functionality these services provide can be simulated, or provided in a way that is suitable for testing purposes.

IBM SecureWay Directory

IBM SecureWay Directory contains the configuration information for INS and for the users. There is one centralized instance of IBM Secureway Directory for the installation of WebSphere Everyplace Server, so mostly the Secureway Directory will be on a different machine from INS. In that case, the LDAP client is installed. The information in the directory can be accessed using the EveryPlace Suite Manager (the configuration GUI for WebSphere Everyplace Server) as far as INS configuration is concerned, or using DMT, the Directory Management Tool, that ships with IBM SecureWay Directory.

IBM DB2

DB2 is used in this setup as a private installation for INS. It hosts the database INS uses for the triggers and the content, and the database for WebSphere Application Server. It may also host the databases for Secureway Directory if you decide to install Secureway locally on the same system. You do not usually let it host application data in a production environment.

IBM HTTP Server

IBM HTTP Server is used together with the Application server (WAS) to host the Web pages for registration of the triggers, and for serving the saved content of notifications.

Important: Do not configure IBM HTTP Server to run under the root ID. For security reasons, it cannot work under the root ID.

IBM WebSphere Application Server Advanced Edition

WebSphere Application Server is used to host the servlets and JSPs that are used for trigger registration and serving of saved content of notifications.

Suite Manager installs a server called INSSampleServer in your WebSphere Application Server and in that a Web Application called INSSample. The configuration for this server is:

Table 10-1 Configuration of the samples in WebSphere Application Server

setting	value
startup parameters of the Server	-classpath /usr/IBMEPS/INS/lib/ins.jar:/usr/IBMEPS/INS/lib/jlog.jar:/usr/IBMEPS/INS/lib/xalan.jar:/usr/IBMEPS/INS/lib/xerces.jar:/usr/IBMEPS/INS
Log files	/usr/IBMEPS/INS/samples/inssample/stdout.txt /usr/IBMEPS/INS/samples/inssample/stderr.txt
Web appl. classpath	/usr/IBMEPS/INS/samples/inssample/servlets
Web appl. Doc Root	/usr/IBMEPS/INS/samples/inssample/web
Servlets	News: NewsSubscriptionServlet Stock: StockSubscriptionServlet Weather: WeatherSubscriptionServlet content: ContTransServlet Standard Servlets Invoker JSP 1.1 file ErrorReporter

Attention: In this chapter, we will assume that the INS services run on an AIX machine.

On AIX, the base directory for INS is:

```
/usr/IBMEPS/INS
```

On Sun Solaris, the base directory for INS is:

```
/opt/IBMEPS/INS/
```

Log files

The following log files are used:

```
/usr/IBMEPS/INS/samples/inssample/stdout.txt
/usr/IBMEPS/INS/samples/inssample/stderr.txt
```

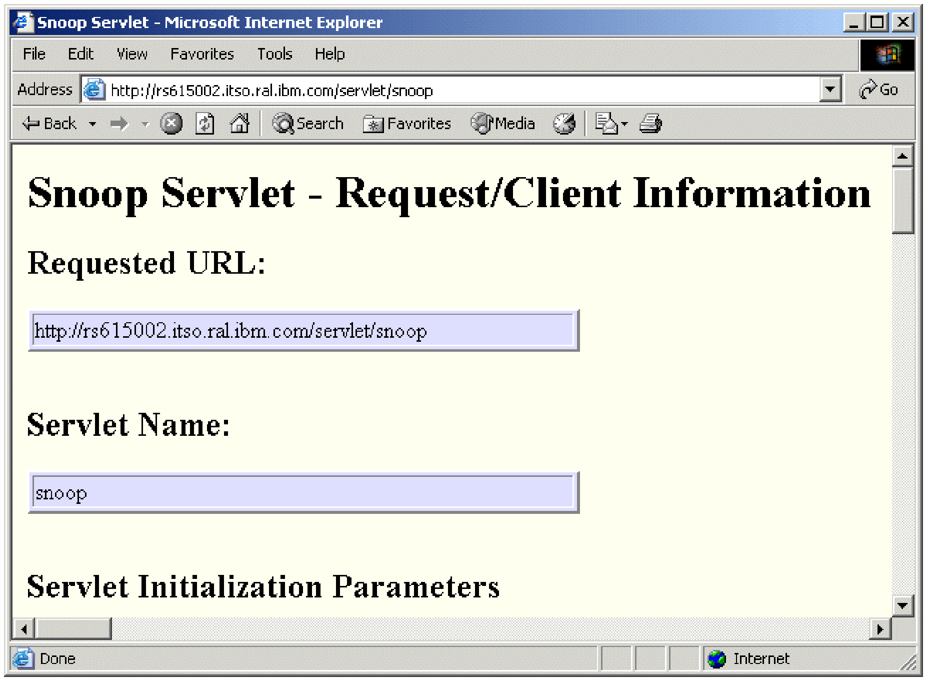
After installation, you can check if the application server is running correctly. This can be done by running the snoop servlet:

1. Ensure that the default server and its servlet engine are running using the WebSphere Application Server administration console.
2. Run the snoop servlet:

   http://<hostname>/servlet/snoop

   Where <hostname> is the name of the server that hosts INS.

   The result should look like Figure 10-6:


Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://rs615002.itso.ral.ibm.com/servlet/snoop
Go
Back Search Favorites Media
Done
Internet

Snoop Servlet - Request/Client Information

Requested URL:

http://rs615002.itso.ral.ibm.com/servlet/snoop

Servlet Name:

snoop

Servlet Initialization Parameters

Figure 10-6 Validation of the Application Server using the snoop servlet.

IBM WebSphere Everyplace Server Intelligent Notification Services

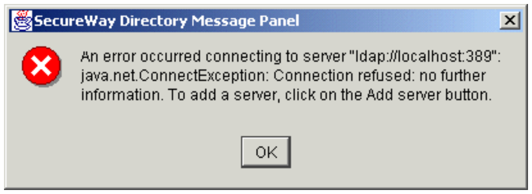
INS contains the core services. They will be discussed in more detail below. For more information on any of these products, you must refer to the WebSphere Everyplace Server Infocenter.

10.2.2 Post-installation configuration

After installing INS, you need to configure it to work correctly in your environment. Apart from setting up the users, which is discussed in 10.3, “Users and user preferences without TPSM” on page 380, you will have to configure INS for connections to the various delivery servers for the notifications. These preferences are in the LDAP Directory, and you can browse and edit them using DMT. DMT can be run from Unix or from Windows. On Unix you can start DMT by entering the following command on the command line:

```
dmt &
```

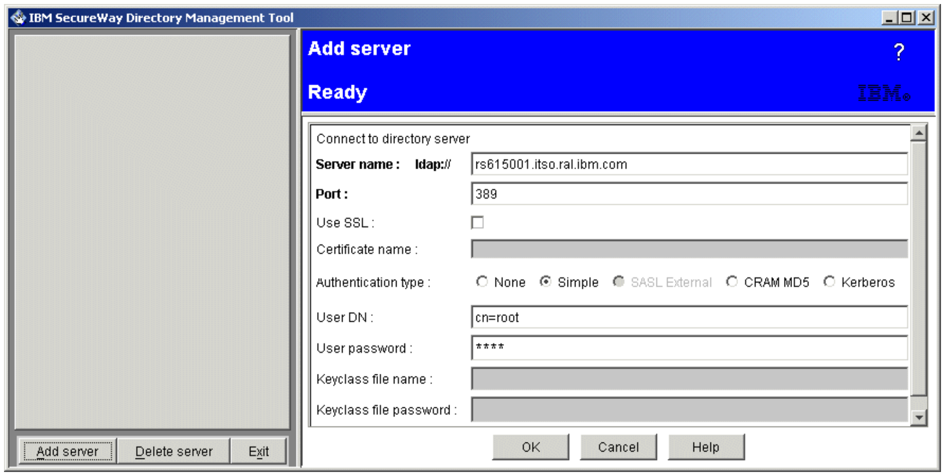
You may get the next window when the server that DMT is configured for is not running:



An error occurred connecting to server "ldap://localhost:389": java.net.ConnectException: Connection refused: no further information. To add a server, click on the Add server button.

Figure 10-7 Error connecting to server

However, this is not a problem, and you can just continue by clicking the **OK** button. Once the main panel of DMT is shown, you can add a server by clicking the **Add server** button:



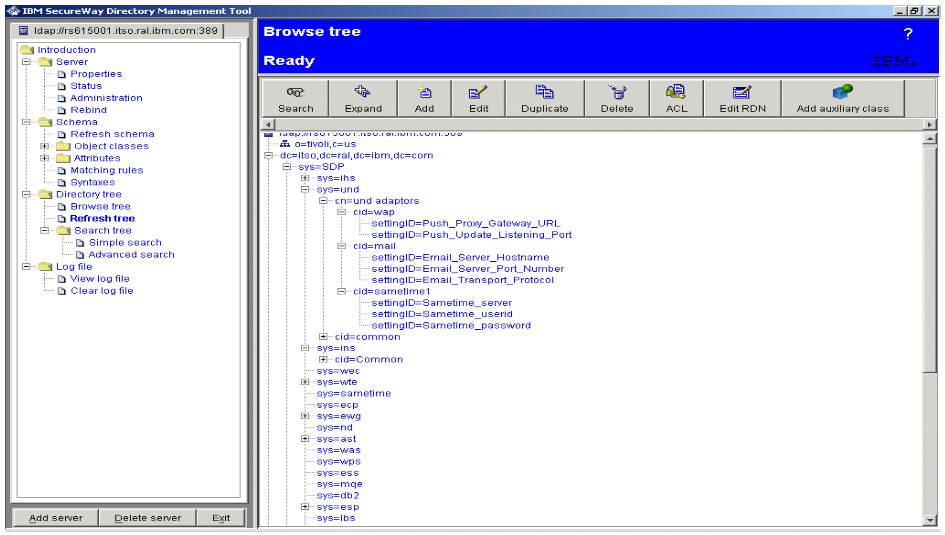
Connect to directory server

Server name : ldap://	rs615001.itso.ral.ibm.com
Port :	389
Use SSL :	☐
Certificate name :	
Authentication type :	○ None [✓ Simple] ○ SASL External ○ CRAM MD5 ○ Kerberos
User DN :	cn=root
User password :	****
Keyclass file name :	
Keyclass file password :	

Figure 10-8 Adding a server to the DMT

You must use a user ID that has the required privileges to access the objects you need to browse or alter. You must enter the DN of the administrator. Here we use `cn=root` as the user ID. Often it is convenient to use the LDAP administration user ID.

After the new server has been successfully added, you can browse the directory tree by selecting **Directory Tree > Browse Tree**. Warnings like `xyz contains no data` can be ignored. By clicking the `+` and `-` symbols, you can collapse and expand a branch of the tree. Under sys=SDP you find the WebSphere Everyplace Server configuration entries. The settings for INS can be found in sys=SDP, sys=und, cn=und adaptors and sys=SDP, sys=und, cid=common.



```
o=tivoli,c=us
dc=itso,dc=ral,dc=ibm,dc=com
  sys=SDP
    sys=ihs
    sys=und
      cn=und adaptors
        cid=wap
          settingID=Push_Proxy_Gateway_URL
          settingID=Push_Update_Listening_Port
        cid=mail
          settingID=Email_Server_Hostname
          settingID=Email_Server_Port_Number
          settingID=Email_Transport_Protocol
        cid=sametime1
          settingID=Sametime_server
          settingID=Sametime_userid
          settingID=Sametime_password
      cid=common
    sys=ins
      cid=Common
    sys=wec
    sys=wte
    sys=sametime
    sys=ecp
    sys=ewg
    sys=nd
    sys=ast
    sys=was
    sys=wps
    sys=ess
    sys=mqe
    sys=db2
    sys=esp
    sys=lbs
```

Figure 10-9 INS settings in LDAP, as viewed with DMT

The branches cid=wap, cid=mail, etc. are the system settings for each of the delivery channels. The branches cid=common under sys=und and sys=ins are the system settings for UND and IQ, respectively. You can edit an object by either double-clicking it or by selecting the line and clicking the **Edit** button.

Configuring the SMTP Mail Server for INS

1. Using DMT, navigate to **sys=SDP, sys=und**. Expand **cn=und** adaptors and then **cid=mail**.
2. Edit settingID=Email_Server_Hostname and specify the fully qualified host name of your SMTP server in the cesProperty field. Click **OK**.
3. Edit settingID=Email_Server_Port_Number and verify that port number 25 is specified. Click **OK**.
4. Edit settingID=Email_Transport_Protocol and specify `smtp` in the cisProperty field. Click **OK**.
5. Stop and restart the UND server.

Tip: For testing purposes, it is often useful to use the SMTP server that is installed with AIX. The user address is <username>@<hostname>, where <username> is an AIX user ID, and <hostname> the host name of the server. The SMTP Server address is then the address of the AIX server.

Configuring the Sametime Server for INS

1. Using DMT, navigate to sys=SDP, sys=und. Expand **cn=und** adaptors and then **cid=sametime1**.
2. Edit settingID=Sametime_server and specify the fully qualified host name of your Sametime server in the cesProperty field. Click **OK**.
3. Edit settingID=Sametime_userid and specify the user ID for accessing your Sametime server in the cesProperty field. Click **OK**.
4. Edit settingID=Sametime_password and specify the password for accessing your Sametime server in the cesProperty field. Click **OK**.
5. Stop and restart the UND server.

Important: When INS delivers notifications to Sametime, it has to log on to the Sametime server. If you use the same user ID here as the user ID you want to deliver the notification to (for example for testing purposes), at the moment the notification is sent, the Sametime client is disconnected from the server, since you can only be logged on once. You must therefore have at least two Sametime user IDs for testing Sametime delivery.

Configuring the WAP Server for INS

1. Using DMT, navigate to **sys=SDP, sys=und**. Expand **cn=und** adaptors and then **cid=wap**.
2. Edit settingID=Push_Proxy_Gateway_URL and specify the URL for the push proxy of the Wireless Gateway in the cesProperty field. Click **OK**.
3. Edit settingID=Push_Update_Listening_Port and specify the port number that the gateway is listening on in the cisProperty field. Click **OK**.
4. Stop and restart the UND server.

Configuring logging and tracing

Logging and tracing output can be directed to the console of the JVM and/or to a file. The console is usually the terminal session in which the process was started.

Under sys=und, cid=common and sys=ins, cid=common, you find the configuration parameters for the logging of INS.

IQ log and trace properties (sys=ins)

- ibm-insSystemLogger.console < true | false >
- ibm-insTracerLogger.console < true | false >
- ibm-insSystemLogger.file < file name >
- ibm-insTracerLogger.file < file name >

UND log/trace properties (sys=und)

- ibm-undConsoleLogOutput < true | false >
- ibm-undConsoleTraceOutput < true | false >
- ibm-undLogFileLocation < file name >
- ibm-undTraceFileLocation < file name >

Pref log/trace properties (sys=pref)

- ibm-prefMessagesLogToConsole < true | false >
- ibm-prefTracesLogToConsole < true | false >
- ibm-prefMessagesLogFile < file name >
- ibm-prefTracesLogFile < file name >

When you change any of the settings for the adapters or logging/tracing, the servers must be restarted for these new settings to take effect.

10.2.3 Starting and stopping the WebSphere Everyplace Server components for INS

The following instructions assume that INS and its prerequisite components have already been installed using the WebSphere Everyplace Server Setup Manager.

DB2

To start DB2, logon to the INS DB2 instance ID (su - insdb2) and enter the following command:

```
db2start
```

To stop DB2, log on to the INS DB2 instance ID (su - insdb2) and enter the following command:

```
db2stop
```

DB2 JDBC listener

To start the DB2 JDBC listener, change the user to the INS instance ID (`su - insdb2)` and enter the following command:

```
db2jstrt 6789
```

Note: 6789 is the default port; the actual port can be found in ibm-insJDBCDatabaseURL in the LDAP Directory.

You can also check that the service has started by entering the following command:

```
ps -ef | grep -i db2jd
```

The result should be similar to the following text:

```
root 16794 21960   0 16:10:29  pts/4  0:00 grep -i db2jd
insdb2 20236     1   0   Sep 12  pts/0  0:00 db2jd 6789
```

If you want to check that DB2 JDBC is listening on the right port, enter the following command:

```
netstat -a | grep -i 6789
```

The result should be similar to the following text:

```
tcp4       0      0  *.6789                *.*                    LISTEN
```

Again, 6789 is the default port used in this scenario.

To stop

Enter the following command:

```
ps -ef | grep db2jd
Result: root 16794 21960   0 16:10:29  pts/4  0:00 grep -i db2jd
```

and kill the processes associated with the INS DB2 instance, in this case 16794:

```
kill -9 16794
```

HTTP server

To start on AIX, from directory /usr/HTTPServer/bin, enter:

```
./apachectl start
```

To stop on AIX, from directory /usr/HTTPServer/bin, enter:

```
./apachectl stop
```

To start on Solaris, from directory /opt/IBMHTTPD/bin, enter:

```
./apachectl start
```

To stop on Solaris, from directory /opt/IBMHTTPD/bin, enter:

```
./apachectl stop
```

WebSphere Application Server

To start WebSphere Application Server on AIX, from directory /usr/IBMWebAS/bin, enter:

```
./startupServer.sh &
```

To start on Solaris, from directory /opt/IBMWebAS/bin, enter:

```
./startupServer.sh &
```

To stop on Solaris and AIX, from the Admin Console, right-click the host node and select **stop**.

WebSphere Application Server administrative console

To start on AIX, from directory /usr/IBMWebAS/bin, enter:

```
./adminclient.sh &
```

To start on Solaris, from directory /opt/IBMWebAS/bin, enter:

```
./adminclient.sh &
```

To start on Windows, from directory <appserver>\bin, enter:

```
adminclient.bat <hostname>
```

where `<appserver>` is the directory where WebSphere Application Server is installed (for example, C:\WebSphere\AppServer) and `<hostname>` is the name of the host of the WebSphere Application Server installation you want to administer.

To stop: from the Admin Console, click **Console** and select **Exit**.

10.2.4 Starting the INS servers

The following procedures will start the INS servers running in the foreground, each in its own window. Unless you have modified the log settings for each server in LDAP, the log output will be displayed in each server's respective window. You can also start the servers in the background. Before you do this, you should modify the log settings so that any output will be written to a file. See Section 10.8, “Problem Determination” on page 433 for instructions on how to change the INS log and trace settings.

Starting the INS Servers in the foreground

Open a new dtterm window, change to the INS bin directory and start the UND, then enter:

```
./startUND <host name>
```

Where `<hostname>` is the name of the server that hosts INS (see note below).

Open a new dtterm window, change to the INS bin directory and start the Gryphon Broker, then enter:

```
./startGB
```

Open a new dtterm window, change to the INS bin directory and start the IQ Server, then enter:

```
./startIQ <host name>
```

The INS bin directory is /usr/IBMEPS/INS/bin on AIX and /opt/IBMEPS/INS/bin on Solaris.

Note: When using the **`startUND`** or **`startIQ`** commands, the format of the hostname argument depends on the ObjectType specified for the directory suffix during SecureWay Directory installation. If the directory suffix is of Domain type, then use a short host name, for example undserver. If the directory suffix is other than Domain, for instance Organizational Unit, then use a fully qualified host name, such as undserver.mysite.myco.com.

Note: When you have tracing enabled, you will see the following exception in the log when UND is started. For example:

```
...

UND started and ready to receive requests

2001.09.14 08:13:58.381
com.ibm.pvc.we.ins.und.server.dispatcher.Connection run

java.io.StreamCorruptedException: Caught EOFException while reading
the stream header

at java.io.ObjectInputStream.readStreamHeader(ObjectInputStream.
java:851)

at java.io.ObjectInputStream.<init>(ObjectInputStream.java:174)

...
```

This is not a problem and can be ignored.

Starting the INS Servers in the background

The INS services can be started in the background by appending **&** to the commands shown above:

```
./startUND <host name> &
./startGB &
./startUND <host name> &
```

Stopping the INS servers

Stopping the INS servers when they are running in the background can be done by closing the terminal session in which they run, or by clicking the escape sequence (**CTRL-C** on AIX).

Stopping the INS servers running in the background can be done from the command line by searching the process that owns them and then killing that process. For example, issue the following command:

```
ps -ef | grep UND
```

with the result similar to the following text:

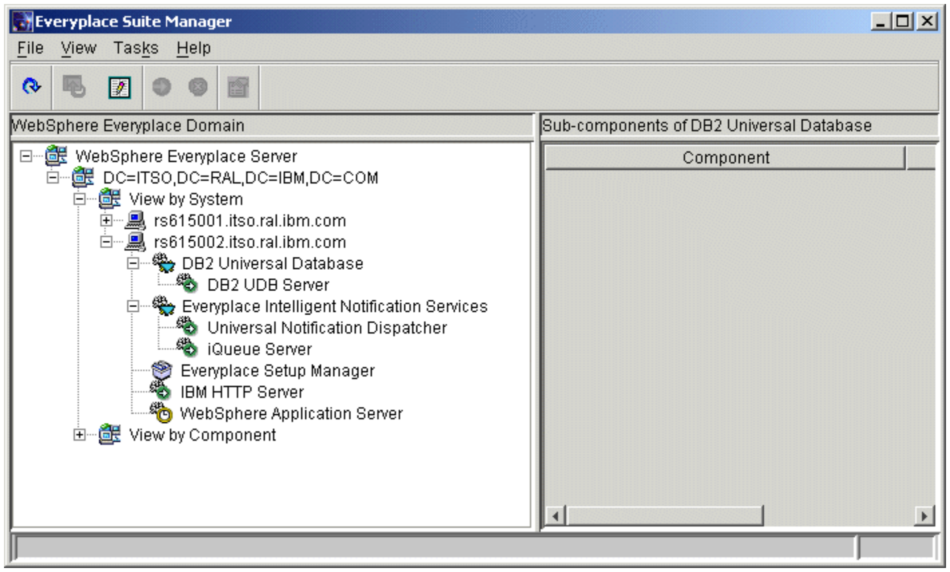
```
root 26926 21960   0 16:45:42  pts/4  0:00 grep UND
root 28756     1   0 16:52:39  pts/0  0:00 ksh /usr/IBMEPS/INS/bin/startUND
rs615002
```

and then you will kill the process (in this case the process number is 28756) by issuing the following command:

```
kill -9 28756
```

Suite Manager

INS applications can also be started and stopped from the Suite Manager. Figure 10-10 shows the Suite Manager console and the INS functions. In the *view by system* branch of the tree, you see the services installed on the server that runs INS (DB2,HTTP Server, WebSphere Application Server, and the INS Components UND and IQueue Server). Note that you can install Suite Manager on a remote machine; it runs also on Windows. From this console, you can start and stop the INS services, by clicking the **Start** or **Stop** buttons


Everyplace Suite Manager
File View Tasks Help
WebSphere Everyplace Domain
Sub-components of DB2 Universal Database
Component

WebSphere Everyplace Server
DC=ITSO,DC=RAL,DC=IBM,DC=COM
View by System
rs615001.itso.ral.ibm.com
rs615002.itso.ral.ibm.com
DB2 Universal Database
DB2 UDB Server
Everyplace Intelligent Notification Services
Universal Notification Dispatcher
iQueue Server
Everyplace Setup Manager
IBM HTTP Server
WebSphere Application Server
View by Component

Figure 10-10   Ins services status overview in Suite Manager

10.3 Users and user preferences without TPSM

One important issue is how to set up users and their preferences without using TPSM for user preference and device management, which will typically be the case in a development environment. You can add users directly to the ldap directory, either manually or by importing an LDIF file. Below, you will find an example LDIF file for a user.

Importing user preferences using LDIF

With an LDIF file you can batch-import information into the directory. Example 10-1 shows an ldif file for importing INS user preferences.

Example 10-1 Example .ldif file for INS user and preferences

```
dn: ou=IBM,dc=itso,dc=ral,dc=ibm,dc=com
objectclass: top
objectclass: organizationalUnit
ou: IBM

dn: cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=com
objectclass: top
objectclass: container
cn: users

# User entries for User Three
dn: uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=com
ibm-tismstatus: C
uid: insuser
objectclass: ibm-SdpUser
objectclass: inetOrgPerson
objectclass: ePerson
objectclass: cimManagedElement
objectclass: eUser
objectclass: organizationalPerson
objectclass: person
objectclass: top
sn: Ins User
cn: Ins User

dn:
cn=insuser/und,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=com
ibm-deviceidlist: cn=mail1/und
ibm-deviceidlist: cn=sametime1/und
ibm-deviceidlist: cn=wap1/und
objectclass: ibm-undUser
objectclass: cimManagedElement
objectclass: eUser
objectclass: top
```

```
ibm-grouplist: cn=group0/und
ibm-grouplist: cn=anonymous/und
cn: insuser/und

dn: cn=anonymous,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=com
objectclass: groupOfNames
objectclass: top
member: uid=user5@IBM
member: uid=user6@IBM
cn: anonymous

dn:
cn=anonymous/und,cn=anonymous,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm
,dc=com
ibm-undauthorizations: None available
objectclass: ibm-undGroup
objectclass: top
ibm-undurgentdevicesauthorizations: mail1,sametime1,wap1
ibm-undfyidevicesauthorizations: mail1,sametime1,wap1
cn: anonymous/und
ibm-undnormaldevicesauthorizations: mail1,sametime1,wap1

dn:
deviceID=mail1,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=com
deviceid: mail1
objectclass: ibm-device
objectclass: cimLogicalElement
objectclass: cimManagedElement
objectclass: cimManagedSystemElement
objectclass: cimLogicalDevice
objectclass: top
ibm-isdeviceenabled: true
ibm-deviceidtype: 1
description: 0

dn: deviceID=wap1,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=com
deviceid: wap1
objectclass: ibm-device
objectclass: cimLogicalElement
objectclass: cimManagedElement
objectclass: cimManagedSystemElement
objectclass: cimLogicalDevice
objectclass: top
ibm-isdeviceenabled: true
ibm-deviceidtype: 1
description: 0

dn:
deviceID=sametime1,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=com
```

```
deviceid: sametime1
objectclass: ibm-device
objectclass: cimLogicalElement
objectclass: cimManagedElement
objectclass: cimManagedSystemElement
objectclass: cimLogicalDevice
objectclass: top
ibm-isdeviceenabled: true
ibm-deviceidtype: 1
description: 0

dn: cn=group0,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=com
objectclass: groupOfNames
objectclass: top
member: uid=usertwo@IBM
member: uid=erongen@IBM
cn: group0

dn:
cn=group0/und,cn=group0,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=co
m
ibm-undnormaldevicesauthorizations: mail1
ibm-undauthorizations: None available
objectclass: ibm-undGroup
objectclass: top
ibm-undurgentdevicesauthorizations: mail1,sametime1
ibm-undfyidevicesauthorizations: mail1
cn: group0/und

dn:
cn=mail1/und,deviceID=mail1,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,d
c=com
ibm-appprotocolversion: none
ibm-appprotocol: none
objectclass: ibm-undDevice
objectclass: top
uid: 0
cn: mail1/und
description: 0
ipserviceport: 25
host: rs615002.itso.ral.ibm.com
ibm-appprotocoltype: mail
ibm-appdeviceaddress: nl69897@rs615002.itso.ral.ibm.com

dn:
cn=wap1/und,deviceID=wap1,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=
com
ibm-appprotocolversion: none
ibm-appprotocol: none
```

```
122  ibm-appdeviceaddress: 9.24.105.209
123  objectclass: ibm-undDevice
124  objectclass: top
125  uid: 0
126  cn: wap1/und
127  description: 0
128  ipserviceport: 0
129  host: pushproxy
130  ibm-appprotocoltype: wap
131
132  dn:
cn=sametime1/und,deviceID=sametime1,uid=insuser,cn=users,ou=IBM,dc=itso,dc=ral,
dc=ibm,dc=com
133  ibm-appprotocolversion: V15
134  ibm-appprotocol: sametime
135  ibm-appdeviceaddress: e_rongen@nl.ibm.com
136  objectclass: ibm-undDevice
137  objectclass: top
138  uid: 0
139  cn: sametime1/und
140  description: 0
141  ipserviceport: 1533
142  host: messaging.ibm.com
143  ibm-appprotocoltype: im
```

Note: The line numbers are not part of the LDIF file.

Explanation of Example 10-1

For an explanation on devices and groups, see Section 10.1.2, “User preferences” on page 362.

- Lines 1-4 define the OU, and lines 6-9 the user’s group that will contain the user defined in this file.
- For each user, there is exactly one instance of class ibm-SdpUser and one of class ibm-undUser. These entries are defined in lines 12-24 and lines 26-36, respectively. The ibm-undUser entry also contains references to the devices cn=mail1/und, cn=sametime1/und and cn=wap1/und. These devices are defined further down in the file. Also, ibm-undUser contains references to the groups that are defined for this user. The Anonymous group must always be present. This is used for notifications by users that are not in any other group.
- Lines 38-43 show the settings for the Anonymous group for this user. Notice that this group has members. The system will not use these member definitions.
- Lines 45-52 show the ibm-undGroup definition for Anonymous. They show the possible delivery channels for Urgent, Normal and FYI notifications.

- Lines 54-88 show the ibm-devices class entry for the delivery channels (WAP, e-mail and Sametime). These entries define whether the device profile is active or not, that is, whether INS can use these devices to deliver notifications.
- Lines 90-104 show another group, as an example.
- Lines 106-117 show the ibm-undDevice class entry for e-mail. The name is referenced in the ibm-device entry for email. The server name, port and the e-mail address (ibm-appdeviceaddress) of the user are defined here.
- Lines 119-130 show the ibm-undDevice class entry for WAP Push notifications. The name is referenced in the ibm-device entry for WAP. In this example, the address is an IP address, referring to a WAP emulator running on a PC. For a real phone, ibm-appdeviceaddress would contain the phone number of the device. The value of the host name is not used. Instead, the values under sys=SDP, sys=und, cn= und adapters, cid=wap are used (see“Configuring the WAP Server for INS” on page 373).
- Lines 132-143 show the ibm-und-Device settings for Sametime. Just as for WAP, the server name is not used. Instead, the values under sys=SDP, sys=und, cn= und adapters, cid=wap are used (see “Configuring the Sametime Server for INS” on page 373).

10.3.1 Importing the LDIF file

You can use the example to load your system with an INS user. To do so, you must make some changes to the file.

Make the following changes to the file:

1. Replace the domain dc=itso,dc=ral,dc=ibm,dc=com with the suffix of your LDAP directory
2. Change ou=IBM with the ou of your users.
3. Change addresses for the adapters.
4. Change the e-mail server address and e-mail address on lines 115 and 117, respectively.
5. Change the WAP Push address on line 122.
6. Change the Sametime user ID on line 135.

Importing the file into your LDAP directory

1. You must upload the LDIF file to the machine that runs IBM Secureway Directory. For example:

   a. Open a DOSbox, change to the directory that contains the file, and enter the following command:

      ```
      ftp <hostname>
      ```

   b. Where **`<hostname>`** is the name of the server to which the file must be uploaded. You must enter a user ID and password. The FTP prompt will appear. Enter the following command:

      ```
      cd <targetdir>
      ```

      where **`<targetdir>`** is the directory to which you want the file uploaded, then:

      ```
      ascii
      put <filename>
      ```

      where `<filename>` is the name of the file, then

      ```
      quit
      ```

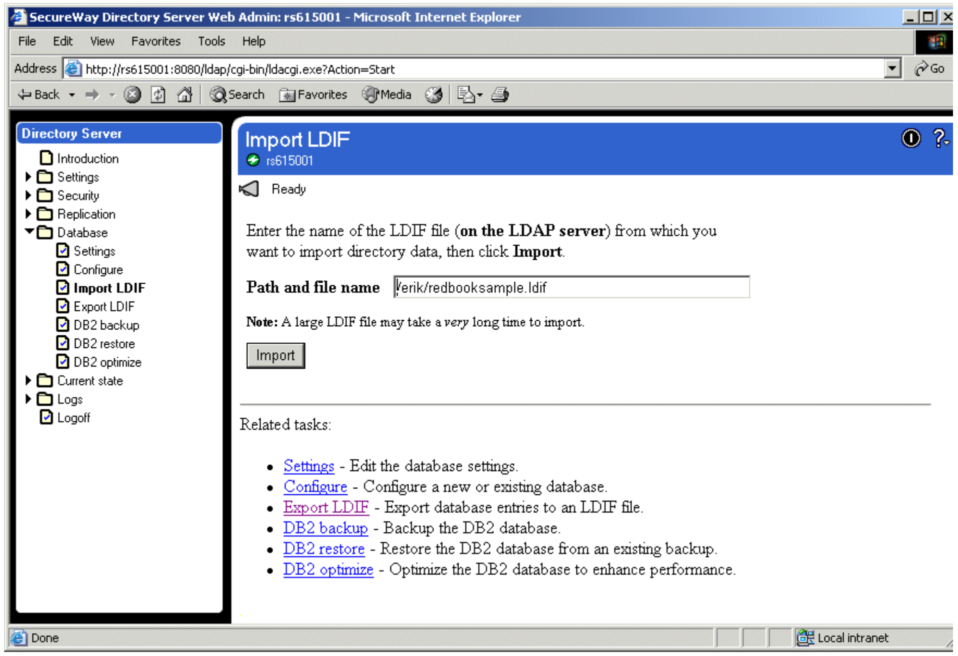
2. Ensure that the IBM HTTP Server is running on the server that hosts IBM SecureWay Directory (see Section 10.2.3, “Starting and stopping the WebSphere Everyplace Server components for INS” on page 374).

3. Start the following URL in a browser:

   `http://<hostname>/ldap`

   and log on using the administrator user ID and password for the directory. For the user ID, you must use the DN, for example cn=ldapadmin.

   Select **Database -> Import ldif** from the menu on the left, which will open the ldif import page.


SecureWay Directory Server Web Admin: rs615001 - Microsoft Internet Explorer

Address http://rs615001:8080/ldap/cgi-bin/ldacgi.exe?Action=Start

Import LDIF

rs615001

Ready

Enter the name of the LDIF file (**on the LDAP server**) from which you want to import directory data, then click **Import**.

Path and file name /erik/redbooksample.ldif

Note: A large LDIF file may take a *very* long time to import.

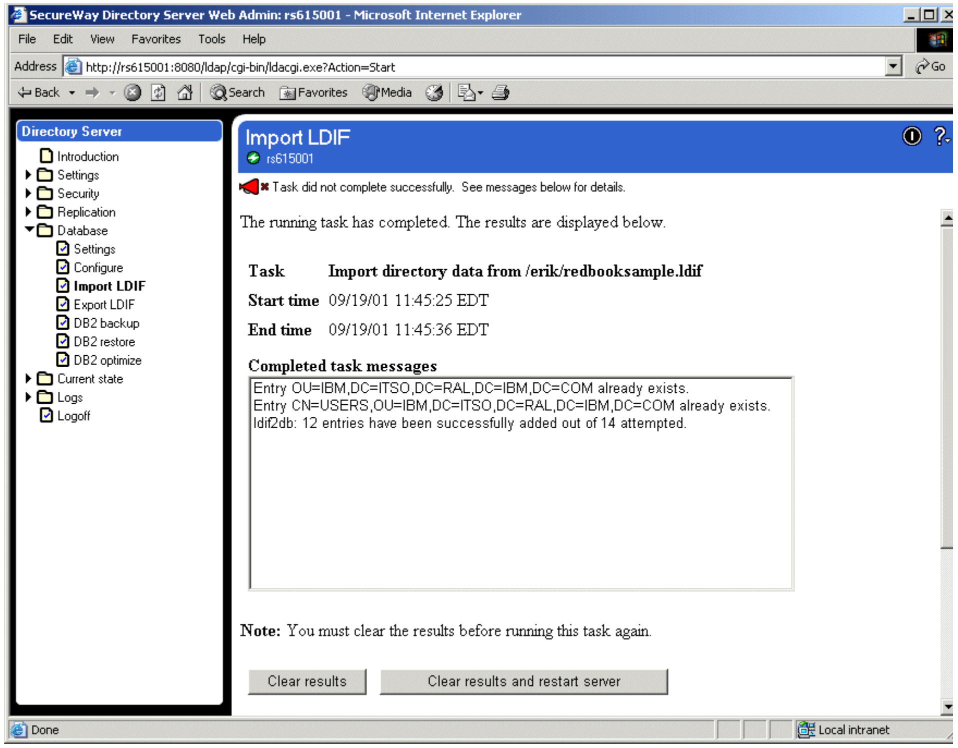
Related tasks:

- Settings - Edit the database settings.
- Configure - Configure a new or existing database.
- Export LDIF - Export database entries to an LDIF file.
- DB2 backup - Backup the DB2 database.
- DB2 restore - Restore the DB2 database from an existing backup.
- DB2 optimize - Optimize the DB2 database to enhance performance.

Figure 10-11 IBM SecureWay Director Import LDIF page

Note that in the above figure, the HTTP server is listening on port 8080. Usually, the port is 80 and does not have to be entered.

4. You must then wait until the file has been imported. The panel will indicate if importing was successful. Even if it indicates that the import was not successful, this is not necessarily a problem. When you import it again, for example when registering the next user, the problem might occur that a duplicate entry is found, as shown in Figure 10-12. This will be reported as an error, but this is no real problem.


SecureWay Directory Server Web Admin: rs615001 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://rs615001:8080/ldap/cgi-bin/ldacgi.exe?Action=Start
Directory Server
Introduction
Settings
Security
Replication
Database
Settings
Configure
Import LDIF
Export LDIF
DB2 backup
DB2 restore
DB2 optimize
Current state
Logs
Logoff
Import LDIF
rs615001
Clear results
Clear results and restart server
Done
Local intranet

Task did not complete successfully. See messages below for details.

The running task has completed. The results are displayed below.

Task	**Import directory data from /erik/redbooksample.ldif**
Start time	09/19/01 11:45:25 EDT
End time	09/19/01 11:45:36 EDT

Completed task messages

```
Entry OU=IBM,DC=ITSO,DC=RAL,DC=IBM,DC=COM already exists.
Entry CN=USERS,OU=IBM,DC=ITSO,DC=RAL,DC=IBM,DC=COM already exists.
ldif2db: 12 entries have been successfully added out of 14 attempted.
```

Note: You must clear the results before running this task again.

Figure 10-12 The result of importing an LDIF file

5. You must restart the LDAP server to activate the changes. Click **Clear results and restart server**. The panel will indicate whenthe task of restarting the server is completed.

You can inspect the new configuration using the Directory Management Tool of IBM Secureway Directory.

10.4 Simple Notification

This section discusses the Simple Notification facility of INS. With Simple Notification, users do not register triggers that are fired upon matching content, but notifications are sent directly to the users, using their preferences.

10.4.1 The Application Programming Interface (API)

To send messages directly through UND, the API is very simple for most applications. You will learn about the API by examining the source of SimpleUNDSend, an example that ships with INS. Simpleundsend is a basic example of a program that sends notifications to users directly through UND.

Example 10-2 SimpleUNDSend.java

```
/*****************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)              *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved          *
 *                                                                          *
 *  Licensed Materials - Property of IBM                                    *
 *                                                                          *
 *  US Government Users Restricted Rights - Use, duplication or disclosure  *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.           *
 *****************************************************************************
*/
// Intelligent Notification Service api
import com.ibm.pvc.we.ins.*;
import java.io.*;

/**
 * Basic sample demonstrating use of the WebSphere Everyplace Server
 * Intelligent Notification Service client API.
 */
public class SimpleUNDSend {

   public static void main(String[] args){
      // create a client service stub for the specified UND server and port
      NotificationService ns =
               new NotificationService("rs615002.itso.ral.ibm.com",55005);

      // specify delivery options
      DeliveryOptions opts = new DeliveryOptions();

      // select devices to which message is delivered
//    opts.devices = opts.IM;
      opts.devices = opts.EMAIL + opts.WAP;

      // select message priority
      opts.priority = opts.NORMAL;

      // opts.ANY indicates that the server should send
      // the message to any one device
      // specified in opts.devices.
      // The server will loop through the devices and try to
      // send the message, and will stop once the message
      // is successfully delivered.
      //
      // specify opts.ALL to send message to all specified devices.
//    opts.multiDevices = opts.ALL;
      opts.multiDevices = opts.ANY;

      // create message in NotificationML format
      String message = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>"
                  + "<message>"
                  + "<to>erongen@IBM</to>"
                  + "<from>erongen@IBM</from>"
                  + "<subject>Testing INS UND</subject>"
                  + "<text> This is a test message</text></message>";
      try {
         // send the message with the specified delivery options
         int i = ns.sendMessage(message,opts);
         System.out.println("sendMessage returned code: "+i);
      }
      // an IOException will be thrown if the NotificationService has
      // trouble communicating with the server
      catch (IOException e)
      {
         e.printStackTrace();
      }

   }


}
```

NotificationService

When the application creates a new instance of com.ibm.pbc.we.ins.NotificationService, it is ready to communicate with the UND settings.

```
NotificationService ns = new NotificationService(hostname,port)
```

Next, the delivery options are set. The DeliveryOptions class contains all possible settings for delivery of the notification.

DeliveryOptions.devices

The value for the devices field is constructed as shown below:

```
   opts.devices = DeliveryOptions.IM
```

or

```
   opts.devices = DeliveryOptions.EMAIL + DeliveryOptions.WAP
```

The devices field defines to which devices you want to send the notification. You can send the notification to multiple devices by adding them. Depending on the priority and multiDevices setting, the notification will be sent to no device, one device or several devices of the user.

DeliveryOptions.priority

This field has three possible values:

1. opts.priority = DeliveryOptions.NORMAL
2. opts.priority=DeliveryOptions.URGENT,
3. opts.priority = DeliveryOptions.FYI

   The priority field has one of the above values and this defines the priority of the notification. If the receiving user has not listed the device set under opt.devices for this priority in the group of which the sending user is a member, no notification is sent.

DeliveryOptions.multiDevices

This field has two possible values:

1. DeliveryOptions.ANY:The notification is sent to one of the devices of the user, depending on the opts.devices and opts.priority settings and the preferences in the directory of the receiving user.
2. DeliveryOptions.ALL: The notification is sent to all devices of the user, depending on the opts.devices and opts.priority settings and the preferences in the directory of the receiving user.

Construction of the message

The notifications are defined in the Notification Markup Language (Notification ML) Notification ML is the markup language used to format a notification message. Notification ML is a form of XML.

Example 10-3 Notification ML example

```
<message>
   <to>Joe_employee@realm1.company.com</to>
   <from>CompanyX_EINS@realm1.company.com</from>
   <subject>CompanyX Stock</subject>
   <text>CX = 110</text>
   <url>http://Svcs_persistant_storage/stock38521.jsp</url>

</message>
```

The corresponding DTD (Document Type Definition) is shown below:

Table 10-2 DTD for Notification ML

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT message (to, from, subject, text, url*)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

Table 10-3 Description of tags

Tag	Description
<message>...</message>	Notification message
<to>...</to>	User ID of message recipient
<from>...</from>	User ID of message sender
<subject>...</subject>	Subject of message
<text>...</text>	Text message body
<url>...</url>	Link to a stored message

Gateway adapters

Gateway adapters connect the Universal Notification Dispatcher to the Everyplace Wireless Gateway and any other gateway being used. Each recipient device type requires a gateway adapter. Usually, you do not have to worry about the gateway adapters. For the delivery mechanisms that are supported by INS (WAP Push, Sametime, e-mail and SMS), the gateway adapters are already defined and you do not have to customize them. You can find them in /usr/INS/IBMEPS/samples/adaptors

The next figure gives a schematic overview of the gateway adapters in INS.

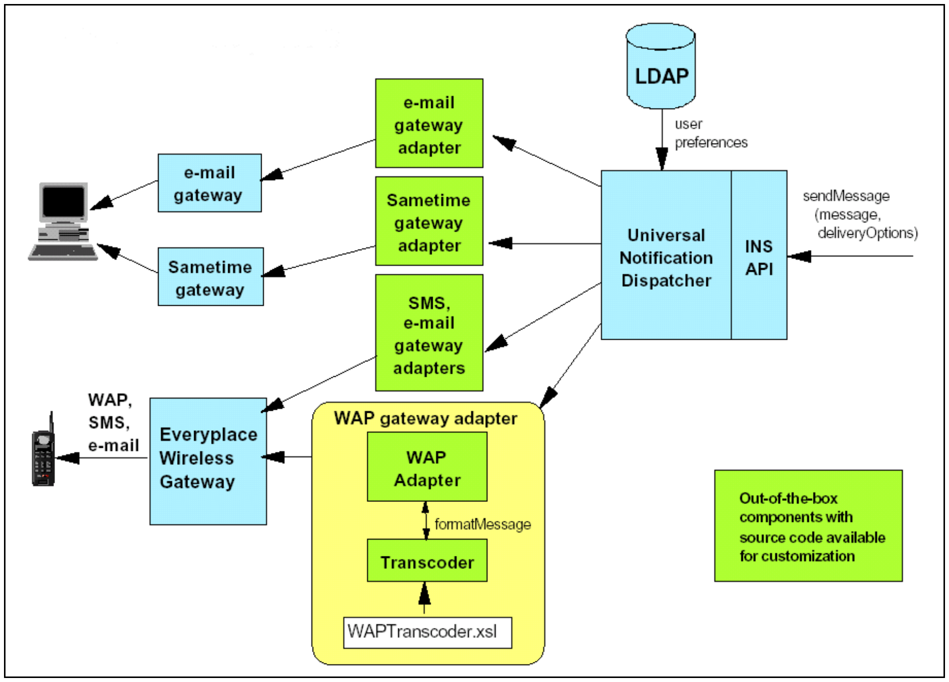

LDAP
user preferences
e-mail gateway adapter
Sametime gateway adapter
SMS, e-mail gateway adapters
e-mail gateway
Sametime gateway
Universal Notification Dispatcher
INS API
sendMessage (message, deliveryOptions)
WAP, SMS, e-mail
Everyplace Wireless Gateway
WAP gateway adapter
WAP Adapter
formatMessage
Transcoder
WAPTranscoder.xsl
Out-of-the-box components with source code available for customization

Figure 10-13 Gateway adapter s

As an example of how a gateway adapter works, the WAP Gateway Adapter will be discussed.

For details, see the Gateway Adapter source code below, and the sources of the class WAPTranscoder and the XSLT StyleSheet WAPTranscoder_SS.xsl.

Example 10-4 Gateway Adapter for WAP

```
//*****************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)               *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved           *
 *                                                                           *
 *  Licensed Materials - Property of IBM                                     *
 *                                                                           *
 *  US Government Users Restricted Rights - Use, duplication or disclosure   *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.            *
 *****************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors;

import java.io.*;
import java.util.*;

import com.ibm.pvc.we.ins.und.server.adaptors.*;
import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.*;

import com.ibm.pvc.we.ins.und.server.dispatcher.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.wireless.push.*;

public class WAP extends PAPUser implements Gateway,
com.ibm.logging.IRecordType
{
        private final static String
insIBMCopyright=GeneralConstants.insIBMCopyright;

        private Transcoder currenttc;
        private DeviceAddress deviceAddr;
        public static Properties gwconfig;
        private String msgbody;
        public static long ADAPTORS = 1 << 37;

        public WAP(){
                super();
                init();
        }

        public int sendMessage(DeviceAddress addr,Msg msg) {

                if (Logging._isTracing) {

Logging._tracer.text(ADAPTORS,this,"sendMessage","Entry");
                        }

                // initialize variable for this session
                deviceAddr = addr;
                currenttc = null;
                msgbody=null;

                formatMessage(addr,msg);

                if (msgbody != null )
                {
                int returnCode=send();
                return returnCode;
                }
                return SEND_ERROR;
        }

        /* * Insert the method's description here.
```

```
         * load transcoder if it has not been loaded yet
         * Creation date: (1/10/2001 3:20:21 PM)
         * @param notificationMsg com.ibm.pcds.ns.server.src.Msg
         */
        public void formatMessage(DeviceAddress deviceAddr, Msg
notificationMsg)
        {
                if (Logging._isTracing) {

Logging._tracer.text(ADAPTORS,this,"formatMessage","Entry");
        }

        // get proper transcoder from tc_collection
        currenttc=TcFactory.createTc(deviceAddr,"WAP_TC");

        if (currenttc != null)
            msgbody=currenttc.format(deviceAddr,notificationMsg);

        }

        /** Insert the method's description here.
         * This method should load gateway configuration file.
         * Creation date: (1/15/2001 3:51:37 PM)
         */
        public void init()
        {
        if ( gwconfig == null ) {
         gwconfig= NS_Configuration.getProperties();
        }

        }

        public int send() {
        if (Logging._isTracing) {
          Logging._tracer.text(ADAPTORS,this,"send","Entry");
        }
          int returnCode=SEND_ERROR;


        String message; // message as strung together from the NS Msg.
        PushMessage p_message;
        PushAddress p_address;
        String _phoneNumber = deviceAddr.getAddress();

          try {
          p_message=new PushMessage();
          p_message.setContent( msgbody, "text/vnd.wap.wml");
           p_message.setDeliveryMethod( DeliveryQoS.DELIVERY_UNCONFIRMED);
          p_address= new PushAddress();
          // "PLMN" replaced by IPv4 for testing: we have no WAP enabled
phones.
          // PLMN== public land mobile network
          p_address.setAddress( _phoneNumber, "IPv4", PAPUser.getHost());

          returnCode= sendMessage( p_message, p_address);
        } catch ( PushException pe ) {
          //pe.printStackTrace();
          // JLog
          if (Logging._isLogging) {
                      Logging._logger.exception(TYPE_ERROR,this,"send",pe);
          }
          return SEND_ERROR;
        }

        return returnCode;

        }
}
```

When the UND needs to send a message to a WAP device, the Gateway Adapter must do two things

1. Transform the NotificationML into WML.
2. Send the result to the wireless gateway with the right address, etc.

Therefore, a WAP Gateway object is instantiated. This will instantiate a WAPTranscoder for transcoding the message. This will be discussed in more detail.

The Constructor: WAP()

First of all, in the init() method that is called from the constructor, the properties of the Adapter are read.

sendMessage()

When the UND has a message to dispatch, the method SendMessage() is called. It will format the message by calling formatMessage(), and after that it will send the message to the server by calling sendMessage().

formatMessage()

formatMessage() will create a transcoder that uses XSL Transformation to convert the notification ML into the appropriate markup for the delivery protocol:

currenttc=TcFactory.createTc(deviceAddr,"WAP_TC");

It creates the transcoder and calls its format() method, which will try to load the stylesheet. If the stylesheet is found, the method formatMessage() of the superclass is called, and does an XSL transformation based on the loaded stylesheet. If the stylesheet is not found, the method formatUseSS(msg) is called, which is implemented in the transcoder itself. After transcoding the message, some extra formatting is performed, namely the setting of the WAP version.

The transcoder and the stylesheets are located in the directory /usr/IBMEPS/INS/samples/adaptors/

Transcoder: WAPTranscoder.java

Stylesheet: WAPTranscoder_SS.xsl

They are both printed below.

send()

Finally, the message is sent to the device by calling the send() message.

Example 10-5 Transcoder for WAP

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors.transcoder;

/**
 * Insert the type's description here.
 * Transcoder will format message specific to communication protocol and
adapter implementation
 * of gateway interface.
 */

import java.util.Properties;

import com.ibm.pvc.we.ins.und.server.adaptors.*;
import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.*;

import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import java.io.*;
import com.ibm.pvc.we.ins.und.server.dispatcher.*;
import com.ibm.pvc.we.ins.GeneralConstants;

public class WAPTranscoder extends GenericTranscoder implements Transcoder
{
        private final static String
insIBMCopyright=GeneralConstants.insIBMCopyright;

        String wapversion;

        /**
         * WAPTranscoder constructor comment.
         */
        public WAPTranscoder()
        {
                super();

        }

        /**
         * Insert the method's description here.
         * Set WAP version
         */
        public String addVersion(String message,String ver)
        {
           // if no version is provided or not a valid version, WAP1.1 is
assumed

           if ( ver == null )
            ver = "1V1";
        else if ( !ver.equals("1V2"))
            ver = "1V1";


            // insert version
        StringBuffer tempmsg= new StringBuffer(message);
        int offset = message.indexOf("<wml>");
        if (ver.equals("1V1"))
        {
        tempmsg.insert(offset,"<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML
1.1//EN\"\n\"http://www.wapforum.org/DTD/wml_1.1.xml\">\n");
        }
        else if (ver.equals("1V2"))
        {
        tempmsg.insert(offset,"<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML
1.2//EN\"\n\"http://www.wapforum.org/DTD/wml12.dtd\">\n");
        }
        // Debug purpose
        return tempmsg.toString();


           }

        /**
         * Insert the method's description here.
         * Format message according to gateway interface
         * For now it just support text format, later rich format support may
be added
         * @return java.lang.String
         */
        public String format(DeviceAddress addr,Msg msg) {
        String message;

         if (Logging._isTracing) {
                     Logging._tracer.text(TRANSCODERS,this,"format","Entry");
                    }
        // check if user provides a style sheet
        setStylesheet();
        if (stylesheet.equals("NULL"))
            message=formatMessage(msg);
        else
            message=formatUseSS(msg);

        message = addVersion(message,addr.getVersion());
         if (Logging._isTracing) {

Logging._tracer.text(TRANSCODERS,this,"format","Transformed Message is \n {0}
",message);
                    }
        return message;
        }

        /**
         * Insert the method's description here.
         * @return java.lang.String
         * @param msg com.ibm.pcds.ns.Msg
         */
        public String formatMessage(Msg msg) {

         if (Logging._isTracing) {

Logging._tracer.text(TRANSCODERS,this,"formatMessage","Entry");
                    }
        String notificationML;
```

```
         notificationML=msg.extractML();


        // WML deck has been tested with Nokia toolkit 2.0
        StringBuffer b=new StringBuffer();
        b.append("<?xml version=\"1.0\"?>\n");

        b.append("<wml>\n<card id=\"Notification\" title=\"Message\">\n");
        b.append("<p>\n");


        b.append("From: ").append(msg.fromUserid).append("<br/>\n");
        b.append("Subject: ").append(msg.subject).append("<br/>\n");

        String
msgbody=notificationML.substring(notificationML.indexOf("<text>")+6,notificatio
nML.indexOf("</text>"));
        b.append("Msg: ").append(msgbody).append("<br/>\n");

        // allow multiple urls

        int index=notificationML.indexOf("<url>");
        while ( index != -1 )
        {
        String
url=notificationML.substring(index+5,notificationML.indexOf("</url>",index));
        b.append("\t").append(url).append("<br/>\n");
        index=notificationML.indexOf("<url>",index+1);
        }

        b.append("</p>\n").append("</card>\n</wml>\n");
          return b.toString();
        }
}
```

Example 10-6 StyeSheet for the WAP Transcoder

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:output encoding="utf-8" method="xml" indent="yes" />
<xsl:template match="message">
<wml>
   <card id="Notification" title="Message">
     <p>
     From:<xsl:value-of select="from"/>
     <br/>
     Sub: <xsl:value-of select="subject"/>
     <br/>
     Msg: <xsl:value-of select="text"/>
     <br/>
     <xsl:for-each select="url">
     <xsl:text>
     </xsl:text><xsl:value-of select="."/>
        <br/>
     </xsl:for-each>
     </p>
   </card>
</wml>
</xsl:template>
```

The following gateway adapters are shipped with INS:

- PAPUser - superclass of sample gateway adapters
- WAP Gateway Adapter
- Sametime/IM Gateway Adapter
- Short Messaging Service (SMS) Gateway Adapter
- SMTP E-mail Gateway Adapter
- E-mail Push Gateway Adapter

Customizing gateway adapters

1. Modify or extend the class in the source code listed above or one of the others to work with the alternate gateway. You may also create new gateway adapter classes by implementing the Gateway interface.
2. Create a new .jar file with the new or modified classes, for example, adapter.jar.
3. Place the new adapter.jar file in the lib directory where the ins.jar file is located. For example:

   ```
   /usr/IBMEPS/INS/lib/adapter.jar on AIX
   /opt/IBMEPS/INS/lib/adapter.jar on Solaris
   ```

4. Modify the startup script for Universal Notification Dispatcher to include the adapter.jar file in the CLASSPATH. For example:

   ```
   /usr/IBMEPS/INS/bin/startUND on AIX
   /opt/IBMEPS/INS/bin/startUND on Solaris
   ```

Note: Ensure that adapter.jar is before ins.jar in the CLASSPATH in order for the classes in adapter.jar to override the corresponding original classes in ins.jar. For example, set the CLASSPATH to:

```
CLASSPATH=/usr/IBMEPS/INS:/usr/IBMEPS/INS/bin:
/usr/IBMEPS/INS/lib/adapter.jar:/usr/IBMEPS/INS/lib/ins.jar
```

5. Modify the package name settings in SecureWay Directory to reflect the new class or package names. These settings will tell the Universal Notification Dispatcher (UND) to instantiate adapters using the new, customized classes. Refer to the section on gateway adapter and transcoder configuration in SecureWay Directory for more details on how to modify these settings in SecureWay Directory (see “Gateway adapters” on page 391).
6. If you altered or created new transcoders or stylesheets, put them in the appropriate place on the file system.
7. Restart the Universal Notification Dispatcher with the modified startup script.

You can find more information on the Gateway Adapters in the InfoCenter.

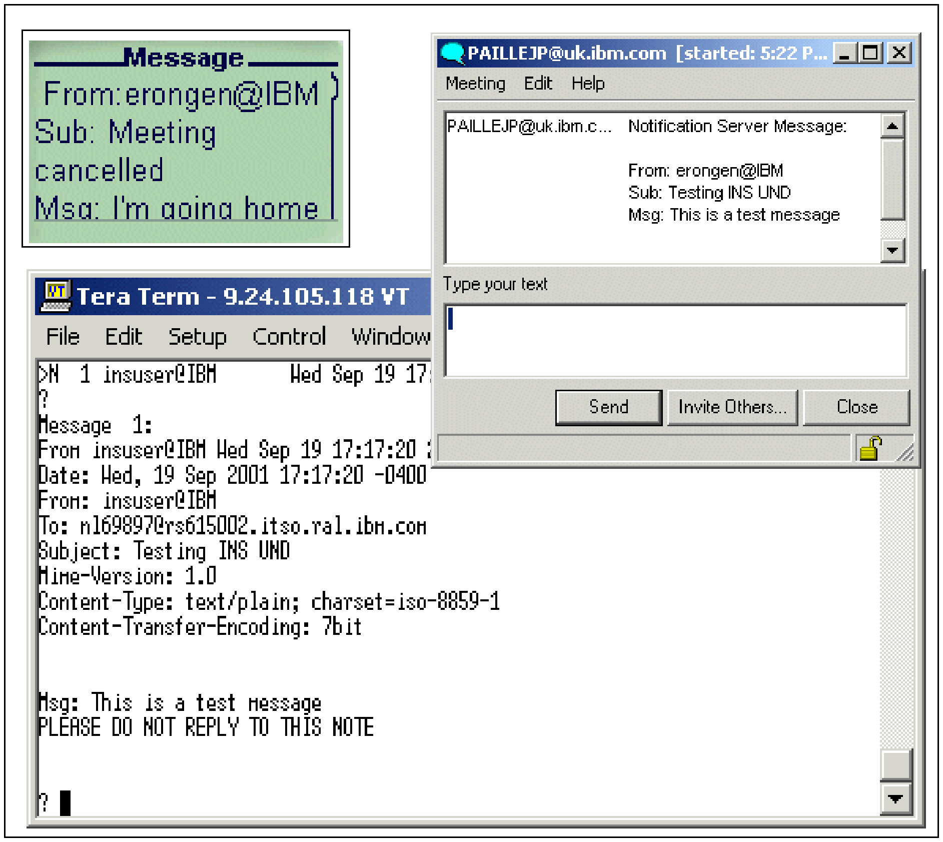
10.4.2 Running the example

SimpleUNDSend

To run the SimpleUNDSend to check your system, it is most convenient to have logging and tracing to the console enabled (see “Configuring logging and tracing” on page 373) and start the UND service in the foreground (see “Starting the INS Servers in the foreground” on page 377), especially if you need to change the UND settings, for which restarting of the service is required. Alternatively, you can view the logs and trace real time while the service is running in the background by entering the following command on in a terminal:

```
tail -f <filename>
```

Where **`<filename>`** is the path and name of the log file. You can use the settings of the user and the DeliveryOptions to determine to what device the notification is to be sent (see 10.4.1, “The Application Programming Interface (API)” on page 388). The results are shown in Figure 10-14.


PAILLEJP@uk.ibm.com [started: 5:22 P...
Meeting Edit Help
Type your text
Send
Invite Others...
Close
Tera Term - 9.24.105.118 VT
File Edit Setup Control Window

Message
From:erongen@IBM
Sub: Meeting
cancelled
Msg: I'm going home

PAILLEJP@uk.ibm.c... Notification Server Message:

From: erongen@IBM
Sub: Testing INS UND
Msg: This is a test message

```
>N  1 insuser@IBM       Wed Sep 19 17
?
Message  1:
From insuser@IBM Wed Sep 19 17:17:20 2
Date: Wed, 19 Sep 2001 17:17:20 -0400
From: insuser@IBM
To: n169897@rs615002.itso.ral.ibm.com
Subject: Testing INS UND
Mime-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 7bit


Msg: This is a test message
PLEASE DO NOT REPLY TO THIS NOTE


?
```

Figure 10-14 Notifications sent to WAP, Sametime and e-mail.

Example 10-7 UND Server trace of a notification being dispatched

```
01
01  2001.09.13 16:16:43.054
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController send
03    client requests send over multiple devices: false
04
05  2001.09.13 16:16:43.055
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController send
06    devices client requests to send over:  im
07
08  2001.09.13 16:16:43.055
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController
getUserCalendarContext
09    Searching for calendar context of erongen@IBM
10
```

```
11  2001.09.13 16:16:43.070 NotificationController getUserCalendarContext IBM
Intelligent Notification Service UND
12    Could not establish link to SCS
13
14  2001.09.13 16:16:43.071
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController getRecipDevices
15    calling preference engine with priority normal context None available
16
17  2001.09.13 16:16:43.871
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController getRecipDevices
18    otained 1 devices to send message to
19
20  2001.09.13 16:16:43.872
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController getAdaptors
21    Obtained device URL header, containing device information
22
23  2001.09.13 16:16:43.872
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationCon
24  troller getAdaptors
25    Device optained
26  owner = erongen@IBM
27  deviceid = sametime1
28  ibm-appprotocolversion = V15
29  ibm-isdeviceenabled = true
30  ibm-appdeviceaddress = e_rongen@nl.ibm.com
31  ibm-appprotocol = sametime
32  dn =
cn=sametime1/und,deviceID=sametime1,uid=erongen,cn=users,ou=IBM,DC=ITSO,DC=
33  RAL,DC=IBM,DC=COM
34  uid = 0
35  objectclass = ibm-undDevice
36  cn = sametime1/und
37  ibm-deviceidtype = 1
38  description = 0
39  ipserviceport = 1533
40  ibm-appprotocoltype = im
41  host = messaging.ibm.com
42
43  2001.09.13 16:16:43.873
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController getAdaptors
44    parsing device URL header im#sametime#V15
45
46  2001.09.13 16:16:43.874
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController send
47    sending over messaging adaptors
48
49  2001.09.13 16:16:43.874 com.ibm.pvc.we.ins.und.server.adaptors.SametimeIMGW
send
50  Message
```

```
  entered method sendMessage

2001.09.13 16:16:43.874 com.ibm.pvc.we.ins.und.server.adaptors.SametimeIMGW
form
atMessage
  Entry

2001.09.13 16:16:43.875 TcFactory createTc(DeviceAddress,Key)
  Entry

2001.09.13 16:16:43.875 TcFactory loadTc
  Entry

2001.09.13 16:16:43.875 TcFactory loadTc
  Entry

2001.09.13 16:16:43.875 TcFactory loadTc
  Entry

2001.09.13 16:16:43.876 TcFactory loadTc
  Entry

2001.09.13 16:16:43.876 TcFactory createTc(DeviceAddress,Key)
  Default transcoder for this device type will be returned.
 Key for default TC is : IMSAMETIME1V5_TC

2001.09.13 16:16:43.876
com.ibm.pvc.we.ins.und.server.adaptors.transcoder.IMSTTranscoder format
  Entry

2001.09.13 16:16:43.954
com.ibm.pvc.we.ins.und.server.adaptors.transcoder.IMSTTranscoder formatUseSS
  Entry

2001.09.13 16:16:43.967
com.ibm.pvc.we.ins.und.server.adaptors.transcoder.IMSTTranscoder format
  Transformed Message is

        From: erongen@IBM
        Sub: Testing INS UND
        Msg: This is a test message



2001.09.13 16:16:43.967 com.ibm.pvc.we.ins.und.server.adaptors.
SametimeIMGW send
  Entry

```

```
95  2001.09.13 16:16:45.036
com.ibm.pvc.we.ins.und.server.adaptors.SametimeIMGW$CommSrvListenernamesResolve
d
96    successfully resolved names
97
98  2001.09.13 16:16:45.039
com.ibm.pvc.we.ins.und.server.adaptors.SametimeIMGW$2 execute
99  creating message channel to Sametime user
100
101 2001.09.13 16:16:46.037
com.ibm.pvc.we.ins.und.server.adaptors.SametimeIMGW$IMSrvListener
messageCreated
103 successfully created message channel to Sametime user
104
105 2001.09.13 16:16:46.042
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController send
106   successful send using adaptor im : 0
107
108 2001.09.13 16:16:46.043
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController send
109   finished sending message over adaptors
110
111 2001.09.13 16:16:46.043
com.ibm.pvc.we.ins.und.server.dispatcher.NotificationController send IBM
Intelligent Notification Service UND
112   1000412206043,erongen@IBMerongen@IBM,succ
113
114 2001.09.13 16:16:46.044 com.ibm.pvc.we.ins.und.server.dispatcher.Connection
run
115
116   finished dispatching request
117
118 2001.09.13 16:16:46.046
com.ibm.pvc.we.ins.und.server.adaptors.SametimeIMGW$IMSrvListener
messageDestroyed
```

- Lines 3 and 6 show the DeliveryOptions that were selected for the notification
- Lines 9-12 show an apparent problem related to the calendar context `could not establish link to CSS`. This problem can be ignored. It relates to future functionality of INS.
- Lines 15, 18 and 21-41 show how INS reads the LDAP for the user and device information.
- Line 47 shows that the gateway adapter is called. Lines 49 and 76 show which gateway adapter and transcoder are loaded. Line 79 shows that the

stylesheet for transcoding was found (see “Gateway adapters” on page 391 for details).

- Line 106 shows the result of the communication with the remote server, the Sametime server in this example.
- Line 112 shows the overall status of the notification.
- Line 116 does not neccessarily mean that the message has been successfully delivered.

10.5 Subscriptions

In addition to sending notifications to users directly using UND, INS provides the possibility of creating applications that allow users to subscribe and receive notifications when the content meet criteria which the user can define. In one of the examples of INS, the user can be notified when a stock of their choice reaches, is above or is below a certain value. The flow and the API elements involved will be discussed in the next section

10.5.1 Flow

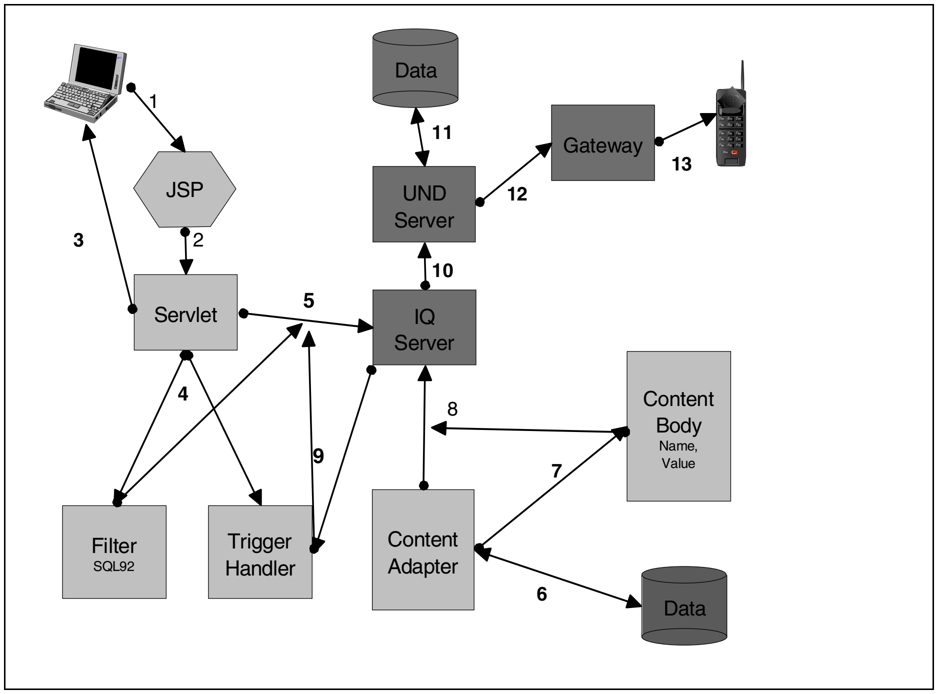


Figure 10-15 Interactions between the individual components of the INS system for complex notification

1. The user opens the page for subscription.
2. The JSP and the servlet gather information for the trigger.
3. In the sample, the user first adds a trigger to a list of triggers, and then submits the list.
4. A TriggerHandler and a Filter are created with the trigger and filtering information.
5. The TriggerHandler and the filter are submitted to the IQ Server.
6. A Content Adapter reads data from a content source.
7. From the raw data, a ContentBocy is constructed, which is a name-value pair representation of the content.
8. The contentBody is submitted to IQ.

9. All triggers for users who subscribed to this source are checked, and the matching triggers are fired.
10. IQ sends notifications to UND (see Appendix 10.7.2, “Simple notification” on page 423 details on how to send simple notifications).

Finally (11,12 and 13), notifications are delivered, based on the user’s preferences.

More details will be given in the next section.

10.5.2 The Application Programming Interface

Being a set of services written in Java, INS provides a Java API for creating customized applications. The classes and methods that are used by an application using INS subscription facilities are discussed in more detail in this section.

com.ibm.pvc.we.ins.util.SubscriptionUtility

This class facilitates application development by providing functions used by the developer

- *getUser(request)*

  This method is used to get the user ID from the request. If INS is run together with WebSEAL-Lite, the user ID is found in the HTTP header, otherwise the user ID is obtained from the user ID attribute in the queryString part of the URL:

  ```
  (http://<url>?userid=insuserid)
  ```

  In this example `<url>` is the URL needed to get to the JSP or servlet, and `insuserid` is the value of the user ID to be used by the user.

  Here is an example ofusing getUser in a Java Server Page:

  ```
  <%
        String Id;
        Id = SubscriptionUtility.getUser(request);
      %>
  ```

- *isSecure()*

  This method is used to find out if the request is operated in a secure environment. In the example Java Server Pages, this is used to add the user ID as a hidden field in the form if security is off.

  ```
  <% if (!SubscriptionUtility.isSecure())
         { %>
           <INPUT name=userid type="hidden" value="<%= Id %>">
     <% } %>
  ```

com.ibm.pvc.we.ins.TriggerManager and com.ibm.pvc.we.ins.SocketClientStub

TriggerManager is an interface which oversees the creation and removal of triggers, the firing of triggers, and the fetching of content persistently stored by the firing of a trigger that returns a retention-specification (see below for details).

SocketClientStub is an implementation of the TriggerManager. It provides the methods through which servlets install and remove triggers. It implements each method of the TriggerManager interface by sending the invocation to com.ibm.iqueue.server.SimpleSocketTriggerManagerImpl over a socket, receiving the outcome of the invocation over a socket, and returning or throwing an exception in accordance with that outcome. The protocol used is documented in the interface com.ibm.iqueue.server.SimpleSocketTriggerManagerConstants.

Here is an example:

```
TriggerManager manager = new SocketClientStub(IQueuehost,IQueueport);
```

And, when all necessary classes have been created (see below for details):

```
TriggerID tid = manager.addTrigger(userid, sourceName, filter, triggerHandler);
```

Methods

- *addTrigger()*, *removeTrigger*() and *RemoveAllTriggers()* are used to maintain the triggers for a user.
- *FetchPersistentContent()* is used to retrieve the content for a user when the notification was saved, and not sent to the user.

com.ibm.pvc.we.ins.TriggerHandler

The TriggerHandler is created when the user subscribes, but is again used by the IQServer when the trigger is fired, in which case *TriggerManager.handleMatch()* is called.

An example of the handleMatch() method (for stock sample) is discussed below.

Example 10-8 TriggerHandler for Stock Example handleMatch() method

```
public TriggerResult handleMatch(ContentBody cb)
  {
     String contentString;
     String notificationString;
     String secondNotificationString;
     String value,change,percentage;
     StringBuffer message = new StringBuffer();
     StringBuffer message2;
     int tResult;

```

```
    //Set the Trigger Option based on user input
    if (triggerOption.equals("once"))
     tResult = TriggerResult.STOP;
    else
  if (triggerOption.equals("always"))
      tResult = TriggerResult.CONTINUE;
     else
      tResult = TriggerResult.SAVE_AND_CONTINUE;

    if (contentOption.equals("save"))
     {
      StringBuffer content = new StringBuffer();
      try
      {
        value = String.valueOf(cb.getFloat(stocktype));
        change = String.valueOf(cb.getFloat("change"));
        percentage = String.valueOf(cb.getFloat("percentage_change"));
        content.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
        content.append("<quotes>\n");
        content.append("<stockquote>\n");
        content.append("<companyName>" + cb.getString("symbol") + "</companyName>\n");
        content.append("<symbol>" + cb.getString("symbol") + "</symbol>\n");
        content.append("<curValueUSD>" + value + "</curValueUSD>\n");
        content.append("<curChangeUSD>" + change + "</curChangeUSD>\n");
        content.append("<curChangePct>" + percentage + "</curChangePct>\n");
        content.append("<timestamp>" + cb.getString("timestamp") + "</timestamp>\n");
        content.append("</stockquote>\n");
        content.append("</quotes>");
        contentString = content.toString();
      }
      catch (IQueueException iqe)
      {
        iqe.printStackTrace(System.out);
        contentString = iqe.toString();
      }
      message2 = new StringBuffer();
      message.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
      message.append("<message>\n");
      message.append("<to>" + toUserid + "</to>\n");
      message.append("<from>" + toUserid + "</from>\n");
      message.append("<subject>STOCK</subject>\n");
      message.append("<text> Follow this URL " + notificationURL + "ContTransServlet?pageid=");
      if (SubscriptionUtility.isSecure())//Are authorization services enabled?
       message2.append(" for details </text>\n");//Yes, do not need to pass userid on request
      else
       message2.append("&amp;userid=" + toUserid + " for details </text>\n"); //No, include userid on request
```

```
57          message2.append("</message>");
58          notificationString = message.toString();
59          secondNotificationString = message2.toString();
60
61          return TriggerResult.newRetentionSpecification(contentString, notificationString,
62                                                              secondNotificationString,
"NOT_SAVED", 15, tResult);
63       }
64       else                // Option was chosen to not save the message
65       {
66          try
67          {
68            value = String.valueOf(cb.getFloat(stocktype));
69            message.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
70            message.append("<message>\n");
71            message.append("<to>" + toUserid + "</to>\n");
72            message.append("<from>" + toUserid + "</from>\n");
73            message.append("<subject>STOCK</subject>\n");
74            message.append("<text>" + cb.getString("symbol") + " = " + value + "</text>\n");
75            message.append("</message>");
76            notificationString = message.toString();
77          }
78          catch (IQueueException iqe)
79          {
80            iqe.printStackTrace(System.out);
81            notificationString = iqe.toString();
82          }
83          return TriggerResult.newNotificationSpecification(notificationString,tResult);
84       }
85   }
```

Lines 12 and 15 show how the trigger options are handled by setting the appropriate triggerResult options (see below).

Line 21 shows how the save options are handled.

If the message should be saved, the messages comprising the notification to the user and containing the URL are created, and a trigger result of the Retention Specifiation is created also.

Note: You could have other options with different trigger behavior from the ones presented in the examples (trigger options and save options). This should then be programmed in the handleMatch method; an example would be a trigger with a defined lifetime.

handleMatch() returns a TriggerResult object. This specifies the actions that IQueue should perform on behalf of the TriggerHandler, and what the disposition of the TriggerHandler should be at the conclusion of the call on handleMatch.

The actions that IQ should perform depend on the specification of the returned TriggerResult:

- TriggerResult.newRetentionSpecification() (see line 61)

  This means that the content of the notification must be saved by IQ, for later retrieval by the user.

- *TriggerResult.newNotificationSpecification(notificationString,tResult)* (see line 83)

  This means the content must not be saved by IQ.

The possible dispositions of the TriggerHandler, when finished, are:

- STOP

  IQueue should remove the TriggerHandler whose handleMatch method is returning this TriggerResult.

- CONTINUE

  The TriggerHandler whose handleMatch method is returning this TriggerResult should remain active; the state of the TriggerHandler upon return from handleMatch should not be saved persistently.

- SAVE_AND_CONTINUE

  The TriggerHandler whose handleMatch method is returning this TriggerResult should remain active; the state of the TriggerHandler upon return from handleMatch should be saved persistently.

Important: The class of the Triggerhandler must be in the classpath of both the application or servlet that creates it and the IQ server. When you change the class on the server, the IQ server must be restarted.

com.ibm.pvc.we.ins.ContentSource and com.ibm.pvc.we.ins.SimpleContentSource

The ContentSource is unique for every content that is fed into the system. This is used to define to which feed a certain trigger applies. Only matching triggers for the same contentType as that of the content submitted by the ContentAdapter will be compared.

com.ibm.pvc.we.ins.ContentFilter and com.ibm.pvc.we.ins.SQLSelector

The SQLSelector implements the ContentFilter interface. The SQLSelector contains the filter that determines when the trigger is fired. The content is sent to IQ as name-value pairs, which can be considered as columns in an SQL database, where the name is the name of the column, and the value is, of course, the value. The filter then describes the *where clause* of the SQL query as it is executed against the ContentBody. The use of SQL92 is defined by the underlying JMS (Java Messaging Services) framework, and as such, you must refer to the JMS Specifications for details on how to set up the query in the SQL Selector:

```
http://java.sun.com/products/jms/docs.html
```

com.ibm.pvc.we.ins.TriggerID

When a trigger is added, a triggerID is created, that can be stored and used later to remove the trigger again.

com.ibm.pvc.we.ins.ContentBody

ContentBody is the class that holds a string representation of XML-formatted content published to the iQueue Server by a ContentAdapter. A ContentBody is a set of name-value pairs. Each name is a string. Each value is a Boolean, byte, short, char, int, float, double, String, or byte[] value. ContentBody provides distinct *get...* and *set...* methods for each type of value. A value with a given name can only be fetched as a value of the type as which it was written, but a value of any type may be fetched as a String value.

Restriction: The size of the ContentBody cannot exceed 65535 bytes. To publish content greater then 64 KB in length requires breaking the content into 64 KB segments.

For more details, please refer to the INS API Javadoc documentation in the WebSphere Everyplace Server InfoCenter.

The Content Adapter

Unlike the other classes, the Content Adapter does not extend a class in the INS API. A Content Adapter is a stand-alone Java application that retrieves feed from a data source, creates the appropriate ContentBody and feeds it into the IQ Server. The XMLContentAdapter class that is shipped with the INS Samples transforms an XML file from the file system into a ContentBody object, and then submits it to the IQ Server.

The XML ContentAdapter takes two input parameters.

- The first is the name of the source. This is used to create a corresponding ContentSource object. This string must match that of the ContentSource object used to create the trigger.
- The next parameter is a directory name. the XMLContentAdapter will read all XML files from this directory and process them.

The basic work is done by the inner class XMLContentHandler, a content handler in the SAX Parsing framework. For details, see the Xerces project of Apache at:

```
http://xml.apache.org
```

In SAX Parsing, events are triggered when a branch or a leaf of the tree structure of the XML document is entered or exited. Upon occurrence of such an event, the corresponding method of XMLContentHandler is called, which will then update the contentBody class (cb in the example below).

Example 10-9 startElement()

```
public void startElement(String namespaceURI, String localName, String rawName,
Attributes atts) throws SAXException
   {
      String attribName;//elementName + local attribute name
      System.out.println("startElement: " + localName);
      elementName = localName;//Store name of element for possible use by
characters method
      elementData = "";//Clear out all data for previous element
      if (!namespaceURI.equals(""))
      {
        System.out.println(" in namespace " + namespaceURI + "(" + rawName +
")");
```

```
      }
      for (int i=0; i<atts.getLength(); i++)
      {
        System.out.println("Attribute: " + atts.getLocalName(i) + "=" +
atts.getValue(i));
        attribName = elementName + atts.getLocalName(i);
        cb.setString(attribName.trim(),atts.getValue(i).trim());
      }
   }
```

The method startElement() is called when a new XML element is encountered by the SAX parser. At that point, a new name-value pair is added to the ContentBody (cb) for each of the attributes of the element.

Example 10-10 endElement()

```
public void endElement(String namespaceURI, String localName, String rawName)
throws SAXException
   {
      System.out.println("endElement: " + localName + "\n");
      if (!(elementData.equals("")))
      {
         try
         {
            float fvalue = (Float.valueOf(elementData)).floatValue();
            cb.setFloat(elementName,fvalue);
         }
         catch(NumberFormatException nfe)
         {
            cb.setString(elementName,elementData.trim());
         }
      }

   }
```

The method endElement() is called when a new XML element is encountered by the SAX parser. At that point, a new name-value pair is added to the ContentBody (cb) for the elementData of the element.

Example 10-11 enddocument()

```
public void endDocument() throws SAXException
   {
      System.out.println("Parsing ends...");
      try
      {
        manager.fireMatchingTriggers(source,cb);//Fire matching triggers
      }
      catch (IQueueException iqe)
      {
         System.out.println(iqe.getMessage());
      }
      cb = null;   //Clean up content body
   }
```

When the document is finished, fireMatchingTriggers() is called for the TriggerManager (in this case the SocketClientStub, as discussed above).

10.5.3 Subscription examples

There are three examples for notifications shipped with INS: Stocks, Weather and News. These examples have basically the same structure, and therefore we will discuss only the Stock example.

Before you start working with the example, you must be sure that the three servers (UND, GB and IQ) are running (see 10.2.4, “Starting the INS servers” on page 376 for details).

In the stock subscription example, a user can subscribe to be notified for changes in stock values. For each trigger, the name of the stock (for example IBM), the operator and the amount can be specified. Also, the trigger options and save options for the trigger are set. Multiple triggers can be defined this way, and when the user has finished, the triggers can be submitted to the IQ server. The User Interface is shown below in Figure 10-16.

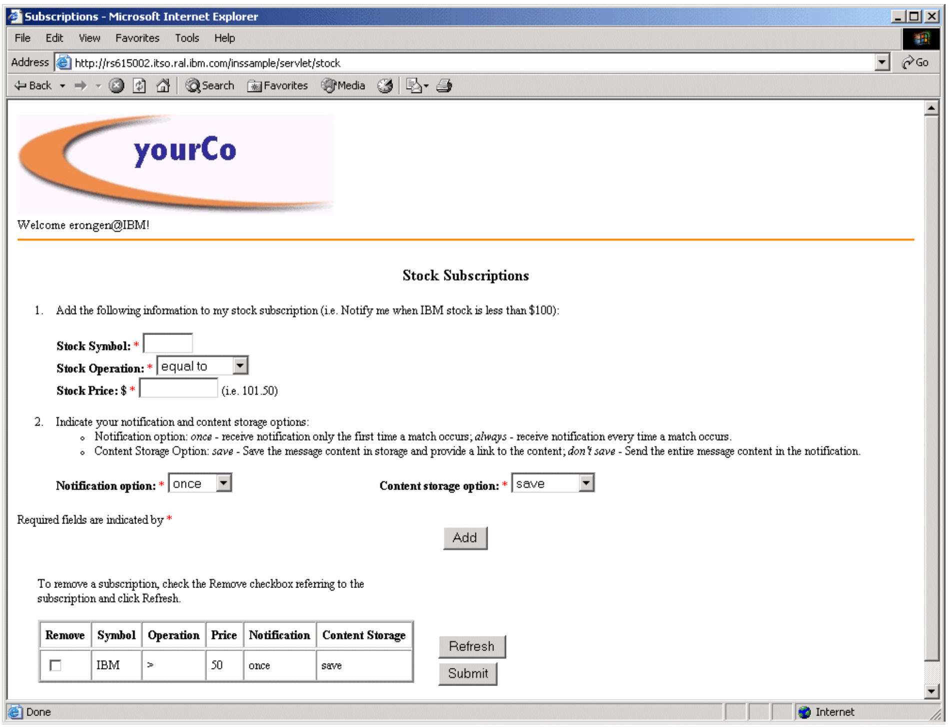
To access this panel, start a browser, and enter the following URL using security:

```
<hostname>:<port#>/inssample/StockSubscriptionForm.jsp
```

When not using security, enter:

```
<hostname>:<portnumber>/inssample/NewsSubscriptionForm.jsp?userid=<youruser
id@host.domain.com>
```

Where `<hostname>` and `<portnumber>` are the host name and port number of the server running the WebSphere Application Server that contains the samples. <youruserid@host.domain.com> is a user ID that is registered with INS


Subscriptions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
yourCo
Done
Internet

Address http://rs615002.itso.ral.ibm.com/inssample/servlet/stock

Welcome erongen@IBM!

Stock Subscriptions

1. Add the following information to my stock subscription (i.e. Notify me when IBM stock is less than $100):

   Stock Symbol: *
   Stock Operation: * equal to
   Stock Price: $ * (i.e. 101.50)

2. Indicate your notification and content storage options:
   - Notification option: *once* - receive notification only the first time a match occurs; *always* - receive notification every time a match occurs.
   - Content Storage Option: *save* - Save the message content in storage and provide a link to the content; *don't save* - Send the entire message content in the notification.

   Notification option: * once **Content storage option:** * save

Required fields are indicated by *

Add

To remove a subscription, check the Remove checkbox referring to the subscription and click Refresh.

Remove	Symbol	Operation	Price	Notification	Content Storage
☐	IBM	>	50	once	save

Refresh
Submit

Figure 10-16 The subscription user Interface of the stock example

The source code for this example has been discussed above in section 10.5.2, “The Application Programming Interface” on page 408, and can be found in the WebSphere Everyplace Server InfoCenter.

Restriction: The examples that ship with INS store user information in the user ID field of the servlet class.(for example in StockSubscriptionServlet.java). This means that this servlet will only work for one user at a time. However, this can easily be changed by adding this information to the session of the servlet.

When the triggers are registered in IQ, the content can be fed to the IQ server:

- Open a session on the AIX system that hosts INS, and change to the directory containing the examples:

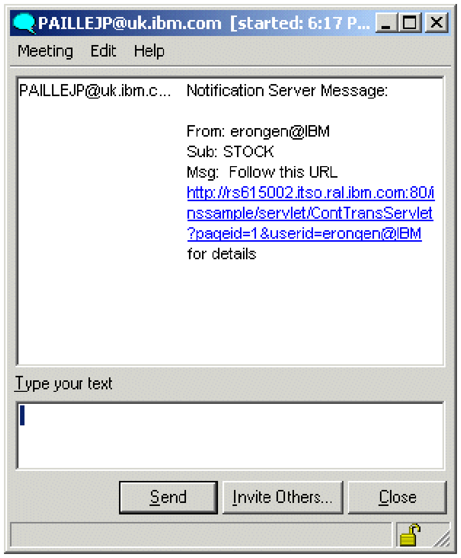
  cd /usr/IBMEPS/INS/samples

- Examine the file contentfeeder.sh (for example with the VI editor by entering `vi contentfeeder.sh` and typing **:q** when you are finished) and run it by entering the following command:

  ```
  ./contentfeeder.sh stocks stock
  ```

This will run the XMLContentFeeder class and will submit stocks content with source name stocks, using all files in the subdirectory stock. The content can be found in /usr/IBMEPS/INS/samples/stock/stock1.xml and /usr/IBMEPS/INS/samples/stock/stock1.xml (see above for details on the content feeder).

Depending on the user preferences and trigger definitions, the notification will be sent to the device the user has specified. The result for a notification through Sametime is shown in Figure 10-17.



PAILLEJP@uk.ibm.c... Notification Server Message:

From: erongen@IBM
Sub: STOCK
Msg: Follow this URL
http://rs615002.itso.ral.ibm.com:80/insample/servlet/ContTransServlet?pageid=1&userid=erongen@IBM
for details

Figure 10-17 Saved notification through SameTime for the stock example

10.6 Application development

10.6.1 Archives and resources

INS ships with several .jar files and resource files. Depending on the situation, you need the different classpath settings.

The following resource files must be in the classpath:

- IQ.properties
- ClientMsgs.properties and ClientMsgs_xx_YY.properties, where _xx_YY represent the locale you want to use.

By default, these classes are installed in /usr/IBMEPS/INS.

Further, examine the following classpath settings which are set up by the system upon installation. This shows you which.jar files are needed for each situation.

- For the WebSphere Application Server Server that runs the samples:

  ```
  -classpath
  /usr/IBMEPS/INS/lib/ins.jar:/usr/IBMEPS/INS/lib/jlog.jar:/usr/IBMEPS
  /INS/lib/xalan.jar:/usr/IBMEPS/INS/lib/xerces.jar:/usr/IBMEPS/INS
  ```

- The webapp has the following added to the above:

  ```
  /usr/IBMEPS/INS/samples/inssample/servlets
  ```

- For running the contentfeeder.sh shell script, that runs the XMLContentAdapter, set the classpath in your shell script by adding the following line:

  ```
  export
  CLASSPATH=<ins_home>:<ins_home>/lib/ins.jar:<ins_home>/lib/jlog.jar:
  <ins_home>/lib/xalan.jar:<ins_home>/lib/xerces.jar:<ins_home>/lib/in
  s.jar:<ins_home>/samples
  ```

  Where `<ins_home>` is the home of the INS installation; the default is /usr/IBMEPS/INS.

  Xalan and Xerces are needed because this content adapter uses XML processing. Other content adapters may need other classes.

10.6.2 Server side classes

As discussed above, the GatewayAdapters and TriggerHandlers are used by IQ. These classes must be installed on the server, and the IQ server must be restarted for the new code to become active. For debugging, you can print information to the console. Check the logging and tracing parameters in the LDAP Directory to see how log and tracing statements are handled (see "Configuring logging and tracing" on page 373 for details).

10.6.3 INS development using IBM VisualAge for Java

You can develop INS Applications using VisualAge for Java. Below, you will find detailed instructions on how to set up VisualAge for Java 4.0 to develop INS applications. If you need additional information on using IBM VisualAge for Java, please refer to one of the following IBM Redbooks:

- *Programming with VisualAge for Java Version 3.5*, SG24-5264-01
- S*ervlet and JSP Programming with IBM WebSphere Studio and VisualAge for Java*, SG24-5755-00

To develop INS applications using IBM VisualAge for JAva, you must load the .jar files discussed above into your workspace. It is good practice to create one or more separate projects for these classes. You can create a new project by selecting the menu item **Selected -> Add -> Project...**, and creating a new project. You can import the resources discussed above (IQ.properties, ClientMsg.properties and its locale extensions ClientMsg_xx_YY.properties) into this project.

Important: When you want to use Web Traffic Express and INS simultaneously in IBM VisualAge for Java, you will encounter the problem that these two features use different versions of the XML classes. Web Traffic Express uses the IBM XML Parser for Java that ships with IBM VisualAge for Java, and INS uses the classes in the file xerces.jar. When you try to load the Java archive xerces.jar, you encounter the error that classes cannot be loaded into two separate projects.

```
Add failed: The package org.w3c.dom cannot be specified by both ITSO
Java Other and IBM XML Parser for JAVA.
```

If you do not load the IBM XML Parser for Java, Web Traffic Express will not start.

To avoid this problem, you must create a new project and load that instead of the IBM XML Parser for Java. This project must contain only the following packages:

- com.ibm.xml.dom
- com.ibm.xml.framework
- com.ibm.xml.internal
- com.ibm.xml.internal.msg
- com.ibm.xml.parser
- com.ibm.xml.parser.util
- com.ibm.xml.parsers
- com.ibm.xml.xpointer

10.7 Extending the enterprise

YourCo was extended to demonstrate what Intelligent Notification Services can add to an existing enterprise. As an example, we have extended the *reserve a room* function of YourCo:

- Users who reserve a room can also invite colleagues to the meeting.
- These users will be directly notified on their device of choice.
- Users can subscribe to the meeting system, and be notified when a meeting is within an adjustable time. These notifications will be delivered to the device of choice.

The source code for this example can be found in Appendix A, “INS sample source code” on page 689. Further information on the YourCo application can be found in Chapter 3, “Enterprise sample applications” on page 77.

10.7.1 Deploying the notifications extensions to WebSphere

1. First, you must set up the database for the extension. A description of this process can be found in “Deploying Expertise Location to the application server” on page 457.
2. Ensure that the *from user* and *to user* you want to test with are registered in the LDAP directory (by default, these are insuser@IBM). Also, ensure that the user you are going to use for notifications (for example, Sean O’Connell) has the correct user ID.

   You can do this by opening the DB2 Command Line Processor and entering:

   ```
   Connect to sample
   Select * from WSDEMO.EMPLOYEE where FIRSTNME = 'SEAN'
   ```

   And, if you want to change something, you can enter for example:

   ```
   update WSDEMO.EMPLOYEE set USERID = 'insuser@IBM' where FIRSTNME =
   'SEAN'
   ```

3. Next, you must add the .jar files and resources described above to the classpath of the WSsamples_app Web application. Edit the file IQ.properties to contain the correct host name and port number (55001 is the default).

   ```
   iqSocketClientStub.port=55001
   iqSocketClientStub.host=<hostname>
   ```

   where `<hostname>` is the name of the server running the INS services.

   If you use the file that was created during setup of INS, these values are automatically set correctly.

4. Unzip the file inssamples.zip to the root directory of the application server, for example

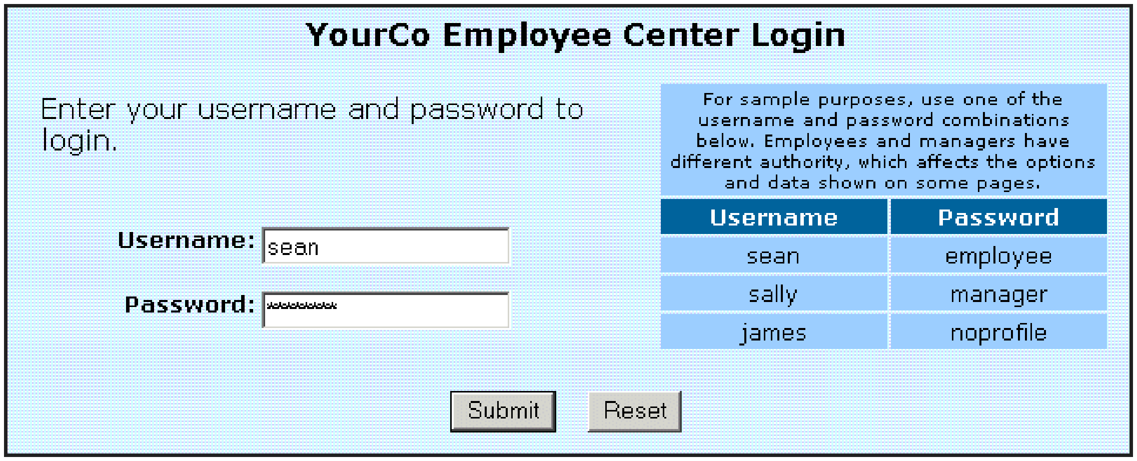
   ```
   C:\WebSphere\AppServer
   ```

5. Restart the Application Server and start the YourCo application as usual.

The following sections describe how to use the new functions.

10.7.2 Simple notification

To schedule a meeting, you must be logged in to YourCo. You must log in (see Figure 10-18) using the user ID you have set up in the previous section under step 2.



YourCo Employee Center Login

Enter your username and password to login.

Username: sean

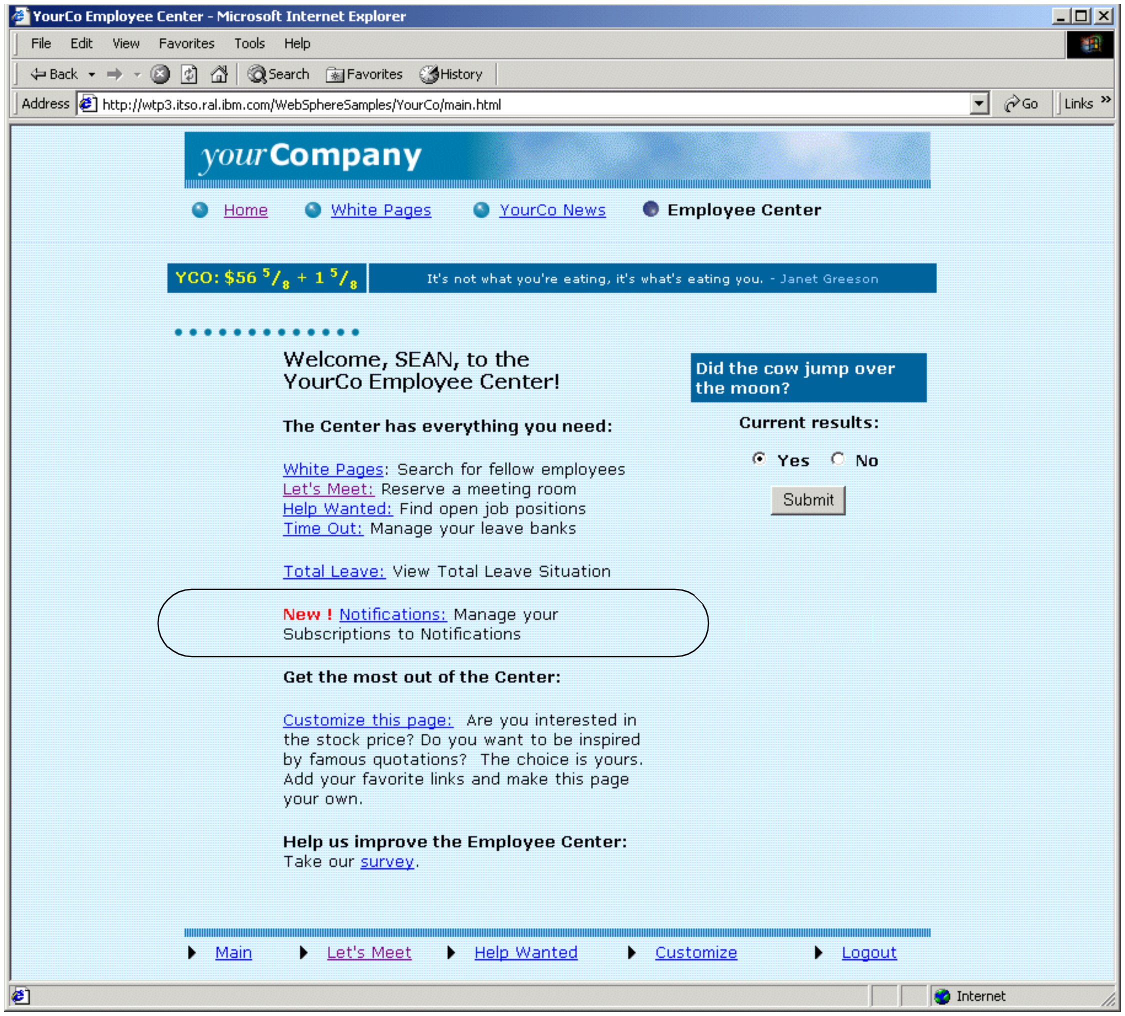
Password: ********

For sample purposes, use one of the username and password combinations below. Employees and managers have different authority, which affects the options and data shown on some pages.

Username	Password
sean	employee
sally	manager
james	noprofile

Figure 10-18 Logging in to the Employee Center of YourCo

After you have successfully logged in, you come to the enhanced Employee Center (see Figure 10-19).


YourCo Employee Center - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://wtp3.itso.ral.ibm.com/WebSphereSamples/YourCo/main.html
Go Links
yourCompany
Home White Pages YourCo News Employee Center
Internet

YCO: $56 $^5/_8$ + 1 $^5/_8$

It's not what you're eating, it's what's eating you. - Janet Greeson

Welcome, SEAN, to the
YourCo Employee Center!

The Center has everything you need:

White Pages: Search for fellow employees
Let's Meet: Reserve a meeting room
Help Wanted: Find open job positions
Time Out: Manage your leave banks

Total Leave: View Total Leave Situation

New ! Notifications: Manage your
Subscriptions to Notifications

Get the most out of the Center:

Customize this page: Are you interested in the stock price? Do you want to be inspired by famous quotations? The choice is yours. Add your favorite links and make this page your own.

Help us improve the Employee Center:
Take our survey.

Did the cow jump over the moon?

Current results:

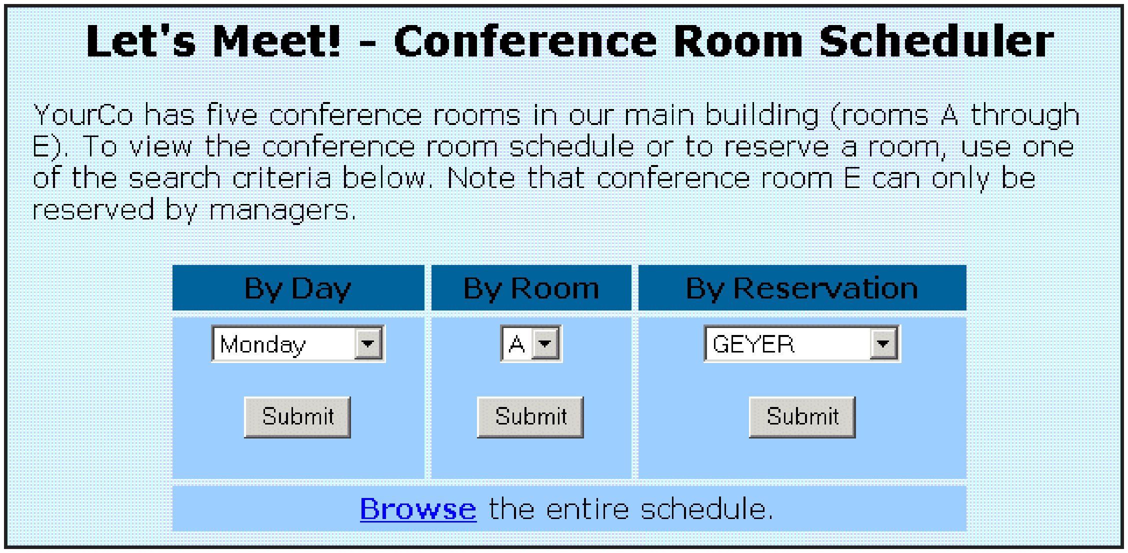
[✓ Yes] No

Submit

Main Let's Meet Help Wanted Customize Logout

Figure 10-19 The Employee Center

The new function to manage your notifications has been added. For now, we are first going to schedule a meeting. Select **Let's Meet**. This opens the reservation panel. Select a day, and select the **Submit** button (see Figure 10-20).



Let's Meet! - Conference Room Scheduler

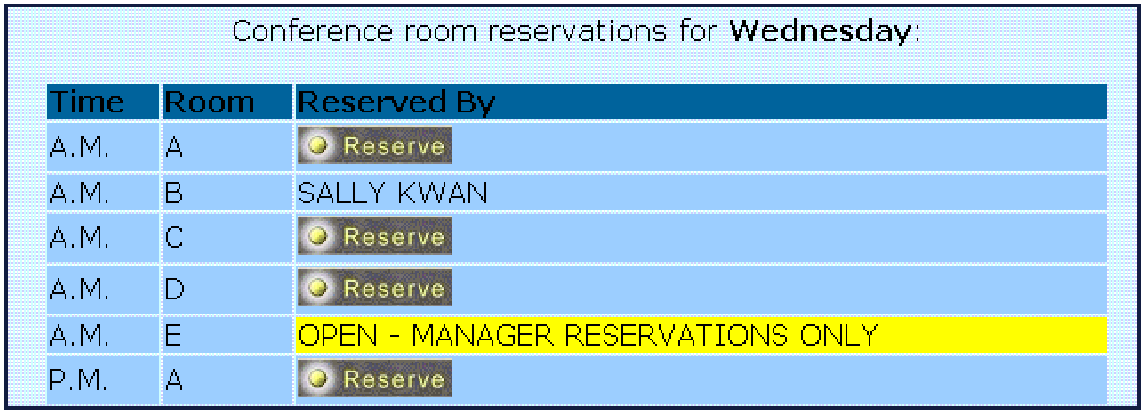
YourCo has five conference rooms in our main building (rooms A through E). To view the conference room schedule or to reserve a room, use one of the search criteria below. Note that conference room E can only be reserved by managers.

By Day	By Room	By Reservation
Monday	A	GEYER
Submit	Submit	Submit

Browse the entire schedule.

Figure 10-20 Conference Room Scheduler

You are presented with a list of available rooms for that day. You can select one of the **Reserve** buttons to make a reservation (see Figure 10-21).

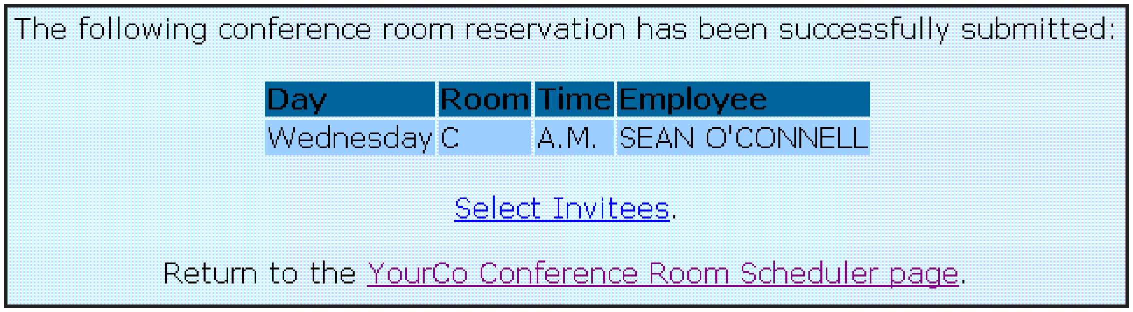


Conference room reservations for **Wednesday**:

Time	Room	Reserved By
A.M.	A	Reserve
A.M.	B	SALLY KWAN
A.M.	C	Reserve
A.M.	D	Reserve
A.M.	E	OPEN - MANAGER RESERVATIONS ONLY
P.M.	A	Reserve

Figure 10-21 Reserving a conference room

As illustrated in Figure 10-22, on the acknowledgement panel, you find the new function to Select Invitees.



The following conference room reservation has been successfully submitted:

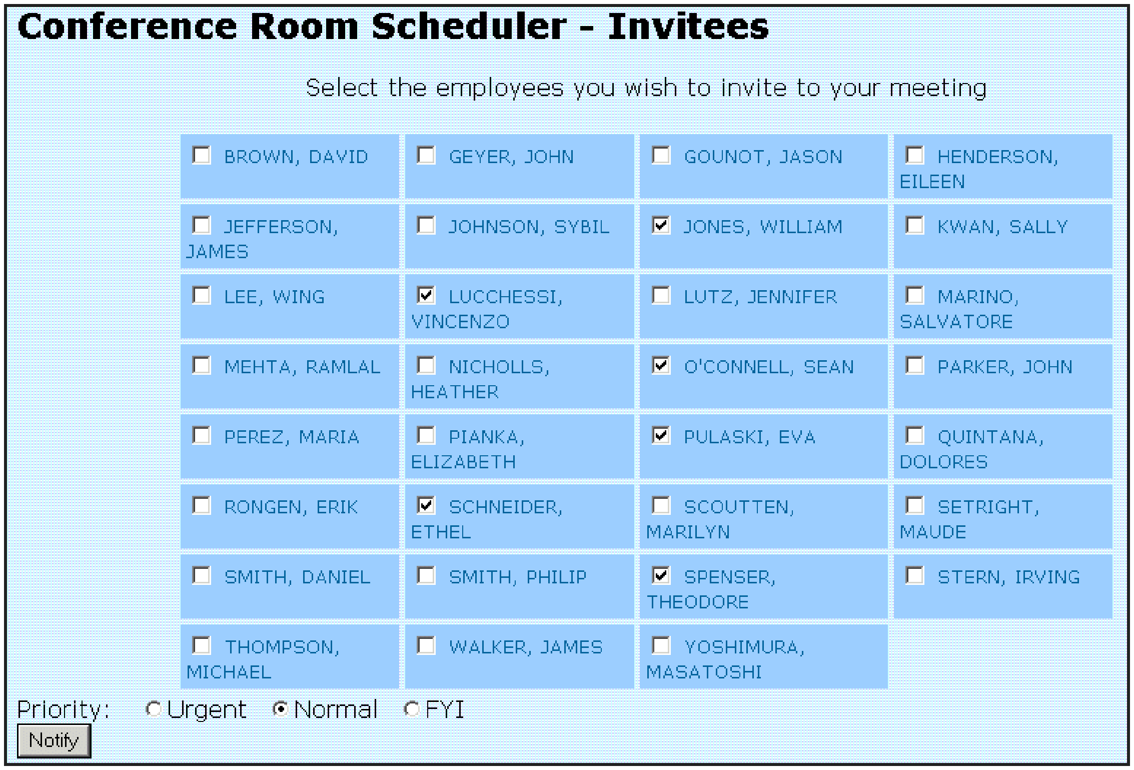
Day	Room	Time	Employee
Wednesday	C	A.M.	SEAN O'CONNELL

Select Invitees.

Return to the YourCo Conference Room Scheduler page.

Figure 10-22 Room reservation acknowledgement - Select Invitees

When you click **Select Invitees**, you will be presented with a list of employees of YourCo (see Figure 10-23), and you can select the users whom you want to be notified. Ensure that the user you have set up for INS is included in the list.



Conference Room Scheduler - Invitees

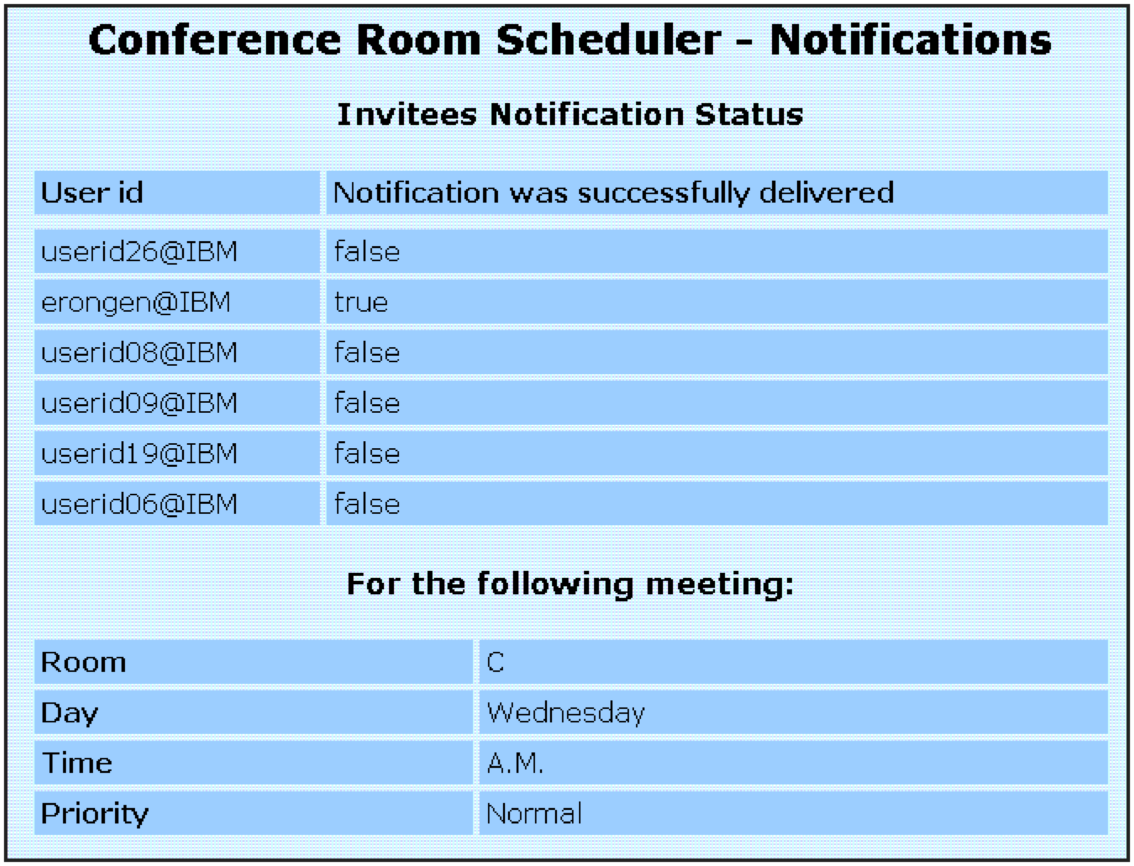
Select the employees you wish to invite to your meeting

☐ BROWN, DAVID	☐ GEYER, JOHN	☐ GOUNOT, JASON	☐ HENDERSON, EILEEN
☐ JEFFERSON, JAMES	☐ JOHNSON, SYBIL	☑ JONES, WILLIAM	☐ KWAN, SALLY
☐ LEE, WING	☑ LUCCHESSI, VINCENZO	☐ LUTZ, JENNIFER	☐ MARINO, SALVATORE
☐ MEHTA, RAMLAL	☐ NICHOLLS, HEATHER	☑ O'CONNELL, SEAN	☐ PARKER, JOHN
☐ PEREZ, MARIA	☐ PIANKA, ELIZABETH	☑ PULASKI, EVA	☐ QUINTANA, DOLORES
☐ RONGEN, ERIK	☑ SCHNEIDER, ETHEL	☐ SCOUTTEN, MARILYN	☐ SETRIGHT, MAUDE
☐ SMITH, DANIEL	☐ SMITH, PHILIP	☑ SPENSER, THEODORE	☐ STERN, IRVING
☐ THOMPSON, MICHAEL	☐ WALKER, JAMES	☐ YOSHIMURA, MASATOSHI	

Priority: Urgent [✓ Normal] FYI

Figure 10-23 Selecting invitees for the meeting, and setting the priority

If you submit this list, you will see the result of the simple notifications (see Figure 10-24). If you have selected users that are not set up properly in INS, you will see that the notifications have not been delivered successfully.



Conference Room Scheduler - Notifications

Invitees Notification Status

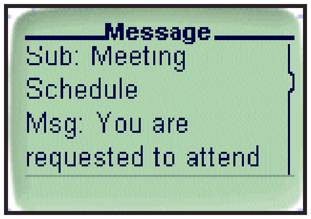
User id	Notification was successfully delivered
userid26@IBM	false
erongen@IBM	true
userid08@IBM	false
userid09@IBM	false
userid19@IBM	false
userid06@IBM	false

For the following meeting:

Room	C
Day	Wednesday
Time	A.M.
Priority	Normal

Figure 10-24 Confirmation of the notification

In the example discussed here, the user ID was set up to have the notifications delivered to a WAP device. Figure 10-25 shows the result of delivering the notification to the WAP simulator.



Sub: Meeting
Schedule
Msg: You are
requested to attend

Figure 10-25   Notification delivered to a WAP Push simulator

10.7.3 Subscription

How it works

We have created a function that does not use news feeds as input, but live application data, in this case information on meetings that have been scheduled. The code of the example is based upon the Stock example shipped with INS.

Connecting to the new function:

From the revised Employee Center (Example A-14 on page 750), the MeetingSubscriptionStartServlet is called (Example A-7 on page 724). This will add the user ID to the queryString and call the MeetingSubScriptionServlet (see Example A-8 on page 726).

Creating the content

As invitees are selected and notified, for each invitee in the meeting, an entry is also entered in the table WSDEMO.TRIGGER. This is done in the method addMeetingForTrigger() of class itso.WebSphere Everyplace Server.ins.samples.Notify (see Example A-3 on page 703). This table is the content that we are later going to feed into the IQ server using our own Content Adapter (see below).

The trigger

The trigger is registered just as discussed in "The Application Programming Interface" on page 408. This is done in servlet MeetingSubscriptionServlet (see Example A-8 on page 726 for details).

The SQL92 clause that is constructed is:

```
sqlsel = "(USERID = '" + userid + "') AND (TIME_TO_MEETING <= " +
meetingHours.elementAt(i) + ")";
```

where `userid` is the user ID of the user using the page, and `meetingHours` is the vector containing, for each trigger, the number of hours until the meeting. Note that the ContentAdapter must compute the number of hours remaining from the time the content is fed into IQ until the time of the meeting (see below for more information on the Content Adapter).

The MeetingHandler TriggerHandler

In Example A-11 on page 736, you can see that the matchHandle() method of the MeetingHandler was changed with respect to the sample. In the samples shipped with INS, it is assumed that the subscription servlets and the content providing servlet for saved messages reside on one instance of WebSphere Application Server. In our situation, we have changed this. Therefore, we must manipulate the URL that is constructed to retrieve saved content. Be it in an oversimplified way, this proves how the server configuration that is chosen in the samples can be changed to suit your own situation.

Example 10-12 Part of handleMatch() where the URL for saved content is changed

```
//change the notificationURL to another server
    System.out.println("MH => Old NotificationURL: " + notificationURL);
    String oldWebApp = "http://wtp3.itso.ral.ibm.com:80/WebSphereSamples";
    String newWebApp = "http://" + SubscriptionUtility.getHost() +
"/inssample";
    String uri = notificationURL.substring(oldWebApp.length());
    notificationURL = newWebApp + uri;
```

We have also added some println statements for debugging. They appear in the console. An example of this is shown below in Example 10-14 on page 432.

The ContentHandler

To see details, pleae see Example A-12 on page 740. This application runs continuously with delays of one minute (delayTime= 60000). Each minute, the database is scanned, and the content is adapted and sent to IQ.

One of the things that must be computed is the time left until the meeting. In the Method timeLeftToMeeting(String day, String time), the time that is left until the meeting is computed in hours. In this test version, it always returns `2`, which makes testing the triggers easy.

The method processContent() processes the data in DB2 and creates the ContentBody. It also calls the method sendContentToIQ(), which submits the content. Note that for each line in the database, a ContentBody is created and submitted.

Running the sample

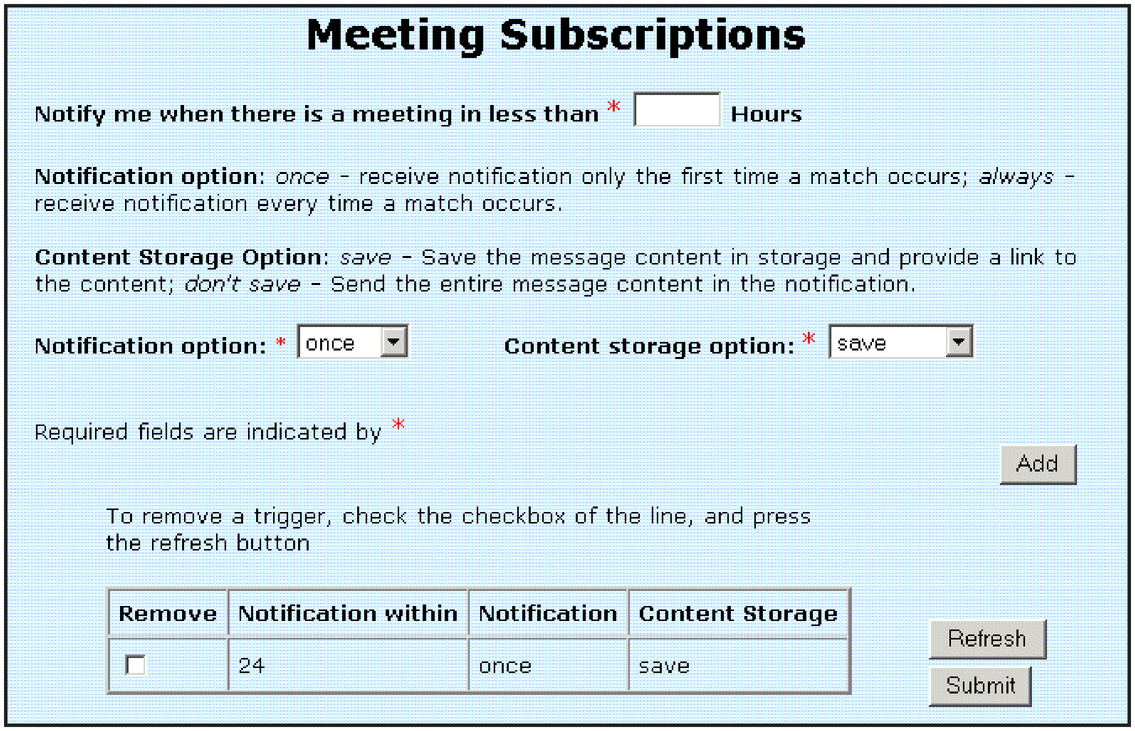
Most of the flow for the subscription has already been performed. When submitting the Invitees in Figure 10-23, for each user, notifications were added to the triggers database.

You can check this by opening the DB2 command line processor and entering:

```
Connect to sample
Select * from WSDEMO.TRIGGERS
```

and examine the entries.

The user must subscribe to the notifications. This is done by entering the URL as shown in Figure 10-26. The host name must be changed to your host name.



Meeting Subscriptions

Notify me when there is a meeting in less than * [ ] **Hours**

Notification option: *once* - receive notification only the first time a match occurs; *always* - receive notification every time a match occurs.

Content Storage Option: *save* - Save the message content in storage and provide a link to the content; *don't save* - Send the entire message content in the notification.

Notification option: * once **Content storage option:** * save

Required fields are indicated by *

To remove a trigger, check the checkbox of the line, and press the refresh button

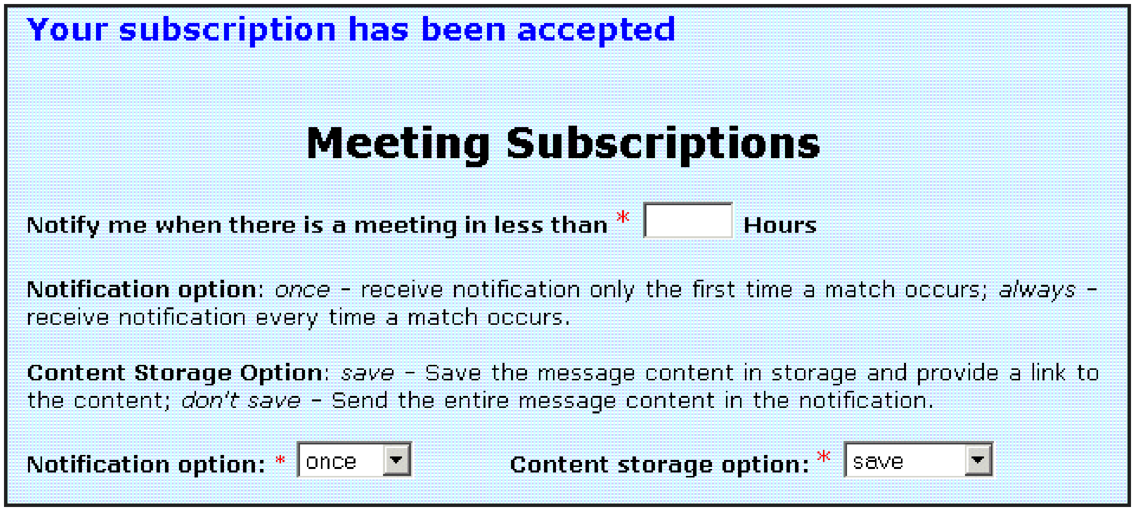
Remove	Notification within	Notification	Content Storage
☐	24	once	save

Figure 10-26 Subscription panel for meeting notifications

Add a subscription for more than two hours. Remember that in our test scenario, the ContentAdapter will, for each meeting, determine that the time left to the meeting is two hours.

Example 10-13 IQ Server trace of trigger being added to IQ

```
2001.09.21 13:37:38.053   IBM Intelligent Notification Service iQueue rs615002
  1001093858053,addTrg,9,erongen@IBM,8000000000000009
```



Your subscription has been accepted

Meeting Subscriptions

Notify me when there is a meeting in less than * [ ] Hours

Notification option: *once* - receive notification only the first time a match occurs; *always* - receive notification every time a match occurs.

Content Storage Option: *save* - Save the message content in storage and provide a link to the content; *don't save* - Send the entire message content in the notification.

Notification option: * once

Content storage option: * save

Figure 10-27 The trigger has been successfully added to INS

Finally, you must run the ContentAdapter. For example:

- Add db2java.zip for jdbc2.0 to your classpath
- You may have to run `C:\Program Files\SQLLIB\java12\usejdbc2.bat` and then use the file C:\Program Files\SQLLIB\java\db2java.zip. if you have not yet set up DB2 to use JDBC2.
- Assuming the files ins.jar, jlog.jar, IQ.properties, ClientMsgs.properties and the appropriate ClientMsgs_xx_YY.properties are in the current directory, and the Java home directory is in the path of your system, you can run the following command to run the MeetingContentAdapter:

```
java -cp ".;ins.jar;jlog.jar;C:\Program Files\SQLLIB\java\db2java.zip"
itso.WebSphere Everyplace Server.ins.triggersample.MeetingContentAdapter
```

You will see that the IQ trace and the UND trace show activity (see below for the IQ trace), and a notification will be delivered to your device.

Example 10-14 IQ server trace of trigger being fired

```
MH => MeetingHandler.handleMatch() called
MH => Old NotificationURL:
http://wtp3.itso.ral.ibm.com:80/WebSphereSamples/serv
let/
MH => New NotificationURL: http://rs615002.itso.ral.ibm.com/inssample/servlet/
MH => triggerOption = once
MH => contentOption = save
MH => Userid = erongen@IBM
MH => Day = friday
MH => Time = A.M.
MH => Room = A
MH => Convenor = friday
MH => Creating content for save
MH => contentString =
<?xml version="1.0" encoding="UTF-8"?>
<meeting>
  <convenor>Erik Rongen</convenor>
  <userid>erongen@IBM</userid>
  <day>friday</day>
  <time>A.M.</time>
  <room>A</room>
</meeting>
MH => Creating message for save
MH => message =
<?xml version="1.0" encoding="UTF-8"?>
<message>
<to>erongen@IBM</to>
<from>erongen@IBM</from>
<subject>Meeting</subject>
<text> Follow this URL
http://rs615002.itso.ral.ibm.com/inssample/servlet/ContTr
ansServlet?pageid=
MH => message2 =
&amp;userid=erongen@IBM for details </text>
</message>
MH => triggerResult =
com.ibm.pvc.we.ins.iqueue.RetentionSpecification@4d194054
MH => triggerResult =
com.ibm.pvc.we.ins.iqueue.RetentionSpecification@4d194054
MH => MeetingHandler.handleMatch() ended successfully for SAVED
2001.09.21 13:37:38.270   IBM Intelligent Notification Service iQueue rs615002
  [5:9] 1001093858270erongen@IBM,8000000000000009
```

10.8 Problem Determination

Some common problems and possible causes are listed in this section.

The service cannot be started

If a service will not start, one possible cause could be that the port is already busy, possibly because of a previous instance of the same service. You can test this situation by entering the following command at the prompt of your Unix system:

```
netstat -a
```

Example 10-15 shows the default ports that are assigned to the INS services.

Example 10-15 Ports assigned to INS

```
1506 - used by Gryphon Broker
6789 - used by DB2 JDBC listener
55001 - used by IQ clients and server
55002 - used by IQ Admin client and server
55003 - used by IQ system logger (if active)
55004 - used by IQ trace logger (if active)
55005 - used by UND for messaging
55006 - used by UND Admin client and server
55007 - reserved for Secure Context Server
55008 - used by UND logging (if active)
55009 - used by UND trace (if active)
55010 - used by Preference Engine logging (if active)
55011 - used by Preference Engine trace (if active)
```

If a process is occupying one of these ports, find the process ID by entering the following command:

```
ps -ef | grep <proc>
```

where **<proc>** is a search string for the grep function and denotes the process you want to terminate.

You will then stop the process by entering the command:

```
kill -9 <procID>
```

where **procID** is the ID of the process that you want to kill. Be careful when you do this, especially when you are logged on using the root ID.

IQ will not start

Ensure that UND and GB have been started and are running properly. Ensure that the JDBC driver is running. see “Starting and stopping the WebSphere Everyplace Server components for INS” on page 374 for details.

Notifications are accepted by IQ, but nothing is delivered

Check in the LDAP directory that for the given *from user* and *priority*, the user has defined devices for delivery.

Check that the remote server which the notification has to be sent to is correctly configured in LDAP and is up and running. Some remote servers may let INS know that the message has been delivered, and others may not. Also, the network may be so slow that the connection to the remote server is timed out before successfully sending the notification.

Also check that the client device or emulator is up and running

The subscription pages do not show

Check that the WebSphere Application Server is up and running, and that the inssampleserver is up an running. The latter can be checked using the Admin Client for WebSphere Application Server

The YourCo Extension does not work

Ensure you have followed the steps laid out in “Deploying the notifications extensions to WebSphere” on page 422.

Ensure you have installed the files to the right directories. If you have exported the .zip file to the right place, the file ins.jar should be located in:

```
<webserver_root>hosts\default_host\WSsamples_app\servlets
```

Check the standard output and standard error files of the application server:

```
<webserver_root>\logs\default_server_stout.log
<webserver_root>\logs\default_server_sterr.log
```

If these files have been relocated, you can find their location in the administration console of the WebSphere Application Server.

Ensure that the INS required resources (IQ.properties, ClientMsgs.properties and ClientMsgs_xx_YY) are in the classpath of the server.

11

Location-Based Services (LBS)

In this chapter, Location-Based Services (LBS) is presented. LBS is a new component provided by WebSphere Everyplace Server (WES) Version 2.1.1 Service Provider Offering (SPO). Location-Based Services (LBS) allows you to use end-user geographical location to provide relevant functionality or content to your Web and enterprise applications.

Note: LBS is not available in WES Version 1.1 Enable Offering (EO).

This chapter discusses the following topics:

- An overview of Location-Based Services
- How to install and run the sample code provided with LBS
- How to develop Location-Based applications using VisualAge for Java
- A sample scenario that shows you how a B2E application can be extended to support Location-Based Services (LBS)
- Integration with the LBS run-time services

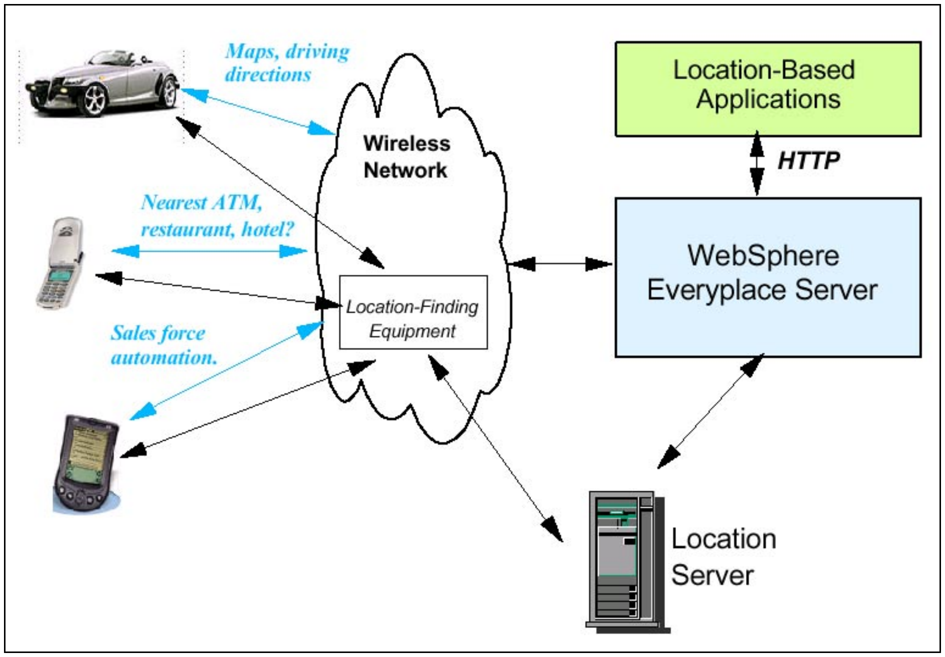
11.1 Overview

Location Based Services (LBS) has been implemented in WebSphere Everyplace Server Version 2.1.1 and provide the infrastructure needed to create *Location-Based Applications* (LBA). When using LBS, Web and enterprise applications use the geographical location of the request originator to provide the user with location-specific information.

The implementation of Location-Based Services in WebSphere Everyplace Server includes the following support:

- Provides applications with user location information.
- Provides a fine-grained authorization mechanism to control the access of applications to the user location information. This way, LBS guarantees the privacy of the users with respect to their location.

For example, if someone wants to find a nearby restaurant, even if the user does not know his or her exact location (address), LBS provides this information to the application, which can then look up a restaurant in the designated area.



Maps, driving directions
Wireless Network
Location-Based Applications
HTTP
Nearest ATM, restaurant, hotel?
WebSphere Everyplace Server
Location-Finding Equipment
Sales force automation.
Location Server

Figure 11-1 Location-Based Services (LBS)

In addition, if for any reason the Service Provider has also implemented an application that sends advertisements about shops in the user's area or vicinity, the user can optionally make use of the integrated authorization system of LBS to disallow the specific application queries about the user's information, and therefore prevent it from sending unsolicited advertisements.

11.1.1 How the user location is calculated

The user location is calculated by defining the position of the device the user has used to place the request. Figure 11-2 illustrates how this mechanism is commonly used and how the location of the mobile device is determined.

For example, a cellular phone, when operational, is always connected to the wireless network through an antenna tower, a so-called *cell tower*. The total area that is covered is serviced by many cell towers and the coverage area of each cell tower, the cell, is split into multiple cell sectors.

The Service Provider uniquely identifies each cell sector by a *Subscriber CellID* which is made up of the following values:

- Market ID
- Switch ID
- Cell ID
- Sector ID

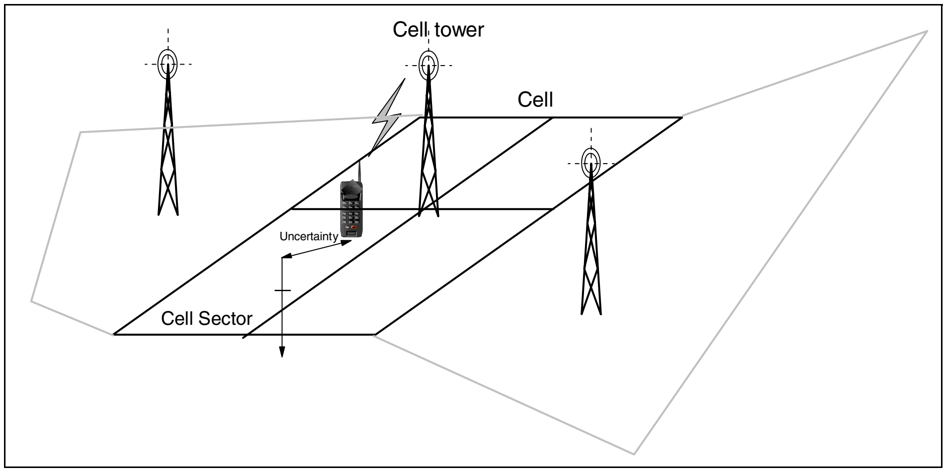


Figure 11-2 Defining the geographical location of the requesting device

Each cellular phone (when operational) is in a coverage area identified by a unique Subscriber CellID. As users are roaming the different cells and sectors with their mobile devices, the Service Provider maintains a dynamic table, mapping the Mobile Identification Number (MIN) of each cellular phone to the Subscriber CellID of the cell sector in which the device is located.

The Service Provider also maintains the latitude and longitude of all its cell towers and therefore can compute the latitude and longitude of any cell sector, which is taken as its geographical position. Since every cellular phone is covered by a Subscriber CellID, the latitude and longitude of the corresponding cell sector is the location of the subscriber.

Note: The center of the cell sector defines the geographical location of the device.

This scheme introduces an uncertainty in position, which is based upon the size of the cell sector, and which typically ranges from a few tens of meters up to a few kilometers in some cases.

11.2 Location-Based Services

The Location-Based Services component is provided as an integrated part of the WebSphere Everyplace Server environment, and therefore takes full advantage of the scalability and security options provided by WebSphere Everyplace Server.The LBS architecture is shown in Figure 11-3.

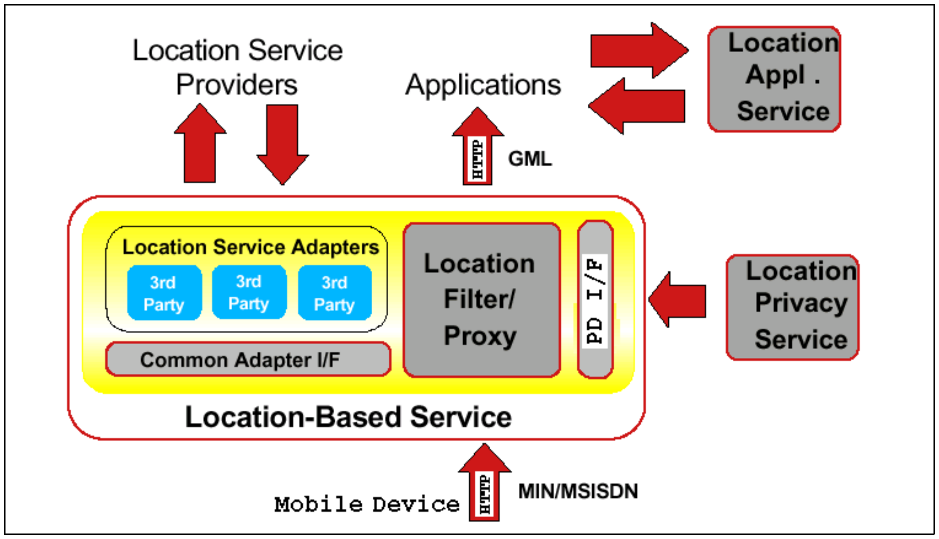


Figure 11-3 LBS architecture

The functionality provided by LBS in a WebSphere Everyplace Server environment provides the following support:

- LBS is a service of WebSphere Everyplace Server that dynamically locates mobile end-users and supplies their location to applications.
- It supports user-initiated location requests.
- It enables administrators to register which applications require location information.
- It allows users to control which location applications may receive information about their location.

  For example, an application can provide the user, based on his or her current location, with a list of the nearest teller machines, if so required.

Note: LBS has been implemented as a plug-in of Web Traffic Express (WTE), and therefore acts as a proxy for HTTP requests. The required location information is passed in the header of the HTTP request.

Figure 11-4 illustrates how LBS integrates within WebSphere Everyplace Server.

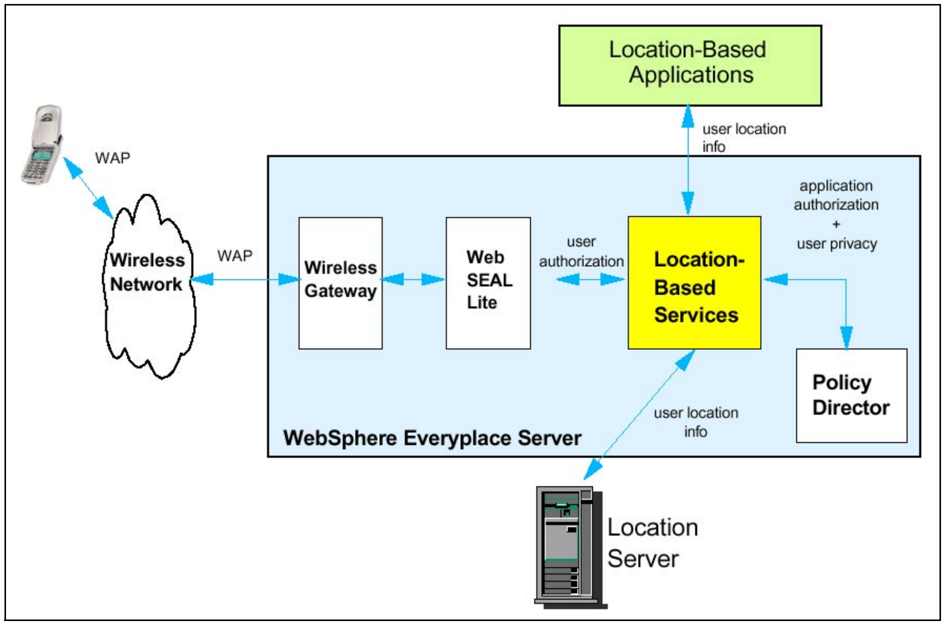


Figure 11-4 Location-Based Services (LBS) in WebSphere Everyplace Server

11.2.1 Location request

The function of each component in a location request flow for a Location Based Application can be summarized as follows:

- The Wireless Gateway maps the incoming request from the wireless device to an HTTP request.
- WebSEAL-Lite is the component in WebSphere Everyplace Server that ensures authentication. In the case of LBS, authentication is required. WebSEAL-Lite checks that the user is authenticated (which has already been done by the Wireless Gateway), and then passes the flow on to the LBS server.
- LBS is a plug-in of Web Traffic Express, and acts as a forward proxy. It has two main functions in the flow:

  a. LBS communicates with Policy Director to see if the application may receive the location information of the user.

  b. It requests the location information from the Location Server.

- The Location Server provides the location information. Based upon the telephone number of the user, the Location Server will determine where the user is located. The Location Server may also provide additional information such as address, nearest intersection, and so on. Currently, LBS supports the SignalSoft Server.
- The Location Services Bean and the GML Parser Bean are part of the LBS API, and are discussed in Section 11.4, “Developing location-based applications” on page 448 in some detail.

LBS request flow

Figure 11-5 illustrates how a request to a Location Based Application (LBA) flows through WebSphere Everyplace Server.

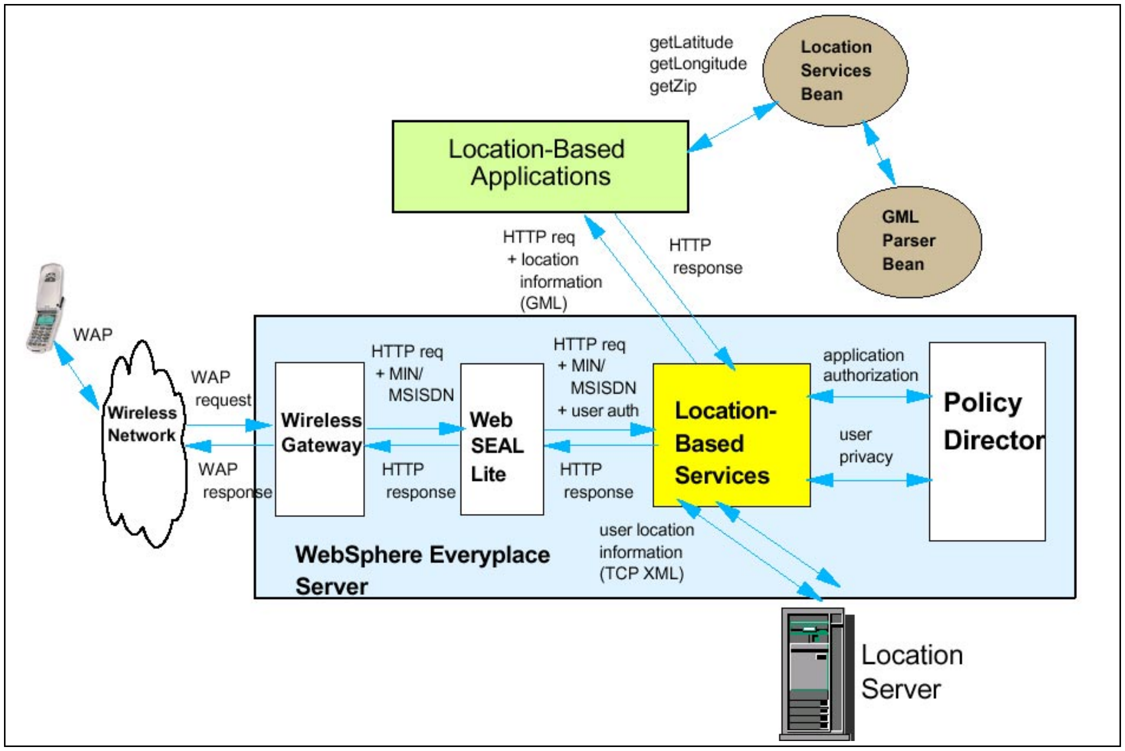


Figure 11-5 Location request in LBS

When a mobile device requests services of a Location-Based Application, the request will result in the following flow:

1. The mobile device sends a WAP request to the application on the wireless network. The request is routed through the Wireless Gateway that the device has configured as its gateway for WAP requests.
2. The Wireless Gateway authenticates the client, creates an HTTP request and adds the user ID and phone number (MIN) to the HTTP request header.
3. WebSEAL-Lite (WSL), configured as a reverse proxy, ensures that every request that gets through is authenticated.

   In WebSphere Everyplace Server, WebSEAL-Lite (WSL) is a plug-in of Web Traffic Express (WTE). Therefore, for performance reasons, WTE can be optionally configured to route to LBS only those requests that require LBS services. When implementing this approach, other requests will flow directly to the application servers. This type of request will have to be based on the target URL of the request.

   Note: In the sample scenarios described in this chapter, the request is always passed on to the Location-Based Services.
4. The Location-Based Services acts as a proxy. Before the actual process of the LBS request, it does the following:
   a. First, it checks in Policy Director if the application is a Location-Based Application (LBA).
   b. Then it checks in Policy Director if the application is allowed to request the location information for the user.
5. When these two previous items are checked, LBS sends the request for location information to the Location Server. LBS adds the response information in a request header, and passes the request on to the Location-Based Application.
6. The application accesses the location information using the LBS API.
7. When the application is finished, the response is sent back to the client device through LBS, WebSEAL-Lite and the Wireless Gateway.

11.2.2 Location Server

LBS supports the SignalSoft Local.info Server Version 3.2 from Solaris, including the customized WhereAmIservice. It uses the MIN to provide the location information. The SignalSoft Service usually resides in the Network Service Provider infrastructure and interacts with the Service Provider information servers to determine the location information, based upon the MIN. Apart from the geographical information of the cell phone (latitude, longitude), SignalSoft can also provide additional information such as an address using the mechanism of reverse geo-coding. This means that additional information about the location of the user is obtained using the longitude and latitude of the location.

LBS interfaces with SignalSoft Using the SignalSoft Service Interface (SSI), and calls the WhereAmI function of the SignalSoft system. The interface uses XML to exchange data with SignalSoft:

Example 11-1 Input to SignalSoft server

```
<?xml version="1.0" ?>
<!DOCTYPE LIContent (View Source for full doctype...)>
<LIContent>
  <LIVersion version="1.3" />
  <LIServiceRequest>
    <liTransactionID>1</liTransactionID>
    <LISubscriberID>
      <liSubscriberIDType type="MIN" />
      <liSubscriberIDValue>9195438817</liSubscriberIDValue>
    </LISubscriberID>
    <LISubLocation>
      <liCellID>6250|1|13|1</liCellID>
    </LISubLocation>
    <liServiceType>WHEREAMI</liServiceType>
    <liNumberBusiness>1</liNumberBusiness>
    <LIContentTypes>
      <liContentItem item="Address" />
      <liContentItem item="Postcode" />
      <liContentItem item="LatLong" />
      <liContentItem item="Phone" />
      <liContentItem item="OtherLocation" />
      <liContentItem item="LocDesc" />
    </LIContentTypes>
  </LIServiceRequest>
</LIContent>
```

Example 11-2 Output from SignalSoft Server

```
<?xml version="1.0" ?>
<!DOCTYPE LIContent (View Source for full doctype...)>
<LIContent>
  <LIVersion version="1.3" />
  <LIServiceResponse>
    <liTransactionID>1</liTransactionID>
    <liServiceType>WHEREAMI</liServiceType>
    <liNumberBusiness>1</liNumberBusiness>
    <liZone>80302</liZone>
    <liLocDesc>22ND ST and SPRUCE ST</liLocDesc>
    <LIContentInfo>
```

```
      <LIContentService>
        <liAddress>2226 PEARL ST</liAddress>
        <liPostcode>803024630</liPostcode>
        <LILatLong>
          <liLatitude>40020712</liLatitude>
          <liLongitude>-105265400</liLongitude>
          <liUncertainty>0</liUncertainty>
        </LILatLong>
        <liPhone>3034404167</liPhone>
        <liOtherLocation>Boulder, Boulder, CO, USA</liOtherLocation>
      </LIContentService>
    </LIContentInfo>
    <LIReturnStatus>
      <liStatusCode code="0" />
    </LIReturnStatus>
  </LIServiceResponse>
</LIContent>
```

Location-Based Services obtains the following information from SignalSoft:

– Latitude
– Longitude
– Uncertainty of the position estimate
– County
– City
– State/province
– ZIP/postal code
– Country
– Nearest intersection

For more details on the SignalSoft Server, refer to the WebSphere Everyplace Server Infocenter.

11.3 LBS example

Location-Based Services come with an example that demonstrates the LBS functionality. This sample demonstrates Location-Based Services finding and displaying location information.

11.3.1 Installation

The latest version of the example can be obtained by downloading the WebSphere EveryPlace Suite SDK from

```
http://www-3.ibm.com/pvc/products/wes/dev_resources.shtml.
```

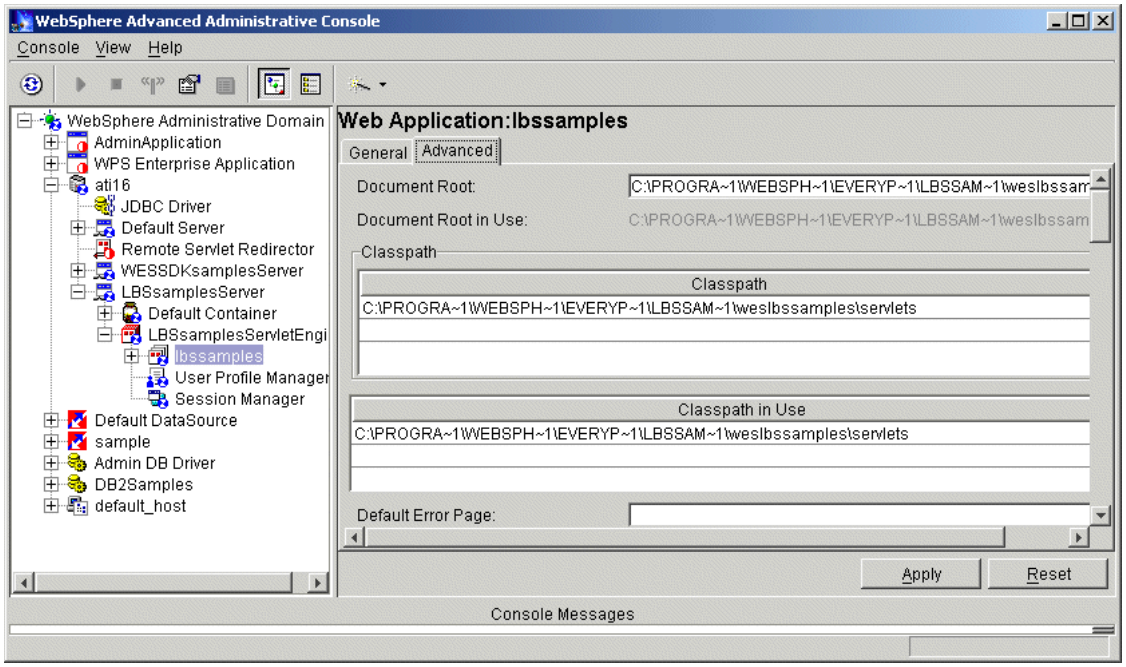
You will download a file named WebSphereEveryplaceSDK.exe, which will install the WebSphere Everyplace Server SDK when executed. Install the SDK using the default settings.

The SDK contains an infocenter that has details on the sample and on the installation of the sample. If you have installed the SDK using the default settings, the infocenter can be started by entering the following address in your browser:

```
C:\Program Files\WebSphere\Everyplace Suite SDK\infocenter\index.html
```

You can find information on how to install the LBS samples by selecting **EveryPlace Suite SDK-> Run Samples-> Location Based Services** and selecting **Location Based Services application sample** from the top of the page, which brings you to the next page, where you will select **Installing the Location-Based Services sample**.

Note: Make sure that WebSphere Application Server Advanced Edition version 3.5.4 or higher is used. The installation of the SDK will install a new server in WebSphere Application Server. After the installation, you can verify this by opening the WebSphere Application Server administration console and browsing the settings of the server LBSSampleServer, as shown in Figure 11-6.



Document Root: C:\PROGRA~1\WEBSPH~1\EVERYP~1\LBSSAM~1\weslbssam

Document Root in Use: C:\PROGRA~1\WEBSPH~1\EVERYP~1\LBSSAM~1\weslbssam

Classpath

Classpath
C:\PROGRA~1\WEBSPH~1\EVERYP~1\LBSSAM~1\weslbssamples\servlets

Classpath in Use
C:\PROGRA~1\WEBSPH~1\EVERYP~1\LBSSAM~1\weslbssamples\servlets

Default Error Page:

Figure 11-6 WebSphere Administration Console showing the LBS server and its contents

11.3.2 Running the sample application

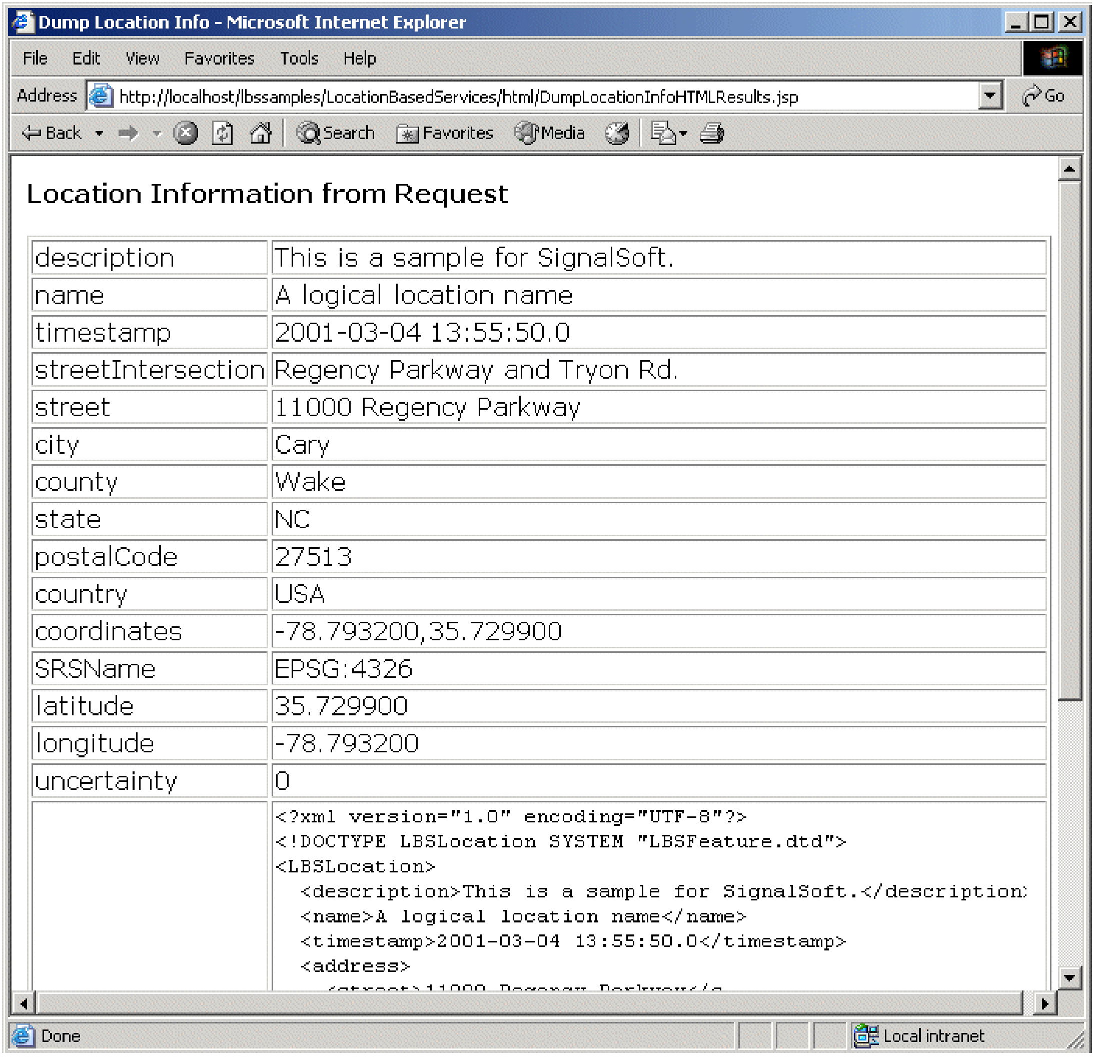
You can run the example by entering the following URL in a browser:

```
http://<hostname>/lbssamples/LocationBasedServices/html/DumpLocationInfoHTM
LResults.jsp
```

or enter this URL in a WAP emulator:

```
http://<hostname>/lbssamples/LocationBasedServices/wml/DumpLocationInfoWMLR
esults.jsw
```

where `<hostname>` is the host name of the server on which you installed the sample. The results are shown in Figure 11-7 and Figure 11-8, respectively.

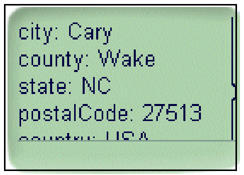

Dump Location Info - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/lbssamples/LocationBasedServices/html/DumpLocationInfoHTMLResults.jsp
Back Search Favorites Media
Done
Local intranet

Location Information from Request

description	This is a sample for SignalSoft.
name	A logical location name
timestamp	2001-03-04 13:55:50.0
streetIntersection	Regency Parkway and Tryon Rd.
street	11000 Regency Parkway
city	Cary
county	Wake
state	NC
postalCode	27513
country	USA
coordinates	-78.793200,35.729900
SRSName	EPSG:4326
latitude	35.729900
longitude	-78.793200
uncertainty	0

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE LBSLocation SYSTEM "LBSFeature.dtd">
<LBSLocation>
  <description>This is a sample for SignalSoft.</description
  <name>A logical location name</name>
  <timestamp>2001-03-04 13:55:50.0</timestamp>
  <address>
```

Figure 11-7 Location Based Services Sample in a browser



```
city: Cary
county: Wake
state: NC
postalCode: 27513
```

Figure 11-8 The Location Based Services Sample in a WAP emulator

If you use a real phone and the required WebSphere Everyplace Server components are set up properly, the sample displays real geographic information for the user's location. However, the LBS API has been designed such that you do not need all of the WebSphere Everyplace Server run-time components shown in Figure 11-5 to be installed in order to test your application. You only need the Java API. If the supporting Everyplace software, such as the Everyplace Wireless Gateway, the Location Based Services box, Tivoli SecureWay Policy Director or SignalSoft, is not set up properly, or the URL of the Location Based Application is not configured as a Location Based Application in Policy Director, the sample displays location information that is hardcoded in a file located in the classpath of the Web Application. In the case of the sample, it is located at:

```
<WES SDK root>\lbssamples\weslbssamples\servlets\com\ibm\lbs\test.xml>
```

where `<WES SDK root>` is the install directory of the WebSphere EveryPlace Suite SDK.

Note: In a production environment, this file should be removed.

11.4 Developing location-based applications

Location-Based Services provide an API that allows you to access the location information that LBS has included in the HTTP header of the request.

The Java classes of the API are part of the WebSphere EveryPlace Suite SDK. The SDK and the LBS example are installed as discussed in section 11.3, “LBS example” on page 444. The API is contained in a .jar file, that can be found at:

```
<WES SDK Root>\lbssamples\weslbssamples\servlets\lbsbeans.jar
```

where `<WES SDK Root>` is the root directory where you installed the WebSphere EveryPlace Suite SDK.

If you do not install the sample, the lbsbeans.jar can be found in the archive:

```
<WES SDK root>\lbssamples\weslbssamples.war
```

which is compressed using the .ZIP format and can be browsed with any regular compression tool or with the **`jar`** command of the Java JDK. You must extract lbsbeans.jar from this archive. It will be extracted to the archive:

```
<root-dir>\WEB-INF\lib\lbsbeans.jar
```

where `<root-dir>` is the directory where the .war file is extracted to.

A file with the hardcoded location information for testing purposes is in the same .war file at:

```
<root-dir>\WEB-INF\classes\com\ibm\lbs\test.xml
```

There are four classes used, which we will discuss now.

LocationServices bean

This bean provides the API that is mostly used in programming for LBS. It uses the other classes below to access the header information. LocationServices provides access to the location information given by LBS, as shown in Table 11-1.

Table 11-1 Location information

Property	**Description**
City	The city in which the mobile device is located.
Coordinates	The geographical coordinates of the mobile device (longitude, latitude.).
Country	The country in which the mobile device is located.
CountryDistrict	The district in which the mobile device is located.
Description	The description of the GML file.
Latitude	The latitude of the location of the device.
Longitude	The longitude of the location of the device.
Name	A logical name of the location.
OtherLocation	The other location information available. This is an array of strings, and can be accessed using the method otherLocation(int index).
PostalCode	The postal or zip code of the location. This depends on the country.
SRSName	Returns the geodetic datum name. This information is used to interpret the coordinates, or longitude and latitude in a world system.
StateProvince	The state or province in which the mobile device is located.

Property	Description
Street	The street in which the mobile device is located.
StreetIntersection	The intersection with this street that is nearest to the mobile device.
Timestamp	The timestamp of the request to the Location Server (Signalsoft).
Uncertainty	The uncertainty with which the location was determined.

The next two classes are used by the LocationServices class, and mostly you do not access them directly:

LocationDTDLocator

This class is used to locate the necessary DTDs for location services. It serves as the entity resolver when parsing the GML.

LocationGMLParse

This class provides easy access to the data in the header.

In addition, the following exception is also used:

NoLocationInfoException

This exception is thrown when there is a problem obtaining the location information. This means that the header *-ibm-pvc-user-location* is not present, or the data in it is not of the correct format.

Note:
When the LBS system is not present, the API will use the text file com\ibm\lbs\test.xml in the classpath to populate the header. A NoLocationInfoException is *only* thrown when this file is also not found.

In Figure 11-9, you find a fragment of the sample JSP we have discussed in the previous section. Next, this sample will be discussed in more detail to explain the LBS Application Programming Interface.

```
... The beginning of the jsp file
<BODY>
  <%@ page errorPage="" %>

  <%@ page import="com.ibm.lbs.LocationServices" %>
  <%@ page import="com.ibm.lbs.NoLocationInfoException" %>
  <jsp:useBean id="locationServices" type="com.ibm.lbs.LocationServices"
class="com.ibm.lbs.LocationServices" scope="request"></jsp:useBean>

<B>Location Information from Request</B><BR><BR>

  <%
  try
  {
    locationServices.setLocation(request);
  }
  catch (com.ibm.lbs.NoLocationInfoException e)
  {
  %>
     <P>No location was information present in the request.</P>
  <%
  }
  try
  {

  %>
  <TABLE border="1">
...
<TR>
<TD>street
</TD>
<TD>
  <%= locationServices.getStreet() %>
</TD>
</TR>
<TR>
<TD>city
</TD>
<TD>
  <%= locationServices.getCity() %>
</TD>
</TR>

... The rest of the JSP file
```

Figure 11-9 Fragment of the sample JSP

Line 6 in Figure 11-9 shows the creation of the locationServices Bean. In line 13 the location of the bean is set, using the information in the header in the httpRequest (or from the file, if the header is not present). At this point, the bean is populated with the location information, and the appropriate getters can be used to access the information. See for example lines 30 and 37 where the Street and City are retrieved.

Note: For more information on JavaServer Pages, refer to the IBM redbook *Servlet and JSP Programming with IBM WebSphere Studio and VisualAge for Java,* SG24-5755-00.

Location Based Services add location information into the HTTP header. The name of the header is x-ibm-pvc-user-location, and you can use this in Java to access the header directly.

The format of the information is Geography Markup Language (GML), an XML-based standard, maintained by the Open GIS Consortium. More information on GML can be found at:

```
http://opengis.net/gml/
```

The DTD (the Document Type Definition, defining the rules of an XML document), used for the GML information in LBS, is shown in Figure 11-10.

```
<?xml version="1.0" encoding="UTF-8"?>
 <!ENTITY % GMLGEOMETRYDTD SYSTEM "gmlgeometry.dtd">
 %GMLGEOMETRYDTD;
 <!ELEMENT LBSLocation (description?, name?, boundedBy?, timestamp?,
address?, geoLocation?, property*) >
 <!ELEMENT address (street?, otherLocation?, zipcode-postalcode?,
intersectionStreet?) >
 <!ELEMENT geoLocation (locationShape?, locationPoint ) >
 <!ELEMENT locationShape (omni|arc|ellipse|otherShape) >
 <!ATTLIST locationShape
          units (Miles|KM) "Miles">
 <!ELEMENT omni (radius) >
 <!ELEMENT arc (innerRadius, outerRadius, startAngle, stopAngle) >
 <!ELEMENT ellipse (majorRadius, minorRadius) >
 <!ELEMENT otherShape (property*) >
 <!ELEMENT locationPoint (Point,uncertainty?) >
 <!-- Simple properties hold the property value as parsed character data.
 The type of the value is specified by the type attribute, which defaults to
the 'string' type.
 The name of the property is specified by the typeName attribute. -->
 <!ELEMENT property (#PCDATA)>
 <!ATTLIST property
          typeName   CDATA #REQUIRED
          type (boolean | integer | real | string ) "string" >
 <!ELEMENT timestamp (#PCDATA)>
 <!ELEMENT street (#PCDATA)>
 <!ELEMENT otherLocation (#PCDATA)>
 <!ELEMENT zipcode-postalcode (#PCDATA)>
 <!ELEMENT intersectionStreet (#PCDATA)>
 <!ELEMENT radius (#PCDATA)>
 <!ELEMENT innerRadius (#PCDATA)>
 <!ELEMENT outerRadius (#PCDATA)>
 <!ELEMENT startAngle (#PCDATA)>
 <!ELEMENT stopAngle (#PCDATA)>
 <!ELEMENT majorRadius (#PCDATA)>
 <!ELEMENT minorRadius (#PCDATA)>
 <!ELEMENT uncertainty (#PCDATA)>
```

All properties are string with the following exceptions:

- uncertainty - integer
- radius, innerRadius, outerRadius, majorRadius, minorRadius - real

Figure 11-10 DTD for the Geography Markup Language used by LBS

Mostly, it is sufficient to use the provided beans to parse the GML information.

11.4.1 Setting up a development environment for VisualAge for Java

In the WebSphere Everyplace Server Infocenter, and in the WebSphere EveryPlace Suite SDK, you can find instructions on how to set up WebSphere Studio and WebSphere Application Server to work with the sample. To create Servlets for Location Based Application, you can use VisualAge for Java.

VisualAge for Java provides extensive functionality across the entire development life-cycle and includes tools for Java code editing and debugging, JavaServer Page debugging, and Web Traffic Express. It also includes a repository that stores project source and compiled code, and an import/export facility that enables interaction with the file system.

One of the most important features of VisualAge for Java is Web Traffic Express. This feature provides application and Web server environments on a development machine, enabling you to test and debug the resources of a Web site locally. This environment provides much of the functionality of a full application server, including access to services such as LDAP and enterprise resources. You need the version of VisualAge for Java that corresponds best to the version of WebSphere Application Server on which the application will be deployed.

As an alternative to developing Location Based Applications using WebSphere Studio, below you will find detailed instructions on how to set up VisualAge for Java 4.0 to develop Location Based Applications. If you need additional information on using IBM VisualAge for Java, please refer to one of the following IBM Redbooks:

- *Programming with VisualAge for Java Version 3.5*, SG24-5264-01
- S*ervlet and JSP Programming with IBM WebSphere Studio and VisualAge for Java*, SG24-5755-00

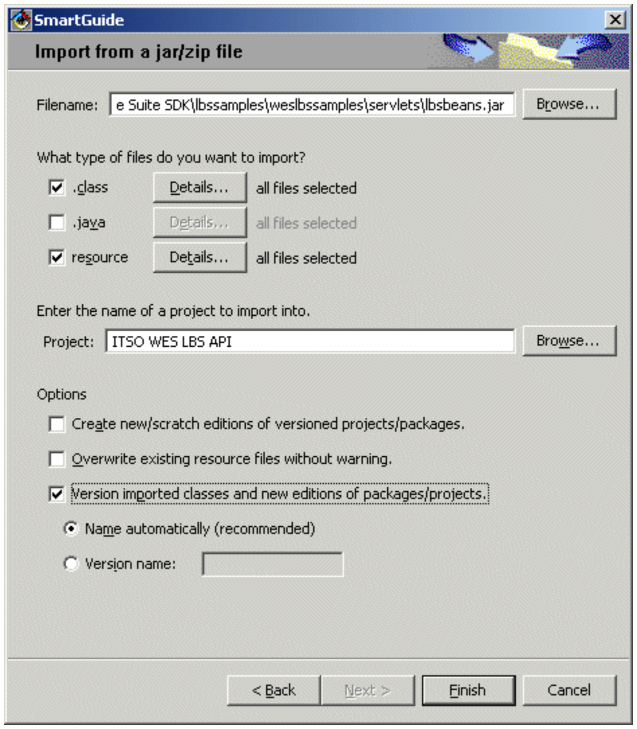
To set up VisualAge for Java to develop LBAs, perform the following steps:

1. Download and install the WebSphere Everyplace Server SDK as described in 11.4, “Developing location-based applications” on page 448.
2. Start VisualAge for Java.
3. Install the prerequisites for LBS in the workspace:

   From **File -> Quick Start**, select **Add Feature** and add the IBM XML Parser for Java.

   For creation of servlets, you will also have to install Web Traffic Express and the Servlet API the same way.
4. Create the project to contain the LBS java API, for example ITSO WES LBS API.

5. Locate the file lbsbeans.jar that contains the Java API for LBS. See 11.4, “Developing location-based applications” on page 448 for details on how to obtain this file. Import this .jar file into the project you have just created using **File -> import**. Enter the values on the screen as in Figure 11-11and click **Finish**.



Filename: e Suite SDK\lbssamples\weslbssamples\servlets\lbsbeans.jar

What type of files do you want to import?

- [✓] .class — all files selected
- [ ] .java — all files selected
- [✓] resource — all files selected

Enter the name of a project to import into.

Project: ITSO WES LBS API

Options

- [ ] Create new/scratch editions of versioned projects/packages.
- [ ] Overwrite existing resource files without warning.
- [✓] Version imported classes and new editions of packages/projects.
  - (●) Name automatically (recommended)
  - ( ) Version name:

Figure 11-11 Entering values using the .jar file

You can add the LBS beans to the palette for visual programming if you want, but we will not use that in this chapter.

6. Place the file test.xml, that contains hardcoded location information, in the classpath of the servlet engine. It belongs to the LBS beans:

   <project_resources>\com\ibm\lbs\test.xml

   Where <project_resources> is the directory of the project the beans are in. If you installed VisualAge for Java in the default directory and you used ITSO WES LBS API for the name of the project, that <project_resources > is:

   C:\Program Files\IBM\VisualAge for Java\ide\project_resources\ITSO WES LBS API

   This way, test.xml is in the classpath of the LBS API Classes in the IBM VisualAge for Java workspace.

7. Start Web Traffic Express (**Workspace -> Tools -> Web Traffic Express...**) and ensure that the required projects are in the classpath: select **Edit Class Path...**, on the next screen click **Select All** and click **OK**.

   At this point, you can start Web Traffic Express. Note that by default, the integrated Web server of Web Traffic Express listens on port 8080, so you have to add this to the URL when testing a servlet:

   `http://localhost:`**`8080`**/servlet/snoop

Now you are ready to develop servlets for Location Based Applications using IBM VisualAge for Java (see 11.5.3, “Expertise location source code” on page 461). We will use this to examine the LBS extension to the YourCo application from within VisualAge for Java.

11.5 Extending the enterprise

As an example of how the Location-Based Services can extend existing applications, below you will find a description and sample code of an extension to the YourCo application that has been described in Chapter 1.

The sources of this example can be found in Appendix B, “LBS sample code” on page 755.

11.5.1 Changing YourCo into a Location Based Application

In order to show how the Location Based Services can extend an enterprise application, we have added Expertise Location to the Yourco application.

Expertise Location

YourCo contains a directory to look up information of Yourco employees: the White Pages. A user can look up the phone number and other relevant information of an employee of Yourco. However, this is not always sufficient.

Suppose a salesperson of YourCo needs to discuss a customer situation with an expert on very short notice. He does not immediately know the person he wants to talk to because YourCo has too many employees to know them all personally. Therefore, he needs an application that shows who the experts are in a certain area. He also needs to find out which of these experts is in the office nearest to him, such that he can set up a meeting with this expert as soon as possible. This is where LBS comes into the picture.

Note: In this chapter, Expertise Location is presented as a Web application. In Chapter 5, “Text clipping” on page 145 it is shown how to enable this application for WAP devices.

11.5.2 Deploying Expertise Location to the application server

This section shows you how to deploy Expertise Location to WebSphere Application Server and integrate it in the existing sample application. Later on, we will use the DB2 tables created in this section to run Expertise Location from VisualAge for Java.

As a prerequisite, you must have WebSphere Application Server Advanced Edition version 3.5.4 installed, and you must have installed and configured the sample application as described in Chapter 3, “Enterprise sample applications” on page 77.

To install the Expertise Location extension, you must perform the following steps:

1. Back up the directory <appserver_root>\hosts\default_host\WSSamples_app

   where <appserver_root> is the install directory of WebSphere Application Server, for example C:\WebSphere\AppServer.
2. Extract the file lbsaamples.zip from the CD to the root directory of the application server, for example:

   C:\WebSphere\AppServer
3. Set up the database:
   a. You may have to run <SQLLIB>\java12\usejdbc2.bat and then use the file <SQLLIB>\java\db2java.zip. if you have not yet set up DB2 to use JDBC2. <SQLLIB> is the root directory of your DB2 installation
   b. Ensure that WebSphere Application Server and the default server are running,
   c. Enter this URL in your browser:

      `http://<hostname>/WSSamples`

      where `<hostname>` is the name of the WebSphere Application Server server.
   d. Select **Database Configuration**.
   e. Select **Start DB2 Database Configuration**.
   f. Select **Step 5**.
   g. Click the **Submit** button.

To verify this part of the setup:

- h. Open the DB2 command line processor (**Start -> Programs -> IBM DB2 -> Command Line Processor**).
- i. Enter **Connect to sample**.
- j. Enter **Select * from wsdemo.address**.

  You should see 5 records.
- k. Enter **Select office from wsdemo.employee**.

  You should see a list of office identifiers (for example 00001).

We have changed the sample database in the following way:

Columns USERID; char (20) and OFFICE; char (5) were added to table WSDEMO.EMPLOYEE.

Two new tables were added:

- WSSDEMO.ADDRESS

  Columns: OFFICE char(5) not null, ADDRESS varchar(100), CITY varchar(50), ZIPCODE varchar(20), STATE char(2), Country varchar(50).
- WSDEMO.TRIGGER

  Columns: USERID char(20), ROOM_ID varchar(3), THEDAY varchar(10), THETIME varchar(4), RESERVED_BY varchar(40).

  (WSDEMO.TRIGGER is not used for LBS, but for INS; for details, see Chapter 10, “Intelligent Notification Services (INS)” on page 359.)

  See also Appendix B, “LBS sample code” on page 755 for details.

4. Ensure that `lbsbeans.jar` is in the classpath
5. Ensure the file test.xml is in the classpath:

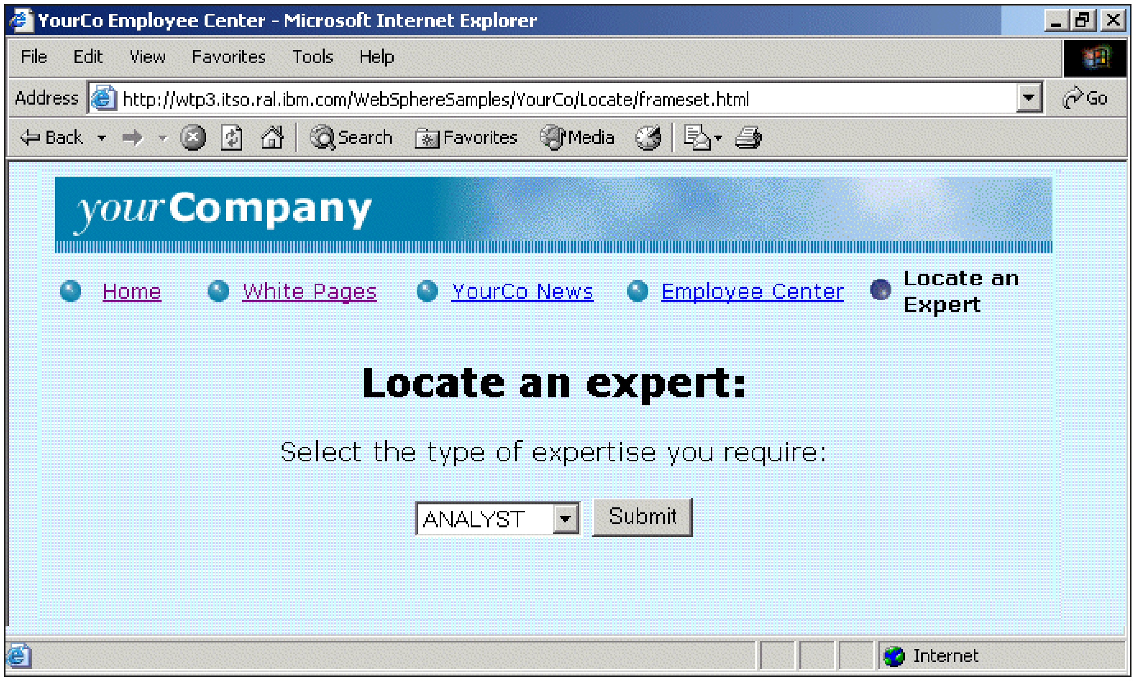
   ```
   <WASroot>\hosts\default_host\WSsamples_app\servlets\com\ibm\lbs\test.xml
   ```

   where `<WASroot>` is the directory where the ApplicationServer is installed.
6. Restart the Web application WSsamples_app using the WebSphere Administration Console.

You can now run the Yourco Sample by entering the following URL in your browser:

```
http://<hostname>/WebSphereSamples/YourCo/index.html
```

Where `<hostname>` is the host name where WebSphere Application Server is installed. You will find a new menu item in the upper right of the page named Locate an Expert. Select this menu item, and you will be presented with an entry form for Expertise Location, as shown in Figure 11-12.


YourCo Employee Center - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://wtp3.itso.ral.ibm.com/WebSphereSamples/YourCo/Locate/frameset.html
Go
Back
Search
Favorites
Media
yourCompany
Home
White Pages
YourCo News
Employee Center
Locate an Expert
Internet

Locate an expert:

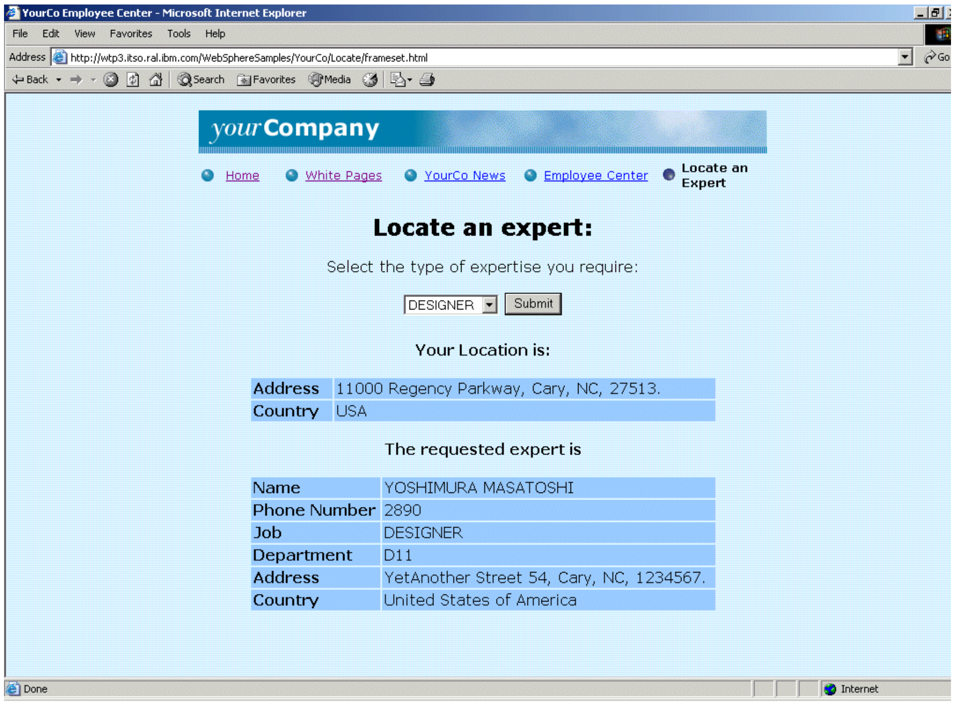
Select the type of expertise you require:

ANALYST

Submit

Figure 11-12 The entry form for Expertise Location

The database was set up such that in the nearest office to the user location (hardcoded in test.xml), there is a designer, but not an analyst. When you select **Designer**, the result is as shown in Figure 11-13, and when you select, for example, **Analyst**, which is not present in the nearest office, the result is as shown in Figure 11-14.


YourCo Employee Center - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://wtp3.itso.ral.ibm.com/WebSphereSamples/YourCo/Locate/frameset.html
Back Search Favorites Media
yourCompany
Home
White Pages
YourCo News
Employee Center
Locate an Expert
DESIGNER
Submit
Done
Internet

Locate an expert:

Select the type of expertise you require:

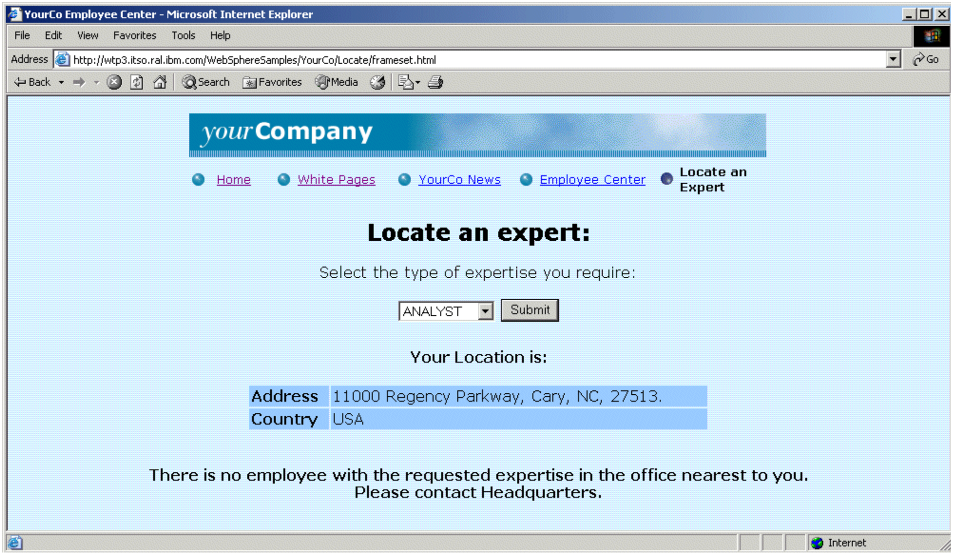
Your Location is:

Address	11000 Regency Parkway, Cary, NC, 27513.
Country	USA

The requested expert is

Name	YOSHIMURA MASATOSHI
Phone Number	2890
Job	DESIGNER
Department	D11
Address	YetAnother Street 54, Cary, NC, 1234567.
Country	United States of America

Figure 11-13 The result of the request when an expert was found in the nearest office


YourCo Employee Center - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://wtp3.itso.ral.ibm.com/WebSphereSamples/YourCo/Locate/frameset.html
Back Search Favorites Media
yourCompany
Home
White Pages
YourCo News
Employee Center
Locate an Expert
ANALYST
Submit
Internet

Locate an expert:

Select the type of expertise you require:

Your Location is:

Address	11000 Regency Parkway, Cary, NC, 27513.
Country	USA

There is no employee with the requested expertise in the office nearest to you.
Please contact Headquarters.

Figure 11-14 The result of the request when an expert was not found in the nearest office

11.5.3 Expertise location source code

The Expertise Location servlet consists of four classes:

FindExpert
This is the servlet. The source code is shown in Example 11-3.

OfficeFinder
This class determines the office that is closest to the user. The source code is shown in Example 11-4. This is done in an oversimplified way, since we do not want to elaborate on the algorithm to find the closest location, but want to focus on the Location Based Services API.

UserLocation
A bean that contains the location information of the user making the request. This bean is for use in the Java Server Page that displays the result.

ExpertInfo
A bean that contains the location information of the expert that was found. This bean is for use in the Java Server Page that displays the result.

Example 11-3 FindExpert Servlet

```
package itso.wes.lbs.samples;
import javax.servlet.*;
import java.sql.*;
/**
 * Insert the type's description here.
 * @author: Erik Rongen
 */
public class FindExpert extends javax.servlet.http.HttpServlet {

   //the OfficeFinder classs provides mapping from the user location
   //to the nearest office (in an oversimplified, non-realistic way)
   private OfficeFinder officeFinder;

   // URL for the database
   String dbUrl = "jdbc:db2:sample";

   String results = "/YourCo/Locate/Results.jsp";
   String resultsError = "/YourCo/Locate/ResultsError.jsp";
   String resultsNotFound = "/YourCo/Locate/ResultsNotFound.jsp";
/**
 * Process incoming HTTP GET requests
 *
 * @param request Object that encapsulates the request to the servlet
 * @param response Object that encapsulates the response from the servlet
 */
```

```
public void doGet(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
java.io.IOException {

   performTask(request, response);

}
/**
 * Process incoming HTTP POST requests
 *
 * @param request Object that encapsulates the request to the servlet
 * @param response Object that encapsulates the response from the servlet
 */
public void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
java.io.IOException {

   performTask(request, response);

}
/**
 * Returns the servlet info string.
 */
public String getServletInfo() {

   return super.getServletInfo();

}
/**
 * Initializes the servlet.
 */
public void init() {
    // insert code to initialize the servlet here

    officeFinder = new OfficeFinder();

    //prepare for DB2 access
    try {
        Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();

    } catch (Exception e) {
        System.out.println(e);
    }

}
/**
 * Process incoming requests for information
 *
 * @param request Object that encapsulates the request to the servlet
 * @param response Object that encapsulates the response from the servlet
 */
public void performTask(
    javax.servlet.http.HttpServletRequest request,
    javax.servlet.http.HttpServletResponse response) {

    //The id of the office that is nearest to the user location
    String officeID = null;

    //The beans with location info for display
    UserLocation userLocation = null;
    ExpertInfo expertInfo = null;

    //The requested experise from the request
    String expertise = null;

    try {

        //Analyse the request
        if (request.getParameter("expertise") == null)
            System.out.println("No expertise attribute found in the request. returning");
        else {

            //get the requested expertise from the request, and convert to upppercase,
            //since all JOBS in the database are in UPPERCASE
            expertise = (String) request.getParameter("expertise").toUpperCase();

            //Create the locationServices object that reads the Location Info from the headers
of the request
            com.ibm.lbs.LocationServices locServices =
                new com.ibm.lbs.LocationServices(request);

            //locServices now contains all the location information of the user that we need
            //List the information to the console
            printDiagnostics(locServices);

            //we can populate the userLocation bean
            userLocation = new UserLocation();
            userLocation.setAddress(locServices.getStreet());
            userLocation.setCity(locServices.getCity());
            userLocation.setCountry(locServices.getCountry());
            userLocation.setState(locServices.getStateProvince());
            userLocation.setZipCode(locServices.getPostalCode());

            //Now find the id of the office closest to the user
            //The OfficeFinder encapsulates this algorithm.
            //In this example we use an oversimplified algorithm for finding
            //the nearest office
            officeID = officeFinder.find(locServices);
```

```
//now that we have the office id, we can search for
//the requested expertise in that location in the database

String url = "jdbc:db2:sample";
Connection con = DriverManager.getConnection(url, "wsdemo", "wsdemo1");

// retrieve data from the database
System.out.println("Retrieve some data from the database...");
Statement stmt = con.createStatement();
String queryString =
    "SELECT * FROM WSDEMO.EMPLOYEE X , WSDEMO.ADDRESS Y"
        + " WHERE X.OFFICE = Y.OFFICE"
        + " AND X.OFFICE = '"
        + officeID
        + "'"
        + " AND X.JOB = '"
        + expertise
        + "'";

ResultSet rs = stmt.executeQuery(queryString);

System.out.println("Received results:");
//rs.next is true when rows are found.
//return the first found employee who meets the criteria
if (rs.next()) {
    //Info of the employee
    expertInfo = new ExpertInfo();
    expertInfo.setFirstName(rs.getString("FIRSTNME"));
    expertInfo.setLastName(rs.getString("LASTNAME"));
    expertInfo.setMiddleInitial(rs.getString("MIDINIT"));
    expertInfo.setPhoneNumber(rs.getString("PHONENO"));
    expertInfo.setDepartment(rs.getString("WORKDEPT"));
    expertInfo.setEmployeeNumber(rs.getString("EMPNO"));
    expertInfo.setJob(rs.getString("JOB"));

    //info of the location
    expertInfo.setOfficeID(rs.getString("OFFICE"));
    expertInfo.setAddress(rs.getString("ADDRESS"));
    expertInfo.setCity(rs.getString("CITY"));
    expertInfo.setCountry(rs.getString("COUNTRY"));
    expertInfo.setState(rs.getString("STATE"));
    expertInfo.setZipCode(rs.getString("ZIPCODE"));
} else {
    System.out.println("No expert found in the office nearest to the user");
}
//we¥re done with DB2
rs.close();
stmt.close();
```

```
            //Now redirect to the output
            if (userLocation != null && expertInfo != null) {
                //Both beans have been populated. Show result
                request.setAttribute("userLocationBean", userLocation);
                request.setAttribute("expertInfoBean", expertInfo);
                RequestDispatcher rd = getServletContext().getRequestDispatcher(results);
                rd.forward(request, response);
            } else
                if (userLocation != null && expertInfo == null) {
                    //Both beans have been populated. Show result
                    request.setAttribute("userLocationBean", userLocation);
                    RequestDispatcher rd =
                        getServletContext().getRequestDispatcher(resultsNotFound);
                    rd.forward(request, response);
                } else {
                    RequestDispatcher rd =
getServletContext().getRequestDispatcher(resultsError);
                    rd.forward(request, response);
                }

        }

    } catch (Throwable theException) {
        theException.printStackTrace();
    }
}
/**
 *
 * @param locServices com.ibm.lbs.LocationServices
 */
private void printDiagnostics(com.ibm.lbs.LocationServices locServices) {

   System.out.println("Location information for the user:");
    System.out.println("City                :  " + locServices.getCity() + "\n");
    System.out.println(
        "Cooirdinates        :  " + locServices.getCoordinates() + "\n");
    System.out.println("Country             :  " + locServices.getCountry() + "\n");
    System.out.println(
        "CountryDistrict     :  " + locServices.getCountyDistrict() + "\n");
    System.out.println(
        "Description         :  " + locServices.getDescription() + "\n");
    System.out.println(
        "Latitude            :  " + locServices.getLatitude() + "\n");
    System.out.println(
        "Longitude           :  " + locServices.getLongitude() + "\n");
    System.out.println("Name                :  " + locServices.getName() + "\n");
    System.out.println(
        "PostalCode          :  " + locServices.getPostalCode() + "\n");
    System.out.println("SRSName             :  " + locServices.getSRSName() + "\n");
    System.out.println(
        "StateProvince       :  " + locServices.getStateProvince() + "\n");
    System.out.println("Street              :  " + locServices.getStreet() + "\n");
    System.out.println(
        "StreetIntersection  :  " + locServices.getStreetIntersection() + "\n");
    System.out.println(
        "TimeStamp           :  " + locServices.getTimestamp() + "\n");
    System.out.println(
        "Uncertainty         :  " + locServices.getUncertainty() + "\n");

}
}
```

When the doGet or doPost methods of the servlet are called, the request is passed on to the Perform Task method. This way, although the servlet will respond to a post from an HTML page in the production environment, you can test it without that page using a query string:

```
<the application URL>?expertise=DESIGNER
```

where `<the application URL>` is the URL of the servlet.

The method where the request is processed is the Perform Task method. In this method, first we get the requested expertise from the request:

```
expertise = (String) request.getParameter("expertise").toUpperCase();
```

Next, the LocationServices object is created; it reads the location information from the headers of the request:

```
com.ibm.lbs.LocationServices locServices = new
com.ibm.lbs.LocationServices(request);
```

Using the constructor that has the request as parameter, the object is immediately initialized with the information from the hardcoded file. The OfficeFinder class is then asked for the office nearest to the user who initiated the request:

```
officeID = officeFinder.find(locServices);
```

The code for the office finder is shown in Example 11-4. As said, the algorithm for finding the closest office is very simple. In fact, it matches the city of the expert's office location to the city in the GML location information in the header. The database is queried to find out if there is an expert in the office in that city, and if so, to get the information. The SQL query used is emphasized in the above sample text.

Finally, the beans are populated, and the request is forwarded to the proper Java Server Page, depending on whether an expert was found or not.

Example 11-4 SOfficeFinder

```
package itso.wes.lbs.samples;

/**
 * The officeFinder will locate the office nearest to the user's location
 * This was done in a very simple way. A hashtable closestOfficeList
 * contains the link between the user location and the nearest office. Note
 * that in real life things are much more complicated.
 * @author: Erik Rongen
 */
public class OfficeFinder {
   private java.util.Hashtable closestOfficeList;
/**
 * OfficeFinder constructor comment.
 */
public OfficeFinder() {
   super();
   closestOfficeList = new java.util.Hashtable();
   closestOfficeList.put("Cary","00005");
   closestOfficeList.put("San Diego","00001");
   closestOfficeList.put("Boulder","00002");
   closestOfficeList.put("Delft","00004");
   closestOfficeList.put("Raleigh","00003");

}
/**
 *
 * @return java.lang.String
 * @param locationService com.ibm.lbs.LocationServices
 */
public String find(com.ibm.lbs.LocationServices locationService) {
   String userCity = locationService.getCity();
   String closestOfficeID =  (String) closestOfficeList.get(userCity);
   return closestOfficeID;
}
}
```

The HTML files and Java Server Pages can be found in:

```
<WASroot>\hosts\default_host\WSsamples_app\web\YourCo\index.html
```

and

```
<WASroot>\hosts\default_host\WSsamples_app\web\YourCo\Locate
```

where you will find:

- frameset.html, the HTML containing the frames of the page.
- LocateInput.html, the HTML page for the input frame (top).
- Results.jsp, the JSP that is shown when an expert was found.
- ResultsNotFound.jsp, the JSP that is shown when no expert was found.
- ResultsError.jsp, the JSP that is shown when an error occurred.
- blank.html, the blank for the lower frame when the page is initially opened.

These are basic HTML JSP files, and will not be discussed in detail. There is no LBS-specific content in these files.

You can run this example as follows:

1. To set up the database, the sample needs DB2. It is easiest to run the sample against the database that has been configured for the sample, as explained in 11.5.2, "Deploying Expertise Location to the application server" on page 457. If this database is not local, change the following:
   a. Use another DB2 driver. For example, change:
      ```
      Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();
      ```
      to
      ```
      Class.forName("COM.ibm.db2.jdbc.net.DB2Driver").newInstance();
      ```
   b. Change the location of the database:
      ```
      String url = "jdbc:db2:sample";
      ```
      to
      ```
      String url = "jdbc:db2:<hostname>/sample";
      ```
   c. Ensure that the JDBC driver is running on the server. On Windows NT or Windows 2000, this can be started from the Services panel.
2. Start VisualAge for Java and import the sample code into a new project (for example ITSO WES Samples); the code can be found in yourcolbssample.ja file on the CD.
3. Start Web Traffic Express.
4. Ensure that the project of the sample is in the classpath of the Servlet Engine (see 11.4.1, "Setting up a development environment for VisualAge for Java" on page 454 for details).
5. Start the Servlet Engine.

6. Run the sample by entering the following URL in your browser:

```
http://localhost:8080/servlet/itso.wes.lbs.samples.FindExpert?expertise=des
igner
```

11.6 Integrated testing

This section discusses how to test your LBS applications in the WebSphere Everyplace Server environment. This WebSphere Everyplace Server environment involves a minimal setup to test the LBS aspects of your application. Some products needed in the production environment will not be used. Instead, tools will be provided that can help you test without these tools.

Tools that can be used to emulate WebSphere Everyplace Server functions and other infrastructure are:

- Simples: tools to emulate the SignalSoft server
- Muffin: a configurable proxy for Windows, with which you can view and manipulate an HTTP request, including the headers.
- The WAP ToolKit.

11.6.1 Planning a Location Based Services installation

Suggested steps for LBS planning are included in this section.

1. First of all, you must decide how to distribute the different functions of the available hosts. If you want to include WebSEAL-Lite for security, you will need at least two hosts, because WSL and LBS, both plug-ins to WTE, cannot be installed in one instance of WTE.
2. LBS requires that the PD client be installed/configured correctly before it is installed on that system. See below for details.
3. LBS requires, among other things, that the ClientID in the request header populate the request with location information. For LBS to work properly, it requires the following WebSphere Everyplace Server components installed:

   a. WebSphere Everyplace Server Wireless Gateway (configured for client ID support)

   b. WebSphere Everyplace Server WebSEAL-Lite

   c. User Preferences-SelfCare (optional)

   Below, we shall show you how you can do without these functions by emulating the services that create the request headers, and by creating users in another way.

4. As to the host for the Location Server (for example, SignalSoft), you will need to know the host name and port for the location server. If you do not have a location server available in your test environment, LBS provides an emulator, called *simples*. Simples can be installed on one of the servers you use for WebSphere Everyplace Server. Simples listens on port 8567.
5. Location-Based Service is installed by WebSphere Everyplace Server Setup Manager. To install LBS:
   a. You will need location (that is, host, port) and WebSphere Everyplace Server LDAP Server information.
   b. You will need location (that is, host, port) and LDAP Server information for Policy Directory Server.
   c. You will need location (that is, host, port) for the SignalSoft server.
   d. If you are planning to import some Location Based Applications (LBAs), you will need a list of these URLs in an input file.
6. Choose **Application Server** to install LBS Application Assist functions and samples.

The installation of the Location Based Services is part of the WebSphere Everyplace Server installation, and is done using the WebSphere Everyplace Server Setup Manager. As a prerequesite for LBS, Policy Director must be installed. The installation of Policy Director cannot be performed using the WebSphere Everyplace Server Setup Manager, and must be done separately.

For details on the installation and Configuration of LBS, see the WebSphere Everyplace Server Information Center.

11.6.2 Policy Director

Policy Director is the component in WebSphere Everyplace Server that is responsible for authorizations, and is an prerequisite for Location Based Services. When a user is authenticated using WebSEAL-Lite, Policy Director is used to define which applications are allowed to obtain the location information of a user. Policy Director is a product that runs on top of IBM's DCE, the Distributed Computing Environment, a secure infrastructure for distributed computing systems.

In the PD section of the WebSphere Everyplace Server information Center, you can find details on the installation of Policy Director. Administration Guides for installation on AIX and Solaris are located in the subirectory tpd, and have names pd37_base_install_aix.pdf and pd37_base_install_solaris.pdf, respectively. Refer to appendix A of the respective documents for a quick installation procedure of IBM DCE (Distributed Computing Environment), the underlying infrastructure component, and Policy Director (PD).

11.6.3 PD client installation and configuration on AIX for WebSphere Everyplace Server

If you are going to run LBS on a different machine from the PD Server, you will have to install the LDAP Client and PD Client on the LBS Host. The steps below will show you how.

1. Install IBM LDAP client:
   - Mount WebSphere Everyplace Server CD #4.
   - Change to the install directory

     ```
     cd /cdrom/swd/aix/ldap32_us/
     ```

     (assuming that the mount point is /cdrom)
   - Perform `smit install` and choose the current directory; select **ldap client** and **gskit** to install.
2. Install the PD Client:
   - mount the PD CD # 14.

     ```
     cd /cdrom/Policy_Director
     ```
   - Use `smit` to install PDRTE and PDAuthADK (PD authorization ADK) from the current directory.
3. Get the PD server certificate:
   a. Enter the following command:

   ```
   cd /opt/PolicyDirector/keytabs
   ```

   b. This brings you to the directory where you want to install the PD Server Certificate. To obtain it from the server, perform the following sequence of commands:

   ```
   ftp PDServer (Enter the user name and password.)
   cd /opt/PolicyDirector/ivmgrd/keytabs
   get pdcacert.b64
   bye
   ```

   Note: Do not enter `ftp`; this is the prompt you will see.

4. Run `smit`, select **Communication Application->Policy Director->PD Config**
   a. Enter `n` when asked if the PD Server is installed on the local machine.
   b. Select **2** for using LDAP.
   c. Type in the host name where the PD LDAP server is installed.
   d. Accept the default port for the LDAP Server.
   e. Type in the host name where the PD Server is installed.
   f. Accept the default port for the PD Server.
   g. Type in the full file name of the PD Server certificate, in our case /opt/PolicyDirector/keytabs/pdcacert.b64
   h. Wait a few seconds, and the PD client should be configured successfully.

11.6.4 Installation of Location-Based Services

Now you are ready to install Location Based Services. Installation of Location Based Services is performed using the WebSphere Everyplace Server Setup Manager. For details on the installation of Location Based Services, please refer to the WebSphere Everyplace Server Infocenter

11.6.5 Using pd_populate.ksh to configure LBAs

The Policy Director (PD) client must be configured using AIX to work with the Policy Director Server. You can perform the following to verify that the Policy Director Client is communicating with the Policy Director Server:

1. Open a Terminal Window.
2. At the prompt, enter the following and click **Enter**.

   ```
   /usr/IBMEPS/LBS/bin/pd_populate.ksh pdTest <password>
   ```

   where `<password>` is the password for the Policy Director admin ID sec_master.
3. If the connection is successful, the system returns a value `return with rc=0`. If the connection is not successful, the system returns a non-zero code. To fix the connection, contact your Policy Director System Administrator.

Adding an application to Policy Director

In this section, we show how to add an application using script pd_populate.ksh. For an application to be available for administrators to enable the application as a Location Based Application, it must be added to Policy Director using the script pd_populate.ksh. When you add an application to Policy Director, it is automatically enabled as a Location Based Application. If you do not want the application to be a Location Based Application, you must disable it. You can use script pd_populate to add and enable applications, but you must use the Policy Director Console to view registered applications. The user must use User Preferences in Tivoli Internet Services Manager to enable the application to gain access to their information, or the administrator must enable the user through the Policy Director Console:

1. Verify the Policy Director Client (refer to Verifying the Policy Director Client).
2. Open a Terminal Window.
3. Enter the URL for the application you want to add. Entering the characters "http://" before the URL is optional. Script pd_populate.ksh only accepts the special characters "/", ":", "~", and ".". If you use other special characters in the URL name, Policy Director returns errors. At the prompt, enter the following and press Enter:

   ```
   /usr/IBMEPS/LBS/bin/pd_populate.ksh initAppl <password> <ldap suffixdn>
   <application URL>
   ```

   where:

   – <password> is the password for the Policy Director administrator ID sec_master.
   – <ldapsuffixdn> is the LDAP distinguished name suffix (for example, o=ibm,c=us).
   – <application URL> is the URL for the application you are adding to Policy Director. For example, www.yourco.com/index.html.
   – The URL is the actual location base address in Policy Director without the preceding LBS_. For example, if the URL in Policy Director is LBS_www.ibm.com, the URL to add would be www.ibm.com.
4. To add another application, repeat step 3.

Deleting an application from Policy Director

In this section we show how to delete an application using script pd_populate.ksh. When an application is deleted from Policy Director, it is no longer available for administrators to enable as a location based application. Applications must be deleted from Policy Director by using the script pd_populate.ksh.

You can use this script to delete applications, but you must use the Policy Director Console to view registered applications. When an application is deleted from Policy Director, it is automatically disabled as a location based application for all users that have enabled this application to see their location information. If you want to disable an application from being location based, you can disable the application instead of deleting the application from Policy Director.

If a user wants to simply disable a specific application from receiving their location information, they should use Tivoli Personalized Services Manager to change their user preferences or have the administrator disable them in Policy Director.

1. Verify the Policy Director Client.
2. Open a Terminal window.
3. Enter the URL for the application you want to delete. At the prompt, enter the following and click **Enter**.

   ```
   /usr/IBMEPS/LBS/bin/pd_populate.ksh delAppl <password> <application URL>
   ```

   where

   – `<password>` is the password for the Policy Director admin ID sec_master.

   – `<application URL>` is the URL for the application you are deleting from Policy Director, for example, `www.yourco.com/index.html`.

   – The URL is the actual location base address in Policy Director without the preceding LBS_. For example, if the URL in Policy Director is `LBS_www.ibm.com`, the URL to delete would be `www.ibm.com`.
4. To delete another application, repeat step 3.

11.6.6 The Policy Director Management Console

If you want to view or manipulate the PD entries directly, without the use of TPSM, you can use the PD Management Console. Installation of the PD Management Console consists of several steps. Next, we will show you how to install the PD Console on Windows.

NetSeat Client installation and configuration

Locate the CD containing the PD Console Software (WebSphere Everyplace Server CD # 15), and locate the win32\client folder. Run **`setup.exe`**. This starts the NetSeat client installation. Select the required language, and the option **DCE Runtime Only**. Next, select a **Typical Installation**, and the files are copied to the PC.

The NetsSeat Configuration dialog is automatically started, or you can start the NetSEAT Configuration manually(**Start -> Programs -> Policy Director -> NetSEAT -> NetSEAT Configuration**). This will show the window displayed below.

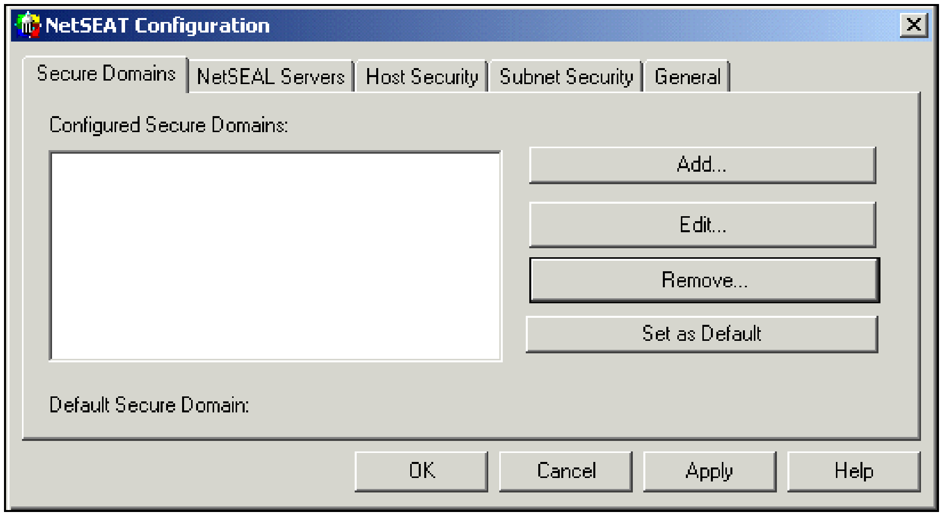
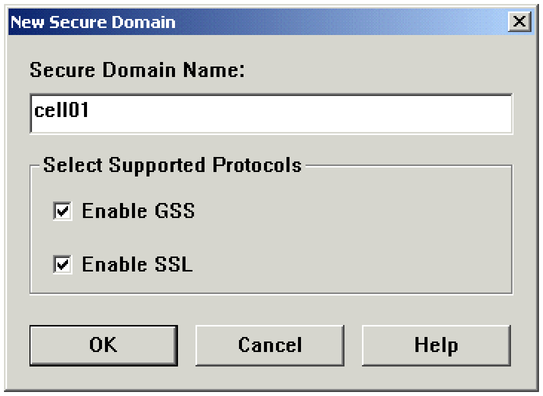


Figure 11-15 NetSEAT configuration

Select **Add...** to add a new secure domain. The result is shown in Figure 11-16.



Secure Domain Name: cell01

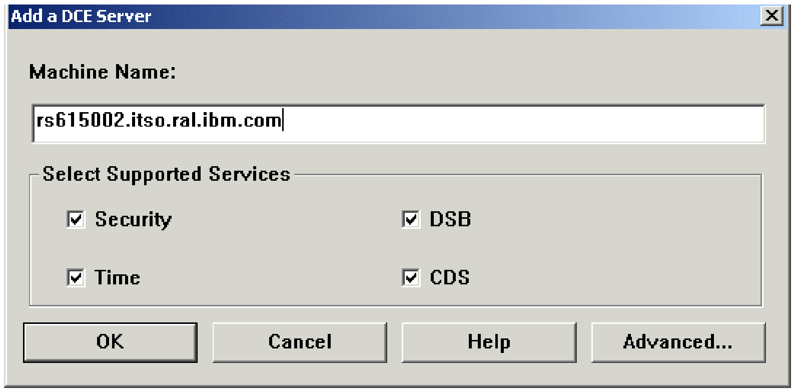
Select Supported Protocols

[✓] Enable GSS

[✓] Enable SSL

Figure 11-16 NetSEAT cell configuration

After entering the name of the cell, click **OK**; you are then presented with the panel in which you can add the servers that are active in this domain. If you select the **Add** button here, you are presented with a panel where you can enter the properties of the DCE Server (the server running the PD Server). This is shown below.

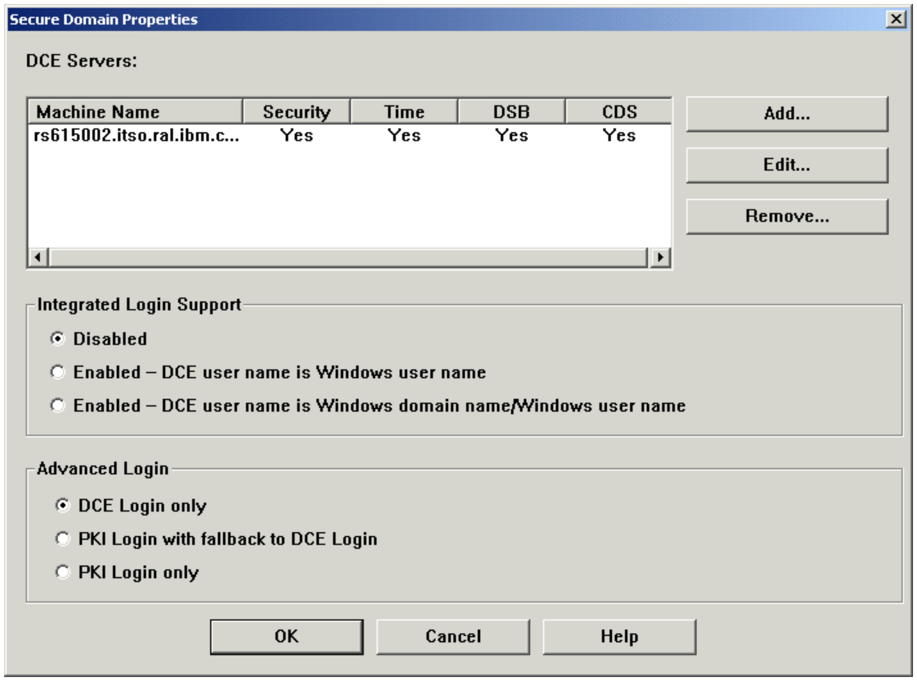


Machine Name:
rs615002.itso.ral.ibm.com

Select Supported Services
☑ Security ☑ DSB
☑ Time ☑ CDS

Figure 11-17 Supported services

Enter the host name of the server and the services that are active on the server, and click **OK**. The result is shown in Figure 11-8.


Secure Domain Properties

DCE Servers:

Machine Name	Security	Time	DSB	CDS
rs615002.itso.ral.ibm.c...	Yes	Yes	Yes	Yes

Integrated Login Support

[✓ Disabled]
Enabled – DCE user name is Windows user name
Enabled – DCE user name is Windows domain name/Windows user name

Advanced Login

[✓ DCE Login only]
PKI Login with fallback to DCE Login
PKI Login only

Figure 11-18   Secure domain properties

Note: Restart the computer when requested.

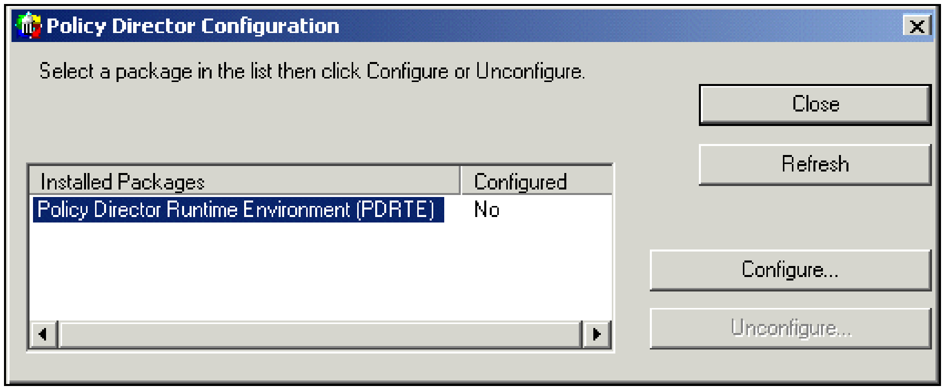
Policy Director Management Console installation

The software you need can be found on CD #15 in the directory Win32\Console. Run `setup.exe` to start the installation.

Select the preferred language, and finish the dialog. After installation is complete, the PC must be restarted.

Next, PD must be configured to work with the PD Server of WebSphere Everyplace Server. To this end, you must perform several configuration steps. First of all, you must configure the PD Runtime Environment for this client. You will need to download the server security certificate. This can be done using FTP. The file you need to download is /opt/PolicyDirector/ivmgrd/keytabs/pdcacert.b64

Next, you must configure the PD runtime (PDRTE) by clicking **Start -> Programs -> Policy Director -> Configuration**. The window shown in Figure 11-19 on page 478 appears. Click the **Configure...** button to continue.

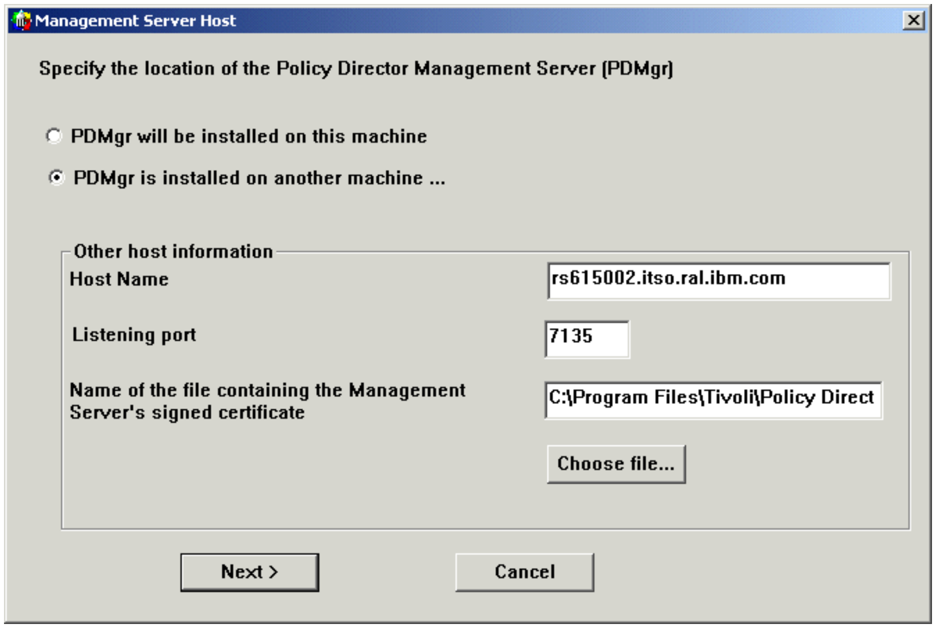


Select a package in the list then click Configure or Unconfigure.

Installed Packages	Configured
Policy Director Runtime Environment (PDRTE)	No

Figure 11-19 Configuration of Policy Director Client

In the following window, you have to provide the information about the PD Server. In this case, the PD Server is installed on another machine, and you have to point to the key file that you have downloaded.



Specify the location of the Policy Director Management Server (PDMgr)

PDMgr will be installed on this machine [✓ PDMgr is installed on another machine ...]

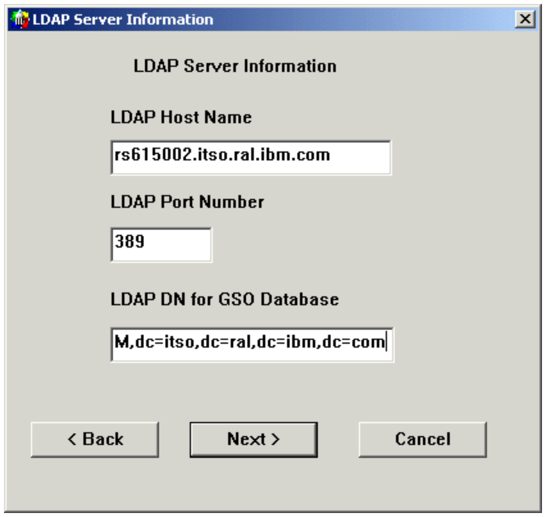
Other host information

Host Name	rs615002.itso.ral.ibm.com
Listening port	7135
Name of the file containing the Management Server's signed certificate	C:\Program Files\Tivoli\Policy Direct

Figure 11-20 Management server host

In the next window, select the **LDAP Directory**, and then, in the panel after that, enter the LDAP information, as shown below. The field LDAP DN for GSO

Database defines the DN of the top level of the LDAP tree under which all PD-related entries can be found.



LDAP Server Information

LDAP Host Name: rs615002.itso.ral.ibm.com

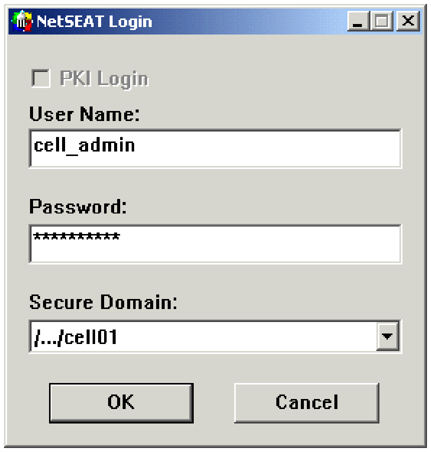
LDAP Port Number: 389

LDAP DN for GSO Database: M,dc=itso,dc=ral,dc=ibm,dc=com

Figure 11-21 LDAP Server information

Next, specify that you do not want to use SSL communication and then complete the configuration. The PDRTE package is now configured.

Once PD and NetSeat are configured, you can log in to the system (**Start -> Programs -> Policy Director -> NetSEAT -> NetSEAT Login**). Enter the user ID and password for the cell, and select the appropriate secure domain.



PKI Login

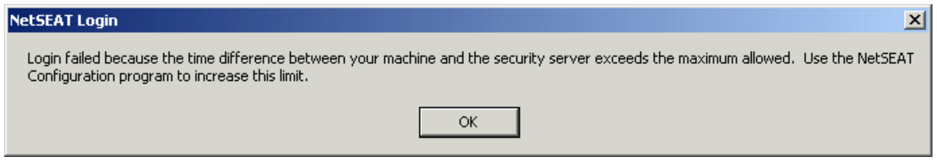
User Name: cell_admin

Password: **********

Secure Domain: /.../cell01

Figure 11-22 NetSEAT Login window

You may find problems when logging in if the system times of the two machines differ too much. In this case, you will see the following error message.

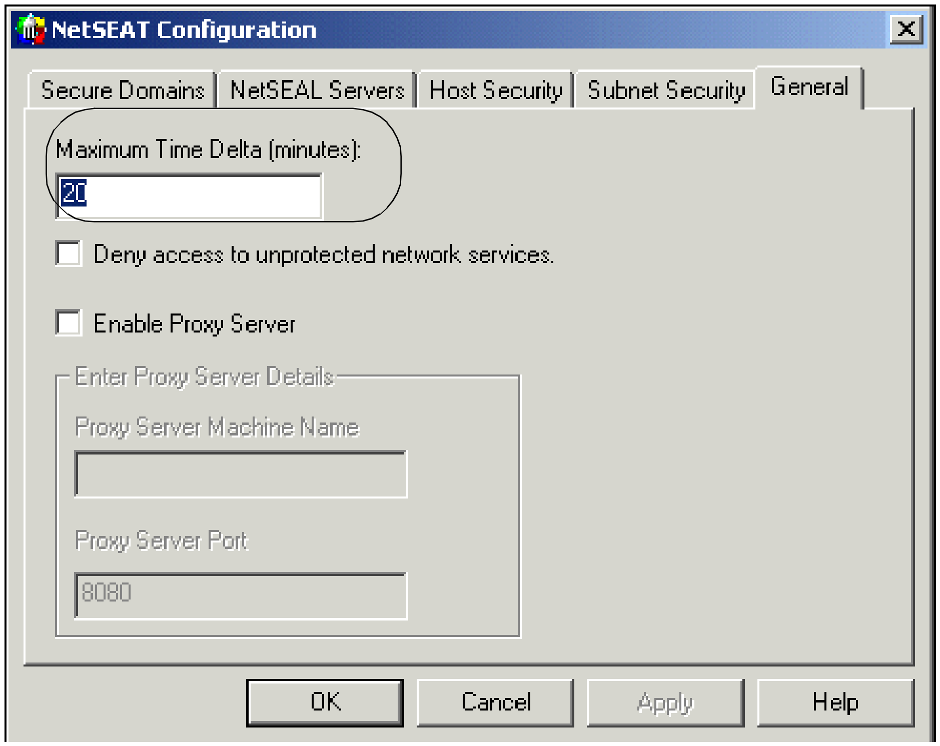


NetSEAT Login

Login failed because the time difference between your machine and the security server exceeds the maximum allowed. Use the NetSEAT Configuration program to increase this limit.

Figure 11-23 Logon error

If this problem occurs, you can change the maximum allowed time difference in the NetSeat Configuration window, as shown below. The maximum allowed difference is 59 minutes. If the difference is more than that, you must change the system time settings of one of the two systems.



NetSEAT Configuration

Maximum Time Delta (minutes): 20

☐ Deny access to unprotected network services.

☐ Enable Proxy Server

Enter Proxy Server Details

Proxy Server Machine Name:

Proxy Server Port: 8080

Figure 11-24 NetSEAT configuration - Maximum Time Delta (minutes)

Running the Policy Director Management Console

Now that the PD Management Console is configured correctly, you can use it to view the resources that are maintained by Policy Director. After you have logged in through NetSEAT (see above), you must log on to PD using sec_master, and then you can view the users, groups and resources, as shown in Figure 11-25 and Figure 11-26.

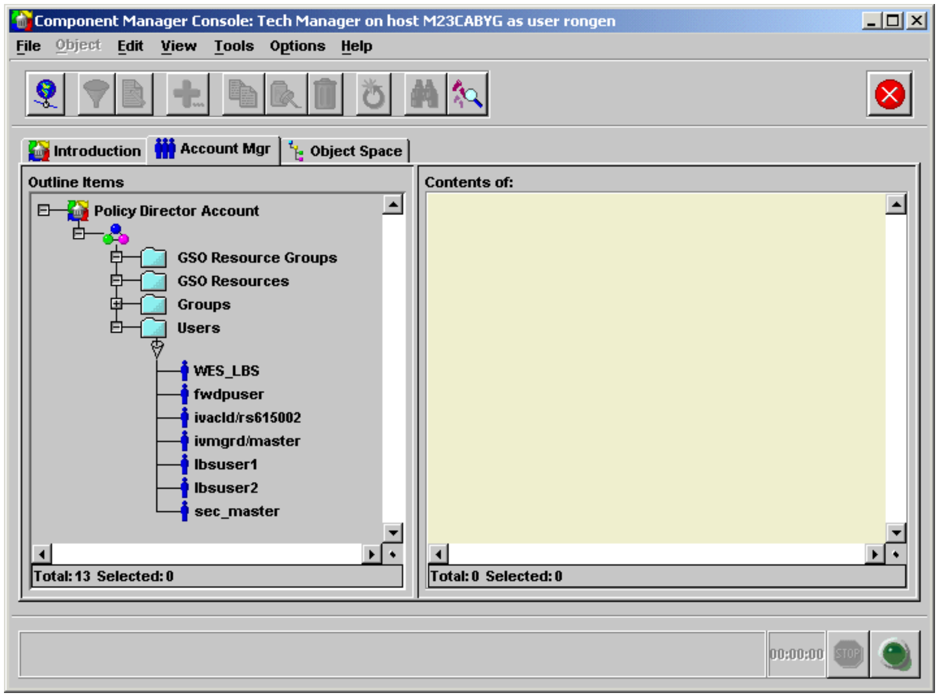


Figure 11-25 Users in the PD console.

Querying and viewing Policy Director

You can review the current set of WebSphere Everyplace Server applications protected by Policy Director by reviewing the WebSphere Everyplace Server Location Based Services Object space tree structure in the Policy Director Console.

Click the **Object Space** tab. The Login to a Secure Domain dialog box is displayed.

1. Enter the Login_Name and Password if requested. You must enter a login name with enough authority to view the applications, such as `sec_master`. Click **OK**. The Policy Director Object Space is displayed.

2. To expand the root of the tree, click the **+** sign for the root of the tree. The list of objects underneath is expanded.
3. To expand WESLBS, click the **+** sign for WESLBS. A list of applications registered for Location Based Services is displayed. The applications LBS_ALL and LBS_NONE are installed when WebSphere Everyplace Server is installed and are reserved for use by IBM. If you entered applications with a slash in the URL, the program creates each piece of text separated by a slash in a separate container.

Important: User WES_LBS, groups LBS_PERMIT_ALL and LBS_DENY_ALL, and applications LBS_ALL and LBS_NONE are reserved for use by IBM. Do not update or delete these users, groups or applications.

Enabling an application in Policy Director Console

For an application to be available for users to allow their location-based information to be used by an application, it must be enabled as a Location Based Application. When you add an application to Policy Director by running the script pd_populate.ksh, it is automatically enabled. You only need to enable an application as location based if you have previously disabled it as a Location Based Application.

1. From the Windows Start Menu, select **Policy Director** and then **Management Console**. The Policy Director Management Console is displayed.
2. Click the **Account Manager** tab. The Log in to a Secure Domain dialog box is displayed.
3. Enter the Login_Name and Password. You must enter a login name with enough authority to view the applications, such as `sec_master`. Click **OK**. The Policy Director Object Space is displayed.
4. To expand the Groups directory, click the **+** sign for Groups. The Load Users dialog box is displayed.
5. Click **Load**. The groups are loaded. Click **Close**. All available applications are displayed. All applications are displayed with the prefix LBS_.
6. Highlight the application you want to enable.
7. Click the right mouse button. From the context menu, select **Properties**. The Group Properties dialog box is displayed.
8. In the Name field, click the **...** button. The Browse Users dialog box is displayed.
9. In the User Name Pattern field, enter a search to limit the users. For example, LBS*.

10. In the Limit To field, enter the maximum number of users you want displayed.
11. Click **Load**. The users matching the search entered in the User Name Pattern field are displayed in the Search Results field.
12. In the Search Results field, select the user **WES_LBS**.
13. Click **OK**. The Group Properties dialog is displayed. WES_LBS is displayed in the Members List. Click **OK**.

Disabling an application in Policy Director Console

If you want an application to not be available for users to allow their location based information to be used by application, you must disable it as a Location Based Application. When an application is disabled as location based, all users enabled for the application are automatically disabled and their location information is no longer transmitted to the application.

1. From the Windows Start Menu, select **Policy Director** and then **Management Console**. The Policy Director Management Console is displayed.
2. Click the **Account Manager** tab. The Log in to a Secure Domain page is displayed.
3. Enter the Login_Name and Password. You must enter a login name with enough authority to view the applications, such as `sec_master`. Click **OK**. The Policy Director Object Space is displayed.
4. To expand the Groups tree, click the **+** sign for Groups.
5. Highlight the application you want to disable. All applications are displayed with the prefix LBS_.
6. Click the right mouse button. From the context menu, select **Properties**. The User Properties dialog is displayed.
7. In the Members List, highlight the user **WES_LBS**. If WES_LBS is not displayed in the Members List, the application is not enabled for Location Based Services.
8. Click **Remove**. WES_LBS is removed from the Members List.
9. Click **OK** . The application is disabled from Policy Director.

Enabling a user in Policy Director Console

If a user wants their location information to be used by an application, you need to enable the user for the application. Note that for a user to be enabled for an application, the application must already have been enabled as a Location Based Application.

1. From the Windows Start Menu, select **Policy Director** and then **Management Console**. The Policy Director Management Console is displayed.
2. Click the **Account Manager** tab. The Login to a Secure Domain dialog box is displayed.
3. Enter the Login_Name and Password. You must enter a login name with enough authority to view the applications, such as `sec_master`. Click OK. The Policy Director Object Space is displayed.
4. To expand the Groups directory, click the **+** sign for Groups. The Load Users dialog box is displayed.
5. Click **Load**. The groups are loaded. Click **Close**. All available applications are displayed. All applications are displayed with the prefix LBS_.
6. Highlight the application for which you want to enable a user.
7. Click the right mouse button. From the context menu, select **Properties**. The Group Properties dialog box is displayed.
8. In the Name field, click the **...** button. The Browse Users dialog box is displayed.
9. In the User Name Pattern field, enter a search to limit the users. For example, `LBS*`.
10. In the Limit To field, enter the maximum number of users you want displayed.
11. Click **Load**. The users matching the search entered in the User Name Pattern field are displayed in the Search Results field.
12. In the Search Results field, select the user you want to enable.
13. Click **OK**. The Group Properties dialog is displayed. The user you enabled is displayed in the Members List. Click **OK**.

Disabling a user in Policy Director Console

If a user no longer wants their location information to be used by an application, you need to disable the user for the application.

1. From the Windows Start Menu, select **Policy Director** and then **Management Console**. The Policy Director Management Console is displayed.
2. Click the **Account Manager** tab. The Login to a Secure Domain page is displayed.
3. Enter the Login_Name and Password. You must enter a login name with enough authority to view the applications, such as `sec_master`. Click **OK**. The Policy Director Object Space is displayed.
4. To expand the Groups tree, click the **+** sign for Groups.

5. Highlight the application for which you want to disable a user. All applications are displayed with the prefix LBS_.
6. Click the right mouse button. From the context menu, select **Properties**. The User Properties dialog is displayed.
7. In the Members List, highlight the user you want to disable. If the user is not displayed in the Members List, the application is not enabled for Location Based Services. Click **Cancel**.
8. Click **Remove**. The user is removed from the Members List.
9. Click **OK**. The application is disabled from Policy Director.

For example, Figure 11-26 illustrates the Groups directory.

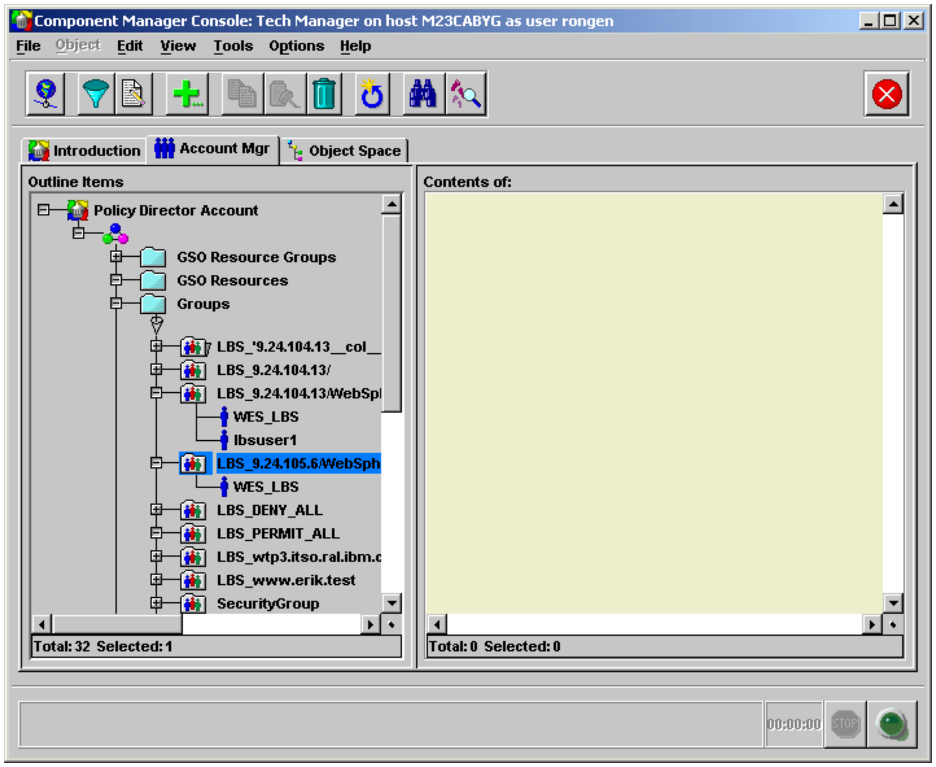
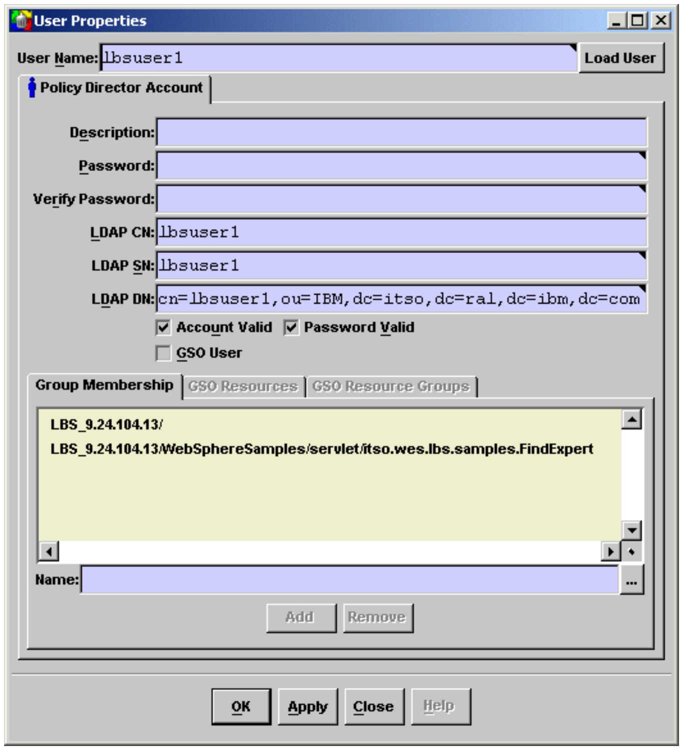


Figure 11-26 Groups in the PD console

For example, Figure 11-27 on page 486 illustrates a user's properties.



User Name: lbsuser1

LDAP CN: lbsuser1

LDAP SN: lbsuser1

LDAP DN: cn=lbsuser1,ou=IBM,dc=itso,dc=ral,dc=ibm,dc=com

[✓] Account Valid [✓] Password Valid

[ ] GSO User

Group Membership

LBS_9.24.104.13/

LBS_9.24.104.13/WebSphereSamples/servlet/itso.wes.lbs.samples.FindExpert

Figure 11-27 User information

LBS interaction with Location Base Applications (LBA)

Three ways that the Location Base Server interacts with Location Base Applications are described below.

Wild-card scheme

If you define an application to be an LBA `www.ibm.com/services` and then user1 allows this LBA access to their user location information, all URL requests to this defined LBA --and also anything underneath this URL tree (`www.ibm.com/services/*`)-- are permitted to get the user1 user location information.

Specific scheme

Another option is to specifically define an LBA, for example `www.ibm.com/services/index.html`. In this case, if user2 allows access to this LBA, only the particular URL request is allowed user location information. If user2 requests `www.ibm.com/services/index1.html`, the URL is not allowed user location information as it is not defined as an LBA in Policy Director. In this scenario, `www.ibm.com/services/index1.html` only gets user2 location information if it is an LBA and the user has enabled this application to see their information.

Mixing wild-card and specific schemes

The two schemes above can be combined, but this can be difficult. With the LBAs defined above, a user must allow access to the LBA that is at the lowest level of the hierarchy tree to allow this LBA access to their user location information. For example, if you have `www.ibm.com/services/index.html` and `www.ibm.com/services` defined as LBAs, a user must give access to the `www.ibm.com/services/index.html` for that LBA to be permitted to see his user location information. However, with the two LBS defined above, a user can access, for example, `www.ibm.com/services/index1.html` and this is permitted by the wild-carding scheme.

11.6.7 Muffin

In a full WebSphere Everyplace Server environment, LBS needs several other WebSphere Everyplace Server components to give information about the user and the mobile device in order to provide the location based information. The interaction between the components is discussed in 11.2, “Location-Based Services” on page 439. The information LBS needs is put in the HTTP header of the request, and can be found in the following header fields:

- X-IBM-PVC-User contains the user ID of the user. This is used by the LBS system to determine the applications authorization to request location information.
- X-IBM-PVC-Client-id contains the ID of the client device that the Location Server uses to define the location of the client (for instance, the Mobile Identification Number (MIN) of the device).

If you do not want to use the full WebSphere Everyplace Server environment, but only the LBS services, for example when you want to test with a WAP emulator, you can use tools that can manipulate the HTTP request and pass it on to the LBS system.

One such tool is Muffin. Muffin is a Java implementation of a filter for HTTP requests that can manipulate the HTTP headers. It is freely available under the GNU Public Licence. For more information on Muffin, go to:

`http://muffin.doit.org`.

Running muffin

Muffin is implemented as a proxy server. To run the Muffin proxy server, you will need Java version 1.3.0_02.

To start Muffin, extract the zipped file into its own directory. You can run Muffin by entering the following command:

```
<java path>jre.exe -cp <muffin home directory> Muffin
```

Where **`<java path>`** is the (optional) path of the Java run-time you want to use, and **`<muffin home directory>`** is the home directory of Muffin (the file Muffin.class resides in this directory).

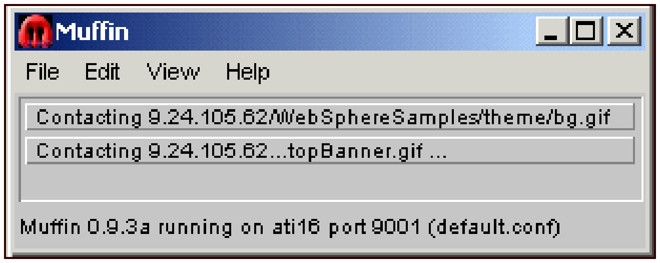
You must configure your browser to use this server as the proxy server. By default, Muffin listens on port 9001 for incoming requests.

After starting Muffin, under **Edit -> Options**, configure the destination proxy and the Host/Admin Allow and Deny options.

For LBS testing and to include the client ID and the user name in the request outgoing header, under **Edit -> Filters**, enable the LBSClientID and LBSUserName filters. In the extract directory, you will find user.txt and client.txt, where you should specify the user name and client ID to be included in the outgoing requests, repectively.

Note: You can use the Snoop filter to inspect the HTTP header.

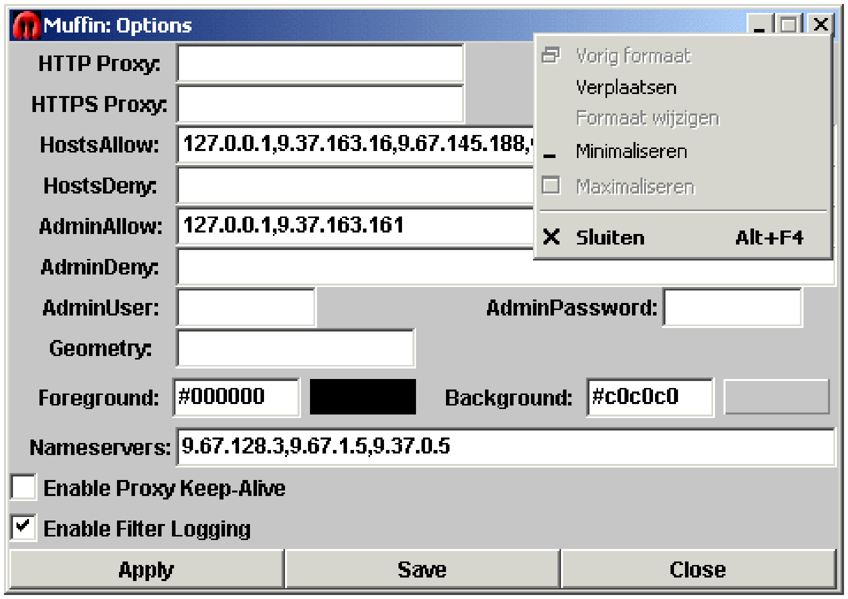
The next figures show the main screen of Muffin and the Options panel. Note how the main window shows a list of connections that exist on behalf of the client browser.



```
Contacting 9.24.105.62/WebSphereSamples/theme/bg.gif
Contacting 9.24.105.62...topBanner.gif ...
Muffin 0.9.3a running on ati16 port 9001 (default.conf)
```

Figure 11-28   Muffin main window

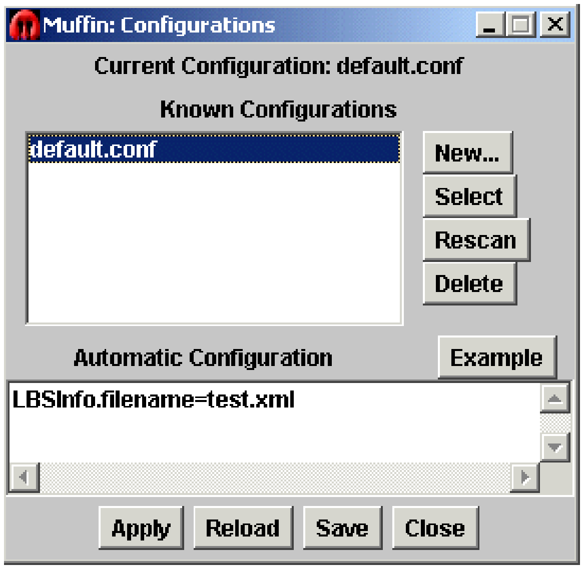
Figure 11-29 shows the Options panel. The fields HostsAllow and HostsDeny define which targets are allowed and disallowed for HTTP requests. The target server must be specified in the HostsAllow field.



HTTP Proxy:
HTTPS Proxy:
HostsAllow: 127.0.0.1,9.37.163.16,9.67.145.188,
HostsDeny:
AdminAllow: 127.0.0.1,9.37.163.161
AdminDeny:
AdminUser: AdminPassword:
Geometry:
Foreground: #000000 Background: #c0c0c0
Nameservers: 9.67.128.3,9.67.1.5,9.37.0.5
☐ Enable Proxy Keep-Alive
☑ Enable Filter Logging

Figure 11-29 Muffin Options Window

Figure 11-30 shows the configuration window. You may select the default configuration, or you can create your own.



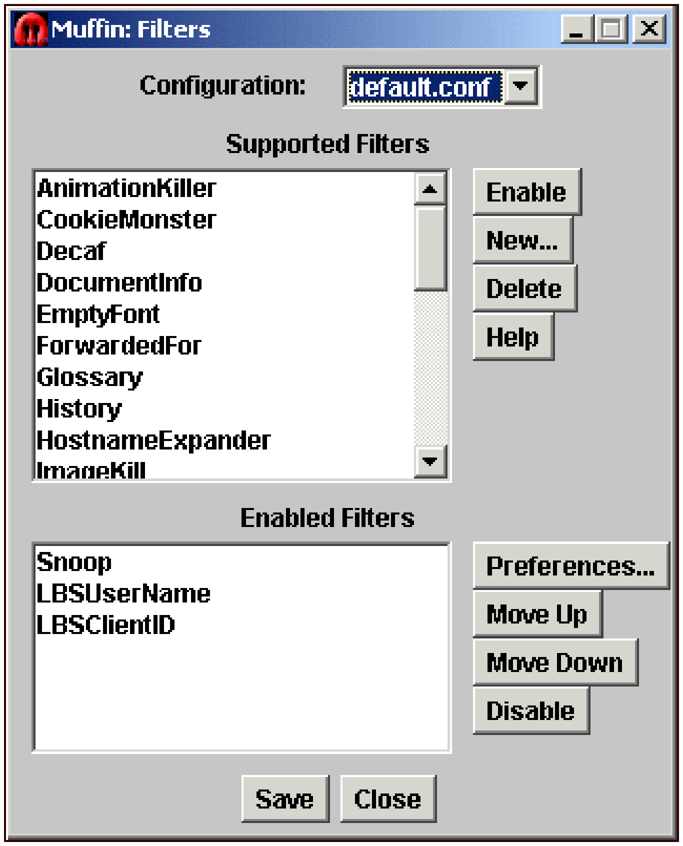
Current Configuration: default.conf

default.conf

LBSInfo.filename=test.xml

Figure 11-30 Muffin configuration window

The next figure shows the Filters window. You can select the filters that should be active, and the order in which they are applied.

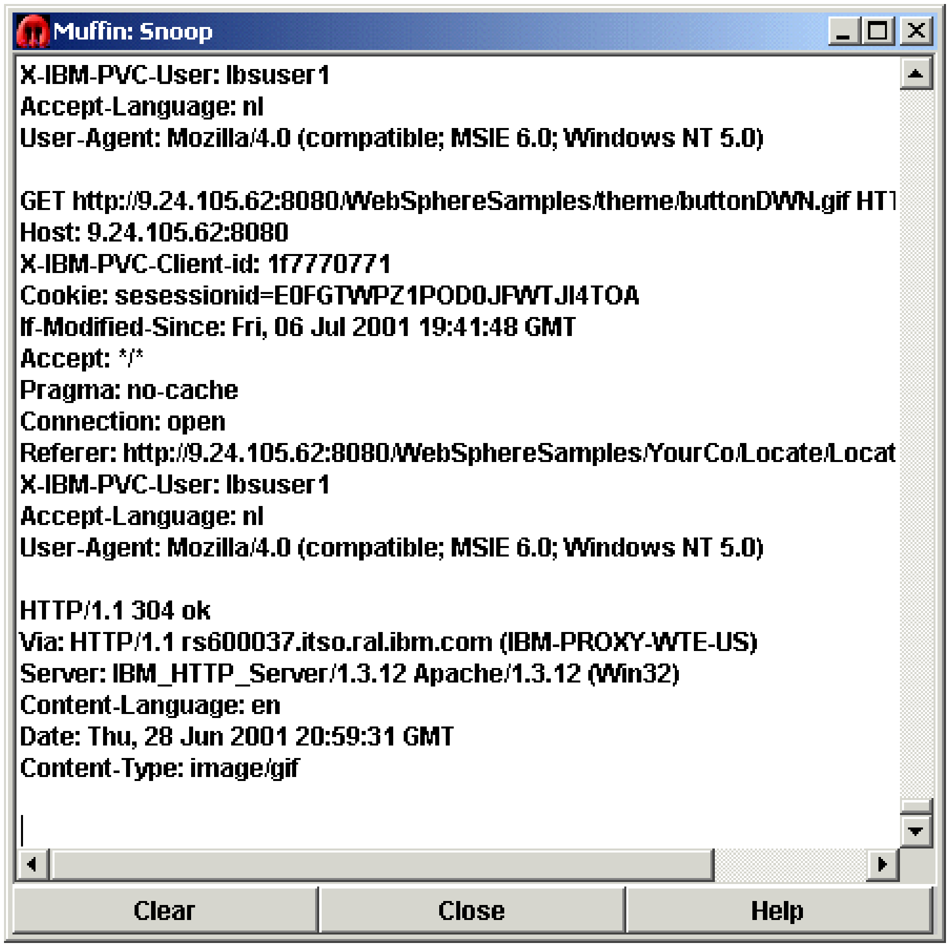


AnimationKiller
CookieMonster
Decaf
DocumentInfo
EmptyFont
ForwardedFor
Glossary
History
HostnameExpander
ImageKill

Snoop
LBSUserName
LBSClientID

Figure 11-31 The Muffin Filters page

If you select **Preferences...** for the Snoop filter, a panel with log information from the Snoop filter is displayed, as in Figure 11-32.

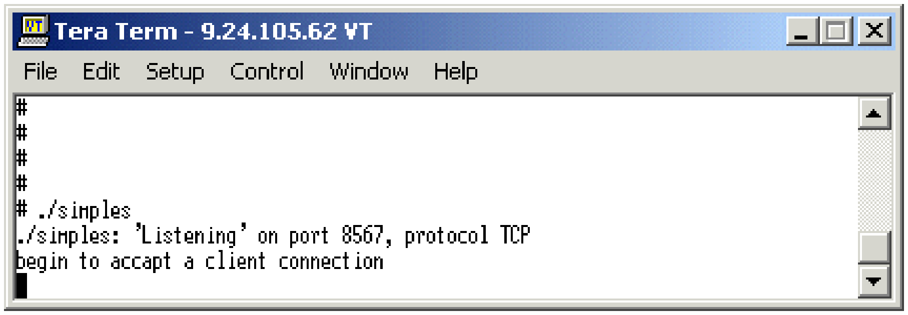


```
X-IBM-PVC-User: lbsuser1
Accept-Language: nl
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

GET http://9.24.105.62:8080/WebSphereSamples/theme/buttonDWN.gif HT
Host: 9.24.105.62:8080
X-IBM-PVC-Client-id: 1f7770771
Cookie: sesessionid=E0FGTWPZ1POD0JFWTJI4TOA
If-Modified-Since: Fri, 06 Jul 2001 19:41:48 GMT
Accept: */*
Pragma: no-cache
Connection: open
Referer: http://9.24.105.62:8080/WebSphereSamples/YourCo/Locate/Locat
X-IBM-PVC-User: lbsuser1
Accept-Language: nl
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

HTTP/1.1 304 ok
Via: HTTP/1.1 rs600037.itso.ral.ibm.com (IBM-PROXY-WTE-US)
Server: IBM_HTTP_Server/1.3.12 Apache/1.3.12 (Win32)
Content-Language: en
Date: Thu, 28 Jun 2001 20:59:31 GMT
Content-Type: image/gif
```

Figure 11-32 Preferences page for the Snoop filter, showing the HTTP header fields

11.6.8 Simples

Simples is an emulator for the SignalSoft Location Server. Simples is automatically installed with LBS and is started from the command prompt, as shown below. Simples listens on port 8567, and LBS must be configured to work with a Location Server on this port at installation time.


Tera Term - 9.24.105.62 VT
File Edit Setup Control Window Help

```
#
#
#
#
# ./simples
./simples: 'Listening' on port 8567, protocol TCP
begin to accapt a client connection
```

Figure 11-33 Running Simples on AIX

11.7 Troubleshooting

In this section, we include a list of possible problems you may encounter when implementing Location Based Services.

Problems with the LBS API

The log files of the Application Server contain information about the LBS API. If the location information cannot be found (the test file or the HTTP header field), this is logged in the Application Server Log.

Problems with the LBS Run-time

The run-time of LBS creates logging files in the WTE directory structure. The log files can be found at /opt/IBMWTE/usr/internet/server_root/logs. Here you can find several logs about WTE and LBS. These files can be read using the WebSphere Everyplace Server Suite Manager.

Problems with pd_populate.ksh

The pd_populate script has its own log file. This can be found at /usr/IBMEPS/LBS/logs/pd_populate.log. This file can be useful to inspect when enabling/disabling of LBAs does not work correctly, or when problems between LBS and PD are suspected.

Problems with Installation of Policy Director components

For detailed information on the installation of Policy Director and its components, refer to the WebSphere Everyplace Server Infocenter (the tpd directory), or the following redbook: *Tivoli SecureWay Policy Director Centrally Managing e-business Security,* SG24-6008.

12

Voice-enabled applications

In this chapter, you will find information describing ways to extend existing applications to include voice access using the IBM WebSphere Voice Server SDK in an Everyplace WebSphere Server environment. This chapter provides information about the VoiceXML browser, the speech recognition engine and text-to-speech engine included in the Voice Server SDK product. It also includes information about transcoding HTML and XML application content into VoiceXML using the IBM WebSphere Transcoding Publisher product.

A sample scenario using the YourCo B2E sample application is included to show how you can easily voice-enable your existing application when running in a WebSphere Everyplace Server environment.

Note: IBM WebSphere Everyplace Server Version 2.1 does not include the WebSphere Voice Server. In addition, you have to make sure that the WebSphere Transcoding Publisher (WTP) release you are using supports VoiceXML. VoiceXML transcoding support was included in WebSphere Transcoding Publisher Version 4.0.

12.1 Introduction

Voice applications are applications in which the input and output go through a spoken, rather than a graphical, user interface. The application files can reside on the local system, an intranet or the Internet. Users can access the deployed applications any time, anywhere, from any telephony-capable device, and you can design the applications to restrict access only to those who are authorized to receive it. The Voice Server model is illustrated in Figure 12-1.

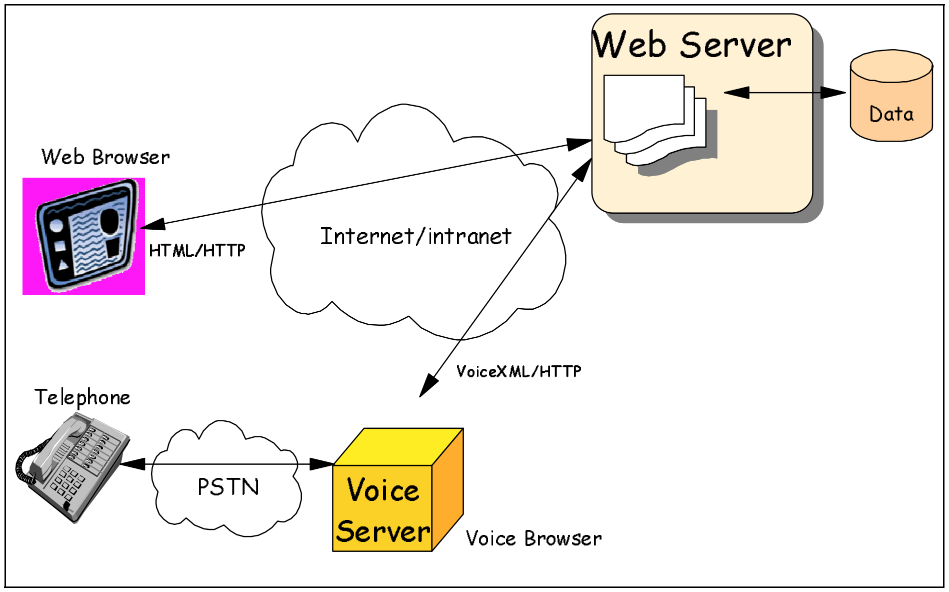


Figure 12-1 The Voice Server model

Voice applications provide an easy and novel way for users to browse the Internet using voice. Users can interact with Web-based data (servlets, JSPs, etc.) using speech rather than a keyboard and a mouse.

The typical voice applications are :

- Queries: a user calls a system to retrieve information from a Web-based infrastructure.
- Transactions: a user calls a system to execute specific transactions with a Web-based back-end.

12.1.1 WebSphere Voice Server offerings

A Voice Server provides speech access to Web content and business data through Speech Recognition, Text-to-Speech and the use of voice browsers. It offers a quick and easy way to develop speech applications using existing Web infrastructure.

IBM offers a number of products in the area. For example:

- The IBM WebSphere Voice Server Version 1.5 allows developers to create voice-enabled applications that utilize a Voice over IP network infrastructure. Voice access to Web applications provides customers with a more natural, easier-to-use method for information access.
- The IBM WebSphere Voice Server for DirectTalk 1.5 makes it possible to create integrated Web and telephone self-service access to business data and processes.
- The IBM WebSphere Voice Server SDK Version 1.5 uses VoiceXML technology to enable developers to create voice-enabled Web applications and test them on a desktop workstation. With these tools and the IBM WebSphere Voice Server, developers can make Internet applications accessible from many wireline or wireless devices.
- IBM Reusable Dialog Components are the building blocks for developing new VoiceXML applications. They allow developers with little VoiceXML experience to speed up application development and write basic functions.
- Reusable Dialog Components include:
  - Subdialogs – simple pieces of code that provide basic functions needed by VoiceXML developers. They can be grouped together to provide a function.
  - Templates – VoiceXML code that uses subdialogs to provide a common function.
  - Grammars – Java Speech Grammar Format (JSGF) grammars for use with the subdialogs.
- The Voice Toolkit, currently available as beta code, can help developers to create voice applications in less time, using a VoiceXML application development environment. The toolkit features grammar and VoiceXML editors so that application developers do not need to know the internals of voice technology. Web developers, telecommunications developers, and other developers can build applications that take advantage of voice as a means of accessing information.

This chapter presents the use of the IBM WebSphere Voice Server Software Developers Kit (SDK) Version 1.5 to adapt and extend existing applications in a WebSphere Everyplace Server environment using transcoding techniques.

For more detailed information about IBM WebSphere Voice Server offerings and general voice topics, see for example:

- The voice systems Web site:

  `www.ibm.com/software/voice`
- The voice systems intranet site:

  `w3.software.ibm.com/voicesystems`
- The VoiceXML forum Web site:

  `www.voicexml.org`

12.2 VoiceXML language

The Voice eXtensible Markup Language (VoiceXML) is an XML-based markup language for creating distributed voice applications, much as HTML is a markup language for creating distributed visual applications. VoiceXML is an emerging industry standard that has been defined by the VoiceXML forum of which IBM is a founding member. It has been accepted for submission by the World Wide Web Consortium as a standard for voice markup language on the Web.

The VoiceXML language enables Web developers to use a familiar markup style and Web server-side logic to deliver voice content to the Internet. For example, a very simple Hello World document in VoiceXML looks like this:

```
<?xml version="1.0"?>
<vxml version="1.0">
    <form>
        <block>
            Hello World!
        </block>
    </form>
</vxml>
```

Using VoiceXML, you can create Web-based voice applications that users can access by phone. VoiceXML supports dialogs that feature:

- Spoken input
- DTMF input (telephone key)
- Recording of spoken input
- Synthesized speech output (Text To Speech)
- Recorded audio output
- Dialog flow control
- Scoping of input

VoiceXML implements a client-server mode, where a Web server provides VoiceXML documents containing dialogs to be interpreted and presented to the user. The user's responses are submitted to the Web server, which responds by providing additional VoiceXML documents, as appropriate.

Unlike a proprietary Interactive Voice Response (IVR) system, VoiceXML provides an open application development environment that generates portable applications. This makes VoiceXML a cost-effective alternative for providing voice access services. The main problem of VoiceXML applications is to recognize the spoken input. Therefore, so-called *grammar tags* are used. All possible and expected input from the user has to be defined and stored here.

It is also possible to include prerecorded audio files in a VoiceXML document. This offers the user a kind of real human interface instead of the common robotic touch of synthesized speech.

To develop VoiceXML applications, IBM WebSphere Studio Version 3.5 or later can be used. The tool provides different wizards, a visual editor and code assistance to create the documents. For testing the VoiceXML files within your application, you can use the IBM WebSphere Voice Server SDK.

12.2.1 VoiceXML application development

To create a voice application written in VoiceXML, you can use the Voice Server SDK and, optionally, a Web site development tool such as IBM WebSphere Studio. The VoiceXML pages can be static or they can be dynamically generated using server-side logic (CGI scripts, Java Beans, ASP, JSP, etc.).

IBM WebSphere Studio provides you with a graphical development environment that helps you create and manage VoiceXML files. For example, WebSphere Studio Version 3.5 has been enhanced to support VoiceXML files: it includes a VoiceXML Editor with code assist features, wizards capable of generating Java Server Pages for use with VoiceXML files, and the ability to preview VoiceXML files using the Voice Server SDK's VoiceXML browser.

12.2.2 VoiceXML sample applications

Before running a VoiceXML application, we recommend that you first install the Voice Server SDK.

For example, WebSphere Voice Server SDK Version 1.5 requires the following:

- Windows NT Version 4.0 plus Service Pack 6a applied.
- Multimedia Adapter and appropriate driver.

- ViaVoice Text To Speech (TTS).
- Java Runtime Environment (JRE) Version 1.3.0, which is included in the Voice Server SDK package but must be installed prior to the Voice Server SDK.

The Voice Server SDK includes some VoiceXML application samples which are stored in the /”install directory”/samples directory. In the /samples/en_US directory, you will find the following subdirectories:

- AudioSample. This contains AudioSample.au (sound format), AudioSample.vxml and vxml.log.
- GrammarBuilderSample. This contains the GrammarBuilderTool.
- VoiceSnoopSample.

You can run the Audio Sample from the Windows Start menu by choosing **Start>Programs->IBM WebSphere Voice server SDK->Audio Sample**.

This sample VoiceXML file will prompt you to say something; it will record this until you say “Stop Now” or click a key on the DTMF keypad. It will then replay the recorded text back to you. The AudioSample is listed here for your reference:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml version="1.0">
<!-- Copyright (c) 2000-2001 IBM Corp.  All Rights Reserved. -->
   <form id="audiosample">
        <property name="bargein" value="false"/>
        <block>
            <prompt>
           <audio src="AudioSample.au"/>
                Welcome to the IBM Voice Server recording and playback sample.
                With this sample, you can record a message and
                then play back your recording.
            </prompt>
        </block>
      <field name="answer" type="boolean">
            <prompt>
                Would you like to continue?
            </prompt>
            <catch event="help nomatch noinput">
                Please say yes or no.
            </catch>
            <catch event="nomatch noinput" count="4">
                Couldn't recognize yes or no.  Exitting.
                <exit/>
            </catch>
            <filled>
                <if cond="answer == false">
                    Goodbye.
                    <exit/>
                </if>
            </filled>
        </field>
          <record name="recording" beep="true">
            <catch event="noinput">
                Couldn't tell if recording finished.
                <clear namelist="answer"/>
      <goto nextitem="answer"/>
            </catch>
            <prompt>
                Begin speaking after the tone.  When finished recording, pause
briefly, then
      say "Stop Now" or press a D T M F key.
            </prompt>
            <grammar>stop now</grammar>
        </record>
        <filled>
            <prompt>
                You recorded:<value expr="recording"/>
                Goodbye.
            </prompt>
        </filled>
    </form>
</vxml>
```

Example of VoiceXML log file

When the VoiceXML browser is running, it writes all trace messages to the vxml.log file. A sample trace is shown here:

```
12:31:52.551 S: Starting V010412 at Thu Sep 13 12:31:52 EDT 2001
12:31:52.551 X: Java: Sun Microsystems Inc. 1.3.0
12:31:52.551 X: requested locale: en_US
12:32:01.554 A: not listening
12:32:01.594 K: creating new index (reason: java.io.FileNotFoundException:
cache/index (The system cannot find the file specified))
12:32:01.614 A: listening
12:32:03.707 S: initializing application
12:32:03.707 V:
file:d:/VoiceServerSDK/samples/en_US/AudioSample/AudioSample.vxml (fetch get)
12:32:03.797 F: file:d:/VoiceServerSDK/samples/en_US/AudioSample/AudioSample.au
(prefetch get)
12:32:04.137 K: put 000000 builtin:grammar/boolean
12:32:04.228 K: put 000001 builtin:dtmf/boolean
12:32:04.398 S: running
12:32:04.398 C: (audio clip)
12:32:04.418 C: Welcome to the IBM Voice Server recording and playback sample.
                                With this sample, you can record a message and
                                then play back your recording.
```

```
12:32:04.598 C: Would you like to continue?
12:32:18.108 ?: check (too short)
12:32:19.720 A: Too quiet (0.2)
12:32:19.940 A: Too loud (0.8)
12:32:21.703 H: yes
12:32:21.743 C: Begin speaking after the tone.  When finished recording, pause
briefly, then
                say "Stop Now" or press a D T M F key.
12:32:21.803 C: (audio clip)
12:32:21.893 A: Audio level (0.6)
12:32:30.505 A: recording started
12:32:32.028 A: Audio level (0.5)
112:32:35.052 ?: stop now (too short)
12:32:35.402 A: Audio level (0.3)
12:32:36.224 A: Too quiet (0.1)
12:32:36.885 ?: stop now (too short)
12:32:46.238 A: Too quiet (0.2)
12:32:46.568 A: Too quiet (0.2)
12:32:51.025 T: 1
12:32:51.095 H: 1
12:32:51.105 A: recording stopped
12:32:51.145 C: You recorded:
12:32:51.185 C: (recorded audio clip)
12:32:51.185 C: Goodbye.
12:33:11.885 A: Audio level (0.4)
12:33:12.005 A: Too quiet (0.1)
12:33:13.087 ?: help please (still speaking)
12:33:15.740 K: clean cache 1
12:33:15.781 K: lock retries 0
12:33:15.781 S: exit (0)
```

12.2.3 Grammars

Speech recognition grammars are a key component of VoiceXML application design. A grammar is an enumeration, in compact form, of the set of utterances (words and phrases) that constitute the acceptable user response to a given prompt. The VoiceXML browser requires all speech and DTMF grammars to be specified using the Java Speech Grammar Format (JSGF).

When you write your application, you can use the built-in grammars and create one or more of your own. In either case, you must decide when each grammar should be active.

Grammar syntax is also important. A JSGF grammar file consists of a grammar header and a grammar body.

- The grammar header provides the version of JSGF and the grammar name.
- The grammar body consists of one or more rules that define the valid set of utterances.

Following is an example of a grammar:

```
<grammar>
John | Nicolas | Isabel | Erik | Patrick
</grammar>
```

Following is an example of a DTMF grammar:

```
<dtmf type=”application/x-jsgf”>
1 | 2 | 3 | 4 | “#” | “*”
</dtmf>
```

This example defines an inline DTMF grammar that allows users to make a selection by pressing the numbers 1 through 4, the pound sign or the asterisk on the DTMF simulator.

Improving grammar performance

Grammar design can have a significant impact on response time. Here are some tips :

- The longer the user utterance (words, phrases), the longer the response time.
- Increasing the rule depth (in using subrules) increases the response time.
- When testing your grammars, you should test words and phrases that are in your grammars, as well as words and phrases that are not.
- If your application has more than one grammar active simultaneously, you should test each grammar separately and then test them together.

For more information about grammars, you can explore the following link :
`http://java.sun.com/products/java-media/speech/forDevelop-ers/JSGF/index.html`

12.3 IBM WebSphere Voice Server SDK

The IBM WebSphere Voice Server SDK brings support for VoiceXML to Web application development activities.

With the Voice Server SDK, you can create and test Web-based voice applications. The SDK uses the workstation's speakers to play audio output. You can input data using the workstation's microphone, prerecorded audio files, or the Voice Server SDK's DTMF simulator.

The Voice Server SDK includes the following:

- A speech browser that interprets VoiceXML markup. This VoiceXML browser includes a DTMF simulator to generate simulated telephone keypress input during desktop testing.
- IBM Via Voice Speech Recognition and Text-To-Speech engines for accepting voice input and generating synthesized speech output.
- Telephony acoustic models to approximate the speech recognition behavior of applications deployed in a telephony environment.
- An audio setup program to configure your microphone and speakers for use with this product.
- User documentation.
- Sample VoiceXML files.

12.3.1 WebSphere Voice Server SDK architecture

The WebSphere Voice Server SDK includes the following components : Speech Recognition engine, Text-To-Speech engine and the VoiceXML browser.

Figure 12-2 on page 503 shows the interaction of Voice Server SDK with information stored on Web application servers and in back-end enterprise databases.

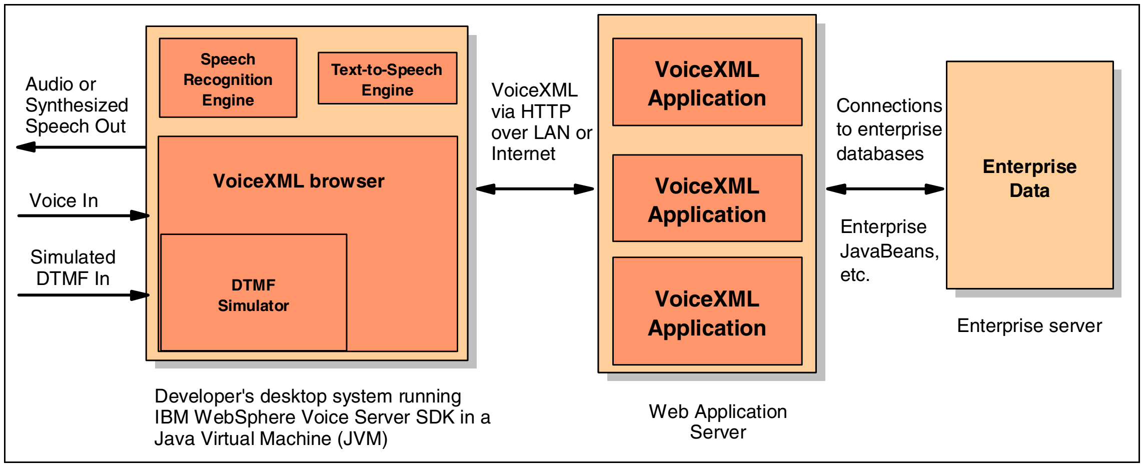

Audio or Synthesized Speech Out
Voice In
Simulated DTMF In
Speech Recognition Engine
Text-to-Speech Engine
VoiceXML browser
DTMF Simulator
Developer's desktop system running IBM WebSphere Voice Server SDK in a Java Virtual Machine (JVM)
VoiceXML via HTTP over LAN or Internet
VoiceXML Application
VoiceXML Application
VoiceXML Application
Web Application Server
Connections to enterprise databases
Enterprise JavaBeans, etc.
Enterprise Data
Enterprise server

Figure 12-2 WebSphere Voice Server SDK architecture

Speech recognition engine

Speech recognition is the ability of a computer to decode human speech and convert it to text.

To convert spoken input to text, the computer must first parse the input audio stream and then convert that information to text output. In the Voice Server SDK, this is done by the ViaVoice Speech Recognition engine.

First, the application developer creates a series of speech recognition grammars defining the words and phrases that can be spoken by the user, and specifies where each grammar should be active within the application.

When the application runs, the speech recognition engine processes the incoming audio signal and compares the sound patterns to the patterns of basic spoken sounds, trying to determine the most probable combination that represents the audio input.

The speech recognition engine then compares the sounds of the list of words and phrases in the active grammars. Speech recognition accuracy key determinants are:

- Audio input quality : the quality of audio input depends on the choice of input device (microphone, telephone), speaking environment (noisy or quiet place), and quality of the user’s pronunciation.
- Interface design.
- Grammar design : the only possible speech recognition candidates are the words, phrases and DTMF key sequences included in active grammars. So

the content of these grammars will have a major impact on speech recognition accuracy.

Text-To-Speech Engine

Text-To-Speech (TTS) is the ability of a computer to generate spoken output from text input. In order to generate synthesized speech, the computer must first parse the input text to determine its structure and then convert that text to spoken output. In the Voice Server SDK, this is done by the ViaVoice Text-To-Speech engine.

The main capabilities and limitations of Text-To-Speech are as follows:

- TTS prompts are easier to maintain and modify than recorded audio prompts. For this reason, TTS is typically used during application development.
- TTS is also a powerful tool for use when the data to be spoken is not known in advance and cannot therefore be prerecorded.
- The synthesized speech is a bit robotic.

The synthesized voice can be altered to add emphasis to some parts of the text. Different genders and ages of voice can be added by inserting VXML tags in the text. Different types of emphasis can also be introduced by adding question marks and exclamation points where appropriate in the text.

VoiceXML browser

One of the primary functions of the VoiceXML browser is to fetch documents to process. The request to fetch a document can be generated either by the interpretation of a VoiceXML document or in response to an external event. The VoiceXML browser is a Java application.

The VoiceXML browser manages the dialog between the application and the user by playing audio prompts, accepting user inputs, and acting on those inputs. The action might involve jumping to a new dialog, fetching a new document or submitting user input to the Web server for processing.

Note: The VoiceXML browser included in WebSphere Voice Server supports a subset of the VoiceXML 1.0 specification.

In addition to spoken input, you can allow users to use the DTMF (telephone keypress) simulator. The DTMF simulator is a GUI tool that enables you to simulate DTMF tones on your workstation.

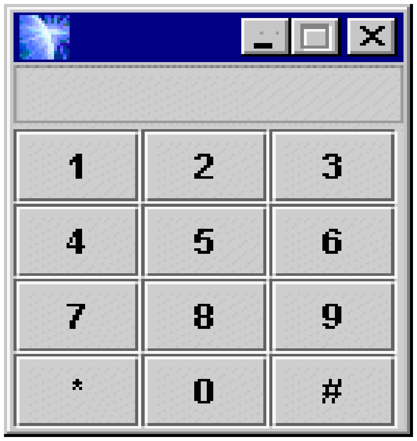


Figure 12-3 DTMF GUI

The VoiceXML browser starts the DTMF simulator automatically. If you close the DTMF simulator GUI window, the only way to restart it is to stop and restart the VoiceXML browser.

During Voice applications' development and testing, the DTMF simulator plus the microphone and speakers take the place of a telephone, allowing you to debug your VoiceXML applications without having to connect to telephony hardware and the PSTN (Public Switched Telephone Network).

Interactions with Text-To-Speech and speech recognition engines

When you start the VoiceXML browser, TTS and speech recognition engines are automatically started. As the VoiceXML browser processes a VoiceXML document, it plays audio prompts using TTS or recorded audio. For TTS output, the VoiceXML browser interacts with the TTS engine to convert the text into audio.

As the VoiceXML browser makes transitions to new dialogs or new documents, it enables and disables different speech recognition grammars, as specified by the VoiceXML application. As a result, the list of valid user utterances changes.

Interactions with the Web server and enterprise data server

You can publish your VoiceXML applications to any Web server running on any platform. However, it is recommended that you use WebSphere Application Server 3.5 (or later version) which has been enhanced to receive VoiceXML pages published from WebSphere Studio 3.5 (or later version).

WebSphere Studio Version 3.5 includes the following support for VoiceXML files:

- A VoiceXML editor with coding assistance and syntax checking features.
- Wizards capable of generating servlets, JavaBeans and JSPs for creating dynamic VoiceXML documents.
- The ability to import VoiceXML applications, follow flow control links, and natively understand VoiceXML tags.
- The ability to preview VoiceXML files using the Voice Server SDK's VoiceXML browser.
- The ability to publish VoiceXML applications into WebSphere Application Server.

When you start the VoiceXML browser, it sends an HTTP request over the LAN or Internet for an initial VoiceXML document from the Web server. The requested VoiceXML document can contain static information, or it can be generated dynamically from data stored in a database using the same type of server-side logic (CGI, JSP, Java Beans, etc.) that you use to generate dynamic HTML documents. The VoiceXML applications can access the same information from enterprise servers that HTML applications do. The VoiceXML browser then interprets and renders the document.

12.3.2 Security Issues

Voice Services is an implementation on WebSphere Everyplace Server which utilizes Voice eXtended Markup Language (Voice XML or VXML) to provide voice-based authentication to the WebSphere Everyplace Server environment.

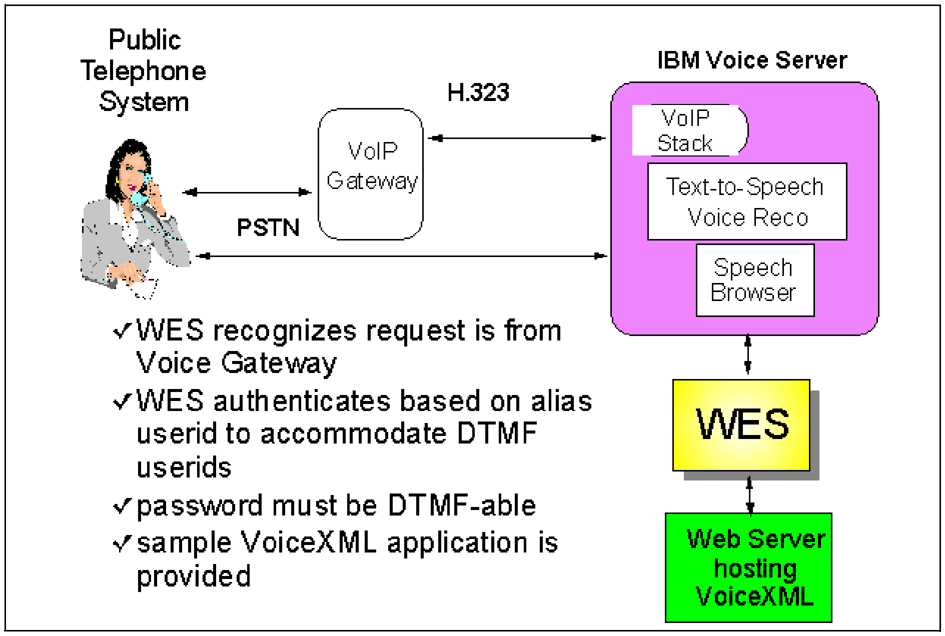
During the WebSphere Everyplace Server installation, WebSEAL-Lite is installed as a plug-in to the WebSphere Edge Server Caching Proxy component. Voice Services is installed during the WebSEAL-Lite installation.

There are three sample files laid down with WebSEAL-Lite, in the Samples directory under WebSEAL-Lite, which may be modified to enable personalized voice services:

- wesloginform.vxml - controls the message the user hears when the page is displayed showing that the user is either authenticated or the login denied. Use this file to customize the attributes of Voice Services particular to your company and the applications accessed by the user.
- wesloginfail.vxml - shows how the VXML Voice Server can authenticate a user ID and password. Use this file to customize the page displayed when the user's authentication fails.

- weshelloworld.vxml - shows how a VXML voice server can authenticate a user ID and password. Use this file to customize the location of the login page.

Figure 12-4 depicts voice-enabling Web sites at a high level.



Public Telephone System
H.323
IBM Voice Server
VoIP Stack
VoIP Gateway
Text-to-Speech Voice Reco
PSTN
Speech Browser
✓WES recognizes request is from Voice Gateway
✓WES authenticates based on alias userid to accommodate DTMF userids
✓password must be DTMF-able
✓sample VoiceXML application is provided
WES
Web Server hosting VoiceXML

Figure 12-4 Authentication through Voice Services

This basic scenario includes a caller dialing through a public switched telephone network to reach the IBM Voice Server. There are two methods available for accessing the voice server, Voice over IP (VoIP) and standard voice communications.

When a call arrives at the Voice Server, it is processed, converted to VXML/HTTP, and passed to the WebSphere Everyplace Server environment where authentication (and authorization, if implemented) is performed. Voice Services allows a user to set up an alternate alias for telephony access. The alias typically consists of a numeric entry (telephone) number to substitute for the user ID when connecting through this method.

Once the user successfully establishes a numeric alias, Voice Services prompts the user to change his or her existing password to a numeric entry. Authentication is based on this numeric alias and password combination as entered through the telephone DTMF (touchtone) buttons.

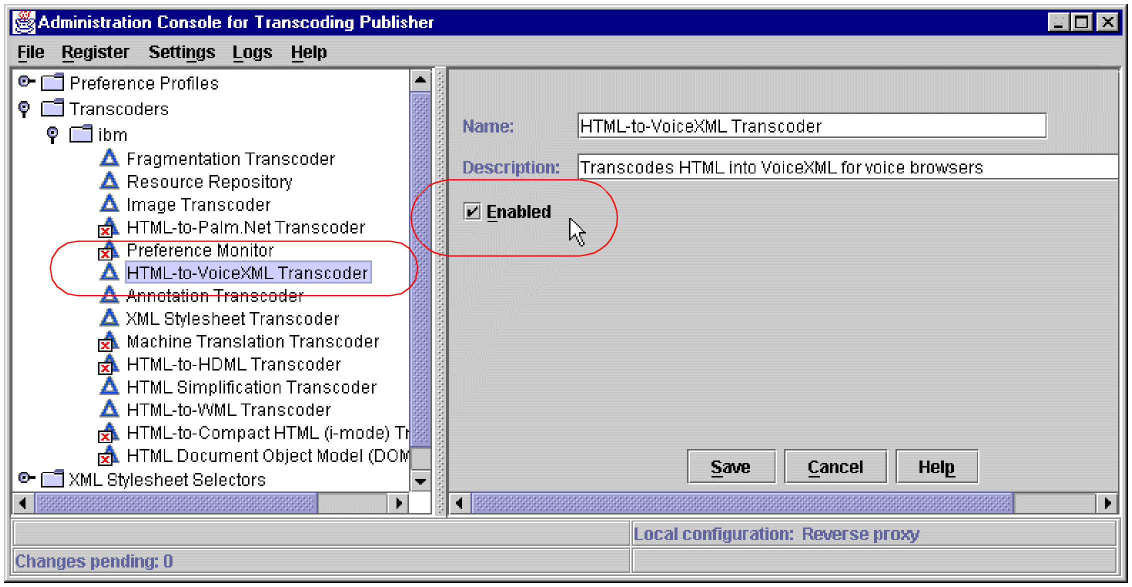
12.4 VoiceXML transcoding

The IBM WebSphere Transcoding Publisher (WTP) 4.0 allows you to transform an HTML or XML document into a VoiceXML document. In addition to transcoding, you probably would like to add text clipping functions for HTML application content and the use of stylesheets for XML application content as required by your voice-enabled applications.

WebSphere Transcoding Publisher provides functions that can simplify an HTML page and make it more usable as a voice application, such as the ability to split the document into sections and add annotations. WebSphere Transcoding Publisher can use annotations to remove unwanted data from an HTML page before it is transcoded into VoiceXML. This transcoding process works on a page by page basis.

Note: For details about using text clipping with annotations in WebSphere Transcoding Publisher, see Chapter 5, “Text clipping” on page 145. For details about using stylesheets, see Chapter 6, “Using stylesheets” on page 211.

Important: The HTML to VoiceXML transcoder is not enabled by default in WebSphere Transcoding Publisher. It should be enabled through the WebSphere Transcoding Publisher Administrator Console (see Figure 12-5).


Administration Console for Transcoding Publisher
File
Register
Settings
Logs
Help
Preference Profiles
Transcoders
ibm
Fragmentation Transcoder
Resource Repository
Image Transcoder
HTML-to-Palm.Net Transcoder
Preference Monitor
HTML-to-VoiceXML Transcoder
Annotation Transcoder
XML Stylesheet Transcoder
Machine Translation Transcoder
HTML-to-HDML Transcoder
HTML Simplification Transcoder
HTML-to-WML Transcoder
HTML-to-Compact HTML (i-mode) Tr
HTML Document Object Model (DOM
XML Stylesheet Selectors
Save
Cancel
Help

Name: HTML-to-VoiceXML Transcoder

Description: Transcodes HTML into VoiceXML for voice browsers

☑ Enabled

Local configuration: Reverse proxy

Changes pending: 0

Figure 12-5 Enabling HTML to VoiceXML transcoder

12.5 Sample scenario

In this section, we present a sample scenario that shows how you can access an HTML application and enable it for voice access using HTML to WML transcoding. This scenario uses the location-based application that was included as an extension of the B2E YourCo application in Chapter 11, “Location-Based Services (LBS)” on page 435.

The following diagram illustrates the components required to access the YourCo application from the Voice Server SDK; they are:

- A Windows machine with a multimedia card for application development (see Figure 12-6 for details).

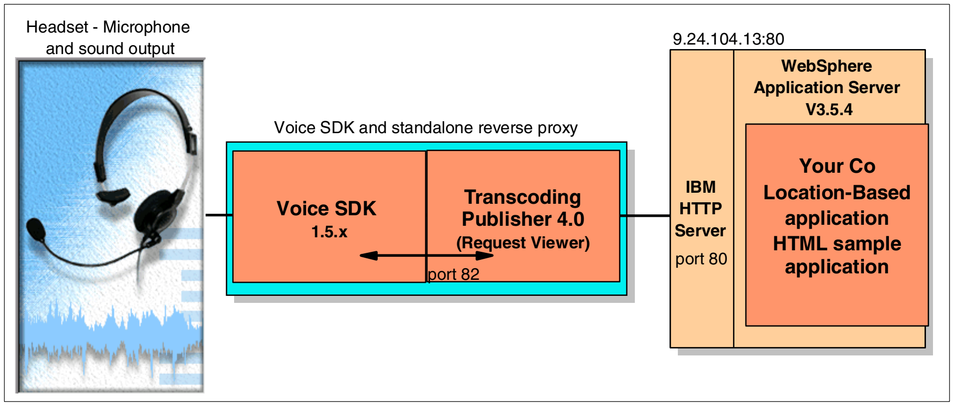
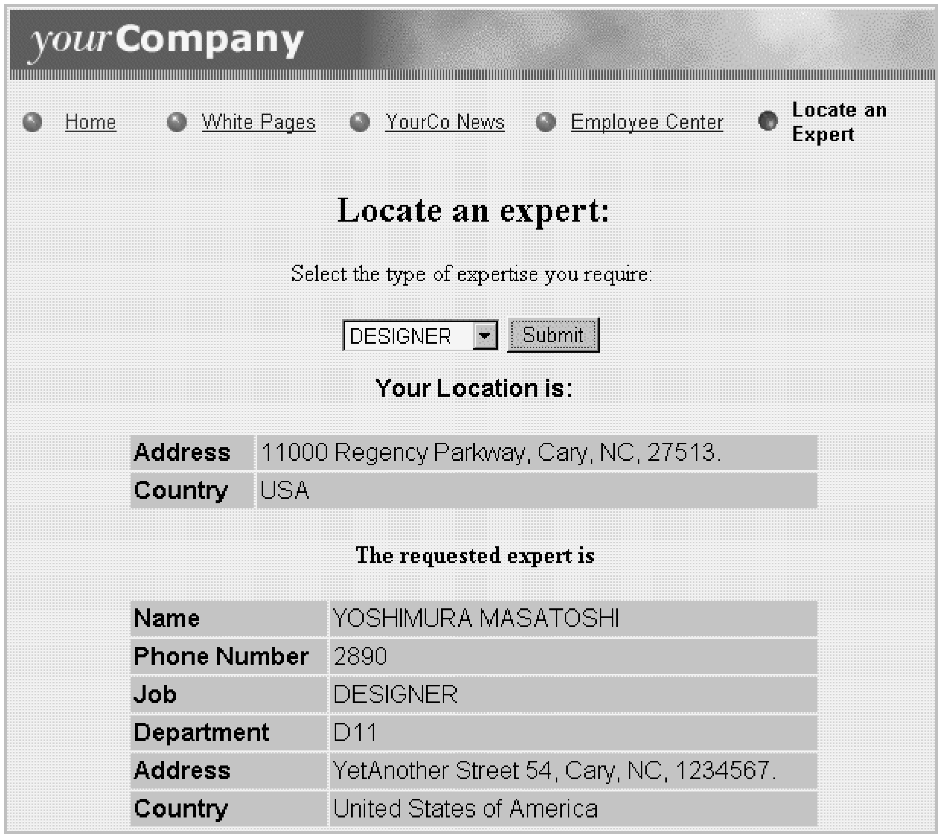

Headset - Microphone and sound output
Voice SDK and standalone reverse proxy
Voice SDK 1.5.x
Transcoding Publisher 4.0 (Request Viewer)
port 82
9.24.104.13:80
IBM HTTP Server
port 80
WebSphere Application Server V3.5.4
Your Co Location-Based application HTML sample application

Figure 12-6 Sample scenario using Voice SDK

- A headset with microphone and sound output connected to the Windows development workstation multimedia card.
- WebSphere Voice Server SDK 1.5.
- IBM WebSphere Transcoding Publisher (WTP) installed as a reverse proxy and listening on port 82. In this scenario, WebSphere Transcoding Publisher is installed on the same machine. For details about using WebSphere Transcoding Publisher as a reverse proxy, see Chapter 4, “Transcoding application content” on page 113. The WebSphere Transcoding Publisher reverse proxy should be configured to forward requests to the application server where the HTML application is actually running.
- An Application Server running the YourCo sample application, specifically the location-based extension implemented in Chapter 11, “Location-Based Services (LBS)” on page 435.

12.5.1 Voice access process

The window in Figure 12-7 shows the look and feel of the Location-Based Services (LBS) application's result when accessed from a desktop browser. The objective of this scenario is to access and run this Web application with voice and be able to *hear* the results shown in this screen shot.



Locate an expert:

Select the type of expertise you require:

Your Location is:

Address	11000 Regency Parkway, Cary, NC, 27513.
Country	USA

The requested expert is

Name	YOSHIMURA MASATOSHI
Phone Number	2890
Job	DESIGNER
Department	D11
Address	YetAnother Street 54, Cary, NC, 1234567.
Country	United States of America

Figure 12-7 Sample HTML application accessed from the browser

Once the hardware and software have been properly installed, you may want to execute the following steps:

1. Open a command window on the SDK machine and change the directory to the \bin directory under the SDK's installation location, for example:

   ```
   cd \Program Files\VoiceServerSDK\bin
   ```

2. Run the VoiceXML browser using the URL for the Locate an Expert application. Remember to use the correct port number for the Transcoder or the browser will receive HTML instead of VoiceXML. Also, make sure the

HTML to VoiceXML transcoder has been enabled. In our example, the command is:

```
vsaudio http://localhost:82/WebSphereSamples/YourCo/LocateInput.html
```

where **`localhost`** is used, since WebSphere Transcoding Publisher is running on the same machine, but you can also use a host name or a real IP address.

3. When the application access is successful, listen through the earphones and follow the instructions to query your location service. For example, you can say in the microphone : “Locate an expert”.

Sample dialog

In general, a dialog will run like this:

```
SDK: Please choose one of the following sections to go to: Main Content, Links,
Exit.
Say: Main Content.

SDK: To exit the browser, say 'Exit,' otherwise, please choose one of the
following topics: Locate an Expert, Exit.
Say: Locate an Expert.

SDK: Locate an Expert...Select the type of expertise you require.
Say: Designer (if you delay to long in replying, the script will start again
from the top).

SDK: Do you want to complete this transaction?
Say: Yes.
```

Note: At this time, the actual submit is sent and you can monitor this second request in WebSphere Transcoding Publisher Request Viewer.

```
SDK: Please choose one of the following sections to go to: Main Content, Exit.
Say: Main Content.

SDK: To exit the browser, say 'Exit,' otherwise, please choose from one of the
following topics: Your Location is, The Requested Expert.
Say: Your location is.
```

Note: At this time, you will listen to your location address and country. Notice that the page has two parts called ‘Your location is’ and ‘The requested expert is’ (see Figure 12-7 for details).

```
SDK: To exit the browser, say 'Exit,' otherwise, please choose from one of the
following topics: Your Location is, The Requested Expert.
Say: The Requested Expert.
```

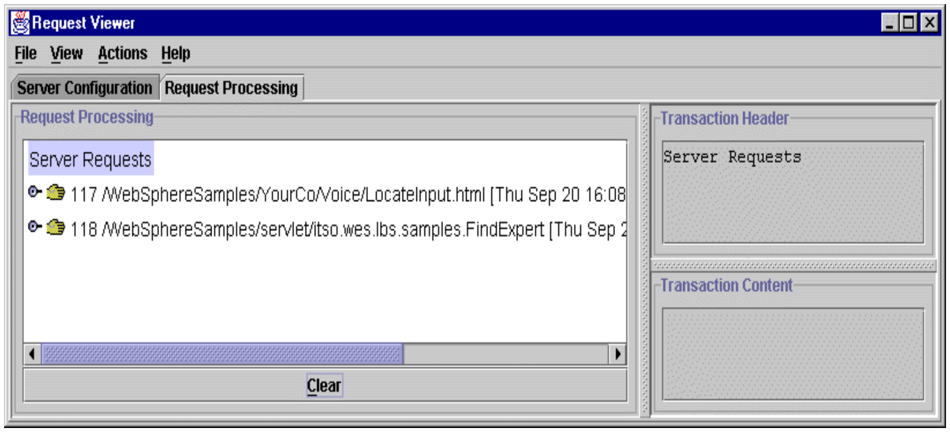
Note: At this time, you will listen to the requested expert information as provided by the HTML application.

Here is the content of the command window which appears during the voice application access process:

```
16:08:01.157 S: Starting V010412 at Thu Sep 20 16:08:01 EDT 2001
16:08:01.157 X: Java: Sun Microsystems Inc. 1.3.0
16:08:01.157 X: requested locale: en_US
16:08:07.126 A: not listening
16:08:07.156 A: listening
16:08:09.109 S: initializing application
16:08:09.149 V:
http://localhost:82/WebSphereSamples/YourCo/Voice/LocateInput.html (re-fetch
get)
16:08:09.299 K: file not modified
16:08:09.920 S: running
16:08:09.930 C: Please choose one of the following sections to go to. <BREAK
SIZE="large"></BREAK>  Main Content. Exit.
16:08:19.544 H: Main Content
16:08:19.564 C: To exit the browser, say exit.  Otherwise, please choose one of
the following topics. <BREAK SIZE="large"></BREAK>  Locate an expert:. Exit.
16:09:04.218 A: Too quiet (0.2)
16:09:05.159 H: Locate an expert
16:09:05.169 C: Locate an expert:
16:09:05.229 C: Select the type of expertise you require:
16:09:11.889 A: Audio level (0.4)
16:09:12.230 A: Too loud (1.0)
16:09:53.679 H: DESIGNER
16:09:53.689 C: Do you want to complete this transaction?
16:09:59.027 H: yes
16:09:59.077 V:
http://localhost:82/WebSphereSamples/servlet/itso.wes.lbs.samples.FindExpert
(fetch post)
16:10:03.543 F: recv cookie sesessionid=BYK2ZHUANYBOKXGIFY2RXEQ for localhost
16:10:03.633 C: Please choose one of the following sections to go to. <BREAK
SIZE="large"></BREAK>  Main Content. Exit.
16:10:12.206 A: Too loud (0.8)
16:10:14.369 H: Main Content
16:10:14.379 C: To exit the browser, say exit.  Otherwise, please choose one of
the following topics. <BREAK SIZE="large"></BREAK>  Your Location is:. The
requested expert. Exit.
16:10:24.503 A: Too quiet (0.1)
16:10:29.431 H: The requested expert
16:10:29.441 C: The requested expert is Name YOSHIMURA MASATOSHI Phone Number
2890 Job DESIGNER Department D11 Address YetAnother Street 54, Cary, NC,
1234567. Country United States of America
16:10:29.601 V: #start
16:10:29.691 C: Please choose one of the following sections to go to. <BREAK
SIZE="large"></BREAK>  Main Content. Exit.
16:10:30.071 A: Too quiet (0.1)
16:10:32.545 A: Audio level (0.3)
```

Monitoring with the Request Viewer

The Request Viewer is the component of WebSphere Transcoding Publisher that allows you to view and monitor the transactions performed through WebSphere Transcoding Publisher. A sample view is shown in Figure 12-8; the first entry is the initial request (form request) and the second entry is the actual submit (post) operation.

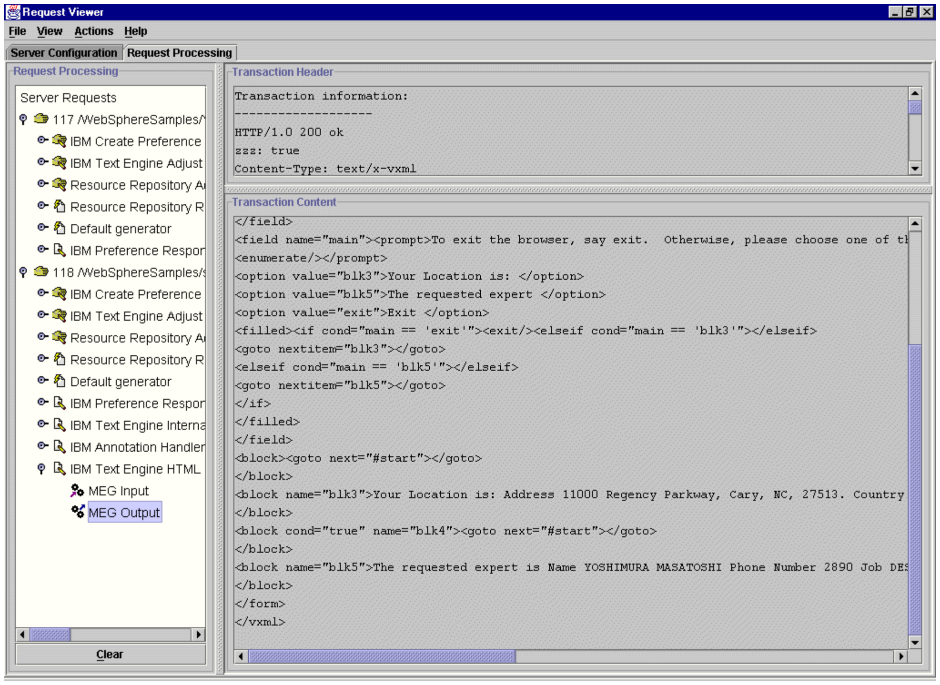


Server Requests
117 /WebSphereSamples/YourCo/Voice/LocateInput.html [Thu Sep 20 16:08
118 /WebSphereSamples/servlet/itso.wes.lbs.samples.FindExpert [Thu Sep 2

Server Requests

Figure 12-8 Monitoring VoiceXML traffic with the Request Viewer tool

Figure 12-9 shows how the Request Viewer tool can also be used to display the transcoded VoiceXML content.



```
Transaction information:
-------------------
HTTP/1.0 200 ok
zzz: true
Content-Type: text/x-vxml
```

```
</field>
<field name="main"><prompt>To exit the browser, say exit.  Otherwise, please choose one of th
<enumerate/></prompt>
<option value="blk3">Your Location is: </option>
<option value="blk5">The requested expert </option>
<option value="exit">Exit </option>
<filled><if cond="main == 'exit'"><exit/><elseif cond="main == 'blk3'"></elseif>
<goto nextitem="blk3"></goto>
<elseif cond="main == 'blk5'"></elseif>
<goto nextitem="blk5"></goto>
</if>
</filled>
</field>
<block><goto next="#start"></goto>
</block>
<block name="blk3">Your Location is: Address 11000 Regency Parkway, Cary, NC, 27513. Country
</block>
<block cond="true" name="blk4"><goto next="#start"></goto>
</block>
<block name="blk5">The requested expert is Name YOSHIMURA MASATOSHI Phone Number 2890 Job DES
</block>
</form>
</vxml>
```

Figure 12-9 Request Viewer showing transcoded content (VoiceXML)

Part 4

PDA applications

In this part of the rebook, we introduce and describe sample scenarios you can use to develop PDA applications that can be used to run in a WebSphere Everyplace Server (WES) environment using WebSphere Everyplace Server components such as IBM Mobile Connect (a component of WebSphere Everyplace Server Enable Offering) or Everyplace Synchronization Manager (a component of WebSphere Everyplace Server Service Provider Offering) using a sample DB2 Everyplace application built with the Mobile Application Builder. Transaction messaging applications are also described using MQSeries Everyplace, a component of WebSphere Everyplace Server, to provide a once-only assured delivery of messages.

13

Data synchronization for enterprise applications

This chapter provides an overview and implementation scenarios of IBM Mobile Connect software package and discusses the differences with the Enterprise Synchronization Manager. We will focus primarily on the Enable Offering (EO) version of WebSphere Everyplace that runs on the Windows NT/2000 platforms. The other version of WebSphere Everyplace is the Service Provider Offering (SPO) and runs on the AIX and Solaris platforms (not the Solaris x86 architecture).

In this chapter, we introduce IBM Mobile Connect Version 2.5.1, a sample DB2 Everyplace (DB2e) application built with the Mobile Application Builder, show configuration examples and discuss issues which may aride when synchronizing data from outside the enterprise (for example, wireless or dial-up). We will discuss the tools used in development and how you may introduce them into your own environment.

Note: IBM Mobile Connect (IMC) Version 2.5.1 is an optional component in IBM WebSphere Everyplace Server Enable Offering Version 1.1.

Note: Everyplace Synchronization Manager (ESM) Version 1.1.3 is a component in IBM WebSphere Everyplace Server Service Provider Offering Version 2.1. In addition, ESM Version 2.1 can also be obtained using WES V2.1 APAR.

13.1 Overview of IBM Mobile Connect (IMC)

IBM Mobile Connect, or IMC, is a data synchronization tool with a server, proxy and client code portions. The purpose of implementing IMC in an enterprise is to allow your mobile users of Palm, Windows CE/Pocket PC and EPOC OS devices access to their e-mail, calendars and enterprise data.

When we refer to enterprise data, we refer to the databases that house your important corporate data: sales figures, Human Resources information or just about anything else your company uses to survive. IMC enables you to connect to your back-end database servers, retrieve specific data from specific rows in the database and filter it based on criteria you determine, or by user ID.

How do you implement this into a usable format for your mobile users? IMC will let us port our back-end to various formats depending on the platform to which we are sending data. For example, we can take data from a DB2 database and create a DB2 Everyplace Database on the fly. While we can use IMC for database synchronization, we can also use to install files and backup/restore our handheld devices.

We will demonstrate installation of IMC Server and clients, backup/restore actions, installation actions and database synchronization. We will demonstrate data exchange with an application built using the Mobile Application Builder. This Palm OS application will consist of a list of employees names and their phone extensions: a phone book.

13.1.1 Implementation scenario

Basically, we have our company phone book stored in a table that resides in DB2 Universal Database (UDB) and we want to send that data to our users in a mobile format on their Palm OS and PocketPC devices. All of our users have the ability to connect to the corporate network either by a modem connected to their mobile device, by a cellular phone connected via cable to their mobile device or by direct connection via serial/USB connection to a PC.

Knowledge of database connectivity and server OS familiarity is assumed; also, several other products are referenced (DB2e, Mobile App. Builder). Installation routines are not necessarily covered for those products.

Figure 13-1 illustrates that we can take connections from various sources, integrated cellular devices, locally connected devices and devices that either connect with a land-line modem or wireless modem, and exchange data with them.

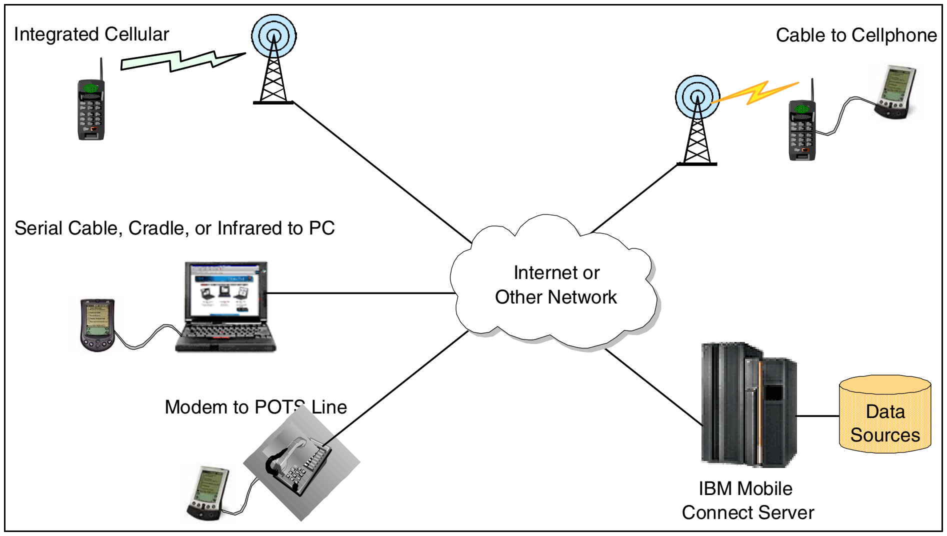

Integrated Cellular
Cable to Cellphone
Serial Cable, Cradle, or Infrared to PC
Internet or Other Network
Modem to POTS Line
Data Sources
IBM Mobile Connect Server

Figure 13-1 IBM Mobile Connect device connectivity

These devices connect to a corporate network using a VPN-type dialer that is supplied by your company; if you connect directly to the Internet, you should have a VPN package that will create the tunnel needed to get to your corporate network.

13.1.2 Enable Offering and Service Provider Offering

The Enable Offering is a Windows NT- and Windows 2000-based set of applications. The Enable Offering uses the IBM Mobile Connect Version 2.5.1 software package for synchronization of data.

The Service Provider Offering of WebSphere Everyplace runs under the AIX and Solaris operating systems. The SPO uses a different version of the code. The actual code is a direct port of the Windows version using a product called MainWin, which preserves the native API calls. The SPO runs a back-level version of the Synchronization Manager code that is patched to include many of the EO version capabilities.

There are differences between the functionality of EO and SPO; Table 13-1 illustrates this for us.

Table 13-1 ESM and IMC in WebSphere Everyplace Server offerings

Function	**Service Provider Offering - ESM 2.1 (WES 2.1 APAR)**	**Enablement Offering - IMC 2.5**
Lotus Notes	Yes	Yes
MS Exchange	Yes	Yes
ODBC Sync.	Yes	Yes
LDAP	Yes - Authorization Only	Yes - Authorization and Sync.
IMAP4	No	Yes
SMTP	No	Yes
WCAP/iCal	No	Yes

In addition to the differences above, there is an additional step when connecting through the Wireless Gateway. The Wireless Gateway has a separate client that authenticates against the LDAP servers and, once that connection is established, you can start the Connect Client which will require another user ID and password. There can be at least three separate logins when connecting through the Wireless Gateway: ISP sign-on, Wireless Gateway and the Connect Client.

IBM Mobile Connect is also a stand-alone application and, because of this, you do not have to go through the Wireless Gateway. However, unless you have a VPN type dialer, you will have to place the IMC server in a DMZ area and then open ports for database, file and mail/calendar replication.

The scenarios listed below were tested with a VPN-type dialer that connected to our ISP and established the secure connection to our corporate network. Limited testing was done by placing an IMC server behind a hardware firewall, opening the required incoming port and then connecting with our normal ISP to the Internet and using the Connect Client; this solution worked, but was vulnerable to DoS and other attacks.

13.2 Installing IBM Mobile Connect

IBM Mobile Connect is a Windows NT and Windows 2000 application and must be installed with a user ID that has Administrator equivalent roles on the server. Proceed with the following steps to install IBM Mobile Connect:

1. Locate and insert your IMC CD in to the CD-ROM drive on your server.
2. Locate and double-click **setup.exe** in the IMC CD-ROM. The following window (see Figure 13-2) will be displayed:

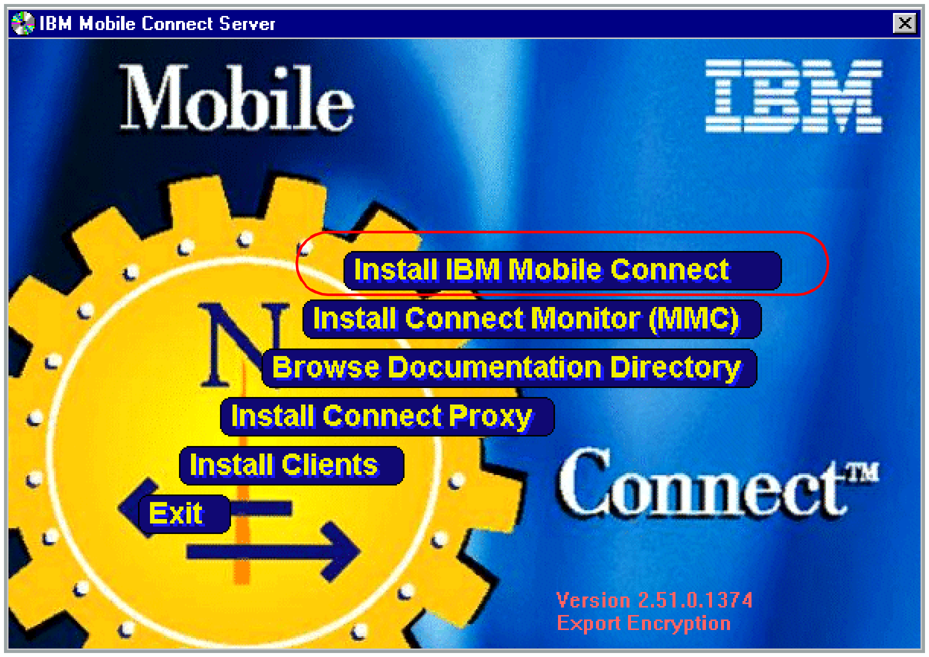
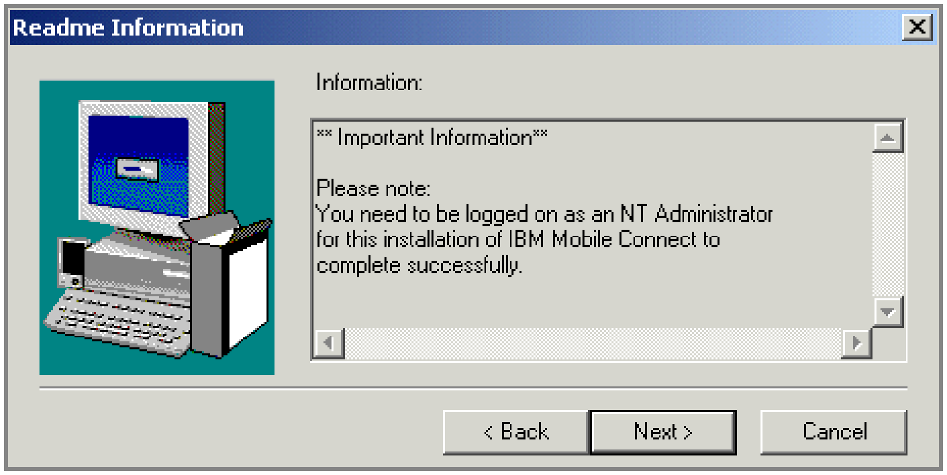

IBM Mobile Connect Server
Mobile
IBM
Install IBM Mobile Connect
Install Connect Monitor (MMC)
Browse Documentation Directory
Install Connect Proxy
Install Clients
Exit
Connect™
Version 2.51.0.1374
Export Encryption

Figure 13-2 IBM Mobile Connect installation

3. Select **Install IBM Mobile Connect** by clicking the first item in the list. The Install Shield setup will begin in about 30 seconds. You are presented with the Welcome and IMC License Agreement windows. You are also presented with a reminder that you must be logged on as an Administrator or equivalent; this window (see Figure 13-3) is displayed no matter who you are currently logged in as.


Readme Information

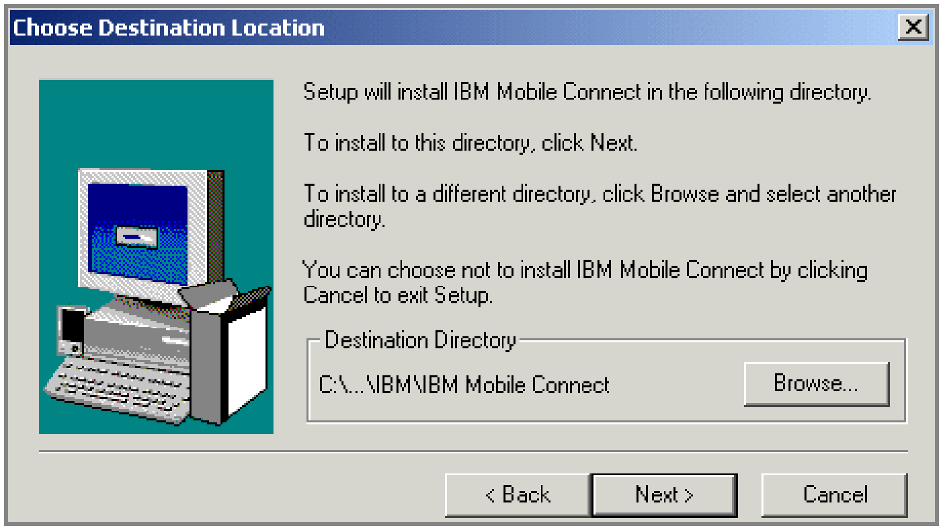
Information:

** Important Information**

Please note:
You need to be logged on as an NT Administrator
for this installation of IBM Mobile Connect to
complete successfully.

Figure 13-3 IMC installation: important information

4. Next, you will choose the installation path for IMC; the default is C:\Program Files\IBM\IBM Mobile Connect, as illustrated in Figure 13-4.


Choose Destination Location

Setup will install IBM Mobile Connect in the following directory.

To install to this directory, click Next.

To install to a different directory, click Browse and select another directory.

You can choose not to install IBM Mobile Connect by clicking Cancel to exit Setup.

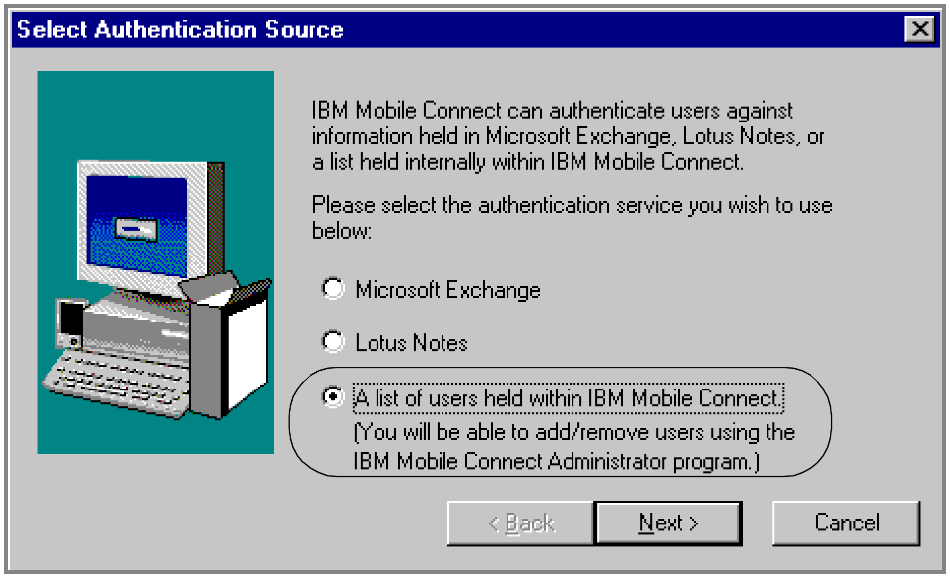
Destination Directory
C:\...\IBM\IBM Mobile Connect

Figure 13-4 IMC installation: target directory

5. You will then be shown the Application Program Group where IMC will reside (the default is IBM Mobile Connect) and a confirmation of everything up to this point. This is the last step before file copying begins.

6. Security and user IDs will be created on an individual basis for this example; select **A list of user will be...** to create a user and assign the associated rights and actions.

   It is possible to use existing Lotus Notes or Microsoft Exchange authentication; you will need to engage your Notes or Exchange administrators to gain access to specific ID files and/or server accounts (see Figure 13-5).



IBM Mobile Connect can authenticate users against information held in Microsoft Exchange, Lotus Notes, or a list held internally within IBM Mobile Connect.

Please select the authentication service you wish to use below:

- ( ) Microsoft Exchange
- ( ) Lotus Notes
- [✓ A list of users held within IBM Mobile Connect.] (You will be able to add/remove users using the IBM Mobile Connect Administrator program.)

Figure 13-5 Authentication sources

7. Click **Create** to bring up the user properties window, then enter a user name and ta password (see Figure 13-6).

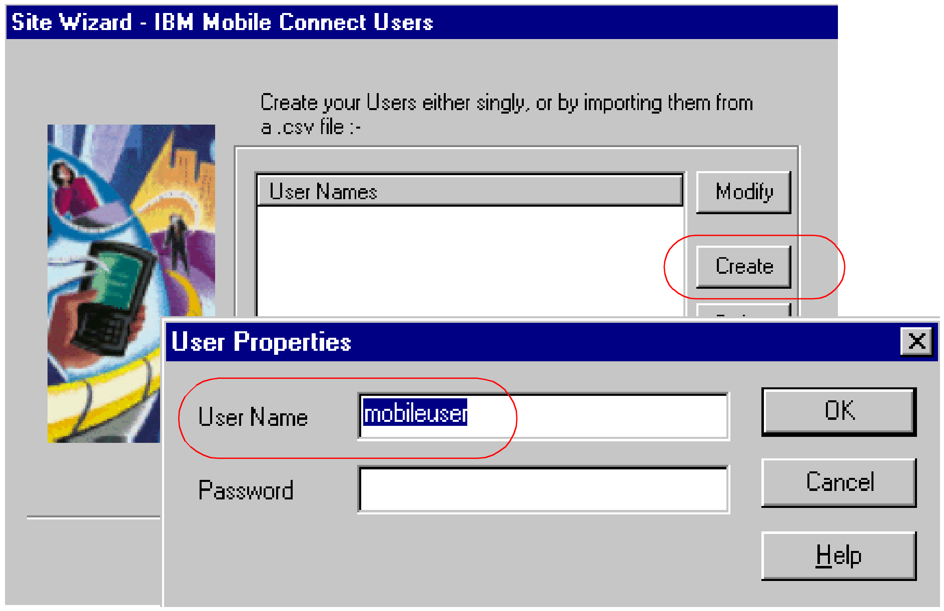

Site Wizard - IBM Mobile Connect Users
Create your Users either singly, or by importing them from a .csv file :-
User Names
Modify
Create
User Properties
User Name
mobileuser
Password
OK
Cancel
Help

Figure 13-6 Entering a user ID and password

8. At this point, you can create other users or proceed. You can always add users later by opening IBM Mobile Connect and stepping through the wizards as shown in Figure 13-7.

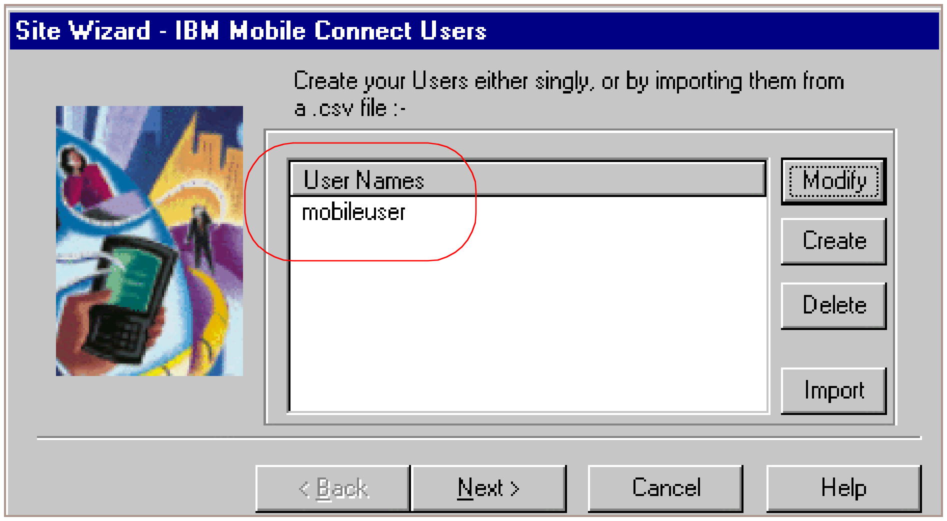
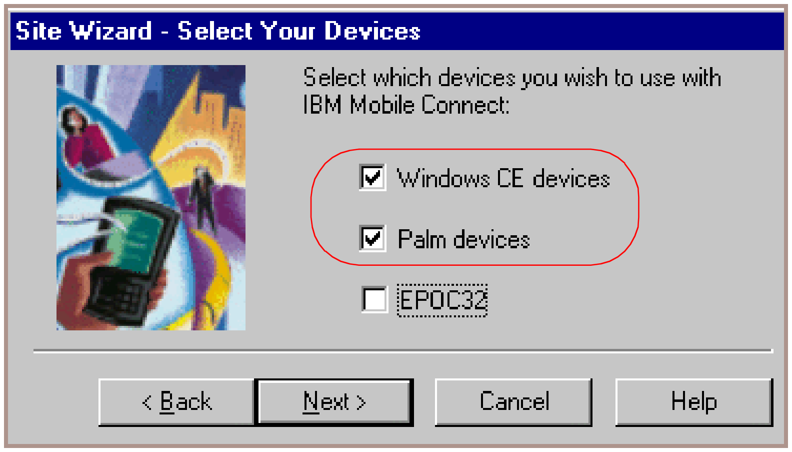

Site Wizard - IBM Mobile Connect Users
Create your Users either singly, or by importing them from a .csv file :-
User Names
mobileuser
Modify
Create
Delete
Import
< Back
Next >
Cancel
Help

Figure 13-7 IBM Mobile Connect users

9. Now you are prompted for the type of devices you want to support. For example, as illustrated in Figure 13-8, in this scenario Windows CE and Palm devices will be used, but the EPOC32 platform will not.



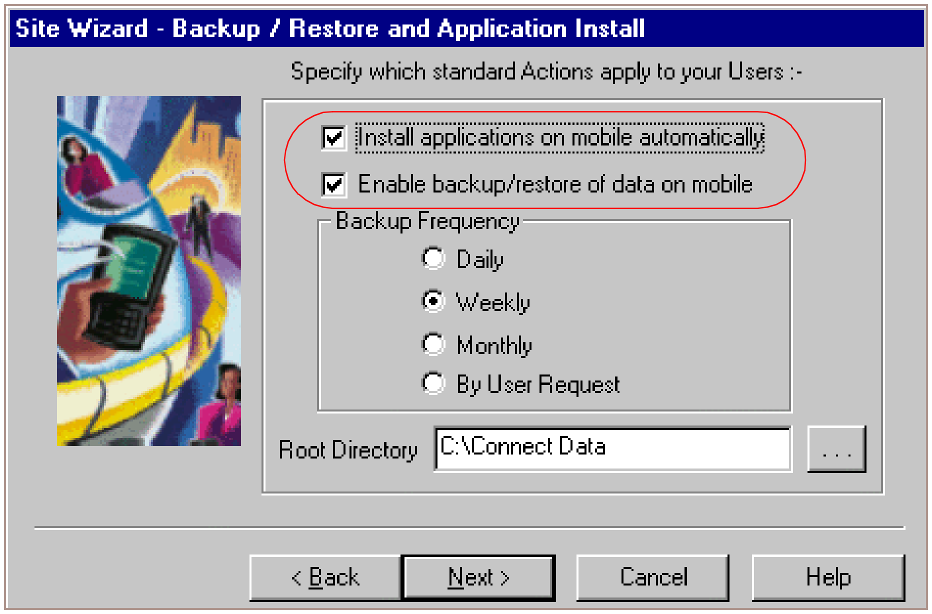
Select which devices you wish to use with IBM Mobile Connect:

- [✓] Windows CE devices
- [✓] Palm devices
- [ ] EPOC32

Figure 13-8 Selecting devices

10. Without much configuration, you can automatically back up a user's handheld device and install applications. For example, in this scenario select the **Install Applications on mobile automatically** feature. This option is not selected by the default, so you will need to select it in order to perform this function. Also by default, the backup/restore feature is enabled for all users on a weekly basis. The restore feature will be demonstrated later in this chapter.

    Note: The default directory for backing up data is C:\Connect\Data, but you can easily change it if required (see Figure 13-9).


Site Wizard - Backup / Restore and Application Install

Specify which standard Actions apply to your Users :-

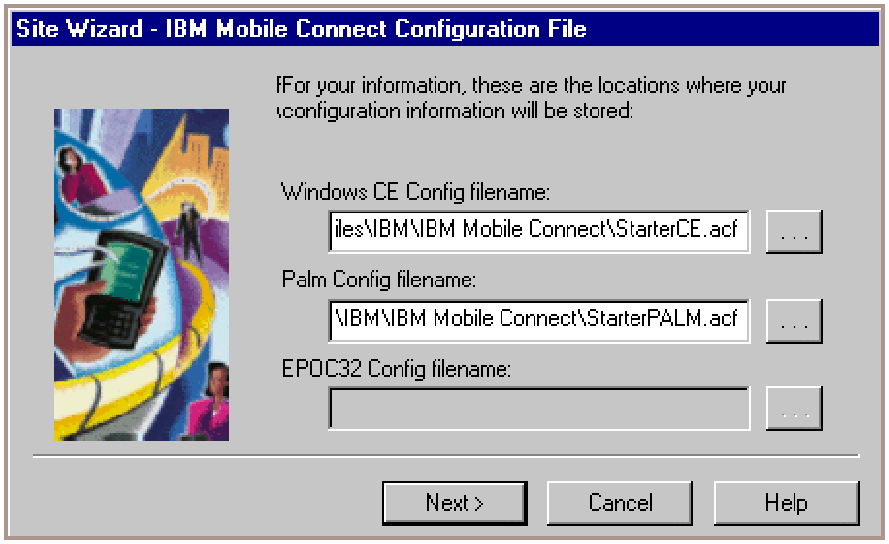
- [✓] Install applications on mobile automatically
- [✓] Enable backup/restore of data on mobile

Backup Frequency: Daily [✓ Weekly] Monthly By User Request

Root Directory: C:\Connect Data

Figure 13-9 Application install and backup/restore options

11.All of the settings that you select for each platform you choose to support are stored in configuration files. The default files are referenced in Figure 13-10.


Site Wizard - IBM Mobile Connect Configuration File

For your information, these are the locations where your configuration information will be stored:

Windows CE Config filename: iles\IBM\IBM Mobile Connect\StarterCE.acf

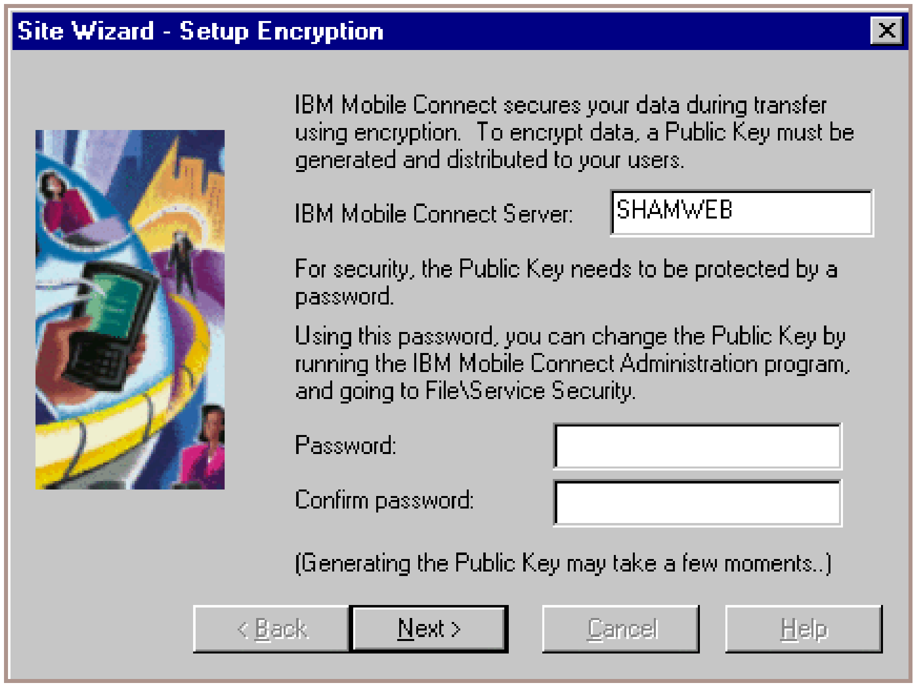
Palm Config filename: \IBM\IBM Mobile Connect\StarterPALM.acf

EPOC32 Config filename:

Figure 13-10 Configuration files

12. Figure 13-11 shows the creation of the Public Key; the content of this window can vary, depending on the level of encryption you are using. Currently, there are three levels of encryption: Strong, Export and None. In the None version, this window is not displayed.

    Enter a password, confirm it and the key will be generated. This key will be required during the setup of the client on your mobile devices, so be certain to keep a reference copy in a safe place.


Site Wizard - Setup Encryption

IBM Mobile Connect secures your data during transfer using encryption. To encrypt data, a Public Key must be generated and distributed to your users.

IBM Mobile Connect Server: SHAMWEB

For security, the Public Key needs to be protected by a password.

Using this password, you can change the Public Key by running the IBM Mobile Connect Administration program, and going to File\Service Security.

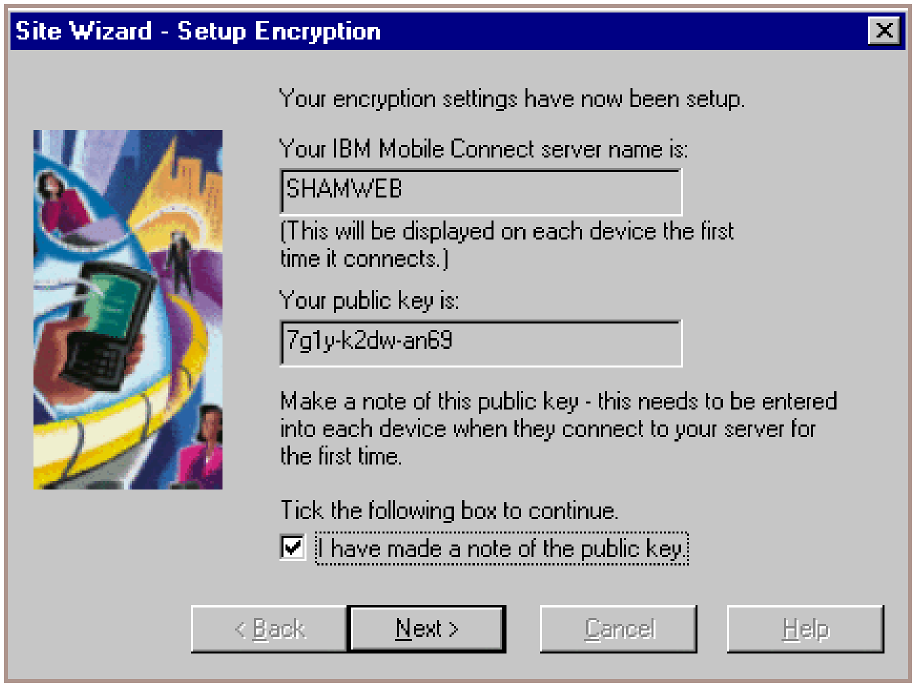
Password:

Confirm password:

(Generating the Public Key may take a few moments..)

Figure 13-11 Generating a Public Key

13. Figure 13-12 shows a completed encryption key creation.You must click the checkbox to state that you have made a reference and backup copy of the key.


Site Wizard - Setup Encryption

Your encryption settings have now been setup.

Your IBM Mobile Connect server name is:

SHAMWEB

(This will be displayed on each device the first time it connects.)

Your public key is:

7g1y-k2dw-an69

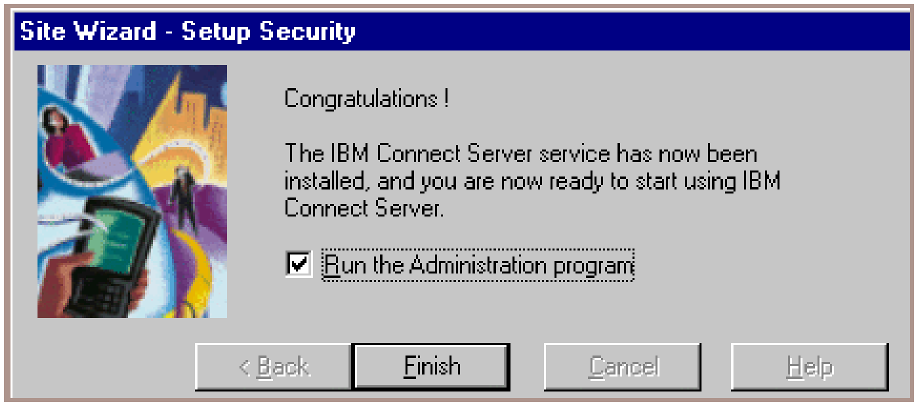
Make a note of this public key - this needs to be entered into each device when they connect to your server for the first time.

Tick the following box to continue.

[✓] I have made a note of the public key.

Figure 13-12 Generated Public Key

14. After the key creation, you are prompted with Congratulations! message stating that you have completed the setup of IBM Mobile Connect. You may choose **Run the Administrator Program** by clicking the checkbox. It is recommended that you reboot the server as soon as possible after installation.


Site Wizard - Setup Security

Congratulations !

The IBM Connect Server service has now been installed, and you are now ready to start using IBM Connect Server.

[✓] Run the Administration program

Figure 13-13 IBM Connect Server installation completed

15. After rebooting the IMC server, locate the IBM Mobile Connect program group in the Start menu and click the **IBM Mobile Connect Administrator** (see Figure 13-14).

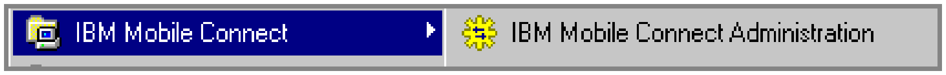


Figure 13-14 IBM Mobile Connect program

16. When IMC Administration starts up, click **File** and then **Service Settings** (see Figure 13-15).

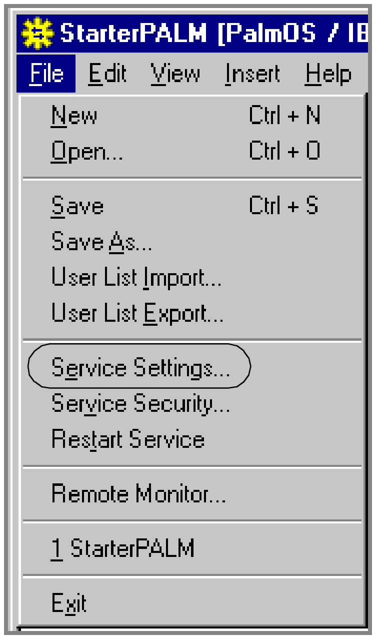
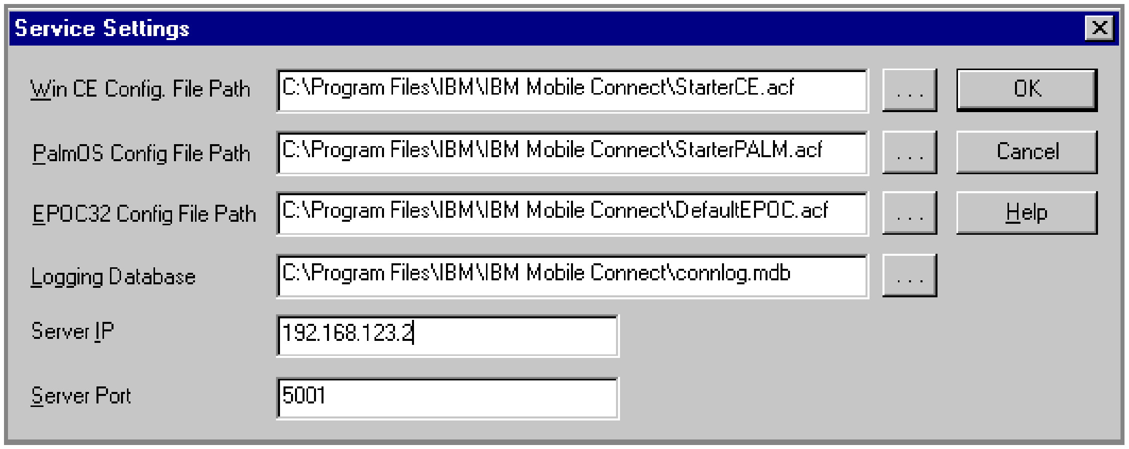


Figure 13-15 IMC Administration: selecting service settings

17. As illustrated in Figure 13-16, the IMC Service Settings dialog appears and here you are prompted with the default configuration files for each handheld operating system, the IP address of the IMC Server and the TCP/IP Port. Enter the IP Address of the IMC server and change the port if needed in your environment. This port must be open from the gateway that will let your clients into your corporate network all the way to the IMC server, and in both directions.

    Click **OK** once you have made your changes; you will be notified that changes will not take effect until you restart the IMC service.


Service Settings

Win CE Config. File Path	C:\Program Files\IBM\IBM Mobile Connect\StarterCE.acf
PalmOS Config File Path	C:\Program Files\IBM\IBM Mobile Connect\StarterPALM.acf
EPOC32 Config File Path	C:\Program Files\IBM\IBM Mobile Connect\DefaultEPOC.acf
Logging Database	C:\Program Files\IBM\IBM Mobile Connect\connlog.mdb
Server IP	192.168.123.2
Server Port	5001

Figure 13-16 Service settings window

18. With the basic configuration changes completed, save the configuration by clicking **File->Save** and restart the service by clicking the **Stoplight** icon.

This completes the installation of IBM Mobile Connect, which included the creation of one user ID and the encryption key. Our next section will begin the configuration of IMC to exchange data between DB2 and the Palm OS.

Very detailed support documentation is included in IBM Mobile Connect, describing further configuration details of the server. We are configuring the server to support data synchronization with a specific application and thus are not configuring every aspect of the product.

13.3 IBM Mobile Connect client installation

In order for your handheld device to be able to synchronize data with IMC, you must install a client piece of code on the device and configure it. You will recall that during setup, we specified that we will support Palm and Windows CE devices.

1. Locate the CD-ROM or file set that you used during the IMC installation; the client code resides here. You will want to make this code accessible from the workstations with which your handheld devices currently synchronize. The client installation is run from the user workstation, not the server.
2. Begin the setup process just as you did the first time. This time, you want to click **Install Clients** (see Figure 13-15).

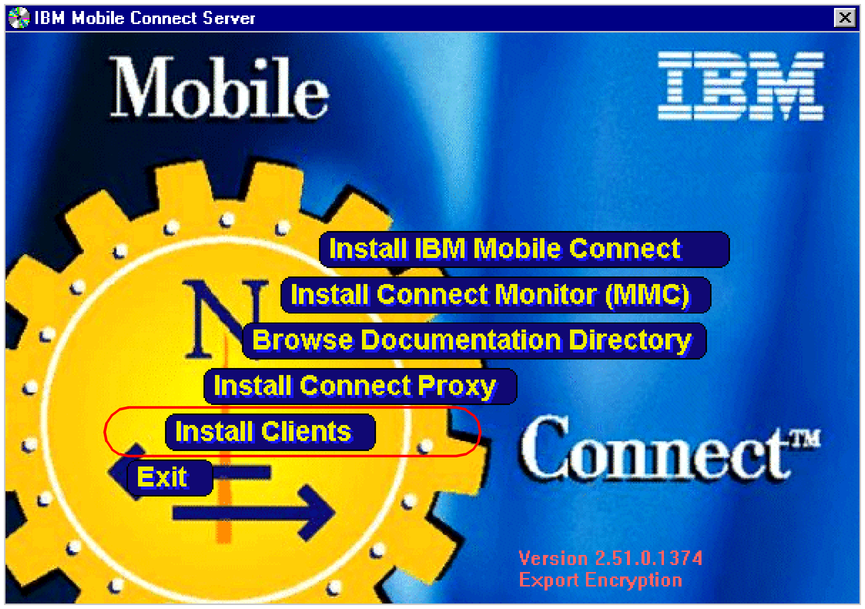
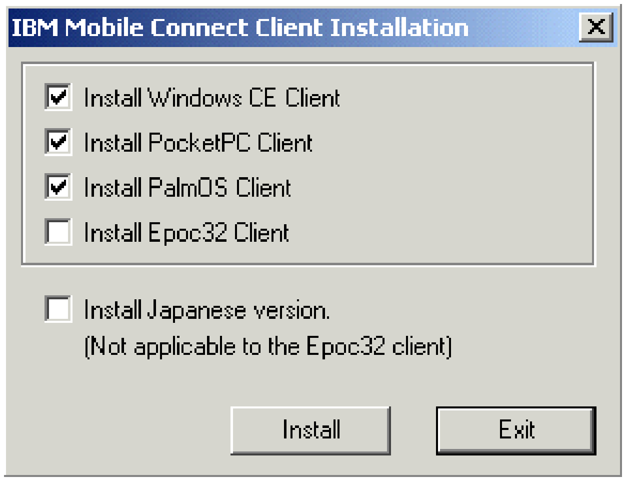

IBM Mobile Connect Server
Mobile
IBM
Install IBM Mobile Connect
Install Connect Monitor (MMC)
Browse Documentation Directory
Install Connect Proxy
Install Clients
Exit
Connect™
Version 2.51.0.1374
Export Encryption

Figure 13-17 Installing clients

3. Since there are multiple types of Windows CE clients (Windows CE and Pocket PC), select the appropriate versions you require and also select the Palm OS client. You also have the option of installing a Japanese version, but we will not select it in this scenario (see Figure 13-18).


IBM Mobile Connect Client Installation

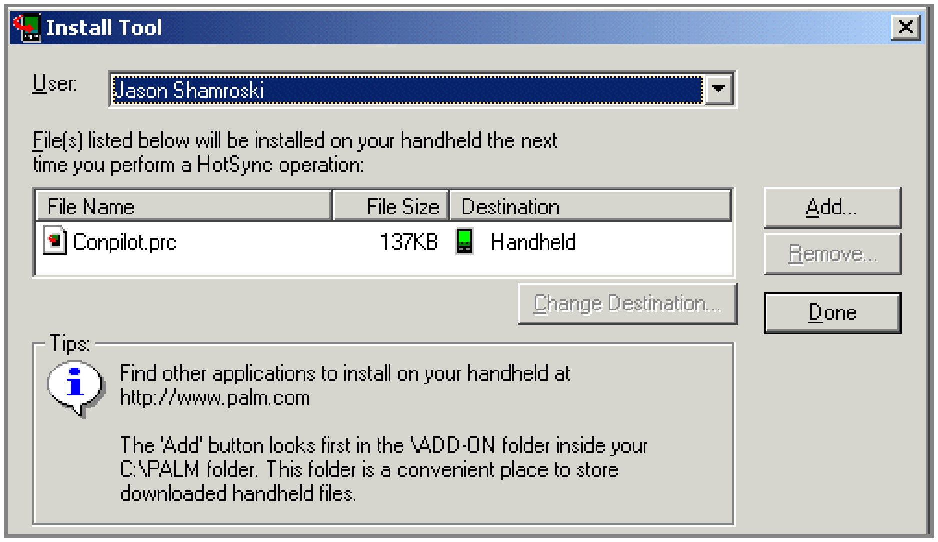
- [✓] Install Windows CE Client
- [✓] Install PocketPC Client
- [✓] Install PalmOS Client
- [ ] Install Epoc32 Client

- [ ] Install Japanese version. (Not applicable to the Epoc32 client)

Figure 13-18 IMC clients - installation

4. Clients will be placed in the Install directory for your specific platform. For instance, for a typical Palm installation, the file is placed in the C:\PALM\%USERNAME%\INSTALL directory and will be ready to be installed on the handheld device on the next Hotsync.

   For example, Figure 13-19 shows a typical installation for the Palm OS client.



File(s) listed below will be installed on your handheld the next time you perform a HotSync operation:

File Name	File Size	Destination
Conpilot.prc	137KB	Handheld

Find other applications to install on your handheld at http://www.palm.com

The 'Add' button looks first in the \ADD-ON folder inside your C:\PALM folder. This folder is a convenient place to store downloaded handheld files.

Figure 13-19 Palm OS client install

5. After the synchronization is complete on your Palm OS device, you will see a new icon simply labeled Connect. The Connect icon represents the entire IBM Mobile Connect client for your Palm OS device (see Figure 13-20).

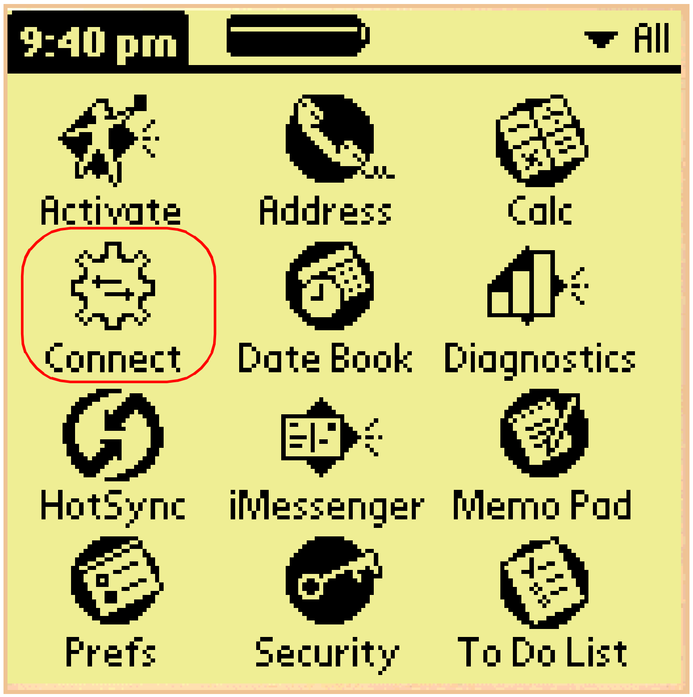


Figure 13-20 Palm OS Connect icon

6. Clicking the **Connect** icon brings you into the IBM Mobile Connect client. By default, the client is set up to attempt to configure a proxy. Proxy configurations are used when connecting from serial or USB cable directly connected to the device. In this scenario, the configuration to use Network will be changed in the next few steps. The network connection is the primary interface for connecting from a wireless modem or dial-up connection (see Figure 13-21).

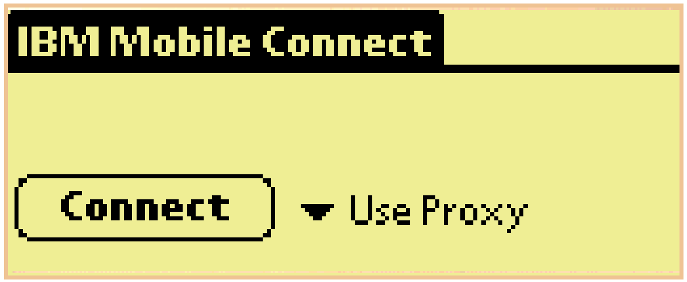


Figure 13-21 Connect option

7. Click the **Menu** icon at the bottom left of your Palm OS device, or use the Graffiti shortcut (see Figure 13-22).

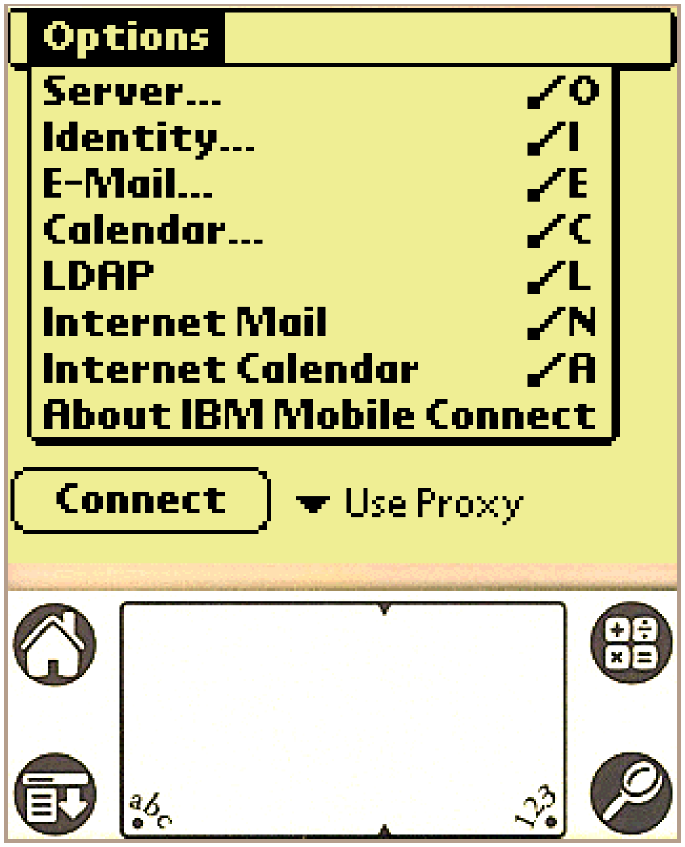
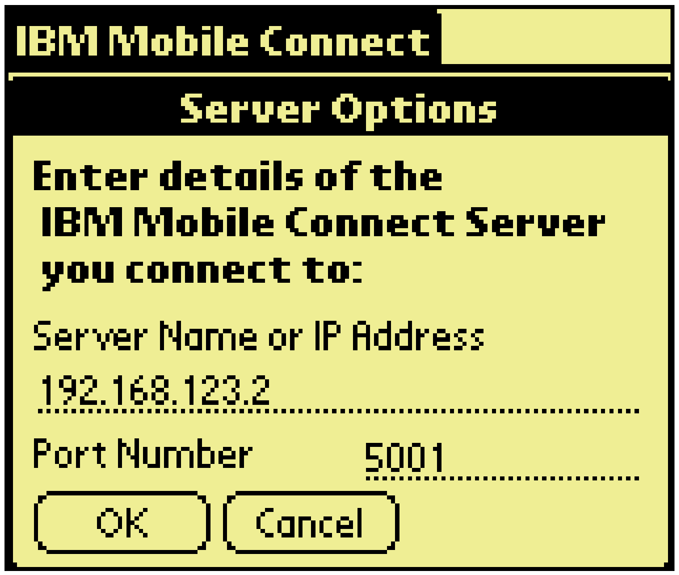


Figure 13-22 Main menu

8. Click **Server** to bring up the Server settings. As shown in Figure 13-23, you will need to provide the IP address of the IMC Server or the registered host name and the TCP port that IMC is listening on (by default this port is 5001).

   In this scenario, the specified IP address is that of the IMC server and the port number is left at the default value. Click **OK** and you will be returned to the IMC client opening window. Additional configuration information for the server side can be found 13.5, “Creating a mobile application with DB2 Everyplace” on page 559.


IBM Mobile Connect

Server Options

Enter details of the IBM Mobile Connect Server you connect to:

Server Name or IP Address 192.168.123.2

Port Number 5001

Figure 13-23 Server options

9. Once returned to the IMC client opening window, click the drop-down menu **Use Proxy** (see Figure 13-24). This brings up the list of available connection methods. In this scenario, the method Network/modem to connect to IMC is used. For further details on the connection methods (Proxy, Hotsync, and so on), refer to the IMC Help Documentation and supporting PDF files for each platform.

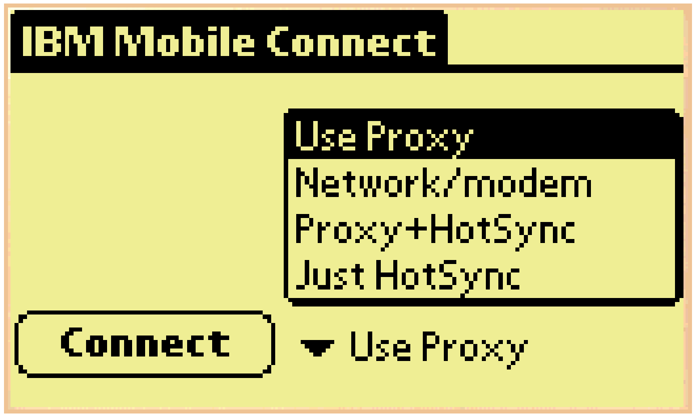
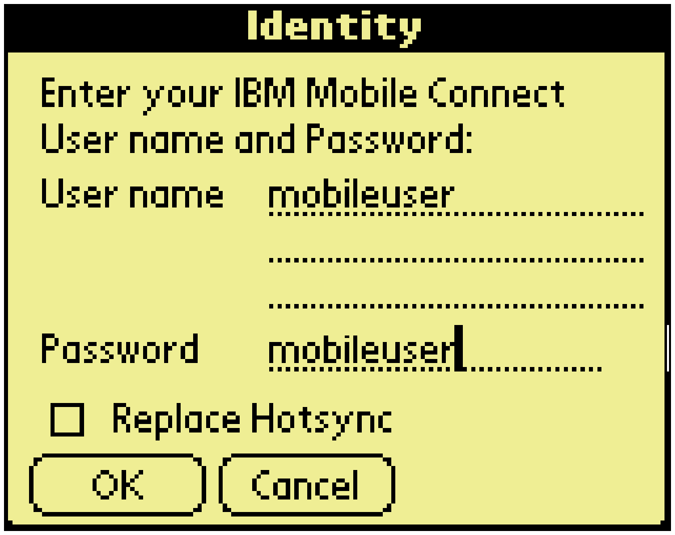


Figure 13-24 Use Proxy option

10. At this point, both the server and the client are set up sufficiently to connect with each other. On your Palm OS device, click **Connect** from within the IMC client. You will be asked for your user identity. Enter the user ID and password that were created during the initial IMC Server install (see Figure 13-25). In this sample scenario, the Replace HotSync option will not be used.

    Note: Passwords are in clear text on the handheld but are encrypted when sent depending on the encryption level of IMC (Export/Strong/None).



Identity

Enter your IBM Mobile Connect User name and Password:

User name mobileuser

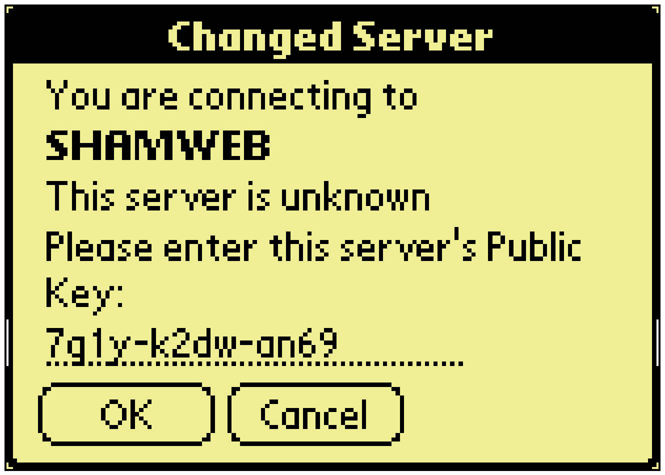
Password mobileuser

☐ Replace Hotsync

Figure 13-25 Entering the user ID and password

11. After clicking **OK**, you will be prompted for the Public Key that was generated during the server installation (see Figure 13-26). You will only be asked for this information the first time you connect to an IMC server; if you connect to another IMC server, you will need to enter the key for that server. If you do not have the key, go to the IBM Mobile Connect Administration tool, click **File -> Service Security** and locate the key. Click **OK** after entering the key and you will receive a confirmation.

    Note: If you attempt to connect to an IMC server of a different encryption strength, various error messages may be returned and no connection will be made. The error messages returned when connecting to a different encryption strength server are not as intuitive as you may like, so be alert to this fact when looking for potential connectivity problems.


Changed Server

You are connecting to

SHAMWEB

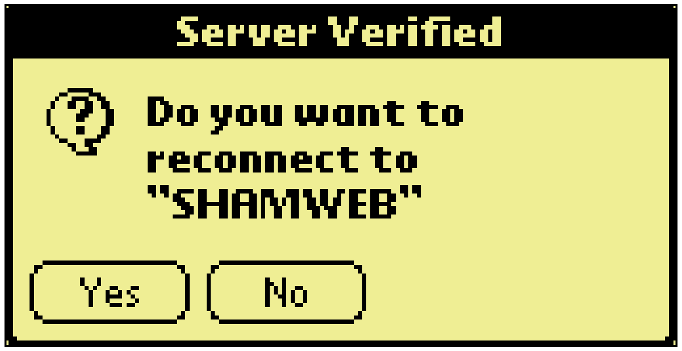
This server is unknown

Please enter this server's Public Key:

7g1y-k2dw-an69

Figure 13-26 Entering server Public Key

12. As illustrated in Figure 13-27, if you wish to reconnect to the server and begin synchronization, click **Yes**. Otherwise, click **No** and you will be returned to the IMC Client.


Server Verified

Do you want to reconnect to "SHAMWEB"

Figure 13-27 Server verification

13. Clicking **Yes** will begin the synchronization process; the default for IMC is to back up a device on its initial connection. In this scenario, no other actions have been defined at this point, so only a backup of the device will occur. When the backup is complete, there will be a new entry on the IMC Client.
14. Should you need to restore data (see Figure 13-28), click the checkbox next to the new item, and on the next synchronization the restore action will be performed. This topic will be covered later in more detail.

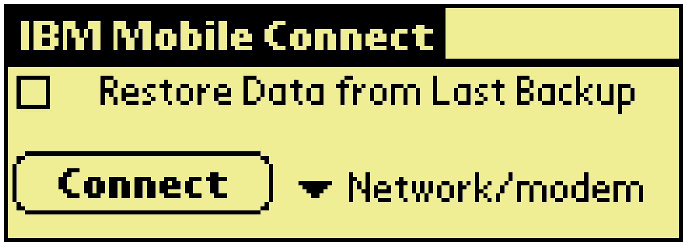


Figure 13-28 Starting the synchronization process

13.4 IBM Mobile Connect actions configuration

Actions or action sets are the instructions that do the work you want performed. After installation of IMC, there are three default action sets to consider:

- Install
- Backup
- Restore

Actions are very powerful and can be created in various ways. There are two wizards (PIM and Replication) that will satisfy the needs of most users. The first wizard is the PIM wizard that is used to set up data replication from MS Exchange and Lotus Notes. The second wizard is the Replication wizard that is used to establish ODBC database replication between handheld applications and back-end databases.

In addition to actions, there are script options; these scripts are written in VBScript and are associated with an IMC event or action set. These VBScript possibilities increase the potential power of IMC exponentially. However, the majority of tasks that you will want to complete are done more simply with the wizards, actions and tags.

Tags are essentially variables that can be passed into events, scripts or actions. An example of a tag is %USER%; this translates to the user ID with which the IMC client is connected.

The default system-defined tags are shown in Table 13-2:

Table 13-2 System defined tags

Tag Name	Definition
USER	User ID with which the Connect client is connected.
PASSWORD	Password which the user entered on the Connect client.
GROUP	IMC Group that the user belongs to. IMC Groups are handy in that each group can be assigned to different actions.
CE_CPU	CPU type for Windows CE/Pocket PC handheld. There is no corresponding value for EPOC and PALM devices. Since these devices have different CPUs, they all require different versions of applications; using this tag allows for a simpler way to point the device to the correct location.

Tags are always preceded and followed by a % sign. Other tags are definable at the system level by double-clicking **Connect Configuration** and selecting the **Tags** tab (see Figure 13-29).

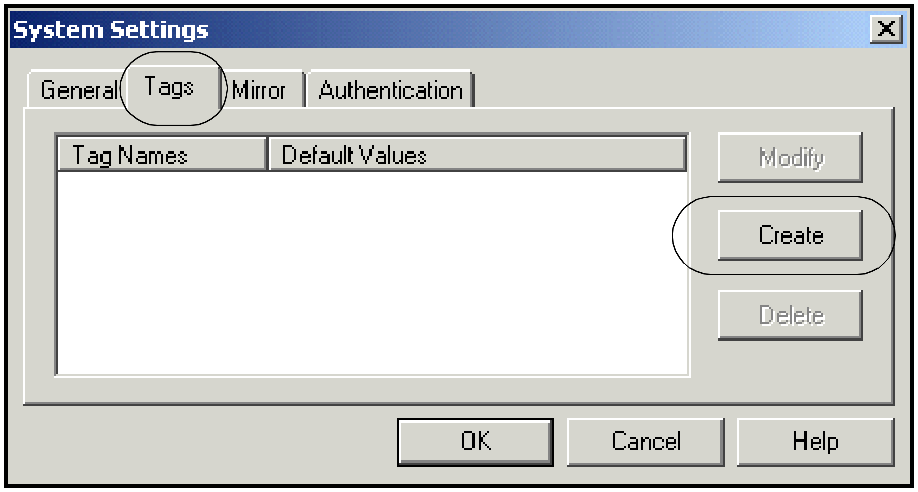
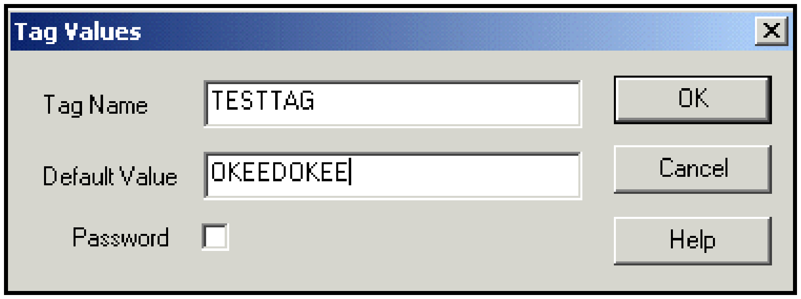


Figure 13-29 System settings

Simply click **Create** to begin the creation of a new tag. The Name field simply requires the name of the new tag and the Default Value field must show the value you want assigned to the tag (see Figure 13-30).



Tag Name TESTTAG
Default Value OKEEDOKEE
Password ☐

Figure 13-30 Tag name and default value

13.4.1 Backup and Restore action configuration

As previously mentioned, the Backup and Restore scenario is enabled during installation of the IMC Server. The frequency of the backup is essential to ensuring that your users' data is safe, but it can be very slow and draining on battery life.

In a simple testing scenario, an 8 MB Palm VIIx was connected using a modem and a connection was established into an IMC server sitting in a DMZ using an AT&T VPN dialer. The Palm had 6 MB of data to be backed up, the results were as follows:

– Connecting via a Palm 33.6 clip-on modem, a connection speed of 26.4 Kbps was obtained; backing up the 6 MB of data took approximately 40 minutes.

– A Wireless Modem/Cellular Modem can vary by phone manufacturer and wireless carrier. For example, using a Motorola Talkabout connected via cable to a Palm at a speed of 14.4 Kbps took over 60 minutes for the 6 MB of data. This scenario also much affected the cellular battery life.

These results show that forcing full backups on a user is not always the most appropriate thing to do, but there is always danger in trusting a user to do the backup himself; however, there is a trade-off between forcing the backup when the battery life on the device is low and letting the user do it when appropriate.

To alter the frequency of Backup, follow these steps:

1. Click **Start -> Programs -> IBM Mobile Connect** and locate the IBM Mobile Connect Administration. Once opened, locate Connect Configuration at the top of the tree on the left hand side, as shown in Figure 13-31.

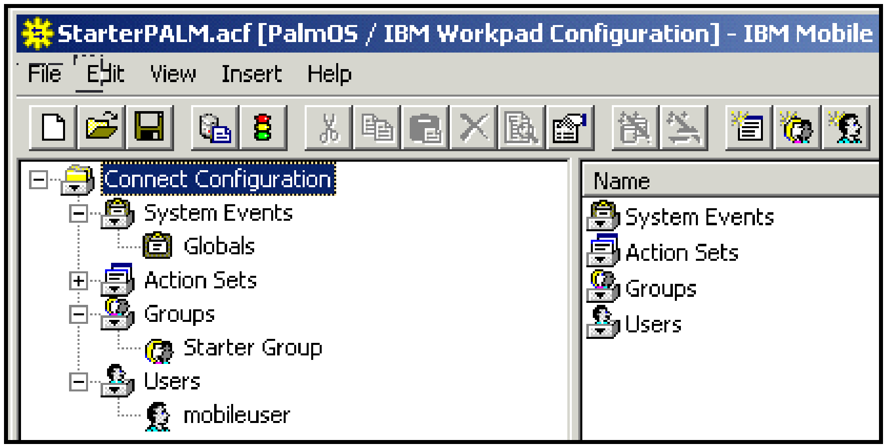
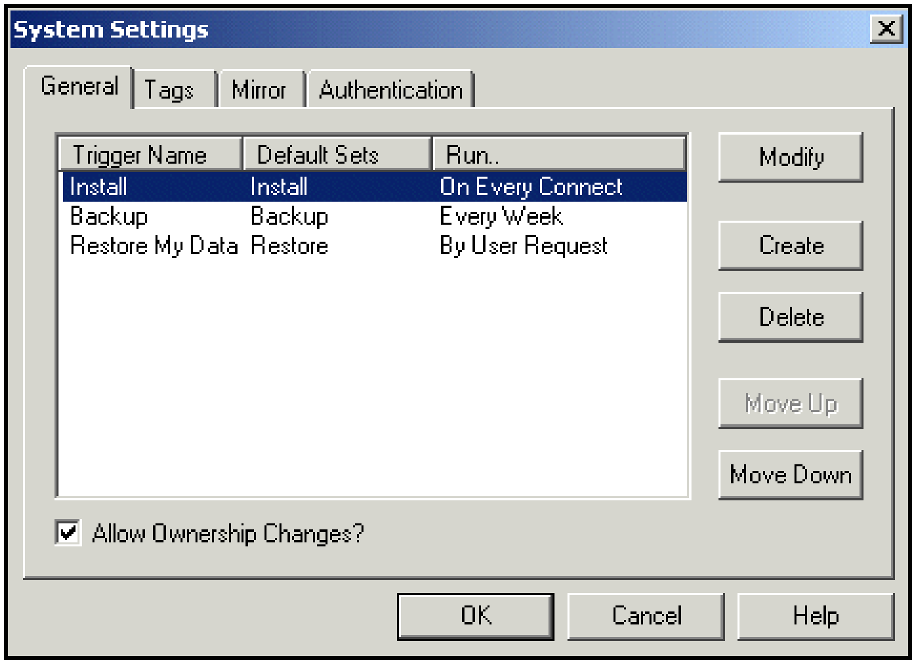


Figure 13-31 Connect Configuration

2. Right-click **Connect Configuration** and select **Properties**. This brings up the System Settings dialog shown in Figure 13-32.

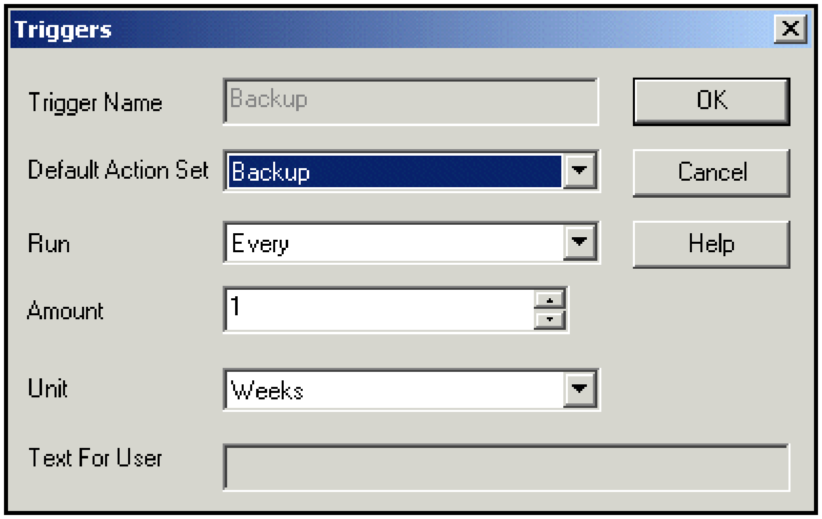


Trigger Name	Default Sets	Run..
Install	Install	On Every Connect
Backup	Backup	Every Week
Restore My Data	Restore	By User Request

☑ Allow Ownership Changes?

Figure 13-32 Systems Settings

3. The order of the triggers listed can be important; for instance, you may not want to restore before installing a new application. The order of the triggers can be changed by highlighting the trigger and then clicking **Move Up** or **Move Down**. The General tab lists the current triggers that are defined. Locate the Backup trigger and double-click it; this brings up the current settings illustrated in Figure 13-33.


Triggers

Trigger Name: Backup

Default Action Set: Backup

Run: Every

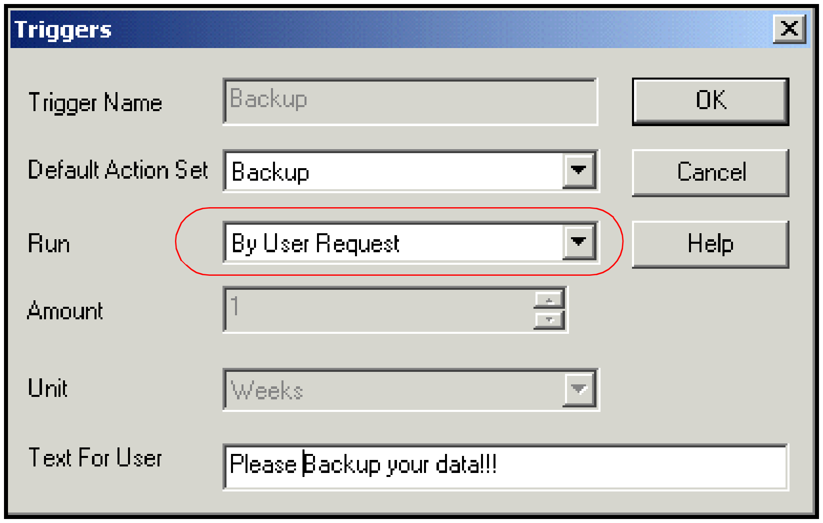
Amount: 1

Unit: Weeks

Text For User:

Figure 13-33 Triggers

4. Now you can change this action to run the By User Request option by clicking the drop-down menu called Run. When you select the option **By User Request**, the text box at the bottom opens up so you can add text to be displayed on the user's handheld device.


Triggers

Trigger Name: Backup

Default Action Set: Backup

Run: By User Request

Amount: 1

Unit: Weeks

Text For User: Please Backup your data!!!

Figure 13-34 Selecting the By User Request option

Click **OK** to close the Triggers dialog and then **OK** again to close the System Settings window. Click the **Disk** icon to save the configuration and then the **Stoplight** icon to restart the IMC Server service.

Now we can connect with our handheld, using the same procedures used the first time, except that now we do not need to enter an encryption key.

After your connect is finished, your IMC Client window should look as illustrated in Figure 13-35.

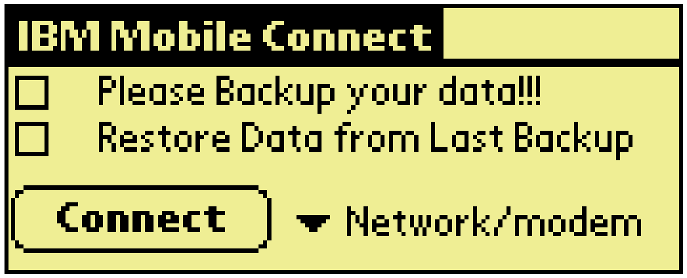
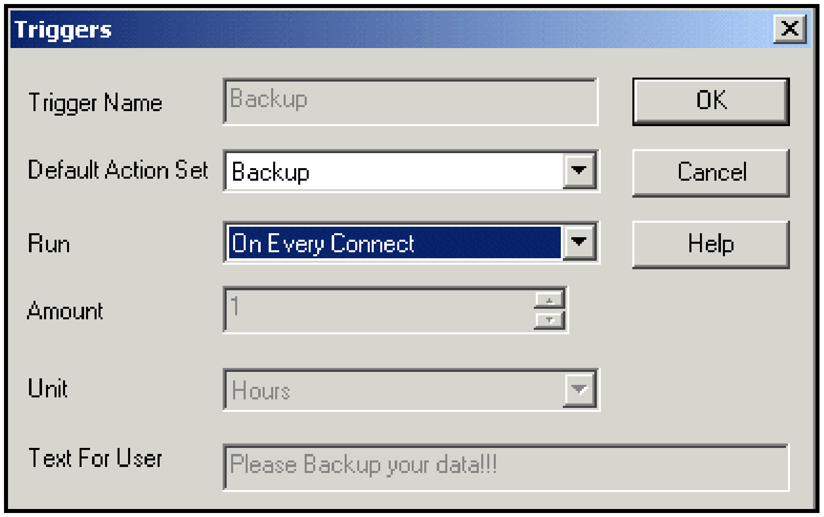


Figure 13-35 Backing up your data

Now to back up the data, you must simply click the checkbox called **Please Backup your data!!!** prior to connecting.

If you go back to the System Settings window and open up the Backup trigger properties window, you can see that there are other options for scheduling the Backup; for instance, you could have selected to run the Backup on every connect (not a good scenario, but possible). This selection is shown in Figure 13-36.


Triggers

Trigger Name	Backup
Default Action Set	Backup
Run	On Every Connect
Amount	1
Unit	Hours
Text For User	Please Backup your data!!!

Figure 13-36 Using the On Every Connect option

There are other options as well, such as *On Losing Ownership* or *On Taking Ownership,* which will perform the action when someone other the original user signs on. The changing ownership options are useful when there is one device that could be used by several people; this way, every user's data is always backed up.

Restoring data is as simple as having the client click the box called *Restore data from last backup* and then click **Connect**. The Restore function does not imply a selective restore, but is a complete restore from the last backup. There are no checkpoint restores or incrementals.

13.4.2 Installation of new applications

The default Install action has an Install action listed under it that will install any files in the specified directory to the handheld device. If you are using IMC with a locally attached Palm OS device and you do *not* intend on using the HotSync software, you may try changing the install path on the Install Action to

```
C:\Palm\%User%\Install
```

Changing this setting on the Install action will allow any new applications to interact with the Palm Install Tool as normal, but the actual installation can take place through IBM Mobile Connect. You must make certain that the IMC user ID corresponds to the Palm Desktop user name.

This method is not always appropriate and will not work correctly with Windows CE and Pocket PC platforms. However, it is another way to standardize configurations when using the Palm OS platform.

13.4.3 Database action configuration

We have a DB2 UDB Version 7.1 database running on Windows 2000 SP2. Our goal is to transfer phone book type data from the EMPLOYEE table in the SAMPLE database set up during the DB2 installation to a DB2 Everyplace table on the Palm OS.

Unfortunately, just getting the data there will not allow you to use it, so you will have to build an application with the Mobile Applications Builder; this will be explained in later sections of this chapter.

There are several scenarios in which you can transfer data to an existing Palm OS application, such as the Memo Pad or the existing Address Book. However, in some cases the contents of the existing Memo Pad or Address Book entries will be overwritten.

The Replication Wizard steps you through the general setup of a database replication action. You can utilize this method to set up a replication of DB2 data, since it is the most likely to be used by users who do not write code and need to get database data into a handheld device.

Replication wizard

To start the replication wizard, click **Start->Programs->IBM Mobile Connect-> IBM Mobile Connect Administration.** Once open, expand the trees on the left-hand side, right-click **Action Sets** and select **New Action Set** (see Figure 13-37).

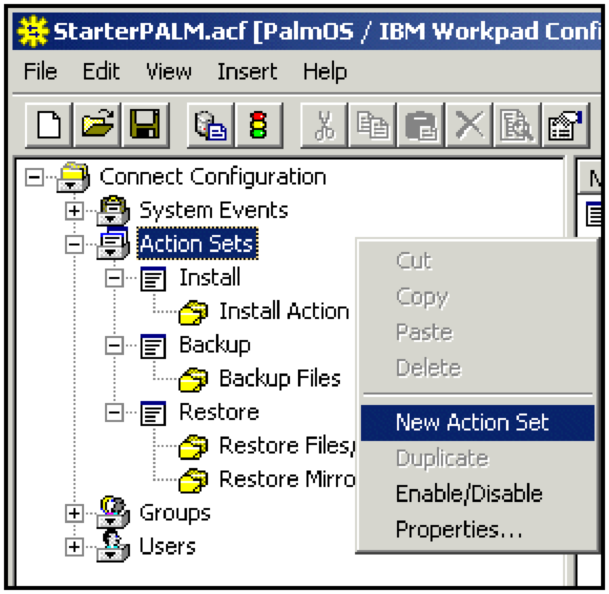
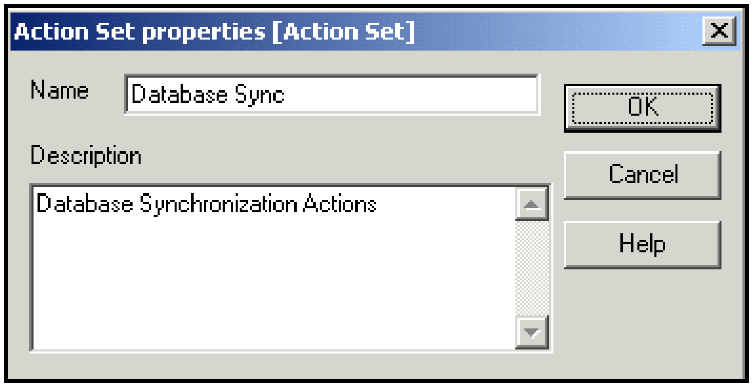


Figure 13-37 Action sets

Give the new action a name and description (see Figure 13-38).


Action Set properties [Action Set]

Name: Database Sync

Description: Database Synchronization Actions

Figure 13-38 Action set properties

Click the new action to highlight it and then click the **Replication Wizard** icon.

The Replication Wizard dialog starts out with the System ODBC data sources that are defined on the server. The ODBC entries are set up manually in the Control Panel area of Windows NT and Windows 2000 servers.

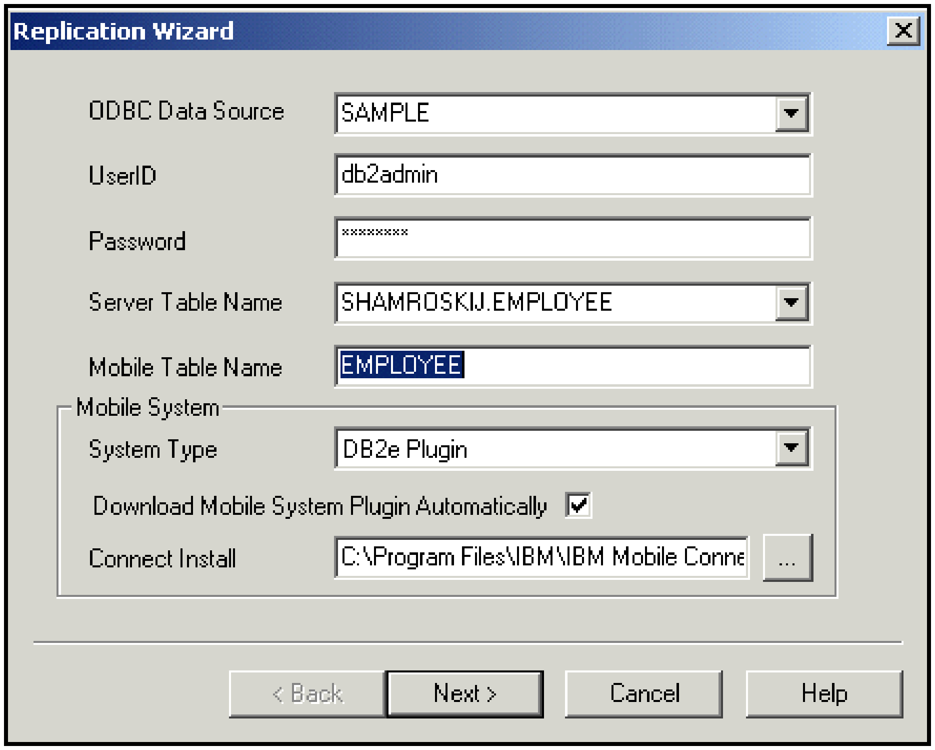
You can see in Figure 13-39 that the ODBC Data Source dialog is a drop-down menu which will list all ODBC entries listed in the System tab of the ODBC/DataSources menu in the Control Panel.

The user ID and password are for database authentication, so this user ID and password must have valid read/write access on the tables you are selecting. At this point, tags will not be used, but you will need to come back later and reconfigure the database action to use a tag in the password field.

When you click the **Server Table Name** field, all of the tables for the data source that the user ID has access to will appear. You can only select one table; if you need to synchronize more than one table, you must run the wizard again and a second action will be created.

The Mobile Table Name will default to the Server Table Name, without the hostname prefix. Note that you must use the same table name and case when creating an application in Mobile Application Builder.

In the Mobile System dialog, you are given the choice of System Type; the only option available is the DB2e Plug-in during the Wizard setup. Once the action is set up, we can go in and change the System Type to other types, such as Memo Pad or Address Book. There is also an option to *Download Mobile System Plug-in Automatically*; this is selected by default and will create a File Install event in the action set where you are currently working. This File Install event will force the download of the DB2e Plug-in library to your handheld devices. The Connect Install dialog must contain the path to the installation of IBM Mobile Connect on your server.


Replication Wizard

ODBC Data Source	SAMPLE
UserID	db2admin
Password	********
Server Table Name	SHAMROSKIJ.EMPLOYEE
Mobile Table Name	EMPLOYEE

Mobile System

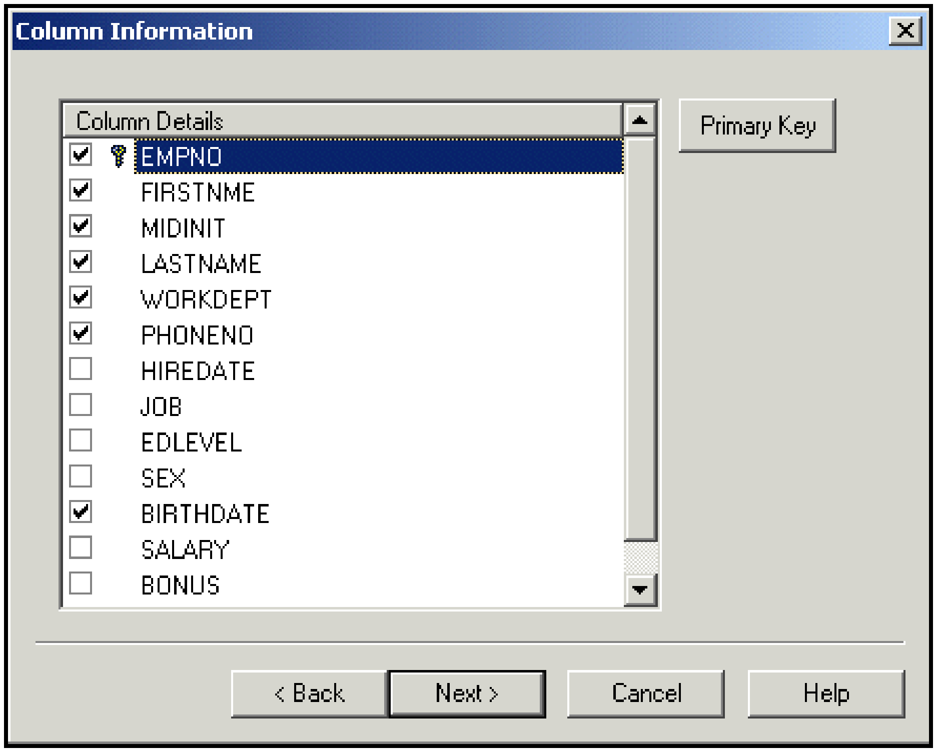
System Type	DB2e Plugin
Download Mobile System Plugin Automatically	✓
Connect Install	C:\Program Files\IBM\IBM Mobile Conne

Figure 13-39 ODBC data source

After clicking **Next**, you are presented with the Column Details from the table you selected. By default, all of the columns are selected for replication. Since the aim is to create a phone book, you do not need to include items such as salary or commission, which might be considered confidential.

For example, you will select only a few of the columns and you must also set a primary key; the primary key for the table is the Employee Number or the EMPNO column. See Figure 13-40 for a list of columns used in this scenario.

Note: Be certain to highlight the EMPNO and click the **Primary Key** button.


Column Information

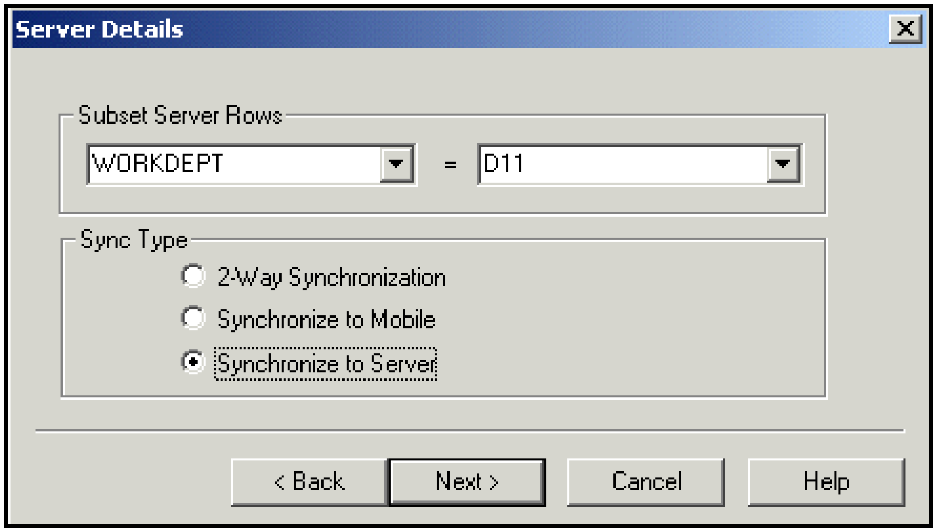
Column Details

✓	EMPNO
✓	FIRSTNME
✓	MIDINIT
✓	LASTNAME
✓	WORKDEPT
✓	PHONENO
☐	HIREDATE
☐	JOB
☐	EDLEVEL
☐	SEX
✓	BIRTHDATE
☐	SALARY
☐	BONUS

Figure 13-40 Database column details

The next dialog shown in Figure 13-41 gives the Server Subset details; a subset is essentially a filter that can be enacted to further restrict flow. For example, you can filter on the WORKDEPT column for contents that only match D11. D11 is the department number for Designers. Using this filter will only allow entries that match WORKDEPT=D11 criteria to be sent to the handheld.

If you wish to implement a multi-column filter or mobile device to a server filter, you must complete the replication wizard and go into the properties of the new Database sync action; this will be covered in a later section. In this sample scenario, the Server to Handheld Sync Type will be selected.


Server Details

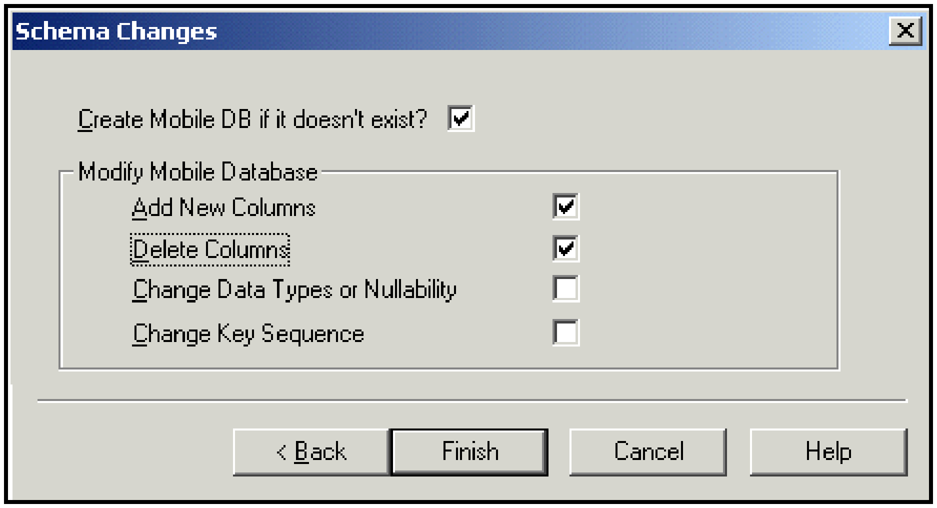
Subset Server Rows: WORKDEPT = D11

Sync Type:
( ) 2-Way Synchronization
( ) Synchronize to Mobile
(•) Synchronize to Server

Figure 13-41 Server details

The next dialog shown in Figure 13-42 demonstrates the ability to allow Schema changes on the mobile database. The first option listed is Create Mobile DB if doesn't exist; this means that if the destination database you have specified does not exist, it will be created. If this option is not selected and the mobile DB does not exist, the action will stop the current action and continue with any remaining actions.

You also have the ability to remove columns and change data type and key sequences. For example, if you want to remove a column of data from the handheld device, you would select **Delete Columns**. Any columns that exist on the Mobile DB that are no longer required for synchronization will be deleted.


Schema Changes

Create Mobile DB if it doesn't exist? ✓

Modify Mobile Database

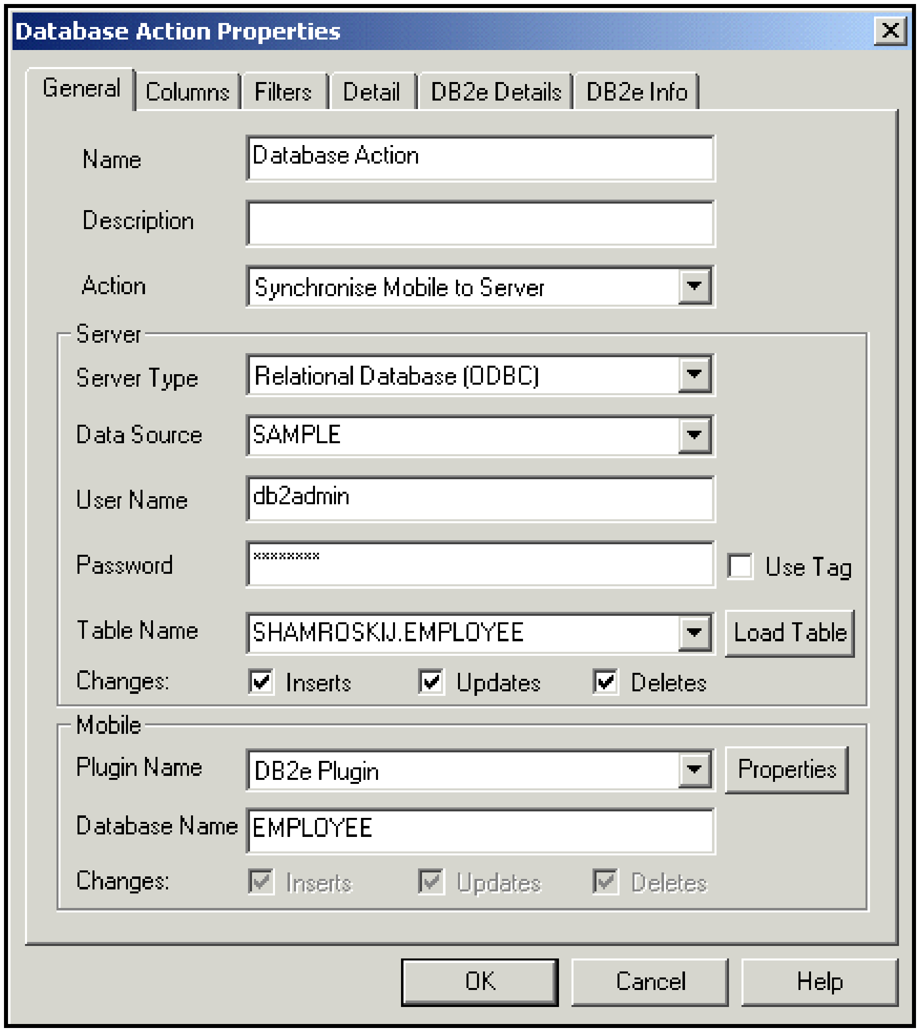
Add New Columns	✓
Delete Columns	✓
Change Data Types or Nullability	
Change Key Sequence	

Figure 13-42 Schema changes

Click **Finish** and you will be returned to the IMC Administration window. Look under the Database Sync action set and you will see two new events. The Download Mobile Plug-in and Database Actions have been created for you.

13.4.4 Changing database action properties

Now that the replication wizard is done, you can look at what was performed. From the IMC administration window, locate the new database action and right-click it, then select **Properties**. The window shown in Figure 13-43 will appear; this represents the results of the wizard configuration that was performed.


Database Action Properties
General | Columns | Filters | Detail | DB2e Details | DB2e Info
OK | Cancel | Help
Load Table
Properties

Name: Database Action
Description:
Action: Synchronise Mobile to Server

Server
Server Type: Relational Database (ODBC)
Data Source: SAMPLE
User Name: db2admin
Password: ********* [ ] Use Tag
Table Name: SHAMROSKIJ.EMPLOYEE
Changes: [✓] Inserts [✓] Updates [✓] Deletes

Mobile
Plugin Name: DB2e Plugin
Database Name: EMPLOYEE
Changes: [✓] Inserts [✓] Updates [✓] Deletes

Figure 13-43 Database action properties

The General tab contains all of the basic settings for database access and type of destination. The Action drop-down menu allows you to specify the type of sync that will be done; this allows you to maintain some additional configuration control over the mobile device. One of the options is Delete Database from Mobile Device; using this option can act as configuration control for the device, keeping databases clean.

Other options include:

- Synchronize Server to Mobile
- Synchronize Mobile to Server
- Truncate Database
- Two-Way database Synchronization

The Server Type drop-down menu lists the possible data sources; these include ODBC, Lotus Notes, MS Exchange and several others: SMTP, LDAP, IMAP and WCAP. Depending on your Server Type choice, the DataSource drop-down menu be different.

The user ID and password are for authentication against your data source; note that you can use the Tag option, allowing for the data to be passed directly from the IMC Connect client.

Note: As previously mentioned, during the replication wizard setup, the Table Name is loaded from the data sources.

The Mobile section allows you to specify the plug-in type that you want to use; this determines where the data will go when it gets to the Mobile Device. During the replication wizard setup, you are only able to select the DB2e plug-in, but now you can select several other plug-ins, as illustrated in Figure 13-44.

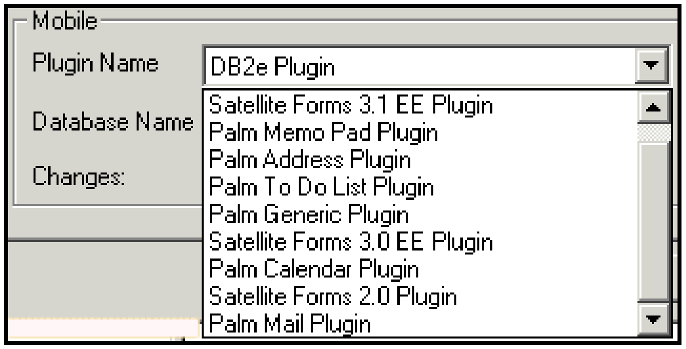
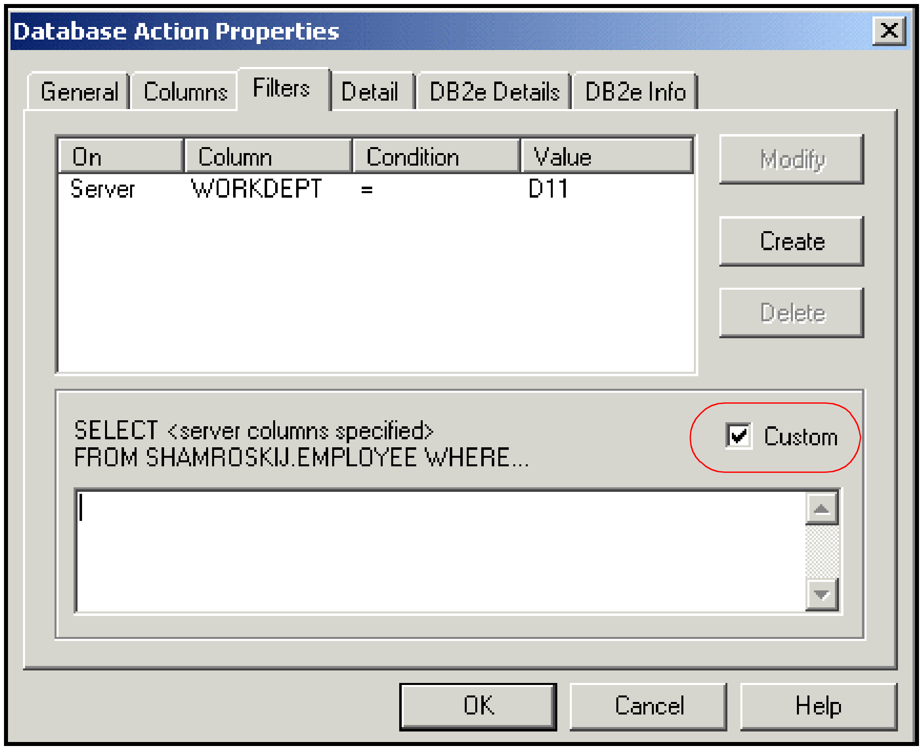


Figure 13-44 Plug-ins

In Figure 13-44 above, you can see the plug-ins available out of the box. The Palm Memo, Address, To Do, Calendar and Mail plug-ins allow synchronization of data directly into those Palm OS applications. The Palm Generic Plug-in allows you to create specific database entries, generate source code and create a Palm DataBase file or PDB file.

You can navigate through the tabs; the first tab, Columns, shows the columns from the SAMPLE DB2 database that will be synchronized in this sample scenario. You can add columns by clicking **Create**, and this will present a dialog that will allow you to create table mappings from the DB2e database to the DB2 server database. This list of columns corresponds to the information on the DB2e Info tab.

The Filters tab shown in Figure 13-45 displays the WORLKDEPT=D11 selection criteria that you specified during setup. Note that there is now a custom area; click the checkbox next to Custom; you can now type your own SQL selection criteria, if required.



On	Column	Condition	Value
Server	WORKDEPT	=	D11

SELECT <server columns specified>
FROM SHAMROSKIJ.EMPLOYEE WHERE...

☑ Custom

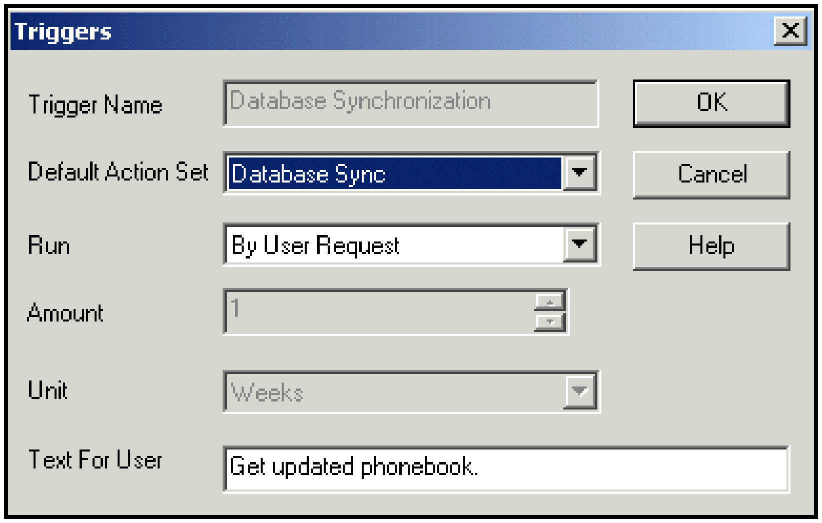
Figure 13-45 Filters

The Details tab allows you to specify a different ODBC connection type, depending on the special requirements of your back-end database. By default, **Default** is selected and works most of the time for most databases.

13.4.5 Creating database synchronization triggers

Before you can begin using the new action set, you need to create a trigger that defines when to run it. In this scenario, a database synchronization action set will run on user requests.

Right click **Connect Configuration** (see Figure 13-31**)** and select **Properties**, then click the **General** tab and click **Create** (see Figure 13-32**)**. This will bring up the dialog to create a new trigger, as shown in Figure 13-46.



Trigger Name: Database Synchronization
Default Action Set: Database Sync
Run: By User Request
Amount: 1
Unit: Weeks
Text For User: Get updated phonebook.

Figure 13-46   Creating a new trigger

Click **OK** and the trigger will be added to the Starter Group list of triggers.

Note: You must save the configuration and restart the IMC Service to ensure the changes will be active at the next connection.

The next time a device connects, the new option will be added to the user's Connect client window. On the second connection attempt, the database will be synchronized, provided DB2e is installed on the Mobile. See section 12.4.5 for a brief overview of the installation of the DB2e client.

13.4.6 DB2 Everyplace client install

Briefly, if you have DB2e installed on a PC/Workstation, the installation of the client is performed by clicking **Start->Programs->IBM DB2 Everyplace-> Install** on the mobile device.

You will be presented with the dialog asking you which clients to install (see Figure 13-47); select the appropriate platform and click **OK**. Depending on the platform, additional setup may be required on the mobile device.

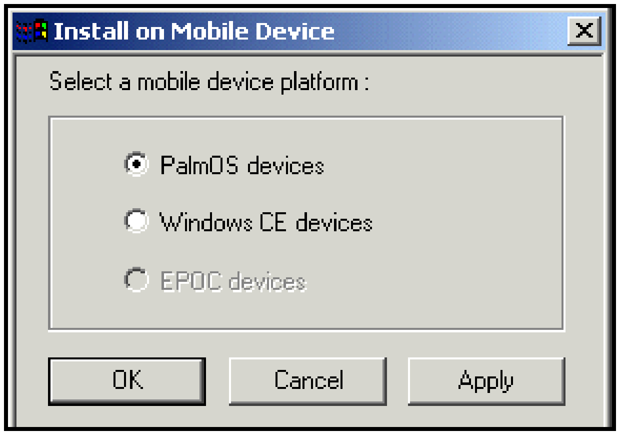

Install on Mobile Device
Select a mobile device platform :
[● PalmOS devices] Windows CE devices EPOC devices
OK
Cancel
Apply

Figure 13-47 Selecting the mobile device platform

For Palm Devices, click **OK** and you are presented with another dialog asking which user gets the client on this PC(see Figure 13-48), whether the color display option is wanted, and which components are to be selected.

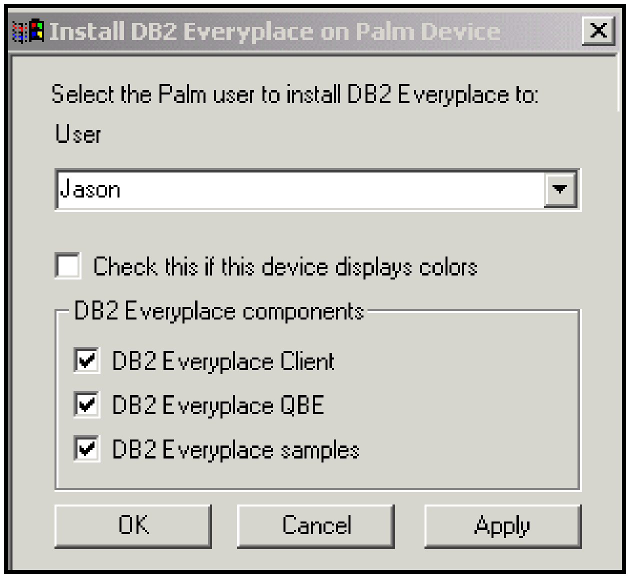
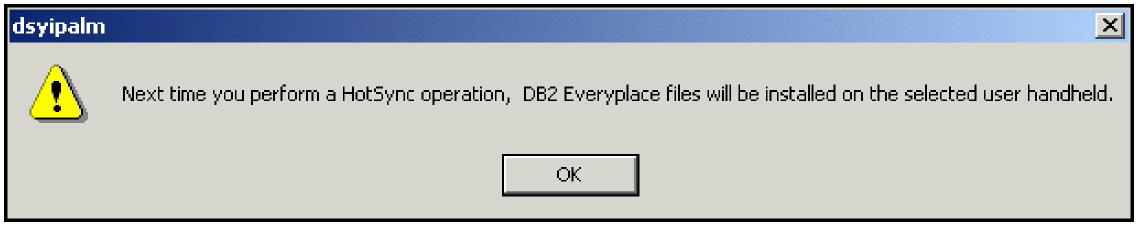

Install DB2 Everyplace on Palm Device
Select the Palm user to install DB2 Everyplace to:
User
Jason
☐ Check this if this device displays colors
DB2 Everyplace components
✓ DB2 Everyplace Client
✓ DB2 Everyplace QBE
✓ DB2 Everyplace samples
OK
Cancel
Apply

Figure 13-48 Specifying a user for Palm devices

Click **OK** and you will receive a confirmation stating that the next time you perform a hotsync operation, the new files will be sent to your mobile device. The message is illustrated in Figure 13-49.



Next time you perform a HotSync operation, DB2 Everyplace files will be installed on the selected user handheld.

Figure 13-49 Hotsync operation message

13.5 Creating a mobile application with DB2 Everyplace

In order to use back-end data once you receive it on your handheld device, you must have an application designed to view it. In this section, the Mobile Application Builder is introduced to create a simple application. Portions of DB2 UBD and DB2 Everyplace will be used in this section as well; these products are well documented and information on their specific functions should be sought in their respective manuals and documents.

IBM's DB2 UDB database server comes with a sample database called SAMPLE; this will be the basis for the data to be retrieved. In this scenario, DB2 Version 7.1 running on a Windows 2000 Service Pack 2 server will be used. Also, default settings have been chosen during installation and after the installation the included Create Sample Database from the First Steps window has been executed.

The DDL information is equivalent to the schema of your database and tables; refer to the Mobile Application Builder documentation and the examples in this chapter for proper formatting information. We will begin with the creation of the sample application by getting to the DDL in a DB2 Version 7.1 database.

In this section, two scenarios are provided for getting the DDL from a DB2 database. The first scenario will involve retrieving the DDL from the DB2 Control Center and the second scenario will do the same using the command line interface (CLI).

13.5.1 Generating DDL using DB2 Control Center

As an example, execute the following steps to generate a DDL using the DB2 Control Center:

1. First, locate your DB2 Command Center and sign on as the administrator for the SAMPLE database, as shown in Figure 13-50.

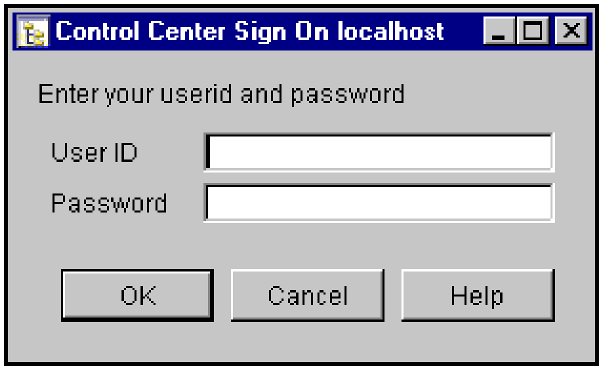

Control Center Sign On localhost
Enter your userid and password
User ID
Password
OK
Cancel
Help

Figure 13-50 Control Center sign-on

2. Locate the SAMPLE database by navigating through the directory trees, as shown in Figure 13-51.

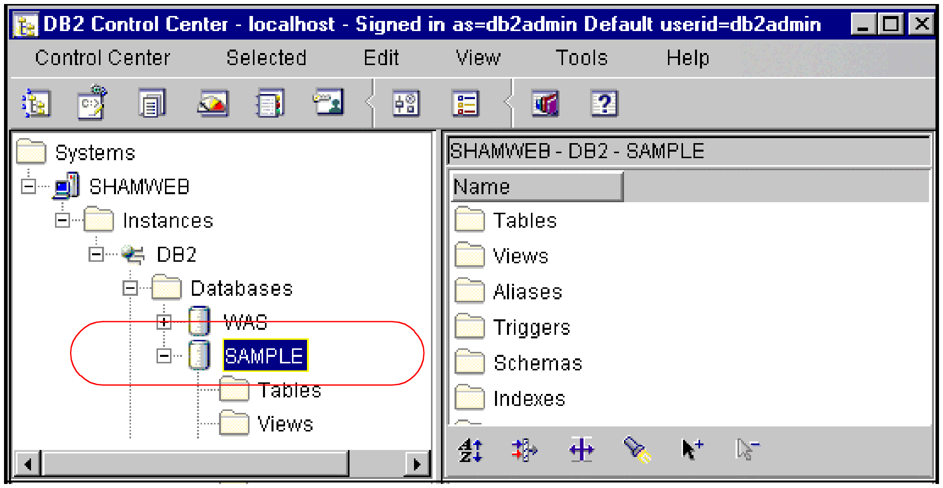


Figure 13-51 Sample database

3. Right-click the **SAMPLE** database and locate **Generate DDL** (see Figure 13-52).

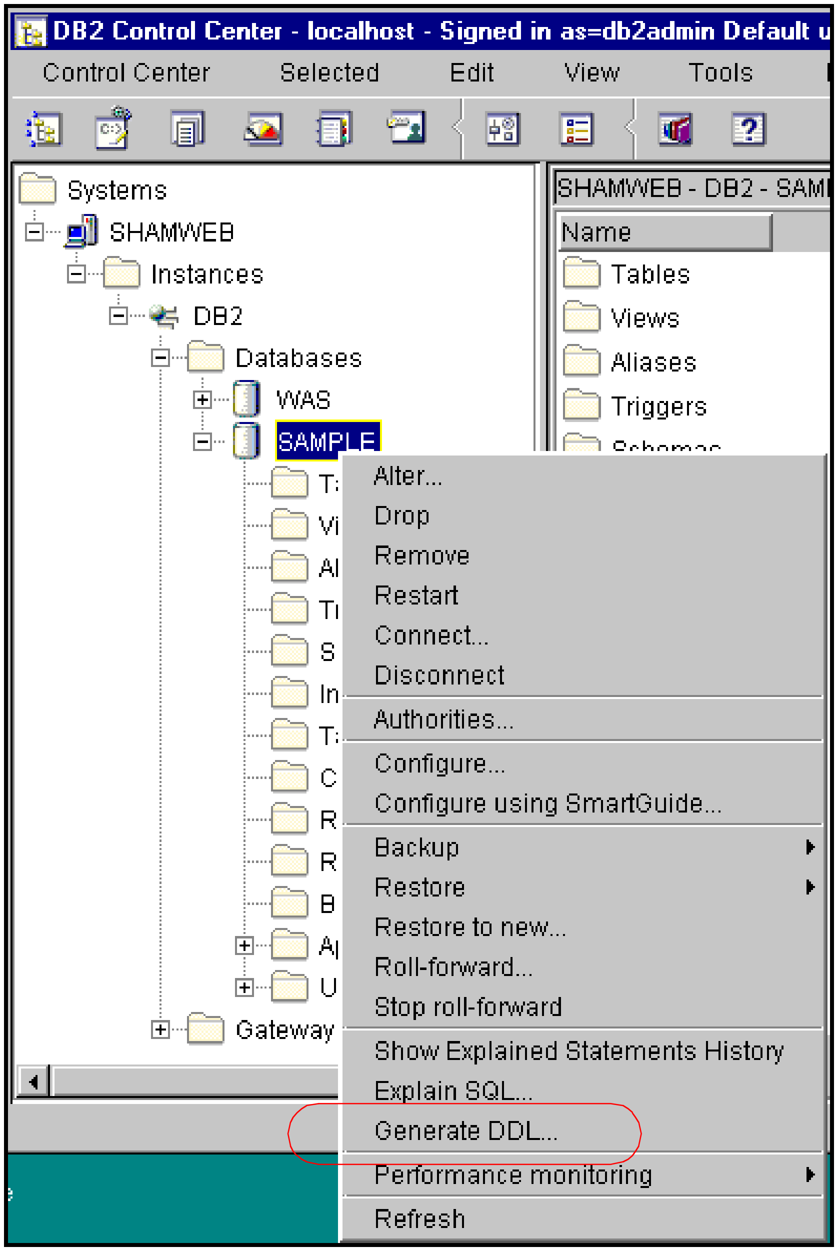
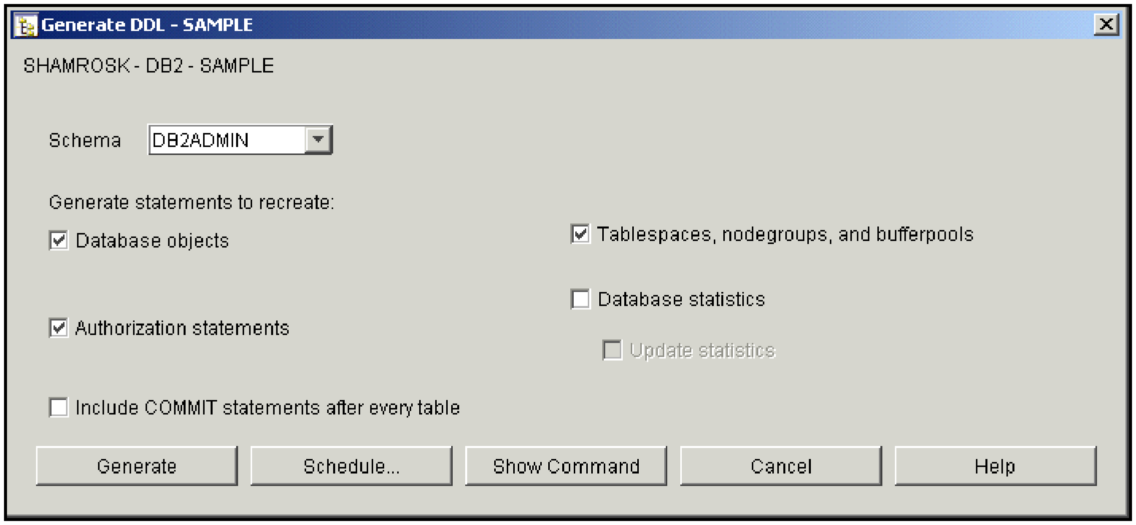


Figure 13-52 Locating Generate DDL

4. Click **Generate DDL** to begin the process. You will be prompted with a window asking for several details; in this scenario, default values will be taken (see Figure 13-53).


Generate DDL - SAMPLE

SHAMROSK - DB2 - SAMPLE

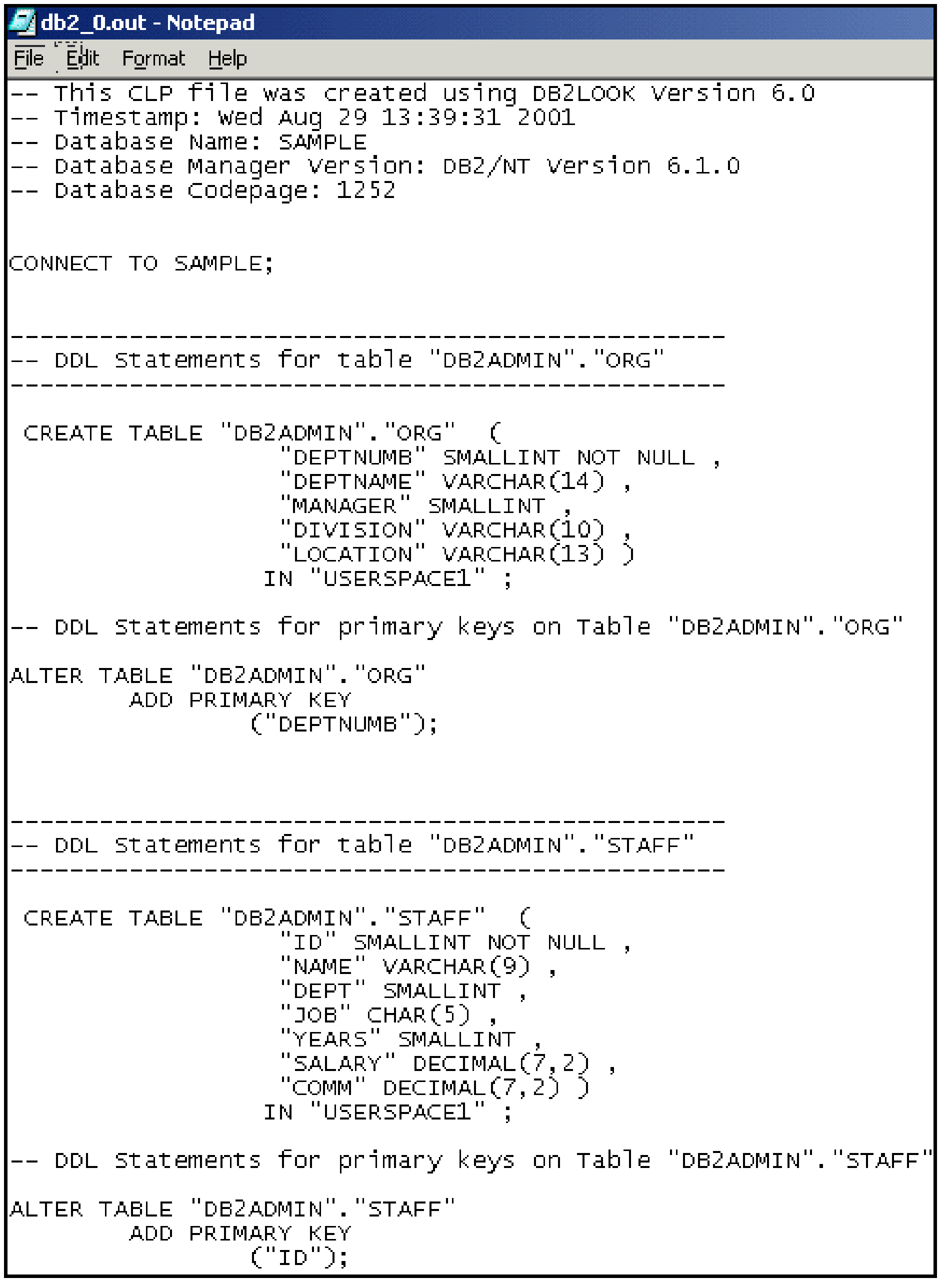
Schema: DB2ADMIN

Generate statements to recreate:

- [✓] Database objects
- [✓] Authorization statements
- [ ] Include COMMIT statements after every table
- [✓] Tablespaces, nodegroups, and bufferpools
- [ ] Database statistics
- [ ] Update statistics

Figure 13-53 Generate DDL window

5. You will be prompted for the user ID and password for the job. The Generate DDL function is actually running the DB2 commands using the internal DB2 job scheduler. The output will take about 15-30 seconds to be written. The DDL is stored in a file, db2_0.out, in the C:\SQLLIB directory on an NT Server. Open this file for viewing. Figure 12-12 shows portions of the generated DDL.



```
-- This CLP file was created using DB2LOOK Version 6.0
-- Timestamp: Wed Aug 29 13:39:31 2001
-- Database Name: SAMPLE
-- Database Manager Version: DB2/NT Version 6.1.0
-- Database Codepage: 1252


CONNECT TO SAMPLE;


------------------------------------------------
-- DDL Statements for table "DB2ADMIN"."ORG"
------------------------------------------------

 CREATE TABLE "DB2ADMIN"."ORG"  (
                  "DEPTNUMB" SMALLINT NOT NULL ,
                  "DEPTNAME" VARCHAR(14) ,
                  "MANAGER" SMALLINT ,
                  "DIVISION" VARCHAR(10) ,
                  "LOCATION" VARCHAR(13) )
                 IN "USERSPACE1" ;

-- DDL Statements for primary keys on Table "DB2ADMIN"."ORG"

ALTER TABLE "DB2ADMIN"."ORG"
        ADD PRIMARY KEY
                ("DEPTNUMB");



------------------------------------------------
-- DDL Statements for table "DB2ADMIN"."STAFF"
------------------------------------------------

 CREATE TABLE "DB2ADMIN"."STAFF"  (
                  "ID" SMALLINT NOT NULL ,
                  "NAME" VARCHAR(9) ,
                  "DEPT" SMALLINT ,
                  "JOB" CHAR(5) ,
                  "YEARS" SMALLINT ,
                  "SALARY" DECIMAL(7,2) ,
                  "COMM" DECIMAL(7,2) )
                 IN "USERSPACE1" ;

-- DDL Statements for primary keys on Table "DB2ADMIN"."STAFF"

ALTER TABLE "DB2ADMIN"."STAFF"
        ADD PRIMARY KEY
                ("ID");
```

Figure 13-54 Generated DDL

Note: The contents of the file are the construct of the actual tables within the database. This will serve as the basis for the application to view data and update records.

13.5.2 Generating the DDL from a command line

All of the steps that were performed using the DB2 Control Center can be accomplished through the command line as well. Sometimes the Command Line Interface is more appropriate for a given situation and other times it is the only interface available.

For example, on a Windows NT or Windows 2000 DB2 server, locate the DB2 Command Window and open it (see Figure 13-55).

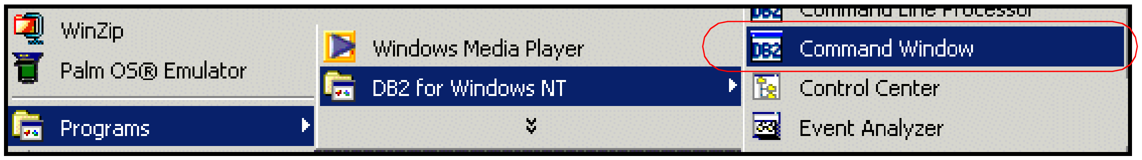


Figure 13-55 Locating the DB2 command line

Once at the DB2 Command window, type in, for example, the following command:

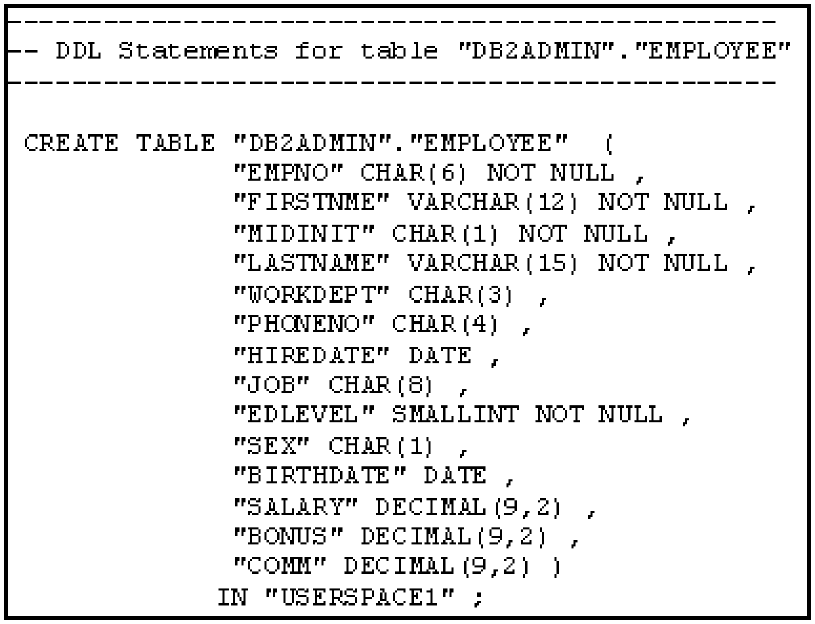
```
db2look -d SAMPLE -u DB2ADMIN -e  -l  -x  -c  -o db2_0.out
```

The above statement extracts the same DDL information as found by going through the DB2 Command Center. The default directory for the DB2 Command Window is C:\SQLLIB\bin; as such, the db2_0.out file will be located here instead of on C:\SQLLIB.

Note: In some cases, you will have to clean up the DDL output to be used in the Mobile Application Builder; some of the characters generated in the output are not supported in DB2e.

13.5.3 DDL preparation

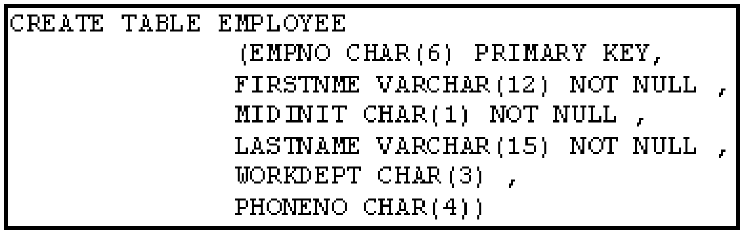
Now that you have extracted the DDL structure, you have to clean it up and pick out only what you need. In this sample scenario, the focus is on the EMPLOYEE table. Locate and extract the data shown in Figure 13-56.



```
------------------------------------------------
-- DDL Statements for table "DB2ADMIN"."EMPLOYEE"
------------------------------------------------

 CREATE TABLE "DB2ADMIN"."EMPLOYEE"  (
              "EMPNO" CHAR(6) NOT NULL ,
              "FIRSTNME" VARCHAR(12) NOT NULL ,
              "MIDINIT" CHAR(1) NOT NULL ,
              "LASTNAME" VARCHAR(15) NOT NULL ,
              "WORKDEPT" CHAR(3) ,
              "PHONENO" CHAR(4) ,
              "HIREDATE" DATE ,
              "JOB" CHAR(8) ,
              "EDLEVEL" SMALLINT NOT NULL ,
              "SEX" CHAR(1) ,
              "BIRTHDATE" DATE ,
              "SALARY" DECIMAL(9,2) ,
              "BONUS" DECIMAL(9,2) ,
              "COMM" DECIMAL(9,2) )
             IN "USERSPACE1" ;
```

Figure 13-56   Sample DDL statement

Now you will need to make the file look like the one shown in Figure 13-57; unfortunately, we are not aware of any automated tool for converting the DDL for use in DB2e, and therefore this is a manual effort in this scenario.



```
CREATE TABLE EMPLOYEE
              (EMPNO CHAR(6) PRIMARY KEY,
              FIRSTNME VARCHAR(12) NOT NULL ,
              MIDINIT CHAR(1) NOT NULL ,
              LASTNAME VARCHAR(15) NOT NULL ,
              WORKDEPT CHAR(3) ,
              PHONENO CHAR(4))
```

Figure 13-57   Updated DDL statement

This change brings the structure into proper formatting for DB2e. Note the subtle differences such as double parentheses on the last line. Save the output into a new file called employee.ddl and remember where you put it. We will come back to it in the next section.

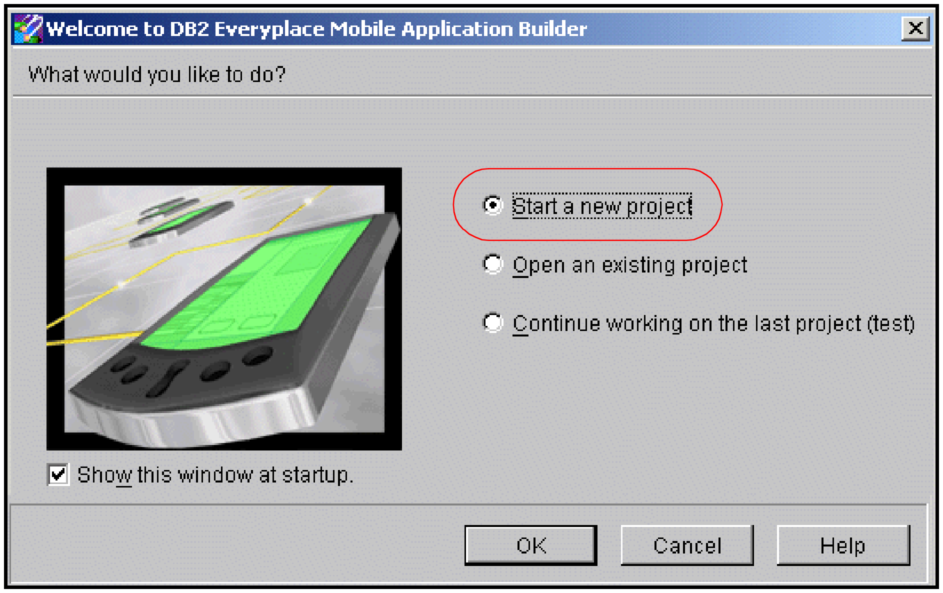
13.5.4 Creating the mobile application

In creating the mobile application, you will be using the IBM Mobile Application Builder; our assumption is that you have the Mobile Application Builder installed with all of its requisites.

Note: If the Mobile Application Builder is not installed, refer to http://www-4.ibm.com/software/data/db2/everyplace/downloads.html for the Mobile Application Builder download, which includes the detailed installation information.

Following are the suggested steps to create the mobile application:

1. Start Mobile Application Builder by clicking **Start->Programs->IBM DB2 Everyplace->Mobile Application Builder**.
2. When presented with a dialog asking if this is a new project, existing project or whether you wish to continue working on the last project, select **New Project** and click **OK** (see Figure 13-58).


Welcome to DB2 Everyplace Mobile Application Builder
What would you like to do?
OK
Cancel
Help

[✓ Start a new project]
Open an existing project
Continue working on the last project (test)

[✓ Show this window at startup.]

Figure 13-58 Starting a new project

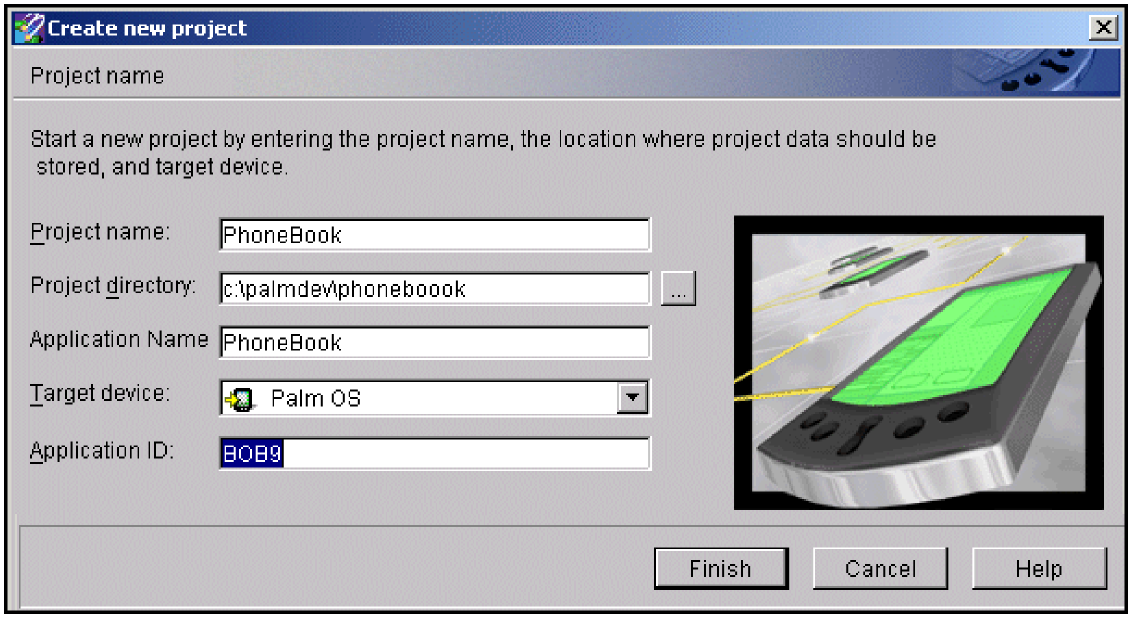
3. The next dialog gives you the options of naming the project, the project directory where you will save it, the application name, the target device type and the application ID.

Make certain with the Application ID that you reference Palm Inc.'s Web site, register your application and get an assigned Application ID. Otherwise, you risk having duplicate Application IDs on the same device. For testing and/or development purposes only, fill out the dialog as demonstrated in Figure 13-59 and click **Finish**.

Note: The URL for registering Palm Applications is:

http://dev.palmos.com/creatorid/

This URL may require further registration once reached.



Start a new project by entering the project name, the location where project data should be stored, and target device.

Project name: PhoneBook
Project directory: c:\palmdev\phonebook
Application Name PhoneBook
Target device: Palm OS
Application ID: BOB9

Figure 13-59 Creating new project options

4. When the Mobile Application Builder opens, you will need to import the employee.ddl file that you have generated in the previous section. As illustrated in Figure 13-60, right-click **Tables** and select **Import Table**.

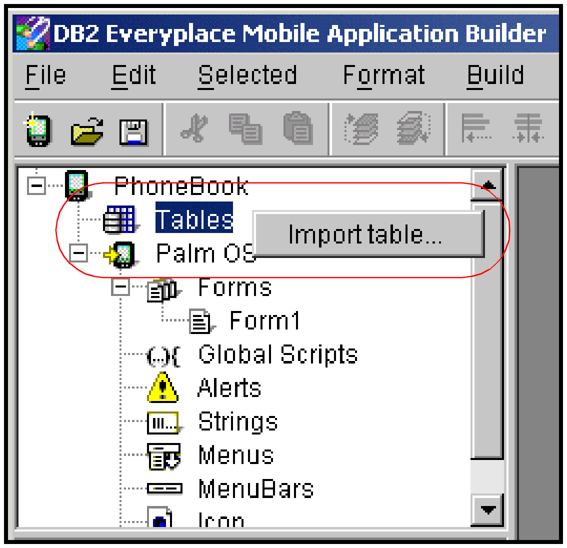


Figure 13-60 Tables

5. Locate the employee.ddl file, highlight it and select **Open**, as shown in Figure 13-61.

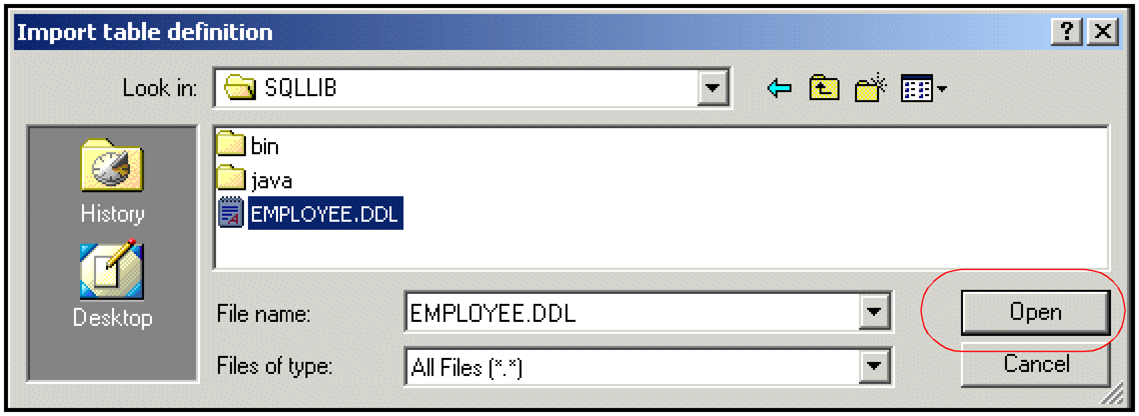


Figure 13-61 Locate DDL

6. You are now back at the Mobile Application Builder, under Tables; you will now see the EMPLOYEE table listed. Right-click the **EMPLOYEE** table and select **Launch form creation wizard**, as shown in Figure 13-62.

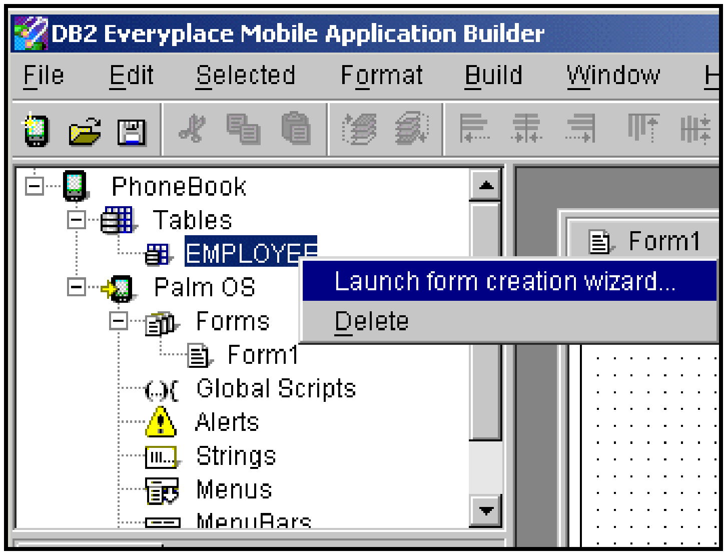
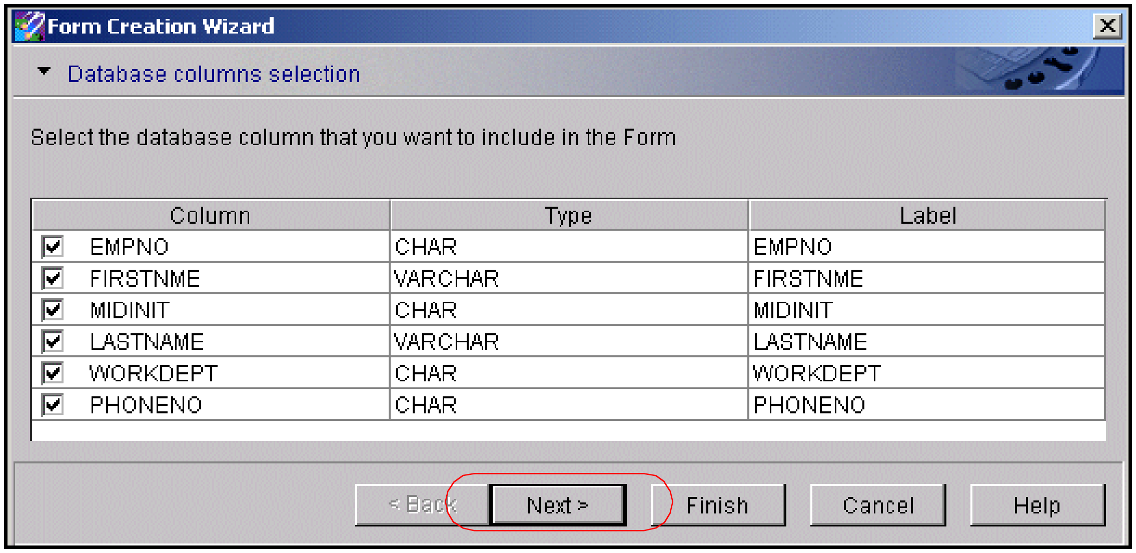


Figure 13-62 Launching the Form Creation wizard

7. The Form Creation Wizard reads all the column entries from the imported employee.ddl file. Since you cleaned up the employee.ddl file earlier, you can leave all of the columns selected. Click **Next** to continue, as illustrated in Figure 13-63.

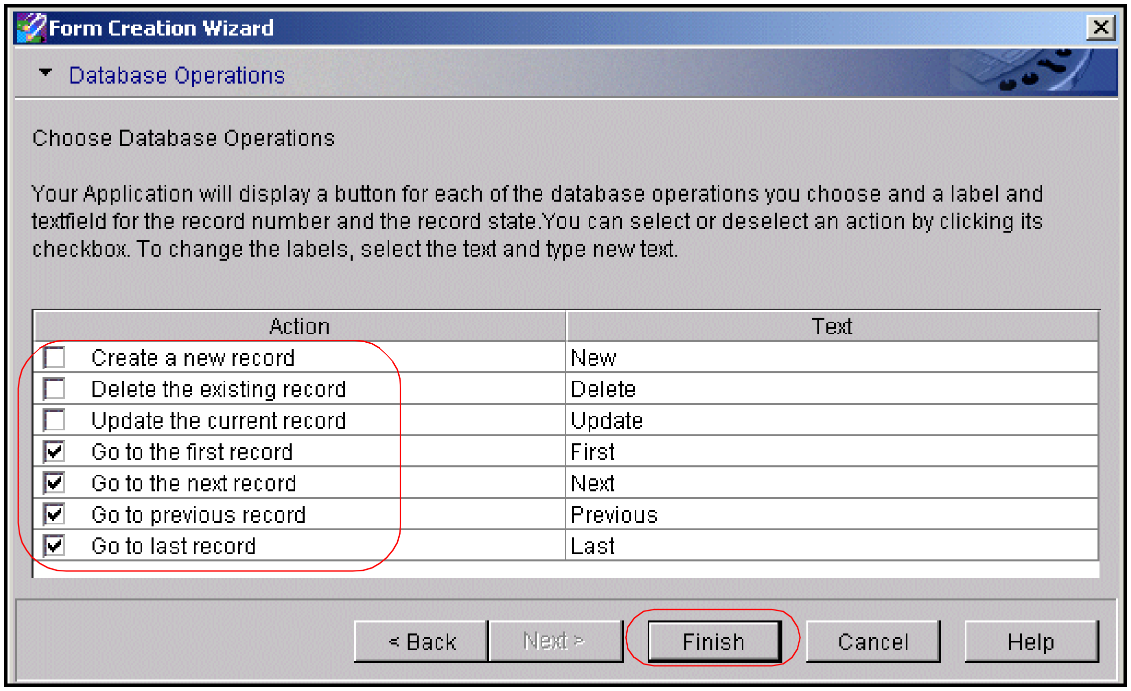


Select the database column that you want to include in the Form

Column	Type	Label
☑ EMPNO	CHAR	EMPNO
☑ FIRSTNME	VARCHAR	FIRSTNME
☑ MIDINIT	CHAR	MIDINIT
☑ LASTNAME	VARCHAR	LASTNAME
☑ WORKDEPT	CHAR	WORKDEPT
☑ PHONENO	CHAR	PHONENO

Figure 13-63 Form Creation Wizard

8. The next window give you options as to what you want to be able to do with the data once you get it, for example, Create, Update, Delete, and so on. The data is controlled at a central source and, in this scenario, you will not be giving the users the ability to Create, Update or Delete. Only viewing the data will be possible. Deselect the **Create**, **Update** and **Delete** options as shown in Figure 13-64 and click **Finish** when you are finished.



Choose Database Operations

Your Application will display a button for each of the database operations you choose and a label and textfield for the record number and the record state.You can select or deselect an action by clicking its checkbox. To change the labels, select the text and type new text.

Action	Text
☐ Create a new record	New
☐ Delete the existing record	Delete
☐ Update the current record	Update
☑ Go to the first record	First
☑ Go to the next record	Next
☑ Go to previous record	Previous
☑ Go to last record	Last

Figure 13-64 Database operations

9. You are now returned to the Mobile Application Builder desktop and are presented with a new form. This new form is the application the wizard has generated the appropriate SQL calls for a DB2e table named EMPLOYEE and has also generated the fields on the form. This is illustrated in Figure 13-65.

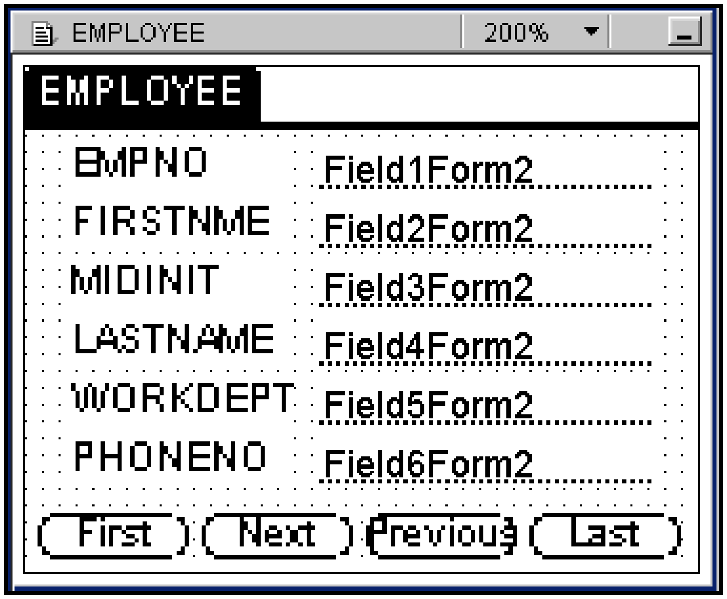


Figure 13-65 Generated fields

10. You will need to remove the default Form 1 that was created. As shown in Figure 13-66, right-click **Form1** in the tree on the left hand side and select the **Delete** option.

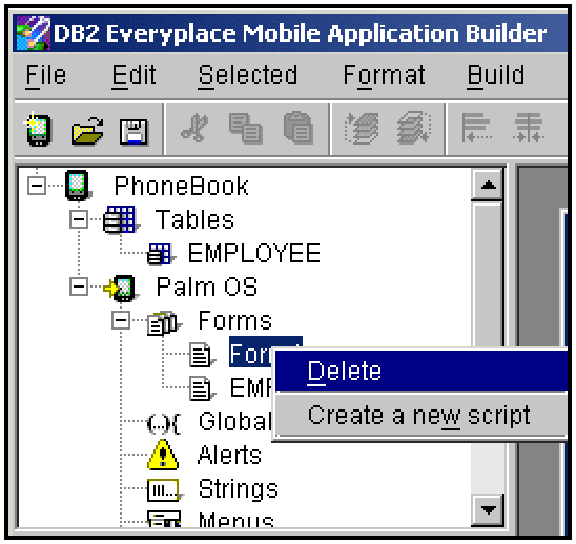


Figure 13-66 Deleting the default form

11. Save the new application by clicking the **disk** icon on the toolbar. Next, you will build the application. Building is the actual process of generating the PRC file for use on the Palm OS devices. Click **Build** and then select **Build** as shown in Figure 13-67.

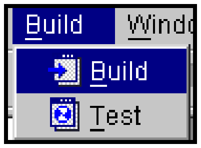
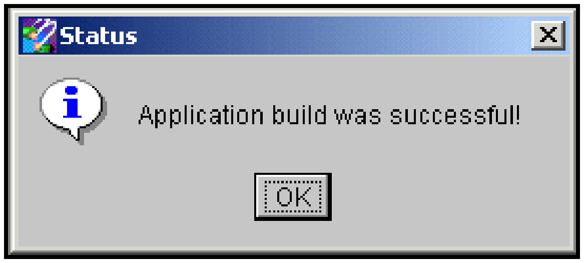


Figure 13-67 Build option

12. Once the Application Build is complete, you will receive an acknowledgment of a successful build, as illustrated in Figure 13-68.

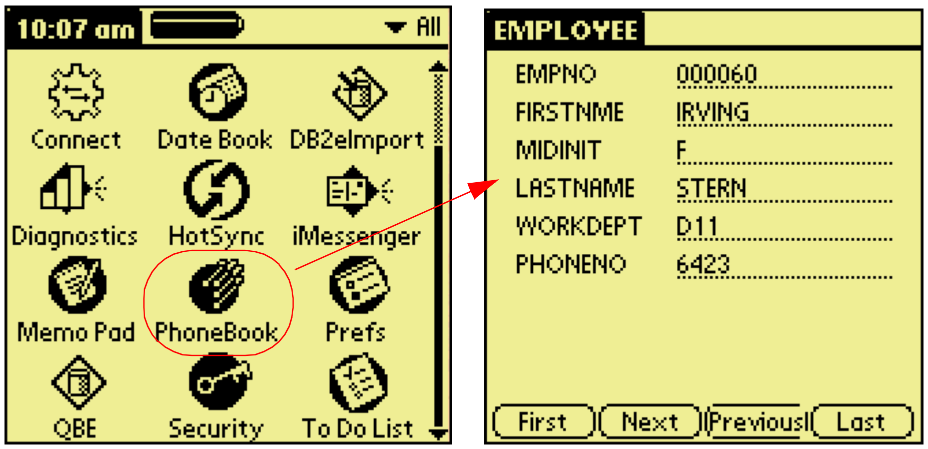


Application build was successful!

Figure 13-68 Application build successful message

13. Now you will need to load the application onto the Palm OS device; make certain that you have already used IMC to synchronize the data into DB2e. See Section 13.4.3, “Database action configuration” on page 547 for details.

14. Once the application is installed, locate the Phone Book icon and click it. Instantly, you are presented with the data stored in the EMPLOYEE table. Click the buttons at the bottom to navigate through the records.



EMPNO	000060
FIRSTNME	IRVING
MIDINIT	F
LASTNAME	STERN
WORKDEPT	D11
PHONENO	6423

Figure 13-69 Running the application

Obviously, this is a very simple application, but it lets you view fairly easily the data that IMC has synchronized to your mobile device.

14

Transaction messaging

This chapter describes the MQSeries Everyplace component of the IBM WebSphere Everyplace Server; it contains:

- “Overview” on page 576
  - Provides a description of MQSeries Everyplace and its features, and also a comparison with standard MQSeries
- “Installation and samples” on page 586
  - Describes the installation of MQSeries Everyplace, and explains how to run one of the supplied examples
- “ChatRoom: an MQSeries Everyplace application” on page 595 to “Setting up the ChatRoom queue managers” on page 620
  - Describe a sample application called the *Chat Room*, the aim of which is to demonstrate various features of MQSeries Everyplace such as:
    - Running the queue manager in client and server mode
    - Running the queue manager as a servlet in IBM WebSphere Application Server
    - Different queue types, such as local and Home Server
    - Synchronous and asynchronous messaging
    - Encryption of messages

- “Extending the YourCo application” on page 661
  - Describe how MQSeries Everyplace can be used to extend access to existing applications
  - Demonstrates a simple customized authentication adapter
- “Integration with WebSphere Everyplace Suite” on page 670
  - Describes how Wireless Gateway can be used to support MQSeries Everyplace applications on wireless type devices

14.1 Overview

The purpose of MQSeries Everyplace is to provide a once-only assured delivery of messages for applications running on devices with one or more of the following characteristics:

- The device typically could not support a fully configured MQSeries queue manager.
- The device will connect using a wireless protocol.

The type of devices that would use MQSeries Everyplace ar:

- Personal Digital Assistants (PDAs)
- Phones
- Sensors
- Laptops

As these sorts of devices are typically used outside of an organization’s intranet, security is an important factor. MQSeries Everyplace provides comprehensive security capabilities to address this potential problem.

A detailed introduction to MQSeries Everyplace can be found in the manual *MQSeries Everyplace Introduction*, GC34-5843. The following sections provide a brief overview of the MQSeries Everyplace functionality. The following diagram provides a high-level overview of the MQSeries Everyplace components.

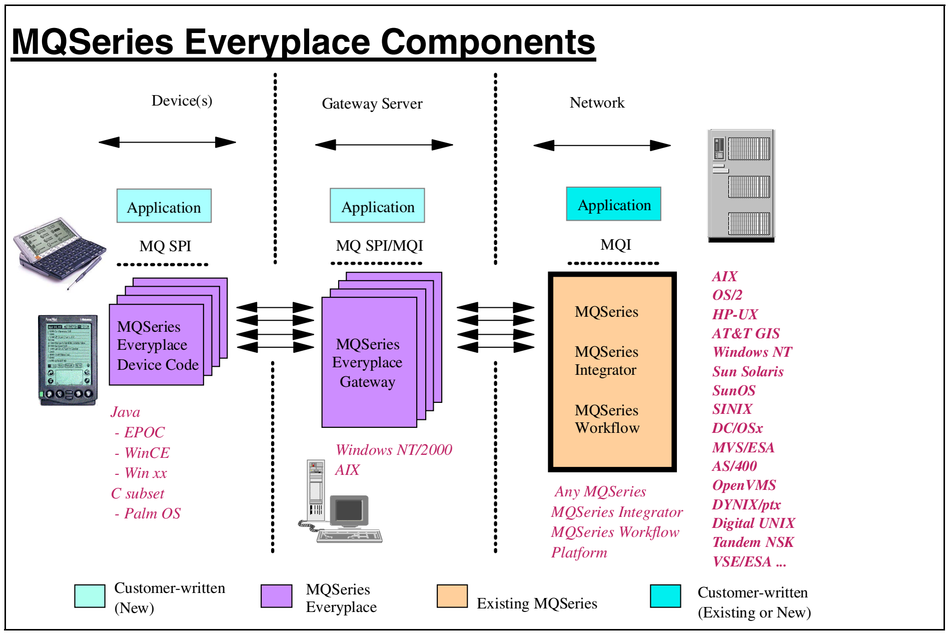


Figure 14-1 MQSeries Everyplace components

14.1.1 Queue Manager comparison

The concept of queue managers is quite different in MQSeries Everyplace and in the standard MQSeries product.

With standard MQSeries, the typical process is for a MQSeries system administrator to install the MQSeries product, then create a queue manager, channels and queues. The queue manager is generally started and remains active for an extended period of time, weeks or longer. Application developers then use the MQSeries APIs to send and receive messages from the queue manager.

In MQSeries Everyplace, however, a set of Java classes and C bindings are provided, which are used by programs to create and control the operation of queue managers. For example, in standard MQSeries, the **`CRTMQM`** command is used to create a queue manager; however, there is no equivalent command within MQSeries Everyplace, in fact there are no commands at all.

MQSeries Everyplace queue managers are object-oriented. Essentially, MQSeries Everyplace queue managers function as part of the application code. They are active only as long as the application program that activates them is running.

For example, in the case where the Java classes are used to create a queue manager, the queue manager exists as an object in the Java Virtual Machine (JVM).

14.1.2 Creating an MQSeries Everyplace Queue Manager

To simplify the discussion, we will describe the process of creating MQSeries Everyplace queue managers using the supplied Java classes.

The process of creating a MQSeries Everyplace queue manager involves writing a program. The program needs to do the following:

a. Create and activate an instance of MQeQueueManagerConfigure
b. Set the queue manager properties and queue manager definition
c. Create definitions for the default queues
d. Close the MQeQueueManagerConfigure instance

Typically, an .ini file is used to store start-up parameters associated with the queue manager. It contains a number of parameters that describe the queue manager, the two most important probably being:

– The name of the queue manager
– Details about the registry location used to store definitional information that describes the queue manager

The registry location is a directory on the disk subsystem where MQSeries Everyplace will store information about the queue manager, such as queue and connection definitions.

Using the information from the .ini file, a program uses the appropriate Java classes to first create the queue manager and then to subsequently start the queue manager when required.

More details about this process can be found in the *MQSeries Everyplace Programming Guide*, SC34-5845.

14.1.3 Types of queue managers

All MQSeries Everyplace queue managers are essentially the same, but the functionality they use determines what sort of role they are being used for.

The three roles or types of MQSeries Everyplace queue managers are:

- Client
- Server
- Gateway

The process of creating and starting the queue manager is still the same regardless of what type of queue manager is being used. The diagram shown in Figure 14-2 is an overview of the configuration for an MQSeries Everyplace queue manager.

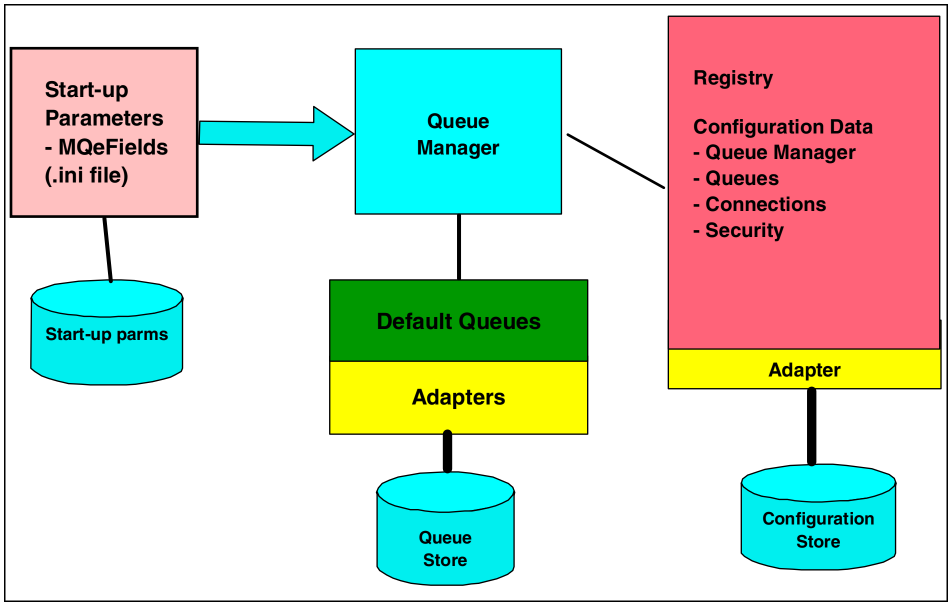


Figure 14-2 Queue manager configuration

Client queue manager

A client queue manager can connect to any number of other MQSeries Everyplace queue managers using a client server type channel. A client queue manager would typically be active for a short period of time.

For example, a salesperson, each time they make a sale, may start up an application on their laptop. The application starts the client queue manager, writes information to a queue, and then ends, stopping the queue manager as well. At the end of the day, the salesman dials up the office from home, starts the application again, which starts the client queue manager, which can now connect to a server queue manager and transfer the messages.

Server queue manager

A server queue manager is one that typically runs for a long period of time. Additionally, it can connect to any number of other non-client type queue managers. Typically, it would be located at some central location in the organization, and the client queue managers would connect to it.

Gateway queue manager

A gateway queue manager is a server queue manager that has been configured with the ability to use the MQSeries-bridge function. This functionality allows messages to flow between MQSeries Everyplace queue managers and standard MQSeries queue managers.

14.1.4 Channel types

In MQSeries Everyplace, your program would define a connection to one or more queue managers. When the program tries to send a message, MQSeries Everyplace will dynamically create a channel to the other queue manager.

MQSeries Everyplace has two types of channels:

– Peer to peer

– Client server

Peer to peer channel

A peer to peer channel has the following characteristics:

– It can be established by the queue manager at either end of the channel.

– The queue manager at each end can send or receive messages.

– A queue manager can have any number of active peer to peer channels to other queue managers.

– A queue manager can only have one active peer to peer channel connected to it.

Client server channel

A client server channel has the following characteristics:

– It can be established from the client end of the connection.

– Only the queue manager at the client end can send or retrieve messages.

– A client queue manager can connect using client server channels to any number of server queue managers.

14.1.5 Adapters

In MQSeries Everyplace, adapters are used to map MQSeries Everyplace components into device interfaces. Certain adapters are also used to control storage of queues into appropriate storage mediums.

MQeDiskFieldsAdapter

This adapter provides support for reading and writing MQeFields object data and message information to a local file system. Typically, this is the default adapter for queues and the registry, since it offers the greatest assurance that data has not been lost. It does not rely on the operating system to do lazy writes to disk.

MQeMemoryFields Adapter

This adapter provides a non-persistent, temporary store for messages in memory. However, it cannot be used for the registry.

MQeReducedDiskFieldsAdapter

This adapter provides support for a high speed alternative to the MQeDiskFieldsAdapter for writing MQeFields object data and message information to disk. However, it does introduce a dependency on the operating system staying up long enough to empty its buffers on the physical disk subsystem.

MQeTcpipAdapter

This adapter provides support for reading data over TCP/IP streams. It is used as the ancestor object for other adapters and cannot be used directly.

MQeTcpipHttpAdapter

This adapter extends the MQeTcpip adapter to provide basic support for the HTTP 1.0 protocol.

MQeTcpipLengthAdapter

This adapter extends the MQeTcpipAdapter to provide a simple, byte-efficient protocol.

MQeTcpipHistoryAdapter

This adapter extends the MQeTcpipAdapter to provide a more efficient protocol that caches recently used data. This adapter takes options, such as <PERSIST><HISTORY>.

MQeUdpipAdapter

This adapter provides support for assured data transfer over UDP/IP datagrams.

MQeWesAuthenticationAdapter

This adapter provides support for tunneling HTTP requests through IBM WebSphere Everyplace Authentication and transparent proxies.

14.1.6 Types of messaging

In standard MQSeries, all messaging is asynchronous, in that a message must be committed to a queue before another process can remove it. In MQSeries Everyplace, there are two types of message delivery: asynchronous and synchronous.

Asynchronous messaging

Asynchronous messaging is similar to standard MQSeries operation. An application on one MQSeries Everyplace queue manager wants to put a message in a queue located on some other MQSeries Everyplace queue manager. The local queue manager requires a remote queue definition of the target queue.

The application puts the message in the remote queue, but it is actually stored locally in the local definition of the remote queue. Some time later, MQSeries Everyplace will deliver that message to the remote queue through the remote queue manager. Actual transmission of the message occusr when the connection between the queue managers becomes available.

Where these messages for remote queues are stored is controlled by the local definition of that queue on the local queue manager. Different adaptors are available to control where these messages are stored. For example, there is an adaptor to have the messages saved to disk, but there is also one to save the messages to memory.

Synchronous messaging

Synchronous messaging is when an application attempts to put a message in a remote queue at a remote queue manager. MQSeries Everyplace will transmit the message only if both the local and target queue managers are online and a connection can be established.

The advantage of synchronous messaging is *performance,* in that the message is not saved locally, but rather transmitted immediately; another advantage is actually knowing that a message has reached its destination.

14.1.7 Messages

In standard MQSeries, there are a number of different types of fixed format messages. The main type is the standard message, which consists of an MQSeries Message Descriptor and the application message. The message descriptor contains a number of fields used by MQSeries.

In MQSeries Everyplace, messages are message objects. There is no concept of a message header or body. In MQSeries Everyplace, a message consists of a unique identifier which is generated automatically, plus one or more named field objects.

Each field object consists of:

– A name

– A type indicator,for example numeric, character, etc.

– A value

The following diagram illustrates this concept:

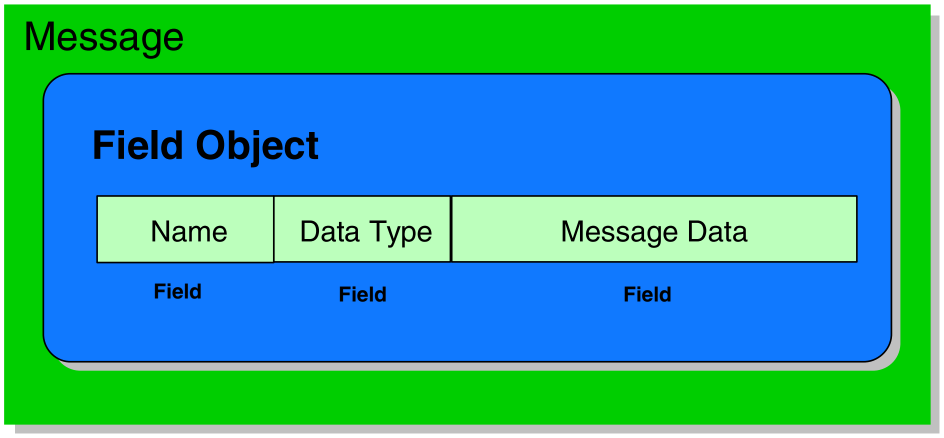


Figure 14-3 MQSeries Everyplace message structure

Each field object is responsible for defining how it is stored and retrieved from a queue. When an application builds a message, it will eventually put the message in a queue; when it does, MQSeries Everyplace uses the dump specification on each field object to determine how the value of the field is stored in the queue for transmission. At the end, when an application retrieves the message, MQSeries Everyplace will use the restore specification on each field object to determine how to restore the transmitted data back to its original format.

14.1.8 Message persistence

There is no concept of persistent and non-persistent messages in MQSeries Everyplace, as essentially every message is treated as persistent. However, the strength of this persistence depends on what method is being used to achieve it.

For example, you could define that messages are to be persisted to memory; clearly, if the device is turned off, the messages are lost. Alternatively, you could specify that the messages should be written to disk. This means that if the device is powered off, the messages are still there on the disk drive for subsequent retrieval and transmission.

MQSeries Everyplace supplies a number of adaptors to handle this persistence, but application programmers can develop their own, for example to persist message data to a database.

14.1.9 MQSeries Everyplace Bridge

The mechanism that allows standard MQSeries and MQSeries Everyplace to exchange messages is referred to as the MQSeries Everyplace Bridge.

This bridging mechanism is in reality a standard MQSeries Java client connection into a standard MQSeries queue manager. Since standard MQSeries client channels are used, once-only assured delivery of messages is performed. In this way, messages transmitted between MQSeries Everyplace and MQSeries cannot be lost.

A default transformer is supplied with MQSeries Everyplace, which handles conversion between the standard MQSeries format and the MQSeries Everyplace message structure. However, it is possible to develop your own customized transformer.

14.1.10 Administration

The standard MQSeries product provides administration tools, for example the **runmqsc** command, which allows you to define queues, channels, etc.

MQSeries Everyplace administration is done using specialized administration messages sent to the queue manager. Performing this type of administration requires that the queue manager be defined with the following two queues:

- AdminQ
- AdminReplyQ

Performing this administration requires writing an application program. The program needs to build the administration request and send it to the AdminQ on the target queue manager. When the message is received by the target queue manager, it is processed. The resource at which the message is targeted uses the administration information in the message to process the request.

14.1.11 ES02 Support Pac

As mentioned in the previous sections, the creation of the queue manager and associated objects is achieved by writing appropriate programs. While not a complex task, it does require some familiarity with MQSeries Everyplace.

To get going faster with MQSeries Everyplace, it is strongly recommended that you use the ES02 Support Pac, available at no charge from IBM. This SupportPac provides what it calls an *Explorer*. This is a Java-based GUI administration tool. It is a set of classes that perform the tasks described above. The GUI interface lets you define queue managers, queues and connections, and perform many other functions related to MQSeries Everyplace.

It can be downloaded from the following Web site:

```
http://www-4.ibm.com/software/ts/mqseries/txppacs/txpsumm.html
```

14.1.12 Security

MQSeries Everyplace provides a comprehensive set of security features to protect message data when held locally or transmitted between queue managers. These features provide authentication, encryption and comclickion.

Full details about security are comprehensively covered in the manual *MQSeries Everyplace Introduction*, GC34-5843. However, we will briefly describe the three categories of security provided:

– Local security, which provides local protection of messages

– Queue-based security, which provides protection of messages between queue managers

– Message level security, which provides message level protection between initiator and recipient

The above security features are invoked when a message is stored or retrieved by MQSeries Everyplace.

Local security

Local security can be used by an application to store a message locally in a queue manager, for example to encrypt a message stored on a local queue.

Queue-based security

Using queue-based security means that the application can leave the issue of security to MQSeries Everyplace. The queues can be defined with attributes that control the type of authentication and encryption used between queue managers.

The only exception is that authentication cannot be performed for asynchronous messaging. If authentication is required, then message level security must be used.

Message level security

Using message level security requires the application to set up the message level attribute when putting the message in the queue. There are two supplied attributes that can be used by applications:

– MQeMAttribute

– MQeMTrustAttribute

MQeMAttribute can be used between queue managers where there is a high degree of trust, as it provides a high level of encryption without the use of Public Key Infrastructure technology.

MQeMTrustAttribute provides a more advanced solution, involving the use of Public Key Infrastructure (PKI). This approach involves the use of digital certificates to authenticate the parties at both ends. As with all certificate-based security mechanisms, it is not a trivial exercise to set up and manage. The documentation in the MQSeries Everyplace covers this area in depth.

14.2 Installation and samples

This section provides a brief overview of the installation of MQSeries Everyplace and the running of some of the supplied samples. A detailed description is available in the *MQSeries Everyplace Read Me First* manual, GC34-5862.

14.2.1 Installation overview

On AIX platforms, the IBM WebSphere Everyplace Server installer can be used to install MQSeries Everyplace.

On Windows platforms, MQSeries Everyplace installation is performed by executing a supplied Java .jar file. When executed, a standard installation process is driven, asking where you want to install the product, etc. For example, after accepting all the defaults during installation on a Windows 2000 system, you would see the product installed into the C:\Program Files\MQe directory.

After installation is complete, the directory where MQSeries Everyplace was installed will contain the Java classes and C bindings that can be used by applications.

Version 1.2.1 of MQSeries Everyplace is shipped with IBM WebSphere Everyplace Server. However, version 1.2.4 of MQSeries Everyplace can be downloaded from `http://www-4.ibm.com/software/ts/mqseries/everyplace/.` This later version, version1.24, was used during the development of this chapter.

14.2.2 Supplied samples

A number of sample programs are supplied with the product; they can be used to verify that the installed classes are working and to provide sample code showing how to use the classes.

Chapter 2 of the *MQSeries Everyplace Programming Guide*, SC34-5845, provides details on the supplied examples and the different functionality they show.

The simplest example to try as a first step would involve creating and using an MQSeries Everyplace queue manager.

Creating a sample queue manager

The first step is to create an example queue manager. The Windows platform is used in this example, but the process is similar on a Unix platform.

First, open a command prompt window and change to the directory where MQSeries Everyplace examples for Windows are installed, in this case to C:\Program Files\MQe\Java\Demo\Windows.

To create a sample queue manager, type in:

```
CreateExampleQm.bat
```

This batch file uses as input an .ini file called ExamplesMQeServer.ini. The output produced by running this command looks lke this:

Example 14-1 Output of CreateExampleQm.bat

```
C:\Program Files\MQe\Java\demo\Windows>createexampleqm

C:\Program Files\MQe\Java\demo\Windows>REM Create the example queue manager -
ExampleQM

C:\Program Files\MQe\Java\demo\Windows>REM
```

```
C:\Program Files\MQe\Java\demo\Windows>REM This batch file invokes the java
class that creates and populates a

C:\Program Files\MQe\Java\demo\Windows>REM registry for the example queue
manager. The registry must be populated

C:\Program Files\MQe\Java\demo\Windows>REM before a Queue Manager can run. The
queue manager created is determined

C:\Program Files\MQe\Java\demo\Windows>REM by entries in queue manager startup
parameters. The examples shipped

C:\Program Files\MQe\Java\demo\Windows>REM with MQSeries Everyplace use ini
files to hold the parameters. By default

C:\Program Files\MQe\Java\demo\Windows>REM .\ExamplesMQeServer.ini startup
parameters file is used.

C:\Program Files\MQe\Java\demo\Windows>REM

C:\Program Files\MQe\Java\demo\Windows>REM Parameters

C:\Program Files\MQe\Java\demo\Windows>REM     java environment name (see
JavaEnv.bat file for details )

C:\Program Files\MQe\Java\demo\Windows>call JavaEnv

C:\Program Files\MQe\Java\demo\Windows>Set JDK=c:\IBM\jdk1.1.8

C:\Program Files\MQe\Java\demo\Windows>set JavaCmd=java

C:\Program Files\MQe\Java\demo\Windows>Set PATH=c:\IBM\jdk1.1.8\bin;C:\Program
Files\ibm\gsk5\lib;C:\IBM
Connectors\Encina\bin;C:\IBMCON~1\CICS\BIN;C:\WINNT\system32;C:\WINNT;C:\WINNT\
System32\Wbem;C:\IMNnq_NT;C:\Program Files\SQLLIB\BIN;C:
\Program Files\SQLLIB\FUNCTION;C:\Program Files\SQLLIB\SAMPLES\REPL;C:\Program
Files\SQLLIB\HELP;C:\WebSphere\AppServer\bin

C:\Program Files\MQe\Java\demo\Windows>set MQE_BASE_DIR=C:\Program Files\MQe

C:\Program Files\MQe\Java\demo\Windows>set CLASSPATH=C:\Program
Files\MQe\java;..\..

C:\Program Files\MQe\Java\demo\Windows>set CLASSPATH=C:\Program
Files\MQe\java;..\..;c:\IBM\jdk1.1.8\lib\classes.zip

C:\Program Files\MQe\Java\demo\Windows>set MQDIR=C:\Program Files\IBM\MQSeries
```

```
C:\Program Files\MQe\Java\demo\Windows>if Exist "C:\Program
Files\IBM\MQSeries\java\lib" set CLASSPATH=C:\Program
Files\MQe\java;..\..;c:\IBM\jdk1.1.8\lib\classes.zip;C:\Program
Files\IBM\MQSeries\java\lib;C:\Program
Files\IBM\MQSeries\java\lib\com.ibm.mq.jar;C:\Program
Files\IBM\MQSeries\java\lib\com.ibm.mqbind.jar;C:\Program
Files\IBM\MQSeries\java\lib\com.ibm.mq.iiop.jar

C:\Program Files\MQe\Java\demo\Windows>if Exist "C:\Program
Files\IBM\MQSeries\java\lib" set PATH=c:\IBM\jdk1.1.8\bin;C:\Program
Files\ibm\gsk5\lib;C:\IBM
Connectors\Encina\bin;C:\IBMCON~1\CICS\BIN;C:\WINNT\system32;C:\WINNT;C:\WINNT\
System32\Wbem;C:\IMNnq_NT;C:\Program Files\SQLLIB\BIN;C:\Program
Files\SQLLIB\FUNCTION;C:\Program Files\SQLLIB\SAMPLES\REPL;C:\Program
Files\SQLLIB\HELP;C:\WebSphere\AppServer\bin;C:\Program
Files\IBM\MQSeries\java\lib

C:\Program Files\MQe\Java\demo\Windows>if Exist "C:\Program
Files\IBM\MQSeries\bin" set PATH=c:\IBM\jdk1.1.8\bin;C:\Program
Files\ibm\gsk5\lib;C:\IBM
Connectors\Encina\bin;C:\IBMCON~1\CICS\BIN;C:\WINNT\system32;C:\WINNT;C:\WINNT\
System32\Wbem;C:\IMNnq_NT;C:\Program Files\SQLLIB\BIN;C:\Program
Files\SQLLIB\FUNCTION;C:\Program Files\SQLLIB\SAMPLES\REPL;C:\Program
Files\SQLLIB\HELP;C:\WebSphere\AppServer\bin;C:\Program Files\IBM\MQSeries\bin;

C:\Program Files\MQe\Java\demo\Windows>java examples.install.SimpleCreateQM.\Ex
amplesMQeServer.ini  .\ExampleQM\Queues\

C:\Program Files\MQe\Java\demo\Windows>
```

After this sample completes, a new directory called ExampleQM will be present. This is the location specified to store the registry information and queues used for the sample queue manager. An expanded view of this directory is shown in Figure 14-4 on page 590.

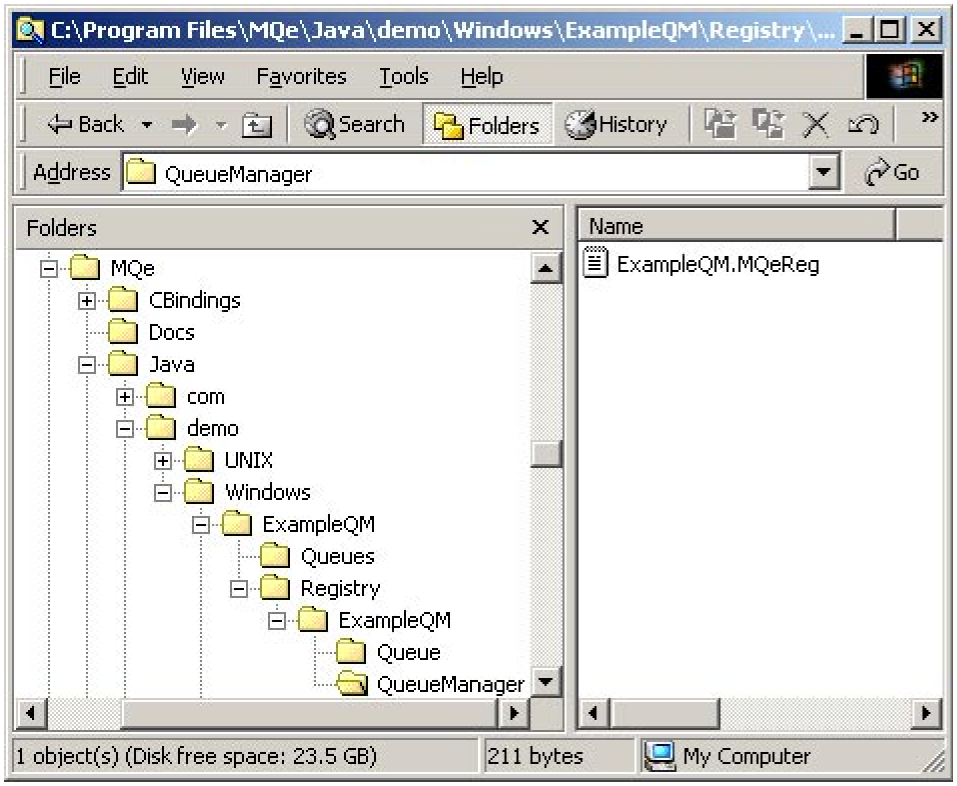

C:\Program Files\MQe\Java\demo\Windows\ExampleQM\Registry\...
File Edit View Favorites Tools Help
Back Search Folders History
Address QueueManager
Go
Folders
MQe
CBindings
Docs
Java
com
demo
UNIX
Windows
ExampleQM
Queues
Registry
ExampleQM
Queue
QueueManager
Name
ExampleQM.MQeReg
1 object(s) (Disk free space: 23.5 GB)
211 bytes
My Computer

Figure 14-4 Expanded view of a queue manager

Using the example queue manager

The next step is to use this example queue manager. From the same command prompt window, type in:

```
ExampleMQeClientTest
```

What the sample does is to write a simple message to the queue SYSTEM.DEFAULT.LOCAL.QUEUE and then retrieve it.

The output produced is shown in Example 14-2.

Example 14-2 Output of ExampleQeClientTest

```
C:\Program Files\MQe\Java\demo\Windows>examplesmqeclienttest
C:\Program Files\MQe\Java\demo\Windows>call JavaEnv
C:\Program Files\MQe\Java\demo\Windows>Set JDK=c:\IBM\jdk1.1.8
C:\Program Files\MQe\Java\demo\Windows>set JavaCmd=java
```

```
C:\Program Files\MQe\Java\demo\Windows>Set PATH=c:\IBM\jdk1.1.8\bin;C:\Program
Files\ibm\gsk5\lib;C:\IBM
Connectors\Encina\bin;C:\IBMCON~1\CICS\BIN;C:\WINNT\system32;C:\WINNT;C:\WINNT\
System32\Wbem;C:\IMNnq_NT;C:\Program Files\SQLLIB\BIN;C:\Program
Files\SQLLIB\FUNCTION;C:\Program Files\SQLLIB\SAMPLES\REPL;C:\Program
Files\SQLLIB\HELP;C:\WebSphere\AppServer\bin

C:\Program Files\MQe\Java\demo\Windows>set MQE_BASE_DIR=C:\Program Files\MQe
C:\Program Files\MQe\Java\demo\Windows>set CLASSPATH=C:\Program
Files\MQe\java;..\..
C:\Program Files\MQe\Java\demo\Windows>set CLASSPATH=C:\Program
Files\MQe\java;..\..;c:\IBM\jdk1.1.8\lib\classes.zip
C:\Program Files\MQe\Java\demo\Windows>set MQDIR=C:\Program Files\IBM\MQSeries
C:\Program Files\MQe\Java\demo\Windows>if Exist "C:\Program
Files\IBM\MQSeries\java\lib" set CLASSPATH=C:\Program
Files\MQe\java;..\..;c:\IBM\jdk1.1.8\lib\classes.zip;C:\Program
Files\IBM\MQSeries\java\lib;C:\Program
Files\IBM\MQSeries\java\lib\com.ibm.mq.jar;C:\Program
Files\IBM\MQSeries\java\lib\com.ibm.mqbind.jar;C:\Program
Files\IBM\MQSeries\java\lib\com.ibm.mq.iiop.jar
C:\Program Files\MQe\Java\demo\Windows>if Exist "C:\Program
Files\IBM\MQSeries\java\lib" set PATH=c:\IBM\jdk1.1.8\bin;C:\Program
Files\ibm\gsk5\lib;C:\IBM
Connectors\Encina\bin;C:\IBMCON~1\CICS\BIN;C:\WINNT\system32;C:\WINNT;C:\WINNT\
System32\Wbem;C:\IMNnq_NT;C:\Program Files\SQLLIB\BIN;C:\Program
Files\SQLLIB\FUNCTION;C:\Program Files\SQLLIB\SAMPLES\REPL;C:\Program
Files\SQLLIB\HELP;C:\WebSphere\AppServer\bin;C:\Program
Files\IBM\MQSeries\java\lib
C:\Program Files\MQe\Java\demo\Windows>if Exist "C:\Program
Files\IBM\MQSeries\bin" set PATH=c:\IBM\jdk1.1.8\bin;C:\Program
Files\ibm\gsk5\lib;C:\IBM
Connectors\Encina\bin;C:\IBMCON~1\CICS\BIN;C:\WINNT\system32;C:\WINNT;C:\WINNT\
System32\Wbem;C:\IMNnq_NT;C:\Program Files\SQLLIB\BIN;C:\Program
Files\SQLLIB\FUNCTION;C:\Program Files\SQLLIB\SAMPLES\REPL;C:\Program
Files\SQLLIB\HELP;C:\WebSphere\AppServer\bin;C:\Program Files\IBM\MQSeries\bin;

C:\Program Files\MQe\Java\demo\Windows>java  examples.application.Example1
ExampleQM .\ExamplesMQeClient.ini
Example1 Started
..Start a queue manager using ini file: .\ExamplesMQeClient.ini
... nested fields [Registry]
LocalRegType    =       FileRegistry

DirName =       .\ExampleQM\Registry\

Adapter =       RegistryAdapter

... nested fields [QueueManager]
Name    =       ExampleQM
```

```
... nested fields [Alias]
QueueManager    =       com.ibm.mqe.MQeQueueManager

DefaultTransporter      =       com.ibm.mqe.MQeTransporter

RegistryAdapter =       com.ibm.mqe.adapters.MQeDiskFieldsAdapter

Trace   =       examples.trace.MQeTrace

MsgLog  =       com.ibm.mqe.adapters.MQeDiskFieldsAdapter

EventLog        =       examples.log.LogToDiskFile

PrivateRegistry =       com.ibm.mqe.registry.MQePrivateSession

FastNetwork     =       com.ibm.mqe.adapters.MQeTcpipHistoryAdapter

FileRegistry    =       com.ibm.mqe.registry.MQeFileSession

ChannelAttrRules        =       examples.rules.AttributeRule

AttributeKey_2  =       com.ibm.mqe.attributes.MQeSharedKey

AttributeKey_1  =       com.ibm.mqe.MQeKey

DefaultChannel  =       com.ibm.mqe.MQeChannel

Network =       com.ibm.mqe.adapters.MQeTcpipHttpAdapter

..Started queue manager: ExampleQM
..Create a message and add data:Example1:Humpty dumpty sat on a wall ...
..Put the message to QM/queue: ExampleQM/SYSTEM.DEFAULT.LOCAL.QUEUE
..Get a message from QM/queue: ExampleQM/SYSTEM.DEFAULT.LOCAL.QUEUE
..Message retrieved contains data Example1:Humpty dumpty sat on a wall ...
Example1 Finished
C:\Program Files\MQe\Java\demo\Windows>
```

As mentioned in the overview section, the queue manager, though defined, only becomes active when an application program activates it.

14.2.3 Integration with VisualAge for Java

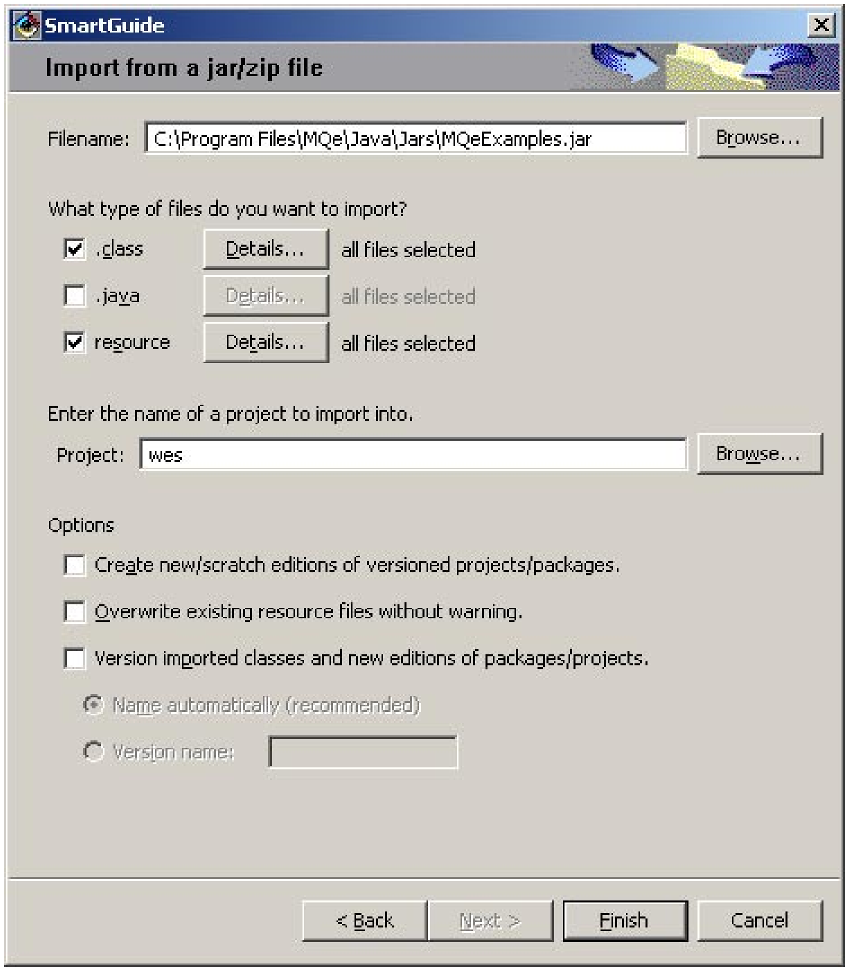
After installing MQSeries Everyplace, notice that in the <install directory>/java/jars directory are the following .jar files containing the Java classes associated with the product:

- MQeExamples.jar
- MQEHighSecurity.jar
- MQeMQBridge.jar
- MQeMiniCertificateServer.jar
- MQeGateway.jar
- MQeDevice.jar
- MQeDiagnostics.jar

To develop programs which use MQSeries Everyplace using VisualAge for Java, the above packages need to be imported. The following section outlines the steps to do this:

1. Start VisualAge for Java
2. Create a project to contain the application you plan to develop, for example ITSO WES MQe Example.
3. Import the MQSeries Everyplace Java .jar files into the project. From the Workbench window, select **File -> import.** Select the radio button to indicate that the source to be imported is a .jar file. Click **Enter**; you will then view the display shown in Figure 14-5 on page 594.

.


SmartGuide

Import from a jar/zip file

Filename: C:\Program Files\MQe\Java\Jars\MQeExamples.jar

What type of files do you want to import?

[✓] .class — all files selected

[ ] .java — all files selected

[✓] resource — all files selected

Enter the name of a project to import into.

Project: wes

Options

[ ] Create new/scratch editions of versioned projects/packages.

[ ] Overwrite existing resource files without warning.

[ ] Version imported classes and new editions of packages/projects.

(●) Name automatically (recommended)

( ) Version name:

Figure 14-5 Importing .jar files

4. Use the Browse button to locate the .jar file to import and then click **Finish**.

Additional information on using VisualAge for Java to develop applications can be found in the IBM redbook, *Programming with VisualAge for Java V3.5*, SG24-5264.

14.3 ChatRoom: an MQSeries Everyplace application

This section describes a sample application that uses MQSeries Everyplace. The aim of this is to show how an application can use MQSeries Everyplace. The application shows the use of:

- Server type queue manager
- Client type queue manager
- Queue manager running as a servlet in IBM WebSphere Application Server
- Local queue
- Remote queue
- Store and forward queue
- Synchronous messaging
- Asynchronous messaging
- Use of queue manager and queue aliasing
- Controlling access to a queue using an authority adapter
- Encryption of messages

The Windows platform is used throughout this chapter to describe the application; however, the AIX platform can be used if desired.

The application consists of three queue managers. They can all be run on one Windows system, or spread out over three if desired.

14.3.1 Overview

This sample application is an implementation of a chat room, except that MQSeries Everyplace is used to transfer the chat as messages between the two participants. The example has been kept simple, and as such this particular chat room only supports two participants, the server and one client.

The diagram in Figure 14-4 on page 590shows an overview of the application and the queue managers used.

First, we will describe how the application works, then describe how to set up this application and the programs that support it.

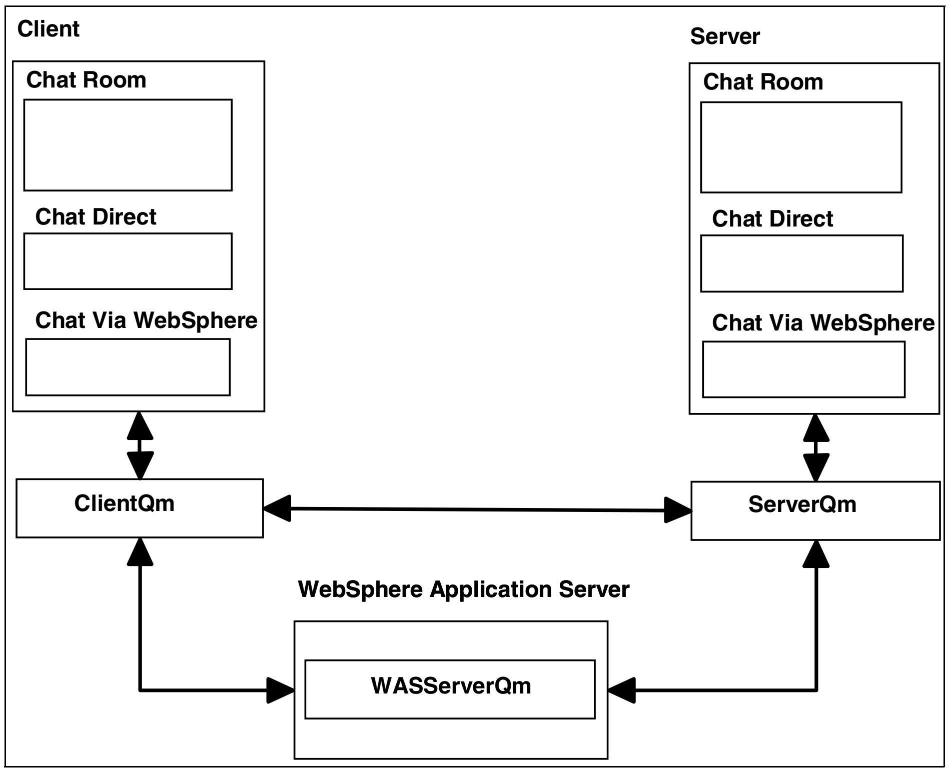


Figure 14-6 Overview of application

The chat room

This application implements a simple chat room. When run, two Java Swing windows are displayed, the titles of the windows being:

- MQSeries Everyplace Server
- MQSeries Everyplace Client

Each window has an output text box to display chat messages that are sent and received.

Each window has two input text boxes. Text entered into the box labelled Chat Direct is sent directly between the ClientQm and ServerQm queue managers.

Text entered into the box labelled Chat Via WebSphere is also sent between the ClientQm and ServerQm queue managers, but passes through the WASServerQm queue manager running in a servlet in the IBM WebSphere Application Server.

Additionally, the client window has three extra buttons labelled:

- Trigger Transmission
- Display Admin GUI
- YourCo Secure Query

The Trigger Transmission button is described in "Asynchronous chatting" on page 653.

The Display Admin GUI button is described in "The administration GUI" on page 655.

The YourCoQuery button is described in "YourCo extensions" on page 662.

When first started, the Chat Direct input text box in the server window is disabled. Only when the client sends a message does it become enabled.

All that is required to send a message to the other participant is to type in some text in the input text boxes, and then click the **Enter** key.

14.3.2 The queue managers

Three MQSeries Everyplace queue managers are used for this application.

Client side

The application that operates the client side of the chat room uses a client type queue manager. This is an example of how an application is started, and then starts the queue manager to perform messaging.

In this example, the client queue manager is called ClientQm.

Server side

The application that operates the server side of the chat room uses a server type MQseries Everyplace queue manager. This is an example of how the queue manager is initially started, and an application loaded after startup is complete.

In this example, the server queue manager is called ServerQm.

IBM WebSphere Application Server

Within WebSphere, a servlet is used to run the queue manager. This queue manager is used to act an a intermediary queue manager between the client and server queue managers, but is also used to allow access to the YourCo sample application that comes with the IBM WebSphere Application Server.

In this example, the queue manager running in WebSphere is called WASServerQm.

Characteristics of the client side queue manager

The client side queue manager can establish a connection to any number of queue managers, which in this case will be to ServerQM and WASServerQM. However, no channel listener is configured for this client side queue manager, thus no other queue manager can initiate a client/server channel connection to it.

This means that applications using this queue manager have two ways of receiving messages:

- Using the GetMessage API to get a message from a remote queue on some remote queue manager; this requires that a connection to a remote queue manager exist.
- Relying on a home-server queue to pull messages from a store and forward queue on a remote queue manager, which the client queue manager will then place in a local queue, from where the application can use the getMessage API to retrieve the message.

Characteristics of the server side queue manager

The server side queue manager has a channel listener configured, so it is able to receive connections from client and server type queue managers. It can also establish connections to other server type queue managers, in this case to WASServerQm.

Applications using this queue manager cannot directly put a message into a queue located at a remote client type queue manager.

Characteristics of the queue manager in WebSphere

The queue manager in WebSphere is started during the initialization phase of an invoked servlet. This queue manager is a server type queue manager, but has no listener configured. In essence, the HTTP server that receives HTTP requests is the de facto listener for the queue manager. Connections can be established in both directions between this queue manager and the server side queue manager.

14.3.3 Connections

There are a number of connections between the various queue managers:

ClientQm to ServerQm

This is a direct channel connection using the default adapter, which is the TCP/IP adapter. Messages are sent in IP packets back and forth over this connection between the queue managers.

ClientQM to WASServerQm

This is a direct channel connection using the HTTP adapter. Messages are wrapped in HTTP headers by the adapter code, and then sent to the machine running IBM WebSphere Application Server. In the definition, the name of the servlet in IBM WebSphere Application Server to be invoked is specified .

ClientQm to ServerQmViaWas

This is an indirect channel definition. When it is defined, it is configured to first send the messages to the queue manager WASServerQm. Configuration information in WASServerQm will then be used to determine how the message is sent on to the queue manager named ServerQmViaWas.

ServerQm to WASServerQm

This is a direct channel connection using the HTTP adapter. Messages are sent wrapped in HTTP headers.

WASServerQm to ServerQm

This is a direct channel connection using the TCP/IP adapter.

14.3.4 Queue discovery

One of the features of MQSeries Everyplace is the ability to perform queue discovery. For example, let us say that there is an existing definition for a queue called ABC on ServerQm. If an application running on another queue manager called ClientQm tried to access that queue, the ClientQm would detect that is has no local definition for this queue. MQSeries Everyplace requires that the queue managers at each end have a local definition of the queue defined with the same attributes. This comes into play when MQSeries Everyplace is establishing a connection between the queue managers, as many connections can be established, but with different attributes, depending on the queues involved.

When ClientQm detects that it does not have a local copy definition of a queue being accessed on a remote queue server, it will query the attributes of the queue defined there, and use those values to define a local definition of the queue.

In this example application, however, we will define all queue definitions required.

14.3.5 MQSeries Everyplace Queue definitions

The following queue definitions are used in this application. Later sections of this chapter explain how to actually define these queues using the ES02 Explorer tool.

Server side queue definitions

```
Queue: ChatRoomQ
Type: Local
Mode: Not applicable
Alias: ChatRoomQAsync, ChatRoomQViaWas, ChatRoomQViaAsync
```

Purpose: this queue receives messages from the client side; the application retrieves the messages from this queue and displays them in the output text area in the window. Note that the client side application can send messages to this queue synchronously or asynchronously.

```
Queue: ChatClientQViaWas
Type: Remote
Mode: Synchronous
Alias: None
Targets: WASServerQm
```

Purpose: This queue is used to demonstrate both indirect messaging and the use of IBM WebSphere Application Server to run a queue manager. The aim is that a message typed into the input text area of the window labelled Chat via WebSphere will still end up in the ChatClientQ on ClientQm, but will travel via the queue manager running in IBM WebSphere Application Server. Messages entered into the Chat via WebSphere area will be placed into this queue.

Client side queue definitions

```
Queue: ChatClientQ
Type: Local
Mode: Not applicable
Alias: None
```

Purpose: this queue receives messages from the server side; the application retrieves the messages from this queue and displays them in the output text area in the window.

```
Queue: ChatRoomQAsync
Type: Remote
Target Queue Manager: ServerQm
Mode: Asynchronous
```

Purpose: if the application is unable to synchronously put the message into the ChatRoomQ local queue on ServerQm, it will put the message into this queue. It is then the responsibility of MQSeries Everyplace to transfer the message to the target queue manager when a connection becomes available.

```
Queue: ChatRoomQViaWas
Type: Remote
Target Queue Manager: ServerQmViaWas
Mode: Synchronous
```

Purpose: messages entered in the Chat via WebSphere box are to be sent to the ServerQm via the queue manager in IBM WebSphere Application Server. The application will put the message to this queue. This is used to demonstrate indirect message routing.

```
Queue: ChatRoomQAsyncViaWas
Type: Remote
Target Queue Manager: ServerQmViaWas
Mode: Asynchronous
```

Purpose: if the application is unable to synchronously put the message into the ChatRoomQ local queue on ServerQm, it will put the message into this queue. MQSeries Everyplace will transfer the message to the target queue manager when a connection becomes available.

```
Queue: ChatSFQ
Type: Home Server
Target Queue Manager: ServerQm
```

Purpose: MQSeries Everyplace polls the corresponding Store and Forward queue of the same name on the specified target queue manager. When it detects that a message is in that queue on the server, it pulls the message from the server to the client. Once the message is received, MQSeries Everyplace will place the message into the local queue specified by the application that originally put the message in the queue. Note that applications cannot access this queue in any way.

```
Queue: ChatSFQViaWas
Type: Home Server
Target Queue Manager: WASServerQm
Mode: Not applicable
Alias: None
Targets: ClientQm
```

Purpose: as for the ChatSFQ queue, MQSeries Everyplace will poll the corresponding Store and Forward queue of the same name on the WASServerQm and pull any messages found there for ClientQm.

WebSphere queue definitions

```
Queue: ChatClientQViaWas
Type: Local
```

Purpose: temporary store for messages entered into the Chat via WebSphere area on the server side.

```
Queue: ChatSFQViaWas
Type: Store and Forward
Mode: Not applicable
Targets: ClientQm
```

Purpose: ClientQM will poll this queue, and pull any messages for ClientQm to the corresponding Home Server queue.

All the queues

The diagram illustrated in Figure 14-7 shows all queues used in the Chat Room sample application. Note that some queues in the diagram are described in later sections of this chapter.

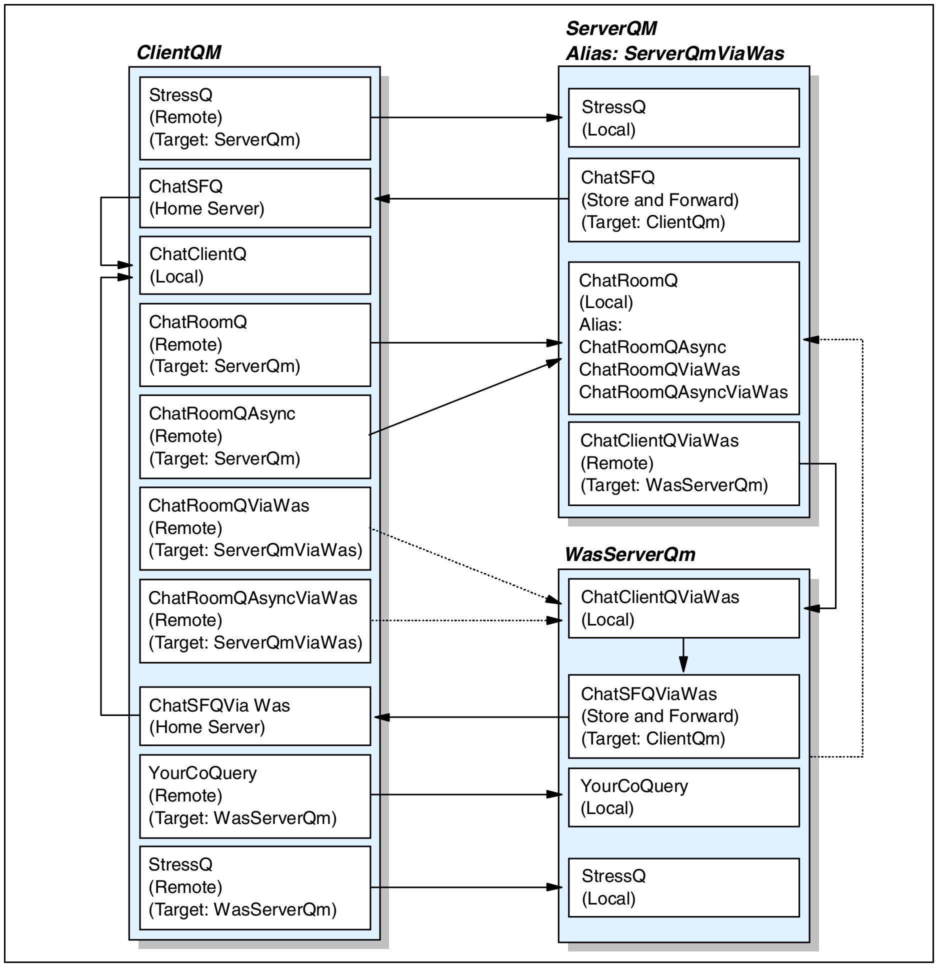

ClientQM
StressQ (Remote) (Target: ServerQm)
ChatSFQ (Home Server)
ChatClientQ (Local)
ChatRoomQ (Remote) (Target: ServerQm)
ChatRoomQAsync (Remote) (Target: ServerQm)
ChatRoomQViaWas (Remote) (Target: ServerQmViaWas)
ChatRoomQAsyncViaWas (Remote) (Target: ServerQmViaWas)
ChatSFQVia Was (Home Server)
YourCoQuery (Remote) (Target: WasServerQm)
StressQ (Remote) (Target: WasServerQm)
ServerQM
Alias: ServerQmViaWas
StressQ (Local)
ChatSFQ (Store and Forward) (Target: ClientQm)
ChatRoomQ (Local) Alias: ChatRoomQAsync ChatRoomQViaWas ChatRoomQAsyncViaWas
ChatClientQViaWas (Remote) (Target: WasServerQm)
WasServerQm
ChatClientQViaWas (Local)
ChatSFQViaWas (Store and Forward) (Target: ClientQm)
YourCoQuery (Local)
StressQ (Local)

Figure 14-7 Chat Room application queues

14.3.6 The application Java packages

The application is written in Java and consists of the following packages.

itso.mqe.chatwindow

This package is used to display the chat room window. It has only one class called RoomWindow. This package is used by the client and server side to display the chat window.

itso.mqe.chatclient

This package is the application used to control the client side of the chat room.

itso.mqe.chatserver

This package is the application used to control the server side of the chat room.

itso.mqe.was

This package contains the code to run the MQseries Everyplace queue manager as a servlet in WebSphere Application Server.

itso.mqe.security

This package contains the code that implements a sample authentication adapter, explained in "YourCo extensions" on page 662.

14.3.7 Client side: class interaction

The diagram in Figure 14-8 on page 604 shows a high-level view of the interaction between the major classes involved on the client side.

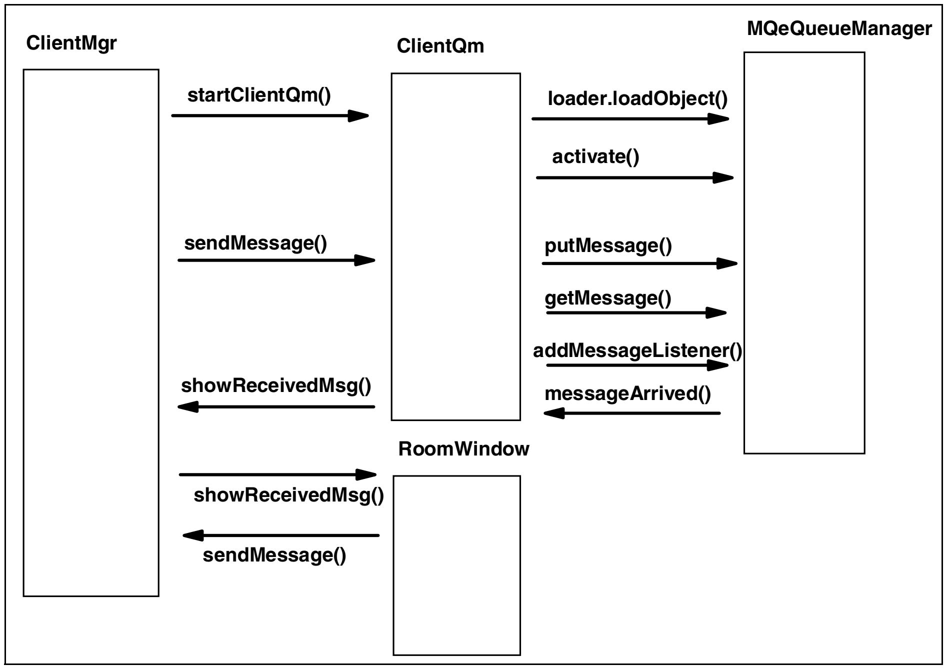


Figure 14-8 Class object interaction - client side

14.3.8 Server side: class interaction

The diagram in Figure 14-9 on page 605 shows a high-level view of the interaction between the major classes involved on the server side.

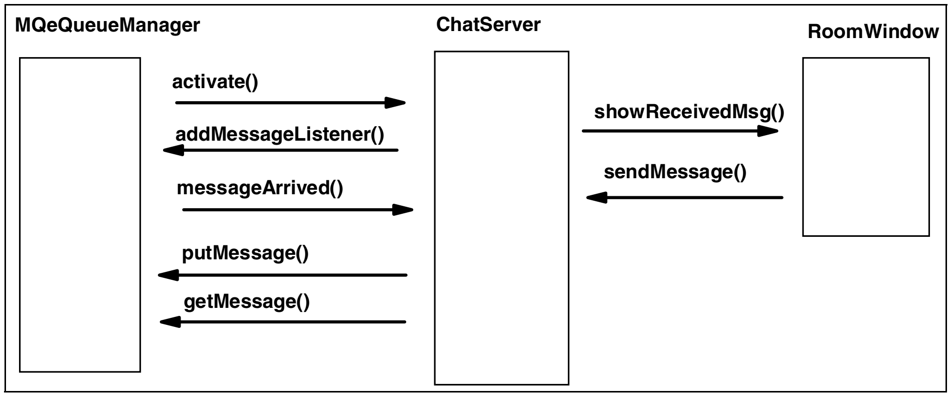


Figure 14-9 Class object interaction - server side

14.4 Starting a queue manager

In the chat room application, queue managers are used in three different ways; this section explains how this occurs.

14.4.1 Started by the application

The client side of the chat room is an example showing how the application is started and then activates the queue manager.

This is the normal approach for client side type applications, as they do not typically require a queue manager running at all times. Rather, an end user will typically want to start the application, have the queue manager started, perform some messaging and then end the application. Such a user does not necessarily require a queue manager running for extended periods of time.

For the chat room application, there are three objects involved on the client side, as follows:

- ClientMgr - the main application
- ClientQm - the queue manager
- RoomWindow - controls display of the GUI chat window

When the application is started, the main method creates a new instance of RoomMgr, which results in a RoomWindow object being created by the constructor, and then calls the startChatRoom method.

The startChatRoom method of the RoomMgr class starts, creates a ClientQm object, and then calls the startClientQm method of that class.

The startClientQm method contains the code to start the queue manager. The code shown in Example 14-3 to Example 14-6 on page 607 is from this method.

First, the .ini file containing information relating to the queue manager is read, as shown below:

Example 14-3 Reading the .ini file

```
// Access the file
        File diskFile = new File(clientIniFile);

      // Create a byte array big enough to hold the file's contents.
        byte data[] = new byte[(int) diskFile.length()];
      // Read the file into the byte array and close the file.
        FileInputStream inputFile = new FileInputStream(diskFile);
        inputFile.read(data);
        inputFile.close();
```

Note: The code above illustrates a standard way of performing file input. An alternate approach would be to read an .ini file into a fields object, using the MQeQueueManagerUtils.loadConfigFile() method.

Once read, the data is parsed and stored in MQeField type objects, as shown in the code below:

Example 14-4 Parsing the .ini file

```
MQeFields iniSections =
                MQeFields.restoreFromString("\r\n", // end of record string
          "[#0]", // section pattern
              "(#0)#1=#2", // keyword pattern
                configDataBuff.toString() + "\r\n");
```

Then a queue manager object is created and alias definitions from the .ini file are processed. Alias definitions are a way of assigning a shorter logical name to class names. These alias definitions can then be used in other sections of the .ini file, if required. The code necessary to do this is shown in Example 14-5 on page 607.

Example 14-5 Processing the alias entries

```
/* Create queue manager object */
         qMgr = new MQeQueueManager();

        if (iniSections.contains(Section_Alias)) {

            // Get all the fields inside the alias section
            MQeFields section = iniSections.getFields(Section_Alias);
            Enumeration keys = section.fields();
            while (keys.hasMoreElements()) {

                // For each key, get the value and add the mapping to the MQe
                // internal alias table
                String key = (String) keys.nextElement();

                MQe.alias(key, section.getAscii(key).trim());

                System.out.println("Key: " + key + " Val: " +
section.getAscii(key).trim());
            }

            //    sectionProcessed(Section_Alias);
        }
```

Finally, we now call the loader method of the MQe class to activate the queue manager, as shown in the code below:

Example 14-6 Activating the queue manager

```
if (iniSections.contains(Section_QueueManager)) {
            qMgr = (MQeQueueManager)
MQe.loader.loadObject(Section_QueueManager);
            if (qMgr != null) {

                // Activate the queue manager.

                qMgr.activate(iniSections);

                // Processing was successful.
            }
        }
```

The queue manager is now active within the Java virtual machine and can be used by the application.

14.4.2 Started by the ES02 support pac

On the server side, the Explorer tool of the ES02 support pac is used to start the queue manager. A program to start the server side queue manager would be similar to one developed for the client side, but it would need to start some other queue manager functionality, such as the channel listener.

Typically, the requirement for a server side queue manager is to be active at all times, so that client type queue managers can connect at any time. Any number of applications may be loaded into the JVM to enable them to perform messaging.

Note: The current release of MQeExplorer v1.25 will allow you to create and start clients, peers and gateways.

Version 1.26, to be released in the future, will eliminate the need for .ini files. Also, properties of the queue manager may be changed whether in the registry or the .ini file. Queue managers can be freely changed between clients, peers, servers and gateways. Necessary components can be created, modified or deleted on an ad hoc basis.

14.4.3 Started by a servlet

The chat room application demonstrates how a queue manager can be run as a servlet in IBM WebSphere Application Server. The queue manager that runs in WebSphere is a server type queue manager.

Normally, a server type queue manager has a listener, which listens on a port to which clients can establish a connection. However, a queue manager running in WebSphere cannot start a listener. The HTTP server is in effect the listener for the queue manager. Other queue managers access the queue manager in WebSphere by sending the message requests wrapped in HTTP headers.

For the chat room application, there are two classes involved in the WebSphere part side, as follows:

– ITSOMQeServlet - starts the queue manager and handles messages received.
– WasQMgr - in essence the application, it acts upon messages.

When another queue manager sends a message to the queue manager running in WebSphere, the HTTP headers will specify the name of the servlet to be invoked within WebSphere, in this case ITSOMQeServlet. The servlet will start the queue manager the first time it is invoked; this occurs in its init method.

The name of the .ini file to use is found by reading a property file called MQe.properties. The MQe.properties file should be in a directory in the classpath. The code to obtain the .ini file name is as follows:

Example 14-7 Obtaining the .ini file when starting in WebSphere

```
String mqePropName = "MQe";

        try {

            PropertyResourceBundle resourceBundle =
                (PropertyResourceBundle)
PropertyResourceBundle.getBundle(mqePropName);
            iniFile = resourceBundle.getString("IniFile");
```

The same code, as shown in “Started by the application” on page 605, is used to read the .ini file, parse it, load the alias and start the queue manager. The only addition is code to activate a channel manager just prior to activating the queue manager. A channel manager is an object used to handle the communication processes involved between queue managers. The code to activate it is shown in Example 14-8:

Example 14-8 Activating a channel manager

```
if ( iniSections.contains( Section_ChannelManager ) ) {
            MQeFields section = iniSections.getFields( Section_ChannelManager
);
            channelManager    = new MQeChannelManager( );
            channelManager.numberOfChannels( section.getInt( "MaxChannels" ) );
          //  sectionProcessed(Section_ChannelManager);
        }
```

Once the init method completes, the queue manager is active within WebSphere Server.

After the queue manager is started, a WasQMgr class object is created, and a reference to the queue manager passed as a parameter. As in the RoomMgr class, the activate method is called, which allows the WasQMgr class to save a reference to the queue manager that has been started.

The doPost method

The doPost method of the ITSOMQeWas servlet is worth discussing here. Queue managers sending message to a queue manager in WebSphere Application Server will wrap the message in an HTTP header, specifying that the HTTP request is a POST to a specified servlet.

In the chat room application, this will cause the doPost method of the ITSOMQeWas servlet to be invoked.

The method is in effect performing the same role as the listener for a server type queue manager. All it does is read the HTTP data received and pass it to the channel manager associated with the queue manager.

The channel manager then takes this data, removes the HTTP headers, and places the message in the queue.

The response from this method call to the channel manager is not a message as such; rather, it is just a standard HTTP reply that is to be sent back to the sending queue manager as part of the normal HTTP flow.

The code for this is show in Example 14-9:

Example 14-9 Passing HTTP input to the queue manager

```
ServletInputStream  httpIn  = request.getInputStream(); // input stream

        // get the request
        read( httpIn, httpInData, max_length_of_data);

        String mqeInput = new String(httpInData);
        System.out.println("MQeInput: " + "mqeInput");

        // process the request
        byte[] httpOutData = channelManager.process(null, httpInData);
        String mqeReply = new String(httpOutData);
        System.out.println("MQeReply: " + "mqeReply");

        // appears to be an error in that content-length is not being set
        // so we will set it here
        response.setContentLength(httpOutData.length);
        response.setIntHeader("content-length", httpOutData.length);

        // Pass back the response
        httpOut.write(httpOutData);
```

14.5 Starting applications

In the chat room application, applications that use queue managers are used in three different ways; this section explains how this occurs.

14.5.1 Client side

Starting the application on the client side of the chat room is exactly the same as starting any Java program. The **`Java`** command, in conjunction with the package and class name, is used to start the application.

14.5.2 Server side application loading

The server side of the chat room is an example showing the application started after the queue manager has started; in effect, it is loaded by the queue manager.

This is the normal approach for server side applications. Typically, the requirement is for the queue manager to be active at all times, so that client type queue managers can connect at any time. Any number of applications may be loaded into the JVM to enable them to perform messaging activities.

For the chat room application, there are two objects involved in the server side, as follows:

– RoomMgr - interacts with the queue manager and the GUI window

– RoomWindow - controls the display of the GUI chat window

The section “Started by the ES02 support pac” on page 608 described how the ES02 support pac was used to start the queue manager.

The application we want to have loaded when the queue manger is started is specified in the .ini file for the queue manager; in this case, it is the chat room application. The lines from the.ini file are as follows:

Example 14-10 Code from the .ini file

```
[AppRunList]
(ascii)App1=itso.mqe.chatserver.RoomMgr
```

Parameters can also be passed to the application from the .ini file, as shown below:

Example 14-11 Parameters from the .ini file

```
[App1]
(ascii)ClientQueue=ChatClientQ
(ascii)ChatRoomQ=ChatRoomQ
```

A detailed explanation of how applications started this way need to be written starts on page 51 of the *MQSeries Everyplace Programming Guide*, SC34-5845-04.

Briefly, however, the RoomMgr class extends the base MQe class, and implements these three interfaces:

- runnable - used to allow it to create a new thread on which to run
- MQeRunListInterface - used to allow queue manager to pass information
- MQeMessageListenerInterface - used to allow application to notify the queue manager as to what queues it is interested in

The class that will be started must have a method called *activate*. This will be the first method executed when the application is started. The first thing it does is to save a reference to the queue manager passed as a parameter. This will allow the application to interact with the queue manager. The following line saves the queue manager ID:

Example 14-12 Saving the queue manager reference

```
   qmgr = (MQeQueueManager) owner;    /* Qmgr is owner of the application */
```

A new thread is then created and started, which will cause the *run* method of the RoomMgr class to be executed. The run method consists of this code:

Example 14-13 RoomMgr run method

```
if (itsoChatRoom == null) {
        itsoChatRoom = new RoomWindow(this);
        itsoChatRoom.showChatWindow();
        itsoChatRoom.sendChatMsg();
        itsoChatRoom.setupWasInputListener();
    }
    try {

        qmgr.addMessageListener(this, chatRoomQ, null);
```

The above code creates a RoomWindow object and then calls a method on that object to create the GUI window and display it. Then, the code adds a message listener on a specified queue. The purpose of the Message listener is explained in “The MQeMessageListener interface” on page 614.

The queue manager and application are now both active.

Application data can be passed to the application that is started in this fashion. In the .ini file, after the section identifying the applications to load, data can be added to pass to the application. For example, we could add these lines to the .ini file:

Example 14-14 Application-related startup data

```
[App1]
(ascii)ClientQueue=ChatClientQ
(ascii)ChatRoomQ=ChatRoomQ
```

Sample code to access this data in the activate method is shown below:

Example 14-15 Accessing application setup data

```
   Enumeration enum = setupData.fields();
   try {
      while (enum.hasMoreElements())
      {
         String fieldName = (String) enum.nextElement();
         String value = setupData.getAscii(fieldName);
         System.out.println("Field name: " + fieldName +
                     " value: " + value);
      }
   }
```

14.5.3 Applications in WebSphere Application Server

Applications that are to run in and access a queue manager in WebSphere Application Server function essentially in the same way as the applications written for a server.

The chat room application demonstrates this. Text entered into the Chat Via WebSphere box on the server side is put to the queue ChatClientQViaWas on the WASServerQm.

When the init method of the ITSOMQeServlet was executed, it created a WasQMgr object, passing it as a reference to the queue manager, and also added a message listener for the ChatClientQViaWas queue. The WasQMgr object is the application.

When a message arrives in this queue, the messageArrived method of the WasQMgr object is invoked. This method then retrieves the message and puts it to the ChatClientQ on the ClientQm queue manager, as shown in the code shown in Example 14-16.

Example 14-16 Processing messages in WebSphere

```
      msgObj = wasQMgr.getMessage(null, "ChatClientQViaWas", null, null, 0);
      System.out.println("From: " + msgObj.getOriginQMgr() +
            " : " + eventQueueName +
            " msg: " + msgObj.getAscii("Message"));
      System.out.println("Relay chat msg to ClientQm : " +
msgObj.getAscii("Message"));
      replyMsg.putAscii("Message", msgObj.getAscii("Message"));
      wasQMgr.putMessage("ClientQm", "ChatClientQ", replyMsg, null, 0);
```

14.6 Listening for messages

This section describes how a queue manager notifies an application that a message is available for processing.

14.6.1 The MQeMessageListener interface

In standard MQSeries, it is quite common for an application to wait for a message to arrive in a queue. It does this by specifying a WAIT option on the GET message API.

In MQSeries Everyplace, the corresponding approach is for an application to implement the MQeMessageListener interface.

For example, the code shown next from the run method of the RoomMgr class tells the queue manager that the application wants to be notified whenever a message is put onto the queue specified in the variable chatRoomQ.

Example 14-17 Adding a message listener

```
qmgr.addMessageListener(this, chatRoomQ, null);
```

An application can add a listener for as many queues as it requires. The application then needs to have a messageArrived method, as this will be the method invoked by the queue manager when a message arrives on any of the queues for which a listener has been added.

The messageArrived method is passed a MessageEvent object, which contains information about the message that has arrived, such as the queue the message is in and the queue manager from which it comes.

It is now up to the application to get the message and process it as required. For example, in the case of the chat room application, when a message arrives in the ChatRoomQ, the messageArrived method in class RoomMgr is called, the message is retrieved from the queue and displayed on the GUI window, as shown in the following code.

Example 14-18 Retrieving a message

```
try {
        MQeMsgObject msgObj = qmgr.getMessage(null, chatRoomQ, null, null, 0);
        /* get the message */

        if (originQMgr == null)
            originQMgr = msgObj.getOriginQMgr();

        System.out.println(
            "From: " + eventQMgr + " : " + eventQueueName
                + "Really: " + originQMgr + " msg: "
                + msgObj.getAscii("Message"));

        itsoChatRoom.showReceivedMsg("From: " + originQMgr + " : " +
             msgObj.getAscii("Message"));
```

The same approach is used on the client side and in WebSphere Application Server. In the WebSphere case, the init method of the ITSOMQeWas servlet adds the message listeners, and the messageArrived method is implemented in the WasQMgr class.

Note that there is no comparable notion of triggering an application in MQSeries Everyplace as there is in standard MQSeries.

14.7 Chat room application flows

The section describes what happens when a message is entered into the various text input boxes in the chat windows.

14.7.1 Chat - Client to Server -Direct

The process that occurs when a message is typed into the Chat Direct input box in the window of the client side of the chat room is as follows:

- The message is typed into input text area, the **Enter** button is clicked.
- The ChatMessage method is sent in the itso.mqe.chatwindow class. RoomWindow is invoked, echoes the message to the output text area, and calls the sendMessage method.

– The sendMessage method in the class itso.mqe.chatclient.ClientMgr performs the task of first trying to send the message synchronously, and, if that fails, it puts the message into the alternate queue, to have the message sent asynchronously; the code from this method is shown in Example 14-19.

Example 14-19 Using the Chat Direct input box

```
try {
        MQeMsgObject msgObj = new MQeMsgObject();
        //String venturing = e.getQueueManagerName();/*get id of Qmgr msg from
*/
        //String eventQueueName = e.getQueueName();/*get queue name */
        try {
            System.out.println("Msg to send: " + message);

            msgObj.putAscii("Message", message); /* set up the message */

            System.out.println(
                "Send to: " + targQMgr + " destQ: " + targQ + "doing PUT: " +
message);

            /* If the string 'Stress Test' do not appear in the text typed
               in, then put the message to the queue to have it sent
               synchronously.
               When the 'Stress Test' string is found, invoke a method
               to handle that case */

            if (message.indexOf("Stress Test") < 0)
                myClientQmgr.putMessage(targQMgr, targQ, msgObj, null, 0);
            else
                stressTest(viaWasFlag, message);

            System.out.println(
                "Sent to: " + destQMgr + " : " + " msg: " +
msgObj.getAscii("Message"));
        } catch (Exception ex) {

           /* If an exception occurs as a result of the put message
              attempt, put the message to the alternate queue to have
              the message sent asynchronously */

           System.out.println("Error sending msg" + ex);
            chatRoomClient.showReceivedMsg(
                "## Chat room server unavailable"
                    + " will attempt to send message asynchronously ##");

            msgObj.resetMsgUIDFields();
            try {
                System.out.println(
                    "Async Send to: " + targQMgr + " destQ: "
                        + targQAsync + "doing PUT: " + message);

                myClientQmgr.putMessage(targQMgr, targQAsync, msgObj, null, 0);

                chatRoomClient.showReceivedMsg(
                    "## Message saved, will be sent Asynchonously ##");

            } catch (Exception ex2) {
                chatRoomClient.showReceivedMsg("## Catastrophic failure, async
PUT failed ##");
                System.out.println("Error doing Async PUT" + ex2);
            }
        }
```

– In this case, targQ is set to a value of **ChatRoomQ**, the method attempts to put the message synchronously to this queue on the remote queue manager called ServerQm.

– If the remote queue manager is unavailable, the process will fail and an exception will be raised; this is caught by the Java code. The application will then put the message to the queue targQAsync, which has been set to a value of **ChatRoomQAsync**; this is now an asynchronous message operation, and MQSeries Everyplace is now responsible for transferring the message to the remote queue manager when a connection becomes available.

  When the connection to the remote queue manager becomes available, the queue manager will send the message. However, the application put the message to a queue called ChatRoomQAsync, and there is no queue by that name on the remote queue manager, but the definition for the ChatRoomQ queue specifies that it has an alias of ChatRoomQAsync. This means that the message will placed in the ChatRoomQ on the remote queue manager

14.7.2 Chat - Client to Server - Via WebSphere

The process that occurs when a message is typed into the Chat via WebSphere input box in the window of the client side of the chat room is as follows:

– The message is typed into the input text area, the **Enter** button is clicked.

– The ChatMessage method is sent in the itso.mqe.chatwindow class. RoomWindow is invoked, as it was when text was entered into the Chat Direc' box; the message is echoed to the output text area, the sendMessage method is called.

– The sendMessage method is implemented in the itso.mqe.chatclient class. ClientMgr performs the task of first trying to send the message synchronously, and, if that fails, it puts the message into the alternate queue, to have the message sent asynchronously.

– The code executed, as shown in Example 14-19 on page 616, is the same as it was for the Chat Direc' case. The difference is that the target queue manager is set to **ServerQmViaWas** and targQ and targQAsync are set to different values; this is done in the sendMessage method using the code shown in Example 14-20.

Example 14-20 Setting value of target queue

```
/* The action listener that is invoked when the Enter key is
   clicked in a text box, sets the  value of viaWasFlag when it calls
   this method.

   The value is null if called from the listener for the 'Chat Direct'
   box, and not null if called from the listener for the
   'Chat via WebSphere' box */

 if (viaWasFlag == null) {
     targQ = destQueue;
     targQMgr = destQMgr;
     targQAsync = destQueueAsync;
 } else {
     targQ = destQueueViaWas;
     targQMgr = destQMgrViaWas;
     targQAsync = destQueueAsyncViaWas;
 }
```

As in the direct case, the application will try to put the message synchronously into the queue ChatRoomQViaWas on ServerQmViaWas. Recall, however, that ChatRoomQViaWas has been defined as an alias for the ChatRoomQ, and that ServerQmViaWas is an alias for the ServerQm queue manager. In effect, this will be put to the ChatRoomQ on the ServerQm. This means the put message request only succeeds if there is a connection right through to the ServerQm.

If the message cannot be put synchronously, it is put asynchronously, to the ChatRooQAsyncViaWas queue, for transmission by the queue manager when a connection via WASServerQm to ServerQm is available.

14.7.3 Chat - Server to Client - Direct

The process that occurs when a message is typed into the Chat Direct input box in the window of the server side of the chat room is as follows:

– The message is typed into the input text area; the **Enter** button is clicked.

– The sendChatMessage method is implemented in the itso.mqe.chatwindow class. RoomWindow is invoked, echoes the message to the output text area, and calls the sendMessage method.

> **Note:** the RoomWindow class is used at both the client and server ends to display the chat window.

– The RoomMgr class has its own sendMessage method, but it is essentially the same as the one in the ClientMgr class; the application will put the message to the ChatClientQ on the ClientQm queue manager; the major difference here is that the server only sends its messages asynchronously,

– When the put message is executed, the queue manager will store the message on the ChatSFQ queue; this is because the server side queue manager cannot send a message to the client side queue manager. The queue manager will store any messages for ClientQm on this store and forward queue.

– On the client side, the ClientQm has a corresponding Home server queue called ChatSFQ; the queue is configured to poll the corresponding store and forward queue on the ServerQm every five seconds; when it detects a message on that queue, it will pull the message and place it in the Home server queue, in this case ChatSFQ.

– The queue manager will then move the message to the target queue specified by the application, in this case ChatClientQ.

– The messageArrived method in the ClientMgr object will be invoked by the queue manager; it will retrieve the message and display it in the output area of the GUI window.

14.7.4 Chat - Server to Client - using WebSphere

The process that occurs when a message is typed into the Chat via WebSphere input box in the window of the server side of the chat room is as follows:

– The message is typed into the input text area, the **Enter** button is clicked.

– The sendChatMessage method is implemented in the itso.mqe.chatwindow class. RoomWindow is invoked, echoes the message to the output text area, calls the sendMessage method, but passes a flag to indicate that the message is to be sent using WebSphere.

- The sendMessage method adds the message to a queue called ChatClientQViaWas on the WASServerQm, a synchronous operation.

Note: This message is sent as an HTTP request to WebSphere, where the ITSOMQeWas servlet will be invoked to handle this POST request.

- The messageArrived method of the WasQMgr object is invoked by the WASServerQm; the message is retrieved from the ChatClientQViaWas and put to the ChatClientQ on the ClientQm queue manager, as shown in Example 14-16 on page 614.
- When the message is executed, the queue manager will store it on the ChatSFQViaWas queue, because the server side queue manager cannot send a message to the client side queue manager; the queue manager will store any messages for ClientQm on this store and forward queue.
- On the client side, the ClientQm has a corresponding Home server queue called ChatSFQViaWas; the queue is configured to poll the corresponding store and forward queue on the ServerQm every five seconds. When it detects a message on that queue, it will pull the message and place it in the Home server queue, in this case ChatSFQViaWas.
- The queue manager will then move the message to the target queue specified by the application, in this case ChatClientQ.
- The messageArrived method in the ClientMgr object will be invoked by the queue manager; it will retrieve the message and display it in the output area of the GUI window.

Note that messages are travelling over two connections using two different protocols. The messages between ClientQM and WASServerQM use HTTP, but HTTP is used between ServerQM and WASServerQm.

14.8 Setting up the ChatRoom queue managers

This section describes how to set up the three queue managers used in the chat room application.

Note that the queue managers can be set up on one system, two or three systems.

14.8.1 Preparing for setup

Creation and configuration of the queue managers is performed using the Explorer tool of the ES02 support pac.

Note: At the time this chapter was written, the ES02 support pac was generally available as Version1.23. However, for defining connections to a queue manager running in WebSphere Application Server, you need to be able to specify port 80 in the connection definition. Version 1.23 does not support this. We obtained a beta copy of the Version 1.24 release which did allow this. This version should be available as of November 2001.

If the default install process is followed, then the product is installed into the C:\program files\MQe directory.

To run the ES02 support pac, either add `C:\Program Files\MQe\Java` to the environment variable *classpath* in the System property of the Control Panel folder, or in a DOS window type in the following:

```
SET CLASSPATH=C:\Program Files\MQe\Java;%CLASSPATH%
```

Open a DOS window and change the directory to the one where the ES02 support pac was installed. Let us assume it has been installed to C:\ES02.

Then, to start the ES02 Everyplace Explorer, type in:

```
MQe_explorer.exe
```

The following window appears:

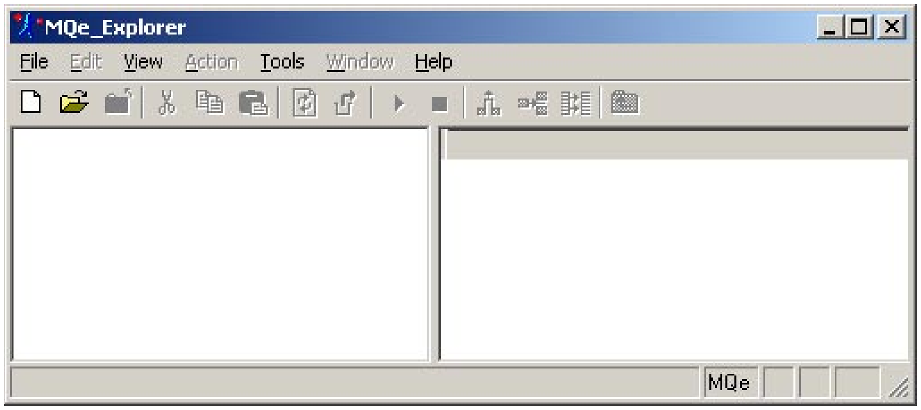


Figure 14-10 Initial MQe_Explorer window

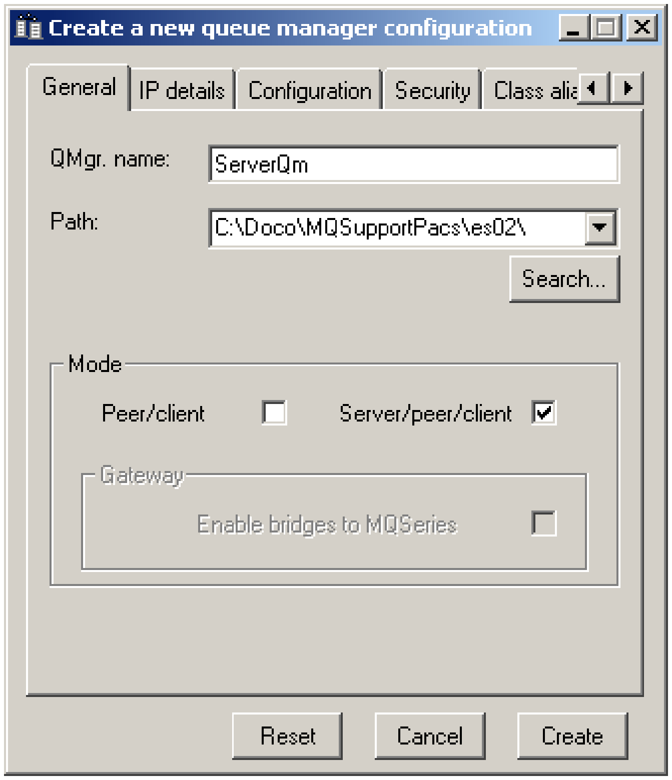
14.8.2 Creating the ServerQm queue manager

From the initial MQe_Explorer window, select **File->New->QueueManager**. A window will appear in which you will define the attributes of the server queue manager. There are several tabs; the initial one displayed is labelled General. We will call this queue manager ServerQm and define it as being a server type queue manager. To do this, follow these steps:

1. Type `ServerQm` into the field labelled QMgr.Name.

   Make sure the box next to the label 'Server/peer/client' is ticked.

   The window should look similar to the one shown in Figure 14-11.



QMgr. name: ServerQm

Path: C:\Doco\MQSupportPacs\es02\

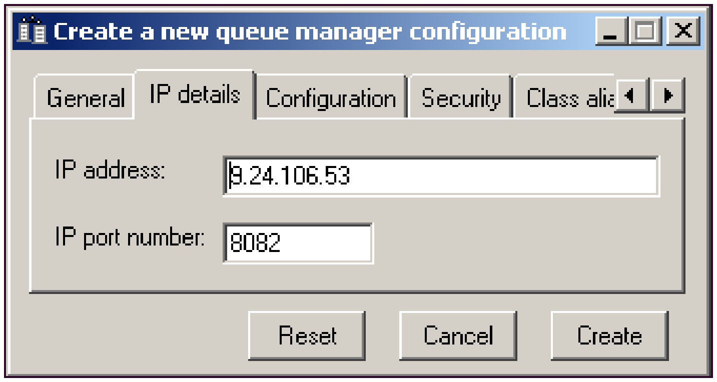
Peer/client ☐ Server/peer/client ✓

Figure 14-11 Setting the name and type of ServerQm

2. Then select the **IP details** tab.

   The IP address of the machine is displayed in the field called *IP address*, and the port that this queue manager will listen on for incoming channel requests is set to the default value used by MQSeries Everyplace, which is 8082.

The window will look similar to the one shown in Figure 14-12:



IP address: 9.24.106.53

IP port number: 8082

Figure 14-12 IP details for ServerQm

3. Then select the **Configuration** tab. This tab defines two important attributes of the queue manager:
   - The type of adapter used to handle incoming channel connections.
   - The type of adapter used to save registry information about the queue manager.

Incoming channel adapter

The field labelled Adapter, in the box titled Incoming communications, specifies the type of adapter to use. This indicates the type of protocol that the server queue manager will expect on incoming connections. Later on, when you define connections to this server queue manager, you will need to specify the type of adapter used to connect. It must match the value you specify here.

In this case, use the default adapter called:

```
com.ibm.mqe.adapters.MQeTcpipHistoryAdapter
```

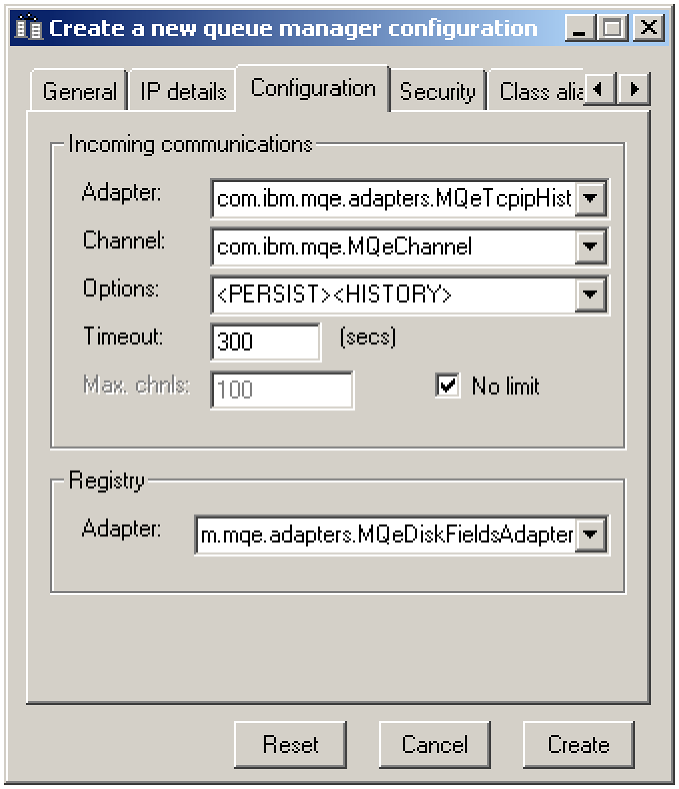
Registry Adapter

MQSeries Everyplace saves information about the queue manager to a registry. In this case, use the default adapter called:

```
com.ibm.mqe.adapters.MQeDiskFieldsAdapter
```

This is a supplied default adapter which saves registry information to disk.

The window should look similar to the one shown in Figure 14-13.



Incoming communications

Adapter: com.ibm.mqe.adapters.MQeTcpipHist

Channel: com.ibm.mqe.MQeChannel

Options: <PERSIST><HISTORY>

Timeout: 300 (secs)

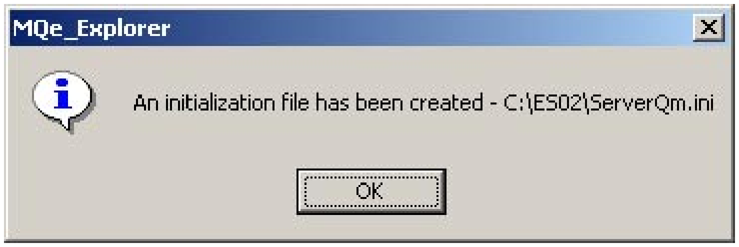
Max. chnls: 100 ☑ No limit

Registry

Adapter: m.mqe.adapters.MQeDiskFieldsAdapter

Figure 14-13 Configuration details for ServerQm

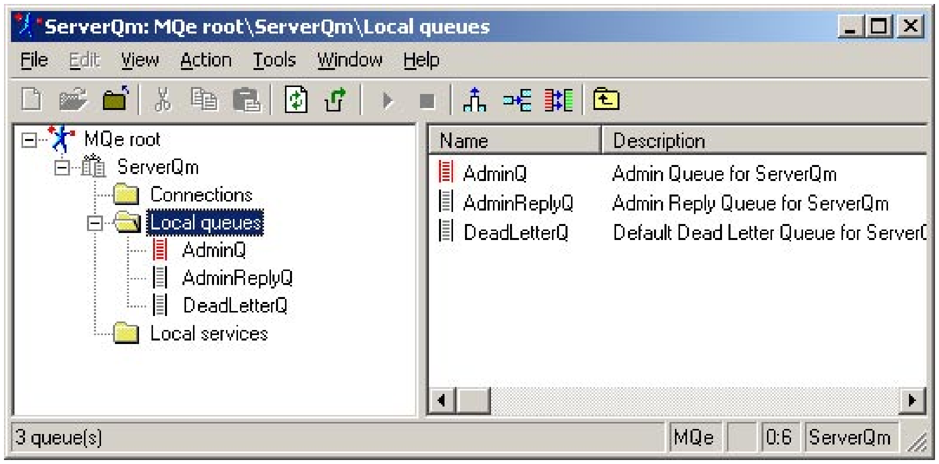
4. Click the **Create** button to now create the server side queue manager. Note that there are many other tabs which are not explained here, as all the defaults are used. A window confirming creation of the queue manager appears, similar to the one shown in Figure 14-14:



An initialization file has been created - C:\ES02\ServerQm.ini

Figure 14-14 ServerQm creation confirmation

The queue manager has now been created, and is in fact running. The MQe_Explorer window will now have an object called MQeRoot with a plus sign beside it. Click the plus sign to expand the object tree under MQRoot, and you will see an object for the ServerQM you have just created. Continue to expand the objects under ServerQm and you will see the default queues that have been set up. The display will be similar to that shown in Figure 14-15:



Name	Description
AdminQ	Admin Queue for ServerQm
AdminReplyQ	Admin Reply Queue for ServerQm
DeadLetterQ	Default Dead Letter Queue for ServerQ

Figure 14-15 Expanded tree view of ServerQm

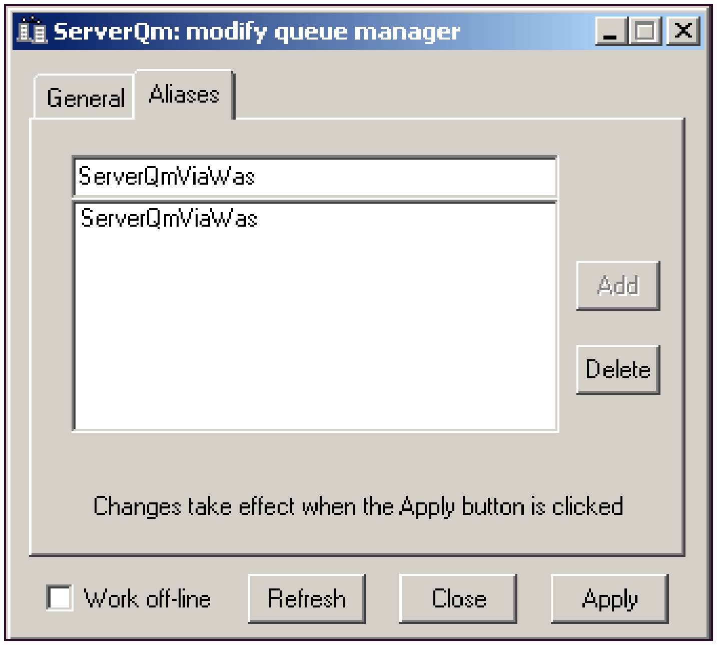
Also, in Figure 14-14 on page 624 the confirmation window shows the location and name of the .ini file where initial configuration information used to start the queue manager is stored. To start the queue manager if it is not running, you must locate this .ini file. The process is to run MQe_Explorer.exe from a DOS prompt, then select **File->Open**, locate the ServerQm.ini file and select it. The queue manager will then be started.

Note: An alternate way to start the queue MQe Explorer may be through a Windows environment. Simply double-click the **Me_ExplorerN** icon located on the desktop or in the installation directory.

Adding an alias to ServerQm

An alias of ServerQmViaWas for ServerQm is used in this application. An alias can only be added once the queue manager is defined. However, it is possible for aliases to be changed thereafter.

1. Right-click the **ServerQm** object in the tree, then select **Properties**. A window appears; click the **Aliases** tab. Type in the alias name `ServerQmViaWas`, and click the **Add** button. The window should now look similar to the one shown in Figure 14-16.



ServerQmViaWas

ServerQmViaWas

Changes take effect when the Apply button is clicked

Figure 14-16 Defining a queue manager alias

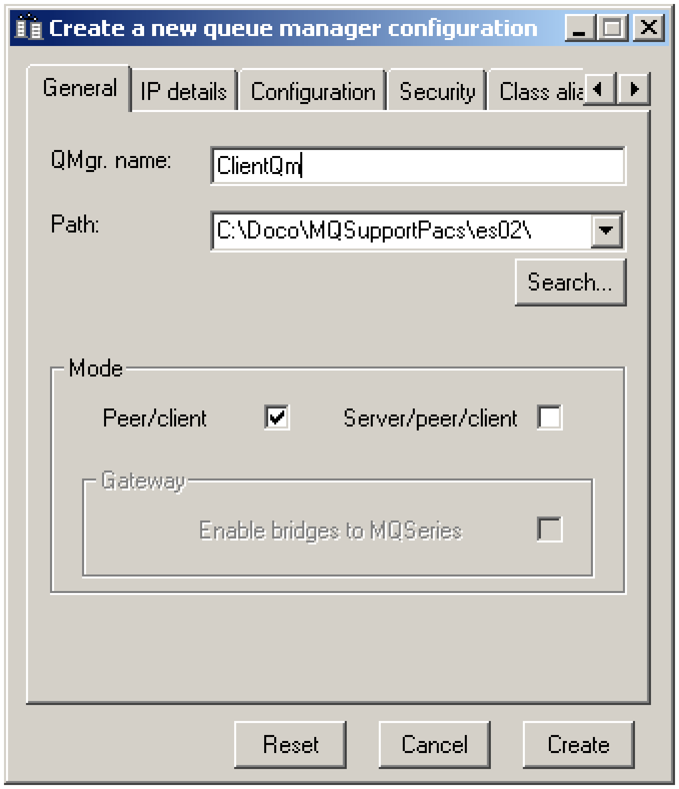
2. Click the **Apply** button to implement the change.

14.8.3 Creating the ClientQm queue manager

The process for defining the client side queue manager is very similar to the process for the server side queue manager.

1. Start another MQe_Explorer session from the DOS prompt, and from the initial MQe_Explorer window, select **File->New->QueueManager**. Enter the name of the client queue manager as `ClientQm` in the field labelled QMgr.Name.
2. Click the box next to the label Peer/client. This will mean that no listener will be set up for this queue manager, as it is to operate in client mode.

   The window should look like the one shown in Figure 14-17.



QMgr. name: ClientQm

Path: C:\Doco\MQSupportPacs\es02\

Mode

Peer/client ☑ Server/peer/client ☐

Gateway

Enable bridges to MQSeries ☐

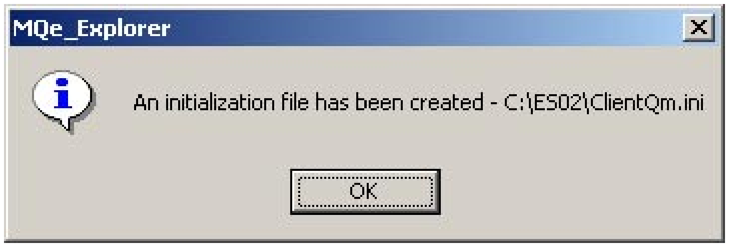
Figure 14-17 Setting the name and type of ClientQm

3. Then select the **IP details** tab.

   The IP address for this client queue manager will be the IP address of the machine on which the queue manager is defined. The port is greyed out, as no listener will be running.

   In this example, we are not using peer type channels, so there will be no connections established to this queue manager. Thus, the communications adapter can just be left to the default setting.

4. Click the **Create** button to now create the client side queue manager. A window confirming the creation of the queue manager appears (see Figure 14-18).



An initialization file has been created - C:\ES02\ClientQm.ini

Figure 14-18 ClientQm creation confirmation

The queue manager has now been created, and is in fact running. As for the server queue manager, you can expand the tree in the MQe_Explorer window to view the default objects.

14.8.4 Configuring the WASServerQm queue manager

The process to configure the WASServerQm queue manager is as follows.

1. Start another MQe_Explorer session from the DOS prompt, and from the initial MQe_Explorer window, select **File->New->QueueManager**. Enter the name of the queue manager as `WASServerQm` in the field labelled QMgr.Name.
2. Check that the box next to the text Server/peer/client is selected.
3. Then select the **IP details** tab. Enter the IP address of the machine. Enter a port number. When the queue manager is running in WebSphere Application Server, the queue manager will not have a listener running, so it will not be using the port you specify here in any case. However, it is useful to specify a valid port. Doing so means that you can run this queue manager outside of WebSphere Application Server to verify connections between it and other queue managers, and also to enable testing without WebSphere in the mix. If running this queue manager on the same machine as ServerQm, be sure to specify a different port, for example 8083.
4. Select the **Configuration** tab. As WASServerQM will be running in WebSphere Application Server, it will be expecting HTTP type communications. In the field labelled Adapter, select this adapter:

```
com.ibm.mqe.adapters.MQeTcpipHttpAdapter
```

5. In the Options field, select **None**. It is most important to select this option here, otherwise connections to this queue manager will not work.
6. Click the **Create** button to now create the queue manager. A window confirming creation of the queue manager will appear.

14.8.5 Creating connections

Prior to defining the queues, you must create the connections between the queue managers. The creation of remote queue definitions is not possible with the ES02 Explorer tool unless a connection of that name exists.

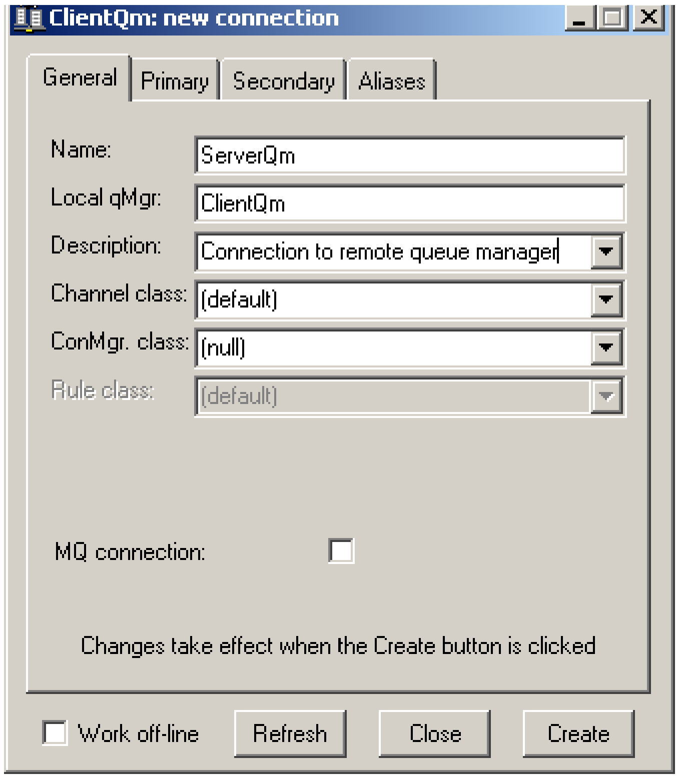
Connection definitions on ClientQm

Using MQe_Explorer, start the ClientQM queue manager.

ClientQm to ServerQm

1. Right-click the **Connection** object in the tree, and select **New connection**.
2. In the window that is displayed, enter `ServerQM` into the field labelled Name.

   The window should look similar to this:



Name: ServerQm
Local qMgr: ClientQm
Description: Connection to remote queue manager
Channel class: (default)
ConMgr. class: (null)
Rule class: (default)
MQ connection: ☐
Changes take effect when the Create button is clicked

Figure 14-19 Defining the connection to remote queue manager

3. Then click the tab labelled **Primary**. This tab is used to specify the IP address where the remote queue manager that you want to connect to is located.

There are three fields to change here.

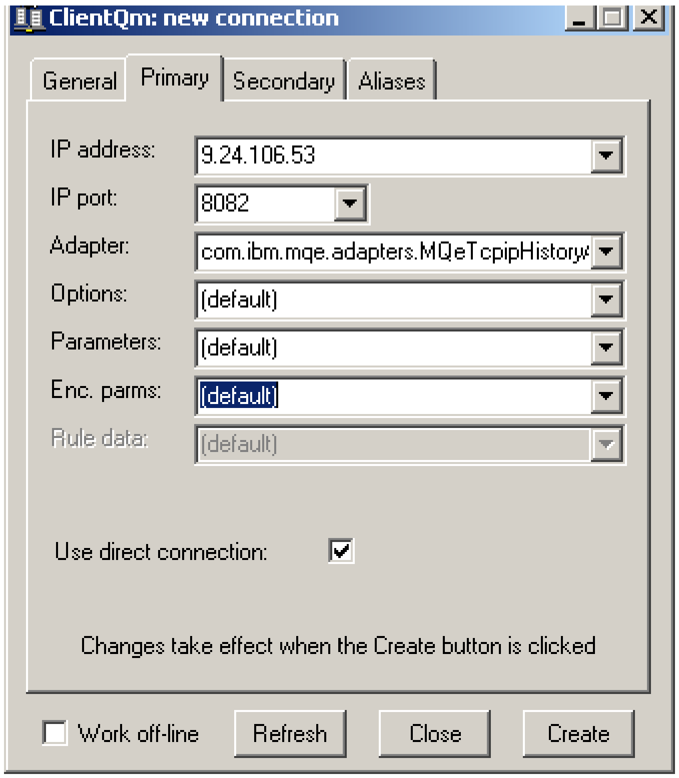
a. Type in the IP address of the machine on which you have defined the server queue manager. It could be that you have defined the client and server queue manager on the same machine, in which case you could just enter the IP address `127.0.0.1`, which is the traditional loop back address. However, it is recommended that you specify the IP address.

b. In the field labelled Port, type in the port that ServerQm is listening on. In the description above for setting up the server queue manager, we used the default value of 8082. Type `8082` into this field.

c. In the field labelled Adapter, select from the drop down menu the same adapter you specified when setting up the server queue manager. In this case, select the adapter called:

   `com.ibm.mqe.adapters.MQeTcpipHistoryAdapter`

This is a supplied adapter, which will result in the messages flowing between the two queue managers using standard TCP/IP. The window should look similar to the one shown in Figure 14-20 on page 631.


ClientQm: new connection

General | Primary | Secondary | Aliases

IP address: 9.24.106.53

IP port: 8082

Adapter: com.ibm.mqe.adapters.MQeTcpipHistory

Options: (default)

Parameters: (default)

Enc. parms: (default)

Rule data: (default)

Use direct connection: ✓

Changes take effect when the Create button is clicked

Figure 14-20 Defining the location of the remote queue manager

4. Then click the **Create** button to create the connection. No confirmation window is displayed, but you can view the connection definition in the MQe_Explorer window.

ClientQm to WASServerQm

1. Right-click the connection object in the tree, and select **New connection**.
2. In the window that is displayed, enter `WASServerQM` into the field labelled Name.
3. Then click the tab labelled **Primary**. This tab is used to specify the IP address where the remote queue manager that you want to connect to is located.

   There are five fields to change here.

   a. Type in the IP address of the machine where the WebSphere Application Server is running.
   b. In the field labelled Por', type in `80` as the port value, as this is the default port for HTTP traffic the HTTP Server will be listening on. Note that version

1.23 and earlier of the ES02 support pac do not allow a value of 80 to be entered into this field.

c. In the field labelled Adapter, select from the drop-down menu the same adapter you specified when setting up the queue manager to run in WebSphere. In this case, select the adapter called:

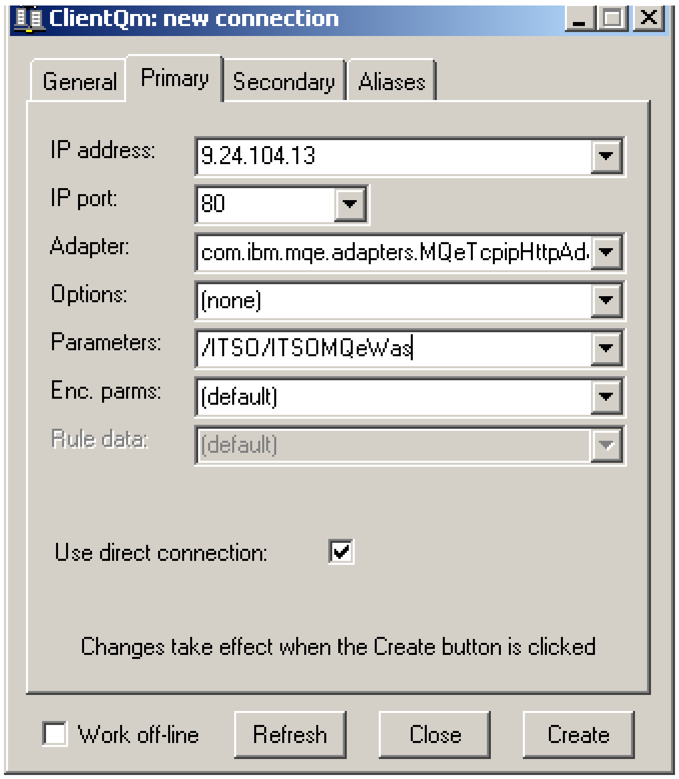
```
com.ibm.mqe.adapters.MQeHTTPAdapter
```

This is a supplied adapter, which will result in the messages flowing between the two queue managers using the HTTP protocol.

d. In the field labelled Options, select **None** from the drop-down menu.

e. In the field labelled Parameters, type in `/ITSO/ITSOMQeWas`, which is the URL that will invoke the servlet in WebSphere.

The window should look like that shown in Figure 14-21:



IP address: 9.24.104.13
IP port: 80
Adapter: com.ibm.mqe.adapters.MQeTcpipHttpAd
Options: (none)
Parameters: /ITSO/ITSOMQeWas
Enc. parms: (default)
Rule data: (default)
Use direct connection: ✓
Changes take effect when the Create button is clicked
Work off-line

Figure 14-21 Defining a connection using HTTP adapter

4. Then click the **Create** button to create the connection. No confirmation window is displayed.

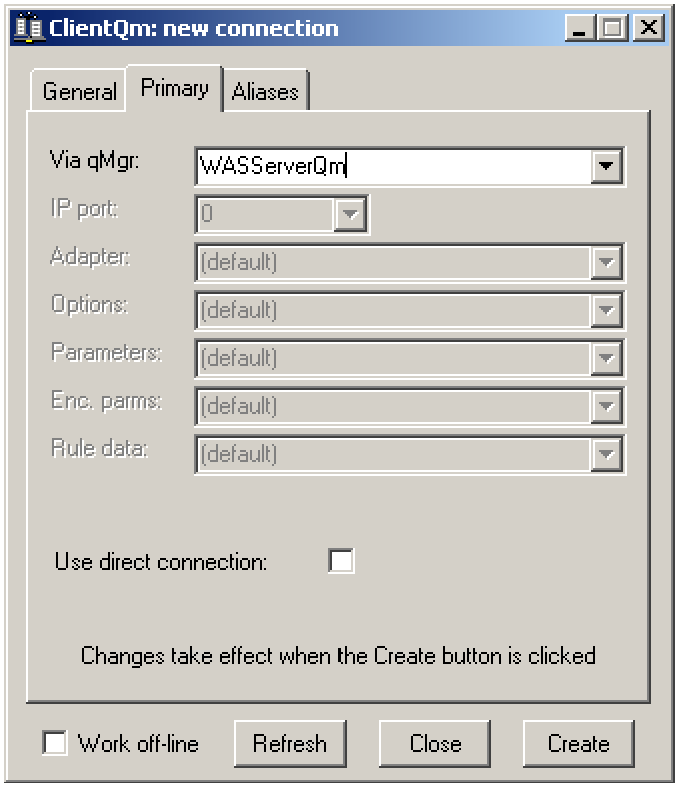
ClientQm to ServerQmViaWas

1. Right-click the **connection** object in the tree, and select **New connection**.
2. In the window that is displayed, enter `ServerQmViaWas` into the field labelled Name.
3. Then click the tab labelled **Primary**.

   The connection to ServerQmViaWas is in reality a connection to the queue manager called ServerQm, and messages are to be sent via the queue manager running in WebSphere, an indirect connection. This is referred to as *via Routing* in MQSeries Everyplace.

   To set this up, select the checkbox labelled **Direct connection** at the bottom of the window. The window will now change, with most fields greyed out. Enter `WASServerQm` into the field labelled Primary, to indicate that messages destined for ServerQmViaWas go through the WASServerQm connection.

   The window will look like this:



Via qMgr: WASServerQm
IP port: 0
Adapter: (default)
Options: (default)
Parameters: (default)
Enc. parms: (default)
Rule data: (default)
Use direct connection: ☐
Changes take effect when the Create button is clicked

Figure 14-22 Defining an indirect connection

4. Then click the **Create** button to have the connection created. No confirmation window is displayed.

Connection definitions on ServerQm

Using MQe_Explorer, start the ServerQm queue manager.

ServerQm to WASServerQm

1. Right-click the **connection** object in the tree, and select **New connection**.
2. In the window that is displayed, enter `WASServerQM` into the field labelled Name.
3. Then click the tab labelled **Primary**. This tab is used to specify the IP address where the remote queue manager that you want to connect to is located.

   There are three fields to address here.

   a. Type in the IP address of the machine where the WebSphere Application Server is running.

   b. In the field labelled Port, type in `80` as the port value, as this is the default port for HTTP traffic the HTTP Server will be listening on. Note that version 1.23 and earlier of the ES02 support pac do not allow a value of 80 to be entered into this field.

   c. In the field labelled Adapter, select from the drop-down menu the same adapter you specified when setting up the queue manager to run in WebSphere. In this case, select the adapter called:

   `com.ibm.mqe.adapters.MQeTcpipHttpAdapter`

   d. In the field labelled Options, select **None** from the drop-down menu.

   e. In the field labelled Parameters, type in `/ITSO/ITSOMQeWas` which is the URL that will invoke the servlet in WebSphere.
4. Then click the **Create** button to create the connection. No confirmation window is displayed.

Connection definitions on WASServerQm

Using MQe_Explorer, start the WASServerQm queue manager.

WASServerQm to ServerQm

1. Right-click the **connection** object in the tree, and select **New connection**.
2. In the window that is displayed, enter `ServerQM` into the field labelled Name.
3. Then click the tab labelled **Primary**. This tab is used to specify the IP address where the remote queue manager that you want to connect to is located.

There are three fields to change here.

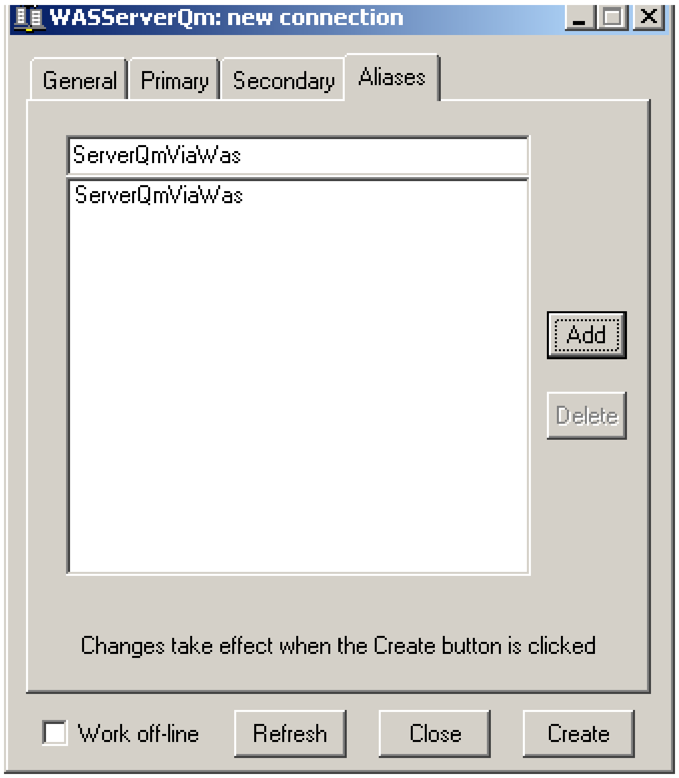
a. Type in the IP address of the machine where the ServerQM queue manager is running.

b. In the field labelled Port, type in `8082` as the port value, the port that the ServerQm is listening on.

c. In the field labelled Adapter, select from the drop-down menu the same adapter you specified when setting up the ServerQm queue manager. In this case, select the adapter called:

`com.ibm.mqe.adapters.MQeTcpipHistoryAdapter`

4. The chat room application will be sending messages to WASServerQm that are destined for ServerQmViaWas, which is an alias for ServerQm. To have these messages forwarded, we use the alias capability of MQSeries Everyplace. Click the **Alias** tab. Type in `ServerQmViaWas` and click the **Add** button. The window should look like the one shown in Figure 14-23:



ServerQmViaWas

ServerQmViaWas

Changes take effect when the Create button is clicked

Figure 14-23 Defining an alias on a connection

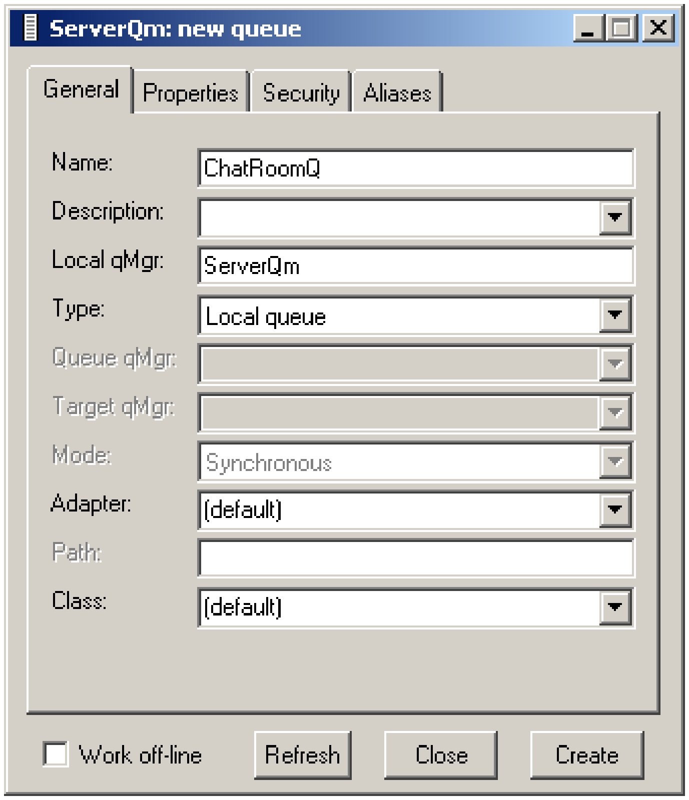
5. Then click the **Create** button to create the connection. No confirmation window is displayed.

14.8.6 Defining ServerQm queues

The following queues need to be defined.

Local queue: ChatRoomQ

1. From the expanded tree view, right-click the object labelled **Local Queues**, then select **New Queue**. A window will appear, in which to enter the details of the queue you wish to define.
2. In the tab labelled General, type `ChatRoomQ` into the field labelled Name. The window should look similar to the one shown in Figure 14-24.



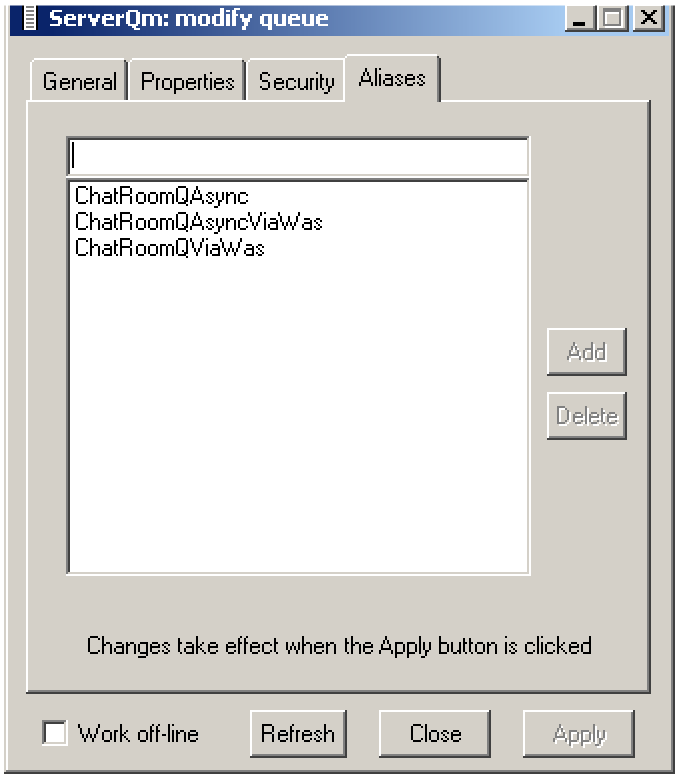
Name: ChatRoomQ
Description:
Local qMgr: ServerQm
Type: Local queue
Queue qMgr:
Target qMgr:
Mode: Synchronous
Adapter: (default)
Path:
Class: (default)

Figure 14-24 Naming the queue to be created

3. Then click the tab labelled **Aliases**. In the input text box of this tab, type in `ChatRoomQAsync`, then click the **Add** button. Add two further entries:

`ChatRoomQViaWas` and `ChatRoomQAsyncViaWas`. These alias entries tell the ServerQm that any messages it receives for these queues are to be placed on the ChatRoomQ queue.

The window should look like that shown in Figure 14-25:



ChatRoomQAsync
ChatRoomQAsyncViaWas
ChatRoomQViaWas

Changes take effect when the Apply button is clicked

Figure 14-25 Setting up alias names for this queue

4. Then click the **Create** button. No confirmation window is displayed, but the Create Queue window is redisplayed, allowing you to define another queue if required.

Remote queue: ChatClientQViaWas

1. You can use the create queue window still displayed from the previous step, or, if you closed that window, then as before from the expanded tree view, right-click the object labelled **Local Queues**, then select **New Queue** and a window will appear.
2. In the tab labelled General, type `ChatClientQViaWas` into the field labelled Name. The window will look like that shown in Figure 14-26 on page 638.

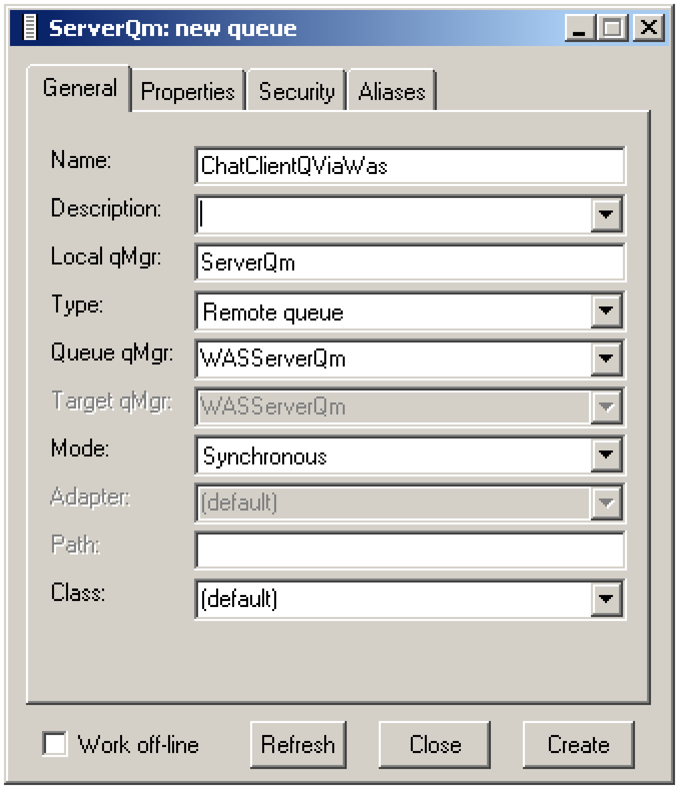
There are two more fields to change here.

a. Change the queue type to Remote by selecting that value from the drop-down menu in the field labelled Type.

b. In the field labeled Queue qMgr, type in the name `WASServerQm`.

c. Check that the field labelled Mode is set to **Synchronous**.

The window should look like this:



Name: ChatClientQViaWas
Description:
Local qMgr: ServerQm
Type: Remote queue
Queue qMgr: WASServerQm
Target qMgr: WASServerQm
Mode: Synchronous
Adapter: (default)
Path:
Class: (default)

Figure 14-26 Define a remote queue

3. Then click the **Create** button. No confirmation window is displayed, but the Create Queue window is redisplayed allowing you to define another queue if required.

Store and Forward queue: ChatSFQ

1. You can use the create queue window still displayed from the previous step, or, if you closed that window, then as before from the expanded tree view, right-click the object labelled **Local Queues**, then select **New Queue**. A

window will appear, in which to enter the details of the queue you wish to define.

2. Under the tab labelled General, type `ChatSFQ` into the field labelled Name.
3. Most importantly, change the queue type to Store and forward queue, by selecting that value from the drop-down menu in the field labelled Type.
4. This queue is used to store messages that will be 'pulled' by remote client type queue managers, in our case the client side queue manager. The server side queue manager needs to know that it is to use this queue to store messages destined for the client side queue manager. To do this, select the **Targets** tab. In the window type in the value `ClientQm` and click the **Add** button. This display will look similar to that shown in Figure 14-27.

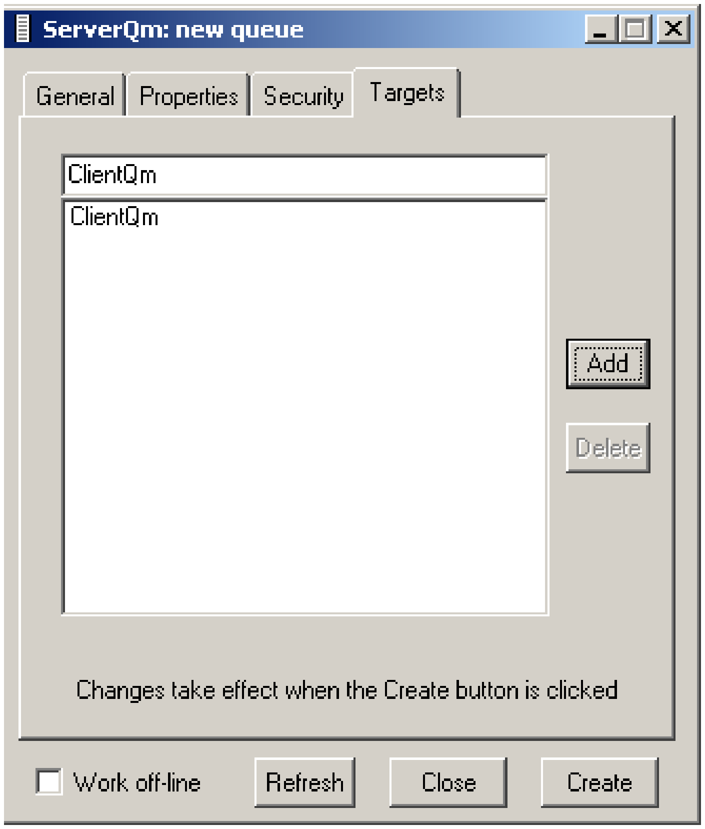


Figure 14-27 Adding the client queue manager as a target to the store and forward queue

5. Then click the **Create** button and the queue will be defined. No confirmation window is displayed.

14.8.7 Define ClientQm queues

Local queue: ChatClientQ

This is the same process as for setting up the local queue called ChatRoomQ on the server queue manager.

1. From the expanded tree view,right-click the object labelled Local Queues', then select **New Queue**. A window will appear, in which to enter the details of the queue you wish to define.
2. In the tab labelled General, type `ChatClientQ` into the field labelled Name.
3. Then click the Create button. No confirmation window is displayed, but the Create Queue window is redisplayed, allowing you to define another queue if required.

Remote Queue: ChatRoomQ

1. You can use the create queue window still displayed from the previous step, or, if you closed that window, then as before from the expanded tree view, right-click the object labelled **Local Queues**, then select **New Queue**. A window will appear, in which to enter the details of the queue you wish to define.
2. In the tab labelled General', type ChatRoomQ into the field labelled Name'.

   There are three fields to change here.

   a. Change the queue type to **Remote queue**, by selecting that value from the drop-down menu in the field labelled Type.
   b. In the box labelled Queue qMgr, select from the drop-down menu the name of the remote queue manager, in this case, **ServerQm**.
   c. Change the mode to **Synchronous**, by selecting that value from the drop-down menu in the field labelled Mode.
3. Then click the **Create** button. No confirmation window is displayed, but the Create Queue window is redisplayed allowing you to define another queue if required.

Remote Queue: ChatRoomQAsync

1. You can use the create queue window still displayed from the previous step, of if you closed that window, then as before from the expanded tree view,right-click the object labelled **Local Queues**, then select **New Queue**. A window will appear, in which to enter the details of the queue you wish to define.

2. In the tab labelled General, type `ChatRoomQAsync` into the field labelled Name.

   There are three fields to change here.

   a. Change the queue type to **Remote queue**, by selecting that value from the drop-down menu in the field labelled Type.

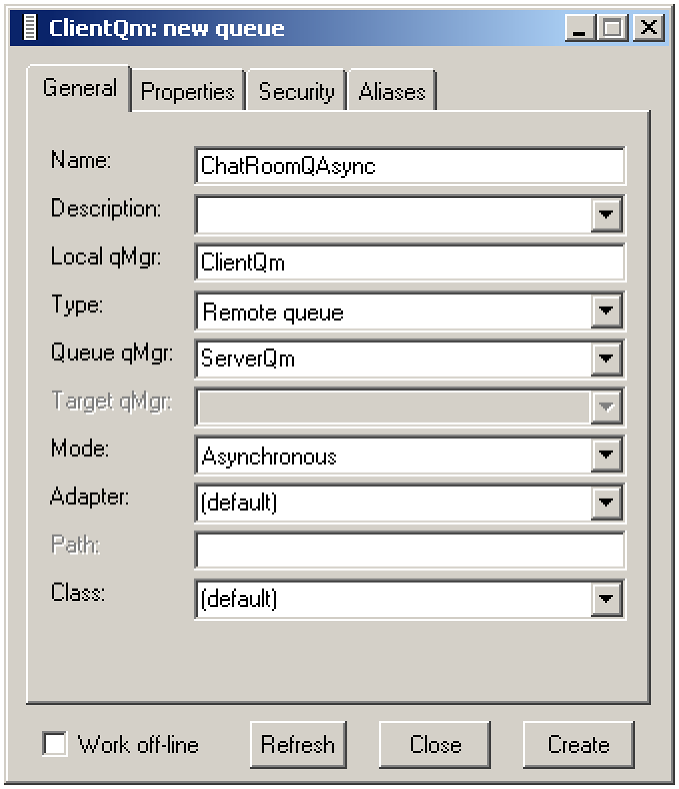
   b. In the box labelled Queue qMgr, select from the drop-down menu the name of the remote queue manager, in this case, **ServerQm**.

   c. Change the mode to **Asynchronous**, by selecting that value from the drop-down menu in the field labelled Mode.

   When the chat room application tries to send a message, it first tries to put the message to the local queue called ChatRoomQ on the server queue manager.

   If there is no connection, the application will put the message to this queue. Because the mode is set to Asynchronous, the queue manager will store the message locally, and then send it to the server queue manager when the connection becomes available.

   If the mode was set to synchronous, and the connection was down, then the put message would fail.


ClientQm: new queue

Name: ChatRoomQAsync
Description:
Local qMgr: ClientQm
Type: Remote queue
Queue qMgr: ServerQm
Target qMgr:
Mode: Asynchronous
Adapter: (default)
Path:
Class: (default)

Figure 14-28 Define remote queue on the client

3. Then click the **Create** button and the queue will be defined. No confirmation window is displayed.

Remote Queue: ChatRoomQViaWas

1. In the window for defining a new queue, under the tab labelled General, type `ChatRoomQViaWas` into the field labelled Name.

   There are three fields to change here.

   a. Change the queue type to **Remote queue**, by selecting that value from the drop-down menu in the field labelled Type.

   b. In the box labelled Queue qMgr, select from the drop-down menu the name of the remote queue manager, in this case, **WASServerQm**.

   c. Change the mode to **Synchronous**, by selecting that value from the drop-down menu in the field labelled Mode.

2. Then click the **Create** button. No confirmation window is displayed, but the Create Queue window is redisplayed allowing you to define another queue if required.

Remote Queue: ChatRoomQAsyncViaWas

1. In the window for defining a new queue, under the tab labelled General, type `ChatRoomQAsyncViaWas`.

   There are three fields to change here.

   a. Change the queue type to **Remote queue**, by selecting that value from the drop-down menu in the field labelled Type.

   b. In the box labelled Queue qMgr, select from the drop-down menu the name of the remote queue manager, in this case, **WASServerQm**.

   c. Change the mode to **Asynchronous**, by selecting that value from the drop-down menu in the field labelled Mode.

2. Then click the **Create** button. No confirmation window is displayed, but the Create Queue window is redisplayed allowing you to define another queue if required.

Home Server Queue: ChatSFQ

1. In the window for defining a new queue, under the tab labelled General, type `ChatSFQ` into the field labelled Name.

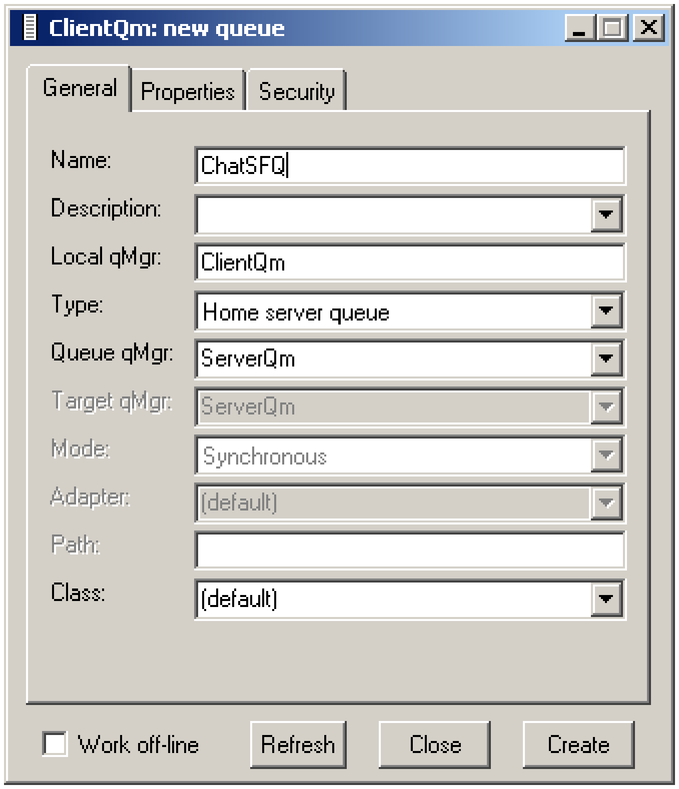
   There are two fields to change here.

   a. Change the queue type to **Home Server queue**, by selecting that value from the drop-down menu in the field labelled Type.

   b. In the box labelled Queue qMgr, select from the drop-down menu the name of the remote queue manager, in this case, **ServerQm**.

   Notice that the mode is set to Asynchronous, as the client queue manager will poll the remote corresponding store and forward queue on the remote queue manager, and pull any messages it find there to this queue.

   The window will look like that shown in Figure 14-29.



Name:	ChatSFQ
Description:	
Local qMgr:	ClientQm
Type:	Home server queue
Queue qMgr:	ServerQm
Target qMgr:	ServerQm
Mode:	Synchronous
Adapter:	(default)
Path:	
Class:	(default)

Figure 14-29 Defining the home server queue on the client

2. Then click the **Properties** tab. There is one field here to be changed, called the *Time interval*.

   Specify a value here in milliseconds. Specifying a value greater than zero tells the client queue manager how often to automatically poll the server side queue. Set this to some reasonable value, such as 5000, which will mean that a check occurs every 5 seconds.

3. Then click the **Create** button and the queue will be defined. No confirmation window is displayed.

Home Server Queue: ChatSFQViaWas

1. In the window for defining a new queue, under the tab labelled General, type `ChatSFQViaWas` into the field labelled Name.

   There are two fields to change here.

   a. Change the queue type to **Home Server queue**, by selecting that value from the drop-down menu in the field labelled Type.

   b. In the box labelled Queue qMgr, select from the drop-down menu the name of the remote queue manager, in this case, **WASServerQm**.

2. Then click the **Properties** tab. There is one field here to be changed called the Time interval.

   Specify a value here in milliseconds. Specifying a value here greater than zero tells the client queue manager how often to automatically poll the server side queue. Set this to some reasonable value, such as 5000, which will mean that a check occurs every 5 seconds.

3. Then click the **Create** button and the queue will be defined. No confirmation window is displayed.

14.8.8 Define WASServerQm queues

Local: ChatClientQViaWas

1. In the window for defining a new queue, under the tab labelled General, type `ChatClientQViaWas` into the field labelled Name.

   The mode should default to `synchronous`, and the type to `local`.

2. Then click the **Create** button. No confirmation window is displayed, but the Create Queue window is redisplayed, allowing you to define another queue if required.

Store and Forward queue: ChatSFQViaWas

1. In the window for defining a new queue, under the tab labelled General, type `ChatSFQViaWas` into the field labelled Name.
2. Most importantly, change the queue type to **Store and forward queue**, by selecting that value from the drop-down menu in the field labelled Type.
3. Select the **Targets** tab. In the window, type in the value `ClientQm` and click the **Add** button.
4. Then click the **Create** button to define the queue. No confirmation window is displayed.

14.8.9 Java Swing setup

The chat room application uses the Java swing classes to display windows. These classes are located in a .jar file called SwingAll.jar. Search your system for one of these files, then add it to the classpath. For example:

```
set CLASSPATH=%CLASSPATH%;c:\Program Files\sqllib\java\swingall.jar
```

Be sure that the MQSeries Everyplace Java classes are also accessible through the classpath, as mentioned in "Preparing for setup" on page 621.

14.8.10 Chatroom application setup

As mentioned, the chat room application consists of a number of packages. In this setup example, we will place the application code in the ES02 directory. Follow these steps:

1. Create a directory called itso under the ES02 directory.
2. Create a directory called mqe under the itso directory.
3. Create three directories under the mqe directory called:
   - chatclient
   - chatserver
   - chatwindow
4. Copy the chat client classes to the chat client directory.
5. Copy the chat server classes to the chat server directory.
6. Copy the chat window classes to the chat window directory.

If you obtained the sample code in a zip file, extracting the files in the zip file will create the appropriate directory structure and put the classes in the correct place.

Modify the classpath so that these classes are found when the application is run, by typing in:

```
set CLASSPATH=C:\ES02\;.;%CLASSPATH%
```

The '.' tells the system to look in the current directory.

14.8.11 Setting up the startup list

The chat room application demonstrates two ways of running applications with MQSeries Everyplace.

On the server side, the MQSeries Everyplace queue manager is started first, then the chat room application is loaded.

On the client side, the chat room application is started first, and it starts the MQSeries queue manager.

Startup list

The startup list approach demonstrates how to use the ES02 support pac so as to have an application loaded when the queue manager starts.

This is done by editing the .ini file associated with the queue manager, in this case ServerQm.ini.

Add the following to the bottom of the .ini file used for ServerQm:

```
[AppRunList]

(ascii)App1=itso.mqe.chatserver.RoomMgr
```

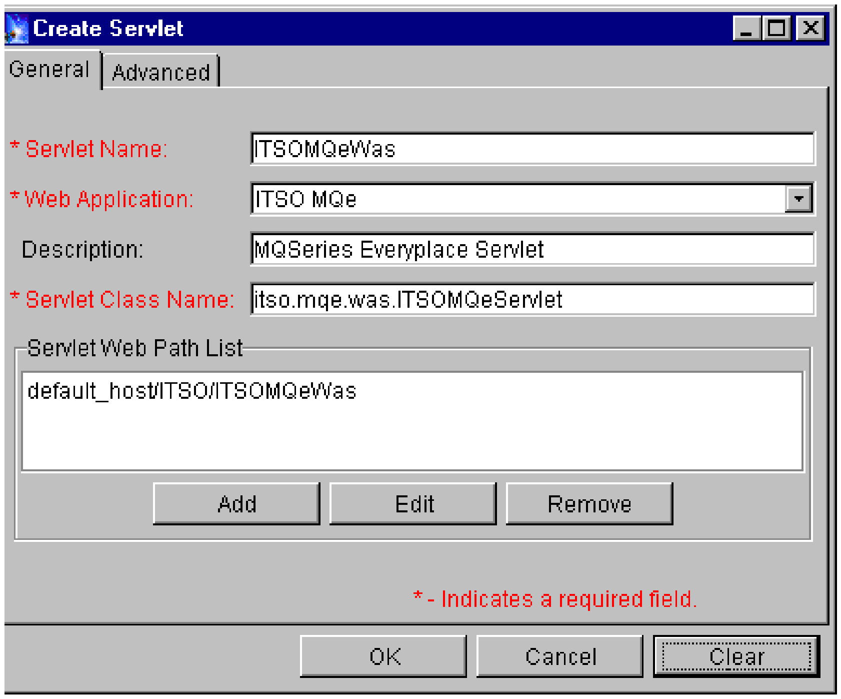
Note that more than one application can be specified, and initialization data can also be passed if required.

14.8.12 Configuring WebSphere

WebSphere Application Server will require some configuration to allow the servlet to be run. The servlet was tested in both V3.5 and V4 of WebSphere Application Server, but we will only describe here deployment of the servlet in V3.5.

Defining the servlet to WebSphere can be done in many ways; however, we chose to define a separate Web application under the default application server.

Use the WebSphere Administrative console to create a new Web application, then create a servlet definition. The window that defines the servlet should look similar to the one shown in Figure 14-30 on page 648.



* Servlet Name: ITSOMQeWas

* Web Application: ITSO MQe

Description: MQSeries Everyplace Servlet

* Servlet Class Name: itso.mqe.was.ITSOMQeServlet

Servlet Web Path List

default_host/ITSO/ITSOMQeWas

* - Indicates a required field.

Figure 14-30 Defining the Chat Room servlet

Add the following directories to the classpath for the Web application that will run the servlet:

– C:\Program Files\MQe\Java

– C:\ES02

– C:\WebSphere\AppServer\hosts\default_host\WSsamples_app\servlets

What is being added here is the location of the MQSeries Everyplace class files, and the classes used by the chat room application. The last classpath is required as the servlet that runs the queue manager and imports this package is part of the YourCo example explained in “Extending the YourCo application” on page 661. Note that you will need to copy the TotalLeaveBean class from the zip file containing the chat room application to the above directory, as this class was developed for this redbook, and is not part of the supplied WebSphere Application Server YourCo sample.

Note also that messages written out by the chat room application are written to standard output, which will appear in the default standard output file for the application server. These messages adviseyou that the queue manager is started successfully, for example.

14.8.13 Setting up property files

The following property files need to be defined.

clientChat.properties

This property file is used only on the client side, and can be used to pass the names of queues and queue managers to the application to be used in place of the defaults coded in the Java programs. A sample is shown below:

Example 14-21 Sample clientChat property file

```
iniFile=C:\\ES02\\ClientQm.ini
clientQueue=ChatClientQ
destQueue=ChatRoomQ
destQueueAsync=ChatRoomQAsync
destQMgr=ServerQm
clientQm=ClientQm
WASIpAddr=9.24.106.53
```

Note that the values shown above match the default values coded in the Java code, so a property file does not need to be used if the defaults are used.

The file should be placed in the ES02 directory.

MQe.properties

This property file is used only by the servlet running in WebSphere Application Server. This property file is used to pass in the location of the.ini file for the queue manager to be started by the servlet, and a flag to do with the YourCo example described in "Extending the YourCo application" on page 661.

The file contains just two lines:

```
IniFile=D:\\ES02\\WasServerQm.ini

YourCo=No
```

If you do plan to use the YourCo example, then set the value of the YourCo property to anything other then `Yes`. If the value is set to `Yes`,you will need to define the YourCoQuery queue in the WASServerQM, otherwise you will get an exception in the servlet. The property file should also be placed in the ES02 directory.

ejbLocation.properties

A number of parameters are required so that the bean used in WebSphere can locate the EJB of the YourCo application. These parameters are specified in this property file. The contents of the file look like this:

Example 14-22 Properties to locate EJB

```
userID=WSDEMO
password=wsdemo1
URL=jdbc:db2:SAMPLE
driver=COM.ibm.db2.jdbc.app.DB2Driver
dataSourceName=jdbc/sample
factory=com.ibm.ejs.ns.jndi.CNInitialContextFactory
accessName=Access
providerURL=iiop://9.24.104.13:900
```

The IP address in the example above needs to be the address of the system that is executing the EJBs.

This file should be placed into the ES02 directory. Note that if you do not plan to try out the YourCo example, then you do not need to set up this property file.

14.8.14 Starting the chat room application

This section describes how to start and use the chat room application.

Starting the server side

To start the server side of the chat room application, follow these steps.

1. Open a DOS window.
2. Set up the classpath as described in “Preparing for setup” on page 621.
3. Change to the ES02 directory.
4. Type in `MQe_explorere.exe`.
   a. This displays MQe Explorer window; select **File->Open**.
   b. A file dialog box appears; locate the .ini file for the server queue manager called ServerQm.ini and select it.
   c. The queue manager will start, and its objects can be viewed in the tree.
   d. It will start the chat room application; this is specified in the AppRunList stanza of the .ini file.
   e. A window will appear titled MQSeries Everyplace Server.

      Note that you cannot send any messages until the client connects.

Server side chat window

The server side chat window will initially look like this (see Figure 14-31):

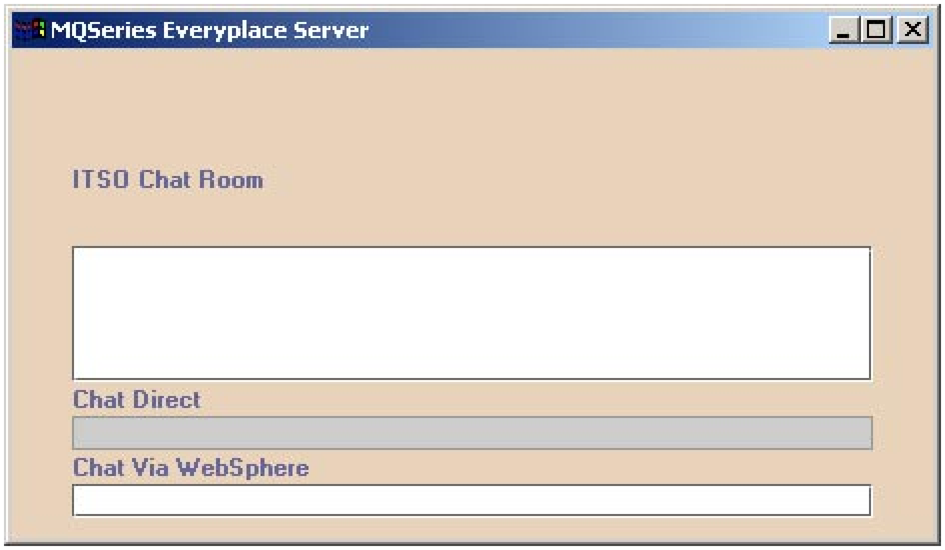


Figure 14-31 Initial server side chat window

Starting the client side

To start the client side of the chat room application, follow these steps:

1. Open a DOS window where ClientQM is set up and set up the classpath.
2. Type in `java itso.mqe.chatclient.ClientMgr`. Note that this will first start the chat room application, then the application will start the queue manager.
   – If you have set up a property file for the client side, add `-p` to the line that invokes the application; this causes the application to read the property file.
3. A window will appear titled MQSeries Everyplace Client.

Client side chat window

The client side chat window will initially look like this (see Figure 14-32):

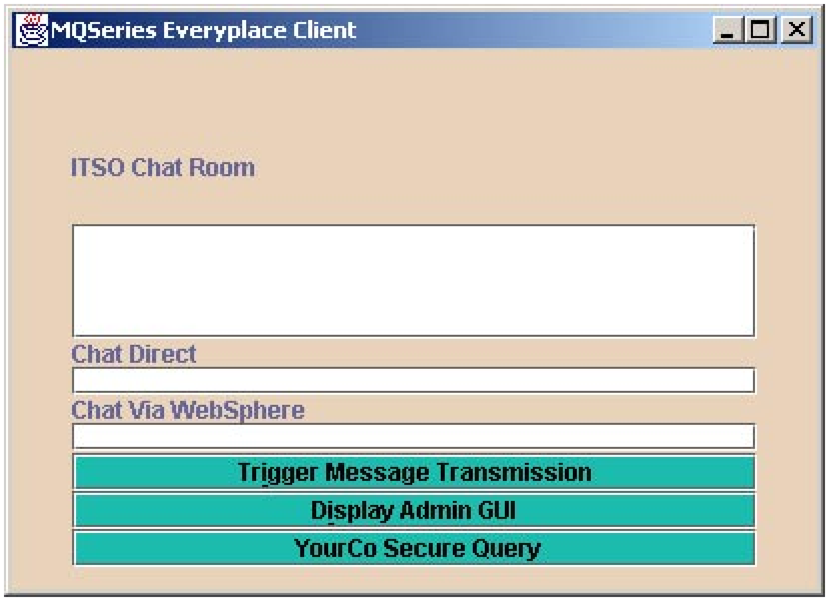


Figure 14-32 Initial client side chat window

14.8.15 Operating the chat window

Once the two windows are displayed, you are ready to begin chatting. Note that the client side must initiate the chat.

In the client window, in the input box below the text `Chat Direct`, type in some text and click **Enter**.

The text you have typed in should then appear in the server side window, in the output text area below the text `ITSO Chat Room`.

The Chat Direct input text area on the server side chat room will now be enabled for input. Type in a message and click **Enter**. There will be a slight delay before the message appears in the output text area on the client window, because the client is polling the server store and forward queue at a defined interval.

To test chatting using WebSphere, type messages into the Chat via WebSphere input boxes.

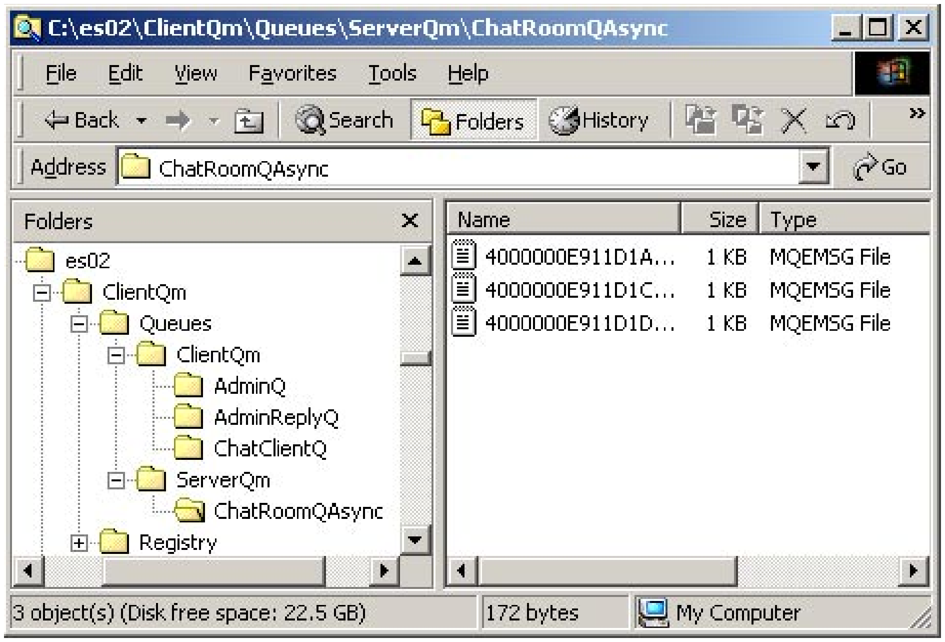
Continue to chat between the two windows as required.

14.8.16 Asynchronous chatting

To demonstrate asynchronous message transfer, the client side of the application can be run without the ServerQm or WASServerQm queue managers being active.

Start just the chat room application on the client side. In the Chat Direct input text box, type a message. The application first tries to put to the local queue on the remote server, but is not able to, as the server side queue manager is down. The application detects this, and puts the message to the remote queue called ChatRoomQAsync. MQSeries Everyplace stores the message locally.

You can verify this by looking in the directory on the file subsystem that is being used to store messages. For example, let us say that you typed in three messages; you can see that there are three files in the corresponding directory, as shown inFigure 14-33 on page 653



Name	Size	Type
4000000E911D1A...	1 KB	MQEMSG File
4000000E911D1C...	1 KB	MQEMSG File
4000000E911D1D...	1 KB	MQEMSG File

3 object(s) (Disk free space: 22.5 GB) | 172 bytes | My Computer

Figure 14-33 Messages waiting to be sent

The messages are in the ChatRoomQAsync directory.

These messages will be sent by MQSeries Everyplace when there is a connection to the server side queue manager.

You can also try stopping the application server where the servlet is defined. Then try to send a message using the Chat via WebSphere boxes. As no connection can be established, the messages will be written to the ChatRoomQAsyncViaWas queue on ClientQm.

Triggering transmission

For asynchronous message transfer to occur, the queue manager must be triggered. When triggered, the queue manager will attempt to send the queued messages. If a connection is available, the messages are sent; if the connection is down, no messages are sent.

The queue manager will not try again until triggered once more. Thus, if you now start the server side queue manager, the messages are not automatically sent.

With the chat room application, there are two ways to trigger this transfer once the server queue manager is restarted.

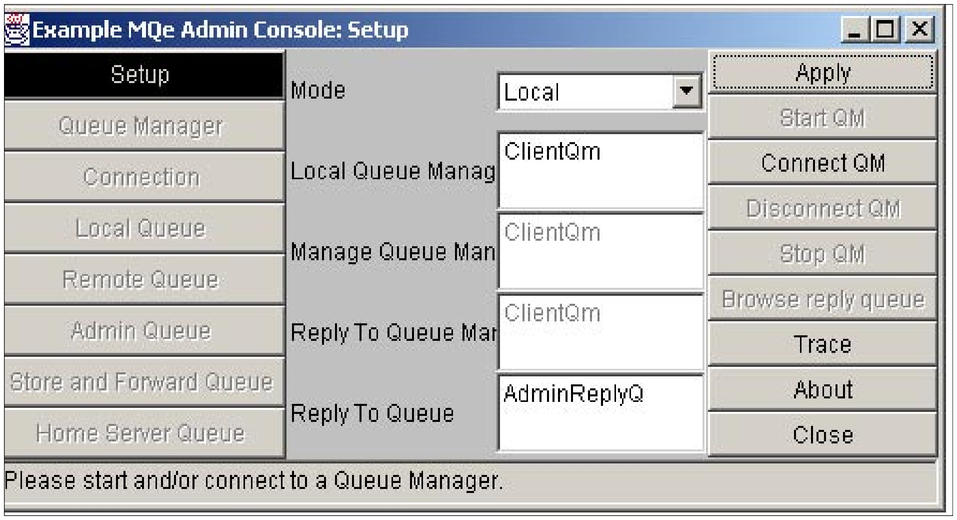
One way is to stop the client queue manager and restart it. When restarted, the trigger transmission method is invoked, which causes it to try to send any queued messages.

The second way is to leave the client chat room application running, and click the button labelled **Trigger Transmission** in the client chat window. This causes the trigger transmission method to be invoked on the queue manager, and the messages to be sent.

It is the responsibility of the application to provide some method to have a trigger transmission method issued on the queue manager, if asynchronous messaging is being used.

14.8.17 The administration GUI

In the client chat room window is a button labelled Display Admin GUI. When clicked, the following window appears:



Mode: Local
Local Queue Manag: ClientQm
Manage Queue Man: ClientQm
Reply To Queue Mar: ClientQm
Reply To Queue: AdminReplyQ
Please start and/or connect to a Queue Manager.

Figure 14-34 The administration GUI

This GUI is an example supplied with MQSeries Everyplace. Its use is described in the MQSeries Everyplace Programming Guide. It can be used to inquire, update and create definitions in the queue manager.

Trace facility

One of the most useful features is the trace option. If you click the **Trace** button, the following window appears (see Figure 14-35).

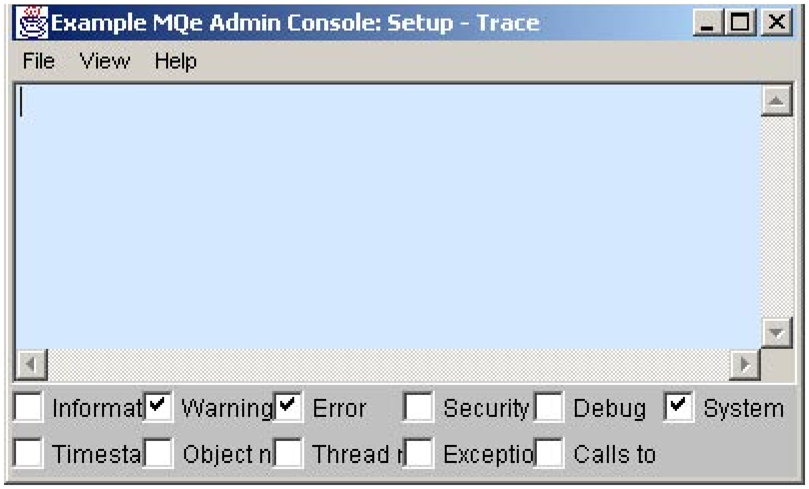


Figure 14-35 MQSeries Everyplace Trace window

This can be used to trace activity in MQSeries Everyplace. If your Java program writes messages to standard output, these will appear in this window.

14.8.18 Encryption and the stress test

As a simple example of using the encryption services supplied with MQSeries Everyplace, and as a measure to compare the respective performances of TCP/IP versus HTTP over TCP/IP for sending messages, a very simple stress test is incorporated into the chat room application.

The following section describes how to set it up, the queues required, how to run the test, and also how to set up and verify encryption.

ClientQm - Queues

1. Shut down the client side of the chat room application if it is running, and start up the ClientQM queue manager using the ES02 package.
2. Open a Create queue window.
3. Under the tab labelled General, type `StressQ` into the field labelled Name.

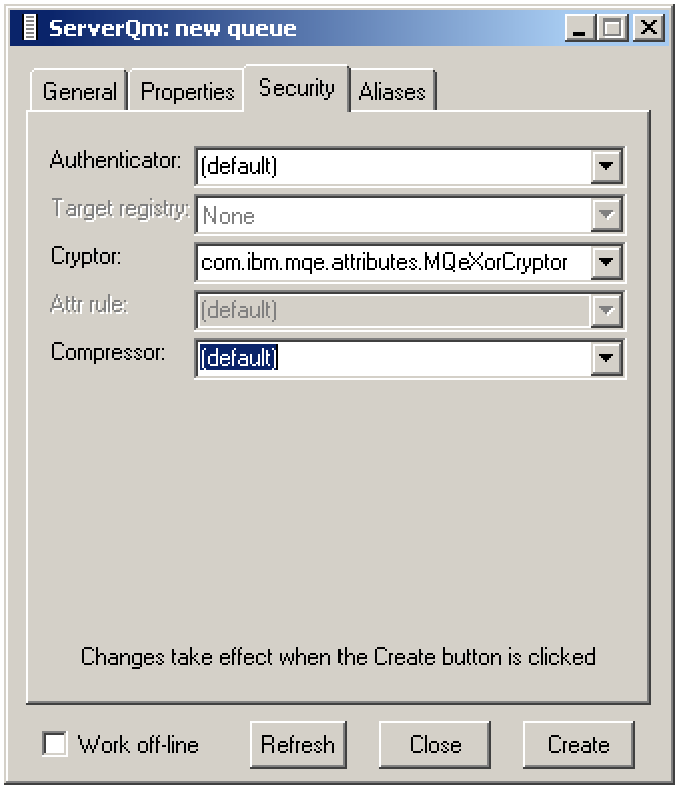
   There are three fields to change here.

   a. Change the queue type to **Remote queue** by selecting that value from the drop-down menu in the field labelled Type.

   b. In the box labelled Queue qMgr, select from the drop-down menu the name of the remote queue manager, in this case, **ServerQm**.

   c. Check that the mode is Synchronous by selecting that value from the drop-down menu in the field labelled Mode.

4. Then click the **Security** tab. In the field labelled Cryptor, select **com.ibm.mqe.attributes.MQeXorCryptor**. The window should look something like this (see Figure 14-36):



Authenticator: (default)
Target registry: None
Cryptor: com.ibm.mqe.attributes.MQeXorCryptor
Attr rule: (default)
Compressor: (default)

Changes take effect when the Create button is clicked

Figure 14-36 Selecting a cryptor adapter

5. Then click the **Create** button and the queue will be defined. No confirmation window is displayed.
6. Repeat this process to create another remote queue, but with a value of `WASServerQm` in the Queue qMgr field.

ServerQm - Queue

1. Using the ES02 package, start the ServerQm queue manager, and open the Create queue window.
2. Under the tab labelled General, type `StressQ` into the field labelled Name.

   The queue type should be local, the mode synchronous.
3. Then click the **Security** tab. In the field labelled Cryptor, select **com.ibm.mqe.attributes.MQeXorCryptor.**
4. Then click the **Create** button and the queue will be defined. No confirmation window is displayed.

WASServerQm Queue

1. Using the WebSphere Application Server console, stop the application server that is running the ITSOMQeWas servlet. Then, using the ES02 package, start the WASServerQm queue manager, and open the Create queue window.
2. Under the tab labelled General, type `StressQ` into the field labelled Name.

   The queue type should be local, the mode synchronous.
3. Then click the **Security** tab. In the field labelled Cryptor, select **com.ibm.mqe.attributes.MQeXorCryptor.**
4. Then click the **Create** button and the queue will be defined. No confirmation window is displayed.
5. Stop the Es02 package, and restart the application server in WebSphere Application Server.

Running a stress test

Start the client side of the chat room application again. Then in the Chat Direct window type in the string `Stress Test` and click **Enter**. The code in the sendMessage method of ClientMgr checks for the above string in the message to be sent. When it detects it, it will invoke the stressTest method. All this method does is send ten messages of 150 bytes each to the ServerQm. It records the time it takes for this to occur, and calculates the throughput rate. It then displays a message with the results in the chat room window.

Typing the same string in the Chat via WebSphere box sends the same set of messages to the StressQ on the WASServerQm.

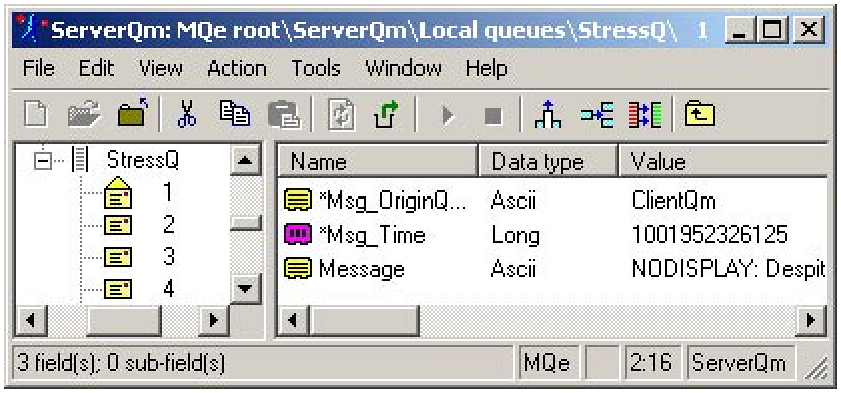
If you set up the client queue manager on one workstation, and the WASServerQm and ServerQm on some other workstation, then you can use this simple stress test to compare sending messages via TCP/IP versus via HTTP.

Encryption

Since the queues were defined using an encryption adapter, the messages have been encrypted. The messages are encrypted as they are put into the queue, and not decrypted until they are retrieved from the queue and passed to the application.

Note that even though this is a synchronous put message to a local queue on a remote queue manager, the encryption would still occur on the client side queue manager before the messages were sent.

Using the ES02 package,right-click the **StressQ** object, and you will see that it contains a number of messages. Double-click one of these messages, and you will have a display similar to the one shown in Figure 14-37:



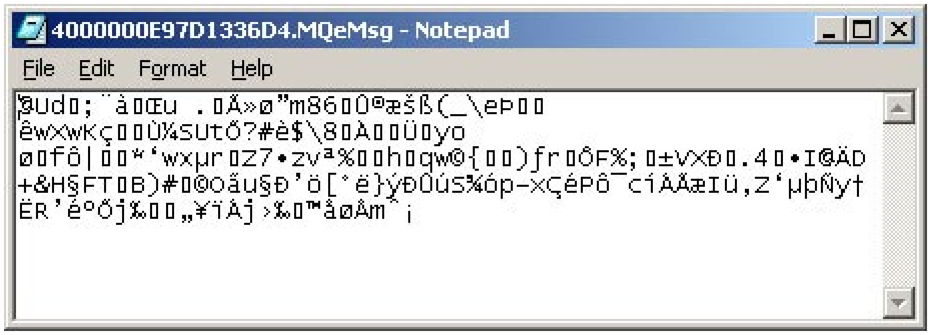
Name	Data type	Value
*Msg_OriginQ...	Ascii	ClientQm
*Msg_Time	Long	1001952326125
Message	Ascii	NODISPLAY: Despit

3 field(s); 0 sub-field(s) MQe 2:16 ServerQm

Figure 14-37 Displaying stress test messages

The text of the message is shown under the heading Value, on the line with a value of `Message` under the heading Name. The message contents can be viewed, because in this case the ES02 package has read the message, and the queue manager has decrypted it.

To check that the message is indeed encrypted, use the Windows Explorer program, and drill down to the StressQ folder in the directory where the queues associated with the ServerQm are stored. In this directory, you will see a number of files, each file representing a message. Double-click one of the messages, and it will display in a default editor. The contents will look like those shown in Figure 14-38.



```
▯Ud▯;¨à▯Œu .▯Å»ø”m86▯Ò®æšß(_\eÞ▯▯
êwxwKç▯▯Ù¼SUtÕ?#è$\8▯À▯▯Ü▯yo
ø▯fô|▯▯*‘wxµr▯Z7•zvª%▯▯h▯qw©{▯▯)ƒr▯ÔF%;▯±vxÐ▯.4▯•I@ÄD
+&H§FT▯B)#▯©oåu§Ð’ö[˚ë}ýÐÛúS%óp-xÇéPô¯cíÀÅæIü,z‘µþÑy†
ËR’ê°Õj‰▯▯„¥ïÀj›‰▯™åøÅm˜¡
```

Figure 14-38 Encrypted message contents

As the message is encrypted, the contents displayed are unintelligible.

MQSeries Everyplace comes with adapters that provide much stronger levels of encryption than the one used in this example, though there are some additional steps involved in using them, which are explained in Chapter 8 in the *MQSeries Everyplace Programming Guide*, SC34-5845.

TCP/IP versus HTTP - a comparison

Using this simple stress test, we compared throughput rate of TCP/IP to that of HTTP between queue managers.

We set up ClientQm on one machine, and WASServerQm and ServerQm on a second machine, connected over a LAN. Then we ran the above stress test on each, with these results:

– It took 250 ms to send 1500 bytes to ServerQm using TCP/IP.

  Throughput rate: 6000 bytes/sec

– It took 1844 ms to send 10 messages of a total of 1500 bytes to WASServerQm using HTTP, with WASServerQm running in WebSphere.

  Throughput rate: 813 bytes/sec

14.8.19 Coding administration messages

Because in our example we have used the ES02 package to configure our queue managers, it has not been necessary to develop any code to perform this task. Without the ES02 package, it would have been necessary to develop programs to perform the administrative task of creating and configuring queue managers.

This configuration process is performed by using MQSeries Everyplace administrative type messages. When a queue manager is defined, two queues to handle administration are defined. They are called:

- AdminQ
- AdminReplyQ

Administration of queue managers is performed by using administration messages. The messages are written to the AdminQ, where the queue manager actions the message and replies on the AdminReplyQ.

Building these administration messages requires writing a program. The ES02 package does this for you, by turning the actions (which you generate by clicking various objects) into administration messages.

In the ClientMgr class there are two methods, which provide sample code to show how to build administration type messages. The addConnection defines a connection, while the addQueue method defines a queue.

14.9 Extending the YourCo application

This section describes how MQSeries Everyplace can be used to extend existing applications.

14.9.1 Overview

The YourCo application is a sample application supplied with WebSphere Application Server. It demonstrates various features of WebSphere, EJBs, servlets, etc., with browser-based access.

As part of the chat room application, some extra code was developed to demonstrate how MQSeries Everyplace can be used to access an existing application running in WebSphere. This example demonstrates how to set up a simple adapter to implement authenticated access to a queue.

The YourCo sample application consists in part of a database containing in one database table a list of employees, and in another table a list of different types of leave that the employees are owed.

The aim of the example is to show how a manager on a remote device could access information securely from an applicatio which, to date, only had a browser interface.

This example is implemented on typical Windows type desktops, with the client application running on a Windows desktop. However, the client application could be implemented on a palm type device, which perhaps does not support a normal browser interface. Using MQSeries Everyplace, and the Wireless Gateway support of the WebSphere Everyplace Suite, a person could gain remote secure access to an application that previously had required a browser for access.

14.9.2 YourCo extensions

The example set up here involves sending a predefined message to the queue YourCoQuery in the WASServerQm queue manager running in WebSphere. When the message arrives, a bean is invoked to determine the total amount of different types of leave that all staff of YourCo have. A message with these totals is returned to the chat room window.

The following describes the extra Java packages set up to handle the interaction with theYourCo application.

Part of the supplied YourCo application is a package called WebSphereamples.YourCo.Timeout. This package is used to display leave information about individual staff members of YourCo on a browser. The code was copied into VisualAge for Java, then the following new classes developed:

– totalLeaveBean - used to store leave values
– totalLeaveServlet - contains the method to retrieve leave information

When the init method of the ITSOMQeWas servlet is run, a message listener is defined for the YourCoQuery queue. When a message arrives on that queue, the messageArrived method of the WasQMgr object is called. Example 14-23 shows the code executed from this method:

Example 14-23 Handling message on the YourCoQuery queue

```
if (eventQueueName.indexOf("YourCoQuery") >= 0) {
   System.out.println("call ejb to get info");
    msgObj = wasQMgr.getMessage(null, "YourCoQuery", null, null, 0);
   System.out.println("From: " + msgObj.getOriginQMgr() +
               " : " + eventQueueName +
               " msg: " + msgObj.getAscii("Message"));
   findTotalLeave();
   String yourCoMsg = "YourCo leave Totals: Vactional: " + sumVactional +
                  " Personal: " + sumPersonal +
                  " Sick: " + sumSick;
   replyMsg.putAscii("Message", yourCoMsg);
   wasQMgr.putMessage("ClientQm", "ChatClientQ", replyMsg, null, 0);
```

The result of the code is that the findTotalLeave method of the WasQMgr object is called. This main part of this method is shown in Example 14-24:

Example 14-24 Calling bean to access YourCo information

```
TotalLeaveBean totalLeaveInfo = new TotalLeaveBean();

        totalLeaveInfo = totalLeaveServlet.calcTotalLeave(null);

        sumVactional = totalLeaveInfo.getTotalVactional();
        sumPersonal = totalLeaveInfo.getTotalPersonal();
        sumSick = totalLeaveInfo.getTotalSick();
```

The above code shows that a Java bean object of type TotalLeaveBean has been created. This bean was written for this example. The totalLeaveServlet class was written initially to run the TotalLeaveBean using a browser to test its functionality, prior to using it in this example.

The above code shows that the calcTotalLeave method of the totalLeaveServlet class is called, which will return a bean of type totalLeaveInfo.

The calcTotalLeave method of the totalLeaveServlet class uses another bean developed for this redbook, called InviteesDBBean. This bean returns a list of all employees from the YourCo database. Then, for each employee in the list, the number of different types of leave they have is obtained, using an existing EJB, and a running total for each type kept. The code is shown in Example 14-25:

Example 14-25 Calculating the total of the different leave types

```
try {
                while (true) {
                    ii = ii + 1;

                    employeeId =
Integer.valueOf(InviteesDBBean.getEMPNO(ii)).intValue();

                    System.out.println(
                        "Employee id: " + employeeId + " String: " +
InviteesDBBean.getEMPNO(ii));

                    try {
                        totalVactional = totalVactional +
access.getBalance(employeeId, 1);

                        totalPersonal = totalPersonal +
access.getBalance(employeeId, 2);
```

```
                        totalSick = totalSick + access.getBalance(employeeId,
3);

                        System.out.println("tv:" +
access.getBalance(employeeId, 1));
                    } catch (Exception e) {
                        System.out.println("TL - Exception: " +
e.getMessage());
                        e.printStackTrace();
                    }

                } // End while
            } // End try
```

What we have demonstrated here is that a remote application using the messaging technology of MQSeries Everyplace can easily be used to access an existing Web-based application.

14.9.3 Customized authenticator adapter

MQSeries Everyplace comes with some sample authentication type adapters. In this example, however, we show how a simple authentication adapter was developed .

A detailed description of the MQSeries Everyplace authentication adapter can be found in Chapter 2 of the *MQSeries Everyplace Programming Reference* manual, SC34-5846.

The following diagram outlines the process that occurs when an authentication adapter is used.

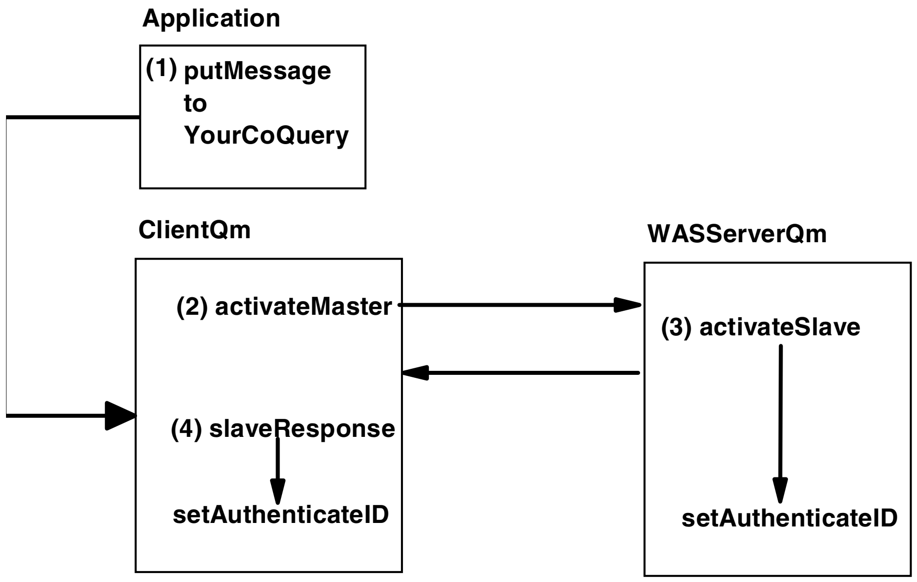


Figure 14-39 Authentication adapter flow

The process, with reference to the above diagram, is as follows:

1. The application issues a putMessage to the YourCoQuery queue.
2. The ClientQm queue manager detects that an authentication adapter is specified in the queue definition, and invokes the activateMaster in the class specified; in this case, the customized adapter is in the package itso.mqe.security.QueueAuthenticator.
    a. The activateMaster method then displays a small window to ask the end user for a password.
    b. The password entered by the user is returned to the queue manager to pass to the corresponding activateSlave method on the server queue manager; the code to do this is shown below:

Example 14-26 Returning password for validation

```
/* Password entered by user is passed back to queue manager, which
       will send it to the corresponding activateSlave method on the server
       queue manager */

   System.out.println("pwd: " + password);
   String replyTxt = "From Master: " + password;
   byte [] replyMsg = replyTxt.getBytes();
   return replyMsg;
```

3. On the WASServerQm queue manager, the activateSlave method of the QueueAuthenticator class is invoked.
   a. The data passed to this method contains the password entered by the end user; the code in the method validates the password and sends back a positive response. The code that does this is shown below:

Example 14-27 Validating the password

```
if (recvMsg.indexOf("shazam") > 0){
 try
  {
    setAuthenticatedID( authID );

    replyMsg = "From Slave: Auth ok: ".getBytes();

    return replyMsg;

     }                                    /*
```

   If the password is incorrect, an exception is thrown, which will result in the activateMaster method being reinvoked on the ClientQm queue manager, which will redisplay the window asking for the password.
   b. The setAuthenticatedID method call tells the queue manager that authentication has been successfully established for the queue.
   c. On the ClientQm queue manager, the slaveResponse method in the QueueAuthenticator class is called; this method simply calls the setAuthenticatedID method to notify the ClientQm that access to the queue has been authenticated.

14.9.4 Queue definitions

To run the example, set up the following queue definitions. It is assumed that you have set up the Chat Room Client example described in “ChatRoom: an MQSeries Everyplace application” on page 595.

On ClientQM: remote queue-YourCoQuery

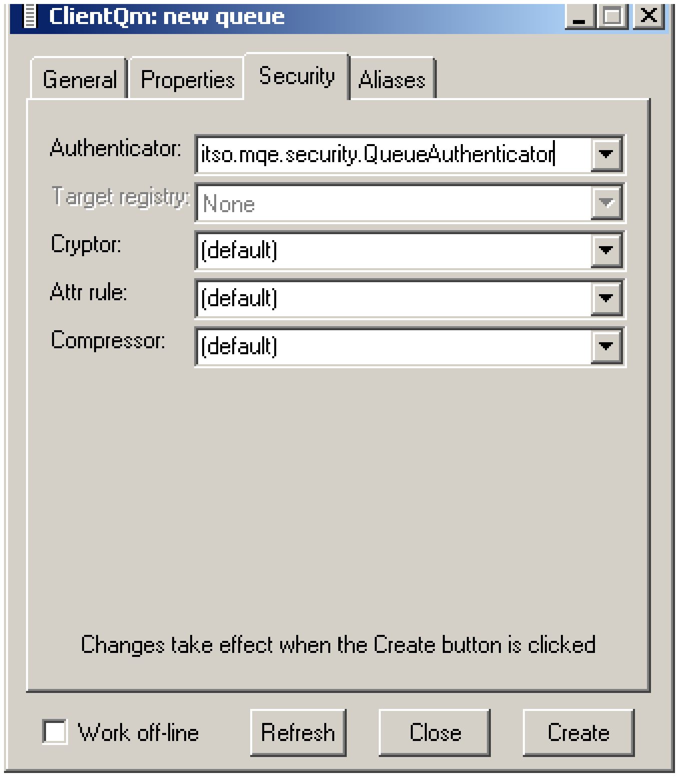
This is a similar process to the one you have been using to define queues in the Chat Room application.

1. Stop the chat room application if running, and use ES02 to load up the ClientQm queue manager.

2. From the expanded tree view,right-click the object labelled **Local Queues**. A window will appear, in which to enter the details of the queue you wish to define.
3. Under the tab labelled General, type `YourCoQuery` into the field labelled Name.
4. Change the queue type to **Remote queue**, by selecting that value from the drop-down menu in the field labelled Type.
5. In the box labelled Queue qMgr, select from the drop-down menu the name of the remote queue manager, in this case, **WASServerQm**.
6. Check that the mode is **Synchronous,** by selecting that value from the drop-down menu in the field labelled Mode.
7. Then click the **Security** tab. In the field labelled Authenticator, type in this value:

```
itso.mqe.security.QueueAuthenticator
```

The window will look like that shown in Figure 14-40:


ClientQm: new queue

Authenticator: itso.mqe.security.QueueAuthenticator
Target registry: None
Cryptor: (default)
Attr rule: (default)
Compressor: (default)

Changes take effect when the Create button is clicked

Figure 14-40 Specifying a customized authentication adapter

8. Then click the **Create** button. No confirmation window is displayed.

On WASServerQM: Local queue-YourCoQuery

1. Stop the application server in WebSphere Application Server if the queue manager is active in the servlet. Then use ES02 to load up the WASServerQm queue manager.
2. From the expanded tree view,right-click the object labelled **Local Queues**. A window will appear, in which to enter the details of the queue you wish to define.
3. In the tab labelled General, type `YourCoQuery` into the field labelled Name.

   The queue type should be local, the mode synchronous.
4. Then click the **Security** tab. In the field labelled Authenticator, type in this value:

```
itso.mqe.security.QueueAuthenticator
```

5. Then click the **Create** button. No confirmation window is displayed.

14.9.5 Property File

Check that in the MQe.property file, the flag for the YourCo property is set to `Yes`. The line in the property file should look like this:

```
YourCo=Yes
```

14.9.6 Additional beans

The YourCo application comes as a supplied example with WebSphere Application Server. Additional beans and servlets were developed to demonstrate the new functionality of WebSphere Everyplace Server. These additional beans are supplied with the zip file for this redbook. They need to be copied to the directory containing the rest of the YourCo example.

This directory also needs to be added to the classpath in the Web Application definition in WebSphere Application Server:

```
   C:\WebSphere\AppServer\hosts\default_host\WSsamples_app\servlets
```

These additional classes:

- WebSphereSamples.YourCo.Timeout.TotalLeaveBean
- WebSphereSamples.YourCo.Timeout.TotalLeaveServlet

need to be copied to this directory:

```
C:\WebSphere\AppServer\hosts\default_host\WSsamples_app\servlets\Web
SphereSamples\YourCo\Timeout
```

while this class:

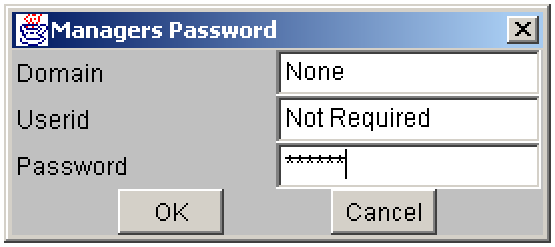
WebSphereSamples.YourCo.Meeting.InviteesDBBean

need to be copied to this directory:

```
C:\WebSphere\AppServer\hosts\default_host\WSsamples_app\servlets\Web
SphereSamples\YourCo\Meeting
```

14.9.7 Running the YourCo example

To run this example, start the client side of the chat room application. Then click the button labelled `YourCo Secure Query`. You will then be prompted to enter a password before access to the YourCoQuery queue is allowed, as shown below:



Domain: None
Userid: Not Required
Password: ******

Figure 14-41 YourCoQuery queue password prompt

Enter the password `shazam` into the password field and click the **OK** button.

After a few seconds, the reply message advising the total leave values by type will appear in the chat room window. The message will be as shown in Example 14-28.

Example 14-28 YourCo query reply message

```
From: WASServerQm : ChatClientQ msg: YourCo leave Totals: Vactional: 11
Personal: 14 Sick: 20
```

14.10 Integration with WebSphere Everyplace Suite

So far in this chapter, the examples of using MQSeries Everyplace with applications have been done on standard desktops using Windows 2000, over a standard LAN. However, one of the reasons that MQSeries Everyplace was developed was to provide assured messaging capability over non traditional networks, such as those now available for wireless connection.

The connectivity support provided by the Wireless Gateway component of WebSphere Everyplace Server means that applications that use MQSeries Everyplace can use this connection support to allow them to run on wireless devices. While this chapter has only described using MQSeries Everyplace on Windows systems, the product does provide support for other devices, such as palm type devices.

The Wireless Gateway consists of a server component which would typically be run within an organization's data center, and a client component installed on the wireless device. This client component handles the process of communicating with the server side of the Wireless Gateway.

One of the advantages of using the Wireless Gateway is that it can be configured to provide authentication and encryption services. Enabling authentication means that when an end user establishes a wireless connection, they will be prompted for their authentication details by the Wireless client. Enabling encryption means that all data transferred between the client and the server is encrypted, preventing unauthorized people from viewing the data.

These authentication and encryption services of the Wireless Gateway can be of use in addition to any authentication and encryption that applications or other products may use communicating through the Wireless Gateway.

Applications using MQSeries Everyplace, when run on a device using the Wireless Gateway for handling communication, require no modifications. Applications using MQSeries Everyplace do not handle any of the communication process; rather, this is done by MQSeries Everyplace. Additionally, MQSeries Everyplace does not require any special configuration to use the Wireless Client support.

An advantage that MQSeries Everyplace provides is sending messages using the HTTP protocol as well as TCP/IP. This means that a site that has the typical firewall setup to allow in HTTP traffic through to back end Web servers does not need to change this setup to allow applications on wireless devices access to the system. Since the packets of data from the MQSeries Everyplace applications will be standard HTTP packets, coming in on the standard HTTP port 80, the firewall will not require modifications.

By using client type queue managers on the wireless device, MQSeries Everyplace will only be establishing connections from the outside world into the organization's site. No connections are established from within the organization's site to devices outside.

Web Traffic Express

As the standard HTTP protocol can be used by MQSeries Everyplace, these requests which will invoke a servlet in WebSphere Application Server can be initially routed to the WebTraffic Express component of WebSphere Everyplace Server. WebTraffic Express can then use the WebSEAL-Lite plug-in, to verify with Policy Director if the request should be allowed to pass through to the back-end server.

A sample authentication adapter is shipped with MQSeries Everyplace; it can be used to add an authentication header to the HTTP request, containing the user ID and password of the end user. The user ID and password are encoded using a base 64 algorithm, just as is done in browsers, meaning that it is a trivial task to decrypt it. However, enabling the encryption support of the Wireless Gateway means that the HTTP requests sent by MQSeries Everyplace will be encrypted using a much stronger algorithm, which protects the user ID and password in the HTTP request.

Figure 14-42 shows how the Chat Room application would be implemented across multiple devices with the Wireless Gateway handling the communication between the client and other queue managers.

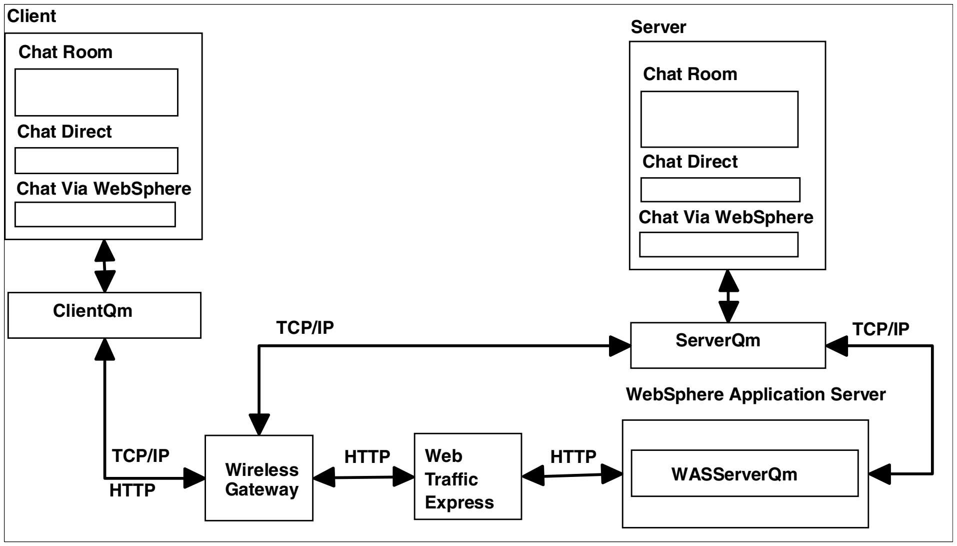


Figure 14-42 Integration with Wireless Gateway

Location-based services

Chapter 11, “Location-Based Services (LBS)” on page 435 describes another feature of WebSphere Everyplace Suite, called Location-Based Services. Location-Based Services allows information about the location of the end user to be added to the HTTP request when received on the server side. Since MQSeries Everyplace could be configured to send messages as HTTP requests, these requests could be routed through the system running the Location Based Services component.

Location information would be added to the HTTP request, which would then flow onto the servlet in WebSphere Application Server. The servlet, as well as passing the message to MQSeries Everyplace, could extract the location information and use this as required.

By using client type queue managers on the wireless device, MQSeries Everyplace will only be establishing connections from the outside world into the organization’s site. No connections are established from within the organization’s site to devices outside.

14.10.1 Using the Wireless Client and Gateway

This section demonstrates how to use the Wireless Gateway in conjunction with MQSeries Everyplace.

Figure 14-43 shows how we initially configured the chat room application to run across two Windows 2000 machines.

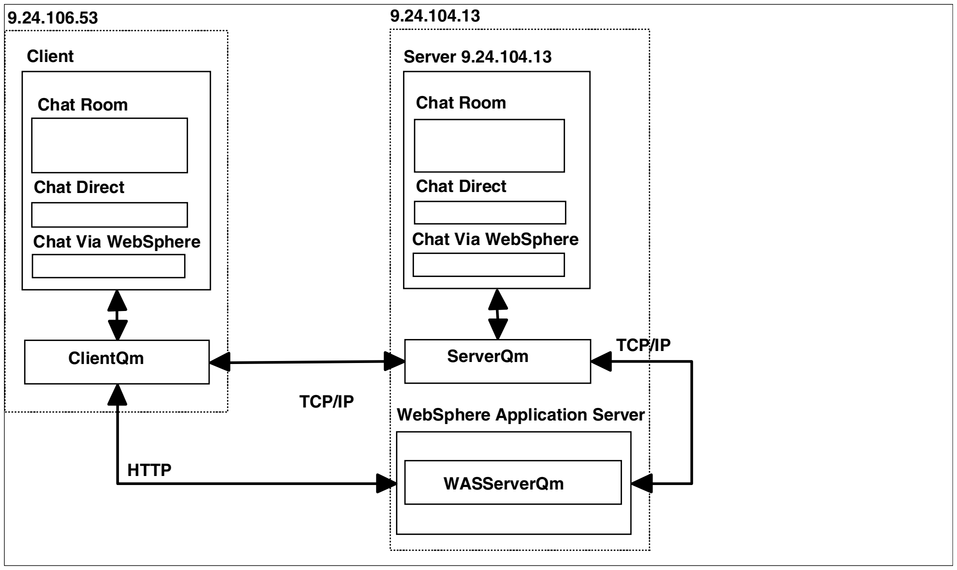


Figure 14-43 Chat Room application over standard LAN

The above diagram shows the client side running on a PC at address 9.24.106.53, while the server queue manager and WebSphere Application Server both run on the PC at address 9.24.104.13.

We now want to use the Wireless Gateway to handle communication between these two devices.

The Wireless Gateway was set up on an AIX system, and the Wireless Client was installed on the client side PC.

The Wireless Gatekeeper is the tool used to administer the Wireless Gateway. In the lab, we had the Wireless Gateway running on an AIX system. A sample screen shot is shown below.

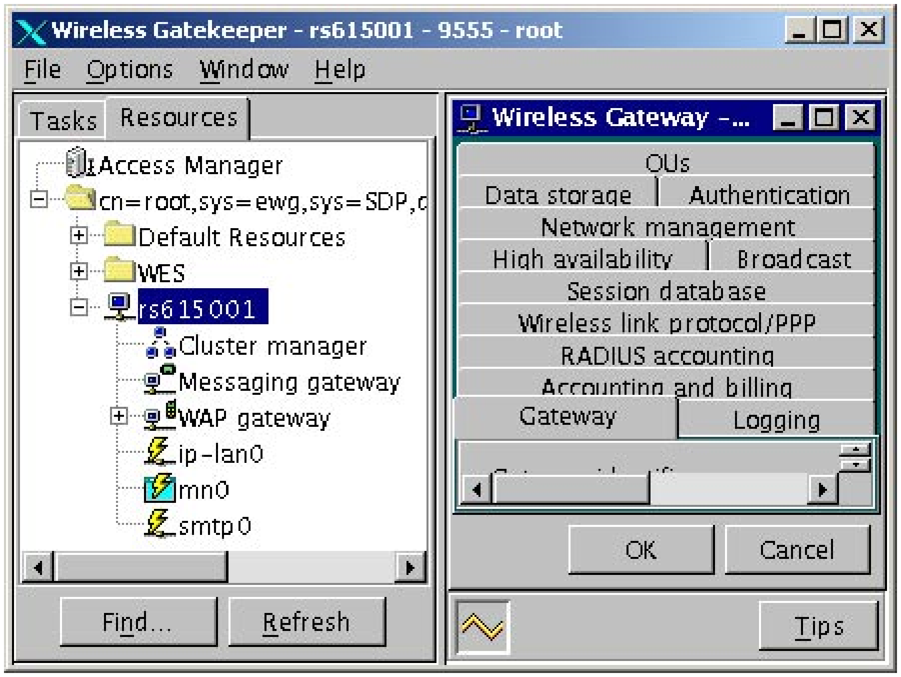


Figure 14-44 Wireless Gatekeeper GUI

On an AIX system, the gatekeeper is started by typing in `wgcfg`.

On the left hand side of the Gatekeeper window is a tree structure showing the various objects being managed by the Gateway. In the tree, the object labelled `RS615001` represents the AIX system running the Wireless Gateway. Under this object are two objects relating to the Wireless Gateway.

Mobile Network Connection

The first is an icon of a connection with a lightning bolt. This icon represents a Mobile Network Connection (MNC). This represents the interface to a network provider for the Wireless Gateway. Right-click this icon, and select **Properties**. The right hand side of the window displays the associated properties. For this example, we do not want to use the authorization facility of the Wireless gateway. To set this level of authentication, click the tab labelled **Security**. Next, click the radio button corresponding to **No validation**, then click the **Apply** button to effect the change (see Figure 14-45).

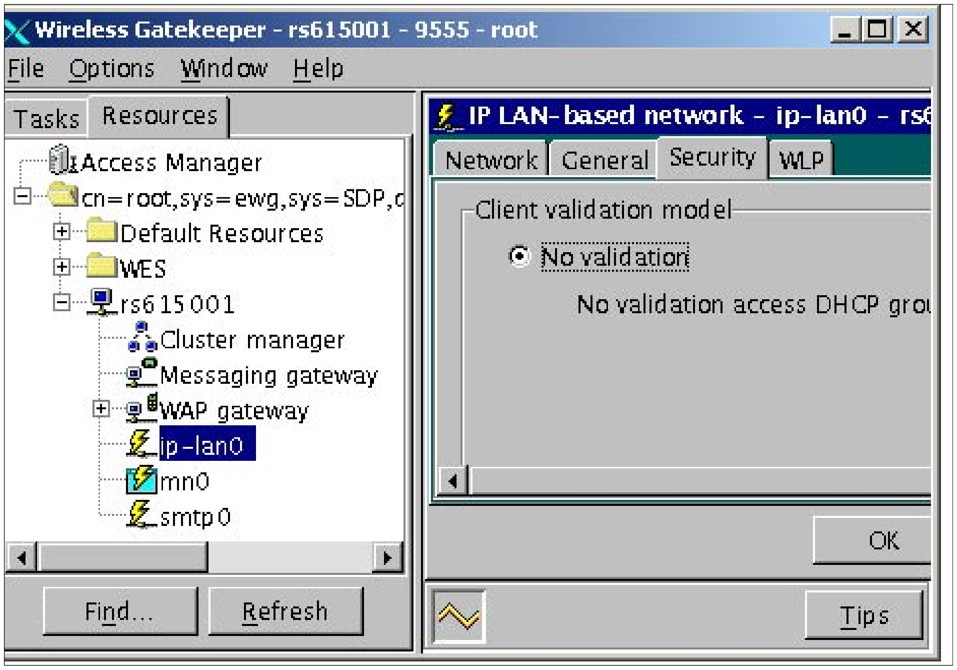


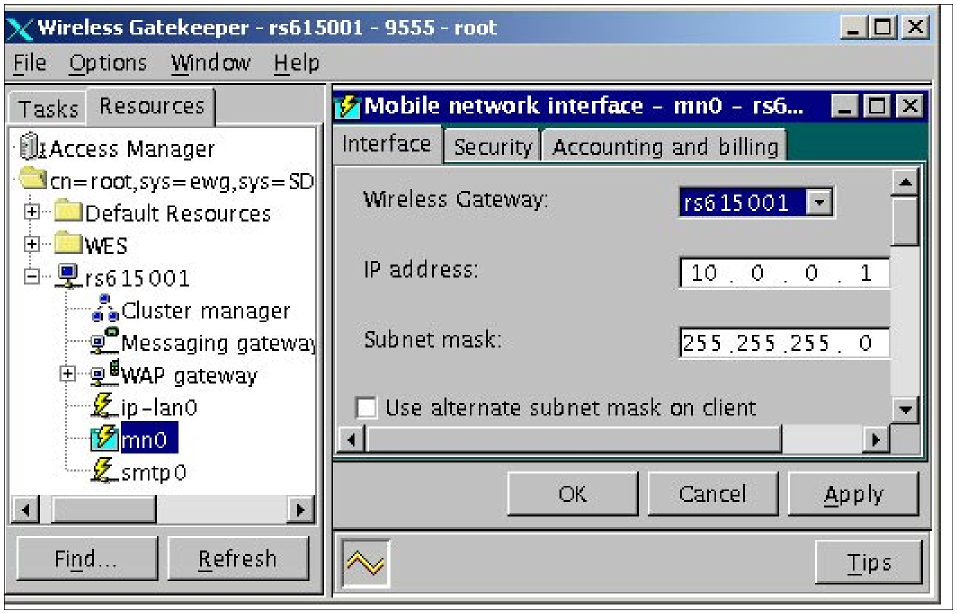
Figure 14-45 Selecting no authentication

Mobile Network Interface

The next object of interest is the one that appears as a light blue icon with a lightning bolt through it. This icon represents the Mobile Network Interface, or MNI. This interface defines an IP subnet, through which the Wireless Gateway routes traffic for Wireless Clients. When a device connects using the Wireless client, it will be allocated an address from this subnet. Right-click this icon, and select **Properties**. The right hand side of the Gatekeeper window will then display the properties of the MNI. Click the tab labelled **Interface**. The field labelled *IP address* is where you specify the IP subnet that will be used to support the clients.

Configuring this IP subnet correctly is a key issue when setting up a wireless gateway.

For our example, we set up a virtual IP subnet at address 10.0.0.1. Thus the value entered in the IP Address field in our case was `10.0.0.1` (see Figure 14-46).



Wireless Gateway: rs615001

IP address: 10 . 0 . 0 . 1

Subnet mask: 255 .255 .255 . 0

Use alternate subnet mask on client

Figure 14-46 Configuring the Wireless Gateway client

Our configuration with the Wireless Gateway incorporated now looks as shown in Figure 14-47.

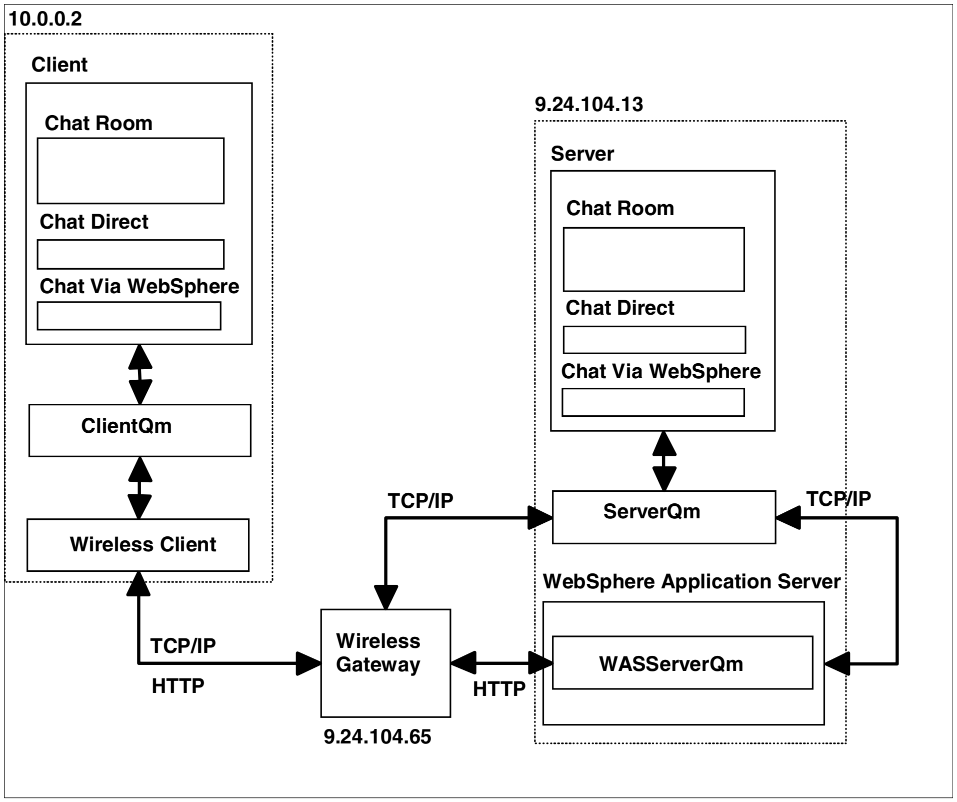


Figure 14-47 Incorporating the Wireless Gateway

Note that the client now has an IP address of 10.0.0.2. This is the IP address it has been allocated by the Wireless Gateway when it connected using the Wireless Client. If it reconnects at a later time, the address may change, for example to 10.0.0.5.

In this case, we are running the wireless protocol over the LAN to demonstrate the use of the Wireless Gateway and Client.

IP packets can now flow from the client through the gateway to the 9.24.104.13 machine. However, an entry needs to be added to the route table on the 9.24.104.13 machine so that it knows where to send reply packets destined for the client address at 10.0.0.2. On the Windows 2000 machine at 9.24.104.13, open a DOS window and enter this command:

```
route ADD 10.0.0.0 MASK 255.255.255.0 9.24.104.65
```

This adds a temporary TCP/IP routing entry, that tells that system to route packets destined for 10.0.0.* to 9.24.104.65, which is the Wireless Gateway.

Note that this setup is for example purposes only. A production implementation would require a proper IP subnet and routing tables to be configured.

14.10.2 Trying out the Wireless Gateway

1. First, set up the chat room application on two machines, as depicted in Figure 14-43 on page 673, and ensure the application is working normally.
2. Then stop the chat room application on the client side. This must be done before starting the Wireless Client.
3. Install the Wireless Client software on the Windows 2000 machine on the client side. This is a straightforward process.
4. Then, using the **Start** button, find the IBM Wireless Client, and select **Connections**. In the window displayed, create a connection definition if one has not already been defined. This creation process is straightforward. The most important item to know is the IP address of the Wireless Gateway which needs to be entered.

   In the lab, we created a connection called France, as shown below:

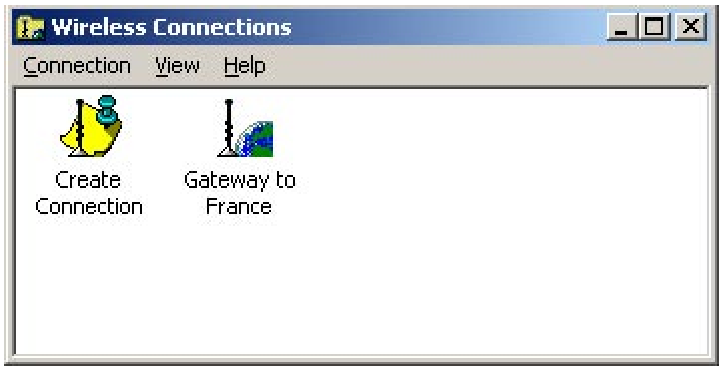


Figure 14-48 Wireless Connections

5. To establish a wireless connection, just double-click the icon labelled Gateway to France. A window similar to the one shown in Figure 14-49 appears.

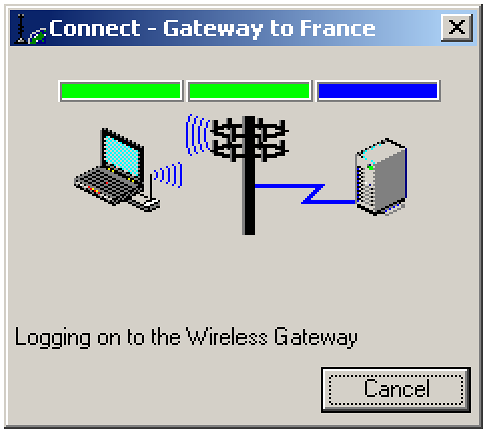

Connect - Gateway to France
Logging on to the Wireless Gateway
Cancel

Figure 14-49 Connecting to the Wireless Gateway

When the three boxes all turn green, the connection is established, and the window disappears. On the right hand side of the Windows 2000 task bar, a small icon of a transmission tower appears.

6. To test your wireless connection, open a DOS window. Type in `IPCONFIG`, and you will get a display similar to this (see Example 14-29):

Example 14-29 IP status

```
C:\>ipconfig

Windows 2000 IP Configuration

Ethernet adapter {21959871-44F7-46A7-BE57-6501A133852C}:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 10.0.0.3
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 10.0.0.1

Token Ring adapter Local Area Connection:

        Connection-specific DNS Suffix  . : itso.ral.ibm.com
        IP Address. . . . . . . . . . . . : 9.24.106.53
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . :
```

This output shows that you still have the LAN connection, but now have also been allocated a new IP address of 10.0.0.3, which is your Wireless connection.

7. Then PING the address of the machine running your server side queue managers. Watch the little transmission tower icon; you will see a lightning bolt flash to indicate that IP traffic is being sent.
8. Restart the client side of the chat room application, and send some messages; it should function as before.

This example demonstrates that MQSeries Everyplace applications can be deployed to run on wireless devices, with the Wireless Gateway providing the communication support.

14.10.3 Tracing

On the client

The Wireless Client provides a tracing capability to assist with resolving communications problems. Right-click the small transmission tower icon, and select **Trace**. A window allowing you to set trace options appears. You can set the trace to various levels as required.

Trace information is written to a file called arttrace.txt, which is located in the directory where the Wireless client was installed:

C:\Program Files\IBM\Wireless Client.

This trace file contains formatted trace output, showing IP traffic that has occurred.

On the server

Various levels of logging and tracing can be enabled in the Wireless Gateway. This is done by using the Wireless Gatekeeper. Right-click the icon representing the AIX system you wish to set logging for, select **Properties**, then, in the panel on the right, click the tab labelled **Logging**.

This tab shows the logging and trace file names, and the level of logging and tracing that is active. These values can be adjusted as required.

14.11 OS/390

The OS/390 platform also provides excellent Java support, so we decided to try out MQSeries Everyplace on OS/390. We had access to an OS/390 system running Z/OS V1. We were able to successfully run the chat room application using MQSeries Everyplace on the OS/390 system. This section describes how this was done.

Note that, at time of writing this redbook, IBM was still determining licensing issues with MQSeries Everyplace on OS/390; its use on OS/390 is restricted at this time.

We only had time to set up the environment on OS/390 to allow messages to be sent from the client chat room to the server chat room. However, it would only require the appropriate definitions to allow the server side to function fully.

Note: No Java programs required any modification or recompiling.

14.11.1 Requirements

OS/390 Z/OS contains an Open/Edition environment which provides a UNIX environment. A Windows server is required to allow the Open/Edition (Unix) Services to display the GUI window of the chat room application when it is run.

We installed an X-Windows server onto a Windows 2000 desktop. Then we opened a Telnet session to the OS/390 system, and issued this command to set the address of the machine on which to display the GUI:

```
export DISPLAY=9.24.106.53:0.0
```

The MQSeries Everyplace product is shipped as a zip file. There is no supplied facility in Open/Edition to unzip a zip file. However, the Infozip product has been ported to run on OS/390 Open/Edition. It can be downloaded from the following site:

http://www-1.ibm.com/servers/eserver/zseries/zos/unix/bpxa1ty1.html

We downloaded this file, ftp'd it to Open/Edition in OS/390, then installed it, which simply involved untarring it.

This then provided us with a way to unzip any zip files we created on the Windows platform.

We then used WinZip to zip up the directory containing the MQSeries Everyplace product, ftp'd this file to the OpenEdition environment and unzipped it.

Then we zipped up the directory containing the chat room application packages, ftp'd this to Open/Edition and unzipped it.

Note: Like all Unix environments, Open/Edition is case sensitive, thus it is very important to ensure that cases of the directory names match the package names coded in the Java programs.

After this transfer process was complete, the MQSeries Everyplace product was located at /u/.edward/mqe3/MQe, while the Chat Room application was located at /u/edward/itso.

14.11.2 Classpath

To be able to define the queue manager and run the application, based on where we had unzipped files, as mentioned above, we set the classpath using this command:

```
export CLASSPATH=/u/edward/mqe3/MQe/Java:/u/edward:.
```

Also, the PATH environment variable should reference the location of the Java executable, for example on our system we set the PATH as follows:

```
export PATH=/usr/lpp/java213/J1.3/bin
```

14.11.3 Configuring ServerQm

The MQe_Explorer tool from the ES02 support pac cannot be run from OS/390. However, the MQSeries Everyplace product comes with many example Java programs, which can be used from a command line to perform queue manager administration tasks. We used these tools to set up the ServerQm on OS/390.

Note that we only set up the server side to allow the client chat window to send messages to the server side.

Creating the .ini file

First, we need to create the .ini file defining the initialization parameters for the ServerQm queue manager. Due to the way the supplied samples have been written, the .ini file is expected to be in ASCII. While you can store the .ini file in ASCII in Open/Edition, you cannot edit it there.

Use Notepad to code up the .ini file, then do a binary transfer of this file to the OpenEdition environment. We placed the .ini file in a directory called /u/edward/os390.

The .ini file is shown below:

Example 14-30 ServerQm ini file for OS/390

```
[Registry]
(ascii)LocalRegType=FileRegistry

(ascii)DirName=/u/edward/os390/ServerQm/Registry/

(ascii)Adapter=RegistryAdapter


[ChannelManager]
(int)MaxChannels=0


[QueueManager]
(ascii)Name=ServerQm


[Listener]
(int)TimeInterval=300

(ascii)Listen=FastNetwork::8082

(ascii)Network=FastNetwork:


[Alias]
(ascii)QueueManager=com.ibm.mqe.MQeQueueManager

(ascii)DefaultTransporter=com.ibm.mqe.MQeTransporter

(ascii)RegistryAdapter=com.ibm.mqe.adapters.MQeDiskFieldsAdapter

(ascii)MsgLog=com.ibm.mqe.adapters.MQeDiskFieldsAdapter

(ascii)PrivateRegistry=com.ibm.mqe.registry.MQePrivateSession

(ascii)FastNetwork=com.ibm.mqe.adapters.MQeTcpipHistoryAdapter

(ascii)FileRegistry=com.ibm.mqe.registry.MQeFileSession

(ascii)Server=examples.queuemanager.MQeServer

(ascii)ChannelAttrRules=examples.rules.AttributeRule

(ascii)Admin=examples.administration.console.Admin

(ascii)AttributeKey_2=com.ibm.mqe.attributes.MQeSharedKey
```

```
(ascii)AttributeKey_1=com.ibm.mqe.MQeKey

(ascii)DefaultChannel=com.ibm.mqe.MQeChannel

(ascii)Network=com.ibm.mqe.adapters.MQeTcpipHttpAdapter
```

Creating the ServerQm queue manager

We then issued this command to create the ServerQM queue manager:

```
java examples.install.SimpleCreateQM /u/edward/os390/ServerQm.ini
```

Basic test of ServerQm

We then ran the supplied examples, just to test that the queue manager could be run successfully, by issuing this command:

```
java examples.application.Example1 ServerQm
/u/edward/os390/ServerQm.ini
```

Adding the Chat Room application

We then added the Chat Room application to the .ini file, and ftp'd that to Open/Edition. The lines added to the bottom of the.ini file are shown below:

Example 14-31   Adding the Chat Room application to ini file

```
[AppRunList]
(ascii)App1=itso.mqe.chatserver.RoomMgr

[App1]
(ascii)ClientQueue=ChatClientQ
(ascii)ChatRoomQ=ChatRoomQ
```

Defining ChatRoomQ

We issued this command to define the local queue, ChatRoomQ, to the ServerQm queue manager:

```
java examples.administration.commandline.LocalQueueCreator ChatRoomQ
null null null nolimit nolimit ServerQm /u/edward/os390/ServerQm.ini
com.ibm.mqe.adapters.MQeDiskFieldsAdapter:/u/edward/os390/ServerQm
```

14.11.4 Modifying ClientQm

We then used the ES02 MQe_explorer tool to modify the connection definition to ServerQm in ClientQm. We changed the IP Address to the IP address of the OS/390 system. We then shut down MQe_explorer.

14.11.5 Starting Chat Room on OS/390

We then started the ServerQm queue manager on OS/390 by issuing this command:

```
java examples.queuemanager.MQeServer /u/edward/os390/ServerQm.ini
```

Before doing this, be sure that you have set the DISPLAY environment variable in your Open/Edition Telnet session, and that you have the X-Windows server running on the system where you want the GUI window to appear.

Once the above command was issued, the server side GUI window of the chat room application appeared on our Windows desktop.

Starting the client side of Chat Room

We then started the client side of the Chat Room application, as explained in “Starting the chat room application” on page 650. The client side GUI window appeared. We then typed a message on the client side, and it duly appeared in the server chat room window.

Part 5

Appendixes

A

INS sample source code

This appendix contains the source code for the YourCo INS Extensions as discussed in Chapter 10, “Intelligent Notification Services (INS)” on page 359.

Simple notification:

- Servlets and Java classes:
  - WebSphereSamples.YourCo.Meeting.Invitees.java
  - WebSphereSamples.YourCo.Meeting.InviteesDBBeans.java
  - itso.wes.ins.samples.Notify.java
  - itso.wes.ins.samples.UserNotificationBean.java
- Java Server Pages and Web pages:
  - /YourCo/invitees.jsp
- Changes made to the original YourCo example:
  - /YourCo/Meeting/ScheduleResults.jsp

Subscriptions

- Servlets and Java classes:
  - itso.wes.ins.triggersample.MeetingSubscriptionStartServlet.java
  - itso.wes.ins.triggersample.MeetingSubscriptionServlet.java
  - itso.wes.ins.triggersample.MeetingSubscriptionData.java

  - itso.wes.ins.triggersample.MeetingSubscriptions.java
  - itso.wes.ins.triggersample.MeetingHandler.java
  - itso.wes.ins.triggersample.MeetingContentAdapter.java
- Java Server Pages and Web pages:
  - /YourCo/Triggers/MeetingSubscriptionForm.jsp
- Changes made to the original YourCo example:
  - /YourCo/Employee/CenterGeneric.jsp

Simple notification

Example: A-1 WebSphereSamples.YourCo.Meeting.Invitees.java

```
// 5648-C84, 5648-C83, (C) Copyright IBM Corporation, 1997, 2000
// All rights reserved. Licensed Materials Property of IBM
// Note to US Government users: Documentation related to restricted rights
// Use, duplication or disclosure is subject to restrictions set forth in GSA
ADP Schedule with IBM Corp.
// This page may contain other proprietary notices and copyright information,
the terms of which must be observed and followed.
//
// This program may be used, executed, copied, modified and distributed
// without royalty for the purpose of developing, using,
// marketing, or distributing.

/**
* This file was generated by IBM WebSphere Studio Version 3.5
*
D:\WebSphere\Studio35\BIN\GenerationStyleSheets\V3.5\JSP1.0\ServletModel\Databa
seServlet.xsl stylesheet was used to generate this file.
*
*
*
*/
package WebSphereSamples.YourCo.Meeting;
// Imports
import com.ibm.servlet.*;
import com.ibm.webtools.runtime.*;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;


// Imports for beans used by this servlet

import WebSphereSamples.YourCo.Meeting.InviteesDBBean;


public class Invitees extends com.ibm.webtools.runtime.StudioPervasiveServlet
implements Serializable

{
```

```
/****************************************************************************
  * Process incoming requests for information
  *
  * @param request Object that encapsulates the request to the servlet
  * @param response Object that encapsulates the response from the servlet
  */
  public void performTask(HttpServletRequest request, HttpServletResponse
response)
  {

    // Uncomment the following line to aid in debugging DB servlets.
    // This can help isolate problems when the servlet appears to be unable to
make
    // a connection to the database. You must recompile, republish, and then
shutdown and
    // restart the server for this change to take effect.
    // java.sql.DriverManager.setLogStream(System.out);

    try
    {

      // instantiate the beans and store them so they can be accessed by the
called page
      WebSphereSamples.YourCo.Meeting.InviteesDBBean InviteesDBBean = new
WebSphereSamples.YourCo.Meeting.InviteesDBBean();
      setRequestAttribute("InviteesDBBean", InviteesDBBean, request);


      // Initialize the bean userID property from the parameters
      InviteesDBBean.setUserID(getParameter(request, "userID", true, true,
false, null));

      // Initialize the bean password optional property from the parameters
      InviteesDBBean.setPassword(getParameter(request, "password", true, true,
false, null));

      // Initialize the bean URL property from the parameters
      InviteesDBBean.setURL(getParameter(request, "URL", true, true, false,
null));

      // Initialize the bean driver property from the parameters
      InviteesDBBean.setDriver(getParameter(request, "driver", true, true,
false, null));

      // Initialize the bean dataSourceName optional property
      InviteesDBBean.setDataSourceName(getParameter(request, "dataSourceName",
true, true, false, null));
```

```
      // Call the execute action on the bean.
      InviteesDBBean.execute();

      // Call the output page. If the output page is not passed
      // as part of the URL, the default page is called.
      callPage(getPageNameFromRequest(request), request, response);
    }

    catch (Throwable theException)
    {
      // uncomment the following line when unexpected exceptions are occuring
to aid in debugging the problem
      // theException.printStackTrace();

      handleError(request, response, theException);
    }

  }

}
```

Example: A-2 WebSphereSamples.YourCo.Meeting.InviteesDBBean.java

```
package WebSphereSamples.YourCo.Meeting;

// Imports

import com.ibm.db.*;
import com.ibm.webtools.runtime.*;
import java.io.*;
import java.math.*;


// Imports for App Server V3 connection pooling
import javax.sql.DataSource;
import com.ibm.ejs.dbm.jdbcext.*;
import javax.naming.*;
import com.ibm.ejs.ns.jndi.*;
import java.sql.*;
import java.util.*;


// Imports for App Server V3.5 connection pooling
import com.ibm.websphere.advanced.cm.factory.*;
```

```
/**  */

public class InviteesDBBean extends java.lang.Object
{

  private static final int LASTNAME_COLUMN = 1;
  private static final int FIRSTNME_COLUMN = 2;
  private static final int EMPNO_COLUMN = 3;
  private static final int UID_COLUMN = 4;

  /**
  * Instance variable for driver property
  */
  protected java.lang.String driver= null;


  /**
  * Instance variable for password property
  */
  protected java.lang.String password= null;


  /**
  * Instance variable for SQL statement property
  */
  protected java.lang.String SQLString = "SELECT WSDEMO.EMPLOYEE.LASTNAME AS
LASTNAME, WSDEMO.EMPLOYEE.FIRSTNME AS FIRSTNME, WSDEMO.EMPLOYEE.EMPNO AS
EMPNO1, WSDEMO.EMPLOYEE.USERID AS USERID FROM WSDEMO.EMPLOYEE ORDER BY
LASTNAME, FIRSTNME";


  /**
  * Instance variable for URL property
  */
  protected java.lang.String URL= null;


  /**
  * Instance variable for userID property
  */
  protected java.lang.String userID= null;



  /**
  * Variable for the SelectStatement
  */
  protected SelectStatement sqlStatement;


  /**
  * Variable for the SelectResult - the SQL result set
  */
  protected SelectResult result;

  /**
  * Variable to hold the database connection bean
  */
  protected DatabaseConnection connBean = null;


  /**
  * Variable to hold DataSource
  */
  protected DataSource ds = null;

  /**
  * Variable to hold connection
  */
  protected Connection connection = null;

  /**
  * Variable to hold the data source name from .servlet file
  */
  protected String dataSourceName = "";




/*****************************************************************************
  * Close the result set and release resources
  */
  public void closeResultSet()
  {
    // Release the SQL statement resources
    try
    {
      if (result != null)
      {
        result.close();
        result = null;
      }
    }
    catch (Exception e)
    {
      System.out.println("Error occurred in com.ibm.db.SelectResult.close");
      e.printStackTrace();
    }
    try
    {
      // Close the App Server V3 connection
      if (connection != null)
        connection.close();
    }
    catch (Exception e)
    {
      System.out.println("Error occurred in java.sql.Connection.close");
      e.printStackTrace();
    }
    return ;
  }

/*****************************************************************************
  * execute action method   * @exception com.ibm.db.DataException
  * @exception java.io.IOException
  * @exception java.lang.Exception

  */
  public void execute()
   throws com.ibm.db.DataException, java.io.IOException, java.lang.Exception
  {

    initialize();

    // Initialize the parameters for the query


    // Execute the SQL statement
    sqlStatement.execute();
    result = sqlStatement.getResult();

  }

/*****************************************************************************
  * Release resources we might be holding when garbage collection occurs.
  */
  protected void finalize() throws Throwable
  {
    closeResultSet();
  }

/*****************************************************************************
  * Get method for the App Server V3 dataSourceName property
  */
  public String getDataSourceName()
  {
    return dataSourceName;
  }

/*****************************************************************************
  * Get method for the driver property
  * @return the value of the driver property

  */
  public java.lang.String getDriver(){
    return driver;
  }

/*****************************************************************************
  * Get method for the indexed EMPNO property
  * @param index The 0 based index of the desired value
  * @return the value of the EMPNO property at the specified index
  */
  public java.lang.String getEMPNO(int index)
        throws java.lang.IndexOutOfBoundsException,
java.lang.ArrayIndexOutOfBoundsException
  {

      return (java.lang.String) valueAtColumnRow(EMPNO_COLUMN, index);

  }

/*****************************************************************************
  * Get method for the indexed FIRSTNME property
  * @param index The 0 based index of the desired value
  * @return the value of the FIRSTNME property at the specified index
  */
  public java.lang.String getFIRSTNME(int index)
        throws java.lang.IndexOutOfBoundsException,
java.lang.ArrayIndexOutOfBoundsException
  {

      return (java.lang.String) valueAtColumnRow(FIRSTNME_COLUMN, index);

  }

/*****************************************************************************
  * Get method for the indexed LASTNAME property
  * @param index The 0 based index of the desired value
  * @return the value of the LASTNAME property at the specified index
  */
  public java.lang.String getLASTNAME(int index)
```

```
   throws java.lang.IndexOutOfBoundsException,
java.lang.ArrayIndexOutOfBoundsException
  {

    return (java.lang.String) valueAtColumnRow(LASTNAME_COLUMN, index);

  }

/*****************************************************************************
  * Attempts to get a connection from a App Server V3 connection pool.
  *
  * @param driver Contains the JDBC driver name to use for the connection
  * @param URL Contains the database url for the connection
  * @param userID Contains the userid to use for the database connection
  * @param password Contains the password to use for the connection
  * @return a pooled JDBC connection or null
  */
  protected Connection getPooledConnection(String driver, String URL, String
userID, String password)
  {
    Connection conn = null;

    try
    {
      // create parameter list to access naming system
      Hashtable parms = new Hashtable();
      parms.put(Context.INITIAL_CONTEXT_FACTORY,
CNInitialContextFactory.class.getName());
      // access naming system
      Context context = new InitialContext(parms);
      // get DataSource factory object from naming system
      ds = (DataSource)context.lookup(getDataSourceName());
      conn = ds.getConnection(userID, password);
    }
    catch (Throwable t)
    {
      // DataSource not found. Try to construct a new DataSource.
      try
      {
        t.printStackTrace();
        com.ibm.websphere.advanced.cm.factory.DataSourceFactory factory =
          new com.ibm.websphere.advanced.cm.factory.DataSourceFactory();
        Attributes attrs = new Attributes();
        attrs.name = getDataSourceName();
        if
(attrs.name.startsWith(com.ibm.websphere.advanced.cm.factory.DataSourceFactory.
DEFAULT_DATASOURCE_CONTEXT_NAME + "/"))
        {
```

```
      attrs.name =
attrs.name.substring(com.ibm.websphere.advanced.cm.factory.DataSourceFactory.DE
FAULT_DATASOURCE_CONTEXT_NAME.length() + 1);
        }
        attrs.driver = getDriver();
        attrs.url = getURL();
        attrs.max = 30;
        ds = factory.createJDBCDataSource(attrs);
        try {
          factory.bindDataSource(ds);
        } catch (javax.naming.NamingException namingExc){
        }
        conn = ds.getConnection(userID, password);
      }
      catch (Throwable t1)
      {
        t1.printStackTrace();
      }
    }
    return conn;
  }

/*****************************************************************************
  * Get method for the SQL statement property
  * @return the value of the SQL statement property

  */
  public java.lang.String getSQLString(){
    return SQLString;
  }

/*****************************************************************************
  * Get method for the URL property
  * @return the value of the URL property

  */
  public java.lang.String getURL(){
    return URL;
  }

/*****************************************************************************
  * Get method for the indexed EMPNO property
  * @param index The 0 based index of the desired value
  * @return the value of the EMPNO property at the specified index
  */
  public java.lang.String getUSER_ID(int index)
        throws java.lang.IndexOutOfBoundsException,
java.lang.ArrayIndexOutOfBoundsException
  {
     return (java.lang.String) valueAtColumnRow(UID_COLUMN, index);

  }

/***************************************************************************
  * Get method for the userID property
  * @return the value of the userID property

  */
  public java.lang.String getUserID(){
    return userID;
  }

/***************************************************************************
  * Initializes the App Server V3 data acess beans
  *
  * @exception com.ibm.db.DataException when a database access exception occurs
  * @exception java.io.IOException when an IO error occurs
  */
  protected void initialize() throws DataException, IOException

  {
    StatementMetaData metaData = null;

    // Instantiate the connection bean and initialize it
    connection = getPooledConnection(getDriver(), getURL(), getUserID(),
password);
    connBean = new DatabaseConnection(connection);


    if (connBean == null)
    {
      return ;
    }

    // The statement must reference the connection to be used
        sqlStatement = new SelectStatement();
    sqlStatement.setConnection(connBean);

    // Add the SQL string to the metaData
    metaData = sqlStatement.getMetaData();
    metaData.setSQL(getSQLString());

    // Each table the query uses is added to the metadata. Then each column
that
    // is returned from the select gets added to the metadata.
    metaData.addTable("WSDEMO.EMPLOYEE");
```

```
     metaData.addColumn("LASTNAME", java.lang.String.class, 12);
    metaData.addColumn("FIRSTNME", java.lang.String.class, 12);
    metaData.addColumn("EMPNO", java.lang.String.class, 1);
    metaData.addColumn("USERID", java.lang.String.class, 1);

     // Create placeholders for the parameters

     return;
   }

 /****************************************************************************
   * Set method for the App Server V3 dataSourceName property
   */
   public void setDataSourceName(String value)
   {
     this.dataSourceName = value;
   }

 /****************************************************************************
   * Set method for the driver property
   * @param value the new value for the driver property

   */
   public void setDriver(java.lang.String value){
     this.driver = value;
   }

 /****************************************************************************
   * Set method for the password property
   * @param value the new value for the password property

   */
   public void setPassword(java.lang.String value){
     this.password = value;
   }

 /****************************************************************************
   * Set method for the URL property
   * @param value the new value for the URL property

   */
   public void setURL(java.lang.String value){
     this.URL = value;
   }

 /****************************************************************************
   * Set method for the userID property
   * @param value the new value for the userID property
```

```
  */
  public void setUserID(java.lang.String value){
    this.userID = value;
  }

/**************************************************************************
  * Utility method to get the value at a specific row and column index
  *
  * @param column the column containing the desired data
  * @param row the row containing the desired data
  * @return the value of the column at the specified row
  * @exception java.lang.ArrayIndexOutOfBoundsException thrown when there is no
data at the specified row
  */
  private Object valueAtColumnRow(int column, int row) throws
ArrayIndexOutOfBoundsException
  {
    // Index is 0 based but rows are 1 based, so increment the index
    int realRow = row + 1;

    // Handle an empty result set by throwing an exception
    if (result == null)
    {
      throw new ArrayIndexOutOfBoundsException ("Result set is empty.");
    }

    // Handle an out of bounds index by throwing an exception
    if (realRow > result.getNumRowsInCache())
    {
      throw new ArrayIndexOutOfBoundsException ("Row is out of bounds.");
    }

    // Adjust the current row to the desired row index
    try
    {
      result.setCurrentRow(realRow);
    }
    catch (Exception e)
    {
      System.out.println("Error occurred in
com.ibm.db.SelectResult.setCurrentRow");
      e.printStackTrace();
    }

    // Return the indexed property element
    try
    {
      return result.getColumnValue(column);
    }
    catch (Exception e)
    {
      System.out.println("Error occurred in
com.ibm.db.SelectResult.getColumnValue");
      e.printStackTrace();
    }
    return null;
  }
}
```

Example: A-3 itso.wes.ins.samples.Notify.java

```
package itso.wes.ins.samples;

import WebSphereSamples.YourCo.Meeting.*;
import java.io.*;
import java.util.*;
import java.security.cert.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.pvc.we.ins.*;
import WebSphereSamples.YourCo.Meeting.ScheduleDBBean;
import WebSphereSamples.YourCo.Login.SessionBean;
import java.sql.*;

/**
 * Notify Servlet returns information about the request. This servlet is
 * useful for checking the request parameters from a particular client.
 * Notify also returns information of existing sessions, application
 * attributes, and request attributes.
 *
 * @version 1.0
 */
public class Notify extends HttpServlet
{
   private java.lang.String fromUser = null;
   private java.lang.String host = null;
   private int port;
   private com.ibm.pvc.we.ins.NotificationService ns;
   private com.ibm.pvc.we.ins.DeliveryOptions opts;
   private java.lang.String notificationResult =
"/YourCo/Meeting/Notifications.jsp";
```

```
   //for triggered notifications
   // URL for the database
   String dbUrl = "jdbc:db2:sample";

/**
 *
 * Creation date: (9/19/2001 8:07:36 PM)
 * @param userID java.lang.String
 * @param roomId java.lang.String
 * @param day java.lang.String
 * @param time java.lang.String
 * @param convenor java.lang.String
 */
private void addMeetingForTrigger(
    String userID,
    String roomId,
    String day,
    String time,
    String convenor) {

    try {

        String url = "jdbc:db2:sample";
        Connection con = DriverManager.getConnection(url, "wsdemo", "wsdemo1");

        // retrieve data from the database
        System.out.println("Insert data into the database...");
        Statement stmt = con.createStatement();
        String queryString = "INSERT INTO WSDEMO.TRIGGER "
                             + "VALUES("
                             + "'" + userID.trim() + "', "
                             + "'" + roomId.trim() + "', "
                             + "'" + day.trim() + "', "
                             + "'" + time.trim() + "', "
                      + "'" + convenor.trim() + "'"
                             + ")";

        System.out.println ("QueryString: " + queryString);
      int result =  stmt.executeUpdate(queryString);
       System.out.println ("Result of update");

       System.out.println("resultCode: " + result);
       System.out.println("SQLCODE: " + stmt.getWarnings());
    } catch (Exception ex) {
        System.out.println("Problem saving input for meeting trigger");
        ex.printStackTrace();
    }
}
```

```
public void doGet(HttpServletRequest req, HttpServletResponse res) {

    try {
        performTask(req, res);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
public void doPost(HttpServletRequest req, HttpServletResponse res) {

    try {
        performTask(req, res);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
/**
 *
 * Creation date: (9/15/2001 9:17:32 PM)
 */
public void init() {

    try {

       //for simple notification
        fromUser = "insuser@IBM";
        host = com.ibm.pvc.we.ins.util.SubscriptionUtility.getHost();
        port = 55005;
        ns = new NotificationService(host, port);
        opts = new DeliveryOptions();

        //for subscribe/trigger notification
        //prepare for DB2 access
       Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
public void performTask(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    System.out.println("\n\nStarting performATask()\n");

    Enumeration e = req.getParameterNames();
    Vector userIds = new Vector();

    UserNotificationBean userNotification = new UserNotificationBean();
    String priority = null;
    String name = null;
    String value = null;
    String userId = null;
    //get the variables from the input
    if (e.hasMoreElements()) {
        while (e.hasMoreElements()) {
            name = (String) e.nextElement();
            value = req.getParameter(name).toString();
            System.out.println("parameter: " + name + "*** value: " + value);
            if (name.equals("Priority") && req.getParameter("Priority") !=
null) {
                priority = req.getParameter("Priority");
            }
            if (name.startsWith("notify:") &&
req.getParameter(name).equals("on")) {
                //the name of the user is the value of the parameter
                name = name.substring("notify:".length());
                userIds.add(name);
            }
        }
    }

    //get variables from the session
    SessionBean sessionBean = null;
    ScheduleDBBean scheduleDBBean = null;
    InviteesDBBean inviteesDBBean = null;

    HttpSession session = req.getSession(false);

    if (session != null) {
        sessionBean = (SessionBean) session.getAttribute("sessionBean");
        scheduleDBBean = (ScheduleDBBean)
session.getAttribute("scheduleDBBean");
        inviteesDBBean = (InviteesDBBean)
session.getAttribute("InviteesDBBean");
    }
    if (scheduleDBBean == null || sessionBean == null) {
        //probably testing
        //setup the session for testing
        sessionBean = new SessionBean();
        scheduleDBBean = new ScheduleDBBean();
        scheduleDBBean.setDay("friday");
        scheduleDBBean.setRoom("A");
        scheduleDBBean.setTime("A.M.");

        sessionBean.setEmpno("069897");
        sessionBean.setFirstName("Erik");
        sessionBean.setLastName("Rongen");
        //end of test code
```

```
}
try {

    String meetingDay = scheduleDBBean.getDay();
    String meetingRoom = scheduleDBBean.getRoom();
    String meetingTime = scheduleDBBean.getTime();

    String convenorFirstName = sessionBean.getFirstName();
    String convenorLastName = sessionBean.getLastName();

    //Ok. we've got everything. Now we notify the users
    userNotification.setPriority(priority);
    userNotification.setDay(meetingDay);
    userNotification.setTime(meetingTime);
    userNotification.setRoom(meetingRoom);

    //      opts.devices = opts.EMAIL;
    //      opts.devices = opts.IM;
    opts.devices = opts.WAP;
    opts.priority = opts.NORMAL;
    opts.multiDevices = opts.ANY;

    String toUser = null;
    int rc;
    Enumeration ids = userIds.elements();
    while (ids.hasMoreElements()) {

        toUser = (String) ids.nextElement();
        String undMsg =
            "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n"
                + "<message>\n"
                + "  <to>"
                + toUser
                + "</to>\n"
                + "  <from>"
                + fromUser
                + "</from>\n"
                + "  <subject>Meeting Schedule</subject>\n"
                + "  <text>\n"
                + "    You are requested to attend a meeting: \n"
                + "    Day : "
                + meetingDay
                + "\n"
                + "    Time: "
                + meetingTime
                + "\n"
                + "    Room: "
```

```
                        + meetingRoom
                        + "\n"
                        + "    By  : "
                        + convenorFirstName
                        + " "
                        + convenorLastName
                        + "\n"
                        + "  </text>\n"
                        + "</message>";

                try {
                    userNotification.addUserId(toUser);
                    userNotification.putUserNotiFicationStatus(toUser, false);
                    System.out.println(
                        "Calling sendMessage, toUser = " + toUser + ", fromUser = "
+ fromUser);
                    System.out.println("  host = " + host + ", port = " + port);
                    rc = ns.sendMessage(undMsg, opts);
                    System.out.println("Return from sendMessage, rc = " + rc);

                    if (rc == 0) {
                        //only if sending was successfull, set status to true
                        userNotification.putUserNotiFicationStatus(toUser, true);
                        System.out.println("Message successfully sent to " +
toUser);

                    } else {
                        System.out.println(
                            "Failed to send message to " + toUser + ". Return code:
" + rc + ".");

                    }
                    //and add the info to the trigger table for the
                    //subscribe/trigger notification

                } catch (Exception exception) {

                    System.out.println("Problem notifying user " + toUser + ":");
                    exception.printStackTrace();
                }
                addMeetingForTrigger(toUser,
                      meetingRoom,
                      meetingDay,
                      meetingTime,
                      convenorFirstName + " " + convenorLastName);


            }
```

```
        //dispatch to the jsp
        if (userNotification != null && scheduleDBBean != null) {
            //Both beans have been populated. Show result
            req.setAttribute("userNotificationBean", userNotification);
            // req.setAttribute("scheduleDBBean", scheduleDBBean);

            RequestDispatcher rd =
                getServletContext().getRequestDispatcher(notificationResult);
            rd.forward(req, res);
        }
    } catch (Exception ex) {
        System.out.println("Exception in Notify: " + ex.toString());
        ex.printStackTrace();
    }

}
    private void print (PrintWriter out, String name, int value)
    {
        out.print("<tr><td>" + name + "</td><td>");
        if ( value == -1 )
        {
            out.print("&lt;none&gt;");
        }
        else
        {
            out.print(value);
        }
        out.println("</td></tr>");
    }
    private void print (PrintWriter out, String name, String value)
    {
        out.println("<tr><td>" + name + "</td><td>" + (value == null ?
"&lt;none&gt;" : value) + "</td></tr>");
    }
}
```

Example: A-4 itso.wes.ins.samples.UserNotificationBean.java

```
package itso.wes.ins.samples;

/**
 *
 * Creation date: (9/15/2001 9:35:24 PM)
 * @author: Erik Rongen
 */
```

```
public class UserNotificationBean {
   private java.util.Hashtable userNotificationStatus = new
java.util.Hashtable();
   private java.lang.String room;
   private java.lang.String day;
   private java.lang.String time;
   private java.lang.String priority;
   private java.util.Vector userIds = new java.util.Vector();
   private java.util.Hashtable userNames = new java.util.Hashtable();
/**
 * UserNotificationBean constructor comment.
 */
public UserNotificationBean() {
   super();
}
/**
 *
 * Creation date: (9/15/2001 9:36:36 PM)
 * @param userName java.lang.String
 */
public void addUserId(String userName) {
   userIds.add(userName);}
/**
 *
 * Creation date: (9/15/2001 11:00:52 PM)
 * @return java.lang.String
 */
public java.lang.String getDay() {
   return day;
}
/**
 *
 * Creation date: (9/15/2001 11:01:27 PM)
 * @return java.lang.String
 */
public java.lang.String getPriority() {
   return priority;
}
/**
 *
 * Creation date: (9/15/2001 11:00:33 PM)
 * @return java.lang.String
 */
public java.lang.String getRoom() {
   return room;
}
/**
 *
 * Creation date: (9/15/2001 11:01:15 PM)
 * @return java.lang.String
 */
public java.lang.String getTime() {
   return time;
}
/**
 *
 * Creation date: (9/15/2001 9:41:52 PM)
 * @return java.util.Enumeration
 */
public java.util.Enumeration getUserIds() {
   return userIds.elements();
}
/**
 *
 * Creation date: (9/17/2001 11:34:06 AM)
 * @return java.lang.String
 * @param userId java.lang.String
 */
public String getUserName(String userId) {
   return (String)userNames.get(userId);

}
/**
 *
 * Creation date: (9/15/2001 9:43:45 PM)
 * @return boolean
 * @param userName java.lang.String
 */
public boolean NotificationIsSuccessfull(String userId) {
   return ((Boolean)userNotificationStatus.get(userId)).equals(Boolean.TRUE);

}
/**
 *
 * Creation date: (9/15/2001 9:40:08 PM)
 * @param userName java.lang.String
 * @param statu boolean
 */
public void putUserName(String userId, String userName) {
   userNames.put(userId, userName);
   }
/**
 *
 * Creation date: (9/15/2001 9:40:08 PM)
 * @param userName java.lang.String
 * @param statu boolean
 */
public void putUserNotiFicationStatus(String userId, boolean status) {
   userNotificationStatus.put(userId, new Boolean(status));
   }
/**
 *
 * Creation date: (9/15/2001 11:00:52 PM)
 * @param newDay java.lang.String
 */
public void setDay(java.lang.String newDay) {
   day = newDay;
}
/**
 *
 * Creation date: (9/15/2001 11:01:27 PM)
 * @param newPriority java.lang.String
 */
public void setPriority(java.lang.String newPriority) {
   priority = newPriority;
}
/**
 *
 * Creation date: (9/15/2001 11:00:33 PM)
 * @param newRoom java.lang.String
 */
public void setRoom(java.lang.String newRoom) {
   room = newRoom;
}
/**
 *
 * Creation date: (9/15/2001 11:01:15 PM)
 * @param newTime java.lang.String
 */
public void setTime(java.lang.String newTime) {
   time = newTime;
}
/**
 *
 * Creation date: (9/15/2001 9:41:52 PM)
 * @return java.util.Enumeration
 */
public java.util.Enumeration userIds() {
   return userIds.elements();
}
}
```

Example: A-5 /YourCo/Meeting/ScheduleResults.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<!-- 5648-C84, 5648-C83, (C) Copyright IBM Corporation, 1997, 2000 -->
<!-- All rights reserved. Licensed Materials Property of IBM -->
<!-- Note to US Government users: Documentation related to restricted rights
-->
<!-- Use, duplication or disclosure is subject to restrictions set forth in GSA
ADP Schedule with IBM Corp. -->
<!--This page may contain other proprietary notices and copyright information,
the terms of which must be observed and followed. -->
<HTML>
<!-- This file was generated by IBM WebSphere Studio 3.5 using
g:\WebSphere\Studio\BIN\GenerationStyleSheets\V3.5\JSP1.0\ServletModel\HTMLPage
s.xsl -->
<HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Page Designer V3.5 for Windows">
<LINK REL=STYLESHEET HREF="/WebSphereSamples/theme/YourCo.css" TYPE="text/css">
</HEAD>

<BODY background="/WebSphereSamples/theme/bg.gif">

 <jsp:useBean id="scheduleDBBean"
type="WebSphereSamples.YourCo.Meeting.ScheduleDBBean" scope="session" />
 <jsp:useBean id="sessionBean" type="WebSphereSamples.YourCo.Login.SessionBean"
scope="session" />

 <%

 try {

 java.lang.String _p0_1 = scheduleDBBean.getDay(); //throws an exception if
empty
 java.lang.String _p0_2 = scheduleDBBean.getRoom(); //throws an exception if
empty
 java.lang.String _p0_3 = scheduleDBBean.getTime(); //throws an exception if
empty

 java.lang.String firstName = sessionBean.getFirstName();
 java.lang.String lastName = sessionBean.getLastName();
 %>

<CENTER>
<br>
<P>The following conference room reservation has been successfully
submitted:</P>
```

```
 <TABLE border="0">
  <TR align="left" bgcolor="#006699">
   <TD><B>Day</B></TD>
   <TD><B>Room</B></TD>
   <TD><B>Time</B></TD>
   <TD><B>Employee</B></TD>
  </TR>
  <TR bgcolor="#99CCFF">
   <TD><%= _p0_1 %> </TD>
   <TD><%= _p0_2 %> </TD>
   <TD><%= _p0_3 %> </TD>
   <TD><%= firstName %> <%= lastName %></TD>
  </TR>
 </TABLE>
<p><a
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Meeting.Invitees">Selec
t Invitees</a>.</p>
<p>Return to the <a
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Meeting.SelectNames">Yo
urCo Conference Room Scheduler page</a>.</p>

 <%

 }
  catch (java.lang.ArrayIndexOutOfBoundsException _e0) {
 }%>

 <%scheduleDBBean.closeResultSet();%>

 </CENTER>
 </BODY>
</HTML>
```

Example: A-6   WebSphereSamples.YourCo.Meeting.InviteesDBBean.java

package WebSphereSamples.YourCo.Meeting;

```
// Imports

import com.ibm.db.*;
import com.ibm.webtools.runtime.*;
import java.io.*;
import java.math.*;
```

```
// Imports for App Server V3 connection pooling
import javax.sql.DataSource;
import com.ibm.ejs.dbm.jdbcext.*;
import javax.naming.*;
import com.ibm.ejs.ns.jndi.*;
import java.sql.*;
import java.util.*;


// Imports for App Server V3.5 connection pooling
import com.ibm.websphere.advanced.cm.factory.*;



/**  */

public class InviteesDBBean extends java.lang.Object
{

  private static final int LASTNAME_COLUMN = 1;
  private static final int FIRSTNME_COLUMN = 2;
  private static final int EMPNO_COLUMN = 3;
  private static final int UID_COLUMN = 4;

  /**
  * Instance variable for driver property
  */
  protected java.lang.String driver= null;


  /**
  * Instance variable for password property
  */
  protected java.lang.String password= null;


  /**
  * Instance variable for SQL statement property
  */
  protected java.lang.String SQLString = "SELECT WSDEMO.EMPLOYEE.LASTNAME AS
LASTNAME, WSDEMO.EMPLOYEE.FIRSTNME AS FIRSTNME, WSDEMO.EMPLOYEE.EMPNO AS
EMPNO1, WSDEMO.EMPLOYEE.USERID AS USERID FROM WSDEMO.EMPLOYEE ORDER BY
LASTNAME, FIRSTNME";


  /**
  * Instance variable for URL property
  */
```

```
 protected java.lang.String URL= null;


 /**
 * Instance variable for userID property
 */
 protected java.lang.String userID= null;



 /**
 * Variable for the SelectStatement
 */
 protected SelectStatement sqlStatement;


 /**
 * Variable for the SelectResult - the SQL result set
 */
 protected SelectResult result;

 /**
 * Variable to hold the database connection bean
 */
 protected DatabaseConnection connBean = null;


 /**
 * Variable to hold DataSource
 */
 protected DataSource ds = null;

 /**
 * Variable to hold connection
 */
 protected Connection connection = null;

 /**
 * Variable to hold the data source name from .servlet file
 */
 protected String dataSourceName = "";




/******************************************************************************
 * Close the result set and release resources
 */
 public void closeResultSet()
```

```
  {
    // Release the SQL statement resources
    try
    {
      if (result != null)
      {
        result.close();
        result = null;
      }
    }
    catch (Exception e)
    {
      System.out.println("Error occurred in com.ibm.db.SelectResult.close");
      e.printStackTrace();
    }
    try
    {
      // Close the App Server V3 connection
      if (connection != null)
        connection.close();
    }
    catch (Exception e)
    {
      System.out.println("Error occurred in java.sql.Connection.close");
      e.printStackTrace();
    }
    return ;
  }

/****************************************************************************
  * execute action method   * @exception com.ibm.db.DataException
  * @exception java.io.IOException
  * @exception java.lang.Exception

  */
  public void execute()
   throws com.ibm.db.DataException, java.io.IOException, java.lang.Exception
  {

    initialize();

    // Initialize the parameters for the query


    // Execute the SQL statement
    sqlStatement.execute();
    result = sqlStatement.getResult();

  }
```

```
/****************************************************************************
  * Release resources we might be holding when garbage collection occurs.
  */
  protected void finalize() throws Throwable
  {
    closeResultSet();
  }

/****************************************************************************
  * Get method for the App Server V3 dataSourceName property
  */
  public String getDataSourceName()
  {
    return dataSourceName;
  }

/****************************************************************************
  * Get method for the driver property
  * @return the value of the driver property

  */
  public java.lang.String getDriver(){
    return driver;
  }

/****************************************************************************
  * Get method for the indexed EMPNO property
  * @param index The 0 based index of the desired value
  * @return the value of the EMPNO property at the specified index
  */
  public java.lang.String getEMPNO(int index)
        throws java.lang.IndexOutOfBoundsException,
java.lang.ArrayIndexOutOfBoundsException
  {

     return (java.lang.String) valueAtColumnRow(EMPNO_COLUMN, index);

  }

/****************************************************************************
  * Get method for the indexed FIRSTNME property
  * @param index The 0 based index of the desired value
  * @return the value of the FIRSTNME property at the specified index
  */
  public java.lang.String getFIRSTNME(int index)
        throws java.lang.IndexOutOfBoundsException,
java.lang.ArrayIndexOutOfBoundsException
  {
     return (java.lang.String) valueAtColumnRow(FIRSTNME_COLUMN, index);

  }

/*****************************************************************************
  * Get method for the indexed LASTNAME property
  * @param index The 0 based index of the desired value
  * @return the value of the LASTNAME property at the specified index
  */
  public java.lang.String getLASTNAME(int index)
   throws java.lang.IndexOutOfBoundsException,
java.lang.ArrayIndexOutOfBoundsException
  {

    return (java.lang.String) valueAtColumnRow(LASTNAME_COLUMN, index);

  }

/*****************************************************************************
  * Attempts to get a connection from a App Server V3 connection pool.
  *
  * @param driver Contains the JDBC driver name to use for the connection
  * @param URL Contains the database url for the connection
  * @param userID Contains the userid to use for the database connection
  * @param password Contains the password to use for the connection
  * @return a pooled JDBC connection or null
  */
  protected Connection getPooledConnection(String driver, String URL, String
userID, String password)
  {
    Connection conn = null;

    try
    {
      // create parameter list to access naming system
      Hashtable parms = new Hashtable();
      parms.put(Context.INITIAL_CONTEXT_FACTORY,
CNInitialContextFactory.class.getName());
      // access naming system
      Context context = new InitialContext(parms);
      // get DataSource factory object from naming system
      ds = (DataSource)context.lookup(getDataSourceName());
      conn = ds.getConnection(userID, password);
    }
    catch (Throwable t)
    {
      // DataSource not found. Try to construct a new DataSource.
      try
      {
        t.printStackTrace();
        com.ibm.websphere.advanced.cm.factory.DataSourceFactory factory =
          new com.ibm.websphere.advanced.cm.factory.DataSourceFactory();
        Attributes attrs = new Attributes();
        attrs.name = getDataSourceName();
        if
(attrs.name.startsWith(com.ibm.websphere.advanced.cm.factory.DataSourceFactory.
DEFAULT_DATASOURCE_CONTEXT_NAME + "/"))
        {
      attrs.name =
attrs.name.substring(com.ibm.websphere.advanced.cm.factory.DataSourceFactory.DE
FAULT_DATASOURCE_CONTEXT_NAME.length() + 1);
        }
        attrs.driver = getDriver();
        attrs.url = getURL();
        attrs.max = 30;
        ds = factory.createJDBCDataSource(attrs);
        try {
          factory.bindDataSource(ds);
        } catch (javax.naming.NamingException namingExc){
        }
        conn = ds.getConnection(userID, password);
      }
      catch (Throwable t1)
      {
        t1.printStackTrace();
      }
    }
    return conn;
  }

/*****************************************************************************
  * Get method for the SQL statement property
  * @return the value of the SQL statement property

  */
  public java.lang.String getSQLString(){
    return SQLString;
  }

/*****************************************************************************
  * Get method for the URL property
  * @return the value of the URL property

  */
  public java.lang.String getURL(){
    return URL;
  }
```

```
/*****************************************************************************
  * Get method for the indexed EMPNO property
  * @param index The 0 based index of the desired value
  * @return the value of the EMPNO property at the specified index
  */
  public java.lang.String getUSER_ID(int index)
        throws java.lang.IndexOutOfBoundsException,
java.lang.ArrayIndexOutOfBoundsException
  {

     return (java.lang.String) valueAtColumnRow(UID_COLUMN, index);

  }

/*****************************************************************************
  * Get method for the userID property
  * @return the value of the userID property

  */
  public java.lang.String getUserID(){
    return userID;
  }

/*****************************************************************************
  * Initializes the App Server V3 data acess beans
  *
  * @exception com.ibm.db.DataException when a database access exception occurs
  * @exception java.io.IOException when an IO error occurs
  */
  protected void initialize() throws DataException, IOException

  {
    StatementMetaData metaData = null;

    // Instantiate the connection bean and initialize it
    connection = getPooledConnection(getDriver(), getURL(), getUserID(),
password);
    connBean = new DatabaseConnection(connection);


    if (connBean == null)
    {
      return ;
    }

    // The statement must reference the connection to be used
        sqlStatement = new SelectStatement();
    sqlStatement.setConnection(connBean);
    // Add the SQL string to the metaData
    metaData = sqlStatement.getMetaData();
    metaData.setSQL(getSQLString());

    // Each table the query uses is added to the metadata. Then each column
that
    // is returned from the select gets added to the metadata.
    metaData.addTable("WSDEMO.EMPLOYEE");

    metaData.addColumn("LASTNAME", java.lang.String.class, 12);
   metaData.addColumn("FIRSTNME", java.lang.String.class, 12);
   metaData.addColumn("EMPNO", java.lang.String.class, 1);
   metaData.addColumn("USERID", java.lang.String.class, 1);

    // Create placeholders for the parameters

    return;
  }

/*****************************************************************************
  * Set method for the App Server V3 dataSourceName property
  */
  public void setDataSourceName(String value)
  {
    this.dataSourceName = value;
  }

/*****************************************************************************
  * Set method for the driver property
  * @param value the new value for the driver property

  */
  public void setDriver(java.lang.String value){
    this.driver = value;
  }

/*****************************************************************************
  * Set method for the password property
  * @param value the new value for the password property

  */
  public void setPassword(java.lang.String value){
    this.password = value;
  }

/*****************************************************************************
  * Set method for the URL property
  * @param value the new value for the URL property
  */
  public void setURL(java.lang.String value){
    this.URL = value;
  }

/*****************************************************************************
  * Set method for the userID property
  * @param value the new value for the userID property

  */
  public void setUserID(java.lang.String value){
    this.userID = value;
  }

/*****************************************************************************
  * Utility method to get the value at a specific row and column index
  *
  * @param column the column containing the desired data
  * @param row the row containing the desired data
  * @return the value of the column at the specified row
  * @exception java.lang.ArrayIndexOutOfBoundsException thrown when there is no
data at the specified row
  */
  private Object valueAtColumnRow(int column, int row) throws
ArrayIndexOutOfBoundsException
  {
    // Index is 0 based but rows are 1 based, so increment the index
    int realRow = row + 1;

    // Handle an empty result set by throwing an exception
    if (result == null)
    {
      throw new ArrayIndexOutOfBoundsException ("Result set is empty.");
    }

    // Handle an out of bounds index by throwing an exception
    if (realRow > result.getNumRowsInCache())
    {
      throw new ArrayIndexOutOfBoundsException ("Row is out of bounds.");
    }

    // Adjust the current row to the desired row index
    try
    {
      result.setCurrentRow(realRow);
    }
    catch (Exception e)
    {
      System.out.println("Error occurred in
com.ibm.db.SelectResult.setCurrentRow");
      e.printStackTrace();
    }

    // Return the indexed property element
    try
    {
      return result.getColumnValue(column);
    }

    catch (Exception e)
    {
      System.out.println("Error occurred in
com.ibm.db.SelectResult.getColumnValue");
      e.printStackTrace();
    }
    return null;
  }
}
```

Subscription

Example: A-7   itso.wes.ins.triggersample.MeetingSubscriptionStartServlet.java

```
package itso.wes.ins.triggersample;

/*******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 *******************************************************************************
*/

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import WebSphereSamples.YourCo.Login.SessionBean;
import java.sql.*;


/* This is a subscriber servlet which allows a WES user to subscribe
to meeting content.  This servlet uses the INS API to
subscribe to a publish/subscribe system */

public class MeetingSubscriptionStartServlet extends HttpServlet
{

/*
 * Method Name: doPost
 */
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

       String userId = null;
    try {
        //get session info
        HttpSession session = request.getSession();
        SessionBean sessionBean = (SessionBean)
session.getValue("sessionBean");

        String empNo = sessionBean.getEmpno();

        String url = "jdbc:db2:sample";
        Connection con = DriverManager.getConnection(url, "wsdemo", "wsdemo1");

        // retrieve data from the database
        System.out.println("Retrieve some data from the database...");
        Statement stmt = con.createStatement();
        String queryString =
            "SELECT USERID FROM WSDEMO.EMPLOYEE " + " WHERE EMPNO = '" + empNo
+ "'";
        ResultSet rs = stmt.executeQuery(queryString);

            System.out.println("Received results:");
            //rs.next is true when rows are found.
            //return the first found employee who meets the criteria
            if (rs.next()) {
                //Info of the employee
                userId = rs.getString("USERID");
            }


        RequestDispatcher rd =
getServletContext().getRequestDispatcher("/YourCo/Trigger/MeetingSubscriptionFo
rm.jsp?userid=" + userId);
        rd.forward(request, response);

    } catch (Exception e) {
        e.printStackTrace();
    }

} // end of doPost method
/*
 * Method Name: init
 */
public void init() throws ServletException {

    // URL for the database
    String dbUrl = "jdbc:db2:sample";
    try {
        Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();

    } catch (Exception e) {

    }
}
}               // end of MeetingSubscriptionServlet class
```

Example: A-8 itso.wes.ins.triggersample.MeetingSubscriptionServlet.java

```
package itso.wes.ins.triggersample;

/*******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                 *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved             *
 *                                                                             *
 *  Licensed Materials - Property of IBM                                       *
 *                                                                             *
 *  US Government Users Restricted Rights - Use, duplication or disclosure     *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.              *
 *******************************************************************************
*/

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.*;
import MeetingSubscriptionData;
import MeetingSubscriptions;
import com.ibm.pvc.we.ins.util.SubscriptionUtility;

/* This is a subscriber servlet which allows a WES user to subscribe
to meeting content.  This servlet uses the INS API to
subscribe to a publish/subscribe system */

public class MeetingSubscriptionServlet extends HttpServlet {
    private final static String insIBMCopyright =
GeneralConstants.insIBMCopyright;
    static final String topicMeetings = "meetings";
    private MeetingSubscriptions meetingDataList;
    private int identifier;
    private String userid = null;

    //IQ Server properties
    private String IQueuehost = null;
    private String IQueueport = null;


    /*
     * Method Name: doPost
     */
    public void doPost(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        userid = SubscriptionUtility.getUser(request);

        if (request.getParameter("add") != null) //Add button was pressed
            meetingAddData(request, response);
        else
            if (request.getParameter("refresh") != null) //Refresh button was
pressed
                meetingDeleteData(request, response);
            else //Submit button was pressed
                meetingSubscription(request, response);
    } // end of doPost method
/*
 * Method Name: init
 */
public void init() throws ServletException {
    meetingDataList = new MeetingSubscriptions();
    identifier = 1;
    try{
    //Get the IQueue server host and port from the properties file
    IQueuehost = SubscriptionUtility.getHost();
    IQueueport = SubscriptionUtility.getPort();

    //In this example we simply hard code these values
    //IQueuehost = "rs615002.itso.ral.ibm.com";
    //IQueueport = "55001";
    }
    catch (Exception e)
    {
    e.printStackTrace();
       }


}
    /*
     * Method Name: meetingAddData
     */
    public void meetingAddData(
        HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        MeetingSubscriptionData meetingData = new MeetingSubscriptionData();
        Integer ident = new Integer(identifier);

        meetingData.setId(ident.toString());

        //get Data from the request
        meetingData.setHours(request.getParameter("meetingHours"));
        meetingData.setTriggerOption(request.getParameter("trigopt"));
        meetingData.setContentOption(request.getParameter("contopt"));
        meetingDataList.setSubscription(userid, meetingData);

        //add the data to the list of meetingData
        identifier++;
        request.setAttribute("meetingSubscriptions", meetingDataList);

        RequestDispatcher dispatcher =

request.getRequestDispatcher("/YourCo/Trigger/MeetingSubscriptionForm.jsp");
        dispatcher.forward(request, response);
    }
    /*
     * Method Name: meetingDeleteData
     */
    public void meetingDeleteData(
        HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        String[] deletes = request.getParameterValues("delete");

        if (deletes != null) //Check for Refresh pressed, but no subscriptions
checked
            {
            try {
                for (int x = 0; x < deletes.length; x++) {
                    MeetingSubscriptionData ssd =
meetingDataList.getSubscription(userid, 0);
                    //Throws an exception if empty
                    for (int i = 0;;) {
                        if (ssd.getId().equals(deletes[x])) {
                            meetingDataList.removeSubscription(userid, i);
                            break;
                        } else {
                            i++;
                            try {
                                ssd = meetingDataList.getSubscription(userid,
i);
                            } catch (java.lang.ArrayIndexOutOfBoundsException
e) {
                                break;
                            }
                        }
                    }
                }
            } catch (java.lang.ArrayIndexOutOfBoundsException e) {
            }
        }

        request.setAttribute("meetingSubscriptions", meetingDataList);

        RequestDispatcher dispatcher =

request.getRequestDispatcher("/YourCo/Trigger/MeetingSubscriptionForm.jsp");
        dispatcher.forward(request, response);
    }
    /*
     * Method Name: meetingSubscription
     */
    public void meetingSubscription(
        HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {

       //Get the server name and port number on which this servlet is running.
This will be used
```

```
        //in the notification message to the user that has requested that the
message content be saved.
        //It will be used by the user to retrieve the message from the content
store.
        //This method assumes that all servlets are running on the same server.

        final String server = request.getServerName();
        final int port = request.getServerPort();
        String path = request.getRequestURI();

        //

        //Variable Declarations
        Vector meetingHours = new Vector();
        Vector contentOptions = new Vector(); //Save or Don't Save
        Vector triggerOptions = new Vector(); //Once or Always
        String sqlsel; //SQL selector
        String URL; //URL for retention notification message

        //Prepare for output
        response.setContentType("text/html");

        //Construct the first part of the URL for any retention notification
messages that will be sent.
        //This URL will be passed to the WeatherHandler.
        URL =
            "http://" + server + ":" + port + path.substring(0,
path.lastIndexOf("/") + 1);

        try {
            MeetingSubscriptionData ssd =
meetingDataList.getSubscription(userid, 0);
            //Throws an exception if empty
            for (int i = 0;;) {
                meetingHours.add(ssd.getHours());
                triggerOptions.add(ssd.getTriggerOption());
                contentOptions.add(ssd.getContentOption());
                i++;
                try {
                    ssd = meetingDataList.getSubscription(userid, i);
                } catch (java.lang.ArrayIndexOutOfBoundsException e) {
                    break;
                }
            }
        } catch (java.lang.ArrayIndexOutOfBoundsException e) {
           //No subscriptions were submitted.  Display an error message to the
user.
            request.setAttribute("meetingSubscriptions", meetingDataList);
            request.setAttribute("message1", "<h3>No subscriptions were
specified</h3>");

            RequestDispatcher dispatcher =

request.getRequestDispatcher("/YourCo/Trigger/MeetingSubscriptionForm.jsp");
            dispatcher.forward(request, response);
            return;
        }

        // Connect to INS by calling a TriggerManager
        try {
            //Set up to register a TriggerHandler
            TriggerManager manager = new SocketClientStub(IQueuehost,
IQueueport);
            ContentSource source =
                SimpleContentSource.newSimpleContentSource(topicMeetings);

            ContentFilter filter;
            MeetingHandler triggerHand;
            TriggerID tid;

            for (int i = 0; i < meetingHours.size(); i++) {
                //Retrieve the trigger handler
                triggerHand =
                    new MeetingHandler(
                        userid,
                        (String) triggerOptions.elementAt(i),
                        (String) contentOptions.elementAt(i),
                        URL);
                sqlsel =
                    "(USERID = '"
                        + userid
                        + "') AND (TIME_TO_MEETING <= "
                        + meetingHours.elementAt(i)
                        + ")";
                filter = SqlSelector.newSqlSelector(sqlsel);
                //Call the trigger manager and add the trigger with parameters
                tid = manager.addTrigger(userid, source, filter, triggerHand);
            } // end of loop for trigger conditions

            //If no exception occurred, send back a success message
            meetingDataList.removeAllSubscriptions(userid);
            //Successful send, remove all subscriptions from table
            request.setAttribute("meetingSubscriptions", meetingDataList);
            request.setAttribute(
                "message1",
                "<h3>Your subscription has been accepted</h3>");
            RequestDispatcher dispatcher =

request.getRequestDispatcher("/YourCo/Trigger/MeetingSubscriptionForm.jsp");
            dispatcher.forward(request, response);
        } // end of try clause
        catch (IQueueException iqe) {
            iqe.printStackTrace(System.out);
            request.setAttribute("meetingSubscriptions", meetingDataList);
            request.setAttribute("message1", "<h3>IQueueException
Detected</h3>");
            request.setAttribute("message2", "<h4>" + iqe.getMessage() +
"</h4>");

            RequestDispatcher dispatcher =
                request.getRequestDispatcher("/MeetingSubscriptionForm.jsp");
            dispatcher.forward(request, response);
        } catch (ConfigurationException e) {
            e.printStackTrace(System.out);
            request.setAttribute("meetingSubscriptions", meetingDataList);
            request.setAttribute("message1", "<h3>ConfigurationException
Detected</h3>");
            request.setAttribute("message2", "<h4>" + e.getMessage() +
"</h4>");

            RequestDispatcher dispatcher =
                request.getRequestDispatcher("/MeetingSubscriptionForm.jsp");
            dispatcher.forward(request, response);
        }
        return;
    } // end of meetingSubscription method
} // end of MeetingSubscriptionServlet class
```

Example: A-9   itso.wes.ins.triggersample.MeetingSubscriptionData.java

```
package itso.wes.ins.triggersample;

/*******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                 *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved             *
 *                                                                             *
 *  Licensed Materials - Property of IBM                                       *
 *                                                                             *
 *  US Government Users Restricted Rights - Use, duplication or disclosure     *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.              *
 ******************************************************************************
*/

public class MeetingSubscriptionData
{
  private String triggerOption;
  private String contentOption;
  private String id;

   private java.lang.String hours = null;
  /*
   * Method Name: getContentOption
   */
  public String getContentOption()
  {
    return contentOption;
  }
/**
 *
 * Creation date: (9/20/2001 9:41:47 AM)
 * @return java.lang.String
 */
public java.lang.String getHours() {
   return hours;
}
  /*
   * Method Name: getId
   */
  public String getId()
  {
    return id;
  }
  /*
   * Method Name: getTriggerOption
   */
  public String getTriggerOption()
  {
    return triggerOption;
  }
  /*
   * Method Name: setContentOption
   */
  public void setContentOption(String contentOption)
  {
    this.contentOption = contentOption;
  }
/**
 *
 * Creation date: (9/20/2001 9:41:47 AM)
 * @param newHours java.lang.String
 */
public void setHours(java.lang.String newHours) {
   hours = newHours;
}
  /*
   * Method Name: setId
   */
  public void setId(String id)
  {
    this.id = id;
  }
  /*
   * Method Name: setTriggerOption
   */
  public void setTriggerOption(String triggerOption)
  {
    this.triggerOption = triggerOption;
  }
}
```

Example: A-10   itso.wes.ins.triggersample.MeetingSubscriptions.java

```
package itso.wes.ins.triggersample;

/*******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                 *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved             *
 *                                                                             *
 *  Licensed Materials - Property of IBM                                       *
 *                                                                             *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.              *
 *******************************************************************************
*/

import java.util.*;

public class MeetingSubscriptions
{
  private Vector meetingSubscriptionList;
  private Hashtable subHT;
  private MeetingSubscriptionData subscription;
  /*
   * Constructer: MeetingSubscriptions
   */
   public MeetingSubscriptions()
   {
     subHT = new Hashtable();
   }
  /*
   * Method Name: getSubscription
   */
  public MeetingSubscriptionData getSubscription(String userid, int i) throws
ArrayIndexOutOfBoundsException
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       meetingSubscriptionList = (Vector)subHT.get(userid);
       return (MeetingSubscriptionData)meetingSubscriptionList.elementAt(i);
    }
    else
      throw new ArrayIndexOutOfBoundsException();
  }
  /*
   * Method Name: removeAllSubscriptions
   */
  public void removeAllSubscriptions(String userid)
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       meetingSubscriptionList = (Vector)subHT.get(userid);
       meetingSubscriptionList = null;//Allow the subscription List to be
garbage collected
       subHT.remove(userid);//Remove the userid from the Subscriptions
Hashtable
    }
  }
  /*
   * Method Name: removeSubscription
   */
  public void removeSubscription(String userid, int i) throws
ArrayIndexOutOfBoundsException
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       meetingSubscriptionList = (Vector)subHT.get(userid);
       meetingSubscriptionList.remove(i);
    }
    else
      throw new ArrayIndexOutOfBoundsException();
  }
  /*
   * Method Name: setSubscription
   */
  public void setSubscription(String userid, MeetingSubscriptionData
subscription)
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
      meetingSubscriptionList = (Vector)subHT.get(userid);
      meetingSubscriptionList.add(subscription);
    }
    else
    {
      meetingSubscriptionList = new Vector();
      meetingSubscriptionList.add(subscription);
      subHT.put(userid,meetingSubscriptionList);
    }
  }
}
```

Example: A-11   itso.wes.ins.triggersample.MeetingHandler.java

```
package itso.wes.ins.triggersample;

/*******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                 *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved             *
 *                                                                             *
 *  Licensed Materials - Property of IBM                                       *
 *                                                                             *
 *  US Government Users Restricted Rights - Use, duplication or disclosure     *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.              *
 *******************************************************************************
*/

import java.io.*;
import java.util.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.*;
import com.ibm.pvc.we.ins.util.SubscriptionUtility;

/* This is a trigger handler for a meeting subscription */

public class MeetingHandler extends TriggerHandler
{
   private final static String
insIBMCopyright=GeneralConstants.insIBMCopyright;
   private String triggerOption;
   private String contentOption;
   private String toUserid;
   private String meetingtype = "last";//Default to last
   private String notificationURL;


   public MeetingHandler(String userid, String trigopt, String contopt, String
url)
   {
      this.triggerOption = trigopt;
      this.contentOption = contopt;
      this.toUserid = userid;
      this.notificationURL = url;
   }
public TriggerResult handleMatch(ContentBody cb) {
    String contentString;
    String notificationString;
    String secondNotificationString;
    String value, change, percentage;
    StringBuffer message = new StringBuffer();
    StringBuffer message2;
    int tResult;

    String day = null;
    String time = null;
    String room = null;
    String convenor = null;
    String userid = null;

    //Set the Trigger Option based on user input
    System.out.println("MH => MeetingHandler.handleMatch() called");

    //change the notificationURL to another server
    System.out.println("MH => Old NotificationURL: " + notificationURL);
    String oldWebApp = "http://wtp3.itso.ral.ibm.com:80/WebSphereSamples";
    String newWebApp = "http://" + SubscriptionUtility.getHost() +
"/inssample";
    String uri = notificationURL.substring(oldWebApp.length());
    notificationURL = newWebApp + uri;
    System.out.println("MH => New NotificationURL: " + notificationURL);

    System.out.println("MH => triggerOption = " + triggerOption);
    System.out.println("MH => contentOption = " + contentOption);

    try {
        day = cb.getString("DAY");
        time = cb.getString("TIME");
        room = cb.getString("ROOM");
        convenor = cb.getString("CONVENOR");
        userid = cb.getString("USERID");
        //
        System.out.println("MH => Userid = " + userid);
        System.out.println("MH => Day = " + day);
        System.out.println("MH => Time = " + time);
        System.out.println("MH => Room = " + room);
        System.out.println("MH => Convenor = " + day);

    } catch (IQueueException iqe) {
        //
        System.out.println("MH => IQueueException for save message");

        iqe.printStackTrace(System.out);
        contentString = iqe.toString();
    }

    if (triggerOption.equals("once"))
        tResult = TriggerResult.STOP;
    else
        if (triggerOption.equals("always"))
            tResult = TriggerResult.CONTINUE;
        else
            tResult = TriggerResult.SAVE_AND_CONTINUE;

    if (contentOption.equals("save")) {
        //
        System.out.println("MH => Creating content for save");

        StringBuffer content = new StringBuffer();

        content.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
        content.append("<meeting>\n");
        content.append("  <convenor>" + convenor + "</convenor>\n");
        content.append("  <userid>" + userid + "</userid>\n");
        content.append("  <day>" + day + "</day>\n");
        content.append("  <time>" + time + "</time>\n");
        content.append("  <room>" + room + "</room>\n");
        content.append("</meeting>");
        contentString = content.toString();
        //
        System.out.println("MH => contentString = \n" + contentString);

        //
        System.out.println("MH => Creating message for save");

        message2 = new StringBuffer();
        message.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
        message.append("<message>\n");
        message.append("<to>" + toUserid + "</to>\n");
        message.append("<from>" + toUserid + "</from>\n");
        message.append("<subject>Meeting</subject>\n");
        message.append(
            "<text> Follow this URL " + notificationURL +
"ContTransServlet?pageid=");
        if (SubscriptionUtility.isSecure()) //Are authorization services
enabled?
            message2.append(" for details </text>\n");
        //Yes, do not need to pass userid on request
        else
            message2.append("&amp;userid=" + toUserid + " for details
</text>\n");
        //No, include userid on request
        message2.append("</message>");
        notificationString = message.toString();

        //
        System.out.println("MH => message = \n" + message);
        System.out.println("MH => message2 = \n" + message2);

        secondNotificationString = message2.toString();

        TriggerResult tr =
            TriggerResult.newRetentionSpecification(
                contentString,
                notificationString,
                secondNotificationString,
                "NOT_SAVED",
                15,
                tResult);

        //
        System.out.println("MH => triggerResult = \n" + tr.toString());
        System.out.println("MH => triggerResult = \n" + tr.toString());

        System.out.println(
            "MH => MeetingHandler.handleMatch() ended successfully for SAVED");
        return tr;
    } else // Option was chosen to not save the message
        {

        message.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
        message.append("<message>\n");
        message.append("<to>" + toUserid + "</to>\n");
        message.append("<from>" + toUserid + "</from>\n");
        message.append("<subject>Meeting</subject>\n");
        message.append(
            "<text> Meeting at " + day + " ," + time + " by " + convenor +
".</text>\n");
        message.append("</message>");
        notificationString = message.toString();
        System.out.println(
            "MH => MeetingHandler.handleMatch() ended successfully for NOT
SAVED");
        return TriggerResult.newNotificationSpecification(notificationString,
tResult);
    }
} //End of handleMatch method
}                               //End of MeetingHandler class
```

Example: A-12 itso.wes.ins.triggersample.MeetingContentAdapter.java

```
package itso.wes.ins.triggersample;

/*******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
import java.io.*;
import java.util.*;

import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.*;
import java.sql.*;


public class MeetingContentAdapter implements java.lang.Runnable
{
  private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;
  private ContentBody contentBody;
  private TriggerManager manager;
   private String IQproperties = "IQ";
   private SimpleContentSource contentSource;
   private long delayTime ;
public MeetingContentAdapter() {
    //Get the IQueue server host and port from the properties file
    try {
    PropertyResourceBundle pr =
        (PropertyResourceBundle)
PropertyResourceBundle.getBundle(IQproperties);
            String IQueuehost = pr.getString("iqSocketClientStub.host");
    String IQueueport = pr.getString("iqSocketClientStub.port");
        delayTime = 60000;

        //Set up to register a TriggerHandler
        manager = new SocketClientStub(IQueuehost, IQueueport);
        //Register the source of the content
        //This will change, perhaps to the names of the different sources such
as ap.business, etc.
        contentSource = SimpleContentSource.newSimpleContentSource("meetings");
        contentBody = new ContentBody();

        //prepare for DB2 access
        Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();
        //Register the source of the content
    } catch (Throwable ce) {
        System.out.println(ce.getMessage());
        return;
    }

} //End of constructer
/**
 *
 * Creation date: (9/20/2001 3:40:43 PM)
 * @return java.sql.ResultSet
 */
public ResultSet getDataFromDB() {

    try {
        String url = "jdbc:db2:sample";
        Connection con = DriverManager.getConnection(url, "wsdemo", "wsdemo1");

        // retrieve data from the database
        System.out.println("Retrieve trigger data from the database...");
        Statement stmt = con.createStatement();
        String queryString = "SELECT * FROM WSDEMO.TRIGGER";

        ResultSet rs = stmt.executeQuery(queryString);

        return rs;
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
/**
 *
 * Creation date: (9/20/2001 1:15:10 PM)
 */
public void loop() {
    try {

       //first receive the data from DB2
        ResultSet rs = getDataFromDB();

        //create the XML
        processContent(rs);

    } catch (Exception e) {
        e.printStackTrace();
    }
}
public static void main(String[] args) {
    //Pass in the name of the directory containing the XML files
    try {
        MeetingContentAdapter ca = new MeetingContentAdapter();

        Thread t = new Thread(ca);
            t.start();

        }
     catch (Exception e) {
        e.printStackTrace();
    }
} //End of main method
/**
 *
 * Creation date: (9/20/2001 4:00:30 PM)
 * @return java.lang.String
 * @param rs java.sql.ResultSet
 */
public void processContent(ResultSet rs) {

    int hoursUntilMeeting;
    String day;
    String time;
    String userid;
    String room;
    String convenor;
    try {
        while (rs.next()) {
            //Info of the employee
            //
            userid = rs.getString("USERID").trim();
            contentBody.setString("USERID", userid);

            day = rs.getString("THEDAY").trim();
            time = rs.getString("THETIME").trim();
            room = rs.getString("ROOM_ID").trim();
            convenor = rs.getString("RESERVED_BY").trim();

            contentBody.setString("DAY", day);
            contentBody.setString("TIME", time);
            contentBody.setString("ROOM", room);
            contentBody.setString("CONVENOR", convenor);

            //time left
           hoursUntilMeeting = timeLeftToMeeting(day, time);
            contentBody.setInt("TIME_TO_MEETING", hoursUntilMeeting);

                        System.out.println("\nCreating content:");
          System.out.println("   ->  UserId: " + userid);
          System.out.println("   ->  Day   : " + day);
          System.out.println("   ->  Time  : " + time);
          System.out.println("   ->  Room  : " + room);
          System.out.println("   ->  hours : " + hoursUntilMeeting);


        //and send the content to IQ
        sendContentToIQ();


        }
    } catch (Exception e) {
        e.printStackTrace();
    }

}
/**
 *
 * Creation date: (9/20/2001 1:15:10 PM)
 */
public void run() {
    try {
        while (true) {
            System.out.println("*** Starting one Loop ***");
            loop();
            Thread.currentThread().sleep(delayTime);

        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
/**
 *
 * Creation date: (9/20/2001 4:02:24 PM)
 * @return int
 */
public int sendContentToIQ() {
   try{
      System.out.println("Sending ContentBody");
          manager.fireMatchingTriggers(contentSource,contentBody);//Fire
matching triggers

   return 0;
   }
   catch(Exception e)

   {
      e.printStackTrace();
      return -10;
      }

}
/**
 *
 * Creation date: (9/20/2001 4:16:28 PM)
 * @return int
 * @param day java.lang.String
 * @param time java.lang.String
 */
public int timeLeftToMeeting(String day, String time) {
   return 2;
}
}     //End of MeetingContentAdapter class
```

Example: A-13   /YourCo/Triggers/MeetingSubscriptionForm.jsp

```
<!--
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)            *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved           *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
-->

<%@ page import="itso.wes.ins.triggersample.*, java.io.*, javax.servlet.*,
javax.servlet.http.*, javax.servlet.jsp.*" %>
<%@ page import="com.ibm.pvc.we.ins.util.SubscriptionUtility" %>
<%@ page info="The Meeting Subscription Form allows a user to subscribe to a
meeting and be notified when that meeting price hits a certain value." %>


<!-- This is an example of a subscription form for Meeting.
 * This jsp will call the MeetingSubscriptionServlet to process the final
request.
 -->



<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<!-- This file was generated by IBM WebSphere Studio 3.5 using
g:\WebSphere\Studio\BIN\GenerationStyleSheets\V3.5\JSP1.0\ServletModel\HTMLPage
s.xsl -->
<HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Page Designer V3.5 for Windows">
<LINK REL=STYLESHEET HREF="/WebSphereSamples/theme/YourCo.css" TYPE="text/css">
<title>Subscriptions</title>
  <script language=javascript>
      function validate()
      {
         <!-- Validate the input data -->
         if (document.form1.meetingHours.value == "")
         {
           alert('Meeting Hours is a required field. Please try again.');
           return false;
         }
         else
         {
           return true;
         }
       }

  </script>

<SCRIPT LANGUAGE="JavaScript1.2">
```

```
  positionArray = new Array(8);
</SCRIPT>
</HEAD>
<BODY background="/WebSphereSamples/theme/bg.gif">
<CENTER>


    <%
    try{

      String Id = null;
      Id = SubscriptionUtility.getUser(request);
      String message1 = null;
      message1 = (String)request.getAttribute("message1");
      String message2 = null;
      message2 = (String)request.getAttribute("message2");
    %>
  <% if (Id == null || message1 != null || message2 != null)
     { %>
     <table border=0 cellPadding=0 cellSpacing=0 width="620">
    <% if (Id == null)
     { %>
       <tr>
         <font color = "#0000ff">User ID is null.  Contact your
administrator.</font>
       </tr>
  <% } %>
    <tr>
      <% if (message1 != null)
         { %>
           <font color="#0000ff">
      <%=  message1 %>
           </font>
      <% } %>
    </tr>
    <tr>
      <% if (message2 != null)
         { %>
           <font color="#0000ff">
      <%=  message2 %>
           </font>
      <% } %>
    </tr>
  </table>
    <% } %>

  <BR>
  <h2 align=center>Meeting Subscriptions</h2>
  <form name=form1
action="/servlet/itso.wes.ins.triggersample.MeetingSubscriptionServlet"
method=post>
      <table border=0 cellPadding=0 cellSpacing=0 width="620" align="center">
       <tr>
         <td colspan="2">
           <FONT size="-1"><strong>Notify me when there is a meeting in less
than  </strong></FONT>
          <FONT COLOR="#ff0000">*</FONT>
           <FONT size="-1"><input type="text" name="meetingHours" value=""
size="5"> <strong> Hours </strong> </FONT><br><br>
         </td>
       </tr>
       <tr> </tr>
       <tr>
         <td colspan="2"> <FONT size="-1">
          <b>Notification option</b>: <i>once</i> - receive notification only
the first time a match occurs; <i>always</i> -
            receive notification every time a match occurs. <br><br>
            </FONT>
         </td>
       </tr>
       <tr>
         <td colspan="2">
          <FONT size="-1">
           <b>Content Storage Option</b>: <i>save</i> - Save the message
content in storage and provide a link to the content;
            <i>don't save</i> - Send the entire message content in the
notification.<br><br>
            </FONT>
         </td>
       </tr>
      <tr>
        <td> <FONT size="-1">
          <strong>Notification option:  </strong>
          <FONT COLOR="#ff0000">*</FONT>
          <SELECT name="trigopt">
            <OPTION value="once"> once </OPTION>
            <OPTION value="always"> always </OPTION>
            <!-- <OPTION> always and persist </OPTION> -->
          </SELECT>
           </FONT>
        </td>
        <td>
          <FONT size="-1"><strong>
          Content storage option:  </strong></FONT>
          <FONT COLOR="#ff0000">*</FONT>
          <SELECT name="contopt">
          <FONT size="-1">
```

```
            <OPTION value="save"> save </OPTION>
            <OPTION value="don't save"> don't save </OPTION>
          </SELECT>
          </FONT>
       </td>
      </tr>
      <tr>
        <td>
        <FONT size="-1">
        <br><br>Required fields are indicated by </FONT><FONT
color="#ff0000">*</FONT>
        <td>
      </tr>
      <tr>
        <td colspan="2" align="right">
      <input type="submit" name="add" style="border-style: outset" value=" Add
" onClick="return validate()">
        </td>
      </tr>
    </table>

    <% MeetingSubscriptions meetingSubscriptions =
(MeetingSubscriptions)request.getAttribute("meetingSubscriptions"); %>
    <TABLE cellspacing="0" width="620" align="center">
     <TR>
      <TD>
       <TABLE border=2 align="center" cellPadding=5 cellSpacing=0>
        <THEAD>
         <tr>
          <th> <FONT size="-1">Remove </FONT></th>
          <th> <FONT size="-1">Notification within </FONT></th>
          <th> <FONT size="-1">Notification </FONT></th>
          <th> <FONT size="-1">Content Storage </FONT></th>
         </tr>
        </THEAD>
        <TBODY>
     <trcolspan="4">
     <FONT size="-1">
      To remove a trigger, check the checkbox of the line, and press the
refresh button
      </FONT>
     </tr>

         <!-- Description -->
         </TR>
         <TR><BR></TR>
         <% try
             {
                //Get the subscriptions currently defined for this user
                MeetingSubscriptionData ssd = 
meetingSubscriptions.getSubscription(Id ,0);   //Throws an exception if empty
                for (int i = 0; ; )
                {
          %>
          <!-- Unique ID is used for value of the checkbox to identify which 
subscription is being deleted. -->
                  <TR>
                    <TD><FONT size="-1"><INPUT TYPE="checkbox" NAME="delete" 
value="<%= ssd.getId() %>"></FONT></TD>
                    <TD><FONT size="-1"><%= ssd.getHours() %></FONT></TD>
                   <TD><FONT size="-1"><%= ssd.getTriggerOption() 
%></FONT></TD>
                    <TD><FONT size="-1"><%= ssd.getContentOption() 
%></FONT></TD>
                  </TR>
          <%
                  i++;
                  try
                  {
                    ssd =  meetingSubscriptions.getSubscription(Id,i);
                  }
                  catch (java.lang.ArrayIndexOutOfBoundsException e)
                  {break;}
                }
             }
             catch (java.lang.ArrayIndexOutOfBoundsException e) {}
             catch (java.lang.NullPointerException n) {}//This occurs the very 
first time the page is accessed.
          %>
        </TBODY>
       </TABLE>
      </TD>
      <TD>
      <TABLE>
       <!-- Done for spacing to move the buttons further down on the page -->
       <TR>
         <TD><BR><BR><BR><BR></TD>
       </TR>
       <TR>
         <TD>
           <input type="submit" name="refresh" value="Refresh">
         </TD>
       </TR>
       <TR>
         <TD>
           <input type="submit" name="submit" value="Submit">
         </TD>
       </TR>
      </TABLE>
      </td>
     </TR>
     </FONT>
    </TABLE>
    <% if (!SubscriptionUtility.isSecure())
       { %>
         <INPUT name=userid type="hidden" value="<%= Id %>">
    <% } %>
  </form>

  <%
  }
  catch (java.lang.Exception ex){
  %>
  <h2>An Exception Occurred</h2>

  <%=ex%>

  <%
  }
  %>
</body>
</html>
```

Example: A-14   CenterGeneric.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<!-- 5648-C84, 5648-C83, (C) Copyright IBM Corporation, 1997, 2000 -->
<!-- All rights reserved. Licensed Materials Property of IBM -->
<!-- Note to US Government users: Documentation related to restricted rights
-->
<!-- Use, duplication or disclosure is subject to restrictions set forth in GSA
ADP Schedule with IBM Corp. -->
<!--This page may contain other proprietary notices and copyright information,
the terms of which must be observed and followed. -->
<HTML>
<HEAD>
<TITLE>YourCo Employee Center</TITLE>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<META name="GENERATOR" content="IBM WebSphere Page Designer V3.0  for Windows">
```

```
<LINK rel="STYLESHEET" href="/theme/YourCo.css" type="text/css">
</HEAD>
<BODY background="/theme/bg.gif">

<jsp:useBean id="quoteBean" type="WebSphereSamples.Quote.QuoteBean"
class="WebSphereSamples.Quote.QuoteBean" scope="session" />

<jsp:useBean id="stockBean" type="WebSphereSamples.YourCo.Stock.StockBean"
class="WebSphereSamples.YourCo.Stock.StockBean" scope="session" />

<jsp:useBean id="preferredLinksDBBean"
type="WebSphereSamples.YourCo.Employee.PreferredLinksDBBean" scope="request" />

<jsp:useBean id="getQuestionDBBean"
type="WebSphereSamples.YourCo.Poll.GetQuestionDBBean"
class="WebSphereSamples.YourCo.Poll.GetQuestionDBBean" scope="request" />
<jsp:setProperty name="getQuestionDBBean" property="userID" value="WSDEMO" />
<jsp:setProperty name="getQuestionDBBean" property="password" value="wsdemo1"
/>
<jsp:setProperty name="getQuestionDBBean" property="URL"
value="jdbc:db2:SAMPLE" />
<jsp:setProperty name="getQuestionDBBean" property="driver"
value="COM.ibm.db2.jdbc.app.DB2Driver" />
<jsp:setProperty name="getQuestionDBBean" property="pollId" value="1" />
<jsp:setProperty name="getQuestionDBBean" property="dataSourceName"
value="jdbc/sample" />

<CENTER>
<TABLE>
  <TBODY>
    <TR><%
   HttpSession mySession = request.getSession(true);
   WebSphereSamples.YourCo.Login.SessionBean sessionBean = null;

   if ( (sessionBean =
(WebSphereSamples.YourCo.Login.SessionBean)mySession.getValue("sessionBean"))
!= null )
   {
     if ( sessionBean.getStock() == 1 )
     {
%>
      <TD bgcolor="#006699" width="160" align="CENTER"><FONT size="-1"
color="yellow"><B>YCO: $<%=stockBean.getStockQuote()%>
<SUP><%=stockBean.getStockNumerator()%></SUP>/<SUB><%=stockBean.getStockDenomin
ator()%></SUB> + <%=stockBean.getStockChange()%>
<SUP><%=stockBean.getStockChangeNumerator()%></SUP>/<SUB><%=stockBean.getStockD
enominator()%></SUB></B></FONT></TD>
      <%
     }
     if ( sessionBean.getQuote() == 1 )
     {
%>
      <TD bgcolor="#006699" width="460" align="CENTER"><FONT size="-2"
color="#FFFFFF"><%=quoteBean.getQuote()%> </FONT><FONT size="-2"
color="#99CCFF">- <%=quoteBean.getAuthor()%> </FONT></TD>
      <%
     }
   }
%></TR>
  </TBODY>
</TABLE>
<BR>
<TABLE width="620" height="310">
  <TBODY>
    <TR>
      <TD colspan="5" height="20"><IMG src="/theme/circleLine.gif"></TD>
    </TR>
    <TR>
      <TD colspan="2" valign="top" align="left"></TD>
  <TD rowspan="2" width="300" valign="top" align="left"><B>Welcome,

<%=sessionBean.getFirstName()%>,

      to the<BR>
      YourCo Employee Center!</B> <BR>
      <P><FONT size="-1"><B>The Center has everything you need:</B></FONT></P>
      <P><FONT size="-1"><A
href="/WebSphereSamples/YourCo/Search/frameset.html" target="_top">White
Pages</A>: Search&nbsp;for fellow employees</FONT><BR>
      <FONT size="-1"><A
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Meeting.SelectNames">Le
t's Meet:</A> Reserve a meeting room</FONT><BR>
      <FONT size="-1"><A href="/WebSphereSamples/YourCo/Jobs/jobs.jsp">Help
Wanted:</A> Find open job positions</FONT><BR>
      <FONT size="-1"><A
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Timeout.TimeoutServlet"
>Time Out:</A> Manage your leave banks</FONT><BR>
      <FONT size="-1"><A
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Timeout.TotalLeaveServl
et">Total Leave:</A> View Total Leave Situation</FONT><BR>
      <FONT size="-1"><A
href="/servlet/itso.wes.ins.triggersample.MeetingSubscriptionStartServlet">Noti
fiactions:</A> Manage your Subscriptions to Notifications</FONT></P>

      <P><FONT size="-1"><B>Get the most out of the Center:</B></FONT></P>
      <P><FONT size="-1"><A
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Employee.SelectLinks">C
ustomize this page:</A>
```

```
      &nbsp;Are you interested in the stock price?
      Do you want to be inspired by famous quotations?
      &nbsp;The choice is yours. Add your favorite
      links and make this page your own.</FONT></P>

      <P><FONT size="-1"><B>Help us improve the Employee Center:</B> Take
      our <A
href="/WebSphereSamples/YourCo/Survey/survey.html">survey</A>.</FONT></P>
      </TD>
      <TD rowspan="2" width="20"></TD>
      <TD rowspan="2" valign="top">

      <FORM method="post"
action="/WebSphereSamples/servlet/WebSphereSamples.YourCo.Poll.PollServlet">

      <TABLE border="0" cellspacing="4" cellpadding="4" width="200">
        <TBODY>
          <TR>
            <TD valign="top" colspan="2" bgcolor="#006699"><FONT
size="-1"><B><FONT color="#ffffff">
<%
  try
  {
    getQuestionDBBean.execute();
    out.print(getQuestionDBBean.getQUESTION(0));
  }
  catch(Exception e)
  {
    out.println("Error: " + e.getMessage());
  }
%>
            </FONT></B></FONT></TD>
          </TR>
          <TR align="center">
            <TD valign="top" colspan="2"><FONT size="-1"><B>Current results:
</B></FONT></TD>
          </TR>
          <TR>

            <TD align="right">
<INPUT type="hidden" name="pollId" value="1">
<FONT size="-1">
<INPUT type="radio" name="vote" value="yes" CHECKED>
<B>Yes</B>
</FONT>
            </TD>
            <TD align="left"><FONT size="-1">
<INPUT type="radio" name="vote" value="no">
<B>No</B>
```

```
</FONT>
            </TD>
          </TR>
          <TR align="center">
            <TD valign="top" align="center" colspan="2">
<INPUT type="submit" name="submit" value="Submit">
            </TD>

          </TR>
        </TBODY>
      </TABLE>

      </FORM>

      </TD>
    </TR>
    <TR>
      <TD valign="top" colspan="2" width="242" align="left"></TD>
    </TR>
    <TR>
      <TD width="200" colspan="2"></TD>
      <TD width="200"></TD>
      <TD></TD>
      <TD align="center"></TD>
    </TR>
    <TR>
      <TD colspan="5"></TD>
    </TR>
  </TBODY>
</TABLE>
</CENTER>

<%preferredLinksDBBean.closeResultSet();%>
<%getQuestionDBBean.closeResultSet();%>

</BODY>
</HTML>
```

B

LBS sample code

This appendix contains all the source code for YourCo LBSExtensions, discussed in Chapter 11, “Location-Based Services (LBS)” on page 435.

- Servlets and Java classes:
  - itso.wes.lbs.samples.FindExpert.java
  - itso.wes.lbs.samples.ExpertInfo.java
  - itso.wes.lbs.samples.OfficeFinder.java
  - itso.wes.lbs.samples.UserLocation.java
- Java Server Pages and Web pages:
  - /YourCo/Locate/Results.jsp
  - /YourCo/Locate/ResultsNotFound.jsp
  - /YourCo/Locate/ResultsError.jsp
  - /YourCo/Locate/frameset.html
  - /YourCo/Locate/LocateInput.html
  - /YourCo/Locate/blank.html

- Changes made to the original YourCo example:
  - /YourCo/index.html
- Changes made to the database:
  - db2tables.txt
  - ADDRESS.txt
  - EMPLOYEE.txt

Sample source code

Example: B-1 itso.wes.lbs.samples.FindExpert.java

```
package itso.wes.lbs.samples;

import javax.servlet.*;
import java.sql.*;
/**
 * Insert the type's description here.
 * @author: Erik Rongen
 */
public class FindExpert extends javax.servlet.http.HttpServlet {

   //the OfficeFinder classs provides mapping from the user location
   //to the nearest office (in an oversimplified, non-realistic way)
   private OfficeFinder officeFinder;

   // URL for the database
   String dbUrl = "jdbc:db2:sample";

   String results = "/YourCo/Locate/Results.jsp";
   String resultsError = "/YourCo/Locate/ResultsError.jsp";
   String resultsNotFound = "/YourCo/Locate/ResultsNotFound.jsp";
/**
 * Process incoming HTTP GET requests
 *
 * @param request Object that encapsulates the request to the servlet
 * @param response Object that encapsulates the response from the servlet
 */
public void doGet(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws
javax.servlet.ServletException, java.io.IOException {

   performTask(request, response);

}
/**
 * Process incoming HTTP POST requests
 *
 * @param request Object that encapsulates the request to the servlet
 * @param response Object that encapsulates the response from the servlet
 */
public void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws
javax.servlet.ServletException, java.io.IOException {
   performTask(request, response);

}
/**
 * Returns the servlet info string.
 */
public String getServletInfo() {

   return super.getServletInfo();

}
/**
 * Initializes the servlet.
 */
public void init() {
    // insert code to initialize the servlet here

    officeFinder = new OfficeFinder();

    //prepare for DB2 access
    try {
        Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();

    } catch (Exception e) {
        System.out.println(e);
    }

}
/**
 * Process incoming requests for information
 *
 * @param request Object that encapsulates the request to the servlet
 * @param response Object that encapsulates the response from the servlet
 */
public void performTask(
    javax.servlet.http.HttpServletRequest request,
    javax.servlet.http.HttpServletResponse response) {

    //The id of the office that is nearest to the user location
    String officeID = null;

    //The beans with location info for display
    UserLocation userLocation = null;
    ExpertInfo expertInfo = null;

    //The requested experise from the request
    String expertise = null;
    try {

        //Analyse the request
        if (request.getParameter("expertise") == null)
            System.out.println("No expertise attribute found in the request.
returning");
        else {

            //get the requested expertise from the request, and convert to
upppercase,
            //since all JOBS in the database are in UPPERCASE
            expertise = (String)
request.getParameter("expertise").toUpperCase();

            //Create the locationServices object that reads the Location Info
from the headers of the request
            com.ibm.lbs.LocationServices locServices =
                new com.ibm.lbs.LocationServices(request);

           //locServices now contains all the location information of the user
that we need
            //List the information to the console
            printDiagnostics(locServices);

            //we can populate the userLocation bean
            userLocation = new UserLocation();
            userLocation.setAddress(locServices.getStreet());
            userLocation.setCity(locServices.getCity());
            userLocation.setCountry(locServices.getCountry());
            userLocation.setState(locServices.getStateProvince());
            userLocation.setZipCode(locServices.getPostalCode());

            //Now find the id of the office closest to the user
            //The OfficeFinder encapsulates this algorithm.
            //In this example we use an oversimplified algorithm for finding
            //the nearest office
            officeID = officeFinder.find(locServices);

            //now that we have the office id, we can search for
            //the requested expertise in that location in the database

            String url = "jdbc:db2:sample";
            Connection con = DriverManager.getConnection(url, "wsdemo",
"wsdemo1");

            // retrieve data from the database
            System.out.println("Retrieve some data from the database...");
            Statement stmt = con.createStatement();
            String queryString =
                    "SELECT * FROM WSDEMO.EMPLOYEE X , WSDEMO.ADDRESS Y"
                        + " WHERE X.OFFICE = Y.OFFICE"
                        + " AND X.OFFICE = '"
                        + officeID
                        + "'"
                        + " AND X.JOB = '"
                        + expertise
                        + "'";

            ResultSet rs = stmt.executeQuery(queryString);

            System.out.println("Received results:");
            //rs.next is true when rows are found.
            //return the first found employee who meets the criteria
            if (rs.next()) {
                //Info of the employee
                expertInfo = new ExpertInfo();
                expertInfo.setFirstName(rs.getString("FIRSTNME"));
                expertInfo.setLastName(rs.getString("LASTNAME"));
                expertInfo.setMiddleInitial(rs.getString("MIDINIT"));
                expertInfo.setPhoneNumber(rs.getString("PHONENO"));
                expertInfo.setDepartment(rs.getString("WORKDEPT"));
                expertInfo.setEmployeeNumber(rs.getString("EMPNO"));
                expertInfo.setJob(rs.getString("JOB"));

                //info of the location
                expertInfo.setOfficeID(rs.getString("OFFICE"));
                expertInfo.setAddress(rs.getString("ADDRESS"));
                expertInfo.setCity(rs.getString("CITY"));
                expertInfo.setCountry(rs.getString("COUNTRY"));
                expertInfo.setState(rs.getString("STATE"));
                expertInfo.setZipCode(rs.getString("ZIPCODE"));
            } else {
                System.out.println("No expert found in the office nearest to 
the user");
            }
            //we¥re done with DB2
            rs.close();
            stmt.close();

            //Now redirect to the output
            if (userLocation != null && expertInfo != null) {
                //Both beans have been populated. Show result
                response.addHeader("zzz","true");
                request.setAttribute("userLocationBean", userLocation);
                request.setAttribute("expertInfoBean", expertInfo);
                RequestDispatcher rd = 
getServletContext().getRequestDispatcher(results);
```

```
                rd.forward(request, response);
            } else
                if (userLocation != null && expertInfo == null) {
                    //Both beans have been populated. Show result

                    response.addHeader("zzz","false");
                    request.setAttribute("userLocationBean", userLocation);
                    RequestDispatcher rd =

getServletContext().getRequestDispatcher(resultsNotFound);
                    rd.forward(request, response);
                } else {
                   response.addHeader("zzz","error");
                     RequestDispatcher rd =
getServletContext().getRequestDispatcher(resultsError);
                    rd.forward(request, response);
                }

        }

    } catch (Throwable theException) {
        theException.printStackTrace();
    }
}
/**
 *
 * @param locServices com.ibm.lbs.LocationServices
 */
private void printDiagnostics(com.ibm.lbs.LocationServices locServices) {

   System.out.println("Location information for the user:");
    System.out.println("City               :  " + locServices.getCity() +
"\n");
    System.out.println(
        "Cooirdinates        :  " + locServices.getCoordinates() + "\n");
    System.out.println("Country            :  " + locServices.getCountry() +
"\n");
    System.out.println(
        "CountryDistrict     :  " + locServices.getCountyDistrict() + "\n");
    System.out.println(
        "Description         :  " + locServices.getDescription() + "\n");
    System.out.println(
        "Latitude            :  " + locServices.getLatitude() + "\n");
    System.out.println(
        "Longitude           :  " + locServices.getLongitude() + "\n");
    System.out.println("Name               :  " + locServices.getName() +
"\n");
    System.out.println(
        "PostalCode          :  " + locServices.getPostalCode() + "\n");
    System.out.println("SRSName             :  " + locServices.getSRSName() +
"\n");
    System.out.println(
        "StateProvince       :  " + locServices.getStateProvince() + "\n");
    System.out.println("Street              :  " + locServices.getStreet() +
"\n");
    System.out.println(
        "StreetIntersection  :  " + locServices.getStreetIntersection() +
"\n");
    System.out.println(
        "TimeStamp           :  " + locServices.getTimestamp() + "\n");
    System.out.println(
        "Uncertainty         :  " + locServices.getUncertainty() + "\n");

}
}
```

Example: B-2 itso.wes.lbs.samples.ExpertInfo.java

```
package itso.wes.lbs.samples;

/**
 * @author: Erik Rongen
 */
public class ExpertInfo {
   private java.lang.String City;
   private java.lang.String Address;
   private java.lang.String State;
   private java.lang.String Country;
   private java.lang.String officeID;
   private java.lang.String employeeNumber;
   private java.lang.String firstName;
   private java.lang.String lastName;
   private java.lang.String department;
   private java.lang.String middleInitial;
   private java.lang.String phoneNumber;
   private java.lang.String job;
   private java.lang.String zipCode;
/**
 * LocationBean constructor comment.
 */
public ExpertInfo() {
   super();
}
/**
 * @return java.lang.String
 */
public java.lang.String getAddress() {
   return Address;
}
/**
 * @return java.lang.String
 */
public java.lang.String getCity() {
   return City;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getCountry() {
   return Country;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getDepartment() {
   return department;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getEmployeeNumber() {
   return employeeNumber;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getFirstName() {
   return firstName;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getJob() {
   return job;
}
/**
 *
 * @return java.lang.String
 */
```

```
public java.lang.String getLastName() {
   return lastName;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getMiddleInitial() {
   return middleInitial;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getOfficeID() {
   return officeID;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getPhoneNumber() {
   return phoneNumber;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getState() {
   return State;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getZipCode() {
   return zipCode;
}
/**
 *
 * @param newAddress java.lang.String
 */
public void setAddress(java.lang.String newAddress) {
   Address = newAddress;
}
/**
 *
 * @param newCity java.lang.String
 */
```

```
public void setCity(java.lang.String newCity) {
   City = newCity;
}
/**
 *
 * @param newCountry java.lang.String
 */
public void setCountry(java.lang.String newCountry) {
   Country = newCountry;
}
/**
 *
 * @param newDepartment java.lang.String
 */
public void setDepartment(java.lang.String newDepartment) {
   department = newDepartment;
}
/**
 *
 * @param newEmployeeNumber java.lang.String
 */
public void setEmployeeNumber(java.lang.String newEmployeeNumber) {
   employeeNumber = newEmployeeNumber;
}
/**
 *
 * @param newFirstName java.lang.String
 */
public void setFirstName(java.lang.String newFirstName) {
   firstName = newFirstName;
}
/**
 *
 * @param newJob java.lang.String
 */
public void setJob(java.lang.String newJob) {
   job = newJob;
}
/**
 *
 * @param newLastName java.lang.String
 */
public void setLastName(java.lang.String newLastName) {
   lastName = newLastName;
}
/**
 *
 * @param newMiddleInitial java.lang.String
 */
```

```
public void setMiddleInitial(java.lang.String newMiddleInitial) {
   middleInitial = newMiddleInitial;
}
/**
 *
 * @param newOfficeID java.lang.String
 */
public void setOfficeID(java.lang.String newOfficeID) {
   officeID = newOfficeID;
}
/**
 *
 * @param newPhoneNumber java.lang.String
 */
public void setPhoneNumber(java.lang.String newPhoneNumber) {
   phoneNumber = newPhoneNumber;
}
/**
 *
 * @param newState java.lang.String
 */
public void setState(java.lang.String newState) {
   State = newState;
}
/**
 *
 * @param newZipCode java.lang.String
 */
public void setZipCode(java.lang.String newZipCode) {
   zipCode = newZipCode;
}
}
```

Example: B-3 itso.wes.lbs.samples.OfficeFinder.java

```
package itso.wes.lbs.samples;

/**
 * The officeFinder will locate the office nearest to the user's location
 * This was done in a very simple way. A hashtable closestOfficeList
 * contains the link between the user location and the nearest office. Note
 * that in real life things are much more complicated.
 * @author: Erik Rongen
 */
public class OfficeFinder {
   private java.util.Hashtable closestOfficeList;
/**
 * OfficeFinder constructor comment.
 */
public OfficeFinder() {
   super();
   closestOfficeList = new java.util.Hashtable();
   closestOfficeList.put("Cary","00005");
   closestOfficeList.put("San Diego","00001");
   closestOfficeList.put("Boulder","00002");
   closestOfficeList.put("Delft","00004");
   closestOfficeList.put("Raleigh","00003");

}
/**
 *
 * @return java.lang.String
 * @param locationService com.ibm.lbs.LocationServices
 */
public String find(com.ibm.lbs.LocationServices locationService) {
   String userCity = locationService.getCity();
   String closestOfficeID =  (String) closestOfficeList.get(userCity);
   return closestOfficeID;
}
}
```

Example: B-4 itso.wes.lbs.samples.UserLocation.java

```
package itso.wes.lbs.samples;

/**
 * Insert the type's description here.
 * @author: Erik Rongen
 */
public class UserLocation {
   private java.lang.String City;
   private java.lang.String Address;
   private java.lang.String State;
   private java.lang.String Country;
   private java.lang.String zipCode;
/**
 * LocationBean constructor comment.
 */
public UserLocation() {
   super();
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getAddress() {
   return Address;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getCity() {
   return City;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getCountry() {
   return Country;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getState() {
   return State;
}
/**
 *
 * @return java.lang.String
 */
public java.lang.String getZipCode() {
   return zipCode;
}
/**
 *
 * @param newAddress java.lang.String
 */
public void setAddress(java.lang.String newAddress) {
   Address = newAddress;
}
/**
 *
 * @param newCity java.lang.String
 */
public void setCity(java.lang.String newCity) {
   City = newCity;
}
/**
 *
 * @param newCountry java.lang.String
 */
public void setCountry(java.lang.String newCountry) {
   Country = newCountry;
}
/**
 *
 * @param newState java.lang.String
 */
public void setState(java.lang.String newState) {
   State = newState;
}
/**
 *
 * @param newZipCode java.lang.String
 */
public void setZipCode(java.lang.String newZipCode) {
   zipCode = newZipCode;
}
}
```

Example: B-5 /YourCo/Locate/Results.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<!-- This file was generated by IBM WebSphere Studio 3.5 using
g:\WebSphere\Studio\BIN\GenerationStyleSheets\V3.5\JSP1.0\ServletModel\HTMLPage
s.xsl -->
<HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Page Designer V3.5 for Windows">
<LINK REL=STYLESHEET HREF="/WebSphereSamples/theme/YourCo.css" TYPE="text/css">
<title>Expert Locator</title>

<SCRIPT LANGUAGE="JavaScript1.2">
  positionArray = new Array(8);
</SCRIPT>
</HEAD>
<BODY background="/WebSphereSamples/theme/bg.gif">
<CENTER>
<TABLE width="620">
```

```
  <TBODY>
    <TR>
      <TD colspan="10"><IMG src="/WebSphereSamples/theme/topBanner.gif"
border="0" alt="YourCo banner" align="bottom" width="607" height="46"></TD>
      </TR>
    <TR>
      <TD colspan="10">
      <TABLE>
        <TBODY>
          <TR>
            <TD valign="middle" align="left" width="25"><IMG
src="/WebSphereSamples/theme/button.gif" width="20" height="20"
border="0"></TD>
            <TD valign="middle" align="left" width="50"><FONT size="-1"><A
href="/WebSphereSamples/YourCo/index.html" target="_top">Home</A></FONT></TD>
            <TD width="5"></TD>
            <TD><IMG src="/WebSphereSamples/theme/button.gif" width="20"
height="20" border="0"></TD>
            <TD valign="middle" align="left" width="100"><A
href="/WebSphereSamples/YourCo/Search/frameset.html" target="_top"><FONT
size="-1">White Pages</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/WebSphereSamples/theme/button.gif" width="20"
height="20" border="0"></TD>
            <TD valign="middle" align="left" width="100"><A
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.ExpHTMLServlet.Expiring
HTMLServlet" target="_top"><FONT size="-1">YourCo News</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/WebSphereSamples/theme/button.gif" width="20"
height="20" border="0"></TD>
            <TD valign="middle" align="left" width="120"><A
href="/WebSphereSamples/YourCo/main.html" target="_top"><FONT
size="-1">Employee Center</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/WebSphereSamples/theme/buttonDWN.gif" width="20"
height="20" border="0"></TD>
            <TD valign="middle" align="left" width="100"><FONT
size="-1"><B>Locate an Expert</B></FONT></TD>
          </TR>
        </TBODY>
      </TABLE>
      </TD>
    </TR>
    </TBODY>
</TABLE>

<h2>Locate an expert:</h2>
<p>
Select the type of expertise you require:
```

```
<p>
      <form name="locate" method="post"
action="/WebSphereSamples/servlet/itso.wes.lbs.samples.FindExpert"
target="results">
        <select name="expertise">

      <OPTION value="ANALYST">ANALYST</OPTION>

      <OPTION value="CLERK">CLERK</OPTION>

      <OPTION value="DESIGNER">DESIGNER</OPTION>

      <OPTION value="FIELDREP">FIELDREP</OPTION>

      <OPTION value="MANAGER">MANAGER</OPTION>

      <OPTION value="OPERATOR">OPERATOR</OPTION>

      <OPTION value="PRESS">PRESS</OPTION>

      <OPTION value="SALESREP">SALESREP</OPTION>


   </select>
   <INPUT TYPE="submit" NAME="Submit" ID="Submit" VALUE="Submit">
</form>
</body>
</html>
```

Example: B-6 /YourCo/Locate/ResultsNotFound.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">

<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-1">
    <META name="GENERATOR" content="IBM WebSphere Page Designer V3.5 for
Windows">
    <LINK REL=STYLESHEET HREF="/WebSphereSamples/theme/YourCo.css"
TYPE="text/css">
  </HEAD>

  <BODY background="/WebSphereSamples/theme/bg.gif">
```

```
    <jsp:useBean id="expertInfoBean" class="itso.wes.lbs.samples.ExpertInfo"
scope="request"/>
    <jsp:useBean id="userLocationBean"
class="itso.wes.lbs.samples.UserLocation" scope="request"/>


    <CENTER>
       <h4>Your Location is:</h4>


          <%

            try {
              java.lang.String _e0_1 = expertInfoBean.getLastName(); //throws
an exception if empty
              java.lang.String _e0_2 = expertInfoBean.getFirstName(); //throws
an exception if empty
              java.lang.String _e0_3 = expertInfoBean.getPhoneNumber();
//throws an exception if empty
              java.lang.String _e0_4 = expertInfoBean.getJob(); //throws an
exception if empty
             java.lang.String _e0_5 = expertInfoBean.getDepartment(); //throws
an exception if empty

             java.lang.String _e0_6 = expertInfoBean.getAddress(); //throws an
exception if empty
              java.lang.String _e0_7 = expertInfoBean.getCity(); //throws an
exception if empty
              java.lang.String _e0_8 = expertInfoBean.getState(); //throws an
exception if empty
             java.lang.String _e0_9 = expertInfoBean.getZipCode(); //throws an
exception if empty
              java.lang.String _e0_10 = expertInfoBean.getCountry(); //throws
an exception if empty

              java.lang.String _u0_6 = userLocationBean.getAddress(); //throws
an exception if empty
              java.lang.String _u0_7 = userLocationBean.getCity(); //throws an
exception if empty
             java.lang.String _u0_8 = userLocationBean.getState(); //throws an
exception if empty
              java.lang.String _u0_9 = userLocationBean.getZipCode(); //throws
an exception if empty
             java.lang.String _u0_10 = userLocationBean.getCountry(); //throws
an exception if empty

          %>

       <table width="50%" border="0" cellspacing="2" cellpadding="2">
```

```
         <tr align="left" bgcolor="#99CCFF">
           <td><b>Address</b></td><td><%= _u0_6 + ", " + _u0_7 + ", " + _u0_8 +
", " + _u0_9 + "."  %></td>
         </tr>
         <tr align="left" bgcolor="#99CCFF">
           <td><b>Country</b></td><td><%= _u0_10 %></td>
         </tr>
       </table>
       <br>
       <h4>The requested expert is</h4>
       <table width="50%" border="0" cellspacing="2" cellpadding="2">
         <tr align="left" bgcolor="#99CCFF">
           <td><b>Name</b></td><td><%= _e0_1 %> <%=_e0_2 %></td>
         </tr>
         <tr align="left" bgcolor="#99CCFF">
      <td><b>Phone Number</b></td><td><%= _e0_3 %></td>
         </tr>
         <tr align="left" bgcolor="#99CCFF">
      <td><b>Job</b></td><td><%= _e0_4 %></td>
         </tr>
         <tr align="left" bgcolor="#99CCFF">
      <td><b>Department</b></td><td><%= _e0_5 %></td>
         </tr>
         <tr align="left" bgcolor="#99CCFF">
       <td><b>Address</b></td><td><%= _e0_6 + ", " + _e0_7 + ", " + _e0_8 + ",
" +  _e0_9 + "." %></td>
    </tr>
    <tr align="left" bgcolor="#99CCFF">
       <td><b>Country</b></td><td><%= _e0_10 %></td></td>
    </tr>
      </TABLE>

 <%

  }
  catch (java.lang.Exception _e0)
  {
  %>
  An Error occurred
  <%
  _e0.printStackTrace();
  }
 %>


    </CENTER>
  </BODY>
</HTML>
```

Example: B-7 /YourCo/Locate/ResultsError.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">

<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-1">
    <META name="GENERATOR" content="IBM WebSphere Page Designer V3.5 for 
Windows">
    <LINK REL=STYLESHEET HREF="/WebSphereSamples/theme/YourCo.css" 
TYPE="text/css">
  </HEAD>

  <BODY background="/WebSphereSamples/theme/bg.gif">

    <CENTER>

       <h4>An error Occurred. Please try again.</h4>

    </CENTER>
  </BODY>
</HTML>
```

Example: B-8 /YourCo/Locate/frameset.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">

<HTML>
<!-- This file was generated by IBM WebSphere Studio 3.5 using 
g:\WebSphere\Studio\BIN\GenerationStyleSheets\V3.5\JSP1.0\ServletModel\HTMLPage
s.xsl -->
<HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Page Designer V3.5 for Windows">
<TITLE>YourCo Employee Center</TITLE>
<LINK href="/WebSphereSamples/theme/YourCo.css" rel="stylesheet" 
type="text/css">

</HEAD>

<BODY background="/WebSphereSamples/theme/bg.gif">
```

```
</body>
</html>
```

Example: B-9 /YourCo/Locate/LocateInput.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<!-- This file was generated by IBM WebSphere Studio 3.5 using
g:\WebSphere\Studio\BIN\GenerationStyleSheets\V3.5\JSP1.0\ServletModel\HTMLPage
s.xsl -->
<HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Page Designer V3.5 for Windows">
<LINK REL=STYLESHEET HREF="/WebSphereSamples/theme/YourCo.css" TYPE="text/css">
<title>Expert Locator</title>

<SCRIPT LANGUAGE="JavaScript1.2">
  positionArray = new Array(8);
</SCRIPT>
</HEAD>
<BODY background="/WebSphereSamples/theme/bg.gif">
<CENTER>
<TABLE width="620">
  <TBODY>
    <TR>
      <TD colspan="10"><IMG src="/WebSphereSamples/theme/topBanner.gif"
border="0" alt="YourCo banner" align="bottom" width="607" height="46"></TD>
      </TR>
    <TR>
      <TD colspan="10">
      <TABLE>
        <TBODY>
          <TR>
            <TD valign="middle" align="left" width="25"><IMG
src="/WebSphereSamples/theme/button.gif" width="20" height="20"
border="0"></TD>
            <TD valign="middle" align="left" width="50"><FONT size="-1"><A
href="/WebSphereSamples/YourCo/index.html" target="_top">Home</A></FONT></TD>
            <TD width="5"></TD>
            <TD><IMG src="/WebSphereSamples/theme/button.gif" width="20"
height="20" border="0"></TD>
            <TD valign="middle" align="left" width="100"><A
href="/WebSphereSamples/YourCo/Search/frameset.html" target="_top"><FONT
size="-1">White Pages</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/WebSphereSamples/theme/button.gif" width="20"
height="20" border="0"></TD>
            <TD valign="middle" align="left" width="100"><A
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.ExpHTMLServlet.Expiring
HTMLServlet" target="_top"><FONT size="-1">YourCo News</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/WebSphereSamples/theme/button.gif" width="20"
height="20" border="0"></TD>
            <TD valign="middle" align="left" width="120"><A
href="/WebSphereSamples/YourCo/main.html" target="_top"><FONT
size="-1">Employee Center</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/WebSphereSamples/theme/buttonDWN.gif" width="20"
height="20" border="0"></TD>
            <TD valign="middle" align="left" width="100"><FONT
size="-1"><B>Locate an Expert</B></FONT></TD>
          </TR>
        </TBODY>
      </TABLE>
      </TD>
    </TR>
    </TBODY>
</TABLE>

<h2>Locate an expert:</h2>
<p>
Select the type of expertise you require:
<p>
      <form name="locate" method="post"
action="/WebSphereSamples/servlet/itso.wes.lbs.samples.FindExpert"
target="results">
        <select name="expertise">

      <OPTION value="ANALYST">ANALYST</OPTION>

      <OPTION value="CLERK">CLERK</OPTION>

      <OPTION value="DESIGNER">DESIGNER</OPTION>

      <OPTION value="FIELDREP">FIELDREP</OPTION>

      <OPTION value="MANAGER">MANAGER</OPTION>

      <OPTION value="OPERATOR">OPERATOR</OPTION>

      <OPTION value="PRESS">PRESS</OPTION>

      <OPTION value="SALESREP">SALESREP</OPTION>
```

```
   </select>
   <INPUT TYPE="submit" NAME="Submit" ID="Submit" VALUE="Submit">
</form>
</body>
</html>
```

Example: B-10 /YourCo/Locate/blank.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">

<HTML>
<!-- This file was generated by IBM WebSphere Studio 3.5 using
g:\WebSphere\Studio\BIN\GenerationStyleSheets\V3.5\JSP1.0\ServletModel\HTMLPage
s.xsl -->
<HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Page Designer V3.5 for Windows">
<TITLE>YourCo Employee Center</TITLE>
<LINK href="/WebSphereSamples/theme/YourCo.css" rel="stylesheet"
type="text/css">

</HEAD>

<BODY background="/WebSphereSamples/theme/bg.gif">

</body>
</html>
```

Example: B-11 /YourCo/index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<title>YourCo Main Site</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<META name="GENERATOR" content="IBM WebSphere Page Designer V3.0  for Windows">
<LINK REL=STYLESHEET HREF="/theme/YourCo.css" TYPE="text/css">
</head>

<body background="/theme/bg.gif">
```

```
<CENTER>
<TABLE width="620">
  <TBODY>
    <TR>
      <TD colspan="10"><IMG src="/theme/topBanner.gif" border="0" alt="YourCo
banner" align="bottom" width="607" height="46"></TD>
      </TR>
    <TR>
      <TD colspan="10">
      <TABLE>
        <TBODY>
          <TR>
            <TD valign="middle" align="left" width="25"><IMG
src="/theme/buttonDWN.gif" width="20" height="20" border="0"></TD>
            <TD valign="middle" align="left" width="50"><FONT
size="-1"><B>Home</B></FONT></TD>
            <TD width="5"></TD>
            <TD><IMG src="/theme/button.gif" width="20" height="20"
border="0"></TD>
            <TD valign="middle" align="left" width="100"><A
href="/WebSphereSamples/YourCo/Search/frameset.html" target="_top"><FONT
size="-1">White Pages</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/theme/button.gif" width="20" height="20"
border="0"></TD>
            <TD valign="middle" align="left" width="100"><A
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.ExpHTMLServlet.Expiring
HTMLServlet/TheExpiringHTMLServlet" target="_top"><FONT size="-1">YourCo
News</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/theme/button.gif" width="20" height="20"
border="0"></TD>
            <TD width="120" valign="middle" align="left"><A
href="/WebSphereSamples/YourCo/main.html" target="_top"><FONT
size="-1">Employee Center</FONT></A></TD>
            <TD width="5"></TD>
            <TD><IMG src="/WebSphereSamples/theme/button.gif" width="20"
height="20" border="0"></TD>
            <TD valign="middle" align="left" width="120"><A
href="/WebSphereSamples/YourCo/Locate/frameset.html" target="_top"><FONT
size="-1">Locate an Expert</FONT></A></TD>
          </TR>
        </TBODY>
      </TABLE>
      </TD>
    </TR>
    </TBODY>
</TABLE>
```

```
</CENTER>

<CENTER>
<TABLE width="620" height="310">
  <TBODY>
    <TR>
      <TD colspan="5" height="20"><IMG src="/theme/circleLine.gif">
      </TD>
    </TR>
    <TR>
      <TD colspan="2" valign="top" align="left"></TD>
      <TD rowspan="2" width="257" valign="top" align="left">
       <p><B><FONT size="+1">Welcome!</FONT></B><BR>
         <FONT size="-1">You are at the Home page for the YourCo sample Web
</FONT><FONT size="-1">site.
         </FONT></p>
       <p><FONT size="-1">It is true - YourCo is a fictitious company. But, its
         Intanet Web site has many <i><b>real</b></i> features. Take a look
around.
         There is lots to see.<BR>
         <br>
         <B><a href="/WebSphereSamples/YourCo/Search/frameset.html">White
Pages</a></B><BR>
         Browse or search the YourCo employee directory.<BR>
         <BR>
         <B><a
href="/WebSphereSamples/servlet/WebSphereSamples.YourCo.ExpHTMLServlet.Expiring
HTMLServlet">YourCoNews</a></B><BR>
         Get the latest edition of the company news.<BR>
         <BR>
         <B><a href="/WebSphereSamples/YourCo/main.html">YourCo
         Employee Center</a></B><br>
         YourCo employees have the most fun. Log in as a YourCo employee to
customize
         the Center, reserve a meeting room, investigate other job positions,
         take a poll, answer our survey, or manage your time off.<BR>
         </FONT></p>
       <p>&nbsp;</p>
       </TD>
      <TD rowspan="2" width="20"></TD>
      <TD rowspan="2">
       <table width="200">
         <tbody>
         <tr>
           <td valign="top" align="center"><img src="/theme/photo01.jpg"
border="0" align="top"></td>
         </tr>
         <tr>
```

```
           <td align="center"><i><font size="-2">Looking skyward from the
YourCo
             home office - Anywhere, USA.</font></i></td>
         </tr>
         </tbody>
       </table>
      </TD>
    </TR>
    <TR>
      <TD valign="top" colspan="2" width="242" align="left">
      </TD>
      </TR>
    <TR>
      <TD width="200" colspan="2"></TD>
      <TD width="200"></TD>
      <TD></TD>
      <TD align="center"></TD>
      </TR>
    <TR>
      <TD colspan="5"></TD>
      </TR>
  </TBODY>
</TABLE>
</CENTER>
<CENTER>
<TABLE width="620" height="50">
  <TBODY>
    <TR>
      <TD></TD>
    </TR>
    <TR>
      <TD><IMG src="/theme/horzLine.gif"></TD>
    </TR>
    <TR>
      <TD><FONT size="-2">© Copyright 1999 Your Company <I>All rights
reserved.</I></FONT><FONT size="-1"><BR>
      </FONT>
      <FONT size="-2">Questions? Comments? <A
href="mailto:webmaster@yourcompany.com">Email us.</A></FONT></TD>
    </TR>
  </TBODY>
</TABLE>

  <p>&nbsp;</p>

</CENTER>
</BODY>
</HTML>
```

Example: B-12 db2table.txt

```
EMPLOYEE=create table EMPLOYEE ( EMPNO char(6) not null, FIRSTNME varchar(12)
not null, MIDINIT char(1) not null, LASTNAME varchar(15) not null, WORKDEPT
char(3), PHONENO char(4), HIREDATE char(10), JOB char(8), EDLEVEL decimal(5,0)
not null, SEX char(1), BIRTHDATE char(10), SALARY decimal(9,2), BONUS
decimal(9,2), COMM decimal(9,2), USERID char(20), OFFICE char(5) )

ADDRESS=create table ADDRESS ( OFFICE char(5) not null, ADDRESS varchar(100),
CITY varchar(50), ZIPCODE varchar(20) , STATE char(2), Country varchar(50) )

TRIGGER=create table TRIGGER ( USERID char(20), ROOM_ID varchar(3), THEDAY
varchar(10), THETIME varchar(4), RESERVED_BY varchar(40) )
```

Example: B-13 EMPLOYEE.TXT

```
'000010';'ERIK';'H';'R';'A00';'3978';'1965-01-01';'PRES';18;'F';'1933-08-24';00
52750.00;0001000.00;0004220.00;'erongen@ibm';'userid01@IBM';'00001'
'000020';'MICHAEL';'L';'THOMPSON';'B01';'3476';'1973-10-10';'MANAGER';18;'M';'1
948-02-02';0041250.00;0000800.00;0003300.00;'userid02@IBM';'00002'
'000030';'SALLY';'A';'KWAN';'C01';'4738';'1975-04-05';'MANAGER';20;'F';'1941-05
-11';0038250.00;0000800.00;0003060.00;'userid03@IBM';'00003'
'000050';'JOHN';'B';'GEYER';'E01';'6789';'1949-08-17';'MANAGER';16;'M';'1925-09
-15';0040175.00;0000800.00;0003214.00;'userid04@IBM';'00004'
'000060';'IRVING';'F';'STERN';'D11';'6423';'1973-09-14';'MANAGER';16;'M';'1945-
07-07';0032250.00;0000500.00;0002580.00;'userid05@IBM';'00005'
'000070';'EVA';'D';'PULASKI';'D21';'7831';'1980-09-30';'MANAGER';16;'F';'1953-0
5-26';0036170.00;0000700.00;0002893.00;'userid06@IBM';'00001'
'000090';'EILEEN';'W';'HENDERSON';'E11';'5498';'1970-08-15';'MANAGER';16;'F';'1
941-05-15';0029750.00;0000600.00;0002380.00;'userid07@IBM';'00002'
'000100';'THEODORE';'Q';'SPENSER';'E21';'0972';'1980-06-19';'MANAGER';14;'M';'1
956-12-18';0026150.00;0000500.00;0002092.00;'userid08@IBM';'00003'
'000110';'VINCENZO';'G';'LUCCHESSI';'A00';'3490';'1958-05-16';'SALESREP';19;'M'
;'1929-11-05';0046500.00;0000900.00;0003720.00;'userid09@IBM';'00004'
'000120';'SEAN';'';'O''CONNELL';'A00';'2167';'1963-12-05';'CLERK';14;'M';'1942-
10-18';0029250.00;0000600.00;0002340.00;'insuser@IBM';'00005'
'000130';'DOLORES';'M';'QUINTANA';'C01';'4578';'1971-07-28';'ANALYST';16;'F';'1
925-09-15';0023800.00;0000500.00;0001904.00;'userid11@IBM';'00001'
```

```
'000140';'HEATHER';'A';'NICHOLLS';'C01';'1793';'1976-12-15';'ANALYST';18;'F';'1
946-01-19';0028420.00;0000600.00;0002274.00;'userid12@IBM';'00002'
'000150';'ERIK';'';'RONGEN';'D11';'4510';'1972-02-12';'DESIGNER';16;'M';'1947-0
5-17';0025280.00;0000500.00;0002022.00;'erongen@IBM';'00003'
'000160';'ELIZABETH';'R';'PIANKA';'D11';'3782';'1977-10-11';'DESIGNER';17;'F';'
1955-04-12';0022250.00;0000400.00;0001780.00;'userid14@IBM';'00004'
'000170';'MASATOSHI';'J';'YOSHIMURA';'D11';'2890';'1978-09-15';'DESIGNER';16;'M
';'1951-01-05';0024680.00;0000500.00;0001974.00;'userid15@IBM';'00005'
'000180';'MARILYN';'S';'SCOUTTEN';'D11';'1682';'1973-07-07';'DESIGNER';17;'F';'
1949-02-21';0021340.00;0000500.00;0001707.00;'userid16@IBM';'00001'
'000190';'JAMES';'H';'WALKER';'D11';'2986';'1974-07-26';'DESIGNER';16;'M';'1952
-06-25';0020450.00;0000400.00;0001636.00;'userid17@IBM';'00002'
'000200';'DAVID';'';'BROWN';'D11';'4501';'1966-03-03';'DESIGNER';16;'M';'1941-0
5-29';0027740.00;0000600.00;0002217.00;'userid18@IBM';'00003'
'000210';'WILLIAM';'T';'JONES';'D11';'0942';'1979-04-11';'DESIGNER';17;'M';'195
3-02-23';0018270.00;0000400.00;0001462.00;'userid19@IBM';'00004'
'000220';'JENNIFER';'K';'LUTZ';'D11';'0672';'1968-08-29';'DESIGNER';18;'F';'194
8-03-19';0029840.00;0000600.00;0002387.00;'userid20@IBM';'00005'
'000230';'JAMES';'J';'JEFFERSON';'D21';'2094';'1966-11-21';'CLERK';14;'M';'1935
-05-30';0022180.00;0000400.00;0001774.00;'userid21@IBM';'00001'
'000240';'SALVATORE';'M';'MARINO';'D21';'3780';'1979-12-05';'CLERK';17;'M';'195
4-03-31';0028760.00;0000600.00;0002301.00;'userid22@IBM';'00002'
'000250';'DANIEL';'S';'SMITH';'D21';'0961';'1969-10-30';'CLERK';15;'M';'1939-11
-12';0019180.00;0000400.00;0001534.00;'userid23@IBM';'00003'
'000260';'SYBIL';'P';'JOHNSON';'D21';'8953';'1975-09-11';'CLERK';16;'F';'1936-1
0-05';0017250.00;0000300.00;0001380.00;'userid24@IBM';'00004'
'000270';'MARIA';'L';'PEREZ';'D21';'9001';'1980-09-30';'CLERK';15;'F';'1953-05-
26';0027380.00;0000500.00;0002190.00;'userid25@IBM';'00005'
'000280';'ETHEL';'R';'SCHNEIDER';'E11';'8997';'1967-03-24';'OPERATOR';17;'F';'1
936-03-28';0026250.00;0000500.00;0002100.00;'userid26@IBM';'00001'
'000290';'JOHN';'R';'PARKER';'E11';'4502';'1980-05-30';'OPERATOR';12;'M';'1946-
07-09';0015340.00;0000300.00;0001227.00;'userid27@IBM';'00002'
'000300';'PHILIP';'X';'SMITH';'E11';'2095';'1972-06-19';'OPERATOR';14;'M';'1936
-10-27';0017750.00;0000400.00;0001420.00;'userid28@IBM';'00003'
'000310';'MAUDE';'F';'SETRIGHT';'E11';'3332';'1964-09-12';'OPERATOR';12;'F';'19
31-04-21';0015900.00;0000300.00;0001272.00;'userid29@IBM';'00004'
'000320';'RAMLAL';'V';'MEHTA';'E21';'9990';'1965-07-07';'FIELDREP';16;'M';'1932
-08-11';0019950.00;0000400.00;0001596.00;'userid30@IBM';'00005'
'000330';'WING';'';'LEE';'E21';'2103';'1976-02-23';'FIELDREP';14;'M';'1941-07-1
8';0025370.00;0000500.00;0002030.00;'userid31@IBM';'00001'
'000340';'JASON';'R';'GOUNOT';'E21';'5698';'1947-05-05';'FIELDREP';16;'M';'1926
-05-17';0023840.00;0000500.00;0001907.00;'userid32@IBM';'00002'
```

Example: B-14   ADDRESS.TXT

```
'00001';'JustAny Street 432B';'Los Angeles';'1234567';'CA';'United States of 
America'
'00002';'Another Street 1';'Boulder';'1234567';'CO';'United States of America'
'00003';'Building 662 700 Park Office Drive';'Research Triangle 
Park';'27709';'NC';'United States of America'
'00004';'Watsonweg 2';'Uithoorn';'1423 ND';'NH';'The Netherlands'
'00005';'YetAnother Street 54';'Cary';'1234567';'NC';'United States of America'
```

C

Everyplace Wireless Gateway in WebSphere Everyplace Server: installation tips

This appendix provides installation steps and tips to successfully set up the Everyplace Wireless Gateway Version 2.1.1 on AIX in a WebSphere Everyplace Server environment.

Note : The Wireless Gateway should be installed on a new AIX system.

Note : The difference between EWG 2.1.0 and EWG 2.1.1 is the multilingual capability and addition of third party Directory Services Server (DSS) support for existing directory definitions.

EWG 2.1.1 prerequisites

1. Base operating system:
   - AIX : AIX 4.3.3 plus maintenance level 8 and APARs is required (level 4.3.3.51)

   You can obtain this from :

   `http://techsupport.services.ibm.com/rs6k/fixdb.html`

   - Solaris :SunOS 5.7 or later

   Patch-ID 108968-05 is required to read the CD-ROM on SunOS 5.8.
2. Disk space:
   - The ODBC server needs at least 50 MB of table space to install EWG. EWG writes records and requires DB administration to remove them. The frequency of DB administration tasks is determined by the amount of disk space available to the DB server.
   - When enabled, EWG logging can easily use hundreds of megabytes. It is recommended that the log and trace file directories reside in their own file system.
3. Database:
   - DB2 7.1 plus FixPack2a is required and fully included in the WebSphere Everyplace Server 2.1.1 CDs.
   - Oracle 8.1.5, 8.1.6, 8.1.7.

   Oracle 8.1.5 and 8.1.6 require Merrant DataDirect ODBC v3.6.0.

   Oracle 8.1.7 requires Merrant v3.7.0.
4. LDAP : IBM SecureWay Directory 3.2.1 is required and fully included in the WebSphere Everyplace Server 2.1.1 CDs.

EWG 2.1.1 prerequisites installation

1. Install AIX 4.3.3.
2. Install AIX 4.3.3 maintenance level 8 plus APARs:
   - Run Smitty.
   - Install and update from the latest available software.
   - Run **`Update_all`**: 128 files are installed or updated.
3. Check the disk space.
4. Set the debug level and type the following AIX command:

```
#export IBMEPS_DEBUG_LEVEL=5
```

5. Insert WebSphere Everyplace Server CD1 in the CD drive, then type the following AIX command lines :

```
#cd /
#mkdir cdrom
#mount -rv cdrfs /dev/cd0 /cdrom
#cdrom/install.sh
```

6. You are asked for installing Java 1.3.0 (answer yes by entering `y`) and Java 2 (answer yes).
7. Setup Manager opens.
8. It is important to read the Readme file before continuing this installation. Check the latest information.
9. In the Setup Manager interface, select **An XML file does not exist.**
10. Enter the Install Key provided with the CDs.
11. Install SecureWay Directory on this machine:
    - Check the Directory suffix.
    - The default port number is 389.
    - Enter your user ID, `cn=root` for example, and your password.
12. WebSphere Everyplace Server components to install:

    Do *not* select any of the WebSphere Everyplace Server components. The Setup Manager will install IBM HTTP Server, LDAP and DB2 by itself.

    For a secure installation, it is recommended that you install the WebSphere Everyplace Server components in the second part of the installation.
13. You can use the same user ID or different user IDs to access the WebSphere Everyplace Server components.
    - Enter your IBM HTTP Server user ID and password.
    - Enter your DB2 user ID and password.
14. An Installation Summary window appears, including:
    - Everyplace Server package files
    - HTTP Server
    - DB2 + fixpack
    - LDAP
15. Click **Install**.

16. During this installation, you can check the log files:

```
#cd /usr/IBMEPS/setup/logs
#tail -f everyplace_install.trace
```

17. When the installation is complete:
    - Select **Start LDAP**.
    - View the log file. You should see that all the components have been installed successfully; the last log's line should say that the configuration for Everyplace Server post-configuration has succeeded.
    - Click **Finish**.
18. The IBM SecureWay Directory Management Tool opens:
    - Click **Add Server** (left pane, at the bottom).
    - Enter the LDAP Server name.
    - Select **Simple** for the Authentication type.
    - Click **OK**.
19. For LDAP post-configuration checks, click **Browse tree** in the left pane of the Directory Management Tool window and check, in the right pane, that the following appears:

```
dc=yourmachine
db2
ihs
swd
```

20. For Smitty post-installation checks, you can check that the following software have been successfully installed on your machine:
    - Java 130 1.3.0.7
    - Java_dev2 1.2.2.8
    - DB2 7.1.0.28
    - http_server 1.3.12.3
    - LDAP 3.2.1.0

EWG 2.1.1 installation

1. You must restart WebSphere Everyplace Server to install the Wireless Gateway. Type the following AIX commands:

```
#cd /
#cdrom/install.sh
```

2. In the Setup Manager interface, select **An XML file does exist.**

3. Select **Use a retrieve LDAP** (you will use the LDAP that has just been installed on your machine).
4. Enter the LDAP information: Server name, user ID, password.
5. WebSphere Everyplace Server component to install: select **EWG**.
6. Follow the Setup Manager windows, then unmount CD1 when prompted and mount CD7.

   The following components will be installed:

   – Gateway

   – Gatekeeper

   – Ardis Support

   – DataTAC Support
7. When the installation is complete, you can view the log file.

   Note: You should see that all the components have been installed successfully; the last log line should indicate that the configuration for Everyplace Server post-configuration was also successful.

   Click **Finish**.
8. For post-installation checks, you can run Smitty and verify that the Wireless Gateway has been successfully installed: wg 2.1.1.0 appears in the installed software.
9. The Wireless Gatekeeper allows you to configure the Wireless Gateway. Open the Gatekeeper by entering the following AIX command:

   ```
   #wgcfg
   ```
10. You are prompted with the message `Are all the Wireless Gateways that you will be managing WAP-only gateways ?`

    Answer `No`.
11. Enter your login information.
12. Enter your Wireless Gatekeeper login information.
13. The following message appears: `The LDAP schema needs to be updated.`

    Click **OK**. The login process can take more than five minutes.
14. When the Gatekeeper is ready, click **Add Resource** in the Tasks pane, then double-click **Wireless Gateway**.
15. Enter your login (machine’s login) and password (machine’s password).
16. Add a New Gateway. Follow the panels and enter the necessary infornation.

17. The following message appears : `Would you like to add WAP support to this Wireless Gateway ?`

    Answer `Yes.`

18. An HTTP proxy is necessary to run the Wireless Gateway. You can use a proxy which is already running on one of your machines, or you can install WebSphere Test Environment from the WebSphere Everyplace Server CDs on the EWG's machine.

19. The Gatekeeper allows you to add a WAP device resolver, messaging gateway, MNI, MNC, and so on. All these functions are explored in this redbook.

Nte : For more information about the Wireless Gateway, refer to the *Wireless Gateway Administrator's Guide*.

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 793.

- *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5: Extending Web Applications to the Pervasive World*, SG24-6233
- *IBM WebSphere Everyplace Server: A Guide for Architects and Integrators*, SG24-6189
- *Tivoli SecureWay Policy Director Centrally Managing e-business Security*, SG24-6008
- *IBM WebSphere Edge Server: Working with Web Traffic Express and Network Dispatcher, SG24-6172*
- *WebSphere V3 Performance Tuning Guide*, SG24-5657
- *WebSphere Scalability: WLM and Clustering Using WebSphere Application Server Advanced Edition*, SG24-6153
- *Connecting WebSphere to DB2 UDB Server*, SG24-6219
- *WebSphere Scalability: WLM and Clustering Using WebSphere Application Server Advanced Edition*, SG24-6153

Other resources

These publications are also relevant as further information sources:

- *WebSphere Edge Server for Multiplatforms Administration Guide,* GC09-4567
- *WebSphere Edge Server for Multiplatforms Getting Started Guide Version 1.0*, SC09-4566
- *IBM Load Balancer User’s Guide Version 3.0 for Multiplatforms,* GC31-8496
- *Network Dispatcher Administration Guide,* GC31-8496

Referenced Web sites

These Web sites are also relevant as further information sources:

- The WAP Forum

  `http://www.wapforum.org`
- SignalSoft Corporation

  `http://www.signalsoftcorp.com`
- IBM Research, Gryphon message broker

  `http://www.research.ibm.com/gryphon/Gryphon/gryphon.html`
- IBM announcement for WebSphere Voice Server 1.5

  `http://www-4.ibm.com/software/speech/enterprise/ep_1.html.`
- IBM announcement for WebSphere Studio, Advanced Edition, Version 3.5

  `http://www-4.ibm.com/software/webservers/studio/.`
- IBM announcement for VisualAge for Java, Enterprise Edition

  `http://www-4.ibm.com/software/ad/vajava/.`
- IBM announcement for WebSphere Translation Server

  `http://www-4.ibm.com/software/speech/enterprise/ep_8.html`
- IBM WebSphere Everyplace Server Infocenter

  `http://www-3.ibm.com/pvc/products/wes_provider/infocenter/index.html`
- Visual Age Micro Edition information and download

  `http://www.embedded.oti.com/learn/1_4.html`
- IBM announcement for Database 2 Everyplace

  `http://www-4.ibm.com/software/data/db2/everyplace/library.html`
- IBM WebSphere Portal Server home page

  `http://www-4.ibm.com/software/webservers/portal/`
- DB2 Everyplace software download site

  `http://www6.software.ibm.com/dl/db2/everyplace-p`
- IBM Message Center home page

  `http://www-4.ibm.com/software/speech/enterprise/ep_7.html`
- WebSphere Commerce Suite MarketPlace Edition home page

  `http://www-4.ibm.com/software/webservers/commerce/wcs_me/index.html`
- NTT Do Co Mo home page, for information on i-mode

  `http://www.nttdocomo.com`

- Web Intermediaries at IBM Research

  `http://www.almaden.ibm.com/cs/wbi`
- IBM Distributed Computing home page

  `http://www.ibm.com/software/network/dce/`
- Netegrit, Inc., home of SiteMinder

  `http://www.netegrity.com`
- IBM WebSphere Edge Server library

  `http://www.ibm.com/software/webservers/edgeserver/library.html`
- IBM Developerworks paper: *A highly Available and Scalable LDAP Cluster in an IBM AIX Environment*

  `http://www-1.ibm.com/servers/esdd/articles/ldap/index.html`
- IBM WebSphere Everyplace Server code fixes and Plug-In Support for OEM WAP/Wireless Gateways

  `http://www-3.ibm.com/pvc/products/wes_enable/code_fixes.shtml`
- IBM Directory Management Tool publications

  `http://www-4.ibm.com/software/network/directory/library/publications/31/dmt/dparent.htm`
- Web Traffic Express Programming Guide

  `http://www.ibm.com/software/webservers/edgeserver/library.html`
- VoiceML Forum

  `http://www.voicexml.org`

How to get IBM Redbooks

Search for additional Redbooks or Redpieces, view, download, or order hardcopy from the Redbooks Web site:

ibm.com/redbooks

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become Redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the **CD-ROMs** button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Special notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Glossary

ACL. Access control list. (1) In computer security, a collection of all access rights for one object. (2) In computer security, a list associated with an object that identifies all the subjects that can access the object and their access rights; for example, a list associated with a file that identifies users who can access the file and identifies their access rights to that file.

Applet. A small application program, typically written in Java, which can be embedded within a Web page or downloaded by a user along with a Web page.

Authentication. The process of verifying a user's identity to determine what type of access, if any, one may be granted to information or other online resources and transaction capabilities. Users are most frequently authenticated by a user name and password, although more sophisticated methods such as a digital certificate can also be used.

Browser. A program used to view, download, or otherwise access content on the World Wide Web or a corporate intranet. Browsers display coded pages (such as HTML or ASP) that reside on servers and are "rendered" as a Web page. Netscape Navigator and Microsoft Internet Explorer are the two most popular browsers.

Cache server. Some networks use a cache server to store Web pages and other data, so that if the same pages are requested frequently, they can be served from the cache rather than repeatedly retrieved from external Web servers. The external cache is an HTTP proxy such as IBM Web Traffic Express. IBM WebSphere Transcoding Publisher can use it to store and retrieve transcoded Web pages and intermediate results to avoid repeating the transcoding of frequently accessed pages, delivering better performance.

Cache. A cache stores cachable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server while it is acting as a tunnel.

CDMA. (Code Division Multiple Access) A method for transmitting simultaneous signals over a shared portion of the spectrum. The foremost application of CDMA is the digital cellular phone technology from QUALCOMM that operates in the 800MHz band and 1.9GHz PCS band. CDMA phones are noted for their excellent call quality and long battery life. CDMA is less costly to implement, requiring fewer cell sites than the GSM and TDMA digital cell phone systems and providing three to five times the calling capacity. It provides more than 10 times the capacity of the analog cell phone system (AMPS). CDMA has become widely used in North America. Unlike GSM and TDMA, which divides the spectrum into different time slots, CDMA uses a spread spectrum technique to assign a code to each conversation. After the speech code converts voice to digital, CDMA spreads the voice stream over the full 1.25MHz bandwidth of the CDMA channel, coding each stream separately so it can be decoded at the receiving end. For more information, contact the CDMA Development Group (CDG) at www.cdg.org.

CDMA. Code Division Multiple Access. A second generation digital cellular network standard.

CDPD. (Cellular Digital Packet Data) A digital wireless transmission system that is deployed as an enhancement to the existing analog cellular network. Based on IBM's CelluPlan II, it provides a packet overlay onto the analog (AMPS) network and moves data at 19.2 Kbps over ever-changing unused intervals in the voice channels. If all the channels are used, the data is stored and forwarded when a channel becomes available. CDPD is used for applications such as public safety, point of sale, mobile positioning and other business services. CDPD networks cover most of the major urban areas in the U.S.

CDPD. Cellular Digital Packet Data. Designed to work as an overlay on analog cellular networks.

Cell phone. CELLular telePHONE is the first ubiquitous wireless telephone. Originally analog, all new cellular systems are digital. This has enabled the cell phone to turn into a smart phone that has access to the Internet.

CGI. Common Gateway Interface. A standard way of communicating between different processes.

cHTML. Compact HTML is a more efficient variation of HTML specifically designed for use by the i-mode wireless service.

Clustering. Clustering is a technique used to provide scalability through the use of multiple copies of an application on the same machine or on separate machines. Careful management of the different applications is necessary to ensure that they work together effectively. WebSphere has limited clustering support in Version 2.x and more support in Version 3.0.

Common Gateway Interface (CGI). A standard way to run programs on a server from a Web page; enables the server to pass a user's request to an application program and to receive data back to forward on to the user.

cookie. Netscape's term for a small amount of data permanently or temporarily stored by the Web browser (the user) and associated with a particular Web page or Web site. Cookies serve to give the Web browser a memory, so that it can use data that was input on one page in another page, or so it can recall user preferences or other variables when the user leaves a page and returns. (2) Cookies were implemented as an extension to the HTTP protocol. Cookies are transmitted to and from the server and allow a Web page or Web site to "remember" things about the client -- for example, that the user has previously visited the site, has already registered and obtained a password or has expressed a preference about the color and layout of Web pages.

daemon. A program, typically in UNIX, that executes in the background It functions like an extension to the operating system. I typically is an unattended process that is initiated at startup. Typical daemons are print spoolers and e-mail handlers or a scheduler that starts up another process at a designated time. The term comes from Greek mythology meaning "guardian spirit.

DSB. Directory Services Broker is a proxy process that allows NetSEAT clients to make Cell Directory Service requests. This is necessary for the Management Console on Windows NT to operate correctly

DTD. A document type definition that is the schema that defines the XML tags.

DTMF. (Dual-Tone MultiFrequency) The type of audio signals that are generated when you press the buttons on a touch-tone telephone.

Enterprise Java Beans. Despite the name, Enterprise Java Beans (EJBs) are not Java Beans. Enterprise Java Beans are server-side Java components that are designed for distributed environments. They do not exist in isolation but rather are deployed in containers that provide services such as security, naming and directory services, and persistent storage. WebSphere Application Server is just such a container. See http://java.sun.com/products/ejb/ for more information.

EPOC. A 32-bit operating system for handheld devices from Symbian Ltd. Used in Psion and other handheld computers, it supports Java applications, e-mail, fax, infrared exchange, data synchronization with PCs and includes a suite of PIM and productivity applications. See http://(www.symbian.com for more information.

eXtensible Markup Language (XML). A markup language, similar to HTML, used to create documents containing structured information (e.g., words, pictures, transaction data, server APIs, etc.). XML is in many ways superior to HTML since it does not employ a limited set of predefined tags to create and display documents, but instead employs user-defined tags specified in either customized stylesheets or by the programs used to create and serve the documents. In short, this means that XML used in Web documents will enhance the user's ability to effectively and efficiently search for information living on the Web. XML has the ability to allow data providers to define new tags as needed to better describe the data domain being represented. For more information see http://www.software.ibm.com/xml

Firewall. A computer installed between the publicly-accessible and private areas of a network, designed to secure the private network and prevent unauthorized users from gaining access. A security procedure that sets up a barrier between an internal LAN (local area network) and the Internet

Gateway. A server which acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource; the requesting client may not be aware that it is communicating with a gateway. Gateways are often used as server-side portals through network firewalls and as protocol translators for access to resources stored on non-HTTP systems.

GPRS. General Packet Radio Service or GPRS is an enhancement to the GSM mobile communications system that supports data packets. GPRS enables continuous flow of IP data packets over the system for such applications as Web browsing and file transfer. GPRS differs from GSM's short messaging service (GSM-SMS) which is limited to messages of 160 bytes in length.

GSM. (Global System for Mobile Communications) A digital cellular phone technology based on TDMA that is the predominant system in Europe, but also used around the world. Operating in the 900MHz and 1.8GHz bands in Europe and the 1.9GHz PCS band in the U.S., GSM defines the entire cellular system, not just the air interface (TDMA, CDMA, etc.). GSM phones use a Subscriber Identity Module (SIM) smart card that contains user account information. Any GSM phone becomes immediately programmed after plugging in the SIM card, thus allowing GSM phones to be easily rented or borrowed. SIM cards can be programmed to display custom menus for personalized services. GSM provides a short messaging service (SMS) that enables text messages up to 160 characters in length to be sent to and from a GSM phone. It also supports data transfer at 9.6 Kbps to packet networks, ISDN and POTS users. GSM is a circuit-switched system that divides each 200 kHz channel into eight 25 kHz time slots.

HDML. Handheld Device Markup Language is a specialized version of HTML designed to enable wireless pagers, cell phones, mobile phones and other handheld devices to obtain information from Web pages. HDML was developed by Phone.com (formerly Unwired Planet) before the WAP specification was standardized. It is a subset of WAP with some features that were not included in WAP. AT&T Wireless launched the first HDML-based service in 1996.

HTML. Hyper Text Markup Language is a document format used on the World Wide Web. Web pages are built with HTML tags, or codes, embedded in the text. HTML defines the page layout, fonts and graphic elements as well as the hypertext links to other documents on the Web. Each link contains the URL, or address, of a Web page residing on the same server or any server worldwide, hence the term “World Wide” Web.

HTTP proxy. An HTTP proxy is a program that acts as an intermediary between a client and a server. It receives requests from clients, and forwards those requests to the intended servers. The responses pass back through it in the same way. Thus, a proxy has functions of both a client and a server. Proxies are commonly used in firewalls, caching and transcoding machines.

HTTP. Hyper-text Transfer Protocol. The communications protocol or "language" used by servers and browsers to transfer Web pages across the Internet or an intranet.

HTTPS. Secure Hypertext Transfer Protocol. A Web protocol which employs Netscape Communication's Secure Socket Layer (SSL) within the regular HTTP communication and encrypts data sent from a user to the Web server - a "secure server" - and decrypts pages returned to the user.

IMAP. Internet Messaging Access Protocol. A standard mail server expected to be widely used on the Internet. It provides a message store that holds incoming e-mail until users log on and download it. IMAP4 is the latest version.

IMAP is more sophisticated than the Post Office Protocol (POP3) mail server. Messages can be archived in folders, mailboxes can be shared, and a user can access multiple mail servers. There is also better integration with MIME, which is used to attach files.

i-Mode. A packet-based information service for mobile phones from NTT DoCoMo (Japan). i-mode provides Web browsing, e-mail, a calendar, chat rooms, games, and customized news. It was the first smart phone system for Web browsing and its popularity grew very quickly after its introduction in 1999. i-mode is a proprietary system that uses a subset of HTML, known as cHTML, in contrast to the global WAP standard that uses a variation of HTML, known as WML. The i-mode transfer rate is 9600 bps, but is expected to increase to 384 kbps in 2001, using W-CDMA.

IrDA. The Infrared Data Association develops standards for wireless, infrared transmission systems between computers. With IrDA ports, a laptop or PDA can exchange data with a desktop computer or use a printer without a cable connection. IrDA requires line-of-sight transmission like a TV remote control. IrDA products began to appear in 1995. See http://www.irda.org for more information.

Java. An object-oriented programming language developed by Sun Microsystems which can be used to create applications that will run independent of the platform (e.g., Windows, UNIX, etc.). Small Java programs (Applets) are frequently used in Web pages to provide active elements such as animations or scrolling headlines or interactive features such as calculators. Sun has created a complete Web site devoted to Java and its various applications.

JavaBeans. JavaBeans are Java components designed to be used on client systems. Java Beans may or may not be visual components. See http://www.javasoft.com/beans/docs for more information.

JavaServer Page (JSP). JSPs provide a simplified, fast way to create dynamic Web content. JSP technology enables rapid development of Web-based applications that are server and platform independent. JavaServer Pages are compiled into servlets before deployment.

LTPA. Light weight third party authentication is a protocol used with WebSphere to for authentication.

MNC. MNC is a mobile network connection. A mobile network connection is a resource that is assigned to a Everyplace Wireless Gateway and defines a specific type of network connection. The MNC consists of a line driver, a network protocol interpreter, and one or more physical ports. You configure one MNC for each network provider that you will use.

Mobile device. A mobile device is a portable, generally small, wireless device that can be used to access the Internet via a browser. It includes a wide range of capability and functionality. Mobile devices include mobile phones, wireless PDAs, and wireless laptops.

Mobile phone. A mobile phone is a wireless smart phone that has a microbrowser to access Internet content. Other names for a mobile phone include cell phone and wireless phone.

MSISDN. Mobil Station ISDN. This is the identification number of the specific telephone that is making a call.

NAS. A server in a network dedicated to authenticating users that log on. It may refer to a dedicated server or to the software service within a server. Typically the RADIUS protocol is used in the authentication process.

ODBC. Open Database Connectivity. A database programming interface from Microsoft that provides a common language for Windows applications to access databases on a network. ODBC consists of the function calls programmers write into their applications and the ODBC drivers themselves.

Openwave browser. Openwave Mobile Browser is a microbrowser produced by Openwave Systems Inc. that brings the full power and accessibility of the Internet to mobile phones, PDAs and other devices that use mobile communication networks for information access.

PDA. (Personal digital assistant) A handheld computer that serves as an organizer for personal information. It generally includes at least a name and address database, to-do list and note taker. PDAs are typically pen based and use a stylus to tap selections on menus and to enter printed characters. The unit may also include a small on-screen keyboard which is tapped with the pen. Data is synchronized between the PDA and desktop computer via cable or wireless transmission.

pervasive computing. The use of a computing infrastructure that supports information appliances from which users can access a broad range of network-based services, including Internet-based e-commerce services. Pervasive computing thus provides users with the ability to access and take action on information conveniently.

POP3. Post Office Protocol 3. A standard mail server commonly used on the Internet that POP3 uses the SMTP messaging protocol. It provides a message store that holds incoming e-mail until users log on and download it. POP3 is a simple system with little selectivity. All pending messages and attachments are downloaded at the same time.

Proxy. A server that receives requests intended for another server and that acts on the client's behalf (as the client's proxy) to obtain the requested service. A proxy server is often used when the client and the server are incompatible for direct connection (for example, when the client is unable to meet the security authentication requirements of the server but should be permitted some services).

Public-Key Infrastructure (PKI). A comprehensive set of functions required to provide public-key encryption and digital signature services, including: key and certificate lifecycle management; certification authority functions; directory for storing and retrieving certificates; certificate revocation system; a key backup and recovery system; and time-stamping services.

PvC. Popular short form within IBM for pervasive computing (see pervasive computing).

reverse proxy. A reverse proxy acts as a proxy on behalf of the server(s) as opposed to acting on behalf of the client.

RTSP. . Real Time Streaming Protocol - a protocol, developed by Netscape and Progressive Networks, for transmitting audio and video over the Internet.

Scalability. Scalability is an abstract attribute of software that refers to its ability to handle increased data throughput without modification. WebSphere handles scalability by allowing execution on a variety of hardware platforms that allow increased performance and clustering.

Servlets. Servlets are Java classes that run on Web servers to provide dynamic HTML content to clients. The servlets take as input the HTTP request from the client and output dynamically generated HTML. For more information, see http://www.software.ibm.com/ebusiness/pm.html#Servlets.

SMS. Short Message Service or SMS is text message service that enables short messages of generally no more than 140-160 characters in length to be sent and transmitted from a cell phone. SMS is supported by GSM and other mobile communications systems. Unlike paging, short messages are stored and forwarded in SMS centers.

SMTP. (Simple Mail Transfer Protocol) The standard e-mail protocol on the Internet, is a TCP/IP protocol that defines the message format and the message transfer agent (MTA), which stores and forwards the electronic mail. SMTP was originally designed for only ASCII text, but MIME and other encoding methods enable program and multimedia files to be attached to e-mail messages. SMTP servers route SMTP messages throughout the Internet to a mail server, such as POP3 or IMAP4, which provides a message store for incoming mail.

SOCKS. A SOCKS server is a proxy server that uses a special protocol, sockets, to forward requests. Transcoding Publisher connects through a SOCKS server that is configured with a firewall to manage network traffic and to protect your network from outside intrusion (it supports Versions 4 and 5 SOCKS servers).

SSL. Secure Sockets Layer. A secure protocol used for authentication and encryption. SSL can be used over HTTP, RMI, Telnet and other protocols.

TAI. TAI is a WebSphere Application Server plug-in that intercepts the incoming data from the authentication tool, instructs WebSphere Application Server not to authenticate again and to trust the user-identity from the authentication tool.

TCP/IP.TCP/IP is a set of protocols developed to allow cooperating computers to share resources across a network.

TDMA. (Time Division Multiple Access) A satellite and cellular phone technology that interleaves multiple digital signals onto a single high-speed channel, by dividing each channel into three subchannels providing service to three users instead of one.

TDMA. Time Division Multiple Access. A second-generation digital cellular network standard.

TLS. Transport Layer Security. The standard (IEFT) security protocol on the Internet. It is expected to eventually supersede SSL.

TomCat. Tomcat is a free, open-source implementation of Java Servlet and JavaServer Pages technologies developed under the Jakarta project at the Apache Software Foundation.

Transcoding. Transcoding is a new technology that gives you the ability to make Web-based information available on handheld and other new type devices economically and efficiently, or on the slow network connections like a dial up modem connection. With transcoding, users receive information (text and images) tailored to the capabilities of the devices they are using and also tailored to the capacity of the network being used.

Transcoding is also the process whereby the MEGs modify the request and generate the original resource and all of the document (or resource) editing (or transcoding).

UNIX. A multi-user, multitasking operating system developed by Bell Laboratories for multi-user environments. UNIX is the most commonly-used operating system for servers on the Internet. IBM's version of UNIX is called AIX. The emergence of a new version of UNIX called Linux is revitalizing UNIX across all platforms.

URI. Universal Resource Indicator is the encoded address for any resource -- HTML document, image, video clip, program, etc. -- on the Web.

URL. Uniform Resource Locator. An "address" used to locate Web pages and other resources on the World Wide Web.

Voice XML. Voice XML is an extension of XML that defines voice segments and enables access to the Internet via telephones and other voice-activated devices. AT&T, Lucent and Motorola created the Voice XML Forum to support this development. For more information, visit http://www.vxml.org.

WAP. Wireless Application Protocol. An open, global, wireless communication specification that is defined and managed by the WAP Forum - a consortium of more than 300 wireless network operators, wireless manufacturers and affiliates. The WAP protocols are network independent.

Web Application Server. A Web application server is a software program designed to manage applications at the second tier of three-tier computing, that is, the business logic components. A Web application server manages applications that use data from back-end systems, such as databases and transaction systems, and provides output to a Web browser on a client. For more information see http://www.software.ibm.com/ebusiness/appsrvsw.html

Web browser. To access the World Wide Web, you must use a Web browser. A browser is a software program that allows users to access and navigate the World Wide Web.

Wireless Gatekeeper. A Java-based administrator's console that enables one or more administrators to work with Wireless Gateways remotely. It provides an easy-to-use interface that enables an administrator to configure Wireless Gateways, define wireless resources, group resources to control access, and assign administrators to perform operations on the resources as needed.

Wireless LAN. A wireless LAN is a local area network that transmits over the air, typically in an unlicensed frequency such as the 2.4 GHz band. A wireless LAN does not require lining up devices for line of sight transmission, as IrDA does. Wireless access points (base stations) are connected to an Ethernet hub or server and transmit a radio frequency over an area of several hundred to a 1000 feet, which can penetrate walls and other non-metal barriers. Roaming users can be handed off from one access point to another like a cellular phone system. Laptops use wireless modems that plug into an existing Ethernet port or that are self contained on PC cards, while stand-alone desktops and servers use plug-in cards (ISA, PCI, etc.).

Wireless network. Used to transmit data between wireless devices such as a mobile phone, PDA, or personal computer without the use of a physical cable or wire.

Wireless service provider. An organization that provides wireless services, including cellular services, satellite services and ISPs.

WLP. Wireless link protocol. A modified version of the Point-to-Point Protocol (PPP) used by the IBM Wireless Gateway to support wireless (non-WAP) client devices.

WML. Wireless Markup Language. XML-based, WML tags are used to mark up content in decks for WAP-enabled devices.

WTLS. Wireless Transport Layer Security. A simplified version of TLS designed specifically for WAP devices. It uses mini-certificates.

WWW. The World Wide Web (known as the Web) is a system of Internet servers that supports hypertext to access several Internet protocols on a single interface.

XML. See Extensible Markup Language

XSL. Extensible Style Language. XSL stylesheets are documents that describe a mapping between XML documents and visual data that can be presented to a client in a browser or mini-browser.

Abbreviations and acronyms

ACL	access control list
AO	Access Offering
APAR	authorized program analysis report
ASP	application service provider
AST	Active Session Table
B2C	business to consumer
B2E	business to employee
CBR	content-based routing
CDMA	code division multiple access
CDPD	cellular digital packet data
CHTML	compact HTML
CSR	Customer Service Representative
DCE	Distributed Computing Environment
DIT	directory information tree
DMS	Device Management Server
DMZ	demilitarized zone
DN	distinguished name
DNS	domain name system
DTD	document type definition
DTMF	dual tone multi-frequency
ECP	Everyplace Cookie Proxy
EJB	enterprise Java bean
EO	Enable Offering
ESCP	Edge Server - Caching Proxy
ESM	Everyplace Synchronization Manager
EWG	EveryPlace Wireless Gateway
FTP	File Transfer Protocol
GIF	graphics interchange format
GSM	Global System for Mobile communication
GSO	Global Sign-on
GUDA	generalized user directory access
HACMP	high availability cluster multiprocessing (AIX)
HDML	handheld device markup language
HTTP	Hypertext Transport Protocol
HTTPS	Secure Hypertext Transport Protocol
IBM	International Business Machines Corporation
IMAP	Internet Message Access Protocol
IMC	IBM Mobile Connect
INS	Intelligent Notifications Services
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISP	Independent service provider
ISV	Independent Software Vendor
ITSO	International Technical Support Organization
JDK	Java Development Kit
JPEG	joint photographic experts group
JSP	Java Server Page
JVM	Java Virtual Machine
LBS	Location-Based Services
LDAP	Lightweight Directory Access Protocol
LOB	lines of business

LTPA	lightweight third party authentication
MNC	mobile network connection
MQe	MQSeries Everyplace
MSISDN	mobile station ISDN
NAS	network access server
NNTP	Network News Transport Protocol
ODBC	open database connectivity
PC	personal computer
PDA	Personal Digital Assistant
PICS	platform for Internet content selection
PIM	personal information manager
PKI	Public Key Infrastructure
POP	Post Office Protocol
PSTN	Public Switched Telephone Network
QBE	query by example
RADIUS	remote authentication dial-in user service
RAID	Redundant Array of Independent Disks
RDBMS	relational database management system
RDN	relative distinguished name
RPC	remote procedure call
RPSS	reverse proxy security servers
RTSP	real-time streaming protocol
SDK	Software Development Kit
SLA	service level agreement
SMS	Short message service
SMTP	Simple Mail Transport Protocol
SNA	Systems Network Architecture
SPO	Service Provider Offering
SSI	Secure Sockets Layer
STEP	Sametime Everyplace
TAI	trusted association interceptor
TCP	Transmission Control Protocol
TDMA	time division multiple access
TLS	Transport Layer Security
TPSM	Tivoli Personalized Services Manager
TSHDM	Tivoli Smart Handheld Device Manager
UDB	Universal Database
UDP	User Datagram Protocol
UND	universal notification dispatcher
URI	universal resource indicator
URL	universal resource locator
USB	universal serial bus
VoIP	Voice over IP
VPN	Virtual Private Network
WAN	wide area network
WAP	Wireless Access Protocol
WAS	WebSphere Application Server
WBMP	wireless bitmap
WES	WebSphere Everyplace Server
WLP	wireless link protocol
WML	Wireless Markup Language
WSP	Wireless Session Protocol
WTE	Web Traffic Express
WTLS	Wireless Transport Layer Security
WTP	WebSphere Transcoding Publisher
XML	Extensible Markup Language
XSL	Extensible Style Language

Index

IBM

Redbooks

IBM WebSphere Everyplace Server Service Provider and Enable Offerings: Enterprise Wireless Applications

(1.5” spine)
1.5”<-> 1.998”
789 <->1051 pages

IBM WebSphere Everyplace Server Service Provider and Enable Offerings: Enterprise Wireless Applications

Adapt your enterprise applications for access from wireless networks

Extend applications to support new wireless technologies

Develop and deploy PDA applications

This redbook helps you to adapt and extend new and existing enterprise applications for access from wireless devices, such as WAP phones and PDAs, using the IBM WebSphere Everyplace Server (WES) Service Provider Offering (SPO) and Enable Offering (EO).

The information provided in this redbook targets Business-to-Employee (B2E) enterprise applications, but most of the scenarios presented apply to Business-to-Consumer (B2C) applications as well. In this redbook, you will find step-by-step examples and scenarios showing ways to rapidly integrate your enterprise applications into a WebSphere Everyplace Server environment, making them also available from wireless devices through the implementation of new and enhanced capabilities included in the current releases of WebSphere Everyplace Server offerings, such as transcoding, annotators for text clipping, stylesheets and the Wireless Gateway.

Once your enterprise applications are available from wireless devices, you can deploy new state of the art technologies such as Push messages, Location-Based Services, Intelligent Notifications Services and Voice applications. You will find many scenarios describing recommended ways to develop applications using the APIs provided by the WebSphere Everyplace Server components. Although WebSphere Everyplace Server offerings do not provide a Voice Server, we have also included guidelines to develop Voice XML applications using transcoding capabilities provided by WebSphere Everyplace Server. This redbook includes scenarios using IBM Mobile Connect and Synchronization Manager with a sample DB2 Everyplace application built by the Mobile Application Builder. We also describe transaction messaging applications using the WebSphere Everyplace Serve MQSeries Everyplace component to provide a once-only assured delivery of messages.

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

SG24-6519-00 ISBN 0738424129