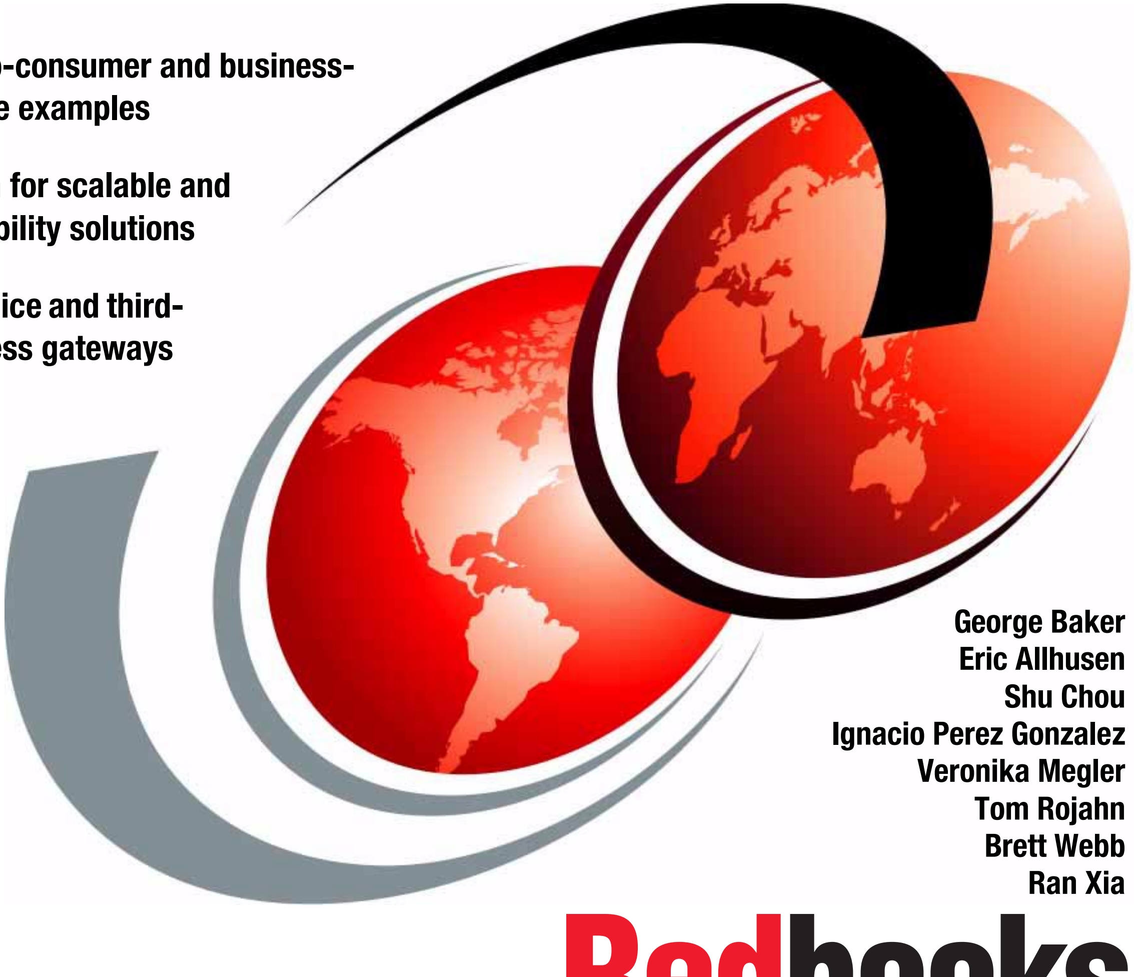
IBM WebSphere Everyplace Server:

A Guide for Architects and Integrators



- Business-to-consumer and business-to-employee examples
- How to plan for scalable and high-availability solutions
- Integrate voice and third-party wireless gateways

George Baker
Eric Allhusen
Shu Chou
Ignacio Perez Gonzalez
Veronika Megler
Tom Rojahn
Brett Webb
Ran Xia

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM WebSphere Everyplace Server: A Guide for Architects and Integrators

December 2001

SG24-6189-00

Take Note! Before using this information and the product it supports, be sure to read the general information in “Special notices” on page 431.

First Edition (December 2001)

This edition applies to IBM WebSphere Everyplace Server, Service Provider Offering for Multiplatforms Version 2.1.1 for use with AIX or Sun Solaris, and IBM WebSphere Everyplace Server Enable Offering Version 1.1 for use with the Windows 2000, Windows NT, AIX or Sun Solaris.

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

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Preface

The purpose of this book is to help information technology (IT) architects and system integrators understand IBM WebSphere Everyplace Server enough to architect a successful mobile e-business solution that is based on IBM WebSphere Everyplace Server. Basically, IBM WebSphere Everyplace Server is the magic box in Figure 0-1, and the purpose of this book is to help you understand what is in the magic box and how to design its implementation.

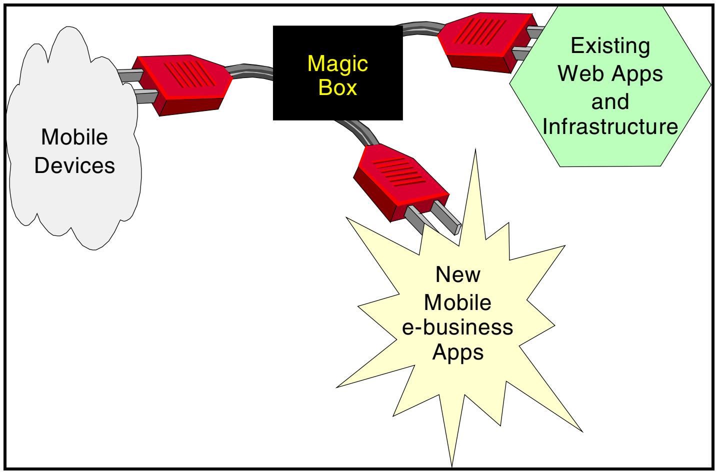

Mobile Devices
Magic Box
Existing Web Apps and Infrastructure
New Mobile e-business Apps

Figure 0-1 IBM WebSphere Everyplace Server - the magic box

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.

George Baker is a Senior I/T Specialist at the International Technical Support Organization, Raleigh Center. He writes and teaches IBM classes worldwide on host integration software. Before joining the ITSO in 2000, George worked for over 30 years in the field as a programmer, technical specialist, sales specialist and manager in the areas of large systems, networking and workstation software systems.

Eric Allhusen is an IT Specialist with IBM Global Services, Wireless e-business Services. His areas of expertise include Java and Internet technologies, intelligent systems, and human-computer interaction. Eric received a Master of Science in Computer Science from the Georgia Institute of Technology. He also holds a Bachelor of Science in Computer Science and a Bachelor of Arts in Mathematics from Augsburg College.

Shu Chou is a certified Consulting IT Architect and has been with IBM for 27 years. She has many consulting engagements in wireless and e-business solutions with major investment banks and market makers. She holds an MBA in Finance from Johns Hopkins University, an M.S. in Computer Science from the University of Michigan, and a B.S. in Electronic Engineering from Chiao-Tung University in Taiwan. She is a member of the MIT enterprise forum.

Ignacio Perez Gonzalez is an IT Specialist with IBM Business Innovation Services in Madrid, Spain. He has four years of experience in Information Technology and Consulting. He holds a degree in Computer Science from Madrid Polythechnic University .His areas of expertise include Java, Internet technologies and protocols, Enterprise JavaBeans, IBM WebSphere, performance and security.

Veronika Megler is a certified Consulting IT Architect with IBM's Solutions Integration Technology Center, Server Division, based in Beaverton, Oregon, USA, specializing in solutions integration for wireless infrastructures. She has 23 years of experience in IT, 15 of them with IBM. Her previous experience ranges from operations, application development, systems programming, pre-sales technical support, systems management disciplines, project management, and IT management consulting. Most recently, she spent five years in IBM Global Services as an architect, implementing e-business architectures. She holds a B.Sc. from Melbourne University in Australia.

Tom Rojahn is an IBM Certified IT Architect with IBM Business Innovation Services in Oslo, Norway. In the 1-1/2 years Tom has been with IBM, he has specialized in wireless e-business, completing three wireless e-business projects, two of them involving WES. Before joining IBM, Tom was an independent consultant for 17 years with his own consultancy firm specializing in UNIX systems. He has been an advisor, consultant and a developer of operational software for UNIX and has facilitated numerous courses in UNIX, C and Perl programming. Before his independent consultancy, Tom had 10 years of experience as a systems programmer on UNIVAC, DEC and IBM mainframes.

Brett Webb is a Consulting Client IT Architect in Dallas, Texas, USA. He has 20 years of experience in IBM, and two years' experience prior to IBM as a mainframe computer operator and independent programmer. He holds a degree in Systems Science from the University of West Florida, and a master degree in business from the University of Texas at Dallas. He started his IBM career in technical writing and editing, and has worked with most IBM systems made in the last 20 years. Most recently he has focused on e-business and mobile e-business technologies as a consultant to system integrators. He has written on such topics as business intelligence, system maintenance and operations, networking, user interfaces, and security.

Ran Xia is an Advisory IT Specialist in Raleigh, NC, he has worked with WebSphere Application Server, HoD, and Host Publisher. He is currently working with the Advanced Technical Support Pervasive Pre-sale Tech support team. He holds a computer science degree from North Carolina State University.

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

Juan Rodriguez
International Technical Support Organization, Raleigh Center

Reed Bittinger
Joan Boone
Alan Booth
Andrew Capella
Dan Conrad
Ivan Heninger
Don Jones
Doug Jones
Brett King
Kwabena Mireku
James Thrasher
Bill Trautman
Bill Wimer
Tony Wrobel
IBM Research Triangle Park, NC

Scott Weller
IBM Rochester, MN

Michele Granger
IBM France

Andy Brown
Henry Orton
Steve Wright
IBM Hursley, UK

Nobuhiro Asai
Mitsugu Kanetake
Takashi Terao
IBM Japan

Special notice

This publication is intended to help architects and integrators to understand WebSphere Everyplace Server so that they can design a successful wireless solution. The information in this publication is not intended as the specification of any programming interfaces that are provided by WebSphere Everyplace Server Service Provider Offering or WebSphere Everyplace Server Enable Offering. See the PUBLICATIONS section of the IBM Programming Announcement for WebSphere Everyplace Server Service Provider Offering and WebSphere Everyplace Server Enable Offering 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®
Database 2™
DB2®
DB2 Universal Database™
DirectTalk®
Everyplace™
IMS™
Mindspan™
MQSeries®
Net.Commerce™
NetVista™
OS/2®
OS/390®
PowerPC®
pSeries™
Redbooks Logo™
Redbooks™
RS/6000®
S/390®
SecureWay®
SP™
Tivoli®
ViaVoice®
VisualAge®
WebSphere®
WorkPad®
zSeries™
Lotus®
Lotus Notes®
Sametime®
Domino™
Lotus QuickPlace ™
Lotus Sametime™
QuickPlace™

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

Overview

WebSphere Everyplace Server is an IBM Pervasive Computing Division product. To understand WebSphere Everyplace Server, it is helpful to understand what IBM means by pervasive computing, and how WebSphere Everyplace Server fits into IBM's pervasive computing vision.

Around 1996 IBM CEO, Lou Gerstner, introduced the concept of pervasive computing when he said:

> ...*Picture a day when a billion people will interact with a million e-businesses via a trillion interconnected devices*…

In response, IBM Pervasive Computing Division's Bill Bodin wrote in his article *The Pervasive Computing Paradigm:*

> *The challenge this presents to the IBM Pervasive Computing Division is the creation of an architecture that provides an end-to-end capability to deliver content and function to and from these intelligent devices. This architecture must implement aspects of reliability, security and information scaling in such a way that the user doesn't think about how the data is delivered; rather, the user is simply enabled to make intelligent and informed decisions based on the* pervasive *availability of the data.*

Pervasive computing, then, is the utilization of all the various intelligent devices and networks to deliver value to the user. It includes vehicle networks, home networks, and mobile/wireless (cell phone and PDA) networks. Mobile/wireless is a subcategory of pervasive computing, and WebSphere Everyplace Server is the implementation of the architecture.

Many of the products from the IBM Pervasive Computing Division carry the brand *Everyplace.* Not all IBM Pervasive Computing Division products carry the Everyplace brand, and not all products with the Everyplace brand are from the Pervasive Computing Division, but all products carrying the Everyplace brand are intended to support pervasive devices (or at least mobile/wireless devices).

1

Introduction

In our discussion of WebSphere Everyplace Server, we first give a general overview of WebSphere Everyplace Server, discussing the benefits, features and functions in the latest version. Later we discuss the various functions of WebSphere Everyplace Server by breaking the capabilities into smaller categories, finally providing an overview of each.

There are three products with the Everyplace brand, each of which is a suite of complementary components:

- Everyplace Access Offering
- WebSphere Everyplace Server Enable Offering
- WebSphere Everyplace Server Service Provider Offering

This chapter describes each of the offerings. Chapter 2, “Differentiating the offerings” on page 45, discusses the differences between the Everyplace offerings. The remainder of this book focuses primarily on WebSphere Everyplace Server Service Provider Offering, making occasional reference to Everyplace Access Offering and WebSphere Everyplace Server Enable Offering.

1.1 IBM WebSphere Everyplace Server overview

The IBM WebSphere Everyplace Server is a comprehensive software platform for extending the reach of new and existing applications into the mobile e-business space. WebSphere Everyplace Server enables a new class of pervasive devices, including WAP phones, PDAs, Internet appliances, and embedded devices to access e-business applications, enterprise data, and Internet content. WebSphere Everyplace Server helps us move closer to the vision of pervasive computing, the ability to access and utilize any information, at anytime, from anywhere, on any device.

WebSphere Everyplace Server has been designed to address the specific requirements of mobile e-business. It provides solutions for connectivity, security, content handling, optimization, and subscriber and device management.

1.1.1 Benefits of WebSphere Everyplace Server

IBM WebSphere Everyplace Server has been designed in response to the following set of fundamental business objectives:

- Preserve existing investment by integrating into existing applications and subsystems easily
- Increase customer loyalty for greater retention and/or less turnover
- Deploy new services quickly and cost effectively
- Support future (unknown) client devices quickly and cost effectively
- Adjust end-user experience appropriate to multiple client devices
- Provide end-to-end security, with minimal end-user disruption
- Provide usage information appropriate to existing accounting and billing systems

These software components provide access to online information from a wide variety of pervasive devices such as cellular phones, personal digital assistants (PDAs), and mobile computers, among other wireless and traditionally connected devices.

IBM WebSphere Everyplace Server addresses these needs for three types of customer sets:

- Enterprise customers who seek to extend their intranet applications to pervasive devices. These include job-task devices (such as electronic package delivery and tracking systems) and multifunction devices (such as laptop computers and PDAs). These customers may also wish to deliver select Internet content to users.

- Content providers who wish to deliver data and applications to consumers. These customers include enterprises providing Internet commerce, finance, information sites, and Internet portals.
- Internet service providers who wish to provide connection services to consumers and enterprise users.

In support of these customers and providers, Everyplace Server does the following:

- Allows you to leverage your existing e-business investments by extending e-business applications across new communications channels such as wireless networks to users of new classes of devices such as PDAs, browser-equipped phones, and Internet appliances.
- Offers you the flexibility to support your users' requirements for either client/server or browser-based applications, in either online or sometimes-connected networking models.
- Provides investment protection and interoperability through its support for open standards.
- Enables you to support your deployments through user, software, and device management.
- Includes the capability for you to deliver existing content to new devices.
- Allows you to extend your portal to users of pervasive devices, who can perform their own enrollment and receive personalized information on a broad range of cellular phones, Internet appliances, PDAs, and PCs.
- Is a secure and flexible IT infrastructure that can scale to meet the growing number of network-connected devices.
- Enables you to start now and add support for new technology (such as the latest handheld device or next-generation network technology) as it becomes available.

1.1.2 New in WebSphere Everyplace Server

WebSphere Everyplace Server has changed its name from WebSphere Everyplace Suite. The name change reflects the addition of new function to previous versions. A revised Setup Manager eases the installation process while providing a single interface for installing the primary WebSphere Everyplace Server components. Also new is Suite Manager, a tool that allows you to configure the primary WebSphere Everyplace Server components through a single, easy-to-use interface. This new version also has some new components, Location-Based Services and Intelligent Notification Services.

The new version of WebSphere Everyplace Server:

- Allows users or applications to more easily manage information with intelligent notifications that are triggered when events occur and/or content is available based on preferences
- Supports location-based information to dynamically determine the user's device position information, pass that information to an application, and manage the privacy of that location information
- Extends applications with hands-free access by integrating with the IBM WebSphere Voice Server giving users voice recognition and voice application access and integration to application content
- Provides access to Domino applications via integration with Domino application adapters in Domino Everyplace Server
- Incorporates encryption and authentication capabilities that enable you to deploy security-rich applications and integrates with Policy Director for enhanced authorization and access control
- Intends to provide scalability to support enterprise-wide and service provider-sized deployments

1.2 WebSphere Everyplace Server functions

IBM WebSphere Everyplace Server provides the functions necessary to enable both network access and application and content serving (e-business applications) to multiple device types, including WAP phones, PDAs, Internet appliances and screen phones, in addition to the large installed base of Internet browsers.

Figure 1-1 on page 7 provides a simplified view of WebSphere Everyplace Server. In this figure, we have a person using a handheld wireless device to surf the ‘Net, eventually reaching the applications and data on the right-hand side of the diagram. In the middle, WebSphere Everyplace Server provides the connection to the protected network housing the data and applications, and adapts the communications between the application and the mobile device.

WebSphere Everyplace Server includes components that provide the following functions:

- Connectivity -- connecting carrier mobile/wireless network to your wired network in a secured fashion, optimizing for the bandwidth and the device capabilities

- Content Handling -- transferring the information between the Web applications and the mobile device, performing any adaptations to the format so that the mobile device can receive, understand, and process, and render it
- Management -- setting up who can access what, and with which device

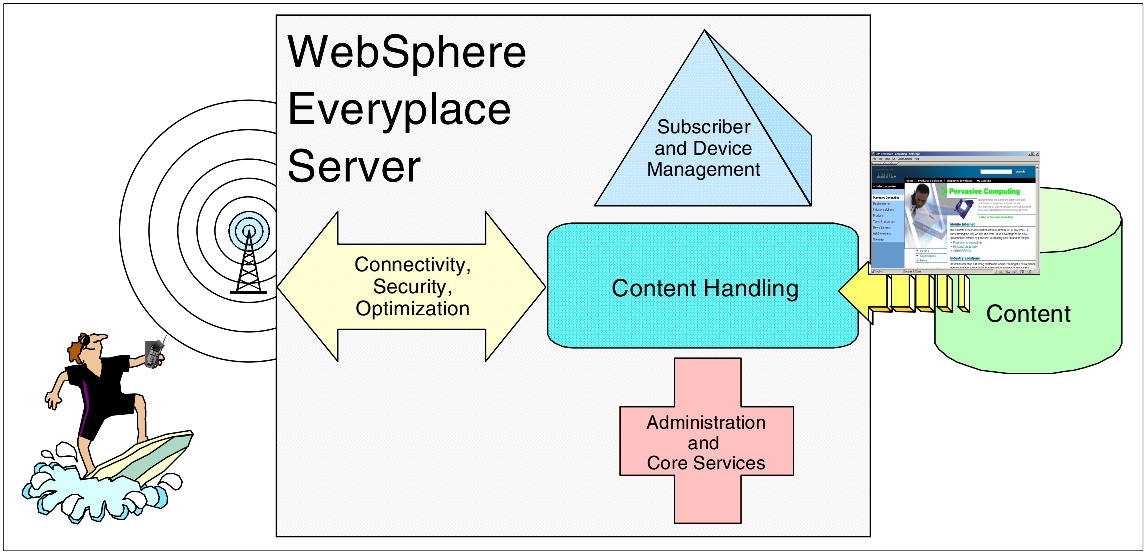

WebSphere Everyplace Server
Subscriber and Device Management
Connectivity, Security, Optimization
Content Handling
Content
Administration and Core Services

Figure 1-1 Simplified view of Everyplace Server

- Administration and Core Services -- underlying components, and tools for implementing and administering the WebSphere Everyplace Server environment

WebSphere Everyplace Server is not part of the infrastructure that comprises the business applications and data. In other words, WebSphere Everyplace Server is not the content provider. Rather, WebSphere Everyplace Server provides the infrastructure to enable many different devices to access existing Web applications and new mobile e-business applications.

Figure 1-2 illustrates, in a more technical format, how WebSphere Everyplace Server fits into a deployment of a multi-channel, end-to-end mobile e-business solution. In Figure 1-2, we have broken the overall capabilities of WebSphere Everyplace Server into six categories: connectivity, content handling, security, optimization, subscriber and device management, and core services. These categories are useful in understanding the relationships and interactions between WebSphere Everyplace Server components.

- **Connectivity:** Connectivity is the function that allows various types of device to connect to Everyplace Server, and from Everyplace Server into your business applications and data. Functionally there are two aspects to this:

- Provision of gateways to physically connect various types of networks to our end-to-end solution
- Convert the networking protocols to forms that can be directly used to access our business applications and data

So, the gateways provide the method of connecting such things as X.25, SNA, or WAP to our solution. Protocol conversion allows us to provide a TCP/IP programming interface on a remote system, and then to flow the resulting TCP/IP packets over the physical X.25 network that remote device is connected to, and out into our business applications.

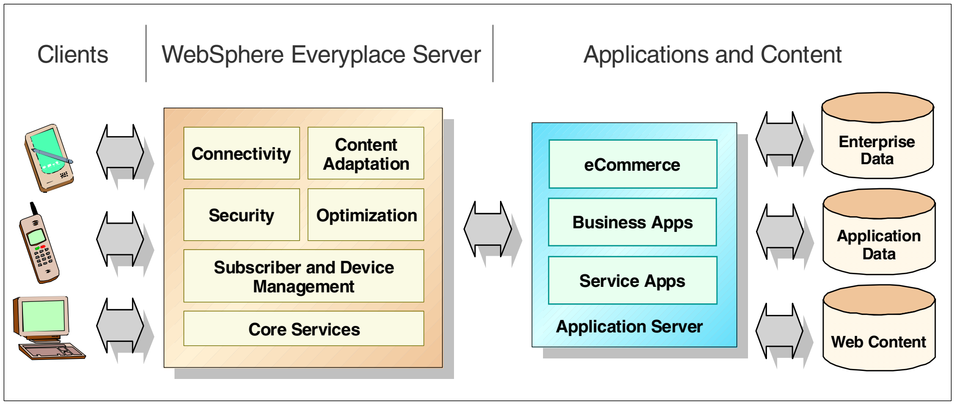


Figure 1-2 Everyplace Server end-to-end solution

- **Security**: WebSphere Everyplace Server has an integrated security model that provides single sign-on across all the components of an end-to-end mobile e-business solution. It fully integrates with standard security techniques such as virtual private network (VPN) technology, firewall protection, etc.
- **Optimization**: It is difficult to attract and retain customers if the system has poor performance. To optimize performance, WebSphere Everyplace Server includes caching and load balancing support to ensure high performance and scalability, TCP retransmission supression, IP header reduction, and compression.
- **Content handling:** Content handling is the issue of moving content from your e-business solutions to the point where it is required. This typically has two main aspects: moving the data and then ensuring that it is in a form that can be used once it gets to its destination.

  To that end, we can break down content handling into three main areas:

  - Transcoding -- by this we mean reformatting the data so that the receiving device can use it and display it.

    - Asynchronous messaging -- Asynchronous messaging is a complementary technology to Internet browsing that is designed for applications where a “fatter” client is acceptable, and where disconnected modes of operation are required. To support these it provides assured, asynchronous, once-only delivery of data across a broad range of hardware and software platforms.
    - Data synchronization -- IBM WebSphere Everyplace Server has the capability to manage the automatic exchange and updating of e-mail, schedules, transactions, and database exchanges between popular pervasive devices and database servers. This allows users to work offline, and connect to the network whenever it's convenient.
- **Subscriber and device management**: In the world of pervasive computing, a single device may have multiple users, and a single user may access the network using multiple devices. IBM WebSphere Everyplace Server includes Tivoli technology to manage subscribers and their devices, easing the burden of administration and system maintenance.
- **Core services**: The core services are the underlying framework on which the rest of WebSphere Everyplace Server is built. This includes the directory and repository services where all the data is stored, and the installation, configuration and administration, and all the other services that support them, including the HTTP servers, databases, etc.

1.3 Overview of WebSphere Everyplace Server

In this section we provide a high-level overview of each of the components that make up the WebSphere Everyplace Server. In doing so, we discuss how each component relates to the six functional areas presented in the previous section. This section is intended to provide the background on each component. Other chapters in this book contain more in-depth discussions of the components, and Chapter 3, “Component relationships” on page 71 discusses the interactions between the components.

The components of WebSphere Everyplace Server are:

- IBM Everyplace Wireless Gateway
- WebSEAL-Lite
- Tivoli SecureWay Policy Director
- Everyplace Cookie Proxy
- IBM WebSphere Edge Server
    - Edge Server Caching Proxy

  - Edge Server Load Balancer
- IBM WebSphere Transcoding Publisher
- IBM MQSeries Everyplace
- Sametime Everyplace Server Technology First Look (includes Domino and Domino Everyplace SMS)
- Location Based Services
- Intelligent Notification Services
- IBM Everyplace Synchronization Manager
- Tivoli Personalized Services Manager
  - Enrollment and Self Care
  - Customer Care
  - Device Manager
- Everyplace Setup Manager
- Everyplace Suite Manager
- IBM SecureWay Directory
- Active Session Table Server

Some of these components are only available in the WebSphere Everyplace Server Service Provider Offering. For a discussion of the differences between the separate Everyplace Server offerings, see Chapter 2, “Differentiating the offerings” on page 45.

The WebSphere Everyplace Server CD-ROM package includes other, underlying infrastructure products: WebSphere Application Server, DB2, and the HTTP Server. Documentation for these components is readily available elsewhere, so we do not discuss them in this book.

Figure 1-3 on page 11 shows the WebSphere Everyplace Server components organized by their primary functional category. Certainly at this level of detail there is bound to be some overlap, although some components clearly fit in multiple functional areas. For example, IBM Everyplace Wireless Gateway provides connectivity to the WebSphere Everyplace Server domain from devices on non-IP networks, but also provides an element of security, as it is capable of authenticating users that pass through it.

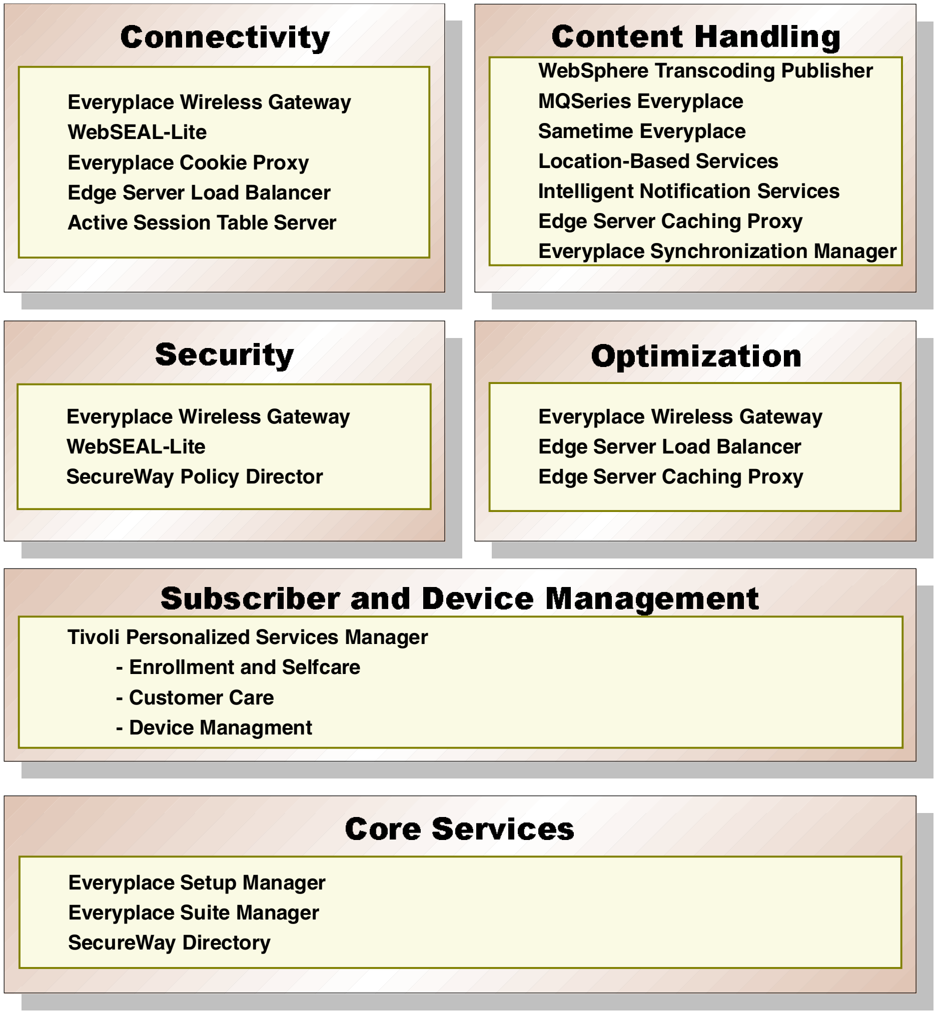

Connectivity
Everyplace Wireless Gateway
WebSEAL-Lite
Everyplace Cookie Proxy
Edge Server Load Balancer
Active Session Table Server
Content Handling
WebSphere Transcoding Publisher
MQSeries Everyplace
Sametime Everyplace
Location-Based Services
Intelligent Notification Services
Edge Server Caching Proxy
Everyplace Synchronization Manager
Security
Everyplace Wireless Gateway
WebSEAL-Lite
SecureWay Policy Director
Optimization
Everyplace Wireless Gateway
Edge Server Load Balancer
Edge Server Caching Proxy
Subscriber and Device Management
Tivoli Personalized Services Manager
- Enrollment and Selfcare
- Customer Care
- Device Managment
Core Services
Everyplace Setup Manager
Everyplace Suite Manager
SecureWay Directory

Figure 1-3 WebSphere Everyplace Server components by functional area

Figure 1-4 depicts one possible configuration of the WebSphere Everyplace Server (shown as WES in the figure) Service Provider Offering components in IBM @server pSeries racks. The purpose of this picture is to give the reader a general feel for how the Service Provider Offering might be physically implemented, not as an indication of minimum or maximum configuration.

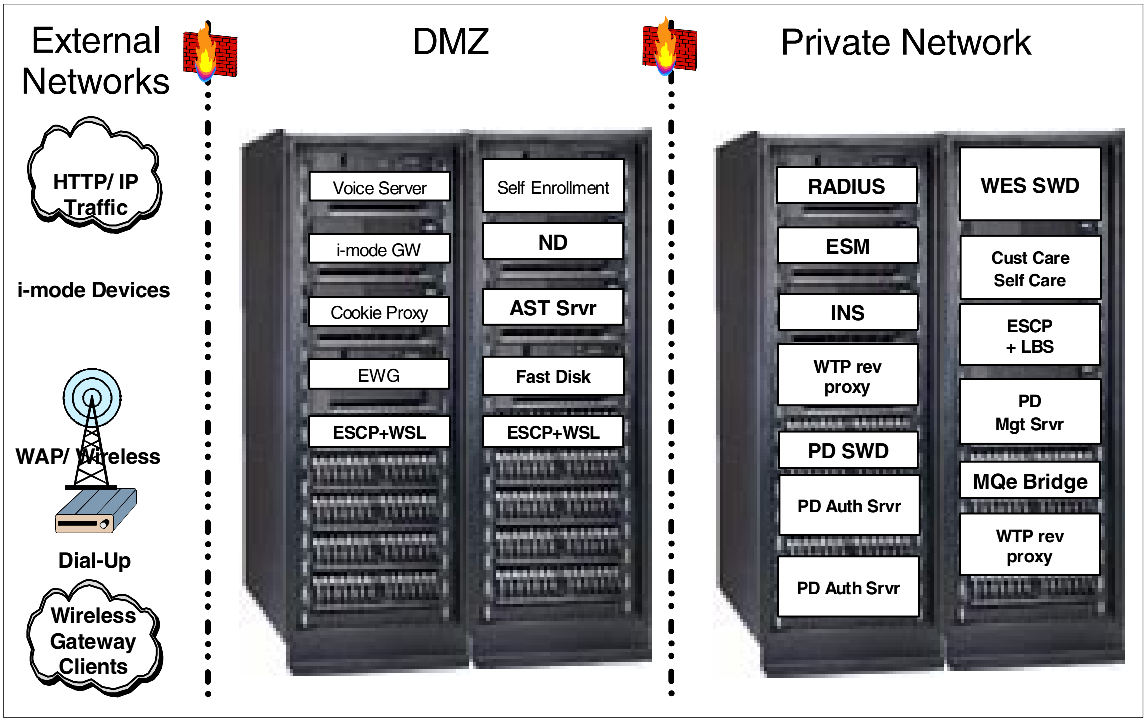


Figure 1-4 Everyplace Server Service Provider Offering in server racks

1.3.1 Everyplace Wireless Gateway

IBM Everyplace Wireless Gateway (EWG) 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. While the Everyplace Wireless Gateway's main function is to provide connectivity between various networks, it plays an important role in security and optimization as well. It can encrypt, compress, and minimize the data that passes through the wireless link, thereby increasing the speed of messaging. It also has the capability to authenticate users.

Components

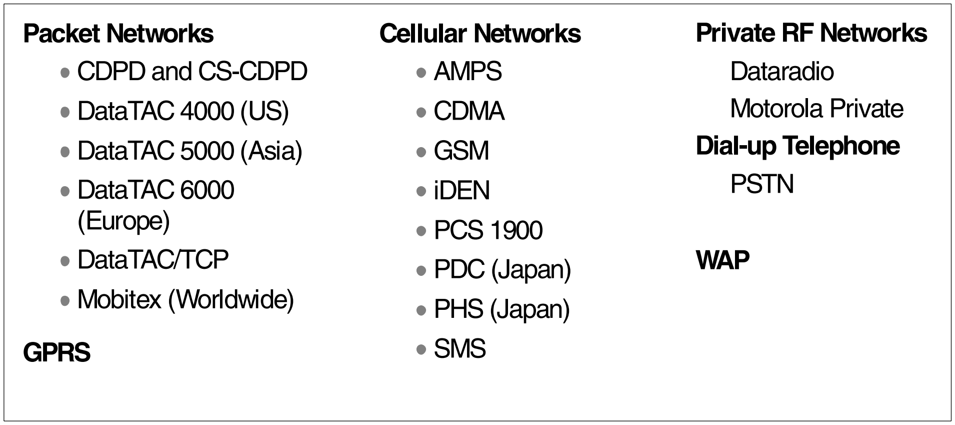
The main components of the Everyplace Wireless Gateway are:

- The Wireless Gateway, which runs on the IBM AIX and Sun Solaris operating systems, provides a standard communications interface 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 remotely with Wireless Gateways.
- The Wireless Client provides an optimized and secure IP tunnel for communication with the Wireless Gateway using a variety of wireless and wire line networks.
- Persistent data storage, which 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, an AIX or Solaris daemon that manages communications among Wireless Gatekeepers, the Everyplace Wireless Gateway and persistent data storage.

Supported Networks

Everyplace Wireless Gateway (EWG) supports networks shown in Figure 1-5.



Packet Networks

- CDPD and CS-CDPD
- DataTAC 4000 (US)
- DataTAC 5000 (Asia)
- DataTAC 6000 (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

Dial-up Telephone

PSTN

WAP

Figure 1-5 Supported networks

Features

Some of the key features and benefits provided by Everyplace Wireless Gateway are:

- Scalability

Everyplace Wireless Gateway supports clustering of gateways for larger systems and backup. Thus it is possible to add gateways to handle increases in traffic without a disruption of service.

- Messaging Gateway

  Everyplace Wireless Gateway provides push messaging capabilities for clients such as WAP phones, SMS messaging and others, allowing information delivery without end user intervention.

- Security

  Everyplace Wireless Gateway 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

  Everyplace Wireless Gateway 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

  Everyplace Wireless Gateway allows standards-based support for devices with WAP browsers installed.

- Worldwide network technology support

  Everyplace Wireless Gateway delivers applications to mobile users over a wide variety of wireless and wired networks, whether or not they natively support IP.

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 Millennium Edition
- TCP application, a new WAP service resource, 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

Connectivity

IBM Everyplace Wireless Gateway supports a variety of wireless and dial-up network technologies. All data traffic that flows through the Wireless Gateway uses a 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 at least one MNC for each network provider that you will use. There is a different MNC for each specific type of network or bearer through which Wireless Clients or WAP clients connect. For a list of available MNC types, see “Supported networks and platforms” on page 422.

Existing applications that use a TCP/IP interface may use either wireless networks or wire line networks. Using Everyplace Wireless Gateway shields network-specific details from the user application and provides network-specific enhancements, such as data compression, data encryption, data optimization, and authentication.

Connections to mobile devices can be divided into three general categories:

- Wireless Clients (intelligent devices that run the Everyplace Wireless Client software)
- Messaging clients
- WAP devices
- Dial-in clients

Everyplace Wireless Gateway also supports non-IP packet-oriented connections, such as Mobitex or DataTAC. Therefore, Everyplace Wireless Gateway supports such devices as the Blackberry device from Research In Motion. Everyplace Wireless Gateway also supports wireless applications that use Short Message Service (SMS).

WAP Gateway

When Everyplace Wireless Gateway is configured as a WAP Gateway, it performs a protocol conversion between WAP and HTTP, using the following process. The WAP Gateway listens for data and messages arriving from WAP clients. When data arrives from a WAP client, Everyplace Wireless Gateway:

1. Translates Wireless Session Protocol (WSP) requests into HyperText Transport Protocol (HTTP) requests
2. Forwards the HTTP requests to an HTTP proxy
3. Converts the response headers from HTTP into Wireless Session Protocol (WSP) response headers

4. Encodes the content from WML and/or WMLScript into binary XML (wbxml)
5. Forwards the content to the WAP client

Everyplace Wireless Gateway support when configured as a WAP Gateway includes:

- Compliance with the WAP Version 1.2.1 WAP Forum standard. For a discussion of the specifications, visit http://www.wapforum.org.
- Increased efficiency and caching speed using a plug-in for the IBM WebSphere Edge Server Caching Proxy.
- Persistent storage of cookies on behalf of WAP clients (Everyplace Cookie Proxy).
- Secure features based on the Internet Transport Layer Security standard using a Wireless Transport Layer Security (WTLS) connection between a WAP client and the WAP Gateway.
- Secure HTTP (HTTPS) requests using Transport Layer Security (TLS) on behalf of WAP clients.
- A choice of encryption key strengths for both key exchange and bulk encryption methods.
- Interfaces to IP network types; for example, circuit-switched or dial networks such as Global System for Mobile communication (GSM), time division multiple access (TDMA), or code-division multiple access (CDMA).
- Interfaces to Short Message Service (SMS) network types; for example, SMS-SMPP and SMS-UCP.

Messaging Gateway

The Everyplace Wireless Gateway can be configured as a messaging gateway. The messaging gateway supports push technology to transmit information to clients without previous user action. In a WAP network, the messaging gateway is a push proxy gateway (PPG).

Authentication

One key aspect of the Everyplace Wireless Gateway is its ability to authenticate users entering the gateway. The gateway is a WebSEAL-Lite trusted server, so WebSEAL-Lite does not reauthenticate users entering through the Everyplace Wireless Gateway. This allows for single sign-on. Everyplace Wireless Gateway is also able to share information with other WebSphere Everyplace Server components through an LDAP directory server. For more information on the authentication capabilities of Everyplace Wireless Gateway and its interaction with WebSEAL-Lite, please see Chapter 9, “Security” on page 359.

Everyplace Wireless Gateway provides a secure and optimized connection to mobile devices. Wireless devices with installed client software can take advantage of Everyplace Wireless Gateway's WLP support. Everyplace Wireless Gateway provides Wireless Transport Layer Security (WTLS) support using:

- Diffie-Hellman or RSA key exchange (RSA 1024, 768 or 512 bits)
- Encryption using RSA RC5 (40, 56, or 128 bit)
- DES and Triple DES algorithms
- SHA 1 (Secure Hash Algorithm) Message Authentication Codes (40-80 bit)

Everyplace Wireless Gateway uses TLS/SSL to establish secure connections between the gateway and back-end servers.

Clustering

The Everyplace Wireless Gateway can be configured to be a principal node or subordinate node in a cluster of Everyplace Wireless Gateways. In this manner the Everyplace Wireless Gateway distributes and services communication requests and provides load-balancing efficiency. A cluster manager is automatically installed when you install the Everyplace Wireless Gateway.

1.3.2 WebSEAL-Lite

Generally, Web servers maintain and enforce the authentication and authorization of the users to whom they provide resources. In a domain of several Web servers, this means several lists of users and several access control lists (ACLs). Having several users lists and ACLs becomes an unwieldy administration effort and an inconvenience to users, who might be forced to log on several times within a domain.

Even if you solve the problem of having multiple user lists and ACLs within a domain of Web servers, adding new entry points (such as a wireless gateway) can introduce a challenge to offering users a single sign-on capability. Tivoli Systems, in close partnership with the IBM Pervasive Computing Division, developed WebSEAL-Lite to address this issue specifically for the WebSphere Everyplace Server environment. WebSEAL-Lite is unique to WebSphere Everyplace Server.

WebSEAL-Lite is an authentication and authorization plug-in for Edge Server Caching Proxy (ESCP) that provides a central authentication point for all users who wish to access resources within a secured domain. It combines the caching Web proxy of Edge Server Caching Proxy with the authorization engine of Policy Director (see 1.3.3, "Policy Director" on page 18) to deliver protected resources to authorized users.

WebSEAL-Lite is the central point of user authentication for the Everyplace Server domain. It authenticates users who are defined to the Everyplace Server domain when they attempt to access Everyplace Server services. WebSEAL-Lite also allows you to use gateways other than Everyplace Wireless Gateway if desired. At least one version 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.

Edge Server Caching Proxy is a prerequisite for WebSEAL-Lite since WebSEAL-Lite runs as a plug-in to the Edge Server Caching Proxy. WebSEAL-Lite can be configured in one of two modes: reverse proxy or forward proxy.

Note that WebSEAL-Lite allows for single sign-on (user ID and password) for all services within the WebSphere Everyplace Server domain. With this feature, user authentication only needs to be done once to access services requiring a user ID and password. For example, users logging on to an enterprise site that uses WebSphere Everyplace Server give their user ID and password, which WebSEAL-Lite authenticates. If users want to change their password (which they do through Tivoli Personalized Services Manager Self Care) they do not need to enter their user ID and password again to access this or any other services in the WebSphere Everyplace Server domain.

1.3.3 Policy Director

Tivoli SecureWay Policy Director (Policy Director) is a robust and secure policy management tool for e-business and distributed applications. It is the core element of a security framework for providing centralized authentication and authorization services for business network systems without replacing existing systems. Policy Director is a complete authorization, network security, and policy management solution that provides end-to-end protection of resources over geographically dispersed intranets and extranets. Policy Director is optional in WebSphere Everyplace Server.

Without Policy Director, WebSphere Everyplace Server WebSEAL-Lite performs authentication, but that is the extent of the access control. Without Policy Director, there is no mechanism to grant or deny partial access (such as read-only, or update but not delete), nor to set up access based on group membership, nor to store privacy information. Policy Director provides the WebSphere Everyplace Server environment with privacy control and fine-grain access control, based on groups or individual users.

The major functions of the Policy Director are:

- Authentication -- affirming that the user is who they say they are
- Authorization -- checking whether the users has permission to perform this action (read, write, update, create, delete, execute, traverse) on the object they're trying to access (data, directory, script, executable)
- Centralized management -- reducing the administration effort of each application maintaining its own side files, by providing a single security console that manages access control to all the Web objects on all the Web servers in the Policy Director domain
- Authorization Application Programming Interface (aznAPI) -- allowing external programs to communicate with and use the functions of Policy Director

The above-mentioned functions are provided by the following five major components, each of which is typically on a dedicated machine:

- Security Servers (these are processes and can run on the same machine or spread across several machines)
  - WebSEAL -- for HTTP and HTTPS traffic (not the same as WebSEAL-Lite)
  - NetSEAL -- for all other TCP/IP traffic (FTP, Telnet, etc.); requires a NetSEAT client on the requesting system
  - CorbaSEAL -- for Common Object Request Broker Architecture objects
- Authorization Server(s)
- Directory Server and User Registry
- Management Server (one per domain)
- Management Console (can manage multiple domains)

Figure 1-6 on page 20 shows, at a high level, a picture of the Policy Director components and how they interact.

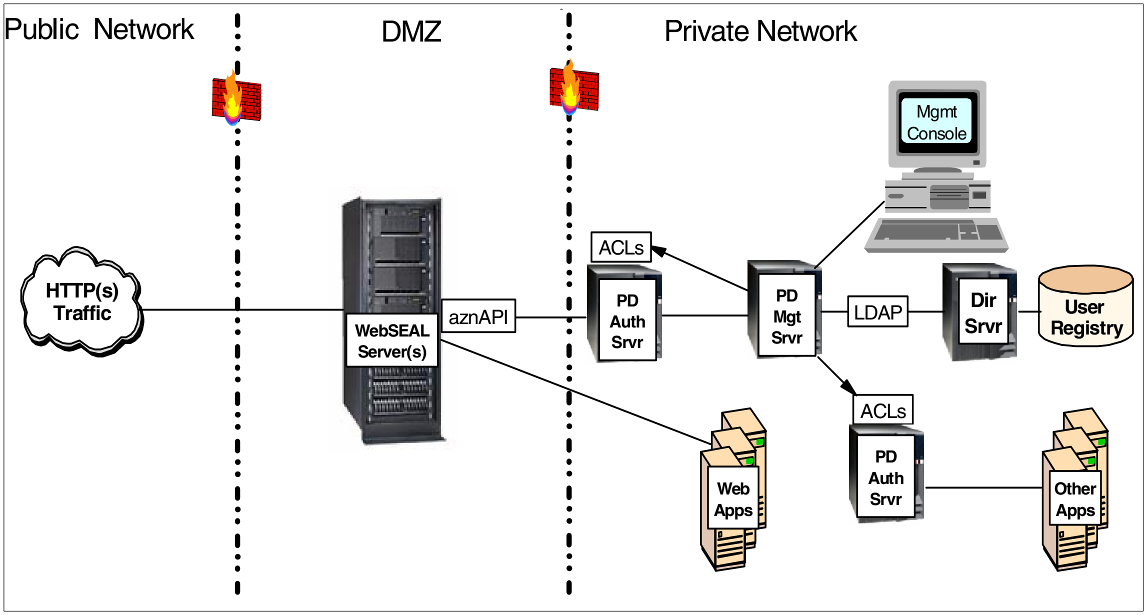


Figure 1-6 Policy Director components

The core of Policy Director is the Management Server. The Management Server:

- Owns the master ACL
- Publishes ACL replicas to the Authorization Servers (Auth Servers)
- Updates the Authorization Server cache with any changes to the ACL
- Manages the user registry
- Performs updates to the user registry on behalf of other systems through aznAPI requests.

The Management Server relies on DCE Kerberos security and remote procedure calls (RPCs) to achieve secure communications with the other Policy Director components that are running on different machines. For this reason, Policy Director requires a DCE client on each of its servers. The Management Server also requires an LDAP client (in WebSphere Everyplace Server, the SecureWay Directory client) so that it can communicate with the Directory Server. The Management Server also includes a proxy process called the Directory Services Broker (DSB). The DSB provides a proxy to allow NetSEAT clients to make Cell Directory Service requests. This is necessary for the Management Console on Windows NT or Windows 2000 to operate correctly.

WebSEAL sits in the DMZ and acts as a reverse proxy for HTTP traffic. WebSphere Everyplace Server uses WebSEAL-Lite instead of WebSEAL. For more information about WebSEAL-Lite and how it differs from WebSEAL, please see 3.3.5, “WebSEAL-Lite” on page 92.

Policy Director employs a user registry, and requires a directory server to manage it. In previous versions of Policy Director, the directory server was a DCE Cell Directory Server, but in the version of Policy Director that WebSphere Everyplace Server includes, it is an LDAP directory server. Policy Director supports IBM SecureWay Directory and Netscape Directory Server. Policy Director also permits using multiple LDAP directory servers, enabling high availability and scalability.

Different WebSphere Everyplace Server components could use different replicas as their primary directory, and access the master or another replica if their primary directory is not available. Implementing multiple LDAP directories and servers can greatly improve the performance of many Policy Director operations, such as WebSEAL or WebSEAL-Lite user authentication and GSO junctions. For more discussion of LDAP directory server availability and scalability considerations, please see Chapter 8, “Availability” on page 345 and Chapter 7, “Scalability” on page 301, respectively.

The Policy Director Management Console provides a graphical interface to the management subsystem. The console is a Java application available on all supported Policy Director platforms and it communicates with the Policy Director Management Console via RPC. The management console depends upon a DCE client being installed on the system running the application. Often, the management console is run on a Windows NT workstation rather than on the server platform. In this case, the NetSEAT client is required to provide the lightweight DCE connectivity.

Policy Director's authorization capabilities are available to application developers through the Authorization API (aznAPI). When the aznAPI is used in remote model, the authorization server receives (via RPC) authorization requests from the application server. The Policy Director authorization server requires both a DCE client and an LDAP client.

When architecting a solution that includes Policy Director, be sure to account for the following prerequisites:

- Distributed Computing Environment (DCE), which includes a DCE Security Server and a DCE Cell Directory Server. This will be used within the Policy Director complex.
- Optionally, an LDAP directory server.

Policy Director is designed to unite core security technologies around common security policies. This helps reduce implementation time and management complexity, thereby lowering the total cost of secure computing.

Policy Director provides the following features:

- Access control for Web objects
- Centralized security to your existing Web and TCP/IP applications
- Replication and load balancing
- Consistent, manageable access control policy
- Extensible authentication and authorization
- Secure remote access and personalized access
- One-time authentication capability with access to multiple Web resources
- Reduced administration costs
- Support for Public Key Infrastructure (PKI)

Note: Policy Director and WebSphere Everyplace Server use different nomenclature for user information in the LDAP directory. Thus, by default, Policy Director and WebSphere Everyplace Server have separate LDAP directories. WebSphere Everyplace Server compensates for this by having Tivoli Personalized Services Manager publish to both LDAP directories and keep them both current. There is a Policy Director update, 3.7.1-POL-0004E, that allows this sharing to occur, by allowing Policy Director to create or import user definitions that are named with the `uid` attribute.

1.3.4 Everyplace Cookie Proxy

The Cookie Proxy provides support for NTT DoCoMo i-mode phones. This component is a plug-in for Edge Server Caching Proxy and saves important information for Web browsers with limited functions. This can include cookie values, and other values for session management and basic authentication.

Note: Currently, Everyplace Cookie Proxy is only available when installing on machines in Japan.

When an HTTP response is downloaded, the Everyplace Cookie Proxy reads its header and saves any cookie values (that is, name=value pair, path and expires data) into its own table. Then, when the Web browser issues an HTTP request to the Web server, the Everyplace Cookie Proxy retrieves the saved cookie values, adds them to the request header, and passes the request to the back-end server.

The Everyplace Cookie Proxy checks the expiration value of a cookie only when it is saved in its own table, and it does not check the expiration value of any cookies when they are retrieved. All saved cookie values remain in the Everyplace Cookie Proxy's table until the session is cleaned up by the cleanup daemon. This means that a cookie is passed to the back-end server as long as the cookie entry remains in the table, and monitoring for expiration is the responsibility of the back-end server.

When a user tries to access a Web application for the first time in a session (or after the expiration of a time-out delay since this user last had access to this application), the Everyplace Cookie Proxy generates a session ID. If the host name of the URL is that of the back-end server or the Everyplace Cookie Proxy, this session ID is added to the URLs of the hyperlinks in the HTTP response. Every time an HTTP request is submitted, the Everyplace Cookie Proxy checks whether the incoming session ID is valid. This session information is stored in the table of Everyplace Cookie Proxy until the session cleanup daemon removes it. The session expiration period and the cleanup interval can be defined during the configuration of the Everyplace Cookie Proxy.

If the Web browser submits an HTTP request and the back-end server requires basic authentication, but the Everyplace Cookie Proxy has not yet stored a valid user ID and password, the Everyplace Cookie Proxy sends a login form to the Web browser. (This form can be customized by the back-end Web application developer or the administrator who sets up the Everyplace Cookie Proxy.) When the user logs in, the Everyplace Cookie Proxy reads the body of the HTTP request and saves the user ID and password. Then, throughout the rest of the session, whenever the browser submits an HTTP request, the Everyplace Cookie Proxy

- Retrieves the user ID and password
- Adds the default realm after the user ID if the realm is specified in its configuration file
- Encodes the resulting string, using BASE64
- Adds the string to the request header
- Passes the header to the back-end server

Performance and scalability issues must be addressed when the Everyplace Cookie Proxy is implemented. The physical architecture will have an important effect on both performance and scalability, so for better throughput and security, the Edge Server Caching Proxy, including the Everyplace Cookie Proxy, should be installed on a machine separate from the other components of WebSphere Everyplace Server.

The use of HTTPS is one factor that affects the number of HTTP requests processed per unit time and the turnaround time per request. In general, the overhead for processing HTTPS is much larger than the overhead for cookie processing by the Everyplace Cookie Proxy.

For the best use of memory, the MaxSession parameter, defined in the configuration file for the Everyplace Cookie Proxy, jpas.conf, should not be too large relative to the estimated number of user accesses.

1.3.5 WebSphere Edge Server

The IBM WebSphere Edge Server (or simply Edge Server) helps information technology administrators to provide better service both to users who access documents stored on the enterprise's server machines and to their internal users who access the Internet. In other words, Edge Server helps you both to host Web-accessible content and to provide Internet access more efficiently and economically. The name Edge Server indicates that the software usually runs on machines that are close (in a network configuration sense) to the boundary between an enterprise's intranet and the Internet. The Edge Server includes two main components that provide complementary capability: the Caching Proxy and the Load Balancer.

WebSphere Edge Server Caching Proxy

The Caching Proxy component intercepts data requests from end users, retrieves the requested information from content-hosting machines, and delivers it back to the end users. Most commonly, the requests are for documents stored on Web server machines (also called origin servers or content hosts) and delivered via the HyperText Transfer Protocol (HTTP). However, you can configure the Caching Proxy to handle other protocols, such as File Transfer Protocol (FTP).

When retrieving certain types of content, the Caching Proxy stores it in a local cache before delivering it to the requester. The most prominent example of cacheable content is static Web pages (those without portions that are dynamically generated at access time). Caching enables the Caching Proxy to satisfy subsequent requests for the same content directly from the cache, which is much quicker than retrieving it again from the content host.

The Caching Proxy can be useful both when hosting Web-accessible content and when providing Internet access:

- When used by content hosts, the Caching Proxy is installed as a reverse proxy between the Internet and the enterprise's content hosts. It intercepts user requests arriving from the Internet, forwards them to the appropriate content host, caches the returned data, and delivers it to the users across the Internet.

- When used by Internet access providers, the Caching Proxy is installed as a forward proxy between an enterprise’s end users and the Internet. It forwards users’ requests to content hosts located across the Internet, caching and delivering the retrieved data to users.

The Caching Proxy hosts the WebSEAL-Lite plug-in and serves as the central point of entry into the WebSphere Everyplace Server domain from third-party gateways and the Internet. It is also the next hop for HTTP traffic coming from the IBM Everyplace Wireless Gateway.

WebSphere Edge Server Load Balancer

The Load Balancer also intercepts data requests from end users, but rather than actually retrieving data, it forwards the request to the server machine that is currently best able to fill the request. In other words, it balances incoming requests among a defined set of machines that service the same type of requests. A load balancer is sometimes termed a sprayer because it divides up an incoming stream of requests and distributes them to the machines that service them. The Load Balancer can distribute requests to many types of servers, including both HTTP origin servers and Caching Proxy machines. If desired, you can write rules that specify the criteria used by the Load Balancer when determining which server can best handle a request.

Like the Caching Proxy, the Load Balancer can be useful both when hosting Web-accessible content and when providing Internet access.

- When used by content hosts, the Load Balancer is installed between the Internet and the enterprise’s back end servers, which can be content hosts, Caching Proxy machines, or mail server machines that service the POP3 or IMAP protocols. The Load Balancer acts as the enterprise’s single point-of-presence on the Internet, even if the enterprise uses multiple back-end servers because of high demand or a large amount of content. If the Load Balancer’s Content Based Routing (CBR) module is installed together with the Caching Proxy, HTTP requests can even be distributed based on URL or other administrator-determined characteristics, eliminating the need to store identical content on all back end servers.
- When used by Internet access providers, the Load Balancer is installed between an enterprise’s end users and two or more Caching Proxy machines in the enterprise’s intranet, to balance the load between them. Load balancing multiple Caching Proxy machines provides highly reliable access to the Internet even in the face of high demand. You can also guarantee high availability by installing a backup Load Balancer to take over if the primary one fails temporarily.

The Load Balancer provides dynamic load balancing, scalability, and high availability for servers, boosting overall server performance by automatically finding the optimal server within a group of servers to handle each incoming request. It can be used with Web servers, e-mail servers, distributed parallel database queries, and other Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) applications.

The Load Balancer contains the following subcomponents:

- **Content Based Routing**: Performs balancing in one of two ways:
  - For HTTP, Content Based Routing performs balancing based on the content of an HTTP client request. This method requires the Caching Proxy on the same machine.
  - For IMAP and POP3, Content Based Routing performs balancing on IMAP or POP3 mail servers. It selects the appropriate server based on the user ID and password provided by the client and does not require the Caching Proxy.
- **Dispatcher**: An IP packet-level load balancer. It provides high performance, low latency load balancing using weights and measurements that are dynamically set. It also provides built-in support for protocols such as HTTP, FTP, SSL, NNTP, IMAP, POP3, SMTP, and Telnet, but it can be extended to support both TCP and UDP.
- **Interactive Session Support**: Balances the load on servers using a domain name server. This is done by communicating with server agents that are used to monitor the load and then altering the IP address returned to the client based on this load. Interactive Session Support can also provide the same server load information to the Dispatcher subcomponent.

1.3.6 Everyplace Synchronization Manager

Everyplace Synchronization Manager enables handheld computing devices to link remotely to desktop applications. Mobile users can easily synchronize data with Microsoft Exchange, Lotus Notes or any ODBC compliant database, such as DB2, Oracle or Sybase. The mobile device can synchronize using a modem, a cellular phone, the Internet, a wireless connection, the intranet, a local area network (LAN) or a wide area network (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 also 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.

- **Everyplace Synchronization Manager administrator**

  – 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.

- **Exchange connector**

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

- **Notes connector**

  Enables Synchronization Manager to synchronize with Lotus Notes. 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 a LAN. It 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.

> **Note:** This component is only available with specific Everyplace Server license keys.

1.3.7 WebSphere WebSphere Transcoding Publisher

The WebSphere Transcoding Publisher adapts Web content based on the destination device characteristics and network service level. You can enhance the performance of the WebSphere Transcoding Publisher by installing Edge Server Caching Proxy, which stores transcoded material. Doing this removes the necessity of retranscoding Web pages each time they are retrieved.

IBM WebSphere Everyplace Server V2.1 uses IBM WebSphere Transcoding Publisher V3.5. Each WebSphere Transcoding Publisher server can run in any of three basic configurations:

- As a network proxy
- As a network proxy with an external cache
- As a filter in WebSphere Application Server

Network proxy

In this configuration WebSphere Transcoding Publisher is a single service that tailors content coming from many different Web servers. The proxy intercepts HTTP requests and responses as they flow between the user and the Web server. This configuration does not tailor content that is encrypted between the user and the Web server.

Network proxy with an external cache

If you use a cache server in your network, WebSphere Transcoding Publisher can use it to store and retrieve transcoded Web pages and intermediate results. This may enable WebSphere Transcoding Publisher to avoid repeating the transcoding of frequently accessed pages. You will need to supply the address and port number used by the cache server when you configure WebSphere Transcoding Publisher.

If you run WebSphere Transcoding Publisher as a network proxy, with or without a cache server, and your network uses a firewall, you will need to configure WebSphere Transcoding Publisher with the address and port number used by the firewall.

As a reverse proxy

When you use WebSphere Transcoding Publisher as a network proxy, it acts on behalf of your client devices, which are configured so that their HTTP traffic is routed through the proxy. You can also use WebSphere Transcoding Publisher as a reverse proxy, which means that it acts as a proxy on behalf of a single Web server or several Web servers, rather than on behalf of clients. In this configuration, client browsers or wireless gateways are not configured to use the WebSphere Transcoding Publisher server as a proxy. Instead, you supply links to your clients in which the host name of the WebSphere Transcoding Publisher server is substituted for the host name of the Web server for which WebSphere Transcoding Publisher serves as a reverse proxy. When WebSphere Transcoding Publisher receives a request for a document from the server for which it is acting as a reverse proxy, WebSphere Transcoding Publisher will:

1. Redirect the request to retrieve the proper document
2. Transcode the document appropriately for the device and network types

3. Modify the links in the document so that when a user selects a link, the request will be routed through WebSphere Transcoding Publisher
4. Return the document to the user

You can use WebSphere Transcoding Publisher as a reverse proxy when it is configured as a network proxy with or without a cache server. WebSphere Transcoding Publisher can operate as a reverse proxy and as a regular proxy at the same time. The WebSphere Transcoding Publisher server can be installed on the same machine as the Web server or on a different machine.

Filter running in IBM WebSphere Application Server

When running as a filter WebSphere Transcoding Publisher tailors the content generated by the WebSphere Application Server on which it is running. It can tailor the content before it is encrypted and sent to a user. This configuration is not supported on Linux. When you run WebSphere Transcoding Publisher as a filter, you cannot use ports to select network preference profiles.

WebSphere Transcoding Publisher works with WebSphere Application Server Version 3.5. If you use WebSphere Application Server in your network, running WebSphere Transcoding Publisher as a filter enables it to tailor the output of Web applications for your pervasive devices. If you use encryption within the WebSphere Application Server, then you must use WebSphere Transcoding Publisher as a filter so that it can modify content before it is encrypted. If you do not use encryption, then you could run WebSphere Transcoding Publisher either as a filter or as a proxy. Running as a proxy, WebSphere Transcoding Publisher can transcode non-encrypted documents from any version of WebSphere Application Server.

If you run WebSphere Transcoding Publisher as a WebSphere Application Server filter, you must identify the Web applications whose output you want WebSphere Transcoding Publisher to transcode.

Building programs using JavaBean components

The programs inside the WebSphere Transcoding Publisher server that modify content are called transcoders. The transcoders are also provided as JavaBean components, which can be then run independently of WebSphere Transcoding Publisher. This provides a means for other server programs, such as servlets, independent content-providing programs, or Java Server Pages (JSPs), to invoke single transcoders directly. The JavaBean wrapper provides the transcoder with the same information about the system and the request that it receives when inside the WebSphere Transcoding Publisher transcoding server. This allows the component to operate the same way in both contexts.

The JavaBean components are always installed when you install the WebSphere Transcoding Publisher server on AIX, Sun Solaris, and Linux. When installed on Windows NT and Windows 2000 servers the installation of the JavaBean components are optional.

Using a central directory

If you have several WebSphere Transcoding Publisher servers in your network, you can choose to store the configuration information for each WebSphere Transcoding Publisher server on that server's machine, or you can use a central directory, IBM SecureWay Directory, Version 3.2, to store one or more server models to be used by your WebSphere Transcoding Publisher servers. A server model contains configuration information that can be used by any number of WebSphere Transcoding Publisher servers in your network.

Using a central directory enables you to define common configurations to be used by several servers and to maintain them from an administration console anywhere in your network. Then, if you need to make a change to the shared information (for example, to add a new stylesheet or modify a device profile), you can make the change to the server model and all WebSphere Transcoding Publisher servers that use that server model will recognize the change. Refer to Chapter 6.3 in *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5 Extending Web Applications to the Pervasive World*, SG24-6233 for more information on using a central directory.

Security

Users can edit local settings without logging in, and they can access all data at an administration console when a central directory is not being used; however, users must log on to the administration console if a central directory is being used, and the login is verified by the directory server.

The transcoding server does not provide a login mechanism. It will be accessible to any user with permission to log on to the machine (either locally or by telnet) and execute commands in the WebSphere Transcoding Publisher directories. If you want to limit access to WebSphere Transcoding Publisher, consider installing it on a machine with limited access or in a file system with limited permissions.

Hardware and software prerequisites

WebSphere Transcoding Publisher Server is a Java product that performs best on machines that are designed to be servers. These machines are generally built for improved scalability and performance. For example, server machines usually have room for large amounts of memory and offer large L1 and L2 caches. The following list details some of the prerequisites that are important in the WebSphere Everyplace Server environment:

- To use centralized configuration data, you must have SecureWay Directory, Version 3.2, installed somewhere in your network.
- If you run WebSphere Transcoding Publisher as a filter in WebSphere Application Server, Version 3.5, the prerequisites for WebSphere Transcoding Publisher are the same as those for WebSphere Application Server.
- WebSphere Transcoding Publisher must be installed on the same machine as the authentication server, or else WebSphere Transcoding Publisher will not accept the authentication, since the source IP will be WebSphere Transcoding Publisher, not the authentication server. Manual configuration is needed to make WebSphere Transcoding Publisher acknowledge the Authentication servers from a different IP address.

Note: WebSphere WebSphere Transcoding Publisher is intended to be deployed as a proxy in the Everyplace Server domain. It is not intended to be used as a servlet or a JavaBean within the Everyplace Server domain. A servlet filter mode for WebSphere Transcoding Publisher has not been tested; thus we do not recommend using WebSphere Transcoding Publisher as a servlet.

1.3.8 MQSeries Everyplace

MQSeries Everyplace is a toolkit designed with an emphasis on the frugal use of system resources, both in the messaging client and over the associated client network link. It enables pervasive devices to queue messages and transactions, and assure their completion (once and only once), in a secure and efficient manner in both connected (online) and disconnected (offline and store-and-forward) end-user scenarios.

MQSeries Everyplace integrates many of the functions that require application programming in other MQSeries family members. Thus encryption and compression are built-in. Similarly, MQSeries Everyplace efficiently supports both reliable and unreliable communications (for example, local area networks, PSTN over land lines or mobile links, and communications over selected packet radio networks). This communication support is designed to operate with a minimum of user intervention, with system entities such as channels and transmission queues being effectively hidden from users, programmers, and administrators.

MQSeries Everyplace allows mobile workers to access corporate data and applications on many platforms in an MQSeries network with all the trusted benefits of MQSeries such as:

- Industrial strength messaging
- Reliable communications

- Assured, once-only delivery of messages
- Powerful encryption
- Optimized data streams
- Runs on laptops and handheld devices
- Provides immediate server or mainframe interaction when a link is available, and queues messages when it is not
- Has simple setup options for security, and can easily work through firewalls
- Enables up-to-the-minute information, such as stock prices, to be received automatically from servers

MQSeries Everyplace is supported on the following platforms:

- EPOC
- Palm OS
- Pocket PC (Windows CE)
- Windows 95
- Windows 98
- Windows NT
- Windows 2000

MQSeries Everyplace consists of Java and C components enabling developers to create an MQSeries Everyplace gateway and client on a variety of devices and platforms. Java support is currently offered for EPOC, Palm OS, and Windows CE devices.

1.3.9 Location Based Services

Everyplace Location Based Services (LBS) provides the physical location (longitude/latitude, city/state/country) of the end user, and supplies location-based information or content. For example, this information could be used by the application to point a customer to the nearest automatic teller machine or gas station. It could also be used as context information to tailor the content to be delivered.

Location Based Services is typically deployed in a carrier/service provider environment for business reasons (cellular service providers/carriers typically do not release their databases to outside entities, but might grant access to individual customer records if the customer grants permission). Location based policies are two fold:

- Administrators can determine what applications are location-based (determined by registering location-based applications).
- Depending on which applications are location-based, an administrator or user can enable his/her location information to be available to enabled location-based applications. The user must also enable each location-based application to transmit their specific location information. If the user does not enable an application, the location information is not provided.

Location Based Services is shipped with and requires WebSphere Everyplace Server Service Provider Offering. The Suite Manager in WebSphere Everyplace Server Service Provider Offering is the administration tool for Location Based Services, and Everyplace Setup Manager in WebSphere Everyplace Server Service Provider Offering is the tool for installing Location Based Services. However, you must install the following software components:

- **SignalSoft Corporation’s Wireless Location Services** - IBM is a Wireless Network Infrastructure Partner for SignalSoft Corporation, which provides the third-party application software required to use Location Based Services. For more information about SignalSoft, please visit:

  `http://www.signalsoftcorp.com`

- **Tivoli SecureWay Policy Director** - Policy Director is used to manage privacy settings and authorize applications to use a subscriber’s location-based information.

1.3.10 Intelligent Notification Services

Everyplace Intelligent Notification Services delivers messages to pervasive users based on the users' preferences and subscriptions. For example, a user can tell Everyplace Intelligent Notification Server to send them the URL of any Web-based news article published with "wireless computing" in the headline. The user can also specify message-sending behaviors based on the urgency of the message. For example, if the message is marked FYI, send it to e-mail. If the message is marked urgent, send it via Sametime Everyplace instant messaging.

Everyplace Intelligent Notification Services is highly customizable. In fact, it is necessary to customize and develop code on top of Everyplace Intelligent Notification Services in order to implement a viable notification system solution.

Intelligent Notification supports the following notification methods:

- Lotus Sametime instant messaging
- Wireless Application Protocol (WAP)
- Simple Messaging Service (SMS)
- SMTP e-mail
- Push e-mail

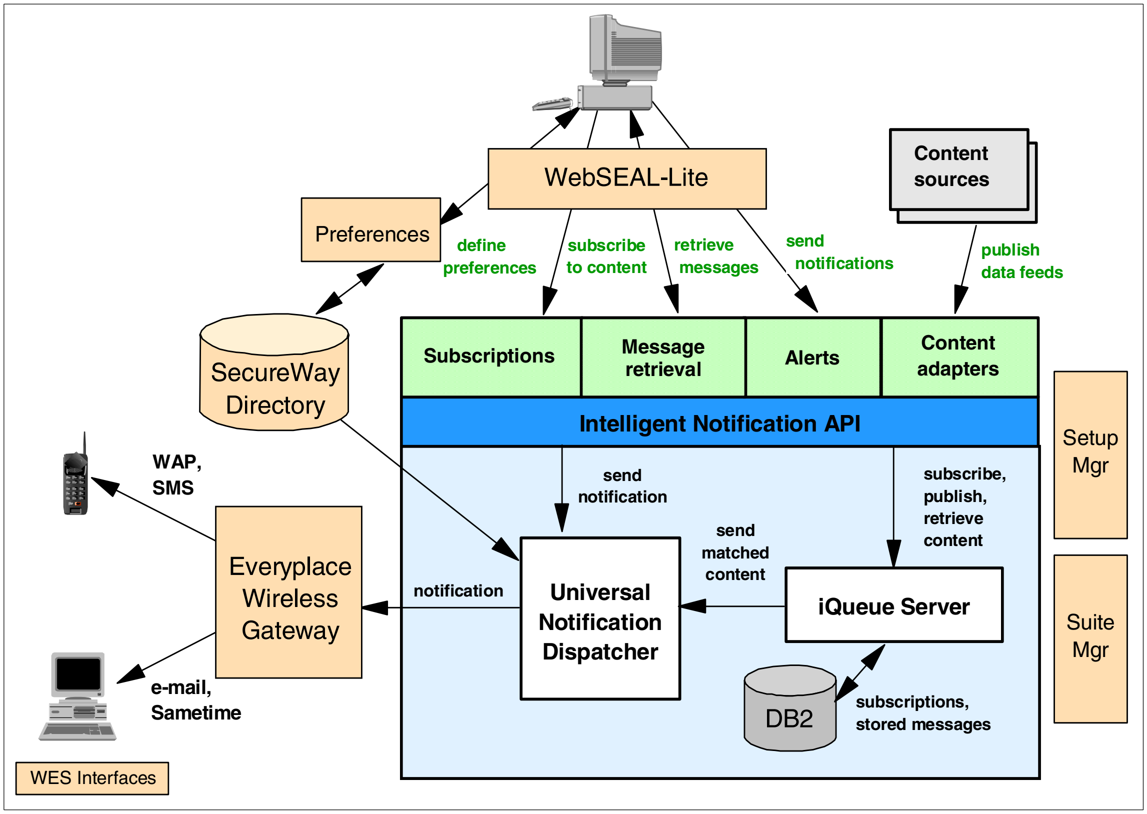


Figure 1-7 Functional overview of Intelligent Notification Services

Everyplace Intelligent Notification Services consists of the following core components:

- **Trigger management:** Manages user subscriptions, receipt of published content, and trigger activation when content matches a subscription.
- **Universal Notification Dispatcher:** Sends messages to the user by various means, including instant messaging, WAP, phone, and e-mail.

Figure 1-7 shows the functional overview of the Intelligent Notification Services architecture. A content adapter collects data from network content feeds and delivers the data to the iQueue Server. The iQueue Server matches published content to user content topics and filters and sends it to the Universal Notification Dispatcher. The Universal Notification Dispatcher dispatches messages to user notification mechanisms such as e-mail, Sametime instant messaging, WAP, and SMS via the Everyplace Wireless Gateway and device-specific Gateway Adapters.

A JSP-generated subscription form allows users to specify content topics and filters. A User Preferences form allows users to specify user, group, and device information. The iQueue Server uses the content topics and filters to filter content. The iQueue Server passes the messages in Notification Markup Language format to the Universal Notification Dispatcher. The Universal Notification Dispatcher gateway adapters format and distribute the messages for the selected notification mechanisms. The Universal Notification Dispatcher then uses the user, group, and device information to decide how and where the messages should be distributed.

Details on how to write applications that take advantage of Intelligent Notification Services may be found in the IBM Redbook *Enterprise Wireless Applications using WebSphere Everyplace Server Offerings*, SG24-6519.

1.3.11 Tivoli Personalized Services Manager

Tivoli Personalized Services Manager is an infrastructure for enabling and managing the delivery of Internet-based services and the customers who use those services. Tivoli Personalized Services Manager allows service providers to centrally manage subscribers and devices. 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. Tivoli Personalized Services Manager also allows service providers to build, manage and market services offerings, and to maintain an integrated customer/service offering database.

An overview of the architecture for Tivoli Personalized Services Manager is shown in Figure 1-7 on page 34.

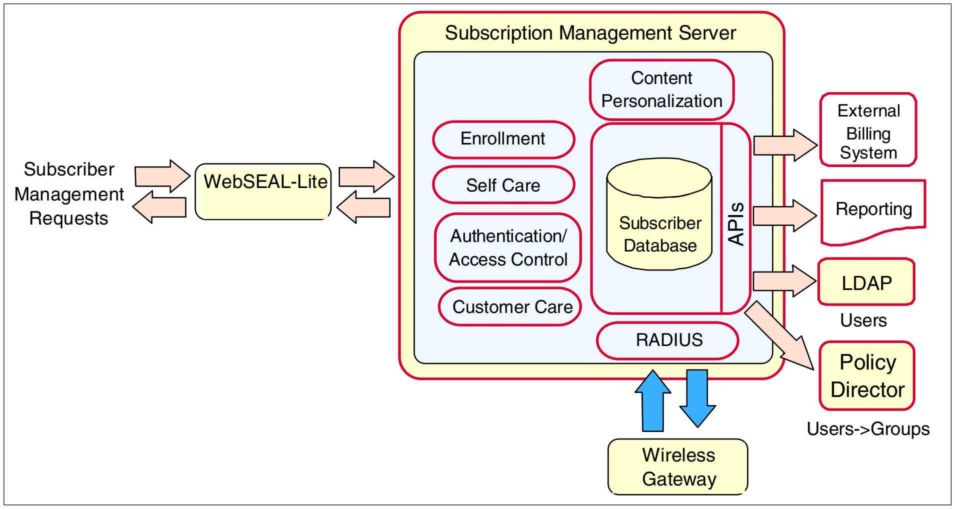


Figure 1-8 Tivoli Personalized Services Manager overview

Tivoli Enrollment and Self Care

Tivoli Enrollment and Self Care components allows customers to manage their own subscription information, and are sometimes referred to together as Subscriber Management. The Enrollment and Self Care components provide an easy-to-use customizable interface to create and modify personal information regarding their subscribed services. The components consist of JSPs, Servlets, and associated image and configuration files. They are dependent on the Tivoli Personalized Services Manager database, a Web server, and authentication. The Enrollment and Self Care components run in an WebSphere Application Server. Each of these components can be customized to a specific look and feel, as well as personalized for existing customers.

Self Care allows subscribers to modify the portal pages for their mobile devices, modify some of their profile data, including address and telephone data, billing plan, payment method, and the registering for Tivoli Personalized Services Manager content. Generally, the tasks displayed by Self Care are a subset of those displayed by Customer Care.

The Enrollment component enables users to enroll and subscribe to new services through the Internet. It also facilitates the capture of a new user’s name, billing information, deal, user ID and password, or any other information needed to subscribe to a new service. If Everyplace Server is configured to use Policy Director, user information may be added to the LDAP directory during enrollment. When Policy Director is used, a Policy Director user is created and added to the appropriate Policy Director group.

Tivoli Customer Care

The Customer Care component of Tivoli Personalized Services Manager and is used by Customer Service Representatives to provide account management for subscribers. The Customer Care component can be used to enroll new subscribers into the system, search for and modify a subscriber's profile and account information. It also provides linkage to reports concerning subscriber's profiles and account usage information. During enrollment and update, any integrated external systems, such as billing or mail systems, can be automatically updated.

Similar to the Enrollment and Self Care components, Customer Care consists of customizable JSPs, Servlets and associated files.

Tivoli Device Manager Server

Tivoli Device Manager Server is an infrastructure to manage pervasive devices both wireless and wired, and to provide for remote configuration of the devices. Remote configuration would consist of installing or updating device software, the software configuration, or the distribution of a personalized rest page.

Device Manager Server consists of management operations, graphical interfaces to configure the devices, device-specific plug-ins, and management APIs. An agent application running on the device is used to communicate with the Tivoli Device Manager Server. Currently Tivoli Device Manager Server supports Palm OS, Windows CE, and NetVista IAD (Neutrino).

Packaged with the Service Provider Offering, Device Manager Server provides integration with Tivoli Personalized Services Manager. The enrollment, Self Care and Customer Care interfaces can be used to manage a device through a Web browser. Some examples are:

- The Customer Care interface allows the service provider customer service representative to set up and help configure the device with the user via the telephone.
- The Self Care interface allows a device owner to configure and select devices through a Web browser.
- A Device Management servlet allows a pervasive device to communicate with the Device Manager Server directly.

Device Manager Server can also be used in a stand-alone environment with WebSphere Everyplace Server Enable Offering. Since users are enrolled outside of WebSphere Everyplace Server Enable Offering, Device Manager Server can be configured to use the customer’s LDAP directory server.

Packaged with Service Provider Offering, Tivoli Device Manager Server provides integration with Tivoli Subscription Manager. It also allows WebSEAL-Lite to provide authentication for Device Manager Server servlets.

1.3.12 Everyplace Setup Manager

The Everyplace Setup Manager provides an easy-to-use Java-based graphical interface to install, configure, and integrate the WebSphere Everyplace Server components that a customer chooses. The graphical interface provides a single look and feel through the installation of the various WebSphere Everyplace Server components. Setup Manager collects information about the installation, configuration, and customization up front. It then performs the installation tasks in a silent, express mode. Throughout the installation process, The Setup Manager ensures that the prerequisites are met before installing each components. Setup Manager wizards allow information to be initialized in XML files. This allows the reuse of configurations for subsequent installations.

1.3.13 Everyplace Suite Manager

The 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 the SecureWay Directory (LDAP) directory.

Everyplace Suite Manager allows users to perform the following management tasks from one centralized location:

- Change initial Everyplace Server configuration settings after Everyplace Server installation is complete
- Perform Everyplace Server administration tasks, including managing the Active Session Table Server
- Launch the administration consoles of individual Everyplace Server components

1.3.14 IBM SecureWay Directory

The SecureWay Directory is a Lightweight Directory Access Protocol (LDAP) server that runs as a stand-alone daemon. It is based on a client/server model that provides client access to an LDAP directory server over TCP/IP. LDAP is only a protocol that runs over TCP/IP. The LDAP protocol standard not only includes low-level network protocol definitions, but also data representation and handling capability. A directory that is accessible through LDAP is, therefore, commonly referred to as an LDAP directory. It must be noted and understood, however, that the LDAP standard does not define how the data is actually stored in the directory.

IBM's SecureWay Directory ships with WebSphere Everyplace Server and provides an easy way to maintain directory information in a central location for storage, updating, retrieval, and exchange.

LDAP overview

LDAP directory stores and organizes data structure as "entries". A directory entry usually describes an object such as a person, a server, etc. Each entry has a unique distinguished name (DN). The DN consists of a sequence of parts called relative distinguished name (RDNs) separated by commas. The entries can be arranged into a hierarchical tree-like structure based on their distinguished names. This tree of directory entries is called the Directory Information Tree (DIT). The sequence of RDNs making up a DN names the ancestors of a directory entry up to the root of the DIT. Each RDN is composed of an attribute value from the directory entry. For example, the following DN:

cn=Joe Smith,ou=Sales,o=QuickClaim,c=US

represents a directory entry for a person with the common name (cn) Joe Smith, under the organizational unit (ou) Sales, in the organization (o) QuickClaim, in the country (c) US. Each RDN in a DN corresponds to a branch in the DIT leading from the root of the DIT to the directory entry. Each RDN is derived from the attributes of the directory entry. Figure 1-9 shows a sample LDAP directory tree:

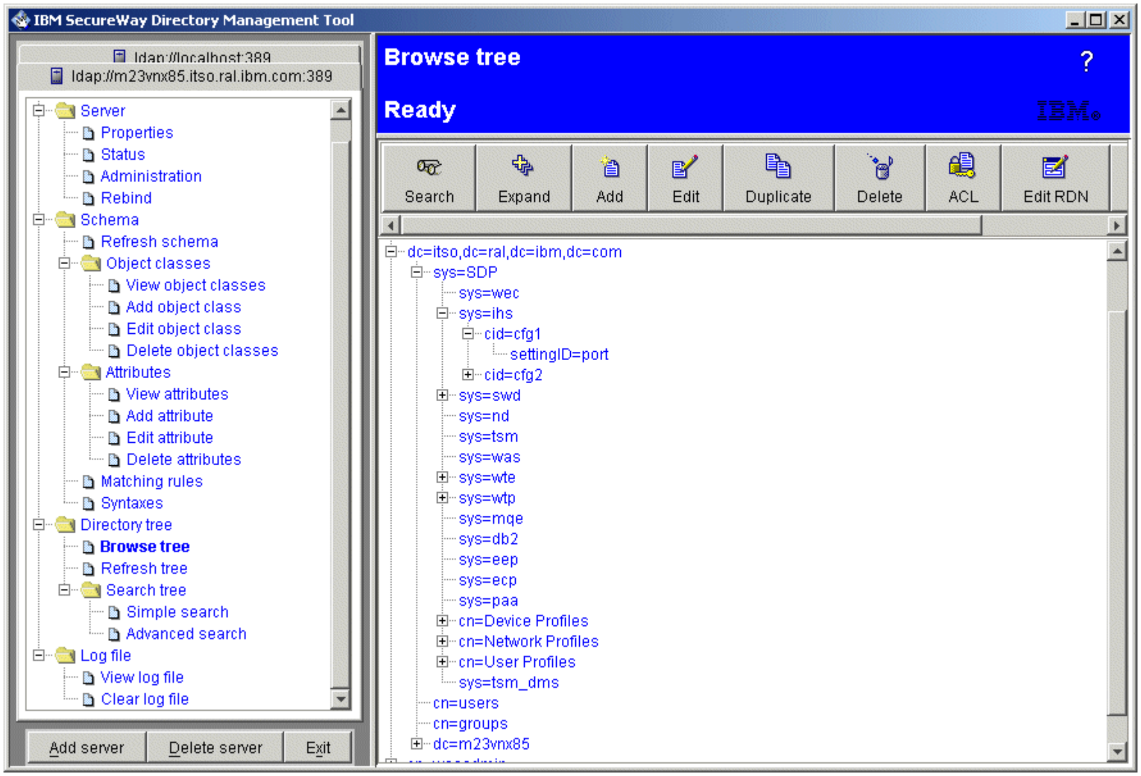


Figure 1-9 Sample LDAP DIT

A directory entry is an instance of an object class depicted in the left pane of Figure 1-9. An object class provides general description of an entry, similar to template. Each entry contains one or more attributes that describe the entry. Each attribute has a type and a value. For example, the `inetOrgPerson` object class has `uid` and `givenname` as optional attributes. The directory entry for an `inetOrgPerson` can have a “`uid`” attribute with the value of “JoeSmith”. Figure 4-11 on page 172 shows the `inetOrgPerson` object class. Schema defines what object classes are allowed where in the directory, what attributes they must contain, what attributes are optional, and the syntax of each attribute.

Attribute types and syntaxes are used to build schema that describe object classes. A schema lists what attributes a directory entry must or may have. Every directory entry has an `objectclass` attribute that lists one or more schema that describe the entry. For example, a directory entry could be described by the object classes `inetOrgPerson` and `ePerson`. If an entry’s “`objectclass`” attribute includes the value “`extensibleObject`”, it can contain any attribute.

Many schema and attributes commonly accessed by directory clients are already defined by X.500. For instance, attributes such as `cn`, `uid` and `postalAddress` are defined, and object classes such as `country`, `groupOfNames`, `inetOrgPerson`, `ePerson` are also defined. RFC 2256, A Summary of the X.500 User Schema for use with LDAP V3, provides an overview of those attribute types and object classes that LDAP directory servers should recognize.

Object class can be subclassed from other class. One special object class, called `top`, has no super class. The `top` class includes the mandatory `objectClass` attribute. Each directory entry has an `objectClass` attribute and one of the values must be either `top` or `alias`. `Alias` is used if the entry is an alias for another entry; otherwise `top` is used.

1.3.15 Active Session Table Server

The Active Session Table contains information about users who are currently connected to the WebSphere Everyplace Server domain. The Active Session Table server manages the Active Session Table and stores its contents in high-speed cache. This replaces the capability previously provided by the Tivoli Internet Services Manager product.

- The Active Session Table client communicates with the Active Session Table Server, using one or more standard TCP/IP connections and continues to provide service for each connection as long as the client remains logged on.
- The IP address of each WebSphere Everyplace Server client that attempts to connect to the Active Session Table Server must appear in the accept list of the Active Session Table Server. The accept list is one of the Active Session Table Server's configuration parameters. The Active Session Table server refuses connections from clients who are not in the accept list. This list is built at installation time. If you add clients after installation you must also update this list.
- The WebSphere Everyplace Server client can send one or more requests on a single connection; however, the Active Session Table Server responds to each request in the order in which they are received.
- The Active Session Table Server performs field validation on each request and, if an error is detected, returns a message describing the first error detected.

1.3.16 Sametime Everyplace

Lotus Sametime Everyplace extends the collaborative power of Sametime messaging to the mobile business user. Enterprises can have both desktop and mobile user awareness and instant messaging integrated together in a secure, reliable environment. This will enable business users to have accurate information about their colleagues' availability to communicate and choose the appropriate vehicle to communicate with each other (phone, e-mail or instant messaging).

Lotus Sametime Everyplace provides the following key features:

- Online awareness:

  Users can see when others are available from the desktop and mobile device.

- One-to-one and one-to-many real-time chat:

  Allows chat between multiple mobile and non-mobile users, interacting through different devices.

- Instant messaging between Sametime desktop and Sametime Everyplace mobile users:

  Allows Sametime desktop users to send messages to mobile users and vice-versa.

- Auto-logon/logoff:

  Allows users to predefine when they are available when mobile.

- User-configurable mobile profile:

  Users can modify preferences from WAP browser or Web browser.

Sametime Everyplace will work with any mobile device that has an Openwave or WAP 1.1 browser. Sametime Everyplace will also provide SMS messaging to one and two-way paging devices or SMS-enabled phones that are auto-logged on to Sametime Everyplace.

In order to deliver instant messages from a Sametime Connect client (PC desktop) to a Mobile Sametime user who is not in an active state (active state defined as auto logged on, logged on from the Web, or left the application without logging off), Domino Everyplace SMS Server, which is included with Sametime Everyplace, sends this text to the mobile device. This is because WAP devices do not currently support "push" transmissions. If the Sametime mobile user initiates the instant messaging chat, then Domino Everyplace SMS does not play a part of the two-way chat conversation; rather the chat is facilitated over WAP.

Currently, Sametime Everyplace has been tested on the following devices:

- Ericsson R380
- Nokia 6210
- Nokia 9110
- Nokia 7110
- Mitsubishi T250
- Sprint Touch Point 250
- Sprint Touch Point 3000
- i-Book and AU browsers in Korea

Note: Sametime Everyplace Server is shipped with the IBM WebSphere Everyplace Server Service Provider Offering as a technology first look. It is not an integral part of WebSphere Everyplace Server and is not installed using the Everyplace Server installation program. It must be installed separately. See the documentation on the Sametime Everyplace CD for installation instructions.

2

Differentiating the offerings

In Chapter 1 we briefly mentioned the three Everyplace offerings:

- WebSphere Everyplace Access
- IBM WebSphere Everyplace Server Enable Offering
- IBM WebSphere Everyplace Server Service Provider Offering, available in two configurations

The goal of this chapter is to help architects and integrators understand which offering is best suited for particular sets of needs. To achieve this goal, we provide in this chapter:

- A discussion of the common components
- A discussion of the unique features of each offering
- Comparisons of:
  - The intended use of each of the offerings (the target markets, the business need that it addresses, its key strengths, etc.)
  - Functional characteristics and components

2.1 Common features

The three Everyplace Server offerings have in common certain base components, certain platforms on which they run, and certain devices they support as clients.

2.1.1 Common components

The following components, to which we will refer hereafter as the common components, are in all three Everyplace offerings:

- WebSphere Application Server Advanced Edition
- Java Development Kit (JDK)
- IBM HTTP Server (based on Apache HTTP Server), licensed for the private use of Everyplace -- not to be used as a general Web server
- IBM WebSphere Transcoding Publisher

2.1.2 Common support for devices

Since all the Everyplace offerings include WebSphere Transcoding Publisher, they all support devices that can use any of the following: XML, HTML, WML, CHTML, i-mode, VoiceXML, Palm OS HTML, and WBM. In general this includes:

- Internet-capable mobile phones
- Internet-capable PDAs
- i-mode phones

> **Note:** For a detailed discussion of i-mode, please read Veronika Meglar's Redpaper, *The Semi-Walled Garden: Japan's "i-mode Phenomenon",* PEDP0166 available from the ITSO Web site.

2.1.3 Everyplace Server components

In addition to the common components, both Enable Offering and Service Provider Offering include the following components[1]:

- WebSphere Edge Server, which includes, and counts as, two components:
  - Load Balancer (formerly known as and sometimes still referred to as Network Dispatcher)

[1] Everyplace Access does not bear the Everyplace **Server** brand, and does not include the Everyplace Server components.

  - Caching Proxy (formerly known as and sometimes still referred to as Web Traffic Express)
- Everyplace Cookie Proxy (for supporting i-mode in Japan)
- Tivoli Device Manager Server
- MQSeries Everyplace

2.1.4 Common platforms

All three Everyplace offerings support IBM AIX and Sun Solaris platforms.

2.2 Distinctive features

In this section we discuss those aspects of each Everyplace offering that distinguish that offering from the others.

2.2.1 Everyplace Access -- entry level and voice enablement

WebSphere Everyplace Access for Multiplatforms V1.1 is the entry level offering for enterprises and institutions wishing to extend access to their e-business data. Because it has the fewest components, this offering has the smallest requirement for skills, and is the easiest to integrate.

Unique components

In addition to the “Common components” on page 46, Everyplace Access includes the following products that are not in the other Everyplace offerings:

- **IBM WebSphere Voice Server** -- works with a Voice over IP (VoIP) protocol gateway and existing Web infrastructure to help businesses leverage their investments in Web and call center technologies. It uses VoiceXML, Java and H.323 (Voice over IP) standards to allow delivery of voice applications for the Internet. WebSphere Voice Server includes:
  - A voice browser that interprets VoiceXML markup, which complements WAP to allow the delivery of voice applications for the mobile Internet
  - IBM ViaVoice Standard Edition speech recognition and Text-to-Speech engines
  - WebSphere Voice Server SDK (Software Development Kit)
  - WebSphere Studio, Professional Edition
  - VoiceXML tools

  - WebSphere Application Server Standard Edition (which you probably will not use, because Everyplace Access includes WebSphere Application Server Advanced Edition).

  For additional information on WebSphere Voice Server 1.5, refer to Software Announcement 201-146, dated May 8, 2001, or visit the IBM Web site at `http://www-4.ibm.com/software/speech/enterprise/ep_1.html`.

- **WebSphere Studio, Advanced Edition**, Version 3.5 -- For more information, please see Software Announcement 200-254, dated July 25, 2000, or visit `http://www-4.ibm.com/software/webservers/studio/`.
- **VisualAge for Java, Enterprise Edition**, Version 3.5 -- For information, refer to Software Announcement 200-236, dated July 25, 2000, or visit `http://www-4.ibm.com/software/ad/vajava/`.

Platforms

Of the WebSphere Everyplace offerings, Everyplace Access has the fewest components, making it easier to support the most platforms. In addition to the common platforms, Everyplace Access supports Linux, Windows NT and Windows 2000.

Note: WebSphere Voice Server is supported only on AIX and Windows NT.

Supported devices

In addition to the common devices, Everyplace Access supports voice-enabled devices, including regular telephones and screen phones.

Targeted uses

Everyplace Access is an ideal solution for companies that need to:

- Implement sales force automation or enable mobile employees
- Increase access methods to e-business applications
- Deliver time-sensitive e-business content
- Extend business operations via automated self-service
- Replace old/existing voice response systems

2.2.2 WebSphere Everyplace Server Enable Offering -- for integrating

WebSphere Everyplace Server Enable Offering Version 1.1 integrates into the enterprise's current IT environment by providing integration options that support existing:

- User management

- Authentication
- Access control (authorization)
- Directory services

Figure 2-1 on page 50 shows the functional contents of Enable Offering. Table 2-1 on page 51 lists the Enable Offering features, the function that each feature provides, and the benefit of having that function.

Unique components

In addition to the components listed in “Common components” on page 46 and “Everyplace Server components” on page 46, WebSphere Everyplace Server Enable Offering has unique pluggable options, and the following components:

- Suite Installer
- Everyplace Administration Console

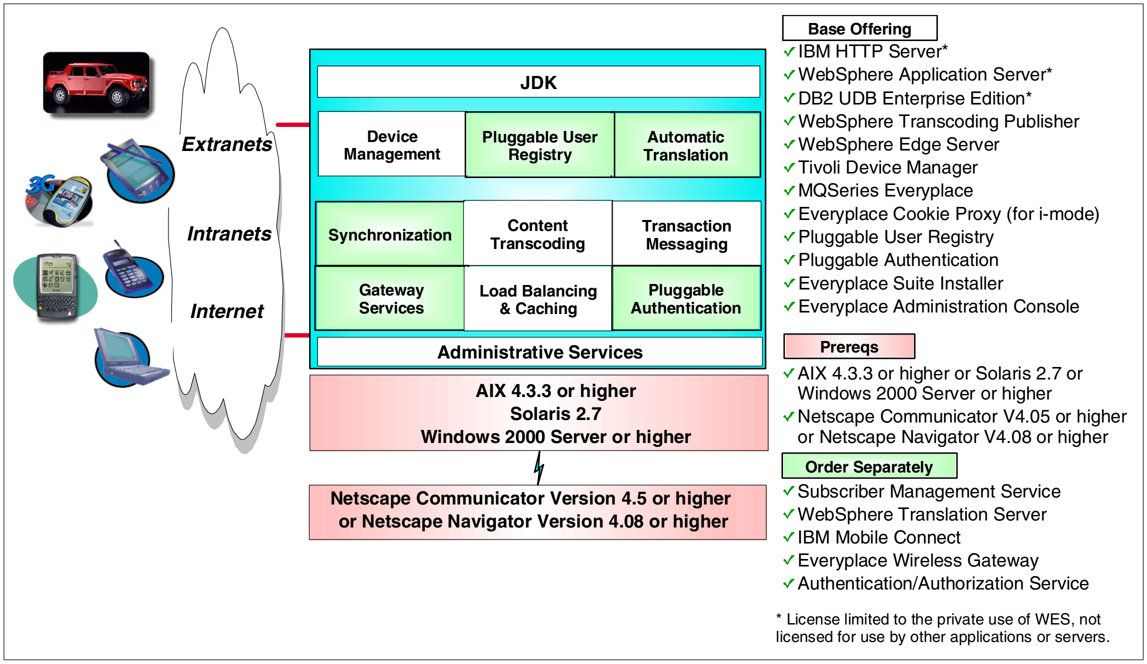
Pluggable options

What makes WebSphere Everyplace Server Enable Offering unique is its use of *pluggable options*. A pluggable option is a pre-coded interface to a specific function or product. WebSphere Everyplace Server Enable includes the following pluggable options:

A *pluggable option* is a pre-coded interface to a specific function or product.

- User directory (either IBM SecureWay Directory or Netscape Directory Server), Lotus Domino server, or Microsoft Active Directory
- Third-party authorization server (either Tivoli SecureWay Policy Director, or Netegrity SiteMinder)
- WebSphere Translation Server for translating text from one national language to another (for example, English to Chinese). For more information about WebSphere Translation Server, please see Software Announcement 201-031, or visit http://www-4.ibm.com/software/speech/enterprise/ep_8.html

.



Base Offering

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

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/Authorization Service

* License limited to the private use of WES, not licensed for use by other applications or servers.

Figure 2-1 Enable Offering contents

Platforms

Enable Offering supports AIX, Solaris, and Windows NT, and Windows 2000.

Targeted users

WebSphere Everyplace Server Enable Offering is most appropriate for (but not limited to):

- Application developers
- Web integrators
- Value Added Resellers (VARs)
- System integrators
- Medium to large companies and institutions.

Table 2-1 Enable Offering features, functions and benefits

Feature	Function	Benefit
Pluggable User Subscriber Management	Facilitates integration with SecureWay Directory and with Netscape Directory Server	Leverages existing infrastructure
Pluggable Authentication Services	Facilitates integration with Tivoli SecureWay Policy Director and Netegrity SiteMinder Authentication Server	Leverages existing infrastructure
Content Adaptation	Transcodes content from one format to another	Reduces development and maintenance effort of delivering information to pervasive devices
Secure Messaging	Encrypted asynchronous message transactions with assured delivery	Assures message delivery, regardless of connectivity reliability
Device Management	Software for managing services for cell phones, PDAs, and other pervasive devices	Simplifies and centralizes management of enterprise devices
Load Balancing	Rules engine routes IP traffic based on content	Optimizes performance, availability and scalability
Caching Proxy	Extensible forward and reverse proxy	Reduces network load, improves performance and response time for delivering content
Cookie Proxy	Support for i-mode devices	Broadens device support for mobile devices in Japan; streamlines transactions
Administration Console and Suite Installer	Unified Administration and Installation console	Simplifies installation and administration of components
Multi-platform: Intel and RISC processors	All components supported on AIX, Solaris, Windows NT and Windows 2000	Offers choice between UNIX and Windows platforms

2.2.3 WebSphere Everyplace Server Service Provider Offering

Service Provider Offering includes and integrates all the middleware infrastructure needed to connect, adapt, manage, transform, and scale today's Web applications and legacy data into tomorrow's pervasive applications. Whereas Enable Offering *integrates with existing* LDAP-compliant user directories and authentication servers, Service Provider Offering *includes* an LDAP-compliant user directory server (IBM SecureWay Directory), an authentication/authorization proxy (WebSEAL-Lite) and has bundled with it an authorization server.

Service Provider Offering also includes as options a gateway to carrier networks and a synchronization manager for keeping PDA-based applications and data consistent with their server-based counterparts. Service Provider Offering has integration support (pluggable options) for instant messaging, voice command/recognition, and collaboration.

WebSphere Everyplace Server Service Provider Offering is available in two configuration options:

- WebSphere Everyplace Server Service Provider Offering
- WebSphere Everyplace Server Service Provider Base Offering

  The Base Offering configuration does not include the Everyplace Wireless Gateway function nor the Everyplace Synchronization Manager function. Otherwise, it is the same as Service Provider Offering. This configuration is intended for situations where either your environment already has gateways to all the carrier networks that it needs, or there is no need for a component to synchronize PDA-based applications and data with their server-based counterparts, or both.

  You can add the Everyplace Wireless Gateway component and/or the Everyplace Synchronization Manager component after installing the Base configuration. However, if you want both of these components, you would be better served in terms of installation and configuration to just order the full Service Provider Offering configuration.

Components unique to Service Provider Offering

Service Provider Offering is a comprehensive and complex offering comprising 18 components (depending on how you count them). It includes the four common components (see “Common components” on page 46), the six Everyplace Server Components (see “Everyplace Server components” on page 46), and the following unique components:

- Tivoli Personalized Services Manager -- Subscription management components are added, including Customer Care, Self Care, Enrollment, and others.
- Active Session Table Server -- a mechanism for sharing user session information between WebSphere Everyplace Server components (for example, between Everyplace Wireless Gateway and WebSEAL-Lite).
- Everyplace Wireless Gateway (Everyplace Wireless Gateway) -- allows users to get to your services over any network, wired or wireless. The Everyplace

Wireless Gateway is designed to work in the WebSphere Everyplace Server Service Provider Offering environment. It updates the Active Session Table Server with user session information.

- WebSEAL-Lite -- an authentication/authorization reverse proxy. It replaces the Everyplace Authentication Server in WebSphere Everyplace Suite V1.1. Tivoli wrote WebSEAL-Lite specifically for the WebSphere Everyplace Server Service Provider Offering environment. WebSEAL-Lite differs from WebSEAL in Policy Director in that WebSEAL-Lite is able to accept the authentication performed by Everyplace Wireless Gateway, whereas WebSEAL would force another authentication. WebSEAL-Lite is not available outside of Service Provider Offering.
- Tivoli SecureWay Policy Director -- although WebSphere Everyplace Server does not have Policy Director integrated into the Setup Manager and Suite Manager, WebSphere Everyplace Server has integration with Policy Director for authentication, and for checking access levels and privacy settings (authorization).
- IBM SecureWay Directory -- the user registry, which complies with the LDAP V3 specification
- Everyplace Intelligent Notification Services -- enables you to notify subscribers of specific events, such as (but not limited to):
  - A gate change for or arrival of a flight
  - When a particular sports team scores
  - The availability of a specific application or printer on the network
  - When their vehicle or part is ready
  - Deal alerts

  Intelligent Notification Services is available only in the WebSphere Everyplace Server Service Provider Offering environment.
- Everyplace Location Based Services -- if the user’s privacy restrictions permit it, Location Based Services integrates with a location-based application server[2] to deliver location-specific (latitude and longitude) information to the user’s mobile device. Location Based Services is only available in the WebSphere Everyplace Server Service Provider Offering environment.
- Setup Manager -- the tool for installing the WebSphere Everyplace Server Service Provider Offering components with proper configurations for working in the WebSphere Everyplace Server environment.

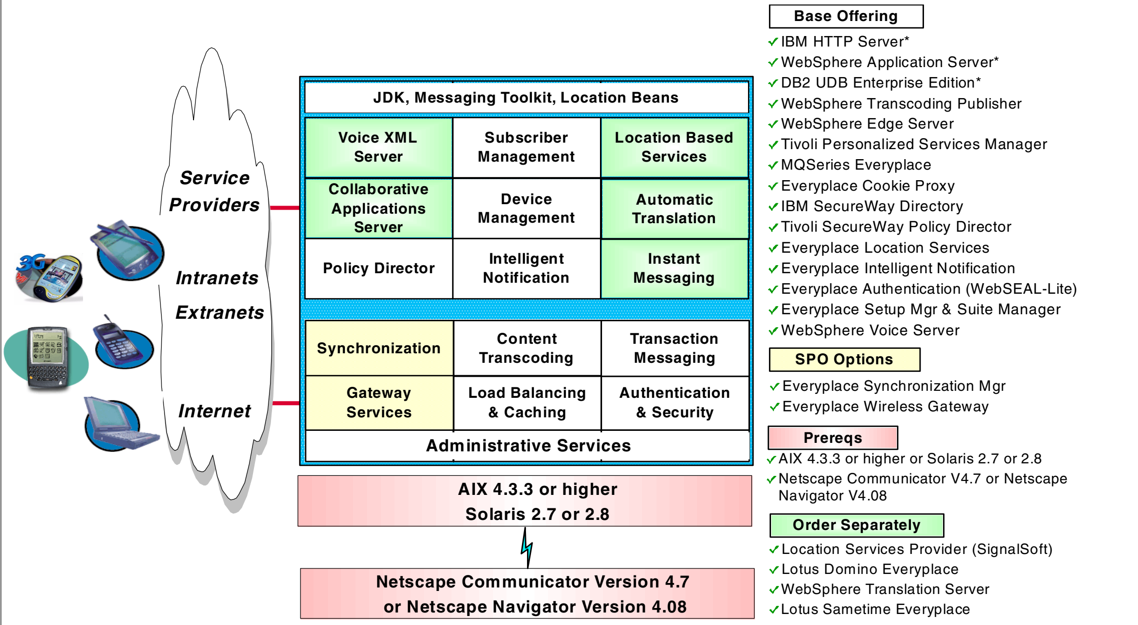
[2] Currently, the only location-based application servers WebSphere Everyplace Server Location Based Services supports are the SignalSoft Wireless Location Services. For more information, please visit `http://www.signalsoftcorp.com`

- Suite Manager -- the tool for centrally administering the WebSphere Everyplace Server Service Provider Offering components.

> **Note:** The Setup Manager and the Suite Manager are significant pieces of work that represent a considerable portion of the total implementation and integration effort. While they are not of major significance to architects, these tools are tremendously valuable to integrators and installers.

- Everyplace Synchronization Manager -- synchronizes PDA data with server-based PIM and relational data. (Included in the CD-ROM package, but requires entitlement.)

Figure 2-2 illustrates the components that WebSphere Everyplace Server Service Provider Offering comprises.


Service Providers
Intranets
Extranets
Internet
3G
JDK, Messaging Toolkit, Location Beans
Voice XML Server
Subscriber Management
Location Based Services
Collaborative Applications Server
Device Management
Automatic Translation
Policy Director
Intelligent Notification
Instant Messaging
Synchronization
Content Transcoding
Transaction Messaging
Gateway Services
Load Balancing & Caching
Authentication & Security
Administrative Services
AIX 4.3.3 or higher
Solaris 2.7 or 2.8
Netscape Communicator Version 4.7
or Netscape Navigator Version 4.08
Base Offering
✓IBM HTTP Server*
✓WebSphere Application Server*
✓DB2 UDB Enterprise Edition*
✓WebSphere Transcoding Publisher
✓WebSphere Edge Server
✓Tivoli Personalized Services Manager
✓MQSeries Everyplace
✓Everyplace Cookie Proxy
✓IBM SecureWay Directory
✓Tivoli SecureWay Policy Director
✓Everyplace Location Services
✓Everyplace Intelligent Notification
✓Everyplace Authentication (WebSEAL-Lite)
✓Everyplace Setup Mgr & Suite Manager
✓WebSphere Voice Server
SPO Options
✓Everyplace Synchronization Mgr
✓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
✓Location Services Provider (SignalSoft)
✓Lotus Domino Everyplace
✓WebSphere Translation Server
✓Lotus Sametime Everyplace

* License limited to the private use of WES, not licensed for use by other applications or servers.

Figure 2-2 Service Provider Offering Content

Note: The WebSphere Everyplace Server Service Provider Offering team put together a very useful documentation tool that is included on the first CD-ROM in the package, called the InfoCenter. The InfoCenter provides a single entry point into the documentation for all the WebSphere Everyplace Server Service Provider Offering components. The InfoCenter is also on the IBM WebSphere Everyplace Server Web site at
`http://www-3.ibm.com/pvc/products/wes_provider/infocenter/index.html`

Table 2-2 Service Provider Offering features, functions and benefits

Feature	Function	Benefits
Intelligent notification services	User configurable real-time event notification	Keeps users up-to-date when events occur and/or when content is available
Location based services	Privacy-controlled capability to dynamically determine a user's device position	Delivers information to users when and where they need it
Pluggable instant messaging	Technology first look of integration with Lotus Sametime Everyplace	Opens up additional channels for communication by supporting instant awareness and real-time communications
Security	End-to-end encryption and authentication and integration with Policy Director for enhanced authorization and access control	Complete infrastructure for deploying security-rich applications
Content adaptation	Transcodes content from one format to another	Reduces effort to deliver information to pervasive devices
Pluggable voice integration	Extends applications with integrated voice technologies	Offers hands-free access to applications
Secure messaging	Asynchronous transaction messages with assured, secure, message delivery	Assures message delivery regardless of connectivity, once and only once
Load balancing	IP traffic load balancing, rules engine, with content-based routing	Optimizes performance, availability and scalability
Intelligent gateway	Optimizes data transmission across networks	Makes network usage more efficient, allows for enterprise-wide as well as service provider-sized deployments, helps reduce usage-based network charges
Pluggable collaborative applications	Integration with Domino application adapters in Domino Everyplace (DEAS)	Access to collaborative application content

Pluggable Options

WebSphere Everyplace Server Service Provider Offering has pluggable options for integrating with:

- Lotus Sametime Everyplace for instant messaging
- WebSphere Voice Server for voice command/recognition
- Lotus Domino and Lotus Domino Everyplace for collaboration applications
- WebSphere Translation Server

Platforms

Some of the Service Provider Offering components do not support Windows or Linux, but all support IBM AIX and Sun Solaris. Therefore, Service Provider Offering is only supported on common platforms, AIX and Solaris.

Targeted uses

Service Provider Offering is targeted for four major industry segments:

- Wireless service providers (Wireless Carriers, Wireless Application Service Providers, Wireless Internet Service Providers)
- Large financial enterprises
- Large retail enterprises
- Large travel and transportation enterprises

2.3 Comparisons

Having looked at each offering individually, we now compare and contrast the Everyplace offerings. We compare their business objectives and benefits, then look at how the offerings compare in what functions they offer.

2.3.1 Comparison of business characteristics

For each offering, Table 2-3 illustrates:

- Target customers -- this is a list of the primary customers anticipated to use this offering
- Business needs -- these are the management-level mobile e-business objectives we are trying to address
- Challenge -- at a high level, the technology challenge to delivering on the business need
- Key strengths of the offering

- Platforms -- operating system platforms the offering supports
- Optional and complementary products that might enhance the solution or simplify the implementation

Table 2-3 Non-functional comparison of the offerings

	Access Offering	**Enable Offering**	**Service Provider Offering**
Target Customers	Enterprises, ISPs, ASPs, Lines of Businesses (LOBs)	Enterprises, Independent Software Vendors (ISVs), Global System Integrators, LOBs	Telephone companies, ISPs, ASPs, Large Enterprises
Business Need	Extend e-business content and applications for wireless and voice interaction	Extend business processes to the mobile work force, while preserving any existing e-business infrastructure	Generate new sources of revenue by building and deploying new value-added services to the mobile marketplace
Challenge	How to allow voice interaction and reach wireless devices?	How to extend e-business to wireless while taking advantage of existing IT services?	How to rapidly create new revenue- generating services leveraging existing and new data sources (contextual notification)?
Key Strengths	Extends existing content Speeds time to market Includes and integrates with WebSphere Application Server for security and scalability	Preserves and expands existing infrastructure by taking advantage of existing user management and authentication Future-proof (adapts to rapidly changing networks, devices and application requirements)	Supports any application, any device and any network Future-proof, (adapts to rapidly changing networks, devices and application requirements) Integrated package
Optional and Complementary Products	Everyplace Wireless Gateway WebSphere Edge Server WebSphere Translation Server	Everyplace Wireless Gateway IBM Mobile Connect Lotus Domino Everyplace	WebSphere Voice Server Lotus Sametime Everyplace Lotus Domino Everyplace
Platforms	AIX, Solaris, Linux, Windows NT and Windows 2000* *Voice technology supported on NT and AIX	AIX, Solaris, Windows NT, Windows 2000	AIX, Solaris

Comparison of markets

Some architects (and possibly some integrators) are concerned about the applicability of such offerings as these to the markets of business-to-consumer, business-to-business, and business-to-employee or business-to-enterprise. Figure 2-3 is a positioning chart to show where each of the Everyplace offerings fits in these markets.

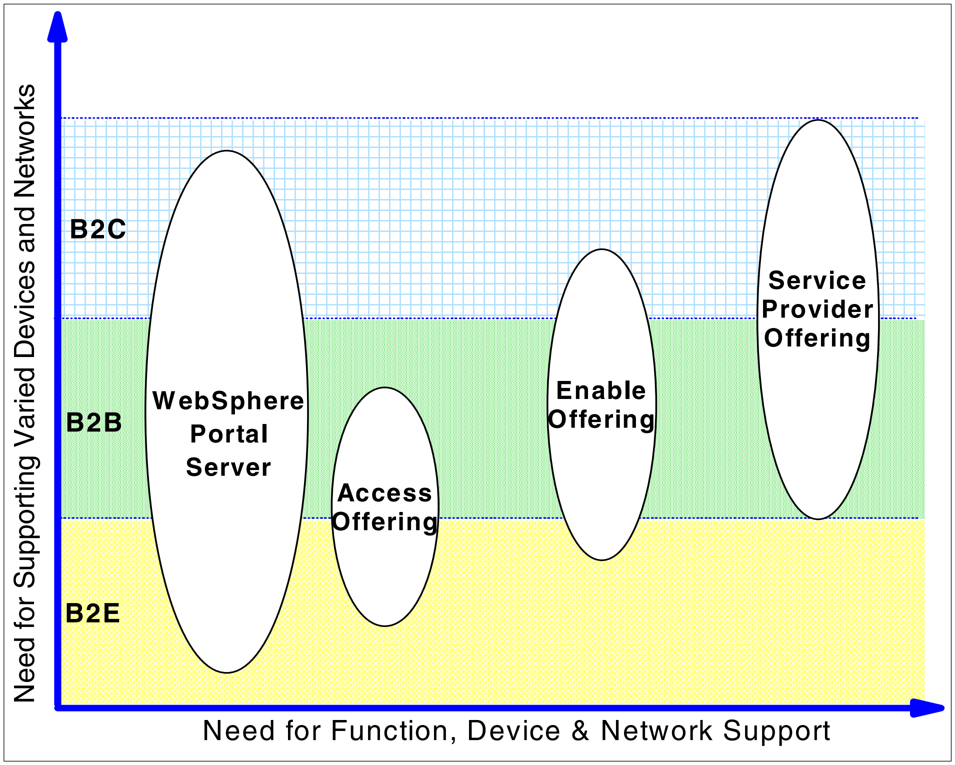

Need for Supporting Varied Devices and Networks
B2C
B2B
B2E
WebSphere Portal Server
Access Offering
Enable Offering
Service Provider Offering
Need for Function, Device & Network Support

Figure 2-3 Market positioning for the Everyplace offerings

Because Everyplace Access is for enabling via voice and wireless Web, it tends to fit better in the business-to-business and business-to-employee applications. Some examples would be using a voice interface to existing Web applications so that:

- Employees can update or check their personal information from a telephone (B2E)
- A customer can check their order status from a telephone (B2B).

Everyplace Enable Offering does not include a voice interface. It is designed to fit into an existing infrastructure where you might want to support only specific devices and specific gateways. Thus it would be well suited for B2E applications, where the infrastructure is in place and you can dictate the device(s) you will support. It is not always feasible to dictate the device you will support for a B2B or a B2C application, so WebSphere Everyplace Server Enable Offering is less well suited to those applications. You can add wireless gateway functions and device support functions to WebSphere Everyplace Server Enable Offering, but you cannot add Intelligent Notification Services nor Location Based Services, which are needed to complete some of the emerging B2C solutions.

Service Provider Offering is really more oriented toward business-to-consumer or business-to-business applications, where you want to offer the most function, you cannot dictate the client device, and you need to be able to connect to whatever carrier network to which the user subscribes.

WebSphere Portal Server is a member of the pervasive computing family and a complement to all three Everyplace offerings, and should be a part of an overall mobile e-business architecture or integration effort. It fits in all three markets, but has minimal device and network control.

2.3.2 Comparison of functional characteristics

This section deals with the functional components in the WebSphere Everyplace Server package offerings; that is, what capabilities are included, optional, or can be ordered separately. For Figure 2-4 on page 60 and Figure 2-5 on page 61:

- Items with “(private)” after them are licensed for the private use of WebSphere Everyplace Server, and may not be used by applications
- Service Provider Offering *Options* are products delivered on the Service Provider Offering CD, for which you need entitlement to use (you must pay for them)
- The arrows indicate the movement from being a separate function or product in one offering to being an included function or product in the next offering.

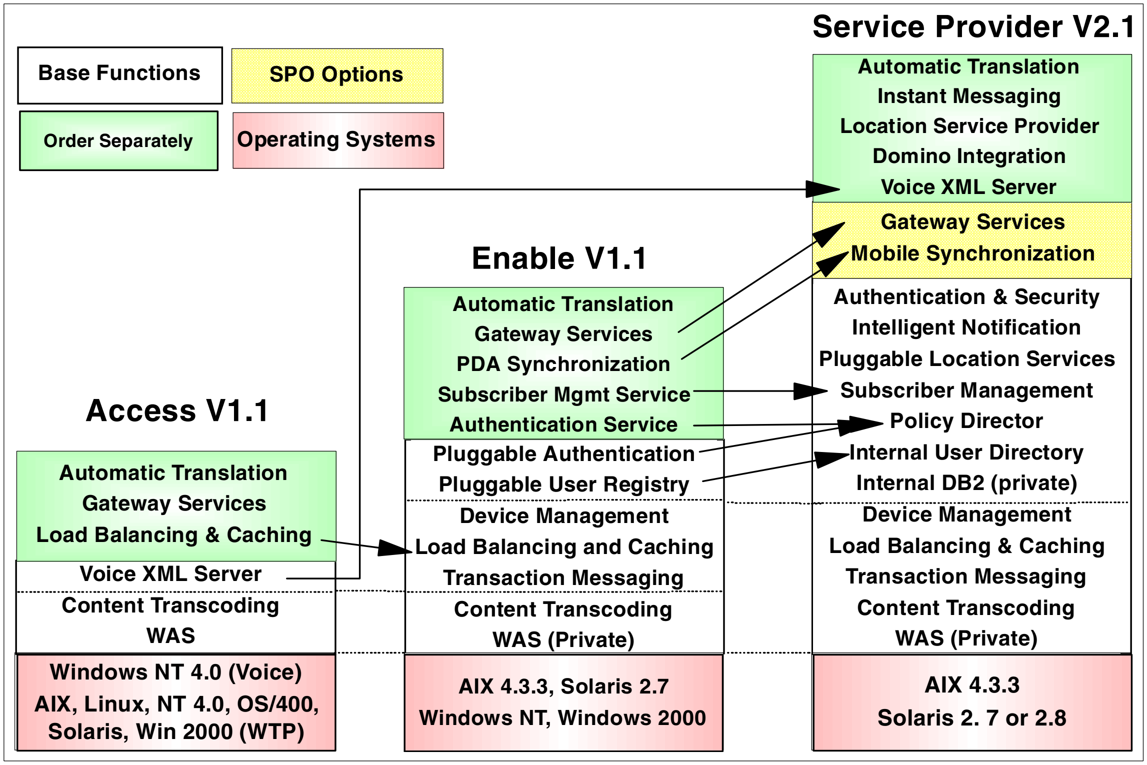
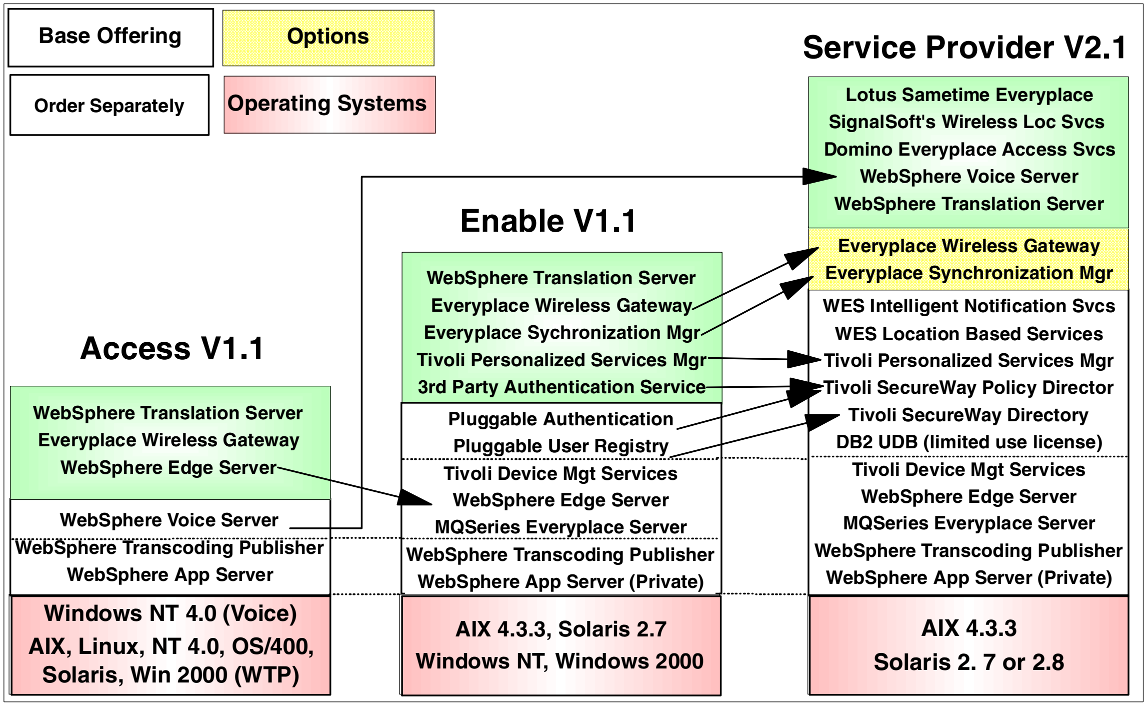


Figure 2-4 Everyplace Server family with functional components

As the charts indicate, the load balancing and caching function (as it is called in Figure 2-4), which is provided by Edge Server (as indicated in Figure 2-5 on page 61), must be ordered separately from Everyplace Access, but as the arrows indicate, it is a base component of WebSphere Everyplace Server Enable Offering. The components between the dashed lines are common between one offering and the next. Note that Everyplace Access includes the VoiceXML function provided by WebSphere Voice Server, and Service Provider Offering supports it as a separately ordered product, Enable Offering has no integration points it.

The pluggable authentication and pluggable user registration in WebSphere Everyplace Server Enable Offering are replaced by integrated authentication (provided by Policy Director) and user registration (provided by SecureWay Directory) in WebSphere Everyplace Server Service Provider Offering. By integrated, we mean WebSEAL-Lite can use Policy Director to check authorization. In this release, Suite Manager and Setup Manager are not integrated with Policy Director.

For WebSphere Everyplace Server Enable Offering, Everyplace Wireless Gateway and IBM Mobile Connect are separately ordered products; whereas for WebSphere Everyplace Server Service Provider Offering they (Everyplace Wireless Gateway and Synchronization Manager) are options, meaning they are included in the product CD package, the Service Provider Offering Setup Manager and Suite Manager support their centralized installation and administration, but you must be licensed to use them.



Base Offering
Options
Order Separately
Operating Systems
Service Provider V2.1
Lotus Sametime Everyplace
SignalSoft's Wireless Loc Svcs
Domino Everyplace Access Svcs
WebSphere Voice Server
WebSphere Translation Server
Everyplace Wireless Gateway
Everyplace Synchronization Mgr
WES Intelligent Notification Svcs
WES Location Based Services
Tivoli Personalized Services Mgr
Tivoli SecureWay Policy Director
Tivoli SecureWay Directory
DB2 UDB (limited use license)
Tivoli Device Mgt Services
WebSphere Edge Server
MQSeries Everyplace Server
WebSphere Transcoding Publisher
WebSphere App Server (Private)
AIX 4.3.3
Solaris 2. 7 or 2.8
Enable V1.1
WebSphere Translation Server
Everyplace Wireless Gateway
Everyplace Sychronization Mgr
Tivoli Personalized Services Mgr
3rd Party Authentication Service
Pluggable Authentication
Pluggable User Registry
Tivoli Device Mgt Services
WebSphere Edge Server
MQSeries Everyplace Server
WebSphere Transcoding Publisher
WebSphere App Server (Private)
AIX 4.3.3, Solaris 2.7
Windows NT, Windows 2000
Access V1.1
WebSphere Translation Server
Everyplace Wireless Gateway
WebSphere Edge Server
WebSphere Voice Server
WebSphere Transcoding Publisher
WebSphere App Server
Windows NT 4.0 (Voice)
AIX, Linux, NT 4.0, OS/400,
Solaris, Win 2000 (WTP)

Figure 2-5 Everyplace Server Family with named components

2.3.3 Comparison of supported devices

In our discussion of device support, we will begin by listing the components that provide the device support. Then we will address each device separately, indicating which Everyplace offering has out-of-the-box support for that device and the type of support it provides.

When architecting a mobile e-business solution, particularly a B2E solution, it is useful to know all the software available for supporting a particular mobile device. For this reason, we provide information about products that are not part of the Everyplace offerings, in the context of supporting specific mobile devices.

Device support in the Everyplace offerings

The following components provide support for pervasive computing devices, and are in one or more of the Everyplace offerings:

- WebSphere Transcoding Publisher -- included in all three Everyplace offerings, WebSphere Transcoding Publisher adapts existing Web pages to the presentation abilities of the various mobile devices
- Everyplace Wireless Gateway -- included only in WebSphere Everyplace Server Service Provider Offering, Everyplace Wireless Gateway includes the Wireless Client, which runs on the mobile devices and allows them to use the IP network and IP-based applications
- Synchronization Manager -- included only in Service Provider Offering, IBM Everyplace Synchronization Manager is for synchronizing mobile devices (mainly PDAs) with host data (PIM data and ODBC-compliant relational data)
- Tivoli Personalized Services Manager -- included only in Service Provider Offering, Tivoli Personalized Services Manager provides operating system-specific agents for mobile devices, to facilitate centralized device management, configuration, software distribution, etc.

Other support for mobile/wireless devices

The following products or offerings complement an architecture or integration effort that extends to the applications that run on the mobile device.

- IBM VisualAge Micro Edition is a development and runtime package for building and running applications on intelligent mobile devices. It brings the productivity gains of virtual-machine-based architectures to the embedded software world and incorporates a complete object-oriented collaborative development environment for creating embedded applications and systems. Visual Age/Micro Edition supports the following operating systems:

Table 2-4 Visual Age / Micro Edition

Supported Operating Systems	Supported Target Platforms
AIX Hard Hat Linux ITRON OSE Palm OS Pocket PC QNX QNX Neutrino QNX RTPSolaris Windows Windows CE	ADS Graphics Client Plus Algorithmics P5064 Ampro Core Module 3SXi Arcom SBC-GXm Development Kit for Embedded Linux Arcom SBC-GXm QNX Realtime Platform Development Kit Casio Cassiopeia EM-500 Compaq iPaq Embedded Planet Blue Planet Windows CE Embedded Planet Linux Planet Development Kit Embedded Planet Motorola PowerPC 823 BDK HP Jornada 420 Hitachi HPW-600ETM (Solution Engine) Hitachi SH7751 Big Sur IBM RS/6000 IBM Walnut Intel Assabet Palm III, Palm V, or Palm Vx Sun Ultra Workstation X86-based PC

On the development side, the product tool set provides function to minimize application resource requirements and connect those applications to your full enterprise. Meanwhile, the runtime environment ensures that your applications run with consistency, speed, and minimum resources when loaded onto the embedded target device.

For more information about Visual Age/Micro Edition and to download the code for free, go to:

`http://www.embedded.oti.com/learn/1_4.html`

- MQSeries Everyplace-- MQSeries Everyplace provides a mechanism through which mobile devices can exchange transactional messages with other mobile devices or with servers that are also running MQSeries Everyplace. While Enable Offering and Service Provider Offering include MQSeries Everyplace, it is only licensed to run in the server; you must acquire the device licenses separately.
- DB2e -- Database 2 Everyplace is a miniature relational database that runs in the mobile device. It facilitates running applications on mobile devices for users who are occasionally connected (rather than always connected).

DB2e works with such development environments as Metrowerks Code Warrior, Microsoft's Visual BASIC, and VisualAge MicroEdition. For the Palm OS environment, IBM provides the Mobile Application Builder for Palm OS.

DB2e supports SyncML, and can synchronize with host data through Synchronization Manager in Service Provider Offering, or through IBM Mobile Connect added to Enable Offering. DB2e can also synchronize with ODBC-compliant databases through DB2e Synchronization Manager (sold separately).

For more information about DB2e, please read Announcement Letter 201-155, or visit the Library at
`http://www-4.ibm.com/software/data/db2/everyplace/library.html`

- WebSphere Portal Server (WPS) extends the capabilities of WebSphere Everyplace by providing an open, scalable framework for enabling the aggregation of information from various content sources and applications into a personalized portal.

  The Portal Server framework provides predefined sample portal applications known as portlets. These portlets provide good examples of how custom portlets can be created. The predefined connectors and portlets are included with source code to allow you to easily modify them for your specific portal needs. Portal Server supports many devices, currently including PCs and selected Wireless Application Protocol (WAP) phones, in both wired and wireless networks.

  Portal Server is not included in any of the Everyplace offerings. Currently, integrating WebSphere Portal Server into a WebSphere Everyplace Server environment (or vice versa) is a mostly manual effort. For more information about Portal Server, please visit
  `http://www-4.ibm.com/software/webservers/portal/`

Supported devices

- Internet-capable cell phones and PDAs

  By “Internet-capable” cell phones and PDAs, we mean PDAs and mobile telephones that have the capability to browse the Internet. These devices typically do not support standard HTML, frames, stylesheets, GIFs and JPEGs, plug-ins, nor many other enhancements available in standard desktop browsers. WebSphere Transcoding Publisher intercepts your unchanged or minimally changed Web pages and transcodes them to suit the needs of the mobile devices.

All three Everyplace offerings include WebSphere Transcoding Publisher. Therefore all three support Internet-capable cell phones and PDAs as mobile Web browsers. Supporting mobile devices as something more than a browser, however, requires additional components. The following subsections address supporting mobile devices for functions beyond Web browsing.

- Palm OS devices
  - WebSphere Transcoding Publisher includes support for transcoding from HTML to Palm OS HTML for Palm.Net devices. All three Everyplace offerings include WebSphere Transcoding Publisher.
  - Tivoli Personalized Services Manager includes cradle support for managing devices using the Palm OS operating system (such as the Palm Connected Organizer Models III and V) while they are attached to their docking station. Provides functions such as configuration and software distribution. Only Service Provider Offering includes Tivoli Personalized Services Manager.
  - Everyplace Wireless Gateway includes a client for Palm OS that enables the device to participate on the IP network and optionally can authenticate the user. Everyplace Wireless Gateway requires WebSphere Everyplace Server Service Provider Offering.
  - Synchronization Manager / IBM Mobil Connect have conduits for Palm devices. Only Service Provider Offering includes Synchronization Manager. You can add IBM Mobile Connect (the non-WebSphere Everyplace Server version of Synchronization Manager) to either an Everyplace Access or a WebSphere Everyplace Server Enable Offering implementation.
  - MQSeries Everyplace currently supports only Palm V and IBM WorkPad C3. It includes support for Java, Code Warrior, and C/C++ for Palm V, and Java for IBM WorkPad C3.

Restriction: Although MQSeries Everyplace offers both synchronous and asynchronous messaging, currently only synchronous messaging and a subset of the security functions is available for Palm OS.

- Other IBM products that support Palm OS devices
  - Visual Age/Micro Edition has libraries for developing applications that will run on Palm OS devices.
  - DB2e has a database that runs on Palm OS.
  - Mobile Application Builder is a simple forms creation tool for creating basic applications. Several sample applications ship with Mobile Application Builder to help the developer get started. Mobile Application Builder for

Palm OS is available for free download from the DB2 Everyplace Web site at `http://www6.software.ibm.com/dl/db2/everyplace-p`.

- Support for Windows CE devices
  - WebSphere Transcoding Publisher -- could be useful if you don't want to re-write your Web pages, since the current release of Windows CE (3.0) only supports HTTP 1.0, and only supports a subset of ASP capabilities. WebSphere Transcoding Publisher is a common component, included in all three Everyplace offerings.
  - Synchronization Manager / IBM Mobil Connect -- have conduits for Windows CE devices. Synchronization Manager is included in WebSphere Everyplace Server Service Provider Offering. You can add IBM Mobil Connect to Everyplace Access and to WebSphere Everyplace Server Enable Offering.
  - Everyplace Wireless Gateway -- the Wireless Client extends the IP network to devices running Windows CE Handheld PC Pro, Windows CE Handheld PC 2000, and Windows CE Pocket PC. Optionally, the Everyplace Wireless Gateway can authenticate the user. Everyplace Wireless Gateway requires WebSphere Everyplace Server Service Provider Offering.
  - Tivoli Personalized Services Manager has a generic Windows CE device plug-in agent that provides basic support for all devices that use the Windows CE operating system. Only Service Provider Offering includes Tivoli Personalized Services Manager. You can add Tivoli Personalized Services Manager to WebSphere Everyplace Server Enable Offering.
  - MQSeries Everyplace has a queue manager for Windows CE/Pocket PC devices (in addition to the desktop versions of Windows).
- Other Products that support Windows CE devices
  - DB2e has a database for the Windows CE/Handheld PC environment.
- Support for Symbian EPOC Devices
  - Synchronization Manager (and likewise, IBM Mobile Connect) allows an EPOC Connect client to update itself via file transfer, and supports draft e-mails when synchronizing with EPOC devices. Only Service Provider Offering includes Synchronization Manager, but you can add IBM Mobile Connect to WebSphere Everyplace Server Enable Offering.
  - Visual Age/Micro Edition supports EPOC devices. It is not included in any of the Everyplace offerings but is available for free download. See "Palm OS devices" on page 65 for download information.

- Support for WAP Devices and WML
  - Everyplace Wireless Gateway, when configured as a WAP Gateway, provides connectivity for multi-vendor WAP 1.1 and WAP 1.2 client devices. WAP cell phones and WAP PDAs can connect to Everyplace Wireless Gateway using a WAP microbrowser. Everyplace Wireless Gateway fully supports the WAP WSP to link the WAP microbrowser with connection-oriented and connectionless, security-enriched and non-secure, WAP services. The performance optimization and security functionality for this mode of connection is that provided by the WAP standard. Everyplace Wireless Gateway requires WebSphere Everyplace Server Service Provider Offering.
  - Tivoli Personalized Services Manager -- Only WebSphere Everyplace Server Service Provider Offering includes Tivoli Personalized Services Manager, but you can add it to an Everyplace Access and it has been tested to work in a WebSphere Everyplace Server Enable Offering implementation.
  - WebSphere Voice Server includes a voice browser that interprets VoiceXML markup, which complements WAP to allow the delivery of voice applications for the mobile Internet. See 2.2.1, “Everyplace Access -- entry level and voice enablement” on page 47 for more detail about the Voice Server.
  - Only Everyplace Access includes WebSphere Voice Server. WebSphere Everyplace Server Service Provider Offering has been tested to support WebSphere Voice Server as an add-on product, and includes some sample VoiceXML scripts on Disc 10.
  - IBM Message Center for DirectTalk V6.4 is a customizable, easy-to-implement, unified messaging system capable of managing voice-mail, fax, and e-mail. Enhancements in Message Center for DirectTalk V6.4 include:
    - Fax support using TR114 fax board from Brooktrout Corporation
    - Enhanced Web e-mail functionality
    - Wireless Application Protocol (WAP) support
    - JavaBean API support
    - Telephony portal
    - LDAP Directory access
    - POP3 e-mail server interface

    For more information on Message Center for DirectTalk, please visit: http://www-4.ibm.com/software/speech/enterprise/ep_7.html
  - WebSphere Portal Server, a separate but complementary product, supports WAP.

  - WebSphere Commerce Suite, MarketPlace Edition is an e-commerce and m-commerce (mobile e-commerce) engine built on the proven foundation of WebSphere Commerce Suite, V4.1 (previously known as Net.Commerce). WebSphere Commerce Suite MarketPlace Edition is a complementary commerce add-on to WebSphere Everyplace Server; or conversely, WebSphere Everyplace Server is a complementary addition to an existing MarketPlace Edition installation. For more information about WebSphere Commerce Suite, MarketPlace Edition, please visit `http://www-4.ibm.com/software/webservers/commerce/wcs_me/index.html.`

- Support for SMS
  - Everyplace Wireless Gateway is the component of WebSphere Everyplace Server that provides support for Short Messaging Service (SMS). Everyplace Wireless Gateway can also act as a push-proxy gateway (PPG). Only WebSphere Everyplace Server Service Provider Offering includes Everyplace Wireless Gateway.

- Support for i-mode devices
  - Everyplace Cookie Proxy provides connection between i-mode phones and Web applications by emulating the browser's cookie function, since i-mode phones do not support cookies.
  - WebSphere Transcoding Publisher includes the ability to transcode HTML to i-mode Compact HTML (CHTML).
  - WebSphere Portal Server supports i-mode. WebSphere Portal Server is a complementary product; it is not included in any of the Everyplace offerings.
  - Policy Director (the version shipped in WebSphere Everyplace Server) does not support i-mode; however, a future release of Policy Director will. Tivoli Systems Inc. announced on July 23, 2001 that it has successfully tested Policy Director for providing secure authentication and authorization to applications and data accessed through i-mode wireless devices.

  For more information about i-mode, visit the NTT DoCoMo Web site at `http://www.nttdocomo.com`

- Support for the Blackberry device by Research In Motion (RIM) is limited to the Everyplace Wireless Gateway support, which allows the *sendmail* function. There is currently no device support nor synchronization support.

2.4 Summary

The Everyplace offerings are packages that enable existing Web applications to be accessible to mobile users who have a supported device. Everyplace also supports new applications designed for mobile users and device-specific applications. Everyplace Access is primarily for Internet-capable devices and voice support. WebSphere Everyplace Server Enable Offering has much more function and is designed to integrate with existing authentication services and user directories. WebSphere Everyplace Server Service Provider Offering is the most comprehensive package that includes its own user directory and authentication and authorization mechanism, in addition to a carrier gateway and a mobile device synchronization manager.

3

Component relationships

In this chapter we discuss the relationships between the various WebSphere Everyplace Server Service Provider Offering components, how they interact, how they differ from the stand-alone versions, and how they interact with separately available IBM products. Specifically, we discuss:

- Interdependencies -- that is, which components depend on other components for specific functions, what those other components are, and what the function is
- Complementary relationships -- how certain WebSphere Everyplace Server Service Provider Offering components interact with certain other IBM products that complement the functions in WebSphere Everyplace Server Service Provider Offerings.

3.1 WebSphere Everyplace Server architecture

This chapter introduces a high-level architecture, showing how the components of IBM WebSphere Everyplace Server can be combined to create complete solutions. Clearly the architecture for any specific solution will vary according to the requirements, and so may have significant differences from what we show in this book. Indeed, your solution may not appear to need all of the components that are included in the suite. However, you should be aware that there are interdependencies among some of the components, and you may also need some requisite components.

3.1.1 Overview

Figure 3-1 is a very high-level view of where Everyplace Server fits into the overall computing infrastructure.

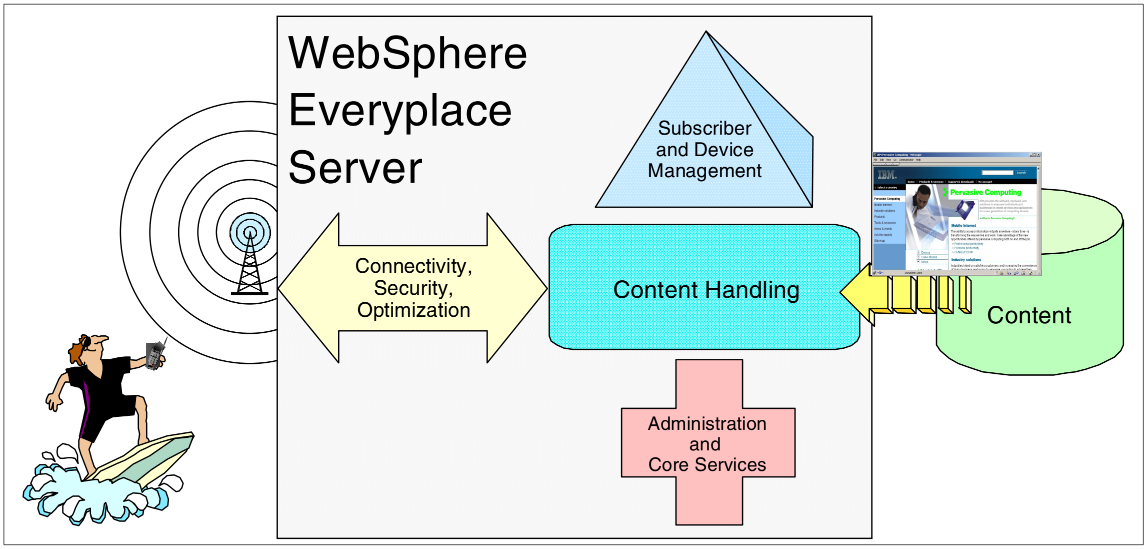

WebSphere Everyplace Server
Subscriber and Device Management
Connectivity, Security, Optimization
Content Handling
Content
Administration and Core Services

Figure 3-1 IBM WebSphere Everyplace Server Service Provider Offering - very high level

3.1.2 Connectivity components

Everyplace Server contains the following components that combine and interact to provide connectivity:

- Wireless Gateway
- Wireless Client
- Edge Server Caching Proxy

- Edge Server Load Balancer
- WebSEAL-Lite plug-in for Edge Server Caching Proxy

Additionally, WebSphere Everyplace Server Service Provider Offering includes interfaces for connecting to the WebSphere Voice Server, if present, to provide a voice interface into the Everyplace Server environment.

3.1.3 Security

Everyplace Server has an integrated security model to enable single sign-on and authorization of users and applications. To enable this model, Everyplace Server contains the following components:

- WebSEAL-Lite
- Policy Director
- Wireless Gateway
- Active Session Table Server

3.1.4 Content adaptation components

These WebSphere Everyplace Server Service Provider Offering components adapt and deliver content destined for wireless devices:

- Edge Server Caching Proxy
- MQSeries Everyplace in gateway mode
- Synchronization Manager
- WebSphere Transcoding Publisher
- Location Based Services
- Intelligent Notification Services

3.1.5 Management services

To manage users and wireless devices, WebSphere Everyplace Server Service Provider Offering includes these components:

- Tivoli Personalized Services Manager
  - Customer Care
  - Device management
  - Self Enrollment
  - Self Care

3.1.6 Core components

IBM WebSphere Everyplace Server is a multichannel network product. By this, we mean that it forms an interface between network(s) and core business applications by means of multiple network channels. Clients can thus be connected to it in several ways, and can connect either to specific application servers or provide access to some types of networks, for example, an intranet. These different requirements make every Everyplace Server deployment more or less unique yet, based on the common set of components, made to fit together as pieces of a puzzle with no single answer.

We will describe here three different deployment models:

- Network access
- Adaptive network access
- Adaptive portal

Each of the following sections describe a model in more detail, showing how an appropriate solution architecture can be built using the Everyplace Server core components. We also touch on the likely business cases that will exist for each.

Please note that we provide these models only as examples of how the components interact at a high level; to create an architecture for a specific deployment, a solution architect will need to understand both the individual components and how they interact (see Chapter 4, “Business-to-employee” on page 151 and Chapter 5, “Business-to-consumer” on page 219, for more realistic deployable solutions).

Network access

Providing large-scale network access is a common business requirement, typically implemented by Internet Service Providers or large enterprises. An ISP typically provides dial-up access to the Internet by setting up a large number of connectivity devices (such as modems). Today they are expanding these services to support wireless and wireline connectivity for new classes of devices.

An enterprise may want to deploy this model to provide wireless or wireline access to its intranet by its employees. In contrast to a public ISP, an enterprise may require the access to be more secure, and so it may use specialized clients such as the Wireless Gateway clients.

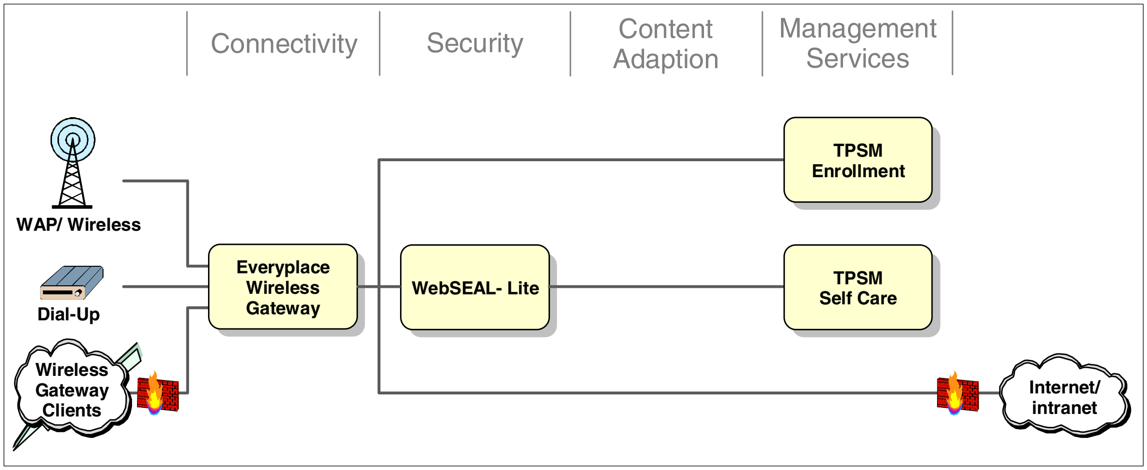


Figure 3-2 WebSphere Everyplace Server core architecture - network access

The core components of an Everyplace Server architecture for this model are shown in Figure 3-2. The central component is the Wireless Gateway, which provides access to the network. If the network is the Internet, a firewall should be deployed to form a *Demilitarized Zone* (DMZ), which protects the Everyplace Server components as illustrated. Alternatively, if the network is a secure intranet, this level of protection may not be required.

In addition to the access itself, the enrollment component of Tivoli Personalized Services Manager offers a service allowing users to self-enroll for the first time. Access to this component must be available to anonymous users, since by definition, self-enrollment is meaningful only when carried out by a user who is unknown to the system. Also, the self enrollment may require so much information by the user that the application will only be made available from a browser as opposed to a less user-friendly small wireless device.

Other user services offered would be handled by the Tivoli Personalized Services Manager Self Care component. Examples are updates of personal data, changing the terms of the user agreement, and viewing account and billing status. These services will require that the user has been authenticated by WebSEAL-Lite.

Adaptive network access

Either Internet Service Providers or enterprises may wish to extend their services from simply providing network access to providing content adaptation.

Adaptation is necessary to allow a device to use applications and content that were designed in a format not normally usable by that device. Examples of this are the conversion between different markup languages (such as XML, HTML, WML and HDML) and conversion between different image formats (such as GIF, JPG, WBMP and black/white versions).

Optimization is another use of adaptation. Content can be changed in several ways to improve response times for users that have low bandwidth connections, such as wireless or dial-up connections. Examples are reductions in the size and/or color depth of images, substituting images with hyperlinks for images, and removing unnecessary information, such as comments or spaces in the markup language.

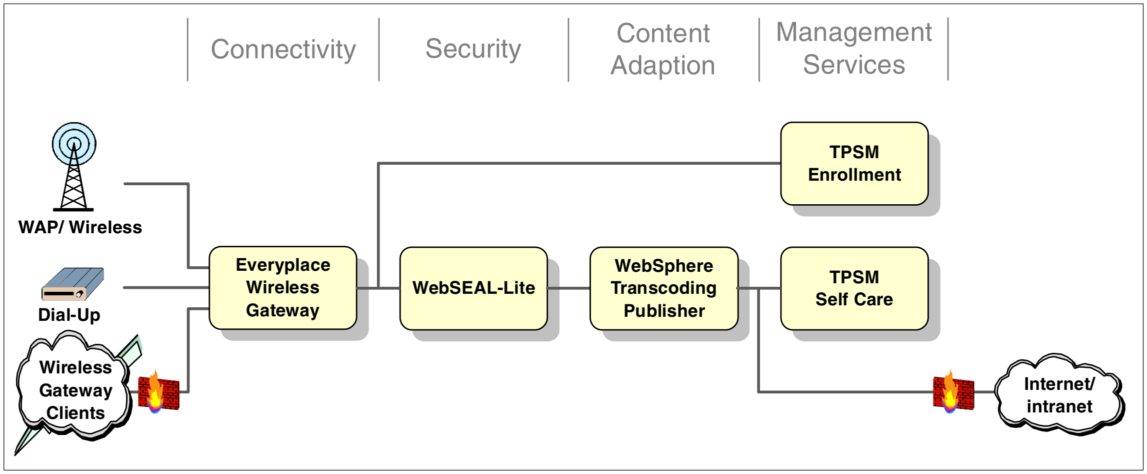


Figure 3-3 Everyplace Server core architecture - adaptive network access

As with the basic network access model, the access to the network itself is still provided by the Wireless Gateway, and the services for enrollment and Self Care are carried out by components of Tivoli Personalized Services Manager[1].

The component that enables adaptation is WebSphere Transcoding Publisher, whcih can be configured to perform the adaptations mentioned in the previous examples.

[1] All the sample Tivoli Personalized Services Manager applications are servlets and JS's with an HTML view and as such may be subject to transcoding and made available on wireless devices, but not necessarily. In many cases you may prefer to make these applications available only from a browser due to the complexity of the application and/or too much data must be entered to make them useful for access from a wireless device.

The main use of WebSEAL-Lite is to provide security service to the solution. However, WebSEAL-Lite also assists in the recognition of the device and type of network that each connection is using. This information is then used to carry out the most appropriate adaptation.

3.1.7 Adaptive (multi-modal) portal

The previous two models are mostly concerned with providing network access. This model is more concerned with providing content and applications to multiple devices.

Everyplace Server is *not* a platform for building core business applications. It is a framework for building secure, integrated, and adaptive access to a portal, while allowing the adaptation of the content that is to be aggregated into that portal.

We believe that this model is most applicable to enterprises that need to implement a multi-modal portal, such as the front-end to existing or new applications. Everyplace Server is equally suitable in the case where content and/or services are being provided by external business partners.

This model is equally applicable to an ISP that wishes to extend its business into the portal market, or to an *Application Service Provider* (ASP) that needs to provide a portal as the cornerstone of its business.

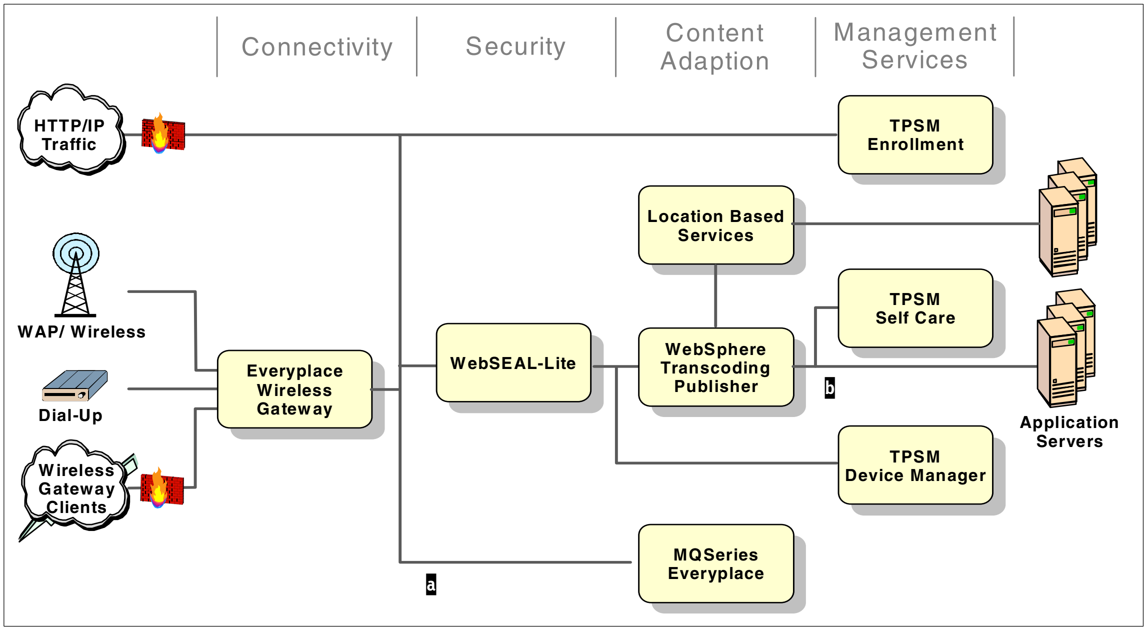


Figure 3-4 Everyplace Server core architecture - adaptive (multi-modal) portal

Figure 3-4 illustrates how the Everyplace Server components form an infrastructure that is the front-end to one or more application servers. The main point of entry will typically be either the Internet or an intranet, in both cases using HTTP running over IP. Optionally, both wireless and wireline access can be added to the solution by incorporating the Wireless Gateway.

Users entering from either the Internet or an intranet need to be authenticated before they can access services in the WebSphere Everyplace Server domain. WebSEAL-Lite performs this authentication and provides single sign-on for those applications and services located inside the Everyplace Server domain.

There are, however, two notable exceptions:

- Tivoli Personalized Services Manager enrollment

  As with the earlier models, the enrollment component of Tivoli Personalized Services Manager must be accessible to non-authenticated users.

- IBM MQSeries Everyplace

  This component is built around a sophisticated security mechanism that is not based on HTTP authentication. As a consequence, at this time there is no advantage to passing its network traffic through WebSEAL-Lite.

If a variety of devices are to be supported then content adaptation will be required, and consequently the WebSphere Transcoding Publisher will generally be the next component in the network.As always; however, there are exceptions. Services such as MQSeries Everyplace and Tivoli Device Manager Server are not appropriate for transcoding. Consequently these services will be placed ahead of the transcoder. In general we would expect this to be the case for any non-browsing service.

Next we find the core business applications, information services, and the Tivoli Personalized Services Manager Self Care component. These all rely on WebSEAL-Lite authentication and, if needed, content adaptation performed by IBM WebSphere Transcoding Publisher.

Finally a portal implementation, as we have described it, can be readily extended to include network access. As with the first two models, the connection to the network (either Internet or intranet) could be attached at either of the points marked a and b in Figure 3-4. Clearly, the choice of attachment points determines if content adaptation is available or not. Firewalls may also be added as appropriate.

3.1.8 Unauthenticated access

Sometimes it is necessary to provide unauthenticated users with access to public resources, either application servers or Everyplace Server components such as Tivoli Personalized Services Manager Enrollment.

There are three obvious solutions:

1. Dual security zones
2. Dual security zones with adaptation
3. Shared security zone with adaptation

The following sections describe each of these in more detail.

Dual security zones

The simplest way to resolve this is to duplicate the infrastructure components as necessary. If you consider that Tivoli Personalized Services Manager Enrollment is a public service, then you can see that all of our previous models took this approach. We placed the Tivoli Personalized Services Manager Enrollment server on a separate leg of the network, making it visible to all incoming connections. Public application servers could also be placed there, as illustrated in Figure 3-5.

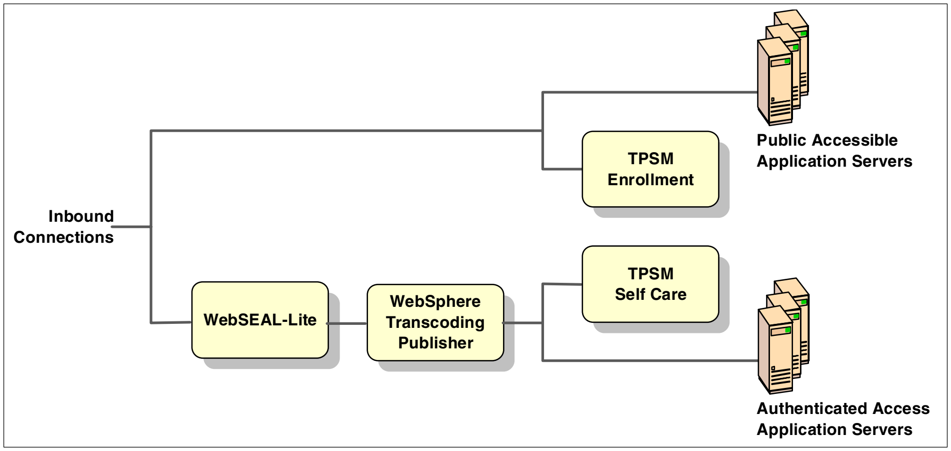


Figure 3-5 Unauthenticated access - dual security zones

This represents a very simple solution to the issue, with an extremely easy-to-understand security model, and no requirement for any additional security configuration.

The drawback to this approach is inherent in its simplicity - it duplicates infrastructure, resulting in additional cost and more maintenance issues.

Dual security zones with adaptation

The dual security zone approach works well, but without enhancement it cannot easily support multiple device types, since there is no facility for content adaptation. This can be simply resolved by taking the approach one step further and duplicating the IBM WebSphere Transcoding Publisher.

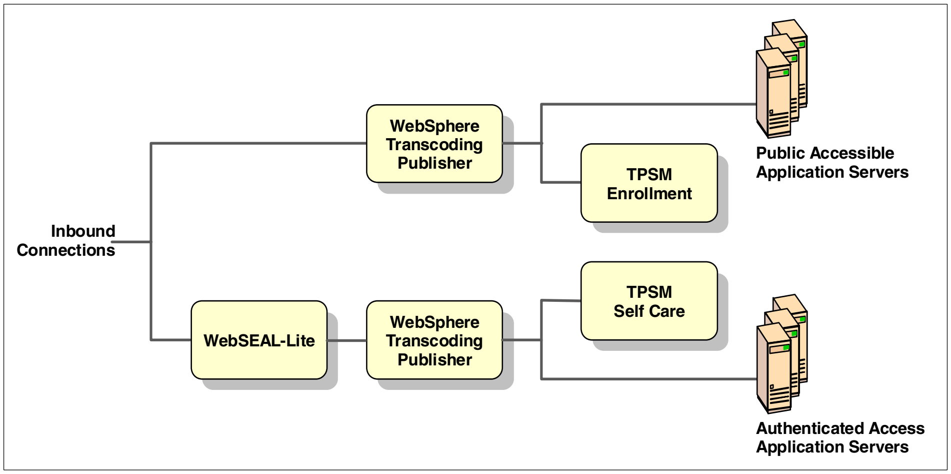

WebSphere Transcoding Publisher
TPSM Enrollment
Public Accessible Application Servers
Inbound Connections
TPSM Self Care
WebSEAL-Lite
WebSphere Transcoding Publisher
Authenticated Access Application Servers

Figure 3-6 Unauthenticated access - dual security zones with adaptation

Unsurprisingly, the advantages are exactly as for the previous approach, but with the addition of content adaptation. Similarly the disadvantages are also as before, but requiring even more duplication of infrastructure.

Shared security zone with adaptation

An alternative to the previous options is to loosen the WebSEAL-Lite security so that users who are unauthenticated by Everyplace Server may access the same infrastructure as authenticated users. This requires changes to the default configuration for WebSEAL-Lite; refer to 3.3.5, “WebSEAL-Lite” on page 92 for more details.

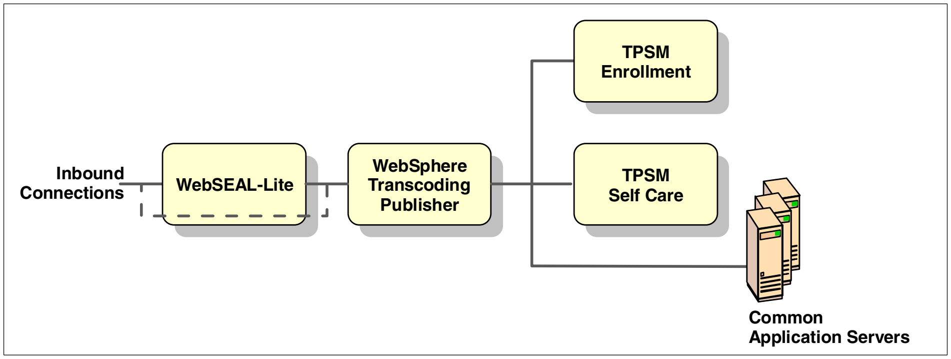


Figure 3-7 Unauthenticated access - shared security zone

The advantage to this is that there is better exploitation of the infrastructure, since there is no need for any duplication. Appropriate security is maintained by configuration of the Policy Director and setting access control lists (ACLs).

3.1.9 Expanded solution

If we then add into these scenarios the new components in WebSphere Everyplace Server, Location Based Services, Intelligent Notification Services, and Policy Director, we can build an overall logical architecture such as the one shown in Figure 3-8.

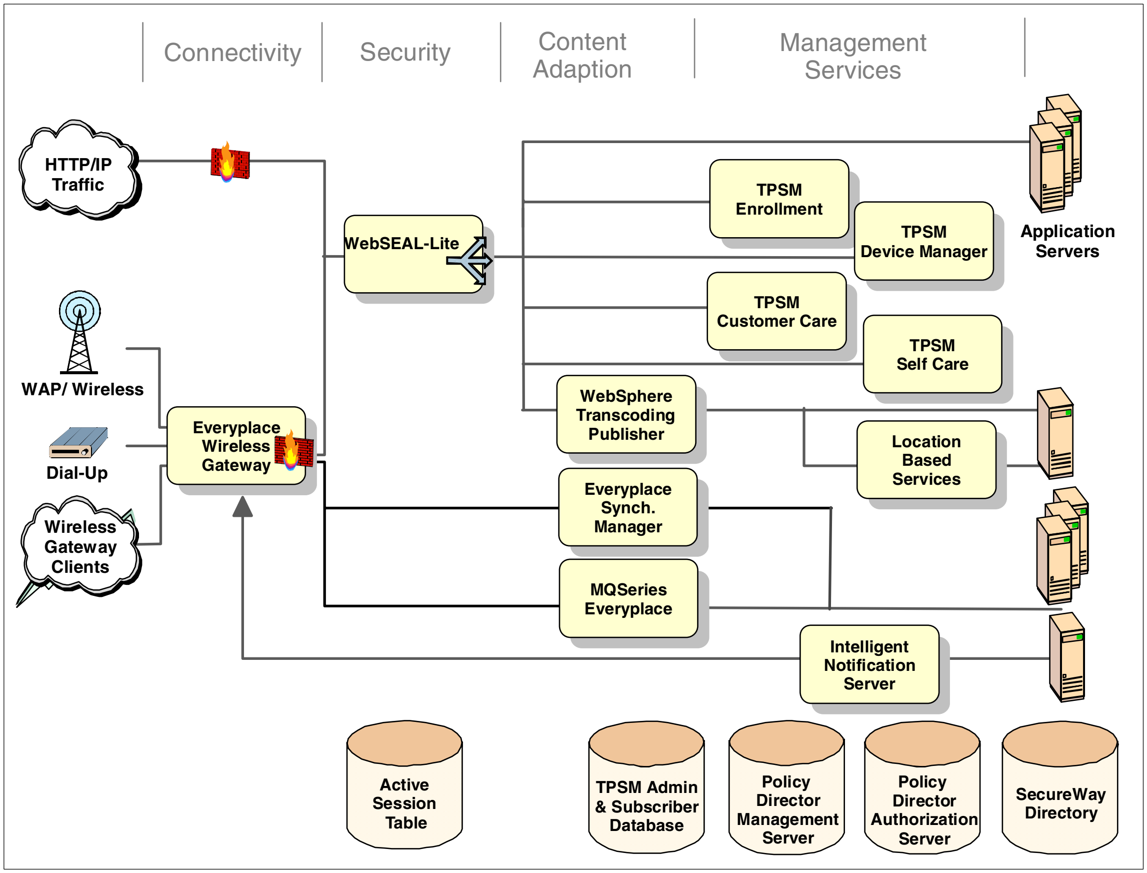


Figure 3-8 Everyplace Server Service Provider Offerings - reference architecture

This figure shows:

- The multi-modal portal, accessed from a variety of IP and non-IP based devices
- The use of WebSEAL-Lite's *junction* capability (further described in 3.3.5, "WebSEAL-Lite" on page 92), to route traffic to the most appropriate part of the infrastructure
- The use of Policy Director in addition to SecureWay Directory
- One view of how functions might be separated between the Demilitarized Zone (DMZ) and the secure zone via a firewall

This is the logical model assumed for the remainder of this chapter, and for chapters 7 and 8.

3.2 Product requisites

Here we summarize the software requisites for the WebSphere Everyplace Server component products. Table 3-1 is provided for background and planning only. You should always check product documentation as you prepare to install.

All components run on AIX 4.3.3 Maintenance Level 8, or on Sun Solaris 7 and 8, SPARC-based systems, unless otherwise documented.

Note that IBM Everyplace Wireless Gateway, in particular, has additional hardware requirements, depending on which networks you wish to connect to, which are not documented here.

Table 3-1 Product requisites

Everyplace Server Component	**Software requisites**
Active Session Table Server	None
Everyplace Cookie Proxy	WebSphere Edge Server Caching Proxy. A dedicated instance of Caching Proxy is required. Cookie Proxy and WebSEAL-Lite cannot coexist using the same Caching Proxy. In order to run both on the same machine two instances of Caching Proxy are necessary and each instance must listen on a separate port.
Intelligent Notification Services	► SecureWay Directory must be installed and running in the domain before installing this component. ► DB2 Universal Database (Server/Client). ► WebSphere Application Server (required for Intelligent Notification Services applications; not required on the Intelligent Notification Services servers themselves). ► IBM HTTP Server. ► Everyplace Wireless Gateway is strongly recommended. User customization is required to use other gateways.
Location Based Services	► SignalSoft must be set up in the domain ► Tivoli SecureWay Policy Director
Everyplace Suite Manager	► SecureWay Directory must be installed and running in the domain before installing this component ► Java Runtime Environment 1.3 ► Browser to view documentation
Synchronization Manager	► DB2 Universal Database (Server/Client) ► Lotus Notes/Domino R5 server (on local machine) if synchronizing with Lotus Notes (UNIX server) ► Microsoft Exchange Server 5.5 (Windows NT 4 or Windows 2000 server) if synchronizing with Microsoft Exchange

Everyplace Server Component	Software requisites
Everyplace Wireless Gateway	► Tivoli SecureWay Policy Director (optional) ► DB2 Universal Database (Server/Client) or Oracle8i database Version 8.1.7 ► If using Oracle8i: Merant DataDirect Connect ODBC 3.6.0 ► The client must be local, but the server can be remotely installed for either DB2 or Oracle ► IBM GSKit SSL Library
MQSeries Everyplace	Java Runtime Environment 1.3
Sametime Everyplace	► Windows NT or Windows 2000 ► Sametime Connect 1.5 or higher ► Sametime 2.0 ► Domino Server ► Mobile Services for Domino ► A WAP gateway
SecureWay Directory	► DB2 Universal Database Server and Client on local machine (Server/Client) ► IBM HTTP Server
Tivoli Personalized Services Manager	► DB2 Universal Database (Server/Client) or Oracle database ► WebSphere Application Server ► IBM HTTP Server ► JDK 1.2.2
Tivoli SecureWay Policy Director	► IBM DCE (Distributed Computing Environment) V3.1 for AIX and Solaris ► SecureWay Directory installed in the domain
Voice Services	► IBM WebSphere Voice Server 1.5 for Windows ► IBM Direct Talk for Windows or AIX ► IBM ViaVoice for Windows Standard Edition
WebSEAL-Lite	► Tivoli SecureWay Policy Director (optional). ► WebSphere Edge Server Caching Proxy. A dedicated instance of Caching Proxy is required. WebSEAL-Lite and Cookie Proxy cannot coexist using the same Caching Proxy. In order to run both on the same machine two instances of Caching Proxy are necessary and each instance must listen on a separate port. ► Everyplace Active Session Table Server
WebSphere Edge Server Caching Proxy	► WebSphere Edge Server Load Balancer -- administration package and device driver ► IBM GSKit SSL Library
WebSphere Edge Server Load Balancer	WebSphere Edge Server Caching Proxy -- if installing Content Based Routing (CBR) subcomponent

Everyplace Server Component	Software requisites
WebSphere Transcoding Publisher	► SecureWay Directory must be installed and running in the domain before installing this component ► JDK 1.2.2

3.3 WebSphere Everyplace Server detailed interactions

This section provides information about specific WebSphere Everyplace Server components, and how they interact with their neighbors in the WebSphere Everyplace Server environment. This information is intended to allow the WebSphere Everyplace Server architect to understand the component relationships in sufficient detail to understand product-specific design constraints, and to be able to communicate their requirements to product specialists.

3.3.1 WebSphere Everyplace Server HTTP headers

Below are the set of headers created by and for WebSphere Everyplace Server components, including an example of each.

- **X-IBM-PVC-Session**

  This header identifies the session. The value is a variable-length string that is unique for a given user, device, and network combination. The session is created by either WebSEAL-Lite or IBM Everyplace Wireless Gateway.

  ```
  x-ibm-pvc-session: tmiller23123984729837492387492837429384
  ```

- **X-IBM-PVC-Session-Location**

  Identifies the location of the Active Session Table where the session identified by x-ibm-pvc-session can be found. This header is added when the session is created.

  ```
  x-ibm-pvc-session-location: ast1.raleigh.ibm.com:8017
  ```

- **X-IBM-PVC-User**

  Identifies the user in the Tivoli Personalized Services Manager space. The value of this header consists of the username and realm separated by the ‘@’ character. This header is added by Wireless Gateway or WebSEAL-Lite during authentication.

  ```
  x-ibm-pvc-user: tmiller@acme.com
  ```

- **X-IBM-PVC-Device-Type**

  Provides a mapping from the client device type. This header is used as a key to the WebSphere Everyplace Server LDAP device profile. This header is always added by WebSEAL-Lite.

  ```
  x-ibm-pvc-device-type: WAP-Device
  ```

- **X-IBM-PVC-Network-Type**

  Provides a mapping from the origin network. This header is used as a key to the WebSphere Everyplace Server LDAP network profile. This header is always added by WebSEAL-Lite.

  ```
  x-ibm-pvc-network-type: default
  ```

- **X-IBM-PVC-Client-Id**

  Provides additional information to uniquely identify the user's session and /or device. The value of this token represents a single device and it is dependent on the connection type and/or the connecting gateway. The token consists of two parts. The first part is a two-digit type indicator that can be:

  08 - Indicates the client ID is an IP address. This field is created by WebSEAL-Lite.

  1F - Indicates the client ID is a phone number (caller number). This is created by IBM Everyplace Wireless Gateway when WAP clients have been authenticated.

  The second part of the token consists of a variable-length string that is the device identifier in the format dictated by the type indication.

  ```
  ibm-ibm-pvc-client-id: 089.87.59.55
  ibm-ibm-pvc-client-id: 1f0119195551212
  ```

3.3.2 IBM Everyplace Wireless Gateway

Note that Everyplace Wireless Gateway's WAP Gateway requires an HTTP proxy. The HTTP proxy can reside on the same machine as the Wireless Gateway. The recommended HTTP proxy is Edge Server Caching Proxy and the HTTP proxy is typically WebSEAL-Lite. An optional WML Binary Encoder and cache storage plug-in is available with the Wireless Gateway package.

Note that there is a configuration option in the gateway that will force all WAP requests to a single Web server. Using this special configuration, an HTTP proxy would not be required.

3.3.3 Edge Server Caching Proxy

A proxy server can be used as an intermediary between a client and a content server, to protect the client from seeing the individual servers that make up the infrastructure, to filter requests, and to check the format of the traffic and reformat any corrupt data that is proxied through it. The proxy server will assume responsibility for retrieving and returning data from a content server, and returning it to a client.

Many people continually retrieve the same Web content. This repetition wastes network bandwidth if identical content traverses the Internet each time a user requests a file. Bandwidth limitation is also aggravated by the actual size of the Web traffic that is flowing through it. In many cases the volume of outbound traffic is substantially greater than that of inbound traffic. For example, HTML and embedded images sent from a server are typically at least 10 times the size of the client URLs that request them.

The Edge Server Caching Proxy provides a way to address these issues. It helps deliver consistently rapid response rates, reduces the load on the Internet and network bandwidths, and increases server content availability.

The Caching Proxy consists of three main components: a proxy server, a cache, and a content filter.

Proxy Server

The Caching Proxy provides the functionality of a proxy server for the following protocols: HTTP, File Transfer Protocol (FTP), Gopher and Real-Time Streaming Protocol (RTSP). The proxy server accepts the client request, regenerates the client request, and sends the request to the content server on behalf of the client. It then retrieves the data from the destination content server and forwards the request back to the client. These functions will be described further in “Proxy server component” on page 88.

Cache

The Caching Proxy is also a cache. Information retrieved from the destination content server can be cached by the Caching Proxy. When it retrieves a file from the content server, it can store a copy so that if it receives another request for the same file, the Caching Proxy does not have to go back to the content server. These functions are further described in “Edge Server Caching Proxy” on page 319.

Content filter

The Caching Proxy can also act as a filter. Using the Platform for Internet Content Selection (PICS) rules, it can restrict clients using the proxy server to accessing only certain types of data. The filter function of the Caching Proxy can be used to filter out the content of Web sites, to exclude violence, language or nudity, etc. The filter can also be used to block any viruses that may potentially be returned with content server responses. The advantage of the filter component of Caching Proxy is that content filters are set at the proxy level rather than at the browser level. Filtering at the browser requires configuration for each browser and can therefore be easily overridden. For a large organization with thousands of browsers to administer, this is both extremely cumbersome and limits the control an organization has to monitor Web content being accessed from their internal organization.

The Caching Proxy as a prerequisite

The Caching Proxy is a prerequisite component for WebSEAL-Lite and for Location Based Services. WebSEAL-Lite and Location Based Services are written to be hosted and launched by the Caching Proxy.

This next section provides a detailed outline of the functions of each Caching Proxy component that can be used in WebSphere Everyplace Server.

See *WebSphere Edge Server for Multiplatforms Getting Started Guide* and *Caching Proxy User's Guide* for a complete description of all the features that the Caching Proxy provides.

Proxy server component

A proxy server can be used for several different deployment scenarios:

- Forward proxy

  Clients can configure their browsers to direct all their Web traffic through a Caching Proxy. In this scenario, the Caching Proxy is configured as a forward proxy.

- Transparent proxy

  The client re-direction to access the Internet can take place at the network level by configuring routers to direct client requests transparently through the Caching Proxy. In this scenario, the Caching Proxy is configured as a transparent proxy.

- Reverse proxy

  Clients can connect directly to the Caching Proxy, which they believe to be the destination content server. For this scenario, the Caching Proxy is configured as a reverse proxy.

How forward proxy works

Figure 3-9 illustrates how Caching Proxy works in forward proxy mode.

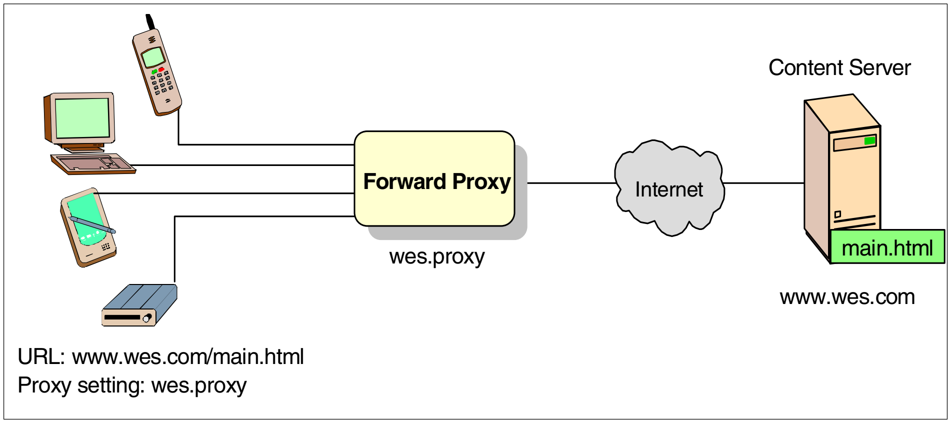


Figure 3-9 Caching Proxy acting as a forward proxy

For a forward proxy, client requests are sent to the content server through a proxy positioned before the Internet. This is the traditional use of a proxy server. A client's browser is configured to direct requests to the proxy server, which is configured to accept them. When a client requests a file, the request is directed to the forward proxy server using the proxy settings configured in the client's browser, as Figure 3-9 shows. Having received the request, the Caching Proxy obtains the requested host name from the HTTP header. It will then regenerate the request with its own IP address, and forwards the client request to the destination content server. Once the proxy receives the response back from the content server, it will direct the returned request to the originating client.

In the forward proxy scenario, the proxy is acting on behalf of the originating client.

How transparent proxy works

For some clients, such as WAP browsers, you cannot or may not want to configure proxy settings at the browser level. To take advantage of the functions that a forward proxy provides, the proxy settings must be configured at the network level, by using routers to re-direct client requests to the proxy server. The Caching Proxy can support this functionality by configuring the proxy server as a transparent proxy, illustrated in Figure 3-10.

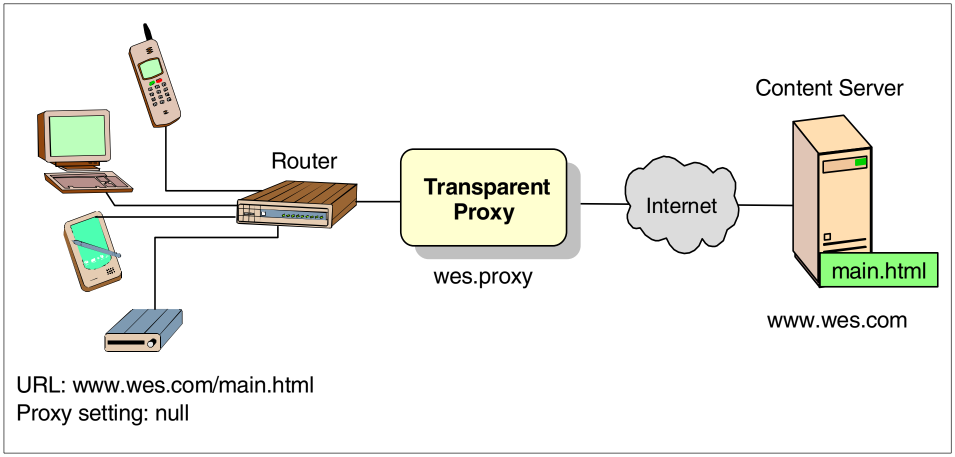


Figure 3-10 Caching Proxy acting as a transparent proxy

A transparent proxy will transparently redirect Web traffic to the proxy server through a router or switch. A transparent proxy server builds upon the principles of a forward proxy server. The main difference is that the client is unaware that the requests for a destination content server are intercepted by a proxy; the client points directly to the destination content server and does not configure any proxy browser settings.

To use a transparent proxy, a router is programmed to listen and redirect all requests from clients to the transparent proxy instead of sending the traffic directly to the destination content server. The Caching Proxy will monitor the network for connection requests to any ports configured to listen for HTTP traffic. When a connection request arrives, it is sent to the Caching Proxy to process.

The transparent proxy server is supported only on AIX and Linux, and applies only to HTTP requests that do not require authentication.

How reverse proxy works

Instead of configuring proxy redirection when accessing a content server either at the browser or at the network level, the client can point directly to the Caching Proxy. In this case, clients access the content server via a reverse proxy server, as shown in Figure 3-11.

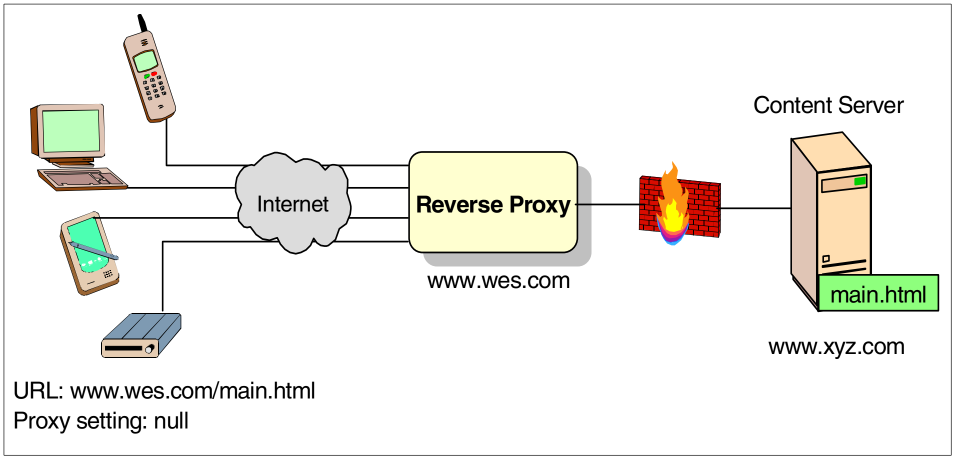


Figure 3-11 Caching Proxy acting as a reserve proxy

When a proxy server is configured as a reverse proxy server, it appears to the client to be the destination content server. To the content server, the reverse proxy server acts as the originator of client requests. If a client wants to access a file, for example main.html as shown in Figure 3-11, the client points its browser to the reverse proxy, `www.wes.com` believing this to be the Internet address of the content server. The reverse proxy server will accept the client request for main.html, retrieve the requested page from the content server, residing on `www.xyz.com`, and return it to the client.

A reverse proxy server hides your content servers from the public Internet because only the reverse proxy server can directly communicate with the content server from outside the firewall.

Protecting a proxy server

As part of the proxy functionality, the Caching Proxy can be configured to protect access to the proxy server and its resources. It can be configured to enable basic authentication to all users that try to access the proxy function, by prompting for a use name and password. When protection is enabled, only authorized users can access the Caching Proxy.

As part of the protection configuration, you define a list of protection rules, or a protection setup, for your proxy server and its resources. The setup directives are very flexible and allow you to customize protection to suit your architecture and security requirements. Generally you define the protection directives for the whole proxy server, individual directories or requests, or any combination of these. Protection of the proxy and its resources is then implemented based on the request that the proxy server receives.

3.3.4 Edge Server Load Balancer

Edge Server Load Balancer can be implemented in front of a cluster of content servers to efficiently and effectively balance the load of client requests.

In the WebSphere Everyplace Server environment, Edge Server Load Balancer is primarily used for scaling. Therefore, Load Balancer is discussed in more detail in “Edge Server Load Balancer” on page 323.

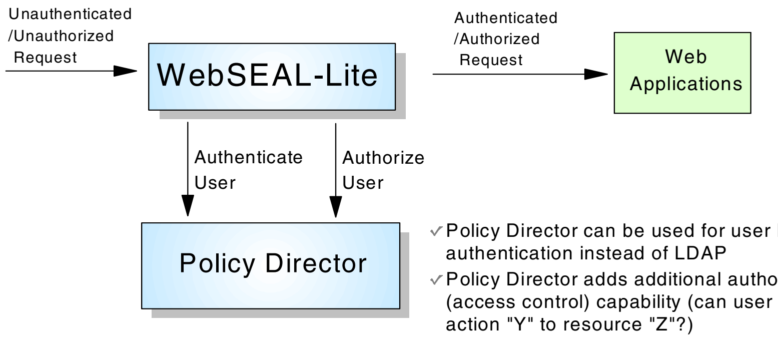
3.3.5 WebSEAL-Lite

WebSEAL-Lite is used to authenticate users. It can be implemented with or without Policy Director; if it is used with Policy Director, it can also provide authorization services for your WebSphere Everyplace Server environment.

Some familiarity with Policy Director will be required to administer access control using WebSEAL-Lite. See “Policy Director” on page 110 to understand how Policy Director implements access control.

SPO Policy Director in WebSEAL-Lite

High-level flows and interactions:



- ✓ Policy Director can be used for user ID/pw authentication instead of LDAP
- ✓ Policy Director adds additional authorization (access control) capability (can user "X" do action "Y" to resource "Z"?)
- ✓ *Users* are WES users
- ✓ *Actions* are read (for GET/POST) and modify (for PUT)
- ✓ *Resources* are URLs

Figure 3-12 WebSEAL-Lite and Policy Director

This section describes WebSEAL-Lite concepts, and also provides a working WebSEAL-Lite configuration example.

Configuring the authentication proxy mode

WebSEAL-Lite runs as a Caching Proxy plug-in; in fact, it runs as Caching Proxy's "authorization proxy" plug-in.

After installation, if WebSEAL-Lite requires a particular configuration setting, it first searches the unique and then the common settings of SecureWay Directory. If the setting is not located within SecureWay Directory, WebSEAL-Lite searches its configuration file, ibmwesas.conf. If the configuration settings is not located in the configuration file, WebSEAL-Lite uses a hard-coded default value for some numeric configuration settings.

There are three configuration files used by WebSEAL-Lite for these settings:

- Base Configuration File (ibmwesas.conf). The base configuration file contains initialization information, and points to the locations of the additional two configuration files. This configuration file also identifies whether WebSEAL-Lite should use Policy Director.
- Object Space Configuration File (osdef.conf) contains information WebSEAL-Lite uses to authorize users.
- User Mapping Configuration File (usermap.conf) is used to map single sign-on and certificate users to Policy Director users.

WebSEAL-Lite may be configured in one of three modes or roles using the 'AuthServerRole' directive in the ibmwesas.conf file:

- Authentication proxy mode: In this mode, WebSEAL-Lite functions as a reverse proxy and issues HTTP 401 challenges whenever Basic Authentication is used. In this configuration, the authentication proxy accepts client requests, then routes those requests to another server. The authentication proxy appears to the client to be the content server, and the client is not aware that the request has been sent to another server. In addition, all SSL (Secure Sockets Layer) connections are terminated at the reverse proxy. You must configure the authentication proxy as a reverse proxy within the ibmproxy.conf file by making use of mapping directives, or by using the WebSEAL-Lite junctioning capability. For more information regarding reverse proxy configuration, refer to the *WebSphere Edge Server Caching Proxy (Web Traffic Express) User's Guide*.
- Transparent proxy mode: In this mode, WebSEAL-Lite functions as a forward proxy and issues HTTP 407 challenges whenever Basic Authentication is used. In this configuration, the authentication proxy accepts client requests and tunnels them through to the intended back-end server. The client must configure an HTTP proxy. In addition, SSL requests are tunneled through to the back-end server. No special proxy mapping or junctioning is required in this mode.

- Detecting mode: In this mode, the authentication proxy dynamically determines, based on the type of request from the browser, whether to treat the request as a reverse proxy or forward proxy request. Using this mode, only a single instance of authentication proxy need be configured to handle both requests for Internet content as well as reverse proxy content.

Configure WebSEAL-Lite for reverse proxy

Configuration for reverse proxy involves providing the mappings from virtual URIs to "real" back-end URIs and servers as well as specifying the type of secure connections required for those back-end connections. Such configuration also requires specifying which destinations require intermediate routing through a proxy.

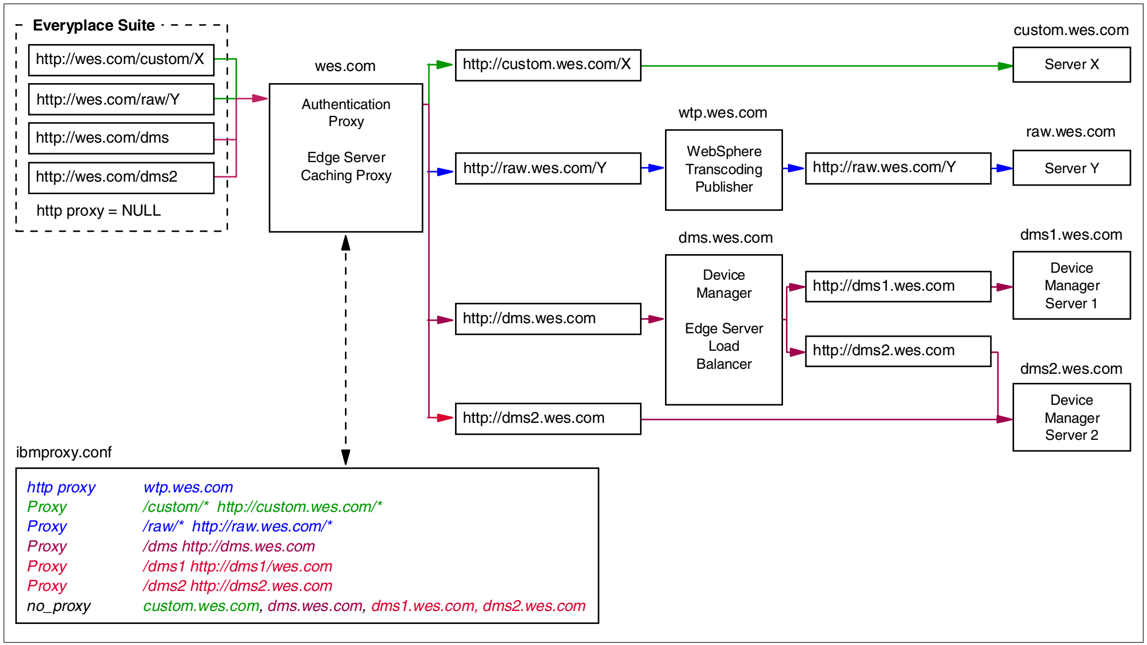
There are two basic ways to configure WebSEAL-Lite as a reverse proxy:

1. Use the WebSEAL-Lite URL mapping, routing, and junction capability.

   This capability is documented in the *WebSEAL-Lite User's Guide* and the sample osdef.conf configuration file. Using WebSEAL-Lite mapping and junctions often greatly reduces the amount of mapping required. Most times, only a single mapping definition is needed for a given back-end Web site. Intermediate proxy routing can also be specified on a junction-basis.

2. Use the Caching Proxies "Proxy" mapping directives in the ibmproxy.conf file.

   Since explicit mappings are required for all URLs that can be accessed on a given Web site, more configuration is required than for WebSEAL-Lite routing. Such control might be desirable. Only a single intermediate proxy can be specified for all back-end servers using the ibmproxy directives, so if more granular routing is needed, use the WebSEAL-Lite proxy routing. Figure 3-13 illustrates a simple configuration and the ibmproxy.conf directives required to achieve it.



```
http proxy      wtp.wes.com
Proxy           /custom/*  http://custom.wes.com/*
Proxy           /raw/*  http://raw.wes.com/*
Proxy           /dms http://dms.wes.com
Proxy           /dms1 http://dms1/wes.com
Proxy           /dms2 http://dms2.wes.com
no_proxy        custom.wes.com, dms.wes.com, dms1.wes.com, dms2.wes.com
```

Figure 3-13 Enabling Applications to work with Reverse Proxy

Setting up applications to work with reverse proxy

WebSEAL-Lite performs authentication and reverse proxy routing for all requests through the Everyplace Server domain. In order to enforce strong authentication and security while enabling single sign-on for applications that reside behind WebSEAL-Lite, all traffic through the Everyplace Server domain must be directed through WebSEAL-Lite via normal Internet routing. Such applications must follow certain rules in order to enable the correct routing, in particular applications using any of the following HTTP/HTML mechanisms:

- BASE tag - specifies the BASE URI to use for referenced documents
- Location: headers in redirection responses - specifies the domain to which HTTP redirection must occur
- Absolute URI references (images, hrefs, etc.) - any full-path reference that includes the server name

You must specify in these references the name of the WebSEAL-Lite proxy server or the WebSEAL-Lite server cluster name (if Load Balancer is used).

After WebSEAL-Lite has successfully been installed and configured, you may issue requests from your Web browser to Caching Proxy using it either as a proxy, or by specifying the domain name of the Caching Proxy machine in your URL as shown below:

```
http://<domain name of Caching Proxy>
```

However, since the object space has not been configured in Policy Director at this point, you will not be allowed to retrieve this request.

Configuring the WebSEAL-Lite proxy

Typically, when a user issues a request to a Web site using a browser, the object represented in the URL corresponds to an object on a Web server. WebSEAL-Lite provides access control by verifying (using Policy Director's aznAPI interface) that the user is allowed to access the requested object on the Web server, before allowing the request to complete. The configuration is slightly different if WebSEAL-Lite is providing access control in reverse proxy mode, or in forward proxy mode.

Reverse proxy

Since Web site content may span multiple Web servers for performance and content distribution, Caching Proxy may be used as a reverse proxy to the internal or back-end Web servers. This is accomplished by configuring the Web site public domain name on the Caching Proxy machine, and specifying a route to the corresponding back-end Web server, as illustrated in Figure 3-14.

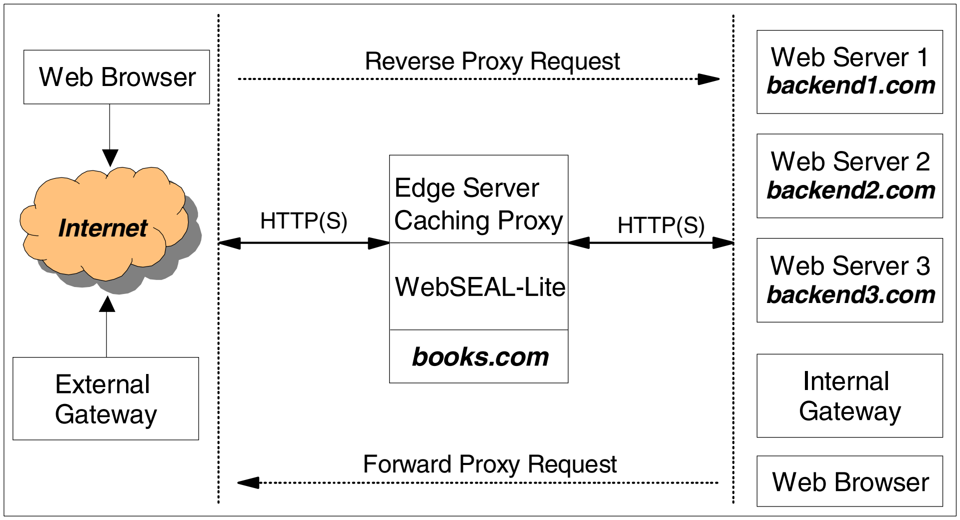


Figure 3-14 Reverse and forward proxy requests with WebSEAL-Lite

In this example, WebSEAL-Lite would be configured to provide access control to the objects on books.com. After a user had been successfully authorized, the request would be routed to the corresponding back-end server either by WebSEAL-Lite, or through the use of a load balancing module, such as the Network Dispatcher Content Based Routing module. WebSEAL-Lite performs simple URL mapping, similar to the functionality provided by the proxy statements in the Caching Proxy configuration file.

WebSEAL-Lite access control is configured through its object space configuration file, osdef.conf. In this file, you would add the following entry to configure this Web site:

```
[Remote: /WebSEAL-Lite/reverse/books.com]
domains = books.com www.books.com
login_method = forms
form_login_file = http://books.com/pub/login.html
form_login_errorfile = http://books.com/pub/loginerr.html
form_logout_file = /account/logout.html
route = https://backend1.com
```

This entry tells WebSEAL-Lite to authorize all requests for books.com and www.books.com using `/WebSEAL-Lite/reverse/books.com` in the Policy Director object space, to use forms as the login method, and to map every URL to the same Web server. By default, WebSEAL-Lite checks `/WebSEAL-Lite/reverse/<domain name>` for reverse proxy requests. There are other options that you could associate with this server definition. If you do not explicitly specify a setting for an option, the setting for that option is inherited from the `[Global]` section of the configuration file.

Although forms was chosen as the login method in this example, there are other settings that may be chosen for login_method. For example, basic may be selected for basic authentication and certificate may be selected for client certificate authentication. All of these settings, as well as options specific to each setting are described in more detail in the sample configuration file, osdef.conf.

Forward proxy

WebSEAL-Lite may also be used to provide access control to outbound requests, as illustrated in Figure 3-14. Users on the internal side of the firewall could configure their browsers to use the same instance of Caching Proxy to get to the Internet. By default, WebSEAL-Lite checks `/WebSEAL-Lite/forward/<domain name>` for forward proxy requests. But you can explicitly override this by creating a server definition in the object space configuration file as shown below:

```
   [Remote: /WebSEAL-Lite/forward/blockedsites]
   domains = games.com *.games.com *.competitor.com
   route = http://backend2.com /pub/browsepolicy.html
```

In this example, all browser requests matching the above domain names would be redirected to the company browsing policy Web page. Alternatively, you could place an ACL at this location in the object space that would prevent anyone from going to the listed Web sites.

Configuring your object space

In order for you to specify what objects users are allowed to access on a Web server, ACLs must be associated with the objects provided by the Web server. This means that the Web server object hierarchy needs to be represented in the Policy Director object space. The simplest way to do this is to import the Web server file system into the Policy Director object space. After this has been done, ACLs may be placed on the desired objects. The WebSEAL-Lite Object Space Manager, wesosm, generates the object space for Caching Proxy, or for other Web servers.

Although Caching Proxy is a proxy, it can function as a Web server when requests are made directly to the primary domain name of the Caching Proxy machine. Typically, informational and error messages are stored in the proxy Web space. WebSEAL-Lite will enforce access control to these objects managed by Caching Proxy. The following server definition in the configuration file is used to represent Caching Proxy:

```
[Local: /WebSEAL-Lite/bookproxy.com]
domains = bookproxy.com
query_command = http://bookproxy.com/cgi-bin/query_contents?dirlist=/
```

The configuration tool executes wesosm to generate the object space for Caching Proxy as its final step in configuring WebSEAL-Lite. After running the configuration tool, you should be able to place ACLs at the appropriate locations in the object space for Caching Proxy. For example, informational and error pages should not require authorization for users to access.

Device type recognition

WebSEAL-Lite identifies devices associated with incoming requests based on mapping rules defined in the device profiles stored in SecureWay Directory (see the *WebSphere Transcoding Publisher Administrator's Guide* for information about configuring device profiles and the device mapping rules). If WebSEAL-Lite cannot determine a device type based on the available mapping rules, WebSEAL-Lite inserts a default network or device type into the request header. The default network and device type are stored in the file where the SecureWay Directory parameters are stored.

Three default network types are used:

Default network	All IP traffic from network or third-party gateways
Wireless network	Wireless traffic from the Wireless Gateway

Dial-up network Dial-up traffic from the Wireless Gateway

Devices are recognized using the HTTP *User-Agent* header. This recognition will be performed by the WebSEAL-Lite using the shared WebSphere Transcoding Publisher configuration stored in the LDAP directory.

A WebSEAL-Lite deployment example

In this section, a complete WebSEAL-Lite configuration for books.com will be designed. This Web site will allow users to browse and purchase books. Many of the key features of WebSEAL-Lite will be illustrated in this example. The Web site design will be divided into the following components:

- Content distribution
- Single sign-on

Content distribution

In this design, rather than storing the entire content for the Web site on one Web server, it will be distributed across several Web servers. It will be divided into the following sections:

- The /home directory will contain the greeting page, and links to other parts of the Web site.
- The /catalog directory will contain a repository of all the books sold at this Web site.

These sections of the Web site will not be protected (not require access control).

- The /account directory will contain HTML and Java code to manage the user accounts.
- The /payment directory will also contain HTML and Java code to receive the user payments for books to be purchased.

Most of these sections of the Web site will be protected. A directory underneath /account will be used to register new users. This directory will not be protected.

An alternative way of designing this Web site would be to assign a public sub-domain name to each section of the Web site. For example, instead of storing the catalog underneath /catalog, it would be stored on catalog.books.com. However, this method requires additional administration to allocate a sub-domain name for each section of the Web site.

Single sign-on

In this design, an application running on the Web server hosting the /account directory will require an encrypted LTPA cookie to identify the authenticated user. Another application running on the Web server hosting the /payment directory will require the HTTP header, App-User with the user ID in it. It will also require basic authentication from WebSEAL-Lite as the trust basis for accepting the authenticated user. WebSEAL-Lite will be required to authenticate itself to this application with the user ID, Weblite, and the password, bookworm.

In this example, a relationship has been established with another vendor, novels.com, to forward authenticated users through WebSEAL-Lite to protected areas of books.com. The gateway at novels.com will be required to authenticate itself to WebSEAL-Lite using an authorization header with the user ID, novelgateway, and the password, bookworm. The gateway will place the authenticated user ID in the cookie, Novel-User, as illustrated in Figure 3-15.

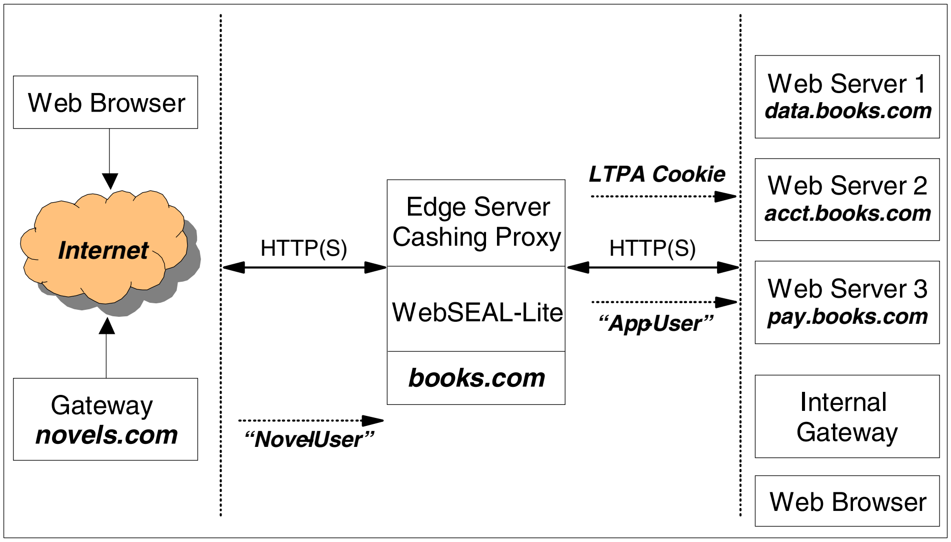


Figure 3-15 Web site design for books.com

Configuring the Web site

In order to provide access control for books.com, WebSEAL-Lite will need to be configured. The configuration will begin by defining the global settings for WebSEAL-Lite, as shown in Example 3-1.

Example 3-1 WebSEAL-Lite - Global configuration

```
[Global]
# Administrator userid and password needed to run wesosm
update_admin_userid = sec_master
update_admin_password = secret5
```

```
# Error message indicating that SSL is required
require_ssl_errorfile = /opt/pdweb-lite/etc/require_ssl.htmls
```

The [Global] definition in the configuration file specifies settings that apply to every request handled by WebSEAL-Lite. In the above configuration, the administrator user ID and password is provided so that the wesosm utility can create and update the object space. Also, if a request requires a secure connection and one is not provided, the specified error page will be returned to the user. However, if possible, the browser will automatically be redirected to the secure site.

The [Local] definition in the configuration file specifies settings that apply to objects on the file system managed by Caching Proxy. There should only be one of these definitions in the configuration file. This server definition should already have been created by the configuration tool, and will be similar to that shown in Example 3-2.

Example 3-2 Sample WebSEAL-Lite - Local configuration

```
[Local: /WebSEAL-Lite/bookproxy.com]
domains = bookproxy.com
query_command = http://bookproxy.com/cgi-bin/query_contents?dirlist=/
```

In Example 3-2, bookproxy.com is the primary domain name of the machine on which the Caching Proxy is running. The alias, books.com, is another domain name with its associated IP address, assigned to the same machine. WebSEAL-Lite differentiates between requests for objects belonging to Caching Proxy and objects belonging to books.com by matching the requested domain name with a server definition in the configuration file. The domains attribute indicates which domains a server definition applies to.

The [Remote] definitions in the configuration file specify settings that apply to external Web servers. There is no limit on the number of server definitions you can have. For this example, the definition shown in Example 3-3 would be appropriate for books.com.

Example 3-3 Sample WebSEAL-Lite - Remote configuration

```
[Remote: /WebSEAL-Lite/reverse/books.com]
domains = books.com www.books.com
# Form login and logout information
login_method = forms
form_login_file = http://books.com/home/login.html
form_login_errorfile = http://books.com/home/loginerr.html
form_logout_file = /account/logout.html
# Change password information
form_chgpasswd_file = http://books.com/home/chgpasswd.html
```

```
form_chgpasswd_submit_url = /account/chgpasswd
# Single sign-on tokens to look for accept_sso = NovelSSO
# Server to map requests and initial page
# Browser will be redirected to greeting page
route = http://data.books.com/home /home/index.html
```

This configuration specifies the login method as forms, and lists the login forms. The login forms may be retrieved from a remote Web server (begins with "http"), or may be retrieved from the local file system. If the form has references to images in it, the URL links in the forms for these images should contain the full path of the images (such as "/home/gifs/banner.gif"). The user will be logged out when the requested URLs path matches the path specified for the logout file. Also, as the configuration shows, a change password form will be sent to authenticated users when their passwords expire.

Single sign-on users will be accepted from the gateway at novels.com, using the NovelSSO single sign-on definition. This single sign-on definition must be defined to the configuration file. Finally, the initial request will be mapped to the /home section of the Web site, by redirecting the browser.

The [Junction] definitions in the configuration file specify settings that apply to virtual directories for this Web server. Each of these definitions represents a virtual directory that inherits its settings from its parent ([Remote]) definition as shown in Example 3-4.

Example 3-4 Sample junction for Books.com

```
[Junction: /WebSEAL-Lite/reverse/books.com:/home]
route = http://data.books.com/home
[Junction: /WebSEAL-Lite/reverse/books.com:/catalog]
route = http://data.books.com/catalog
[Junction: /WebSEAL-Lite/reverse/books.com:/account]
query_command = http://acct.books.com/cgi-bin/query_contents?dirlist=/
require_ssl = yes
submit_sso = LTPA-COOKIE
route = https://acct.books.com
[Junction: /WebSEAL-Lite/reverse/books.com:/payment]
query_command = http://pay.books.com/cgi-bin/query_contents?dirlist=/
require_ssl = yes
submit_sso = PayAppSSO PayAppAuth
route = https://pay.books.com
```

This configuration specifies the single sign-on tokens to submit to each Web server that requires one. It also tells WebSEAL-Lite to map the URL to the corresponding Web server hosting the requested content. If another routing module is to be used in place of the WebSEAL-Lite URL mapping, simply delete all references to the route key word from the configuration file.

The [SSO] definitions in the configuration file define single sign-on tokens that may either be accepted or submitted, as shown Example 3-5.

Example 3-5 Sample single sign-on tokens

```
[SSO: PayAppSSO]
type = header
name = App-User
format = <userid>
[SSO: PayAppAuth]
type = auth_header
format = weblite:bookworm
[SSO: NovelSSO]
type = cookie
name = Novel-User
format = <userid>
trust_basis = basic_auth
trust_list = novelgateway:bookworm
```

After these single sign-on definitions have been created, they may be supplied as parameters to the accept_sso and submit_sso keywords. As illustrated in this configuration, all single sign-on requirements described previously have been satisfied.

3.3.6 IBM WebSphere Transcoding Publisher

The overall architecture of WebSphere Transcoding Publisher is shown in Figure 3-16.

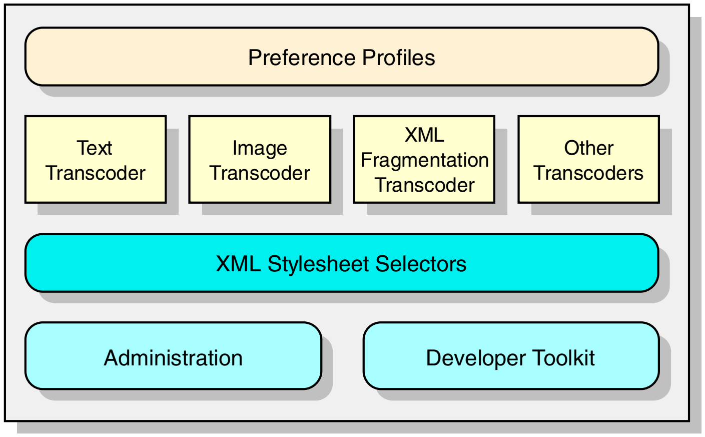


Figure 3-16 IBM WebSphere Transcoding Publisher architecture

A number of *preference profiles* are responsible for recognition of device, network and content types and to apply the right set of transcoders.

WebSphere Transcoding Publisher acts as a host for a number of *transcoder plug-in* units that can modify any type of content in practically any possible way. The default set of transcoders can be extended by custom-written transcoders implemented in Java.

XML content can be matched to a specific XSL by the *XML Stylesheet Selectors*. If the selection criteria is a match, the stylesheet will be applied to the XML content.

WebSphere Transcoding Publisher in a WebSphere Everyplace Server environment uses the LDAP directory for settings (otherwise, often stored in *.properties files). Settings are shared between all instances of WebSphere Transcoding Publisher by using a common LDAP schema.

Two additional WebSphere Everyplace Server-specific features in WebSphere Transcoding Publisher are:

- Everyplace Server Active Session support - to bypass device/network recognition if WebSEAL-Lite has already done so.
- *NO-OP* HTTP header support - the first transcoder will signal “job already done” to the following ones, which then bypass transcoding.

In the Everyplace Server WebSEAL-Lite will perform network and device detection and store the result in an HTTP header for use by WebSphere Transcoding Publisher and other subsequent servers (see “Device type recognition” on page 98).

WebSphere Transcoding Publisher uses two basic types of transcoders:

- MEGs

  A *MEG* is a content Monitor, Editor or Generator. It is the most flexible and powerful building block making up a transcoder. It can handle almost any type of content and do any operation to it. In the WebSphere Transcoding Publisher implementation it takes the shape of a Java Servlet, and is then called a *MEGlet*.

  For more information on the MEG concept in general see:

  `http://www.almaden.ibm.com/cs/wbi`

- XSL transforms

  More and more content is conforming to some XML variant. A specialized way of handling XML input is XSL. An XSL stylesheet can do almost any transform of any XML document, but it is best suited for working at the field and structural level (in contrast to the text character level). The output can be another XML or any non-XML text format, while binary formats in general are not suitable for XSL transforms.

  The XML Stylesheet Selector will choose the proper XSL based on the DTD or schema for the incoming XML.

The first thing to consider is whether to use Java-based MEGlets or XSL. Technically both techniques can help you do almost any transcoding, but one is normally easier to work with. Which one depends heavily on the input type:

Table 3-2 Preferred transcoding techniques

Input content type	Preferred technique	Comments
Any well-formed XML	XSL	For both XML and non-XML output, XSL is designed to do exactly this conversion.
HTML or other non-XML	Primarily MEGlet	Quite complex using XSL, so Java using WebSphere Transcoding Publisher helpers is the choice for text clippers or other transforms.
Binary (such as images)	MEGlet	Java is very suitable, due to availability of existing helper libraries.

You may want to supply your own transcoders or stylesheets to:

- Provide generic support for additional device types
- Do tailored conversion for a specific application to archive optimal results
- Transform specific XML variants

Preference profiles

The determination of which transcoders to invoke for a specific request and its corresponding reply will be done by preference profiles stored and shared in an LDAP directory. The collection of profiles is organized in three groups with a number of profiles initially defined:

Devices profiles: Windows CE, Palm Pilot, WAP phones, Netscape Navigator (desktop), Microsoft Internet Explorer 4 (desktop), XML-capable desktop browsers, and a default.

Network profiles: Wireless network, dial-in network, and network default.

User profiles: (Not used)

Selection of device is based on the HTTP User-Agent field. Selection of the network is based on the selected incoming IP port number.

For most profiles a number of parameters can be configured to enable or adjust the transcoders it will invoke. The selection of configurable parameters is set up when creating a new preference profile and cannot be changed later. Refer to the online *WebSphere Transcoding Publisher Developer's Guide* for information on how to re-create and register the profile again.

Included in WebSphere Transcoding Publisher is an administration console and a toolkit for development and problem determination. A number of tools are provided in WebSphere Transcoding Publisher's Toolkit:

- To easily see the effect of transcoding, the *Transform Tool* provides a split-screen view showing original content and transcoded content side by side. This partly eliminates the need for acquiring a lot of different devices or device emulators. The tool can also be used as a working example of how to use the transcoding JavaBeans in an application. Using the current settings of WebSphere Transcoding Publisher, the tool will show transcoding of both text and images.
- When creating and registering MEGs, the *Request Viewer* can give you a visualization of the operation of the transcoding server. You can view registration and configuration information of the MEGs. Request Viewer also enables you to monitor the flow of requests through the server and observe which plug-ins are triggered and when they are triggered. For each transaction, the Request Viewer also displays the header and content information as they are manipulated by the plug-ins. Notice that WebSphere Transcoding Publisher and the Request Viewer cannot be running at the same time, as the tool will run a complete debug version of WebSphere Transcoding Publisher using the same ports.
- The *Snoop tool* is a MEGlet generator similar to the Snoop servlet known from the IBM WebSphere Application Server. It will display all HTTP headers in both request and reply. This is a very easy way of spotting the User-Agent for any device or browser (the Request Viewer could also be used for that).
- A *Preference Profile Creator*.

Basic flow for Transcoding Publisher acting as a proxy

The basic flow for Transcoding Publisher acting as a network intermediary (transcoding proxy) is shown in Figure 3-17.

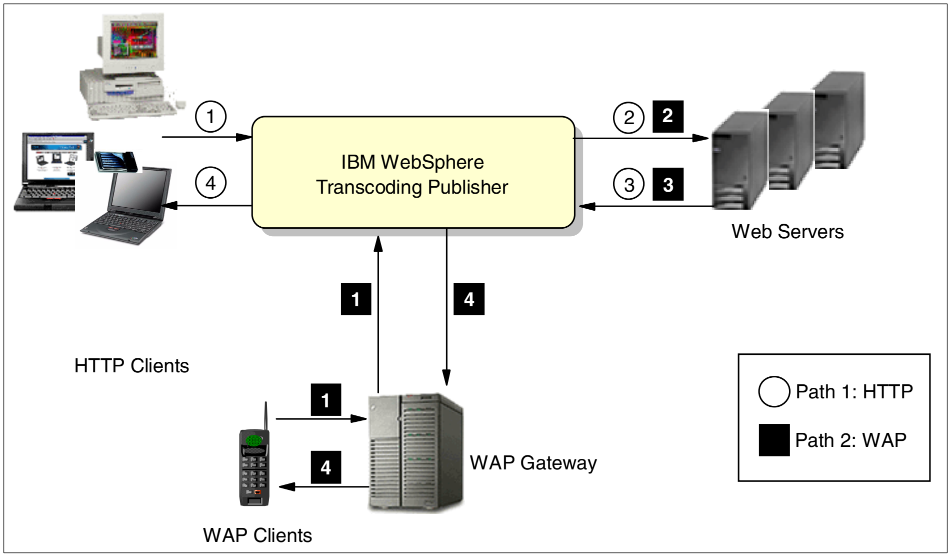


Figure 3-17 WebSphere Transcoding Publisher proxy basic flow - HTTP and WAP

In the diagram shown in Figure 3-17, the arrows indicate the flow of information to and from the transcoding proxy. The flows are numbered to show different paths through the transcoding proxy as follows:

Path 1 in Figure 3-17 shows the transcoding proxy acting on behalf of an HTTP-based client browser that expects HTML or XML pages in return.

1. The client browser sends an HTTP request to the transcoding proxy. The request can be for any Web documents, including HTML pages, XML pages, or GIF or JPEG images.
2. The transcoding proxy can edit the request to modify the URL or to change values in the header fields. It then sends the HTTP request to the Web server to acquire the requested object. The proxy also saves information from the request that identifies the device making the request.
3. The Web server returns the page requested by the transcoding proxy on behalf of the client.
4. The transcoding proxy uses information from the request and response along with rules in the Preference Aggregator to identify the device, user, and network profiles needed to control the way the document is transcoded. When particular transcoding modules running in the transcoding proxy request the values of particular preferences or constraints, the Preference Aggregator uses the profile identification information in evaluating its rules to help determine the correct value to return. Using these preference and constraint values, the transcoding proxy edits the response received from the

Web server, tailoring it for the client device before returning it to the client in an Tivoli Personalized Services Manager HTTP reply.

Path 2 in Figure 3-17 shows the transcoding proxy acting on behalf of a Wireless Application Protocol (WAP) client, such as a smart phone.

1. The wireless client sends a WAP request that is converted to an HTTP request by the WAP Gateway. The WAP Gateway forwards the HTTP request to the transcoding proxy.
2. The transcoding proxy sends the HTTP request to the Web server after any necessary editing.
3. The HTTP reply comes back from the Web server.
4. When the transcoding proxy matches information from the request against device and user profiles, it discovers that the device requires Wireless Markup Language (WML) output. For this example, assume the original document was an XML document. The transcoding proxy uses the preference information to select the correct style sheet to convert the document to its WML form. Then it returns the response to the WAP Gateway, which converts the text-based WML documents to the compressed wireless-ready form before sending it to the client smart phone.

Sample scenario

In this section, we present a sample scenario that shows how you can deploy a WebSphere Transcoding Publisher proxy in the WebSphere Everyplace Server environment. In this scenario, the WebSphere Transcoding Publisher proxy transcodes application content (HTML and XML) from our sample WebSphere Application Server applications. However, other Web applications not using WebSphere Application Server are also transcoded.

Figure 3-18 illustrates the scenario as follows:

- Two client devices, a desktop and a WAP simulator, are connected to the WebSphere Everyplace Server Authentication Server via HTTP.
- The Authentication Server installed in authentication proxy (AP) mode and configured as a reverse proxy.

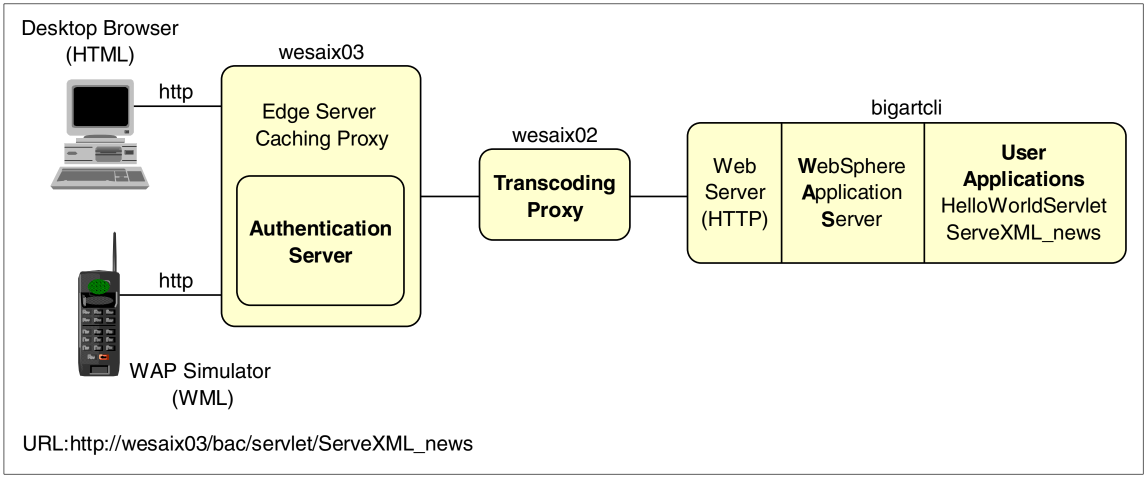


Figure 3-18 Adding transcoding proxy to an Everyplace Server environment

Proxy configuration in WebSEAL-Lite

When running the Authentication Server in AP mode, you need to create two proxy statements in the ibmproxy.conf file that tell the Edge Server Caching Proxy used by the Authentication Server that the request will use a proxy (transcoding proxy in this case) after it goes through the Authentication Server. The directives in Edge Server configuration file are:

- The http_proxy directive is used for requests that pass through the proxy and specifies the host name of the (transcoding) proxy instance that is used as the default Everyplace Suite domain transcoding proxy.
- The no_proxy directive is used for requests that do not pass through the (transcoding) proxy, for example Tivoli Personalized Services Manager Device Manager traffic. The no_proxy directive specifies the domains to which the server should directly connect. This directive does not apply when the proxy goes through a SOCKS server; use socks.conf for that purpose. Also, if you are using neither proxy chaining nor SOCKS, then this directive is not needed.

Next we configure the ibmproxy.conf file on the Caching Proxy, where the Everyplace Server Authentication Server is running as a plug-in, to have the WebSphere Transcoding Publisher act as a proxy server.

In this configuration, the Authentication Server proxy accepts client requests, then routes those requests to the http_proxy (the transcoding proxy), and then the request will go to the application content server.

Note that the authentication proxy appears to the client to be the application content server, and the client is not aware that the request has been sent to another server.

The http_proxy directive is used to specify the name of another proxy Web server. This server should be contacted for HTTP requests rather than contacting the HTTP Server named in the request URL directly.

In our scenario, we specified the transcoding proxy to be:

```
http_proxy http://wesaix02.raleigh.ibm.com:8089/
```

Our request will go to the Authentication Server, wesaix03, then to the http_proxy pointed to by the http_proxy directive in the configuration file, wesaix02, and then it will go to the application server pointed to by the alias back in the configuration file, http://bigartcli/*, and access the application in /servlet/ServeXML_news, where “servlet” is the alias used for default WebSphere Application Server servlets.

3.3.7 Policy Director

Policy Director is a general-purpose authorization and authentication product. Policy Director can be implemented stand-alone, and it can also be implemented within an IBM WebSphere Everyplace Server environment. Everyplace Server uses a subset of the Policy Director functions. This section describes how Policy Director is implemented in the WebSphere Everyplace Server environment.

Policy Director is used by WebSEAL-Lite to provide authentication and authorization, and by Location Based Services to provide user privacy information.

Access control using Policy Director is achieved by integrating the following components of access control:

- Users and Groups represent the individuals that the access control will be applied to. This is the first step in providing access control.
- Access Control Lists (ACLs) are lists of users and groups along with their associated permissions. An ACL says nothing about what object a users permissions are being applied to. It simply states that a collection of users and groups have the specified respective permissions. Users and groups are not bound to one ACL. A user or group may appear in one or more ACLs. This is the second step in providing access control.
- Object Space. This represents the final piece of information needed to provide access control: the objects that the user will be accessing. Policy Director represents objects using a hierarchical representation called the Object Space, similar to the way an operating system represents files and

directories on a file system. Access control is achieved by placing ACLs at the appropriate locations in the object space. The interpretation of each object in the object space along with the associated ACL is subject to each application. Policy Director simply tells the application what ACLs are associated with a given object in the object space.

To define your Policy Director environment, you must define the users, the groups, the ACLs, the object space, and their relationships.

A full Policy Director installation uses the following server processes (daemons):

- Management Server (ivmgrd). The Management Server manages the master authorization (ACL) database and maintains location information about other WebSEAL and NetSEAL servers in a secure domain.
- WebSEAL (secmgrd). WebSEAL is not used in the WebSphere Everyplace Server environment. Instead its functions are performed by the WebSEAL-Lite component.
- NetSEAL (netseald). NetSEAL is not used in the WebSphere Everyplace Server environment.
- Authorization Server (ivacld). The Authorization Server allows third-party applications to make authorization calls (via the Authorization API) to the Policy Director security service.
- Directory Service Broker (dsb). This is not used in the WebSphere Everyplace Server environment.

Figure 3-19 illustrates the Policy Director components in a non-WebSphere Everyplace Server environment.

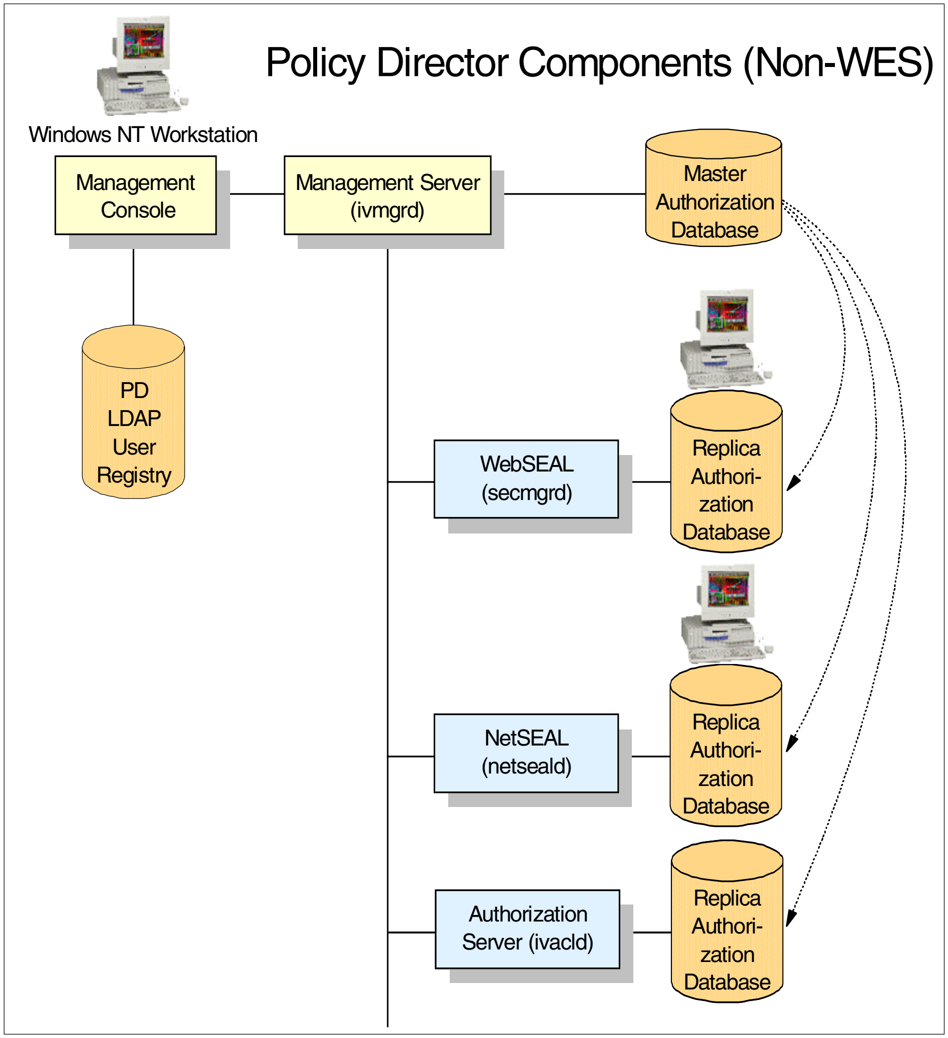

Policy Director Components (Non-WES)
Windows NT Workstation
Management Console
Management Server (ivmgrd)
Master Authorization Database
PD LDAP User Registry
WebSEAL (secmgrd)
Replica Authori-zation Database
NetSEAL (netseald)
Replica Authori-zation Database
Authorization Server (ivacld)
Replica Authori-zation Database

Figure 3-19 Policy Director components in a non-Everyplace Server environment

Figure 3-20 shows the Policy Director components in a WebSphere Everyplace Server environment. Note that, depending on installation choices, the Policy Director LDAP user registry shown in the diagram may be one of the following:

- A separate SecureWay Directory server from that containing the WebSphere Everyplace Server user directory.
- The same SecureWay Directory instance, but with the Policy Director information in a separate Directory Information Tree (DIT) from that used by WebSphere Everyplace Server.

- Or the sameSecureWay Directory instance, with the Policy Director user information merged into the same DIT as the WebSphere Everyplace Server DIT. Please see “Policy Director’s Directory” on page 136 for more information about this.

Thus, in the WebSphere Everyplace Server environment, only two Policy Director installation packages are required: IVBase, and IVMgr. In addition, you may require IVConsole to give you a Policy Director management console.

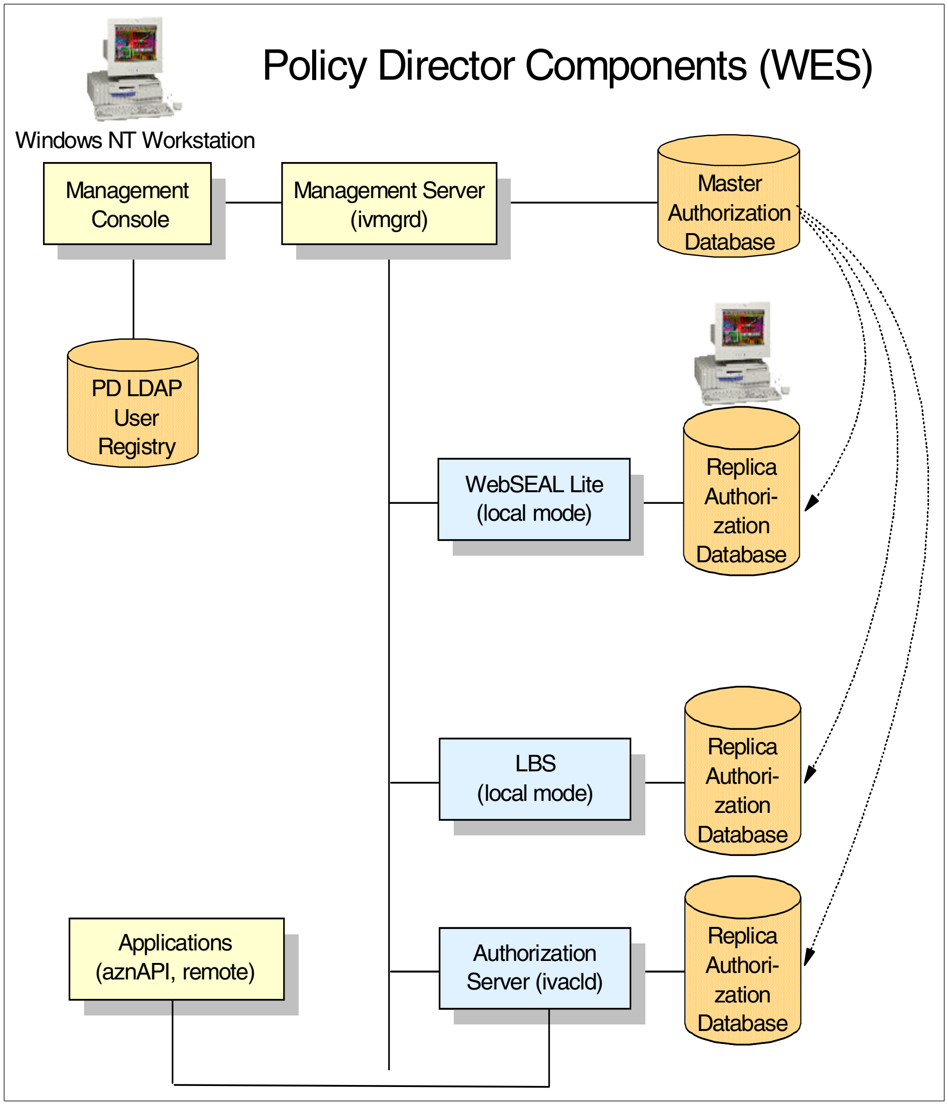

Policy Director Components (WES)
Windows NT Workstation
Management Console
Management Server (ivmgrd)
Master Authorization Database
PD LDAP User Registry
WebSEAL Lite (local mode)
Replica Authori-zation Database
LBS (local mode)
Replica Authori-zation Database
Applications (aznAPI, remote)
Authorization Server (ivacld)
Replica Authori-zation Database

Figure 3-20 Policy Director components in an Everyplace Server environment

Important Policy Director server dependencies include the following:

- There can be only one instance of the Management Server and the master authorization (ACL) database in any secure domain
- The Management Server replicates the authorization database to all other Policy Director servers in the secure domain
- Each resource manager (WebSEAL, NetSEAL, Authorization Server) applies access control policy based on information from the replicated authorization database

Applications that use aznAPI may choose to run in “local” or “remote” mode:

- When running in local mode, applications maintain a replica of the authorization policy database. This replica is acquired either at startup time or at the first request, depending on configuration options. The application may register itself with Policy Director in order to receive updates to the ACLs automatically, or may poll Policy Director at regular intervals to receive changes. However, these applications use the local cache for their own purposes only, and do not allow other applications to call them. Therefore, they cannot be considered to be Policy Director authorization servers.
- In remote cache authorization mode, applications use the function calls provided by the Authorization API (aznAPI) to communicate to the Authorization Server (ivacld). The Authorization Server maintains a replica of the authorization policy database and functions as the authorization decision-making evaluator.

In the WebSphere Everyplace Server environment, WebSEAL-Lite and Location Based Services both run in local mode. Each of them caches a copy of the Policy Director authorization data.

IBM SecureWay Directory installation is not integrated with the WebSphere Everyplace Server installation programs in this release. IBM SecureWay Directory is also not configurable from WebSphere Everyplace Suite Manager.

Prior to install, Policy Director requires a working DCE environment in order to function. IBM DCE V3.1.0.1 for AIX is shipped on the CD with Policy Director V3.7. The DCE components required for the WebSphere Everyplace Server Policy Director install are a security server and a cell directory server. DCE is installed only within the Policy Director server complex and is not used by any of the other WebSphere Everyplace Server components. Additional DCE information is available at the IBM Web site at the following URL:

http://www.ibm.com/software/network/dce/

In addition, there is more information about implementing Policy Director and DCE in *Tivoli SecureWay Policy Director Centrally Managing e-business Security*, SG24-6008.

3.3.8 Tivoli Personalized Services Manager

Tivoli Personalized Services Manager is an infrastructure for enabling and managing the delivery of Internet-based services and the customers of those services. Tivoli Personalized Services Manager is made up of:

- Personalization services, including but not limited to, enrollment, Customer Care, and Self Care
- Device Manager Server (DMS)

Figure 3-21 depicts the interfaces between Tivoli Personalized Services Manager and the other WebSphere Everyplace Server components.

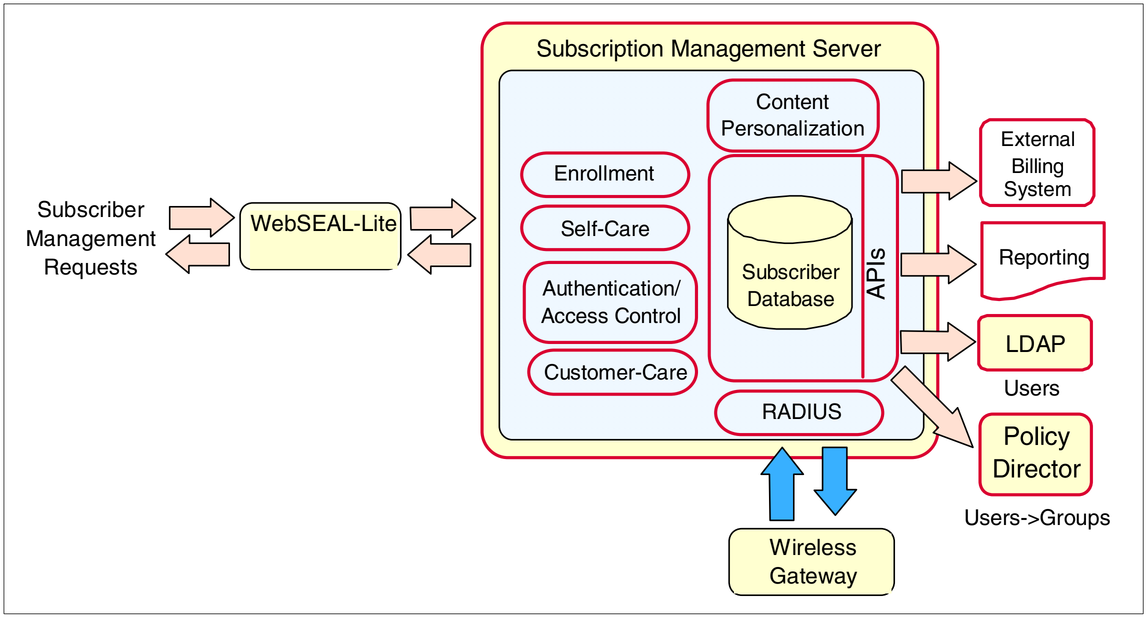


Figure 3-21 Tivoli Personalized Services Manager's Everyplace Server interfaces

Tivoli Personalized Services Manager provides APIs via its toolkits (iTK) to allow you to interface external billing systems, reporting systems, and to provision LDAP, Policy Director, and any other systems of your choice. Samples are provided to show usage of the APIs.

Provisioning interfaces are used to mirror subscribers into the IBM SecureWay Directory (LDAP) and into Policy Director (if implemented in the WebSphere Everyplace Server domain). This allows any system to look up subscriber information using either LDAP, Policy Director, or one of the Tivoli Personalized Services Manager interfaces.

Important: The Tivoli Personalized Services Manager database is the master directory from which the other directories are provisioned, so any update of subscriber information must be done in Tivoli Personalized Services Manager - not using LDAP.

Tivoli Personalized Services Manager receives billing information from the Wireless Gateway via the RADIUS server. This is a duplicate of the billing information captured in the Wireless Gateway's accounting database. Currently, this is the only billing information captured within the WebSphere Everyplace Server environment. This billing information, stored in Tivoli Personalized Services Manager's billing database, may be extracted via the Tivoli Personalized Services Manager billing API.

Figure 3-22 is a high-level view of Tivoli Personalized Services Manager's internal design.

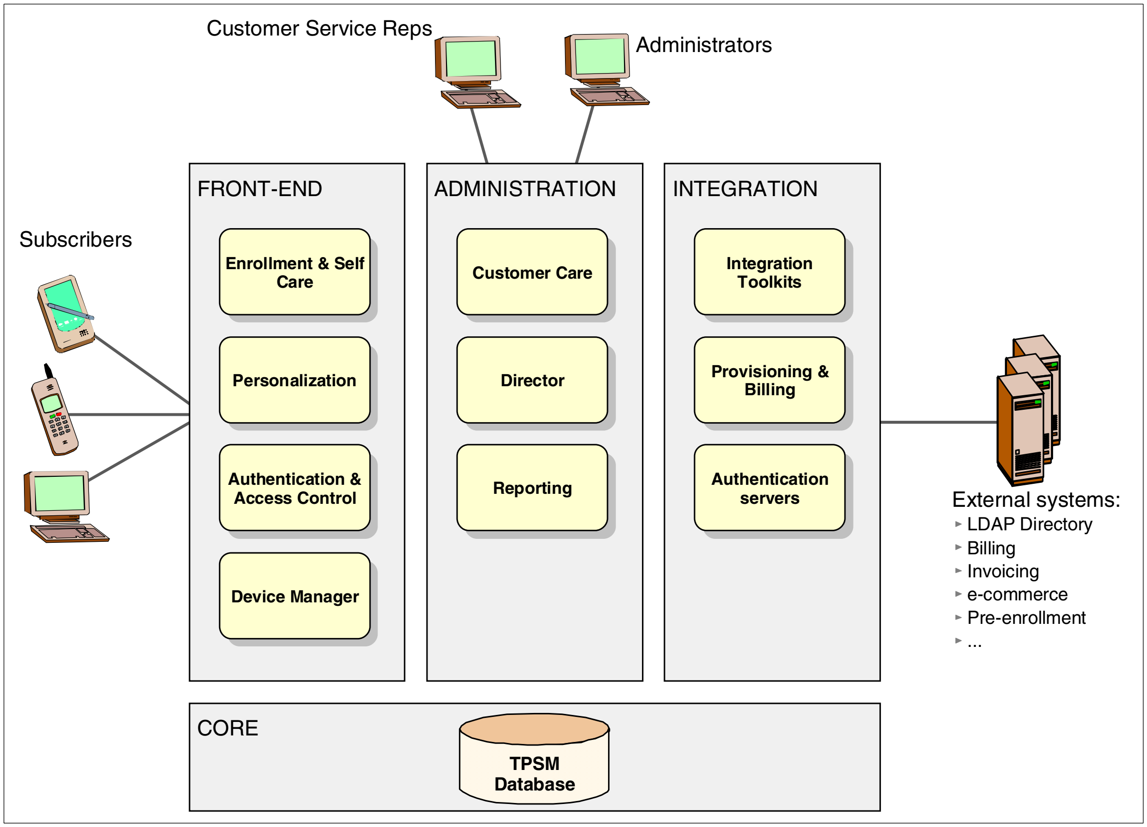


Figure 3-22   Tivoli Personalized Services Manager's internal architecture

The Tivoli Personalized Services Manager "front-end" functions and the Administration Customer Care function in Figure 3-22 are implemented as WebSphere Application Server applications. These applications are initially provided in HTML format only. Although they can be placed behind a WebSphere Transcoding Publisher instance and transcoded for presentation to wireless devices, the screens were not designed with this in mind. This may lead to poor usability. It is therefore recommended that custom versions of these applications be developed for wireless use.

Tivoli Personalized Services Manager provides a "bridge" process that automatically creates, updates, and deletes user information in Policy Director based on corresponding changes in Tivoli Personalized Services Manager (Enroll, place a Customer Order, change Deal, is deleted, etc.) and also associates the user with the correct authorization groups in Policy Director. You must edit the pd_admin.ksh script file to configure and customize the bridge.

To use Tivoli Personalized Services Manager provisioning, you need to understand the data model of your target system or application, and how it relates to Tivoli Personalized Services Manager's data model. Figure 3-23 gives a high-level overview of Tivoli Personalized Services Manager's data model.

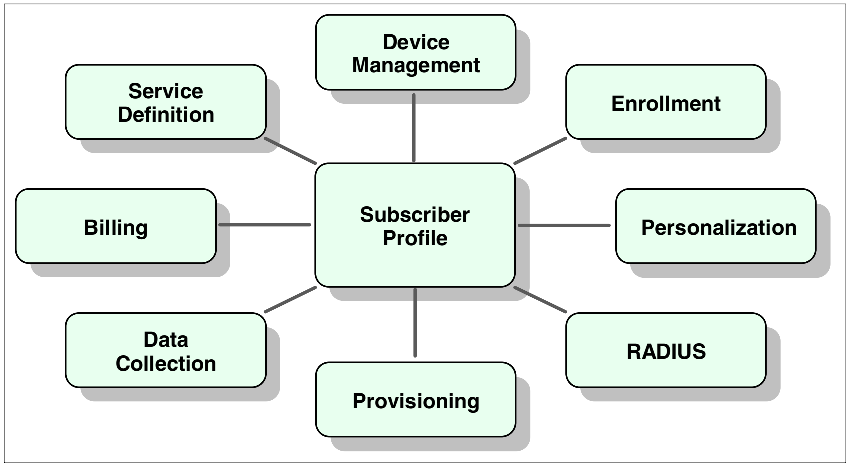


Figure 3-23 Tivoli Personalized Services Manager data model overview

As you can see, the Subscriber Profile is the central concept within the Tivoli Personalized Services Manager data model. The information stored in this profile is reflected in each of the other WebSphere Everyplace Server directories, although in different forms in each instance. Figure 3-24 illustrates the relationships between the Tivoli Personalized Services Manager data types.

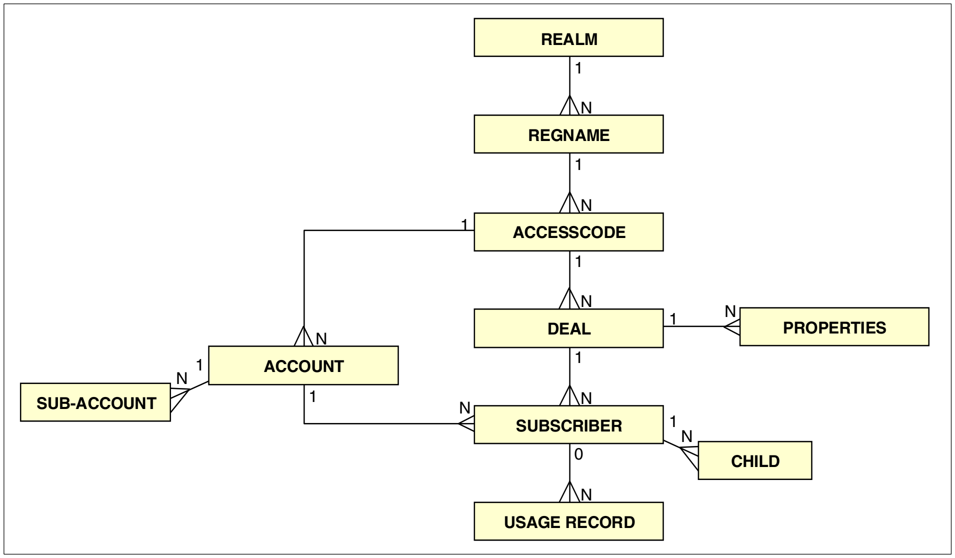


Figure 3-24 Tivoli Personalized Services Manager core data model

The *realm*, *subscriber* and *account* profile metadata is the core of the model. They are related via elements relevant during subscription and billing, such as *Accesscode* and *Deal*.

The *realm* is the top level administrative domain. The scope of a *realm* is dual - it serves as both the administrative domain as well as the domain used for server and subscriber name spaces (for example `ibm.com` with subscriber `lou@ibm.com`). There must be at least one *realm* defined in a Tivoli Personalized Services Manager implementation. By defining multiple *Realms*, an ISP or similar can enable virtual hosting with multiple domain names (such as `a.com`, `b.com`, `c.com`, etc.) on a single installation, and also allow delegated administration.

The realm maps to *organizational unit* in the WebSphere Everyplace Server LDAP and in the Policy Director LDAP directories. *Subscriber* maps to *uid* in the WebSphere Everyplace Server LDAP directory, and to *cn* in the Policy Director directory.

3.3.9 Location Based Services

As shown in Figure 3-25, providing location-based applications require a number of components:

- Location-finding equipment, provided by the telecommunications carrier. This is often provided by mechanisms such as identifying the user’s location by identifying the cell tower that they are closest to.

- A location server, which receives information from the location finding equipment or associated infrastructure. The location server is provided either by the carrier themselves, or by another party who has arrangements with the carriers to receive information from their infrastructure. Note that, currently, SignalSoft is the only location server that is supported.
- IBM WebSphere Everyplace Server infrastructure, which contains the Location Based Services and Tivoli Personalized Services Manager components. The WebSphere Everyplace Server Location Based Services component provides the APIs that allow the location server's information to be accessed by applications, and to request location-related content from the same or another location server. It also provides infrastructure services that identify whether an application is location based, and to ensure that user privacy options are respected (through Policy Director). The Tivoli Personalized Services Manager enrollment and Self Care applications allow users to sign up for access to location-based applications, and to specify their privacy options by application.
- Location-based applications, which use the services provided by the WebSphere Everyplace Server Location Based Services and Tivoli Personalized Services Manager components to simplify their coding effort.

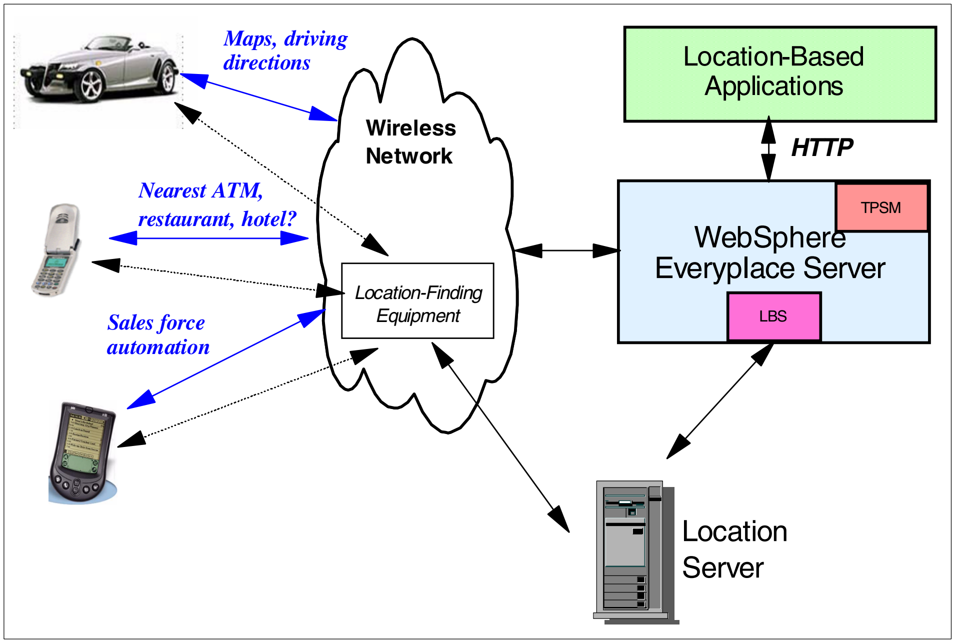

Maps, driving directions
Wireless Network
Location-Based Applications
HTTP
Nearest ATM, restaurant, hotel?
TPSM
WebSphere Everyplace Server
Location-Finding Equipment
Sales force automation
LBS
Location Server

Figure 3-25 Location Based Services context

For more information about how the WebSphere Everyplace Server components interact during a Location Based Services application, please refer to 5.2.5, “Location Based Services” on page 251.

Each Location Based Services server can only communicate with one location server at a time, although multiple Location Based Services servers could be installed to communicate with multiple location servers.

If Policy Director is part of the WebSphere Everyplace Server domain, Location Based Services uses Policy Director’s LDAP directory for privacy information. It continues to use WebSphere Everyplace Server’ SecureWay Directory for configuration information.

> **Important:** A WebSphere Everyplace Server Location Based Services server runs as an IBM WebSphere Edge Server Caching Proxy plug-in. It is recommended that this be installed on a separate server from the WebSEAL-Lite’s instance of Edge Server.

3.3.10 Intelligent Notification Services

The major components in the Intelligent Notification Services infrastructure:

- iQueue Manager
- Universal Notification Dispatcher

Only one instance of each may exist within the WebSphere Everyplace Server domain. This restriction is enforced by the Setup Manager, and within the SecureWay Directory. All three can reside on the same machine or on separate machines.

Figure 3-26 shows the relationship between the Intelligent Notification Services components and other WebSphere Everyplace Server components.

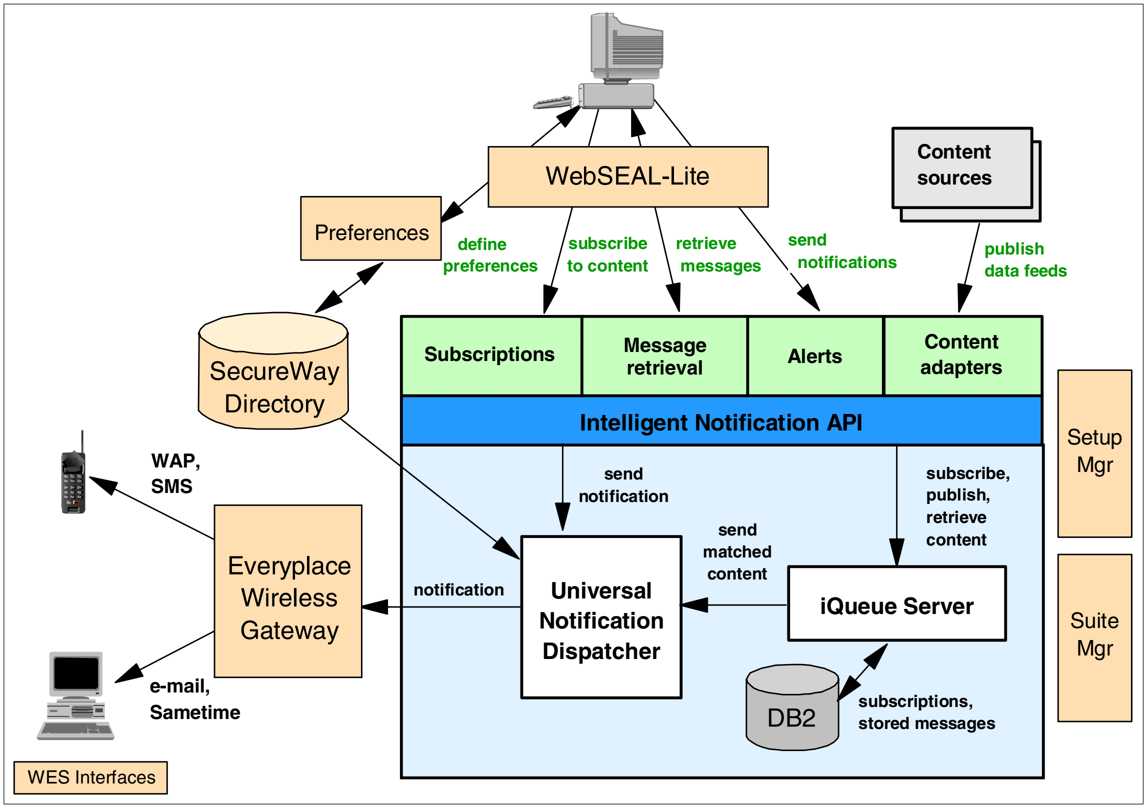


Figure 3-26 Intelligent Notification Services component relationships

The Intelligent Notification Services applications are installed on a WebSphere Application Server. This can be the same WebSphere Application Server that is used for other applications. It is not a required part of the Intelligent Notification Services runtime.

3.3.11 Active Session Table Server

The Active Session Table contains information about each WebSphere Everyplace Server session that is created and/or authenticated. IBM Everyplace Wireless Gateway and WebSEAL-Lite (WSL) interact with it, but it plays only a passive role.

Wireless Gateway’s only interaction with the Active Session Table Server is to maintain session information for Everyplace Wireless clients and for Remote Access clients. Everyplace Wireless Gateway inserts an entry into the Active Session Table Server when a non-WAP client connects, and deletes the entry when the client disconnects or the session times out. Wireless Gateway does not use the data within the Active Session Table Server.

WebSEAL-Lite interacts with the Active Session Table Server to see if Wireless Gateway has provided session information about clients it has not authenticated. When it receives a WAP or IP client request, WebSEAL-Lite checks its internal cache to see if has the results of that request currently cached. If it is not in WebSEAL-Lite’s cache, WebSEAL-Lite checks the Active Session Table Server to see if Everyplace Wireless Gateway has provided authentication information for that session. If not, WebSEAL-Lite authenticates (challenges) the user and inserts a new entry for that session into the Active Session Table Server.

Important: WebSEAL-Lite and Wireless Gateway, are the only components that interact with the Active Session Table during runtime.

Figure 3-27 on page 123 illustrates that Setup Manager installs the Active Session Table Server and Suite Manager administers it.

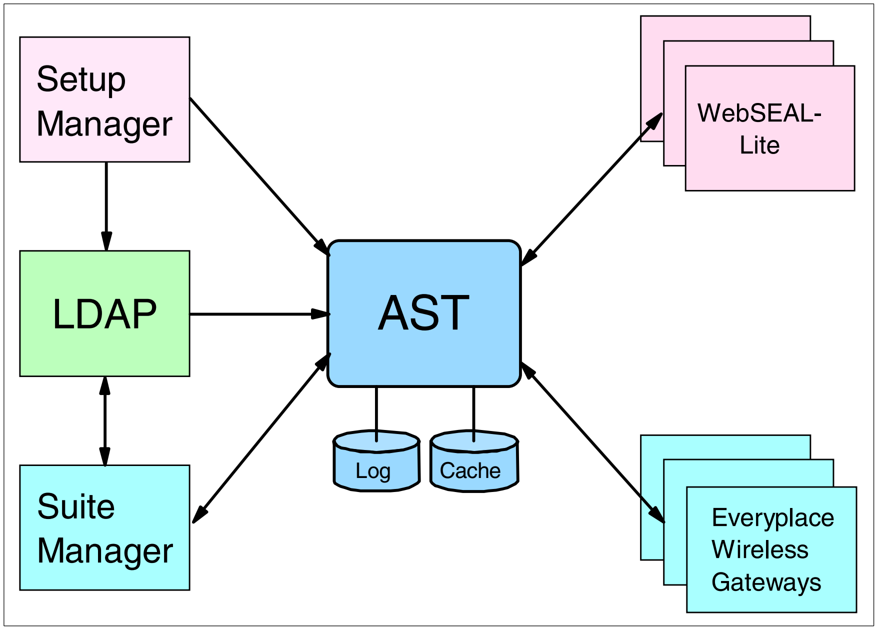


Figure 3-27 Active Session Table Server-Everyplace Server interfaces

3.3.12 Relational databases

A relational database product is required for WebSphere Everyplace Server installation. A number of Everyplace Server components contain instances of relational databases:

- Tivoli Personalized Services Manager uses a relational database to store subscriber information. The WebSphere Application Server that the Tivoli Personalized Services Manager applications are running on may also use a relational database to store WebSphere Application Server management information.
- IBM Everyplace Wireless Gateway uses a relational database to store session information. If configured, it may also use a relational database to store billing information.
- IBM SecureWay Directory uses a relational database to store directory information. If Policy Director is using a separate IBM SecureWay Directory from WebSphere Everyplace Server, then there will be two directory relational databases to be managed.
- Intelligent Notification Services uses DB2 for stored messages and subscriptions.
- IBM Everyplace Synchronization Manager uses DB2.

3.4 LDAP Directories in WebSphere Everyplace Server

Figure 3-28 illustrates, at a high level, how the various WebSphere Everyplace Server components interact with the user registries (LDAP directories) in the WebSphere Everyplace Server domain.

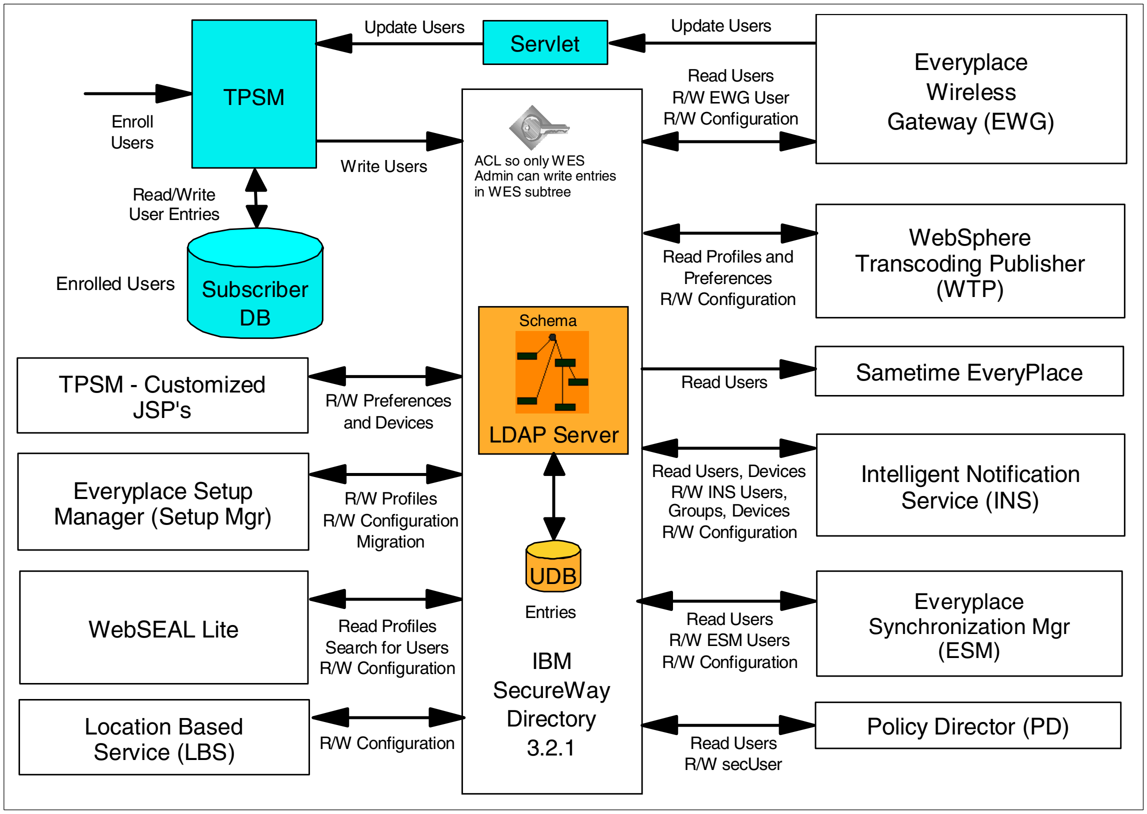


Figure 3-28 Interactions between Everyplace Server components and LDAP Directory

If Policy Director is implemented in the WebSphere Everyplace Server domain, there may, in fact, be two LDAP Directories in the WebSphere Everyplace Server suite:

1. The WebSphere Everyplace Server LDAP Directory
2. The LDAP directory owned by Policy Director

These directories are used by different components.

- WebSphere Everyplace Server LDAP Directory is used by all components for configuration information. It is also used by WebSphere Transcoding Publisher and by WebSEAL-Lite for device and network configuration information; by Intelligent Notification Services for users, group and device information; and by WebSphere Transcoding Publisher for profiles and preferences.

- Policy Director's LDAP directory is used only by:
  - WebSEAL-Lite, for authorization. In addition, WebSEAL-Lite may be configured to use the Policy Director directory for authentication, or it may continue to use the WebSphere Everyplace Server LDAP directory.
  - Location Based Services for privacy, and to identify whether an application is location based or not.

3.4.1 WebSphere Everyplace Server directory structure

The WebSphere Everyplace Server LDAP directory structure is made up of a number of separate collections of information, stored under a common base or root. WebSphere Everyplace Server stores different subsets of configuration and user information in different sections of the overall directory tree. This directory structure is created within the IBM SecureWay Directory instance during install. The contents of the directory are continually modified as components, users and customizations are added, deleted or changed.

Figure 3-29 shows the highest level of the tree, and the three major sections within the tree.

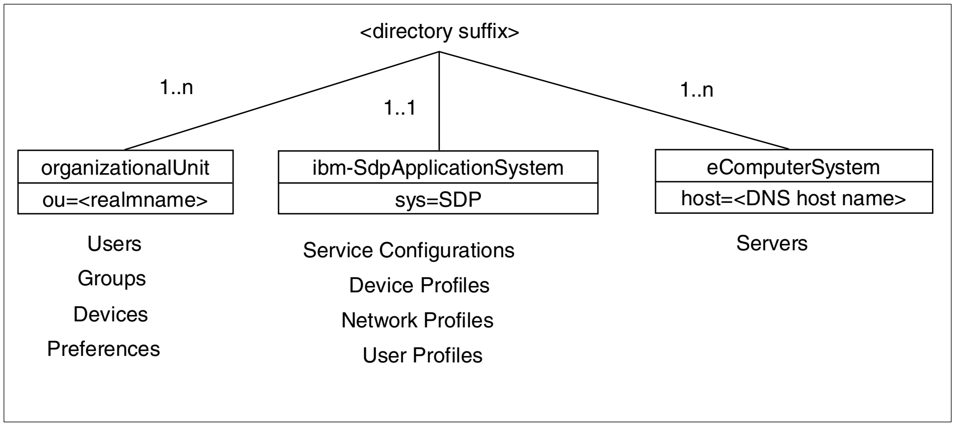


Figure 3-29 Highest level of Everyplace Server directory information tree

The *ibm-SdpApplicationSystem* and *eComputerSystem* sections of the directory together define the WebSphere Everyplace Server domain itself. The realms and their users within the WebSphere Everyplace Server domain are stored under *organizationalUnit*.

Each server within the WebSphere Everyplace Server domain has an entry under eComputerSystem. Each server's entry contains, underneath it, one entry for each component that is currently installed on that server. So, for example, a single server with SecureWay Directory, IBM HTTP Server and DB2 installed would be represented as a single host entry, with three entries: serviceName=SWD, serviceName=IHS, and serviceName=DB2.

Each component's entry also has a pointer to that component's information in the ibm-SdpApplicationSystem tree (under sys=SDP, which is the identifier for WebSphere Everyplace Server). Similarly, each component's entry under ibm-SdpApplicationSystem would contain one pointer to each server on which this component is installed. This way, you can either find all components installed on a specific server, or all instances of a particular component within the WebSphere Everyplace Server domain, and on which server they are installed.

There may only be one ibm-SdpApplicationSystem entry in the tree under root, and it must be set to *sys=SDP*.

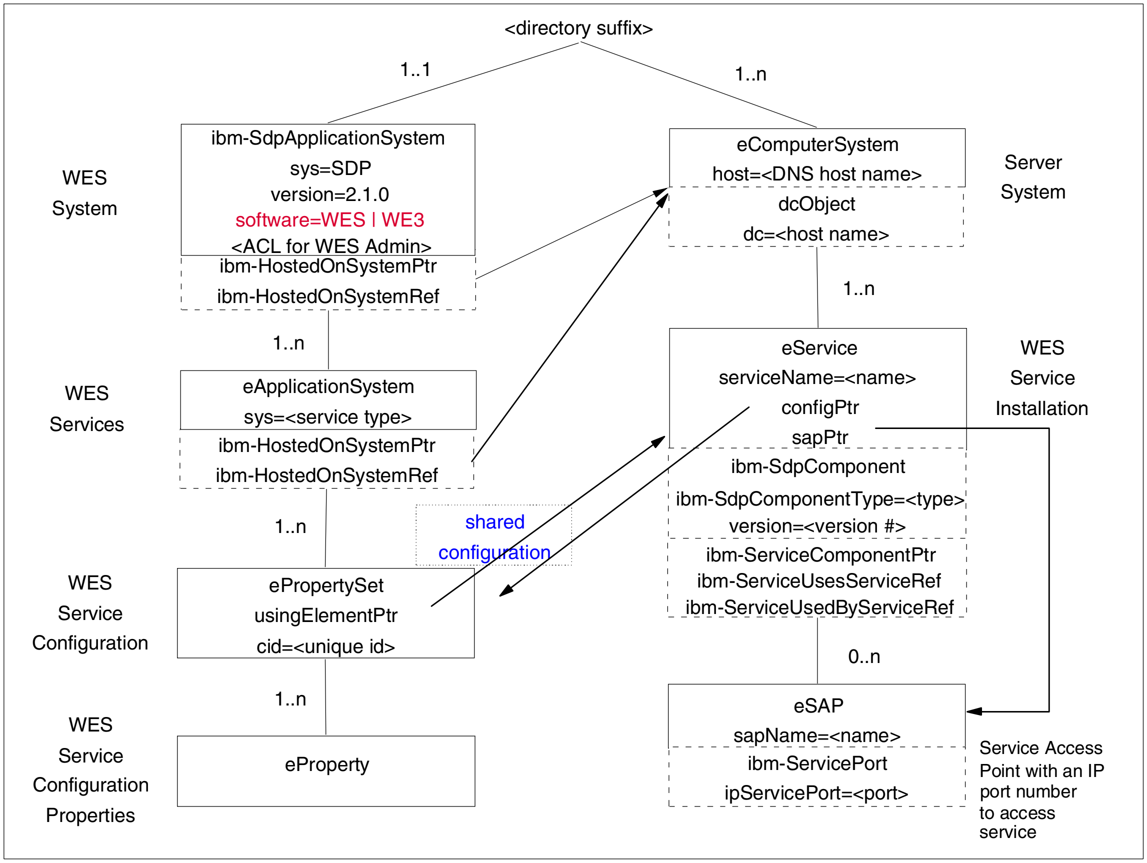


Figure 3-30 Everyplace Server configuration information in the directory

The sys=SDP entry has one *eApplicationSystem* entry (that is, sys=<servicetype>) for each component currently installed in the WebSphere Everyplace Server domain. Table 3-3 shows the valid eServices, that is, the valid serviceNames for each WebSphere Everyplace Server component.

Table 3-3 Component names in the WebSphere Everyplace Server LDAP directory

Component Name (serviceName=<name>)	Component Abbreviation
Edge Server Caching Proxy	wte
WebSEAL-Lite	wep
Location Based Services	lbs
Everyplace Cookie Proxy	ecp

Component Name (serviceName=<name>)	Component Abbreviation
Edge Server Load Balancer	nd nd_bas - Dispatcher nd_cbr - Content Based Routing nd_iss - Interactive Session Support
Active Session Table	ast
Intelligent Notification Services	ins
Suite Manager	wec
Everyplace Synchronization Manager	esm
IBM Everyplace Wireless Gateway	ewg ewg_svr (Gateway) ewg_kpr (Gatekeeper) ewg_ard (Ardis Support) ewg_tac (DataTAC Support) ewg_rad (Dataradio Support) ewg_dal (Dial Support) ewg_lan (IP LAN Support) ewg_mob (Mobitex Support) ewg_sms (Short Messaging Service Support) ewg_smt (SMTP Support) ewg_snp (SNPP support) ewg_src (Modacom-SRC Support) ewg_rnc (Motorola PMR Support)
MQSeries Everyplace	mqe
SecureWay Directory Services	swd
Tivoli Personalized Services Manager	tsm tsmdbs (Database Integration) tsm_dms (Tivoli Device Manager) tsm_ens (Enrollment Server) tsm_ccs (Customer Care Support) tsm_mcs (Member Self Care Support) tsm_sms (System Management) tsm_wes (Everyplace Server Enabler) tsm_ptk (Portal Toolkit)
WebSphere Transcoding Publisher	wtp

Each instance of each component within the domain is represented under its component entry. For example, if four servers each have WebSphere Transcoding Publisher installed, there will be four entries, one with each server name, under the sys=WTP entry. Each entry contains a pointer to the corresponding host=<DNS host name> entry.

In addition, the configuration information for the components (WebSphere Everyplace Server configuration shown in Figure 3-30 on page 128) is stored in elements of object class *ePropertySet*. This allows multiple instances of each component, running on multiple different servers, to share a single configuration. For example, a cluster of WebSphere Transcoding Publishers may share the same configuration parameters.

The information in these sections of the tree is initially created by the Setup Manager, and is maintained through the Suite Manager, or by using the administration consoles of the WebSphere Everyplace Server components.

Figure 3-31 on page 131 illustrates the directory structure used to describe users within WebSphere Everyplace Server. The boxes represent the object classes used in the LDAP directory structure, and the relationships between the object classes. Next to some of the boxes we show how the object class is used within WebSphere Everyplace Server.

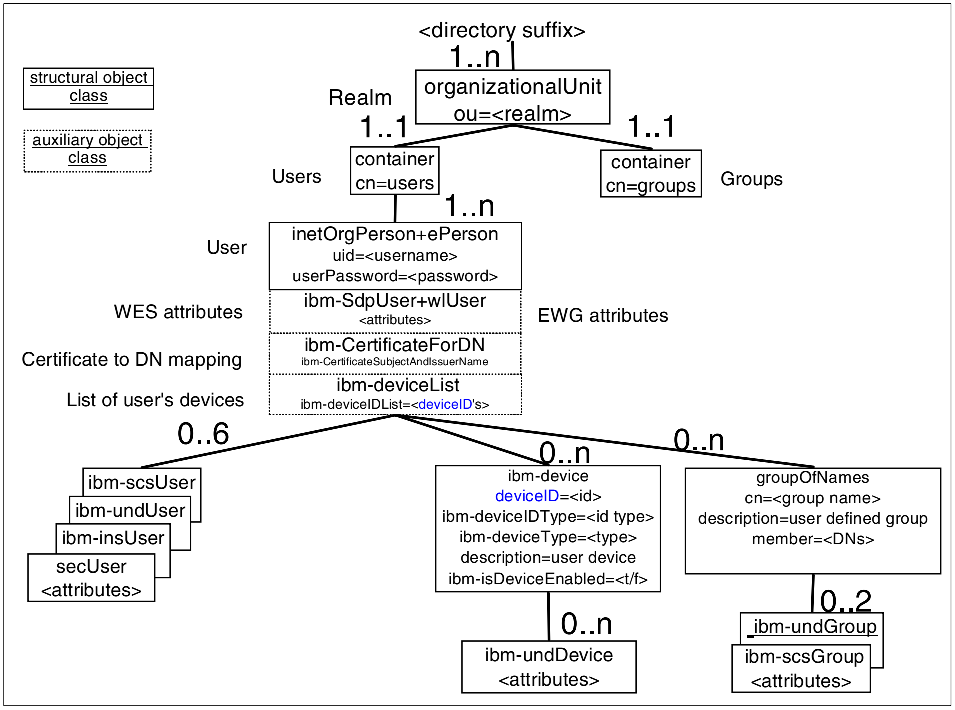


Figure 3-31 Service Provider Offering LDAP realm tree DIT Structure for Users

The information about users and their devices is created or maintained via Tivoli Personalized Services Manager. It can also be maintained by importing LDIF format files containing user information, or through SecureWay Directory tools such as the LDAP client and the Directory Management Tool. The information about a user's devices is required for use by the Device Manager Server.

Tivoli Personalized Services Manager's LDAP Gateway provides an automated way to provision Tivoli Personalized Services Manager subscriber information in LDAP based on user state changes (new user, delete user, user updates profile, etc.) This is handled through a customizable property file, LDAPGateway.properties, that is triggered automatically during Tivoli Personalized Services Manager enrollment/Self Care/Customer Care, and during specific events (such as user password change or realm creation).

WebSphere Everyplace Server provides changes to this script for the default WebSphere Everyplace Server environment. Some additional changes are required to match your environment.

For Wireless Gateway use, you can also add the following attributes to users in the WebSphere Everyplace Server LDAP Directory server:

```
ibmwgclient on|off - does this user have access to wireless services
ibmwapclient on|off - does this user have access to WAP services
```

If the setting is `off`, then Wireless Gateway will reject a connect request from that user. For further documentation, please refer to the *Everyplace Wireless Gateway 2.1 Administration Guide.*

Intelligent Notification Services can use SMS messages, Wireless Application Protocol push messages, e-mails, and instant messaging to notify users when preconfigured events occur.

Users access JSP templates within Tivoli Personalized Services Manager and enter their preferences about the delivery method and type of events sent to them. The user can define their preferences for how they wish to receive notifications, and who is allowed to send them notifications by which methods. The following list shows the kinds of information used by Intelligent Notification Services:

- User profile
  - ID, name, device profiles, group profiles
- Device profile
  - Type, protocol version, device address, gateway info
- Group profile
  - Members of group (team, family, dept.)
  - Type of messages (urgent, normal, fyi) to receive from group members

Attributes related to this component include ibm-deviceList, ibm-scsUser, ibm-undUser, ibm-insUser, and secUser. In addition, ibm-undGroup is used to specify a group of other users that this user wishes to interact with in particular ways.

Only customer service representatives may enter user device identifiers in Tivoli Personalized Services Manager’s User Preferences templates when enrolling or updating customers. This information provides user overrides for device characteristics by altering ePropertySet+eProperty under the person object.

Location Based Services allow the transmittal of messages that are specific to a user's geographical location. Users define a list of URLs and create privacy preferences that permit or restrict each listed Web site from viewing location information. Attributes related to this component include eProperty and ePropertySet. Privacy information is stored in Policy Director.

Voice Server allows a user to set up an alternate alias for telephony access. The alias typically consists of a numeric entry (for example, a 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. The affected object class and attribute are mobileTelephoneNumber/ePerson or mobile/ePerson.

The Language Services section of User Preferences allows the end user to select a language preference for content when a transmission is capable of alternate language delivery. The appropriate object class for this preference is preferredLanguage/ePerson. This can be used by WebSphere Transcoding Publisher V4.0 (not part of WebSphere Everyplace Server V2.1) to identify the language into which the information should be transcoded.

The User Certificate DN attribute allows the user to set the DN from his X.509 certificate (if that user is doing certificate authentication) and allows Everyplace Server to map from that certificate DN back to the original user SecureWay Directory entry. The affected object class is secCertDN/ibm-SdpUser.

WebSphere Portal Server has its own directory structure in the LDAP directory, which is not currently integrated into the WebSphere Everyplace Server directory structure. WebSphere Portal Server accesses the user information via an EJB that could be modified to access the WebSphere Everyplace Server directory structure.

For WebSphere Everyplace Server Enable Offering, user enrollment is external, since there is no Tivoli Personalized Services Manager. A user EJB is provided, with an API that provides four classes:

- User (which maps to inetOrgPerson in the LDAP directory)
- UserManager
- Group (which maps to groupOfUniqueNames in the LDAP directory)
- GroupManager

Figure 3-31 on page 131 shows the remaining portions of the directory information tree (DIT).

- *Device Profiles* are used for the WebSphere Transcoding Publisher transcoding attributes. In WebSphere Everyplace Server Service Provider Offering, WebSphere Transcoding Publisher does not store profiles in its own database.

- *Network Profiles* are used for the Wireless Gateway.

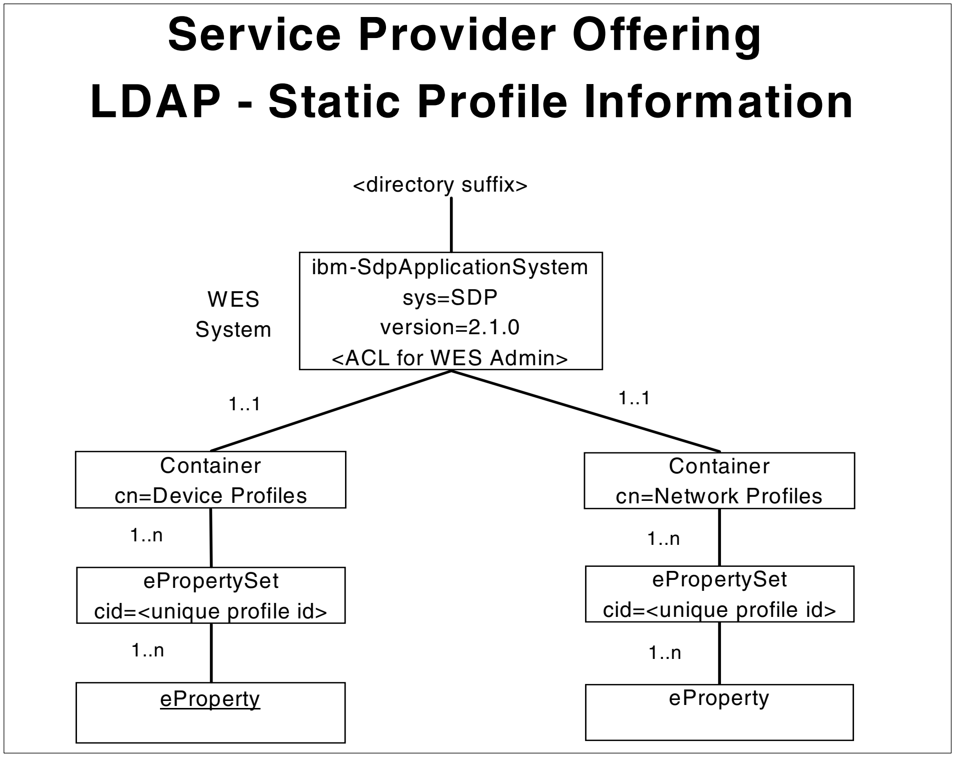
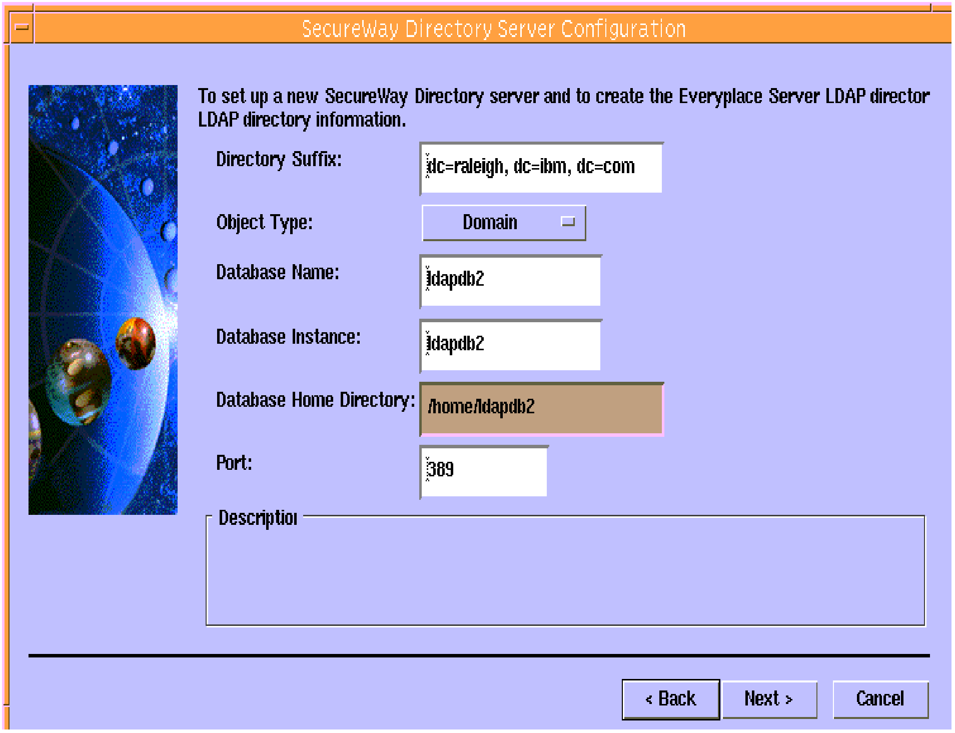


Figure 3-32 Remaining sections of WebSphere Everyplace Server DIT

Merging LDAP Directories

The directory suffix for WebSphere Everyplace Server is defined during installation, as shown in Figure 3-33.



SecureWay Directory Server Configuration

To set up a new SecureWay Directory server and to create the Everyplace Server LDAP director
LDAP directory information.

Directory Suffix: dc=raleigh, dc=ibm, dc=com

Object Type: Domain

Database Name: ldapdb2

Database Instance: ldapdb2

Database Home Directory: /home/ldapdb2

Port: 389

Descriptio

Figure 3-33 Default SecureWay Directory configuration during installation

Note that although the SecureWay Directory suffix format can be set, as shown in Figure 3-33, by changing the “object type” and then specifying a new suffix, the WebSphere Everyplace Server InfoCenter installation documentation specifies that the “dc=” format must be used.

During installation, if you use the default formats and entries for both the WebSphere Everyplace Server and for the Policy Director installs, and direct the install processes to use the same physical SecureWay Directory instance, SecureWay directory’s configuration file (slapd32.conf) will reflect the base entries for two different DITs, as shown in Example 3-6.

Example 3-6 Default slapd32.conf

```
ibm-slapdSuffix: cn=localhost
ibm-slapdSuffix: cn=wesadmin
ibm-slapdSuffix: dc=raleigh, dc=ibm, dc=com     <- entry for WebSphere
Everyplace Server
ibm-slapdSuffix: o=tivoli, c=us <- default entry for Policy Director
ibm-slapdSuffix: secAuthority=Default <- entry for Policy Director
```

In this case, the information for Policy Director and for WebSphere Everyplace Server will be stored in two different Directory Information Trees. Even if the same suffix is given for both installs, Policy Director creates a separate user, since by default it uses the "cn=" format to uniquely identify a distinguished name, whereas WebSphere Everyplace Server uses the "uid=" attribute. In fact, WebSphere Everyplace Server uses "cn=users" as a container, as shown in Figure 3-31 on page 131.

An efix, 3.7.1-POL-0004E, can be applied to Policy Director. This efix now allows the "pdadmin" user import command to use the uid attribute for naming the distinguished name. Since WebSphere Everyplace Server uses the uid attribute for naming, this change must also be made before the DITs can be merged.

3.4.2 Policy Director's Directory

Policy Director's LDAP directory is used by WebSEAL-Lite for centralized user authorization, and by Location Based Services privacy functions. In both cases, the directory is accessed via Policy Director, using the aznAPI interface.

When Policy Director is in the WebSphere Everyplace Server domain, WebSEAL-Lite may also be configured to use Policy Director's aznAPI for authentication, or it can continue to use the SecureWay Directory for authentication. If you do not need these centralized user authorization or Location Based Services, then you do not need to implement Policy Director in your WebSphere Everyplace Server domain.

Users are created and maintained in Policy Director's LDAP directory via the Tivoli Personalized Services Manager Policy Director Bridge. The Tivoli Personalized Services Manager-Policy Director Bridge provides an automated way to provision information in Policy Director based on user enrollment changes (new user, delete user, user changes subscription information resulting in group membership changes). Associations are maintained in Tivoli Personalized Services Manager from realms to one or more Policy Director groups and Deals to one or more Policy Director groups.

To provision the user into Policy Director, Tivoli Personalized Services Manager's customizable shell script, pd_admin.ksh, must be updated for your environment (refer to the README). This script is automatically invoked during Tivoli Personalized Services Manager enrollment, Self Care, and Customer Care, using the parameters shown below:

```
pd_admin.ksh user_create <username> <realm> <firstname> <lastname>
<password> <groups>
pd_admin.ksh user_modify_group <username> <realm> <groups>
pd_admin.ksh user_delete <username> <realm>
pd_admin.ksh user_show <username> <realm>
pd_admin.ksh user_list <username> <realm>
```

For example:

```
pd_admin.ksh user_modify_group JSMITH IBM "basic enhanced"
```

Thus, you must configure your Policy Director environment in coordination with your Tivoli Personalized Services Manager environment, in such a way that these calls can be used to correctly provision your user.

We now show an example of the configuration requirements. A Tivoli Personalized Services Manager sample configuration is shown in Figure 3-34. Refer to 3.3.8, “Tivoli Personalized Services Manager” on page 115 for more information about Tivoli Personalized Services Manager’s data models.

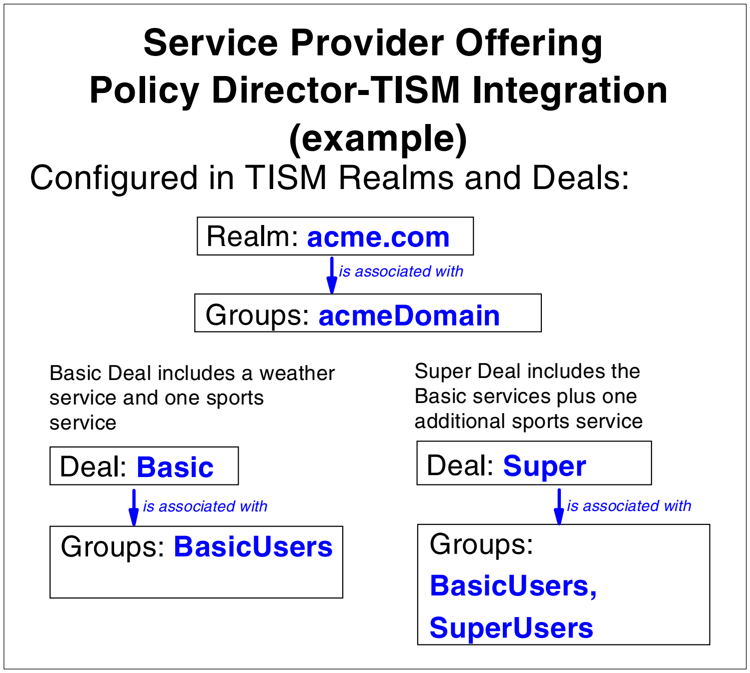

Service Provider Offering
Policy Director-TISM Integration
(example)
Configured in TISM Realms and Deals:
Realm: acme.com
is associated with
Groups: acmeDomain
Basic Deal includes a weather service and one sports service
Super Deal includes the Basic services plus one additional sports service
Deal: Basic
is associated with
Groups: BasicUsers
Deal: Super
is associated with
Groups: BasicUsers, SuperUsers

Figure 3-34 Integration Example: Tivoli Internet Services Manager Configuration

In Policy Director, the following items were manually configured:

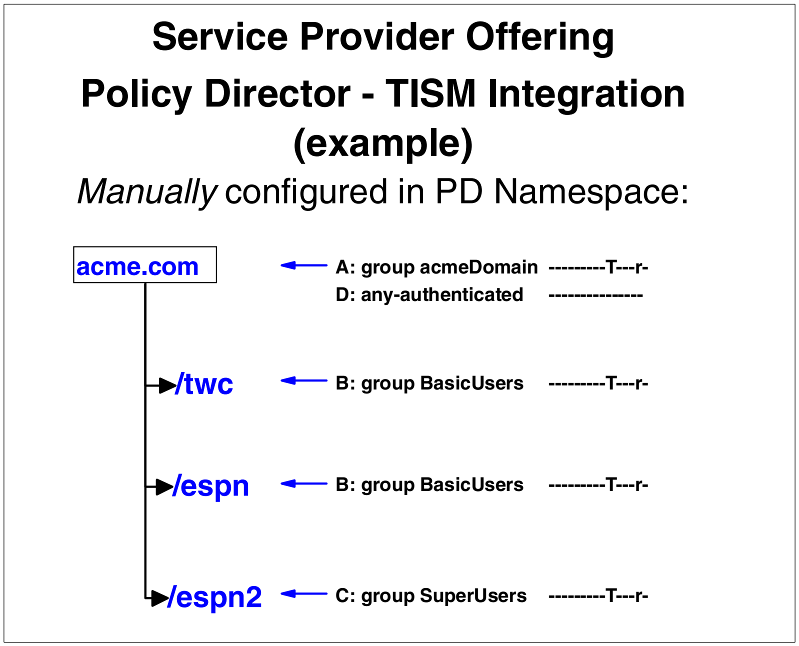
- Groups:

```
BasicUsers
SuperUsers
acmeDomain
```

- ACLs:

```
A: group acmeDomain --------T---r-
B: group BasicUsers ---------T---r-
C: group SuperUsers ---------T---r-
D: any-authenticated ---------------
```

In addition, the relationships between the ACLs and the Web site are established in Policy Director, as shown in Figure 3-35.



Service Provider Offering

Policy Director - TISM Integration (example)

Manually configured in PD Namespace:

acme.com
A: group acmeDomain ---------T---r-
D: any-authenticated --------------
/twc
B: group BasicUsers ---------T---r-
/espn
B: group BasicUsers ---------T---r-
/espn2
C: group SuperUsers ---------T---r-

Figure 3-35 Integration Example: Policy Director Configuration

Then, if a new user uses the Tivoli Personalized Services Manager Self-Enrollment to enroll as “joesixpack”, with a deal of “Basic”, the bridge automatically configures the following in PD:

```
Group BasicUsers <<< joesixpack@acme.com
Group acmeDomain <<< joesixpack@acme.com
```

Policy Director maintains a logical hierarchy of resources (for example, Web applications) against which users may or may not have permission to perform an action or set of actions.

Location privacy is controlled by Policy Director. An application must be registered in Policy Director as a location-based application to receive a user's location information. In addition, the user's privacy settings must authorize an application to obtain location information. Location Based Services uses the Policy Director aznAPI interface to access user privacy information.

At Location Based Services install time, the pd_populate script is run to create an entry for each application URL within Policy Director's directory. For each resource created, Location Based Services creates an access control list (ACL) and a group. The group is bound to the ACL, and the ACL is attached to the resource.

A user, WebSphere Everyplace Server_LBS, also created at install time, is added to each group. This user is used internally to identify whether an application is location-based, or not (if user WebSphere Everyplace Server_LBS can access the URL, then the application is location based).

To enable application URLs for Location Based Services after the installation, you must rerun the script, with parameters defining the application URL to be added:

```
pd_populate.ksh initAppl <pdAdminPassword> <ldapSuffixDN> <Application URL>
```

When a user goes into Tivoli Personalized Services Manager to authorize location-based services, he is presented with a list of Location Based Services. This list is created by identifying all applications that contain the user WebSphere Everyplace Server_LBS. The user can then choose to:

- Deny permission to any applications
- Grant permission to some or all applications to receive his location information

When the user authorizes an application to collect location-based information on their behalf, Tivoli Personalized Services Manager, via the Tivoli Personalized Services Manager-Policy Director bridge, adds the user to the appropriate group. If a user belongs to a group that is associated with a resource, then the user is permitting that resource to use his or her location information.

In addition, Location Based Services maintains two groups, LBS_PERMIT_ALL, and LBS_DENY_ALL, which are used to maintain overriding privacy settings if specified by the user.

For example, imagine we have the following users:

- WebSphere Everyplace Server_LBS (WebSphere Everyplace Server Location Based Services user created at install time)

- User X (has DENIED permission for ANY apps to receive his location information)
- User Y (has GRANTED permission for ALL apps to receive his location information)
- User W (wants http://www.weather.com/forecast to receive his location information)
- User V (wants http://www.services.com/hotels AND http://www.weather.com/forecast to receive his location information, BUT NO OTHER LOCATION-ENABLED APPS)

And, we have the following groups:

- LBS_DENY_ALL
  - User X
- LBS_PERMIT_ALL
  - User Y
- LBS_www.services.com/hotels
  - WebSphere Everyplace Server_LBS
  - User V
- LBS_www.weather.com/forecast
  - WebSphere Everyplace Server_LBS
  - User W
  - User V

How these application URLs would be represented within the overall Policy Director object space is illustrated in Figure 3-36.

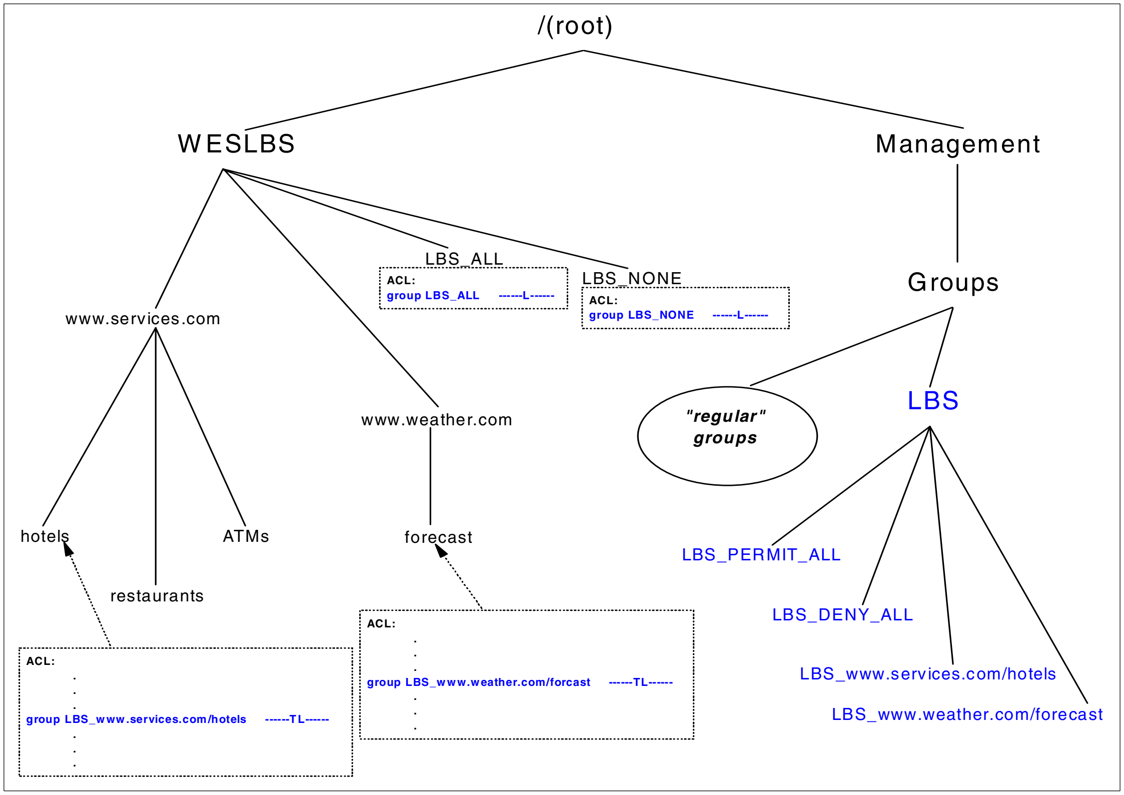
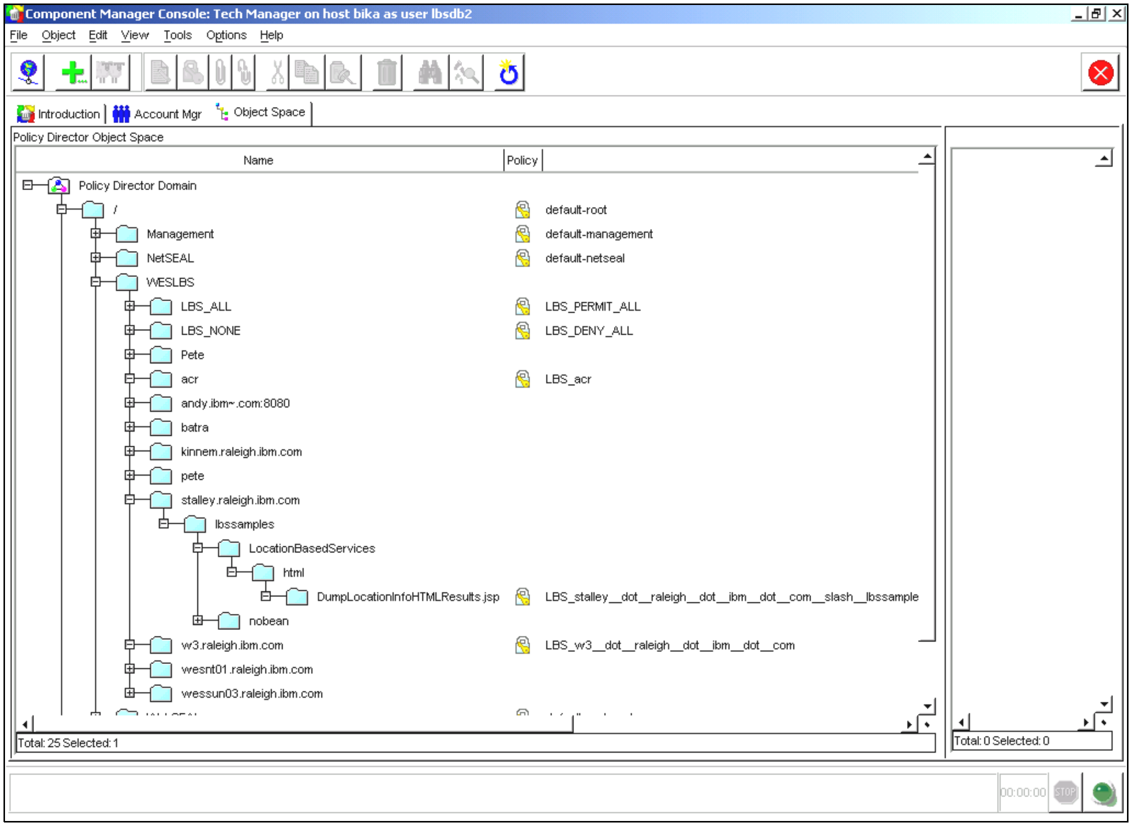

/(root)
WESLBS
Management
LBS_ALL
ACL:
group LBS_ALL ------L------
LBS_NONE
ACL:
group LBS_NONE ------L------
Groups
www.services.com
www.weather.com
"regular" groups
LBS
hotels
restaurants
ATMs
forecast
LBS_PERMIT_ALL
LBS_DENY_ALL
LBS_www.services.com/hotels
LBS_www.weather.com/forecast
ACL:
group LBS_www.services.com/hotels ------TL------
ACL:
group LBS_www.weather.com/forcast ------TL------

Figure 3-36 Policy Director's object space

In this representation, the Web site, www.services.com/hotels has had the ACL LBS_www.services.com/hotels assigned to it. Since User V is in this group (was added when he updated his Tivoli Personalized Services Manager preferences), User V can receive location-based information collected on his behalf by this application.

Figure 3-37 shows the console for Policy Director's Object Space tree, for the object space below WESLBS (see the left-hand side of the object space tree in Figure 3-36). You can see that the "application" LBS_ALL has the policy "LBS_PERMIT_ALL" assigned to it. LBS_NONE has the policy, LBS_DENY_ALL assigned to it.


Component Manager Console: Tech Manager on host bika as user lbsdb2
File
Object
Edit
View
Tools
Options
Help
Introduction
Account Mgr
Object Space
Policy Director Object Space

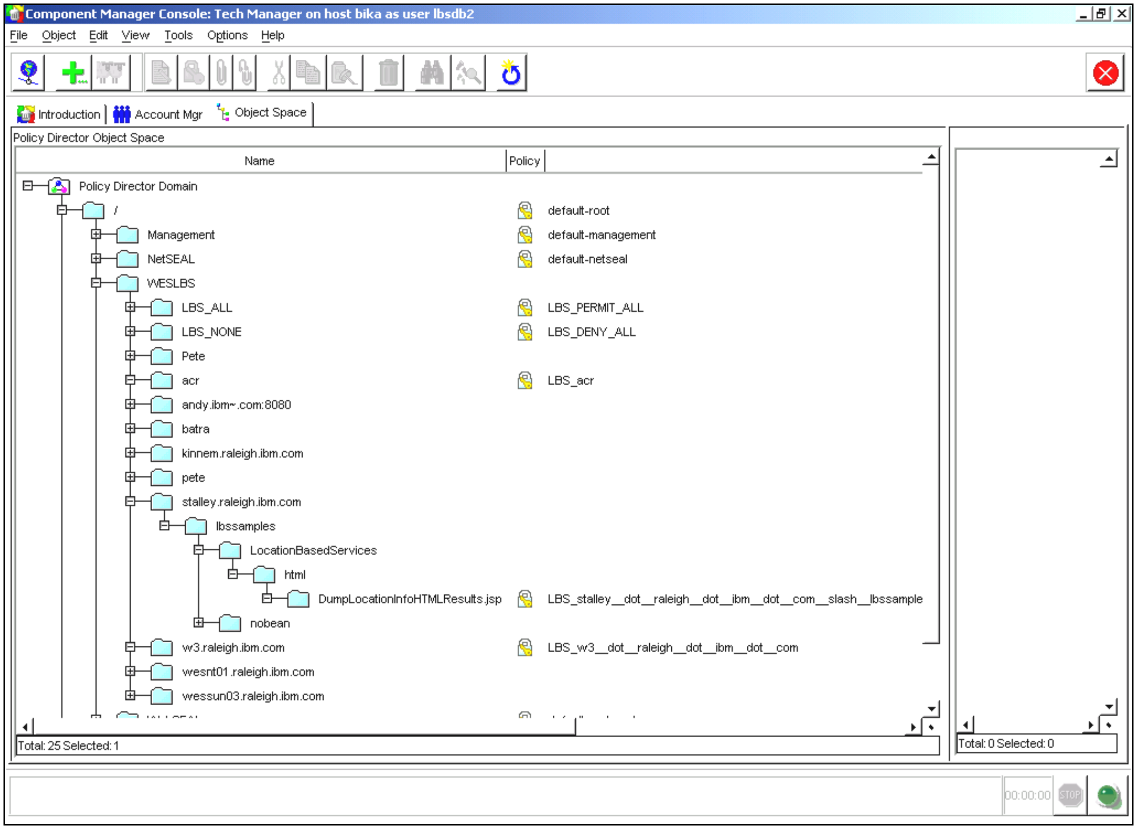
Name	Policy
Policy Director Domain	
/	default-root
Management	default-management
NetSEAL	default-netseal
WESLBS	
LBS_ALL	LBS_PERMIT_ALL
LBS_NONE	LBS_DENY_ALL
Pete	
acr	LBS_acr
andy.ibm~.com:8080	
batra	
kinnem.raleigh.ibm.com	
pete	
stalley.raleigh.ibm.com	
lbssamples	
LocationBasedServices	
html	
DumpLocationInfoHTMLResults.jsp	LBS_stalley__dot__raleigh__dot__ibm__dot__com__slash__lbssample
nobean	
w3.raleigh.ibm.com	LBS_w3__dot__raleigh__dot__ibm__dot__com
wesnt01.raleigh.ibm.com	
wessun03.raleigh.ibm.com	

Total: 25 Selected: 1

Total: 0 Selected: 0

Figure 3-37 Console view of Location Based Services in PD's Object Space tree

Figure 3-38 shows the console view of Location Based Services in Policy Director's Account Manager tree. As you can see, each of the Location Based Services applications has the user WebSphere Everyplace Server_LBS within it. You can also see the groups, LBS_DENY_ALL and LBS_PERMIT_ALL shown in the view.


Component Manager Console: Tech Manager on host bika as user lbsdb2
File Object Edit View Tools Options Help
Introduction
Account Mgr
Object Space
Policy Director Object Space
Total: 25 Selected: 1
Total: 0 Selected: 0
00:00:00
STOP

Name	Policy
Policy Director Domain	
/	default-root
Management	default-management
NetSEAL	default-netseal
WESLBS	
LBS_ALL	LBS_PERMIT_ALL
LBS_NONE	LBS_DENY_ALL
Pete	
acr	LBS_acr
andy.ibm~.com:8080	
batra	
kinnem.raleigh.ibm.com	
pete	
stalley.raleigh.ibm.com	
lbssamples	
LocationBasedServices	
html	
DumpLocationInfoHTMLResults.jsp	LBS_stalley__dot__raleigh__dot__ibm__dot__com__slash__lbssample
nobean	
w3.raleigh.ibm.com	LBS_w3__dot__raleigh__dot__ibm__dot__com
wesnt01.raleigh.ibm.com	
wessun03.raleigh.ibm.com	

Figure 3-38 Console view of Location Based Services in PD's Account Mgr tree

3.5 Setup Manager

Everyplace Server contains many individual components. Installing so many components individually is non-trivial, let alone installing them into a distributed setting. The Everyplace Server installation is designed to be a centralized installation of components.

Many of the components within WebSphere Everyplace Server are already mature products with their own installation process. The centralized WebSphere Everyplace Server installation is a coordinated effort using the software installation and upgrade mechanism already in place for each components. The Setup Manager determines what common configuration parameters can be "factored out" and supplied only once, and 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 the LDAP directory for subsequent usage later in the installation process. This way, the parameters entered during installation can be saved for other Everyplace Server component installations.

Setup Manager can perform the following tasks:

- Set up the Everyplace Server environment
- Allow limited migration from previous Everyplace Server versions, V1.1.2 and V1.1.3, to V2.1.
- Help you save time on subsequent installations of Everyplace Server by allowing you to pre-define component information that is used to automatically populate fields during the installation.
- Provide a mechanism to validate that the installation and configuration completed successfully.

Important: The installation performed by the Setup Manager does not take into consideration production load requirements. You should consult experts in the individual components when designing and implementing your production environment.

Be aware that many products, when installed within the WebSphere Everyplace Server environment, store some or all of their configuration parameters in the LDAP directory. When you are configuring the components for production performance, you must ensure that you have changed settings in the correct place, or you may not get the results you expect.

Policy Director installation is an independent process. If Policy Director is required in your environment, you must perform this installation in the correct order within the overall WebSphere Everyplace Server configuration. If you wish to merge the Policy Director LDAP directory into the WebSphere Everyplace Server LDAP directory, you must do this during the install process.

3.6 Suite Manager

Everyplace Suite Manager provides a centralized method for launching the administration consoles of the installed IBM 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 the SecureWay Directory LDAP Directory. Everyplace Suite Manager allows users to perform the following management tasks from one centralized location:

- Monitor status of Everyplace Server components
- Start and stop Everyplace Server components
- Change initial Everyplace Server configuration settings after Everyplace Server installation is complete
- Launch the administration consoles of individual Everyplace Server components
- View Everyplace Server component logs. Only logs stored in default locations in text format can be viewed. Viewing is not real-time; that is, the Suite Manager makes a static copy of the log file at request time, and you view that copy.

The Setup Manager installs a *remote server* on each system when it installs the WebSphere Everyplace Server components. The Suite Manager communicates with the remote servers over a Java RMI link, using Secure Socket Layer (SSL) technology. Each component, and some subcomponents, have a separate *ServerObject* within the remote server. The remote server uses RMI activation support, so the objects are only loaded when they are being used. The ServerObjects are used to determine the status of a component, start and stop a component, and to retrieve a log file for a given component.

Changing Server Configurations with Suite Manager

Only components with locations and properties stored in the LDAP directory can have their properties updated through the Suite Manager. For example:

- Active Session Table Server
- Location Based Services
- WebSphere Transcoding Publisher

The following products use the configuration from their own configuration files; however, properties for these products are stored in the LDAP directory to allow other components to identify how to connect to them. For these products, if you change properties in their configuration files that are also stored in LDAP, you will need to update the LDAP entry as well:

- IBM HTTP Server
- Edge Server Caching Proxy, and its plug-ins:
  - WebSEAL-Lite
  - Location Based Services
  - Everyplace Cookie Proxy
- SecureWay Directory
- Setup Manager (used only during component install)
- Suite Manager

For example, if you wish to send an HTTP request to the HTTP Server you need to know which port the server is accepting connections on. This is defined in the HTTP Server's httpd.conf file on the local file system. But you need to know this information remotely, so the WebSphere Everyplace Server Installer adds this information into the LDAP directory, in order that others may use this information. If the access port is changed in the HTTP Server's httpd.conf file, then a change is also necessary to the LDAP information. Suite Manager allows the administrator to do this. In addition Suite Manager is a user of this LDAP information, so it is important to keep it up to date.

Managing consoles with Suite Manager

The following components' consoles are started by the Suite Manager, no matter where in the WebSphere Everyplace Server domain the product is installed:

- IBM HTTP Server
- Edge Server Caching Proxy, and its plug-ins:
  - WebSEAL-Lite
  - Location Based Services
  - Everyplace Cookie Proxy
- SecureWay Directory
- Active Session Table Server

The following component consoles can be started by Suite Manager only if the components were installed by Setup Manager on the same server that the Suite Manager is running on:

- Everyplace Wireless Gateway (the Wireless Gatekeeper)
- Edge Server Load Balancer
- WebSphere Application Server administration console
- DB2
- WebSphere Transcoding Publisher
- SecureWay Directory - DMT

The following components' consoles are not started by the Suite Manager, even if they are installed on the same server that Suite Manager is running on:

- Policy Director
- Tivoli Personalized Services Manager (Windows only console)

In general, in a production environment, it is unlikely that these products will be running on the same server as the Suite Manager.

The following components must be modified before they can be monitored or controlled:

- MQ Everyplace
- Everyplace Synchronization Manager

The following component has no administration capability:

- Intelligent Notification Services

Currently, the Suite Manager does not check whether another component should be stopped prior to stopping the component requested.

Starting the Suite Manager

After installation, you can start the console by clicking the console icon, which was created during the installation process. On AIX, the icon can be found in /home/admin_userID/wesconsole, /wesconsole, and /usr/IBMEPS/suite/wesconsole, and on Solaris in /home/admin_userID/wesconsole, /wesconsole, and /opt/IBMEPS/suite/wesconsole, where admin_userID is the user ID specified in the Everyplace Administration Console installation.

After installation, you can also start the console from the command line as follows:

1. Log on with the Administration Console user ID specified during the installation (the Everyplace Administration Console can also be started using the root user ID).
2. Open a terminal window.
3. Execute the following commands:

   For AIX:

   ```
   /usr/IBMEPS/suite/wesconsole.sh
   ```

   For Solaris:

   ```
   /opt/IBMEPS/suite/wesconsole.sh
   ```

Part 2

Business scenarios

In this part we introduce two business scenarios with which we illustrate how you may use IBM WebSphere Everyplace Server Enable Offering and Service Provider Offering to solve a variety of wireless e-business problems. The first scenario shows how you can use Everyplace Server Enable Offering to solve the business-to-employee (B2E) problems of a fictitious insurance company, QuickClaim. The second scenario shows how you can use IBM WebSphere Everyplace Server Service Provider Offering to solve the business-to-consumer (B2C) problems of a fictitious hotel chain, Star Hotels.

4

Business-to-employee

This chapter describes a business-to-employee scenario using WebSphere Everyplace Server. The following scenario is an example, but not the only one possible with WebSphere Everyplace Server Solution. Businesses are looking to increase employee's productivity by making the right information accessible to their employees at the right time from anywhere. The enterprises desire to empower their employees to act on that information at the right time from anywhere. This chapter walks through the process of implementing a business-to-employee solution using WebSphere Everyplace Server. It includes discussions on business benefits, architecture decision, solution outline, component selection and integration considerations. Scalability, availability and operational considerations for a WebSphere Everyplace Server solution are discussed in the other chapters of this book.

WebSphere Everyplace Server Enable Offering is targeted at the business-to-employee, also known as B2E, enterprise market segment. An enterprise most likely has already established a user registry and authentication system for their employees. WebSphere Everyplace Server Enable Offering collaborates with the existing user registry and authentication systems.

The following is an example of a sales force automation system in the insurance industry. The scenario can also be implemented using WebSphere Everyplace Server Service Provider Offering, if the enterprise can adopt WebSphere Everyplace Server Service Provider Offering's security architecture and subscriber management framework; that is using WebSEAL-Lite for authentication, Tivoli Policy Director as the security management system, and TPSM as the subscriber management system. The last section of this chapter describes a WebSphere Everyplace Server Service Provider Offering implementation of the same scenario. A discussion of the major architectural differences between the two implementation is included.

4.1 Business scenario overview

Joe Smith is an insurance agent with QuickClaim insurance company that has a service center located at Sacramento, California. Joe calls on new clients to sell QuickClaim's auto, home owner, health and life insurance policies. The service center representatives support customers in remote locations where QuickClaim does not have sales agents in the region. QuickClaim has developed the following five sales support applications:

- Risk Assessment application

  Risk Assessment application provides real-time assessment of a potential client's risk level by evaluating all the available auto and health insurance claim records, driving records and history records based on an individual's social security number, driver's license number, and vehicle identification number.

- Quote application

  The Quote application calculates the insurance premium based on an individual's profile and risk level.

- Payment application

  The Payment application connects to financial institutions for credit card payment processing.

- Policy application

  The Policy application provides online application forms and processing for QuickClaim's insurance policies

- Customer Information application

  The Customer Information application provides online information on all the QuickClaim's customers.

QuickClaim's agents and service center representatives access these applications using a Web browser from their offices. If Joe can have access to these sales support applications using wireless handheld devices from a customer's location, Joe will be able to:

- Assess whether this customer is "desirable" based on his or her risk level and claim history

- Tailor the policies to meet client's individual needs and provide a higher level of customer services
- Deliver a binding quote and close the insurance deal on the spot without another follow-up visit or call
- Capture policy application profile and information on the handheld devices and replicate the policy application data to QuickClaim's HQ database, improving productivity and accuracy of the information
- Accept credit card or check payment from the customer on the spot to finalize the deal
- Access and synchronize e-mail to monitor and act on information that requires immediate attention while on the road
- Access and synchronize calendar and PIM applications to a update personal calendar and to schedule meetings or events while on the road
- Replicate customer contact information on the handheld device for offline access

The likelihood of an agent to close a deal in a personal visit is much higher than with a follow-up call. Enabling anywhere access to these sales support applications will drive more sales of QuickClaim insurance policies, and reduce the risk of insurance underwriting.

4.1.1 Business justifications for B2E mobile solutions

QuickClaim conducted a business case study then decided to pilot the solution. During this study, QuickClaim found the following key business justifications for companies to roll out B2E mobile solutions:

- Increase communications and productivity among executives, management, development, marketing, staff and sales.
- Increase customer satisfaction by providing timely and accurate information to customers, leading to higher customer retention, with higher customer retention leading to higher profits
- Reduce operational and underwriting risks by leveraging timely, accurate information and business intelligence capabilities, which will allow continuous operations monitoring, sound business assessment and decisions from anywhere at anytime
- A measurable increased ROI (Return on Investment)

4.1.2 Architectural vision

Figure 4-1 depicts QuickClaim's architecture vision for this solution.

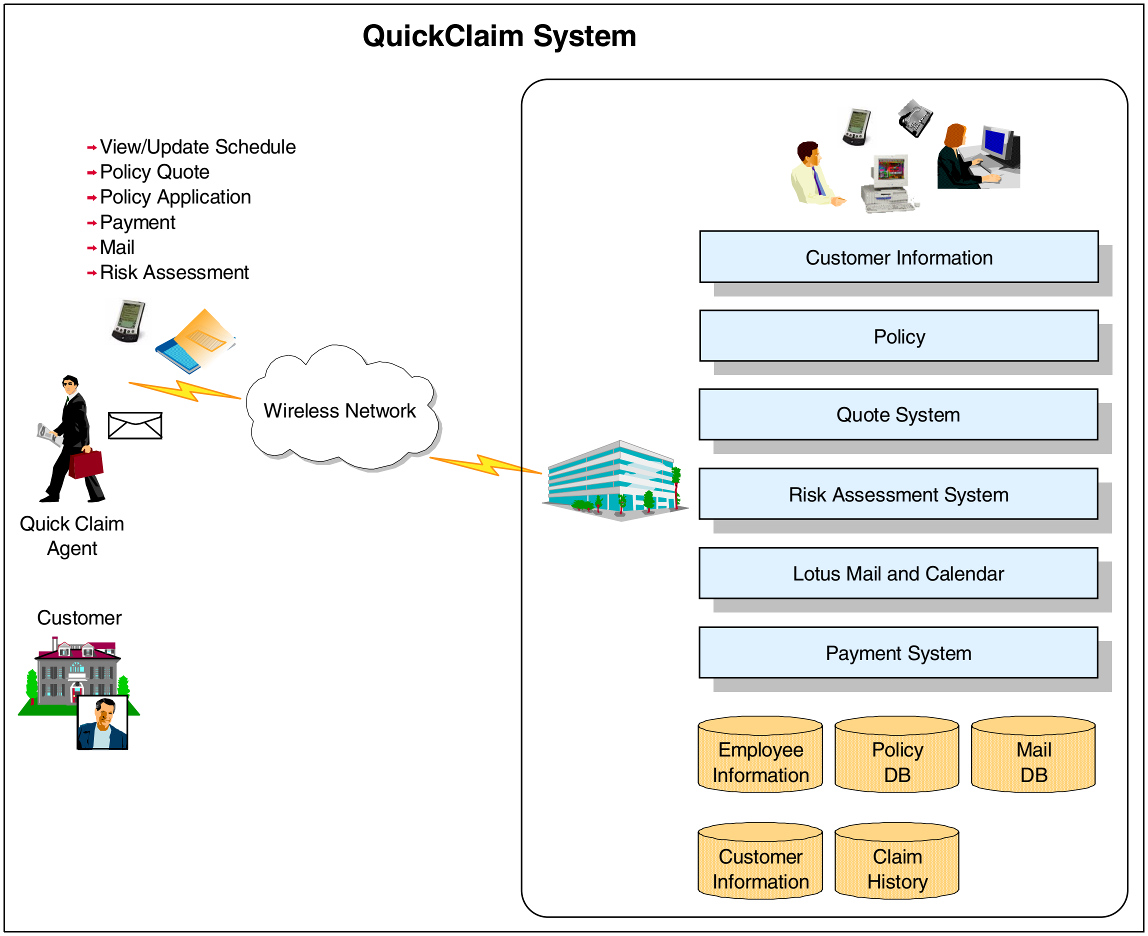


Figure 4-1 Architectural vision

4.1.3 Background information

The following summarizes QuickClaim's existing Information Technology infrastructure:

- QuickClaim uses IBM SecureWay LDAP Directory to implement user registry, and implements WebSphere Application Server security for authentication and access control.
- QuickClaim's policy application is a WebSphere application that stores all policy application data in a DB2 database residing at the company's HQ office.

- The Risk Management and Quote applications are also WebSphere Application Server based. The access to all WebSphere applications is controlled by WebSphere Application Server Security, which uses IBM SecureWay Directory Server.
- QuickClaim's payment application is an MQSeries application using IBM MQSeries to connect to financial institutions. The payment application is implemented using MQSeries client/server architecture and MQSeries security.
- QuickClaim uses Lotus Notes e-mail, calendaring systems and PIM applications.
- A recent survey indicates Pocket PCs are employees' choices for personal digital assistants (PDAs) and mobile devices.

Figure 4-2 depicts QuickClaim's existing IT infrastructure.

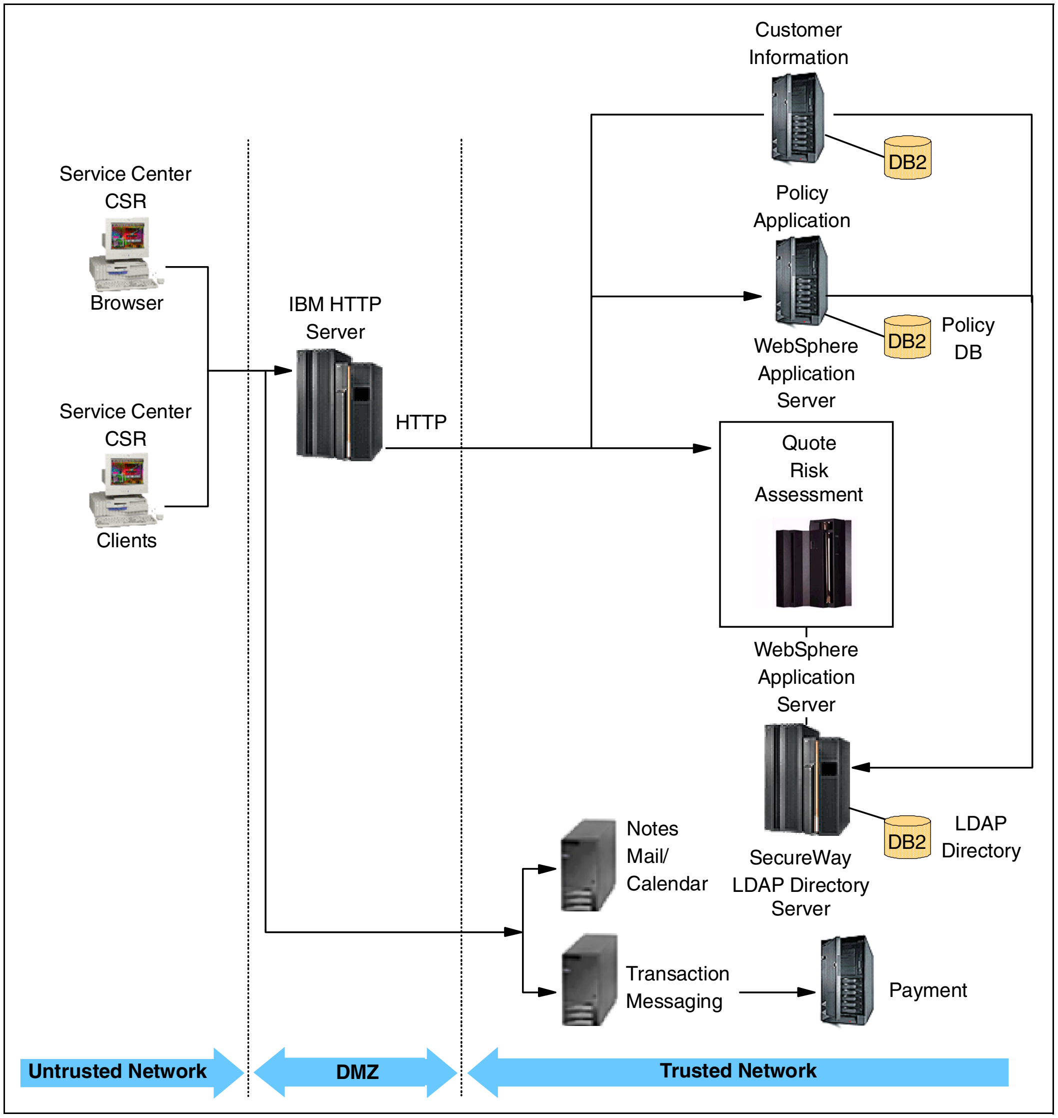


Figure 4-2 QuickClaim technical architecture baseline

4.2 Architecture principles and requirements

QuickClaim uses an Enterprise Architecture (EA) method, such as the IBM Enterprise Architecture method or Zachman's framework, to develop its IT strategy and architecture. Enterprise Architecture is a framework for both business and IT, which guides investment and design decision, and specifies processes, standards, interfaces and common services for the deployment and management of IT assets, in support of business objectives. It defines an environment in which both future known systems and solutions to unforeseen requirements can be built.

Enterprise Architecture is the linkage between the business strategy, the IT strategy and IT implementation. It is an integral part of the strategic planning process and provides guidance to those wishing to implement and manage business and IT resources effectively.

Enterprise Architecture has four parts (business, data, application and technical architectures), which are normally defined in sequence and developed over time, as the architecture evolves to meet changing business and IT objectives. This chapter focuses only on the aspect of the technical architecture. It is not the intent of this book to delineate Enterprise Architecture's principles and practices, but to provide some background and explanation for the concepts and terminology introduced in this chapter.

The architecture principles are the underlying general rules that hold true across the architecture. In a certain sense, architecture principles define the spirit of the architecture, in that they are an attempt to capture the thinking behind it. When an architecture is built upon a coherent set of architecture principles, it has a wholeness and consistency that it otherwise would not have.

On publication of the architecture, the architecture principles can be used to explain and justify why things are the way they are. To convey the impact of implementing the principle on the business, each principle should be further developed to include statements of motivation and impact. This process further validates the feasibility of the architecture principles under development. QuickClaim followed the Enterprise Architecture process to establish and publish their technical architecture principles summarized in the following section.

4.2.1 Architecture principles

Principles are general, fundamental, and durable statements on the role, use or direction of the business and IT, and are developed for each area of the architecture to further direct the deployment of IT in support of the business. The following are QuickClaim's guiding principles for technical architecture:

- To maximize potential re-use, IT solutions will be assembled from structural components. The preferred methods for obtaining application/functional components are to reuse existing, to purchase new, or thirdly to build them.
- To improve time to market and reduce total cost of ownership, IT solutions will leverage commercially available and proven solutions or technology components.
- IT will provide the sales force with timely, accurate and anywhere access to sales decision support information and systems.
- Information and customer databases are corporate assets and should be captured accurately, securely stored and managed in a way that will allow the appropriate level of access and sharing across the enterprise.
- The architecture must be capable of delivering the applications and data necessary for their users to do their jobs. The systems will support multi-channel and multi-mode accesses defined by the business needs.
- All applications will utilize a system-wide centrally managed common directory and security architecture to the extent technically and economically feasible.
- The IT Architecture is a living document that will evolve to accommodate changes in business and technology.
- The IT architecture will be flexible and easily extensible to support any new technology - for example, new mobile network technology and new mobile devices.

4.2.2 Requirements

Architecture requirements are developed using gap analysis based on the prioritization of strategic initiatives, and the current business and technology baseline. Strategic initiatives are usually developed by aligning architectural guiding principles and technology initiatives with business goals. QuickClaim has identified the B2E mobile sales force solution as one of its top strategic initiatives. Following the Enterprise Architecture process, QuickClaim has identified the following business and technical requirements:

- Support online transaction processing for credit-card payments initiated from a PDA.
- Allow agents to enter a policy application offline using their PDA and replicate the data to the enterprise's policy DB2 database at a later time.
- Allow agents to synchronize their e-mail, calendaring and PIM applications on the PDA with the corresponding enterprise systems.
- Allow agents online access to risk assessment and quote application using mobile devices.

- The system should support the scalability and availability requirements of the current business needs and projected growth.
- The system should be adaptable to new mobile networks and new mobile devices.
- The system and applications should be optimized to provide reasonable response time for agents using mobile devices to access sales support systems over the wireless networks.
- The system and applications should be designed to accommodate the fragile mobile communications network.
- Software and applications running on the devices should be frugal in its use of system resources.
- The system should provide facilities to manage mobile devices. This includes device configuration, software distribution and management for mobile devices.
- New applications should integrate with a centralized security management systems. Existing strategic applications should migrate towards a centralized security management system when technical and economical benefits out weigh the cost of implementation. Existing tactical applications can be exempted.

4.3 Architectural overview

This solution leverages WebSphere Everyplace Suite Enable Offering V.1.1.2 to provide many of the architecture building blocks commercially available, and is consistent with QuickClaim's architecture goal as stated in "Architecture principles and requirements" on page 157. This architecture uses the following key WebSphere Everyplace Server Enable Offering features:

- MQSeries Everyplace to provide once only and assured payment transaction from mobile devices
- IBM Mobile Connect for Lotus Notes mail, PIM applications and database synchronization. (Enable Offering optional feature)
- Device Management functions for device configuration and software distribution to mobile devices
- Integration with a centralized security management system for authentication, also known as third-party authentication support provided by the Enable Offering
- WebSphere Transcoding Publisher to dynamically adopt content for multi-channel support and mobile devices

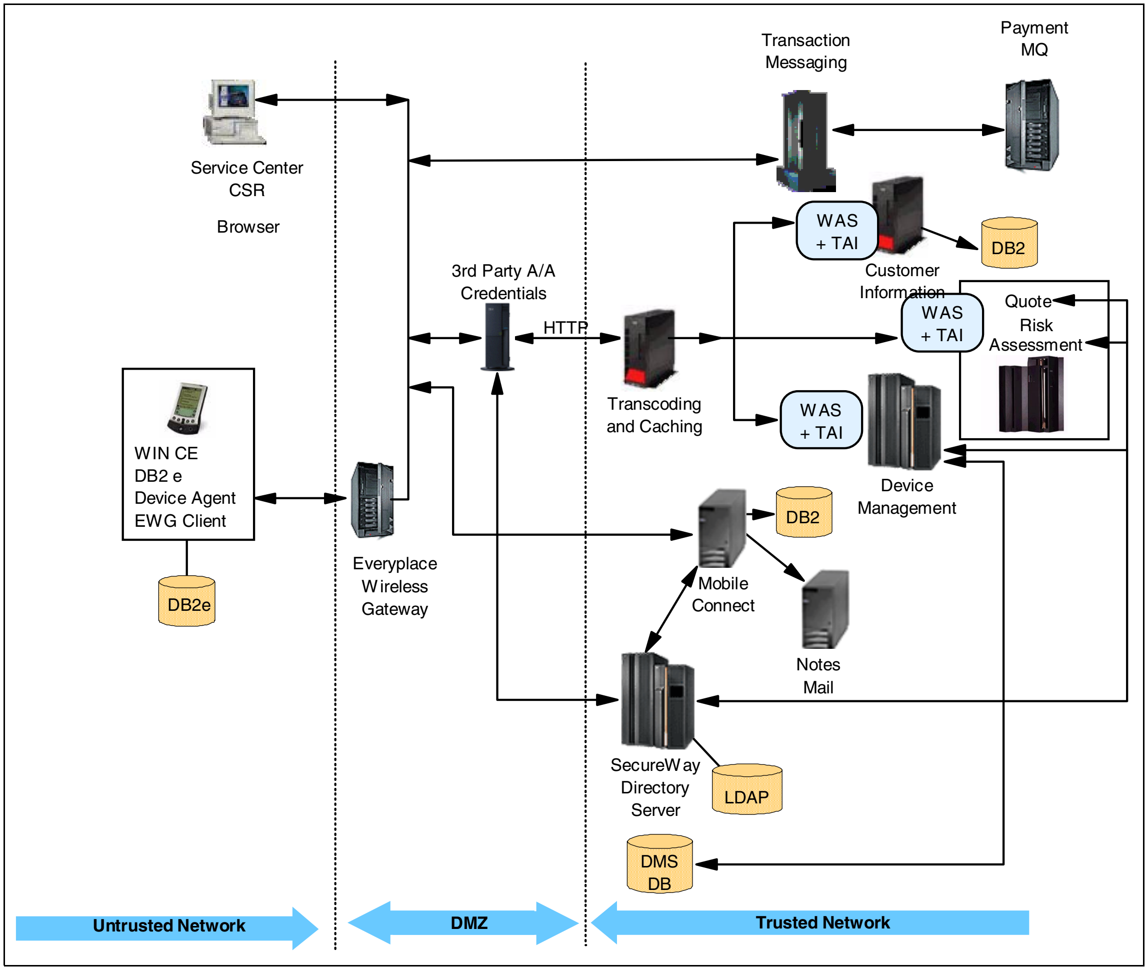


Figure 4-3 Architectural overview

The following additional products and components are added to complete the overall architecture:

- DB2 Everyplace to provide policy application database and customer contact database on mobile devices
- MQSeries Everyplace Bridge to MQSeries server
- MQSeries Everyplace client on devices
- A centralized security management system, for example, Tivoli SecureWay Policy Director or Netegrity SiteMinder

- Wireless gateway hosted by a wireless network connectivity provider, such as a telecommunications company or a network service provider to connect mobile devices to QuickClaim's enterprise systems.

  QuickClaim can choose to implement Everyplace Wireless Gateway (EWG) in house to provide wireless connectivity. Everyplace Wireless Gateway is an optional feature of Enable Offering V1.1. Refer to 1.3.1, "Everyplace Wireless Gateway" on page 12 for an overview of Everyplace Wireless Gateway. "Wireless gateway" on page 215 addresses integration considerations for using Everyplace Wireless Gateway with Enable Offering V 1.1.2.

Requirements allocation matrix

Table 4-1 Requirements mapping to architecture building blocks

Requirements	Architecture building block
Credit card transaction from PDA	MQe PDA client, IBM WebSphere Everyplace Server Everyplace Server (MQe+bridge)
Policy application and DB on PDA	DB2e
Policy database synchronization with DB2	IBM Mobile Connect V2.5
Lotus Notes e-mail, calendar, and PIM synchronization	IBM Mobile Connect V2.5
PDA access to risk management, quote and customer information applications	WebSphere Everyplace Server Enable Offering (WebSphere Transcoding Publisher)
Scalability and availability	WebSphere Everyplace Server Enable Offering (Edge Server and WebSphere Application Server Workload Manager)
System adaptable to new mobile networks and devices	WebSphere Everyplace Server Enable Offering (WebSphere Transcoding Publisher, Device Manager Server)
System responsiveness to PDA	WebSphere Everyplace Server Enable Offering (Edge Server Caching Proxy), Wireless Gateway or wireless network
System tolerates fragile mobile communications network	WebSphere Everyplace Server Enable Offering (MQe), IBM Mobile Connect, Wireless Gateway or wireless network
PDA software frugal on the use of resources	DB2e, MQe
Facilities for device configuration and software distribution to PDA	WebSphere Everyplace Server Enable Offering (Device Management Server)

Requirements	Architecture building block
Support centralized security management system	WebSphere Everyplace Server Enable Offering (third-party authentication)

This solution is a specific implementation instance of the end-to-end WebSphere Everyplace Server Enable Offering architecture framework.

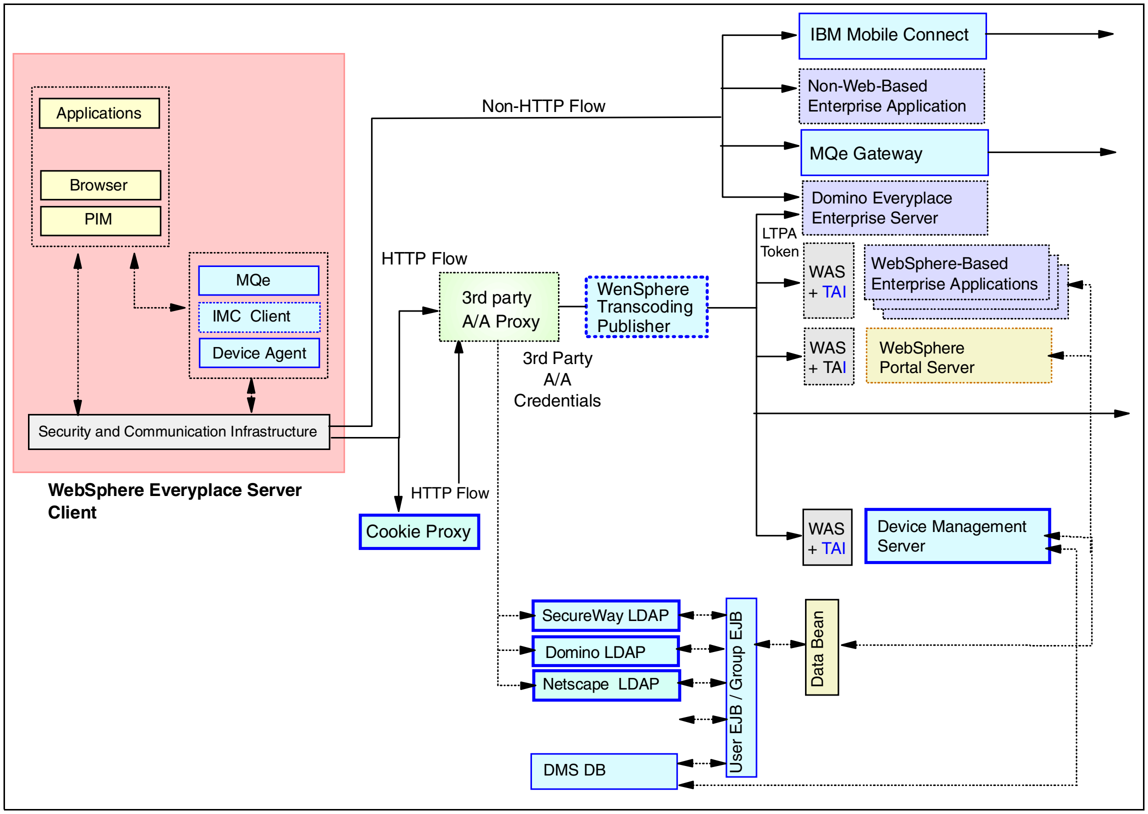


Figure 4-4 Everyplace Suite Enable Offering architecture framework

“Architecture building blocks” on page 179 provides more details on key architectural components referenced in Table 4-1 on page 161 and how each component integrates into the overall architecture. The following section describes the overall architecture in more detail and provides rationale and justifications for these architectural decisions.

4.4 Architecture decisions

Every IT architecture should have a solid architecture underpinning. This section summarizes the architecture considerations and decisions for key areas of the overall architecture.

4.4.1 Security architecture

QuickClaim's current system uses WebSphere Application Server's security feature with IBM SecureWay Directory Server. Figure 4-5 depicts WebSphere Application Server V3.5 security architecture.

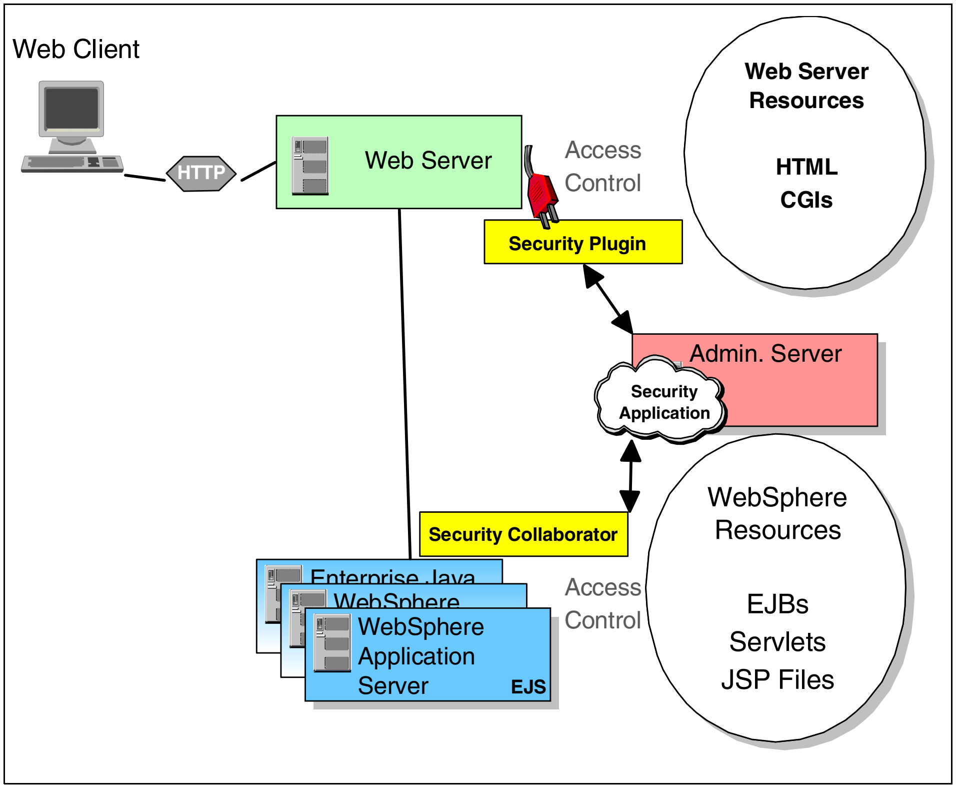


Figure 4-5 WebSphere Application Server V3.5 security architecture

When a user requests a secured servlet URL, the security collaborator in the WebSphere Application Server issues an HTTP 401 challenge back to the browser, through the Web server, requiring the user to enter a user ID and password to access the URL. The security collaborator performs authentication by delegating the task to the security server using the authentication data (user

ID and password) received from the user. On successful authentication, the security collaborator consults the security application to determine whether the user has permissions necessary to access the URL. If authorization succeeds, the security collaborator sets up a security context with the user's credential information, and passes on the request to the servlet engine to service the request. The Security Plug-in, Security Collaborator, and security application depicted in Figure 4-5 on page 163 are integral parts of the WebSphere Application Server V3.5.

Third-party authentication

Beginning with WebSphere Application Server V3.5 PTF 3, WebSphere Application Server was enhanced to support third-party security solution, such as Tivoli SecureWay Policy Director by providing a trusted association solution between WebSphere Application Server and a Reverse Proxy Security Server (RPSS), such as Policy Director. The intent of trusted association design is to have RPSS exist as the exposed entry point. The RPSS authenticates the principle and provides coarse-grained access authorization. WebSphere Application Server security should be implemented to further exploit WebSphere Application Server fine-grained access control. Figure 4-6 depicts WebSphere Application Server V3.5's support for RPSS.

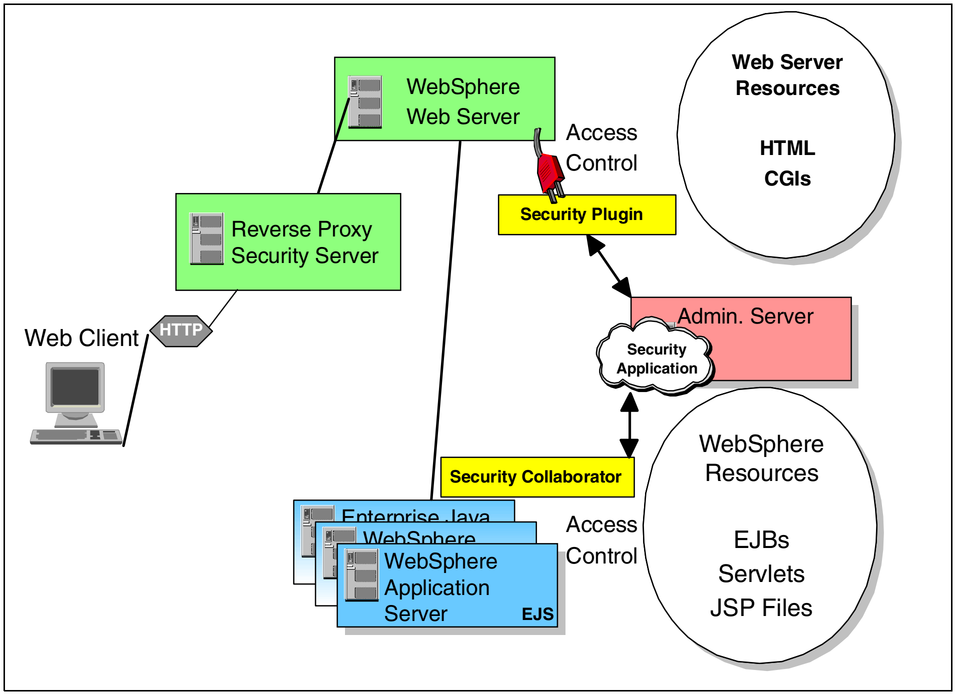


Figure 4-6 WebSphere Application Server security with RPSS authentication

In this configuration, the initial authentication is performed by RPSS, and the user's credential is added to the request headers and forwarded to WebSphere. WebSphere uses the identity information set up by the RPSS in the request header to perform credential mapping and further downstream authorization.

The RPSS server attaches some information about the authenticated user to the HTTP request and forwards the requests to WebSphere. Such a credential should contain a valid user name in the WebSphere user registry, the IBM SecureWay Directory, in QuickClaim. When WebSphere receives the request, the security collaborator in the WebSphere Application Server is configured in such a way that it knows that an RPSS authentication service front-ends WebSphere Web server. Based on the trust relationship with the RPSS, the WebSphere application server retrieves the user information present in the HTTP headers. On successful validation of the user, the security collaborator consults the security application to determine whether the user has the permissions necessary to access the URI. If the authorization succeeds, the security collaborator sets up a security context with the user's credential information and passes on the request to the servlet engine to service the request.

The Trust Association means WebSphere Application Server "trusts" the authentication performed by the third-party authentication tool so that no duplicate authentication will be performed, and accepts user identification passed from authentication tool. Trust association is specific to the characteristics of the RPSS. Therefore, for every type of RPSS, a trust association type is associated and an implementation of the interceptor is required to handle the specific details. The Trust Association Interceptor (TAI) is designed to address these requirements. The TAI is the code to intercept the incoming data from the authentication tool, instruct WebSphere Application Server not to authenticate again and to trust the user identity from the authentication tool.

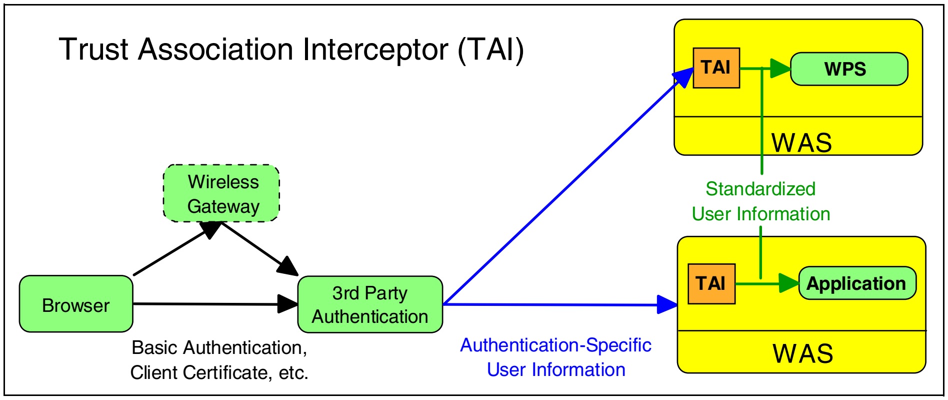
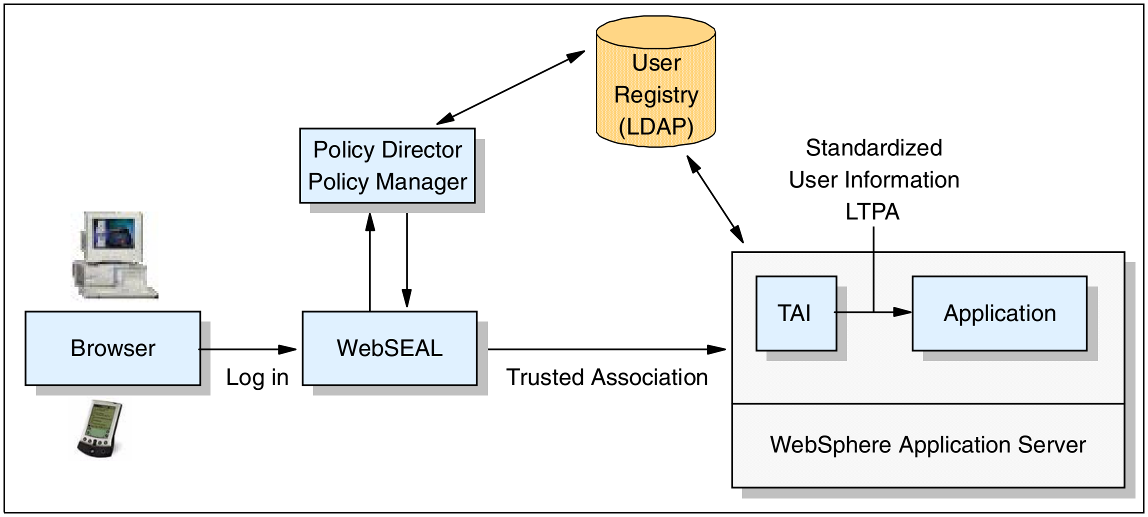


Figure 4-7 Trust Association Interceptor for WebSphere Application Server

Based on the trust model and the values available in the HTTP request stream, it is the responsibility of the interceptor to validate the request stream. WebSphere Application Server V3.5 supplies the TAI for Tivoli Policy Director, and Enable Offering supplies the TAI for Netegrity's SiteMinder. Figure 4-8 depicts WebSphere Application Server integration with Policy Director. Figure 4-14 on page 182 depicts WebSphere Application Server integration with Netegrity's SiteMinder, and "Third-party authentication using Netegrity's SiteMinder" on page 179 provides an overview of Netegrity's SiteMinder and integration considerations.



User Registry (LDAP)
Policy Director Policy Manager
Standardized User Information LTPA
Browser
Log in
WebSEAL
Trusted Association
TAI
Application
WebSphere Application Server

Figure 4-8 WebSphere Application Server integration with Policy Director

Alternatively, a third-party security plug-in using an authorization tool, such as Policy Director's aznAPI, can handle the authentication exit before WebSphere Application Server.

Centralized security management allows administrators to easily manage large numbers of user privileges and entitlements (access control) across all applications and platforms, eliminating the need for proprietary and redundant security systems. Therefore, many enterprises are starting to implement a centralized security management system.

QuickClaim's architecture committee has just completed a study to select a centralized security management system. Their preliminary recommendation for this centralized security management system is Netegrity's SiteMinder. Not all existing applications can be easily and cost-effectively converted to use a centrally managed security system. Therefore, QuickClaim has established an architectural goal to gradually migrate existing strategic applications towards a centralized security management system. To support QuickClaim's architectural principle stated in “Architecture principles” on page 157 and to comply with requirements stated in “Requirements” on page 158, this architecture includes the integration with a centralized security management system, such as Netegrity’s SiteMinder. SiteMinder acts as a Reverse Proxy Security Server (RPSS) front-ending WebSphere Application Server.

Enable Offering V1.1 supports the following authentication products:

- Tivoli SecureWay Policy Director Version 3.7
- Netegrity SiteMinder Version 4.5.1 and 4.6.1

QuickClaim’s Quote, Risk management and Customer Information applications are all WebSphere V3.5 applications using the WebSphere security feature with Secureway LDAP directory. Therefore, all these applications can integrate with Netegrity’s SiteMinder using the RPSS and TAI functions described in this section. SiteMinder’s TAI is shipped with Enable Offering V.1.1. For non-WebSphere applications, Netegrity SiteMinder provides tools and agents to allow application integration with SiteMinder security system, refer to “Third-party authentication using Netegrity’s SiteMinder” on page 179 for more details about Netegrity’s SiteMinder and how Enable Offering V.1.1 integrates with SiteMinder.

4.4.2 Enterprise directory architecture

QuickClaim has already established an LDAP directory server. As documented in “Architecture principles” on page 157, all new applications should use a centralized LDAP directory server. The LDAP protocol was originally intended as a lightweight alternative to DAP for accessing X.500 directories. For an introduction to LDAP concepts and architecture refer to *Understanding LDAP*, SG24-4986.

User directory support

As depicted in Figure 4-9, Enable Offering V 1.1 supports existing customer’s user registry.

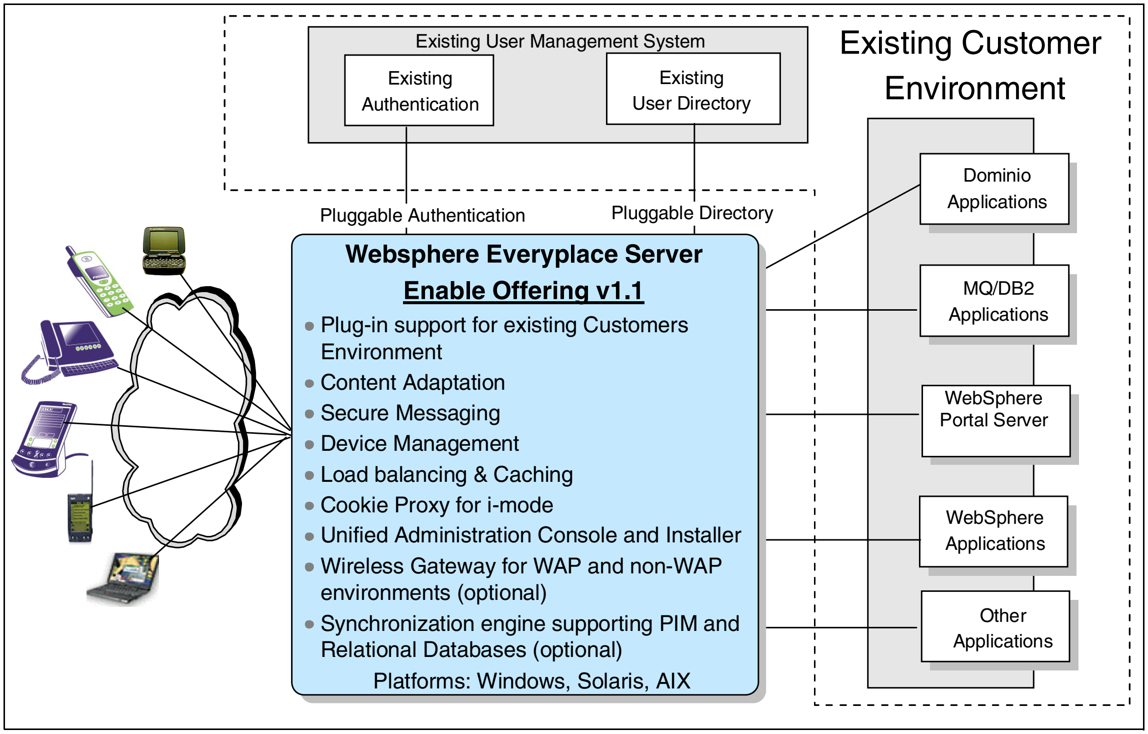

Existing User Management System
Existing Authentication
Existing User Directory
Existing Customer Environment
Pluggable Authentication
Pluggable Directory
Websphere Everyplace Server
Enable Offering v1.1
Plug-in support for existing Customers Environment
Content Adaptation
Secure Messaging
Device Management
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 4-9 Enable Offering integrates with existing customer environment

Enable Offering V1.1.2 supports a standard set of LDAP schema and attributes as detailed in Table 4-2 on page 170. If the attributes and schema required by Enable Offering are defined in the QuickClaim's user LDAP directory. Enable Offering will support QuickClaim's user LDAP directory without modification. Otherwise, additional work may be required to resolve the differences, such as migrating QuickClaim's existing LDAP schema to accommodate the Enable Offering required attributes and LDAP schema, or providing customized user directory support using JNDI calls in lieu of Generalized User Directory Access (GUDA). Alternatively, QuickClaim may choose to extend UserEJB to support QuickClaim's LDAP schema. WebSphere Portal Server V1.2 provides the UserEJB source code and instructions.

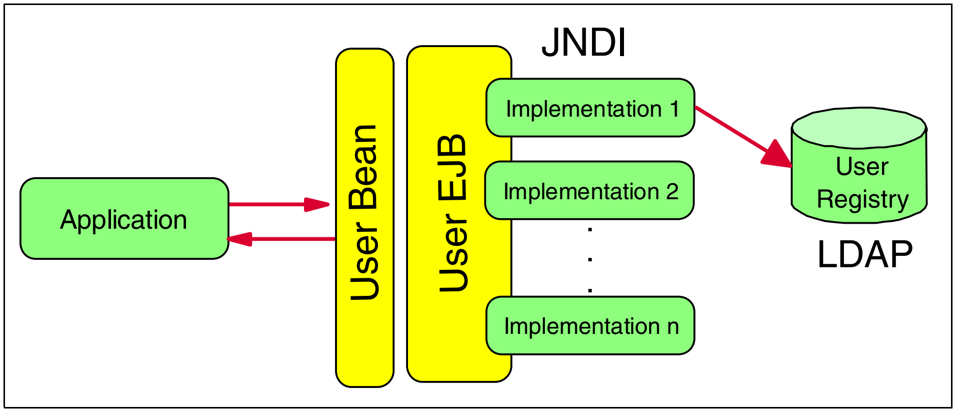


Figure 4-10 Generalized User Directory Access

The Enable Offering uses Generalized User Directory Access (GUDA) to provide user directory support. GUDA is a generalized mechanism used by WebSphere applications for referring to user information in directory products from IBM or other vendors. GUDA is a set of Java classes included in Enable Offering V 1.1 to provide a higher level of abstraction for user and group information. It is intended to eliminate the need for applications to implement JNDI programming calls to interface with different user directories. These classes provide a generic and simplified access to information stored in various user directories. It allows application developers to develop applications that are independent of the underlying user directory schema. This provides a higher level of user and group abstraction layer to developers without requiring them to implement JNDI calls to different variety of user directories.

GUDA is installed automatically by Enable Offering with the selection of the “User Directory Access and Authentication” component during the install. The GUDA Java classes are packaged in the um.jar file. This file contains Enterprise JavaBeans (EJBs) and provides the following four APIs:

- User

  The User class abstracts user information stored in the user directory. Each instance of the User class represents one node that the user has defined in the user directory. The User class provides methods for listing all the groups that a user belongs to, and for referring to and updating the values of the attributes of a user.

- UserManager

  The UserManager class provides methods for determining whether the target user exists, searching for users that satisfy the given search filter, and listing all users. It provides a method for updating the user information stored in the user directory.

- Group

  The Group class abstracts group information stored in the user directory. Each instance of the Group class represents one node of a group defined in the user directory. The Group class provides methods for listing all the members in that group, and for referring to the values of attributes of the group.

- GroupManager

  The GroupManager class provides methods for determining whether the target group exists, searching for groups that satisfy the given search filter, and listing all groups.

Enable Offering V1.1.2 has been tested with IBM SecureWay Directory 3.2, Netscape Directory 4.1, Lotus Domino Server V5.07a and Microsoft Active Directory, which is included in Windows 2000.

The configuration file of the Generalized User Directory Access (um.properties) is configured by Enable Offering V1.1.2 for SecureWay Directory or Netscape Directory server. If Domino or Active Directory is used as a user directory, this configuration file needs to be updated with the corresponding schema - object classes and attributes. The LDAP schema supported by Enable Offering V1.1.2 are as follows:

Table 4-2 LDAP schema and attributes supported by Enable Offering V 1.1.2

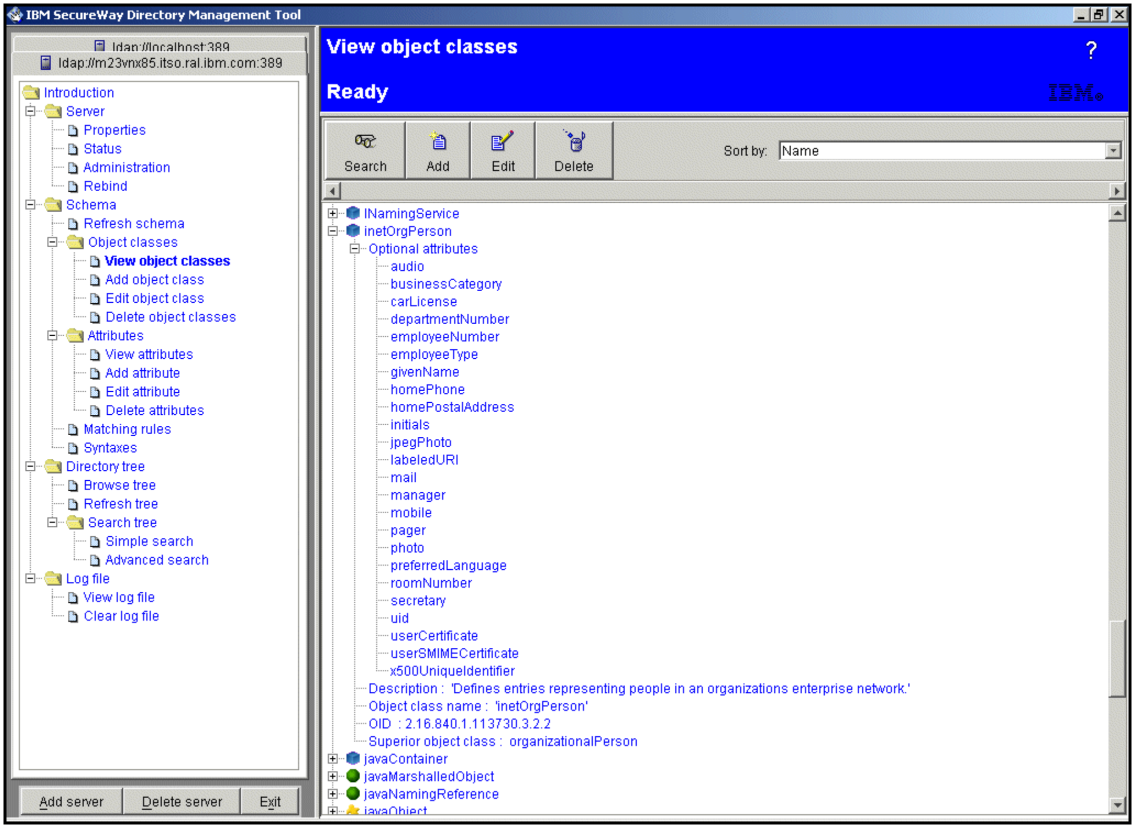
	SecureWay Directory	**Netscape Directory**	**Domino**	**Active Directory**
Object Class	`inetOrgPerson`	`inetOrgPerson`	`inetOrgPerson`	`user`
Attribute	`uid`	`uid`	`uid (shortname)`	`sAMAccountName`
Object Class	`groupOfUniqueNames`	`groupOfUniqueNames`	`groupOfNames`	`group`
Attribute	`uniqueMember`	`uniqueMember`	`member`	`member`

Because Enable Offering V1.1.2 installs um.properties file into different directories to support the following functions, the user must manually copy the um.properties files to synchronize the changes in settings:

- For Generalized User Directory Access (GUDA), the um.properties file is installed in the IBMEEPPA subdirectory (/usr/lpp/IBMEEPPA for AIX, /opt/IBMEEPPA for Solaris, \program files\IBM\IBMEEPPA for Windows).
- For Device Manager Server, the um.properties file is installed in the TivDMS/Class subdirectory (/usr/lpp/TivDMS/Class for AIX, /opt/TivDMS/Class for Solaris, \program files\IBM\TivDMS/Class for Windows), in addition to the GUDA copy of um.properties file.

  The Tivoli Device Manager Server console jar file needs to be recreated to use the updated um.properties file using makeConsoleJar.sh for AIX/Solaris or makeConsoleJar.bat for Windows.

The inetOrgPerson is a standardized scheme and available on many LDAP implementations, except Microsoft's Active Directory. However, Microsoft Active Direcotry can be supported by customizing settings in the um.properties file. Enable Offering V1.1.2 expects LDAP server is using inetOrgPerson to store core user data. The User object can refer to the inetOrgPerson object in any position in the directory tree. The inetOrgPerson directory entries are expected to be created by either a third-party tool, or by WebSphere Portal Server V1.2 using the UserManager class. Figure 4-11 on page 172 shows Secureway Directory's inetOrgPerson object class:

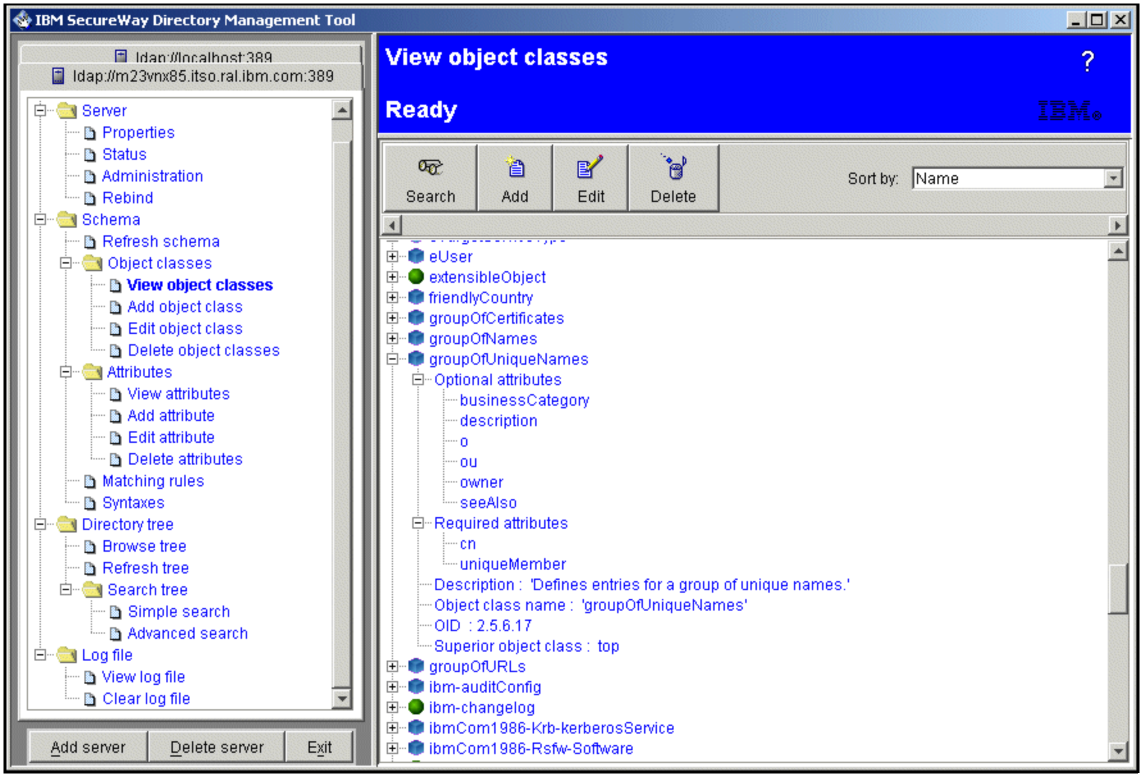


Sort by: Name

```
INamingService
inetOrgPerson
  Optional attributes
    audio
    businessCategory
    carLicense
    departmentNumber
    employeeNumber
    employeeType
    givenName
    homePhone
    homePostalAddress
    initials
    jpegPhoto
    labeledURI
    mail
    manager
    mobile
    pager
    photo
    preferredLanguage
    roomNumber
    secretary
    uid
    userCertificate
    userSMIMECertificate
    x500UniqueIdentifier
  Description : 'Defines entries representing people in an organizations enterprise network.'
  Object class name : 'inetOrgPerson'
  OID : 2.16.840.1.113730.3.2.2
  Superior object class : organizationalPerson
javaContainer
javaMarshalledObject
javaNamingReference
javaObject
```

Figure 4-11 inetOrgPerson object class

The Group object can refer to a groupOfUniqueNames object in any position in the directory tree. In Enable Offering V1.1.2, the Group object is read only. The new groupOfUniqueNames directory entries are expected to be created by a third-party tool, or by WebSphere Portal Server V1.2 using the GroupManager class. Figure 4-12 on page 173 shows SecureWay Directory's groupOfUniqueName object class:



Description : 'Defines entries for a group of unique names.'
Object class name : 'groupOfUniqueNames'
OID : 2.5.6.17
Superior object class : top

Figure 4-12 groupOfUniqueNames object class

Enable Offering V.1.1's GUDA implementation uses the same user management framework provided by WebSphere Portal Server (WPS) V1.2. Therefore, WebSphere Portal Server V1.2's user enrollment and management functions integrate seamlessly with Enable Offering V1.1. This allows QuickClaim to gradually migrate towards a business-to-employee multi-channel and multi-modal portal architecture with a centralized security management system. Figure 4-13 on page 174 depicts the relationship between Enable Offering V 1.1 and WebSphere Portal Server 1.2 in the area of user and group management support for LDAP directories.

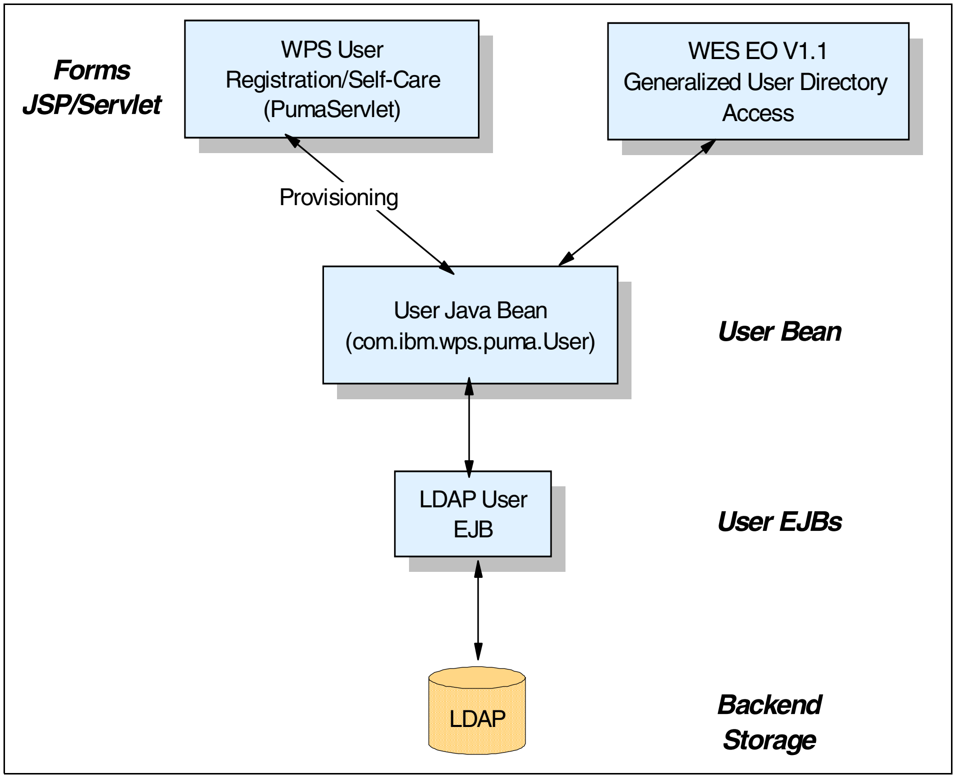


Figure 4-13 Portal Server and Enable Offering user management

If the user directory schema (object class and attribute) is different from the V1.1.2 supported schema as tabulated in Table 4-2 on page 170, the GUDA framework may not support this schema. Custom user directory support using JNDI calls can be developed, in lieu of Generalized User Directory Access (GUDA), to support schemas that are not supported by Enable Offering V1.1.2. Alternatively, the UserEJB can be extended to support different LDAP schema. WebSphere Portal Server V1.2 provides the UserEJB source code and instructions.

4.4.3 Multi-channel application architecture

QickClaim's Quote, Risk Management, and Customer Information applications were developed for PC browser Web clients. These applications now need to support both the PC and mobile devices with much smaller screens, different form factors, and different user interfaces. To minimize the impacts to these existing applications and to reduce time to market, WebSphere Transcoding Publisher is leveraged to transcode the existing application-generated HTML content to fit the mobile device's presentation. WebSphere Edge Server can be leveraged to cache the transcoded content to improve performance. This chapter focuses on the conceptual architecture for pilot configurations. Refer to Chapter 7, "Scalability" on page 301, Chapter 8, "Availability" on page 345, and Chapter 6, "Operations model" on page 269 for more detailed discussions of scalability, availability and performance considerations for deployment. "WebSphere Transcoding Publisher integration" on page 209 describes how V3.5 integrates with Enable Offering.

Alternatively, QuickClaim may choose to implement access to Risk Management, Quote, and Customer Information applications using WebSphere Portal Server V1.2. WebSphere Portal Server portlets can be designed to render directly to the PC browsers and PDA browsers in lieu of WebSphere Transcoding Publisher. None of the QuickClaim's applications require deck fragmentation support; therefore, WebSphere Transcoding Publisher V3.5 is not required, if WebSphere Portal Server is used to format content to render directly to the devices. WebSphere Transcoding Publisher V3.5 provides fragmentation support for WML and i-mode, but not for HTML.

For speed to market, this architecture uses WebSphere Transcoding Publisher V3.5 for content transcoding. However, WebSphere Portal Server V1.2 integrates seamlessly with Enable Offering V1.1. Therefore, this architecture can be easily and naturally migrated towards portlet architecture. When the time comes for QuickClaim to add new applications, it will be a good time for QuickClaim to consider migrating towards portlet architecture using WebSphere Portal Server.

4.4.4 Device architecture

Device selection

There are many factors that an enterprise should consider in selecting the mobile devices when planning for an end-to-end mobile solution for sales force applications. Many of these decisions are interdependent and not stand-alone decisions. For example, a PDA will more likely require CDPD (Cellular Digital Packet Data) network connectivity and WAP phones to work over cellular circuit switched connections. QuickClaim was considering Palm and Pocket PC devices. The following summarizes some of the key considerations:

- Connectivity

  All handheld devices are wireless capable. There are a variety of wireless networks available (refer to Everyplace Wireless Gateway for network types). The decision for connectivity centers around the application characteristics (for example, throughput required) and usage scenario (for example, voice, data or bi-modal). QuickClaim's sale force support applications are data centric and work best with on-demand IP connectivity. Therefore, CDPD is a good candidate for today's deployment and is widely supported by PDA devices and service providers. However, this solution architecture supports any type of IP connectivity and can be easily extended to accommodate different network connectivity.

- Coverage

  Find network carriers that provide the necessary geographical coverage for wireless access by mobile sales personnel. For example, AT&T and Verizon provide CDPD networks with about 90% coverage of the U.S. population. The PDAs and handheld PC applications can be designed to provide some level of offline application availability when the device goes out of coverage areas.

- Software availability

  Table 4-3 provides a comparison of application availability:

Table 4-3 Application availability comparison for devices

Software	Color Pocket PC	Color Palm
Web Browsing software	Internet Explore for pocket PC	Web clipping on Palm VII; limited graphical browser through third-party software loaded into RAM
Operating System	WinCE	Palm OS
MQe Client	Queue manager support	No queue manager support
e-Mail including POP3 and IMAP4	Yes	Requires third-party software for POP3 and IMAP 4 e-mail access
Music player	MP3 and Windows Media player	No
Financial Software	Microsoft Money for Pocket PC in system ROM	Expense application only.

- Application development tools

  Mobile application complexity can range from a simple form-based application developed with a rapid application development tool, such as Satellite Forms Enterprise Edition for Palm OS devices, to more complex projects that require C++. The following are some popular development tools:

  - Embedded Visual Basic and C++ for CE or Pocket PC devices
  - Satellite Forms from PumaTech for a Palm application
  - AppForge for VB development on Palm OS devices
  - Metroworks Code Warrior for Palm OS
  - VisualAge for Java Micro Edition for Java application

- Form factor

  What is the size requirement for the device (for example, phone or PDA)? Does it need to be rugged enough to sustain accidental drops? Does it require an integrated keyboard? A handheld PC or PDA probably works the best for mobile insurance sales agents.

- Device performance

  Table 4-4 provides a sample of device features and performance comparison.

Table 4-4 Device feature/performance comparison

Feature/performance	Color Pocket PC	Color Palm
Memory	16 to 64 MB	8 MB (expandable)
Processor	32 bit up to 206 MHz	33 MHz
Screen resolution	320 X 240 resolution	160 X 160 resolution
Expansion slot	PCMCIA cards, up to 1GB storage, CompactFlash, multimedia, Secure Digital	Multimedia, Secure Digital
USB Connection	Yes, except for HP 520	Yes
Ethernet Connection	Yes, third-party (CompactFlash card)	No
IrDA modem protocol	Yes	Yes
Natural handwriting recognition	Yes	No, Use graffiti language or install third-party application in RAM.
Voice Recording	Yes	No

  WinCE software requires more resources than Palm software; therefore, the memory and processors capabilities should be a relative and not absolute comparison.

- Cost

  QuickClaim decides to adopt a color Pocket PC (a WinCE device) as the standard mobile PDA for its sales agents after conducting a comparative analysis and a survey. The availability of MQe queue manager on WinCE devices was one of the major reasons for selecting Pocket PCs.

Device software and applications

DB2 Everyplace (DB2e) is selected as the database for the PDA under the criteria and requirements established in “Requirements” on page 158. Refer to “DB2 Everyplace for devices” on page 182 for DB2e functions and features. QuickClaim will develop a PDA version of the Policy application that allows the sales agent to complete the Policy application offline using a PDA. This Policy application stores data into PDA’s local DB2e database. The DB2e data can be replicated to QuickClaim’s Policy DB2 database at headquarters when the device is connected to a wireless or wireline network.

QuickClaim will develop a local DB2e application that allows sales agent to look up Customer Information databases offline using devices. This local DB2e is replicated with the Customer Information DB2 database at headquarters.

MQSeries Everyplace (MQe) is selected as the local queue manager for the PDA. This allows QuickClaim’s sales agent to submit a credit card transaction to the back-end MQ-based payment system. If the network connection is not available or unstable, MQSeries Everyplace provides “once only” and assured delivery of this transaction using the device’s local queue manager. An MQe application will be developed for WinCE using queue manager and a local queue. This allows asynchronous messaging and offline access to the PDA’s credit card payment application. Refer to “MQ Everyplace V1.2 for devices” on page 189 for more details on MQe and integration considerations for QuickClaim.

4.4.5 Synchronization architecture

QuickClaim has the following synchronization requirements that are derived from the requirements stated in “Requirements” on page 158:

- Lotus Notes e-mail, calendar, address book and PIM applications
- DB2e synchronization with Policy DB2 database
- DB2e synchronization with Customer Information DB2 database

IBM Mobile Connect V2.5, an optional Enable Offering feature, can be leveraged to provide a common solution for all of the QuickClaim’s synchronization requirements. Refer to “IBM Mobile Connect V2.5 for PIM and DB2e synchronization” on page 184 for more details on using IBM Mobil Connect for synchronization and integration considerations for QuickClaim.

4.4.6 System management architecture

QuickClaim’s new system management architecture will extend QuickClaim’s existing system management architecture to provide the following additional management functions:

- Device enrollment and management
- Software distribution and management for devices

Enable Offering provides a comprehensive set of device management and software distribution functions. Refer to “Device management and software distribution” on page 198 for details about Enable Offering’s Device Manager Server, software distribution functions, and integration considerations for QuickClaim.

4.4.7 Scalability and availability architecture

The scalability and availability of this solution is built on top of the scalability and availability features of individual architecture building blocks, for example, Edge Server Load Balancer, Edge Server Caching Proxy, and WebSphere Application Server’s Work Load Management. Refer to Chapter 7, “Scalability” on page 301 and Chapter 8, “Availability” on page 345 for more in-depth discussions.

4.5 Architecture building blocks

WebSphere Enable Offering V1.1 provides the following key architectural building blocks for QuickClaim’s B2E mobile solution. This significantly reduces QuickClaim’s time to market and total solution cost as articulated in “Architecture principles” on page 157. The functions, features and integration considerations of these building blocks are explained in the following section.

4.5.1 Third-party authentication using Netegrity’s SiteMinder

Netegrity’s SiteMinder is a software platform of shared services that includes single sign-on, authentication management, and entitlement management (also referred to as access control) to e-business Web sites. SiteMinder provides centralized authentication and access control, and leverages these services across all users and applications on an enterprise’s Web sites. SiteMinder consists of the following components:

- SiteMinder Policy Server

  The Policy Server is the heart of the SiteMinder product and provides the following key security operations:

  - Authentication services
    - Basic Authentication (user-name/password)
    - Basic Authentication over SSL

- Authentication schemes including RADIUS proxy, CryptoCard, ACE/Server from Security Dynamics.
- Forms-based authentication
- Digital (X.509) certificates
- Combination of passwords and certificates
- Custom or third-party schemes

– Authorization services

The resources that SiteMinder can protect are any named URL that a Web application or user needs to access. The Policy Server is responsible for managing and enforcing the access control rules established by the administrator.

– Auditing services

- SiteMinder agents

  A SiteMinder agent communicates with the Policy Server to enforce policies for user access to generic resources.

  - Web agents

    Web agents enforce access control to Web content and deliver a user's security context, managed by SiteMinder, directly to any Web application being accessed by the user. The Web agent is integrated through each Web server's extension API. It intercepts all requests for resources (URLs), and determines whether each resource is protected by SiteMinder. If not, the request is passed through to the Web server for regular processing. The Web agent interacts with the Policy Server to authenticate the user, and to determine if access to the specific resource should be allowed. If the user is successfully authenticated by the Policy Server, a strongly encrypted cookie is created and stored in the user's browser. This cookie does not contain any sensitive information such as a password. Instead, it contains the user's full directory name, and a number of timestamps and other information. This cookie can be used later to allow single sign-on across all Web applications in the domain.

  - Application server agents

    Netegrity provides a set of APIs, SiteMinder EJB precompiler and SiteMinder EJB complier to "SiteMinder-enable" fine-grained access control for J2EE components including servlets, JSPs and EJBs.

  - Custom agents

    SiteMinder and its Web agent are used to protect resources on Web sites. The SiteMinder Policy Server with its general-purpose rules engine can protect access to any resource that can be expressed as a string, and operations on those resources.

  - Affiliate agents

    SiteMinder's affiliate agent resides on the Web server of an affiliate site, and provides user profile and entitlement information to applications running on an affiliate site.

SiteMinder supports a range of user directories including leading LDAP directories, and relational database:

- Directory servers
  - IBM SecureWay Directory
  - Netscape/iPlanet Directory Server
  - Oracle Directory Server
  - Microsoft Active Directory
  - Novell NDS/Edirectory
- SQL Database
  - Microsoft SQL Server
  - Oracle
- Windows NT domains

Refer to http://www.netegrity.com for more details on Netegrity's offering.

Integration considerations

Enable Offering's integration with SiteMinder uses WebSphere Application Server V3.5's Reverse Proxy Security Server and trusted association architecture described in "Security architecture" on page 163. The integration between Enable Offering and SiteMinder is discussed in "Enable Offering integration with SiteMinder authentication" on page 182. SiteMinder's Web Agent and Policy Server will perform the user authentication. Using TAI, WebSphere Application Server V3.5 will trust this authentication as if this user is authenticated by WebSphere Application Server.

Both SiteMinder and WebSphere Application Server V3.5 access control mechanisms will be invoked under this scenario. The user can configure whether SiteMinder or WebSphere Application Server controls the access to individual resources. For example, if the access to a resource is to be controlled by SiteMinder, then configure its access control using SiteMinder, and allow all accesses to this resource using WebSphere Application Server V3.5 security functions. WebSphere Application Server V3.5 security must be enabled to support third-party authentication, regardless of whether WebSphere Application Server V3.5 or SiteMinder controls accesses to WebSphere Application Server V3.5 resources. Likewise, if the access to a resource is to be controlled by WebSphere Application Server V3.5, then configure to permit all accesses to this resource using SiteMinder, and allow configure access control to this resource using WebSphere Application Server V3.5 security functions. It is recommended that SiteMinder provide coarse-grain access control, and WebSphere Application Server to provide fine-grain access control, for example EJB.

In our QuickClaim scenario, access control to WebSphere applications, including Quote, Risk Assessment and customer Information, were controlled by WebSphere. QuickClaim can continue to operate this way, and use SiteMinder simply for authentication only, to minimize changes. However, this architecture also allows migration of access control from WebSphere Application Server to SiteMinder's centralized security system to support QuickClaim's long-term architectural goal stated in "Architecture principles" on page 157.

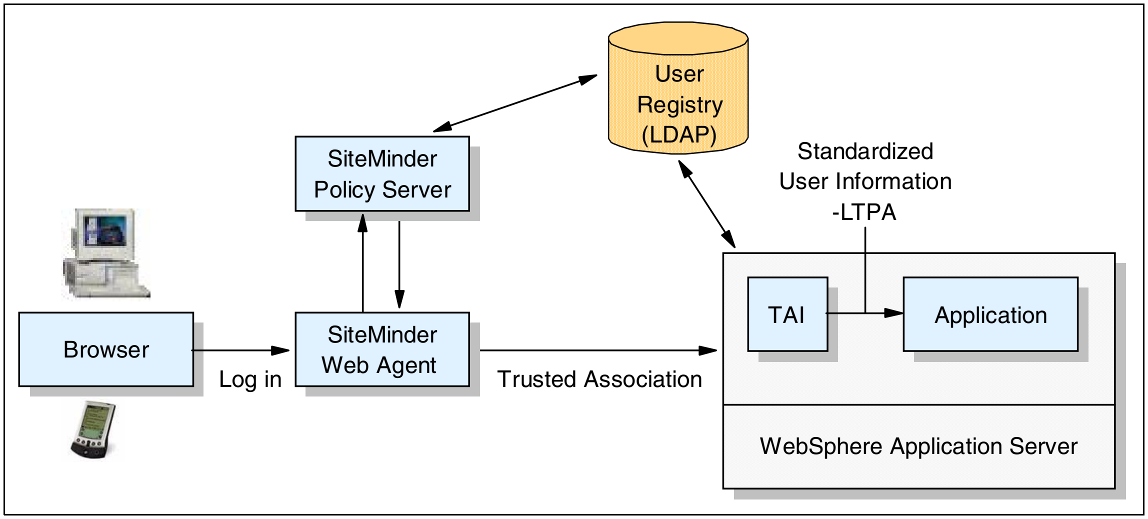


Figure 4-14 Enable Offering integration with SiteMinder authentication

4.5.2 DB2 Everyplace for devices

The DB2 Everyplace (DB2e) database is specifically designed for the mobile and embedded application environment. The DB2 Everyplace database engine supports basic SQL and such relational operations as join, group by, order by, multiple column primary and foreign key, expression functions, aggregate functions, and constraints. Advanced indexing provides a noticeable query performance boots for medium to large DB2 Everyplace tables. Scrollable cursors enable easier application data manipulation. To access data on DB2e, one can write applications using the supported set of DB2 Call Level Interface (CLI) functions, Open Database Connectivity (ODBC) or Java Database Connectivity (JDBC) methods.

The DB2 Everyplace solution has the following two components:

- The DB2 Everyplace database on the mobile device.
- The DB2 Everyplace Sync Server, a synchronization service running on a mid-tier server that synchronizes data between the mobile device and enterprise data sources. DB2 Everyplace Sync Server is included with DB2 Everyplace Enterprise Edition.

The latest available version of DB2 Everyplace solution is V7.2. DB2 Everyplace runs on:

- Palm OS devices
- Symbian EPOC devices
- QNX Neutrino devices
- Embedded Linux devices
- Windows CE and Pocket PC devices
- Windows platform

Integration considerations

DB2e's compact relational database enables QuickClaim to rapidly deploy their customer database and policy applications to the handheld devices. This also provides important offline application capabilities to Joe and greatly improves his productivity. Query-By-Example is provided with the DB2 Everyplace database on the Palm client. A Palm user can use the Query-By-Example (QBE) function to update customer information, insert new customer data, or delete customer data from the DB2e customer table. The more advanced user can use the command-line-processor (CLP) to enter and execute SQL statements. DB2e implements a subset of the SQL 99 standard.

QuickClaim's application developers can use QBE to create tables on the devices and test the mobile DB2e database and policy application functionality. QBE makes viewing the databases on the device easier during development. Changes made to the mobile DB2e database are reflected in QuickClaim's enterprise DB2 database only after the data synchronization using either IBM Mobile Connect V2.5 or the DB2 Everyplace Sync Server.

To reduce the total number of components required, thereby reducing the overall architecture complexity, IBM Mobile Connect V2.5 is selected to provide both e-mail and database synchronization functions. Alternatively, DB2 Sync Server can be added to this architecture for DB2e synchronization with DB2. DB2 Sync Server is an attractive alternative, when the solution requires only database synchronization and does not require e-mail or PIM synchronizations. Refer to “IBM Mobile Connect V2.5 for PIM and DB2e synchronization” on page 184 for more details on how to use IBM Mobil Connect for database synchronization.

4.5.3 IBM Mobile Connect V2.5 for PIM and DB2e synchronization

IBM Mobile Connect (IMC) V2.5 provides server-based synchronization of e-mail, contacts, calendar, and tasks between mobile devices (including Palm computing, Windows CE, Pocket PC and EPOC operating systems), and the leading groupware products (including Microsoft Exchange and Lotus Domino servers).

IBM Mobile Connect also supports synchronization of custom mobile device applications with any ODBC-compliant database including DB2, Oracle, Sybase, and MicroSoft Access. Refer to Figure 4-14 on page 182 for an overview of IBM Mobil Connect synchronization architecture.

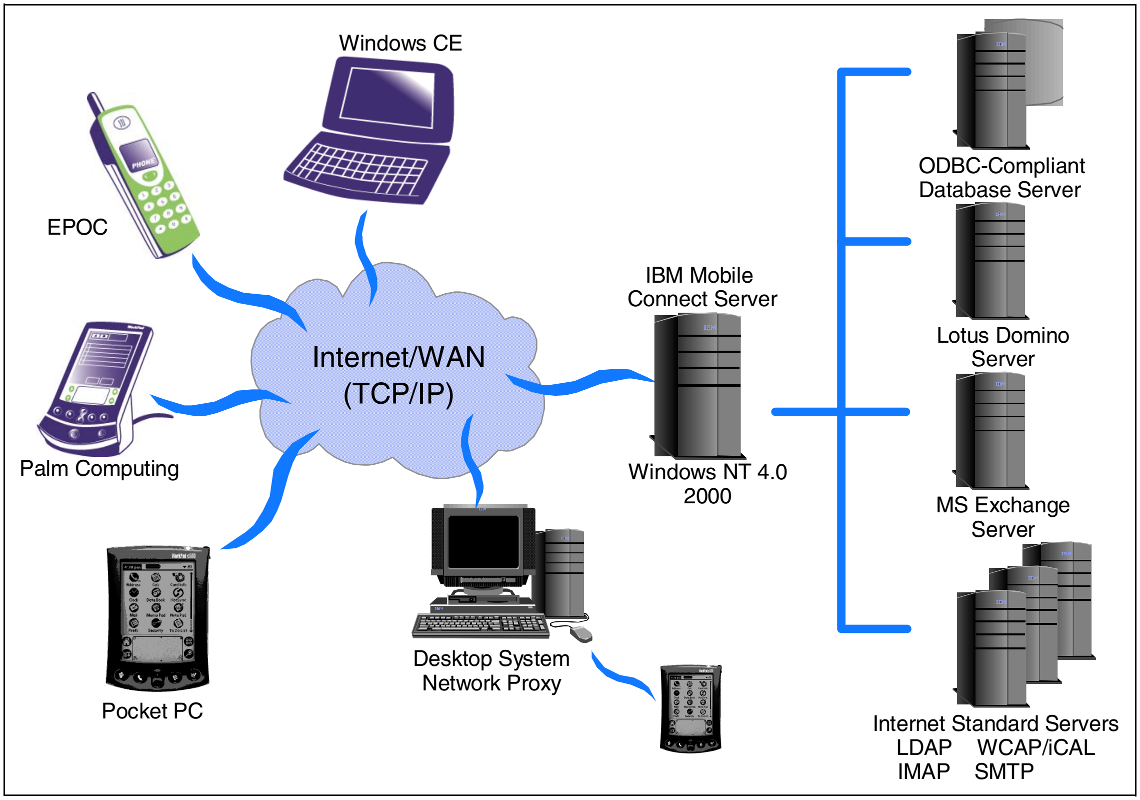


Figure 4-15 IBM Mobil Connect architecture overview

IBM Mobil Connect provides the network proxy facility that allows a mobile device to connect to a networked PC via a serial link for connecting to the distant server through networked PCs. This enables the mobile device to use a networked PC as a network proxy to access a server on the network, thus avoiding the need to configure mobile devices and remote access servers. IBM Mobil Connect network proxy performs the synchronization between the IBM Mobil Connect Client running on the mobile device and the IBM Mobil Connect Server.

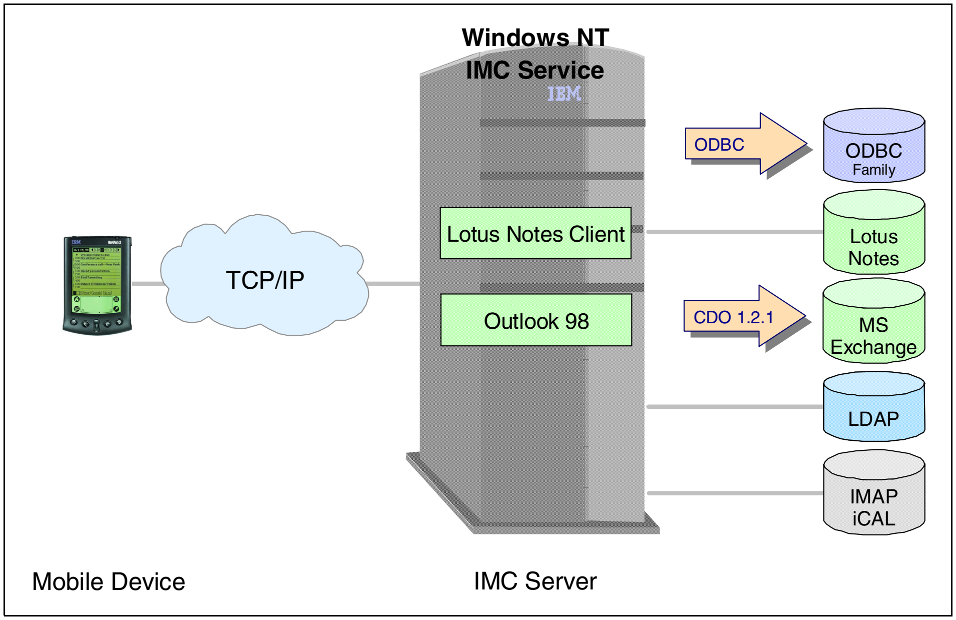


Figure 4-16 IBM Mobil Connect server configuration overview

IBM Mobil Connect can authenticate users against the Lotus Domino Public Address Book using the following two methods:

- “Super ID” method

  IBM Mobil Connect can use a “Super ID” to allow all authentication and synchronization. To successfully authenticate users and access their mail databases, IBM Mobil Connect must have the Lotus Domino rights to do so. This can be accomplished by creating a Lotus Notes super ID account for IBM Mobil Connect, and using this account to manage the IBM Mobil Connect synchronizations. This Super ID file should be saved in the IBM Mobil Connect server. The Super ID should have “read” access to the Lotus Domino public address book containing the HTTP password and “manager” access to all the mail databases of the users that wish to synchronize using IBM Mobil Connect. Individual users will use their HTTP passwords to synchronize their Notes mail.

- "Individual ID" method

  IBM Mobil Connect can use individual user Notes ID for authentication and synchronization. These ID files must reside on the IBM Mobil Connect server or on a mapped drive accessible from the IBM Mobil Connect server. Users will authenticate with the IBM Mobil Connect server using their Notes ID password.

Table 4-5 IBM Mobil Connect authentication methods

Authentication Method	Pros	Cons
Super ID	Little administration overhead	Does not support encrypted e-mails or documents
Individual ID	Enforces Lotus recommended security model Supports encrypted documents	Potentially additional administration if working with copies of ID files

IBM Mobil Connect can be set up to synchronize:

- E-mail
- Calendar
- To Do List
- Address Book
- Custom database
- Journal to the Palm's memo database

ODBC database synchronization

IBM Mobile Connect provides synchronization from the following mobile databases to any ODBC-compliant database:

- Palm OS
  - DB2 Everyplace Version 7.1.2
  - Satellite Forms
- WinCE
  - DB2 Everyplace Version 7.1.2
  - ADOCE (including PocketAccess CDB)
  - CEDB (CE ObjectStore)
- EPOC - native database application

IBM Mobil Connect provides database synchronization using various mobile database plug-ins and provides record-level (not field-level) synchronization. For example, IBM Mobil Connect has Palm OS plug-ins for DB2e, Satellite Forms, and a generic plug-in to synchronize data that is stored with fixed-length fields followed by null-terminated strings. Code Warrior and AppForge applications both can use this generic plug-in as long as the schema is defined accordingly.

IBM Mobil Connect has WinCE plug-ins for DB2e, and ADOCE Plug-in. In order to map data from an ODBC database to a DB2e mobile database, the table schema for the remote database must first be defined in IBM Mobil Connect. Once the database schema is created, the server-side ODBC database columns can then be mapped to the DB2e table. Palm DB2e plug-in works the same way as the WinCE DB2 plug-in. Device-side plug-ins are needed for ADOCE and DB2e, and they can be pushed to the mobile devices using IBM Mobil Connect's replication wizard.

In general, IBM Mobil Connect is concerned with synchronizing the data and requires the mobile application to retain the integrity of the primary key. IBM Mobil Connect will create a mobile database, but will not attempt to create an indexing file for Palm generic databases (.PDB flat file) or WinCE generic databases (Object Store). For ADOCE and DB2e, however, IBM Mobil Connect honors the indexing when creating the mobile databases. While making columns part of the primary key, IBM Mobil Connect would make it part of an automatically generated index. DB2e provides the ability to create indexes on other columns using the Call Level Processor.

IBM Mobil Connect integration considerations

In this scenario, a WinCE DB2e plug-in is needed to replicate the DB2 with DB2e. IBM Mobil Connect's authentication and access control for Notes mail, PIM applications are different from and not integrated with Enable Offering third-party authentication. Therefore, IBM Mobil Connect's authentication and access control are not integrated with SiteMinder out of the box. However, QuickClaim may use tools provided by SiteMinder to integrate these applications with SiteMinder's centralized security system. Because the Notes Mail and PIM applications are existing applications, QuickClaim's architecture principles and requirements allow exceptions.

IBM Mobil Connect V2.5 can be set up to authenticate mobile users against an LDAP directory that contains user names and passwords. It allows a distinguished name (DN) or fully qualified name through configuration. Therefore, IBM Mobil Connect V 2.5 can be configured to use QuickClaim's user directory for authentication in database synchronization. Using the LDAP synchronization plug-in, IBM Mobil Connect V2.5 can also synchronize the LDAP directory to and from databases (for example, contact/address) on mobile devices.

4.5.4 MQ Everyplace V1.2 for devices

MQSeries Everyplace is a member of the MQSeries family of messaging products:

- Distributed messaging: MQSeries for Windows NT, AIX, AS/400, HP-UX, Sun Solaris, and other distributed platforms.
- Host messaging: MQSeries for OS/390
- Pervasive messaging: MQSeries Everyplace.

It is designed to satisfy the transactional messaging needs of handheld devices, such as Personal Digital Assistants (PDAs) and laptop computers, as well as supporting mobility and the requirements that arise from the use of fragile communication networks.

MQSeries host and distributed products

MQSeries host and distributed messaging products support different clients and server configurations. In the simplest case, a stand-alone server is configured, running a queue manager. One or more applications run on that server and exchange messages via queues. An alternative configuration is client/server. Here the queue manager only exists on the server, but the clients each have access to it through a client channel. The client channel is a bidirectional communications link that flows a unique MQSeries protocol implementing something similar to a remote procedure call (RPC). Applications can run on the clients, accessing server queues. One advantage of the client/server configuration is that the client-messaging infrastructure is lightweight, being dependent on the server queue manager. A disadvantage is that clients and their associated server operate synchronously and therefore require the client channel to be always available.

MQSeries Everyplace overview

MQSeries Everyplace supports a wide variety of network configurations through the use of queue managers - with each queue manager configured with appropriate capabilities. MQSeries Everyplace queue managers can act as traditional MQSeries clients or servers but each is in fact simply a queue manager enabled to perform application-defined tasks. An MQSeries Everyplace queue manager can be configured with or without local queues. With local queues it can store messages locally and hence offer applications asynchronous messaging, but without local queues it is restricted to synchronous messaging. An MQSeries Everyplace queue manager can be configured with or without bridge capabilities. With the bridge, it can exchange messages with MQSeries host or workstation queue managers; without the bridge, it can only communicate directly with other MQSeries Everyplace queue managers, although it can communicate indirectly by exploiting other queue managers in the network that do have the bridge capabilities.

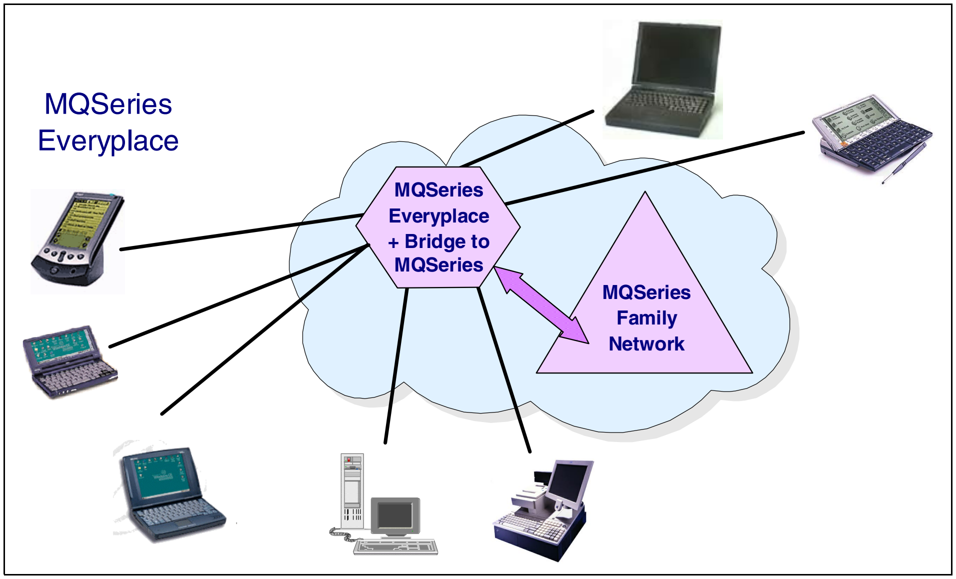


Figure 4-17   MQSeries Everyplace V1.2 configurations

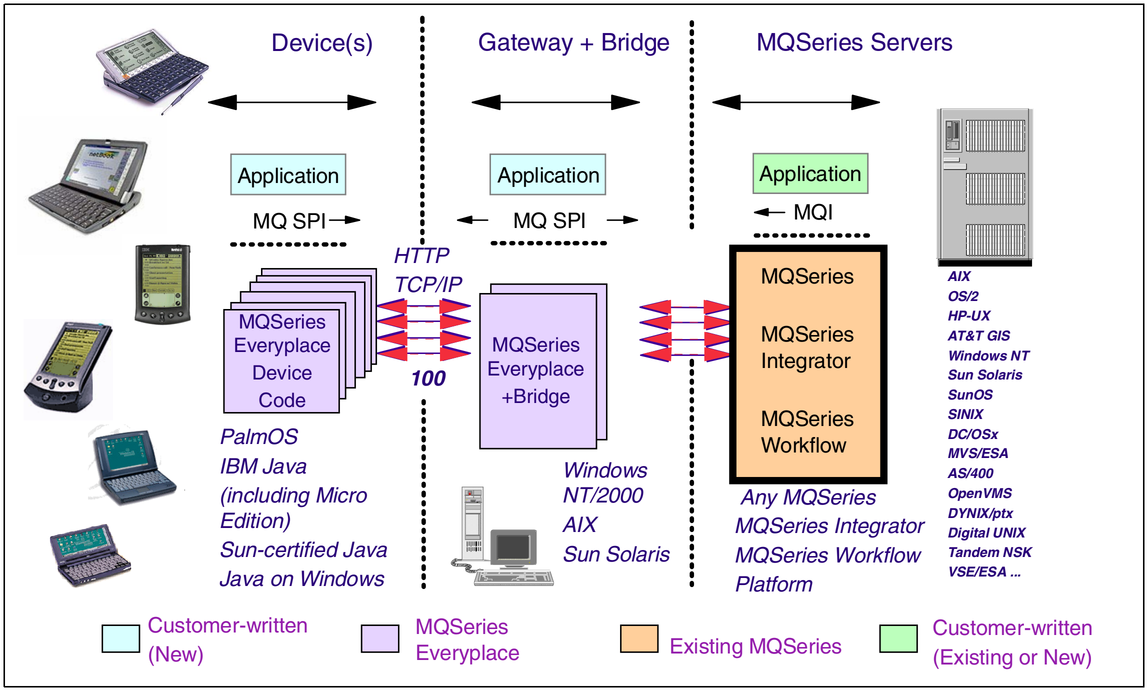


Figure 4-18 MQSeries Everyplace scenario example

The MQSeries Everyplace queue manager is the focal point of the MQSeries Everyplace system. It provides:

- A central point of access to a messaging and queueing network for MQSeries Everyplace applications
- Optional client-side queuing
- Optional administration functions
- Once-only guaranteed delivery of messages
- Full recovery from failure conditions
- Extendable rule-based behavior

The fundamental elements of the MQSeries Everyplace programming model are messages, queues and queue managers. MQSeries Everyplace messages are objects that contain application-defined content. When stored, they are held in a queue and such messages may be moved across an MQSeries Everyplace network. Messages are addressed to a target queue by specifying the target queue manager and queue name pair. Applications place messages on queues through a put operation and typically retrieve them through a get operation. Queues can either be local or remote and are managed by queue managers. Configuration data is stored in a registry. MQSeries Everyplace supports a number of different queue types:

- Local queues

  Local queues are used by applications to store messages in a safe and secure manner. Local queues can be used in either online or offline.

- Remote queues

  Remote queues are local references to queues that reside on a remote queue manager. They have properties concerned with access, such as the mode of access (synchronous or asynchronous), security characteristics and transmission options.

- Asynchronous remote queues

  Asynchronous remote queues are queues that send messages to remote queues but cannot remotely retrieve messages. If the network connection is established then the messages are sent to the owning queue manager and queue. If the network is not connected, the messages are stored locally until there is a network connection and then the messages are transmitted.

- Store-and-forward queues

  Store-and-forward queues are associated with a set of queue manager names for which they will hold messages. It has two main uses. The first is to enable the intermediate storage of messages in a network, such that they can proceed stepwise to their destination. The second use is to hold messages awaiting collection (refer to home-server queues).

- Home-server queues

  Remote queues and store-and-forward queues push messages across the network, the sending queues initiating the transmission. Home-server queues, however, allow messages to be pulled from a remote queue. A home-server queue definition identifies a store-and-forward queue on a remote queue manager. Home-server queues usually reside on a device and are set up to pull messages from a server whenever the device connects to the network.

- Administration queues

  Administration queues are the mechanism through which queue managers are configured, either locally or remotely.

- MQSeries Bridge queues

  This is a specialized form of remote queue with the definition on a gateway (MQe server) and the target queue on an MQSeries queue manager. This form of queue provides a pathway between the MQSeries Everyplace and the MQSeries environments.

Figure 4-19 on page 194 depicts an MQSeries network with MQSeries Everyplace clients, gateway, bridge and MQ servers. An MQSeries Everyplace queue manager can be an interface to an MQSeries server. This type of queue manager is referred to as a gateway queue manager. The MQSeries bridge handles the transfer of messages between the two systems, including the translation between the different message formats.

MQSeries Everyplace stores data securely on queues, ensuring that messages are physically written to the media and not simply buffered by the operating system. However, MQSeries Everyplace does not independently log changes to messages and queues. If recovery from media failure is required, then hardware solutions must be deployed, such as the use of RAID disk systems. Alternatively the queue may be mapped into recoverable storage such as recoverable database subsystems.

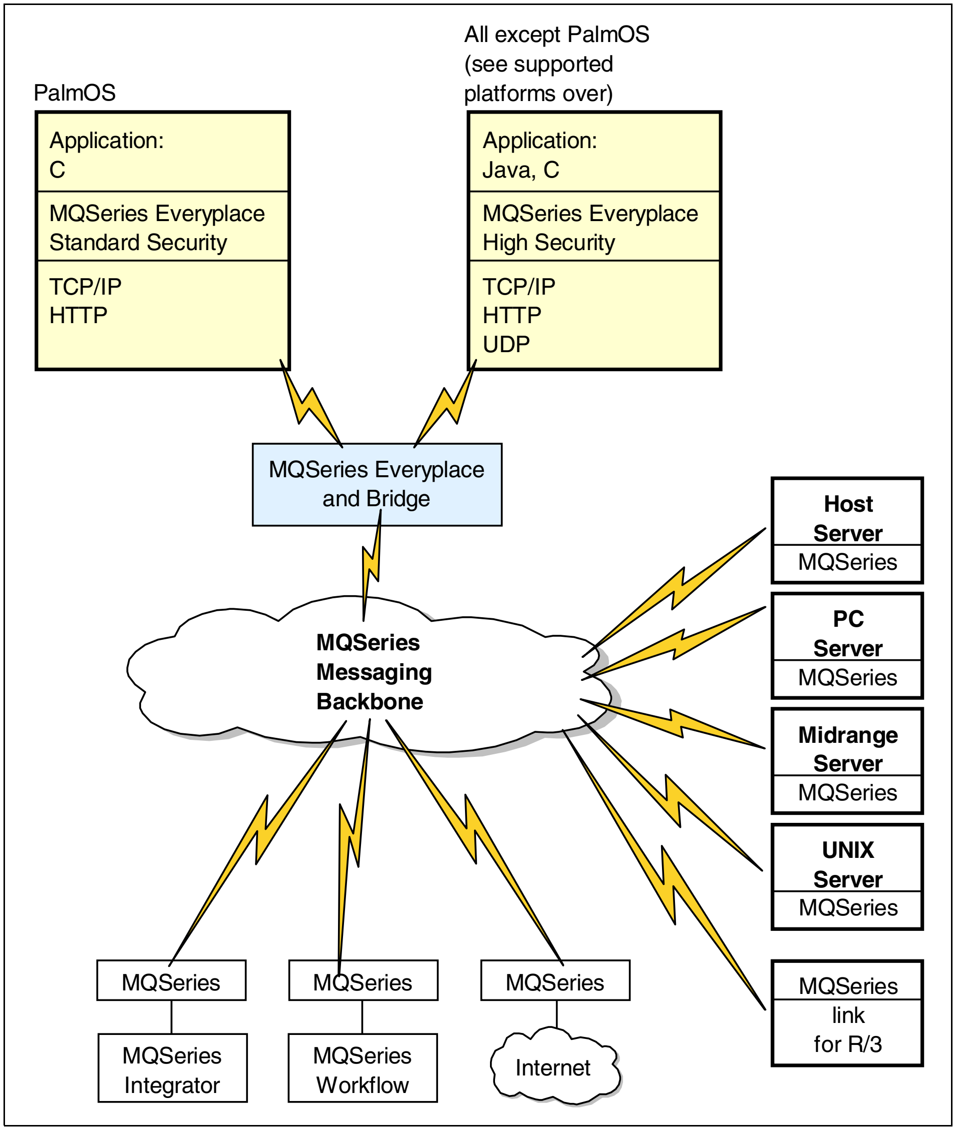


Figure 4-19   MQSeries Everyplace and MQSeries network

In addition, MQSeries Everyplace on pervasive devices can interoperate with an MQSeries Everyplace Java server. It uses HTTP 1.0 protocol to communicate with the server. The use of the HTTP protocol enables MQSeries Everyplace messages to pass through standard firewalls without any need to modify the firewalls.

MQSeries Everyplace security

MQSeries Everyplace includes an integrated set of security features that provide protection for message data, when it is held locally, and when it is being transferred. There are three different categories of security:

- Local security

  Local security provides protection for MQSeries Everyplace messages while they are held by a local queue manager.

- Queue-based security

  Queue-based security automatically protects MQSeries Everyplace message data between an initiating queue manager and a target queue, so long as the target queue is defined with an attribute.

- Message-level security

  Message-level security provides protection for message data between an initiating and receiving MQSeries Everyplace application.

MQSeries Everyplace security uses the authentication, cryptor, and compressor attributes defined in the MQSeries Everyplace channels:

- Authenticator

  This attribute causes authentication to be performed. This is a security function that challenges the message-putting application environment or user to prove their identity.

- Cryptor

  This attribute causes encryption and decryption to be performed on messages passing through the channel. This is a security function that encodes the messages during transit so that they cannot be read without the decoding function.

- Compressor

  This attribute causes compression and decompression to be performed on messages passing through the channel. This attempts to reduce the size of messages while they are being transmitted and stored.

Typically the authenticator is only used when setting up the channel; compressors and cryptors are normally used on all flows.

MQSeries Everyplace supported platforms

MQSeries Everyplace is directly installable only on certain server platforms. Other platforms require transferring programs and Java classes.

Directly supported platforms with installation support are:

- Windows NT 4.0
- Windows 2000
- Windows 95/98/ME
- AIX V3
- Sun Solaris V 7 or 8
- Linux Intel Kernel 2.2
- HP-UX 11.0

Directly supported platforms without installation support:

- WinCE 2.1 running on HP Jornada devices (Models 680 or 820)
- EPOC 32 bit Release 5 running on Psion devices (5MX pro or NetBook)
- Palm OS V3.0 or higher running on Palm V and IBM WorkPad C3
- IBM 4690 OS with Java

Indirectly supported platforms:

- Linux on zSeries running Kernel 2.2
- Series
- OS/2
- EPOC (on devices other than those listed above)
- WinCE (on devices other than those listed above)
- QNX Neutrino
- Pocket PC
- Palm OS (on devices other than those listed above)
- Any other platform running one of the Java environments:
  - IBM JVM 1.3 or later, including Java Micro Edition
  - Any Sun Java (V1.1 or later) certified.
  - Java needs to be fully compatible with that tested on one of the following platforms for support
    - HP Jornada devices (Model 680 or 820) running Windows CE
    - Psion devices (5MX Pro or NetBook) running EPOC
    - One of the server platforms in the directly supported list

MQe integration considerations

MQe V1.2 is bundled with Everyplace Server Enable Offering V1.1, and the Everyplace Server Enable Offering installation supports the installation of MQe V1.2. Figure 4-20 on page 197 depicts the components required on the mobile device and Everyplace Server to support synchronization and assured messaging:

- Everyplace Wireless Gateway client on WinCE device connects to Everyplace Wireless Gateway. Everyplace Wireless Gateway client is required only if Everyplace Wireless Gateway is used as the wireless gateway (optional).
- MQe client on WinCE device connects to the MQe server (gateway), using the MQe gateway and MQe bridge to connect to the payment application running on an MQSeries server.
- DB2e client on WinCE device connects to a DB2 server via IBM Mobile Connect V2.5
- Device agent on WinCE device connects to Device Manager Server

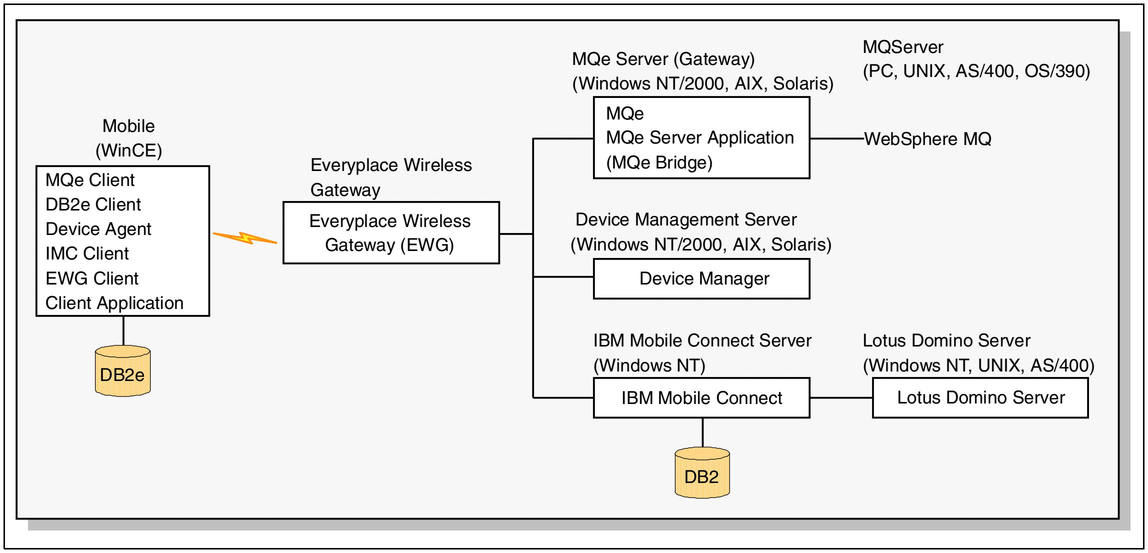


Figure 4-20 IBM Mobil Connect and MQe client/server architecture

The WinCE device (Java client) supports queue manager. Therefore, QuickClaim can design both asynchronous and synchronous assured messaging applications. The asynchronous messaging applications can provide offline functions and store messages on the local queue.

QuickClaim's payment application uses MQ security features to interface with PC clients and financial institutions, and will continue to use MQ security features to interface with the mobile device MQe clients. Therefore, there is no security integration with the SiteMinder central security system. Because Payment application is an existing application for external interfaces, it is exempted from QuickClaim's centrally managed security compliance requirement.

4.5.5 Device management and software distribution

The Everyplace Server Enable Offering provides the following core device management services:

- Enrolling devices for users
- Distributing software to the device
- Updating device configuration remotely
- Updating rest pages (startup pages) for devices

The Enable Offering V1.1 Device Manager Server is equivalent to Tivoli Personalized Services Manager Device Manager Server V1.1. The Device Manager Server keeps all the device management information in its own database. This database houses the following information:

- Device-related information
  - Devices
  - Device classes
  - Device parameters
- Software information
  - URL of the software package to be distributed
  - Version of the software
- Job-related information
  - Job classes
  - Submitted jobs

The device management database can be installed on one of the following databases:

- DB2 UDB 7.1 (local or remote)
- Oracle8i Version 8.1.5 or 8.1.7

Device Manager Server (DMS) is a WebSphere application. With Enable Offering V1.1.2, it runs on AIX, Solaris and Windows NT or 2000. The Device Manager Server has the following components as depicted in Figure 4-21 on page 199:

- Device Manager Server servlet

  When a device connects to WebSphere Everyplace Server Enable Offering, the DMS servlet ensures that the device is enrolled. If it is, DMS coordinates the processing of all scheduled device management jobs for that device.

- Device plug-ins

  Device plug-ins provide the logic that handles device identification, communications, job processing, and high-level management tasks for device configuration, software distribution and rest page management.

- Device enrollment servlet
- Device Management API

  Device management API provides the programming interface for managing devices, jobs, and related resources in the Device Manager Server database.

- Device Manager Server API

  Device Manager Server API defines the programming interface between Device Manager Server servlet and plug-ins.

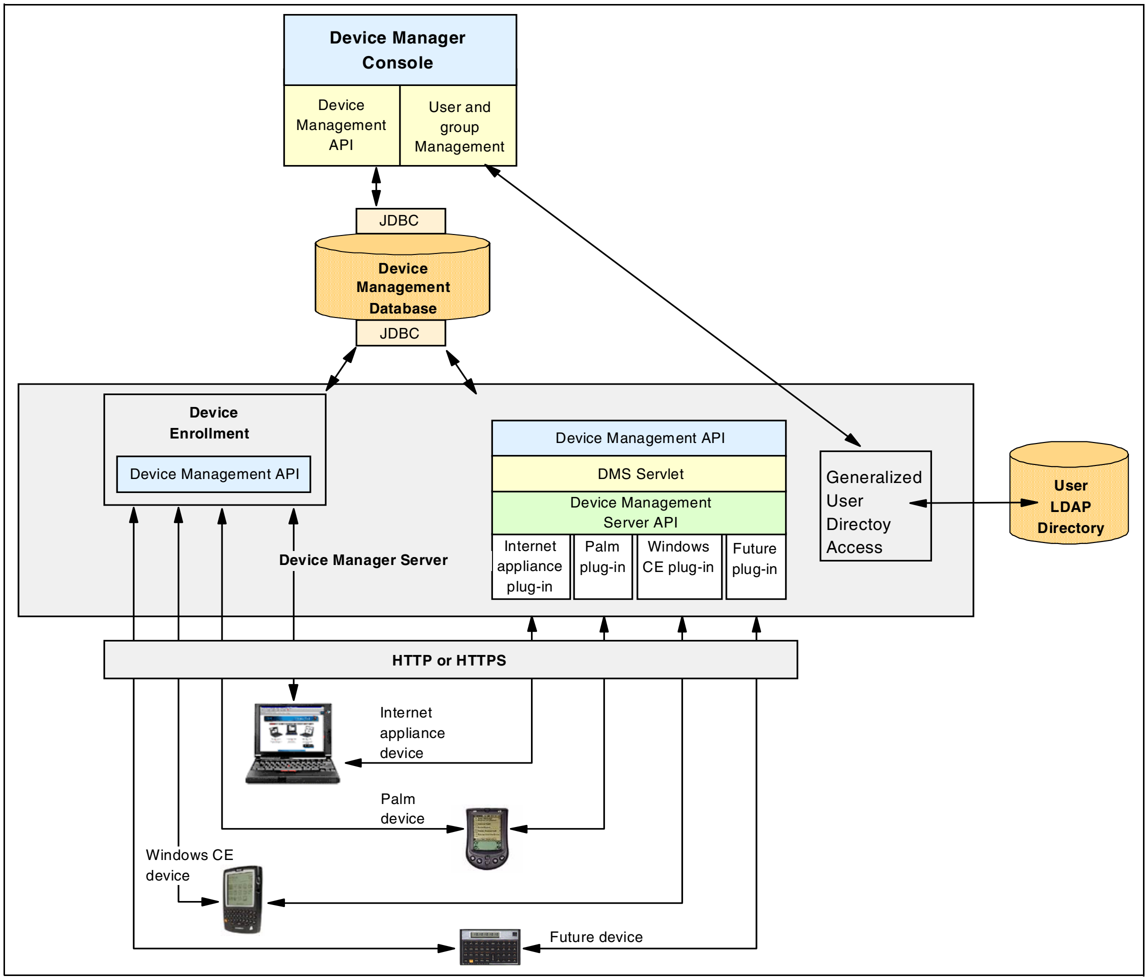


Figure 4-21 Device Management overview

Associate user directory with Device Management

The Device Manager Server can be configured to use a user directory (LDAP). The user directory access and authenticating is registered in the WebSphere Application Server and will allow access to the specified user directory (LDAP) server. The Device Manager Server will validate the user and realm at device enrollment (realm is a name assigned to the user directory name space); therefore, the user must be defined in the user directory.

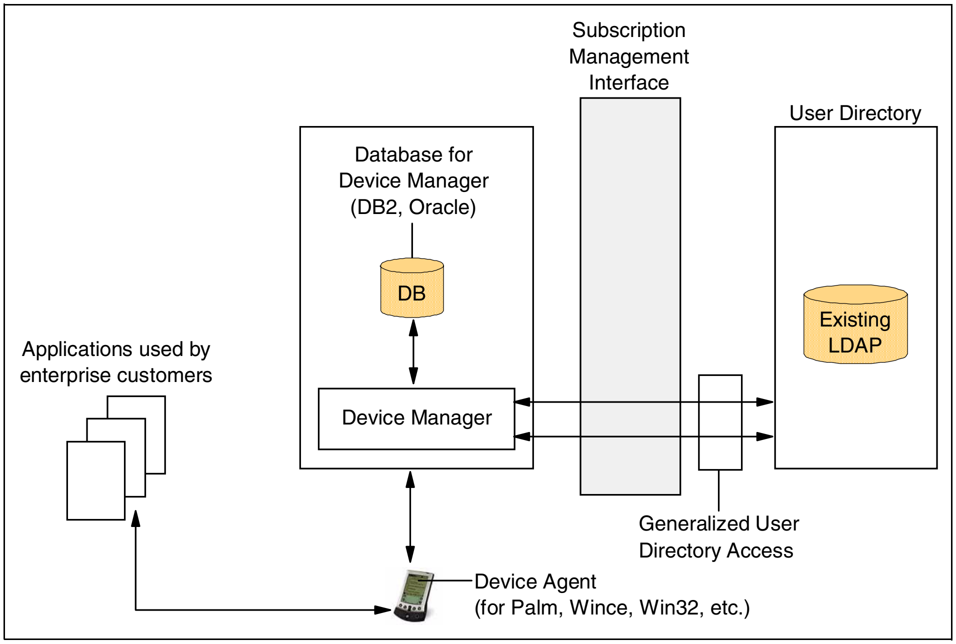

Subscription Management Interface
User Directory
Database for Device Manager (DB2, Oracle)
DB
Existing LDAP
Applications used by enterprise customers
Device Manager
Generalized User Directory Access
Device Agent (for Palm, Wince, Win32, etc.)

Figure 4-22 Associate user directory with Device Management

To associate the user directory with Device Manager Server, the user directory must contain an attribute that uniquely identifies a user. The name of this attribute should be specified during the Device Manager Server installation. The default attribute is `uid`. If groups are defined in the user directory, the Device Manager Server enables you to submit jobs to all the devices owned by the users defined in a single group. The association and disassociation of user directory with the Device Manager Server can be changed after product installation. Refer to *WebSphere Everyplace Server Enable Offering for Multiplatforms Getting Started Version V1.1* for more details.

Device Manager Server may work with a third-party authentication tool, if the third-party authentication tool uses basic authentication. If this is the case, Device Manager Server, which is implemented as a WebSphere application, will take advantage of the TAI function to support third-party authentication as depicted in Figure 4-23. Otherwise, the Device Manager Server device agent does not support other forms of authentication.

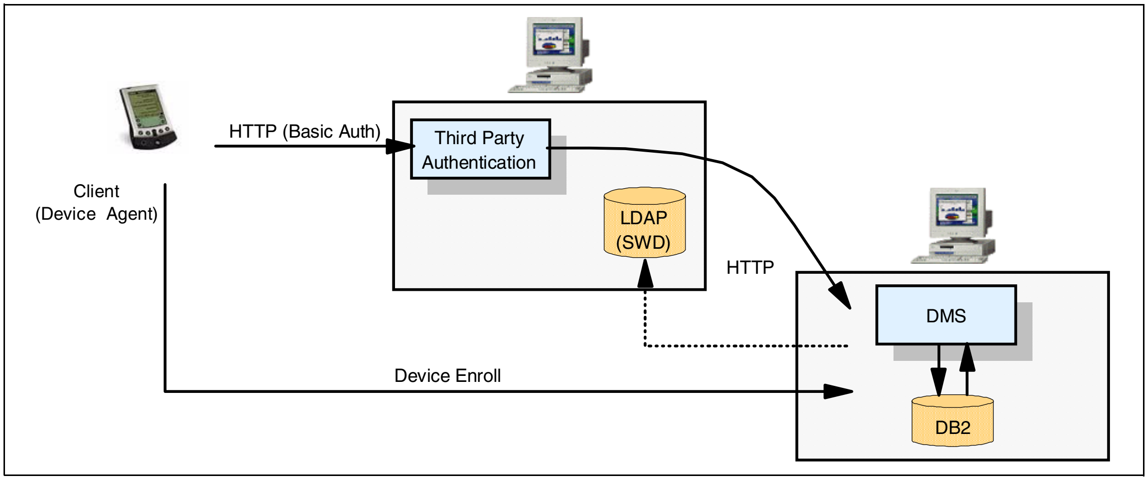
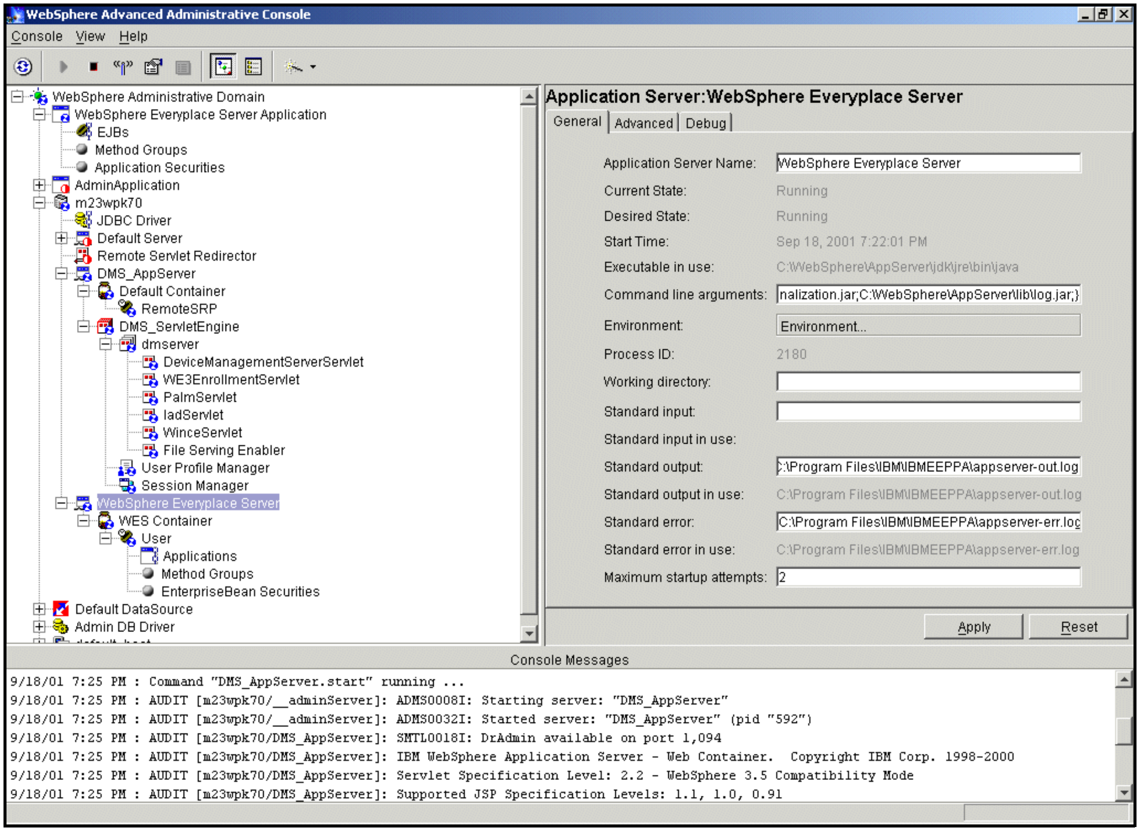


Figure 4-23 Device Agent authentication flow for basic authentication

Figure 4-24 on page 202 shows how Device Manager Server integrates with WebSphere Application Server V3.5 and GUDA. The DMS_AppServer is the Device Manager Server installed by Enable Offering and configured as an WebSphere application server. The Device Manager Server servlet and device enrollment servlet are the corresponding DeviceManagementServerServlet and WE3EnrollmentServlet running in the DMS_AppServer application server. PalmServlet, ladServlet,and WinceServlet are device plug-ins.



Application Server:WebSphere Everyplace Server

Application Server Name: WebSphere Everyplace Server

Current State: Running

Desired State: Running

Start Time: Sep 18, 2001 7:22:01 PM

Executable in use: C:\WebSphere\AppServer\jdk\jre\bin\java

Command line arguments: nalization.jar;C:\WebSphere\AppServer\lib\log.jar;}

Environment: Environment...

Process ID: 2180

Working directory:

Standard input:

Standard input in use:

Standard output: :\Program Files\IBM\IBMEEPPA\appserver-out.log

Standard output in use: C:\Program Files\IBM\IBMEEPPA\appserver-out.log

Standard error: C:\Program Files\IBM\IBMEEPPA\appserver-err.log

Standard error in use: C:\Program Files\IBM\IBMEEPPA\appserver-err.log

Maximum startup attempts: 2

```
9/18/01 7:25 PM : Command "DMS_AppServer.start" running ...
9/18/01 7:25 PM : AUDIT [m23wpk70/__adminServer]: ADMS0008I: Starting server: "DMS_AppServer"
9/18/01 7:25 PM : AUDIT [m23wpk70/__adminServer]: ADMS0032I: Started server: "DMS_AppServer" (pid "592")
9/18/01 7:25 PM : AUDIT [m23wpk70/DMS_AppServer]: SMTL0018I: DrAdmin available on port 1,094
9/18/01 7:25 PM : AUDIT [m23wpk70/DMS_AppServer]: IBM WebSphere Application Server - Web Container.  Copyright IBM Corp. 1998-2000
9/18/01 7:25 PM : AUDIT [m23wpk70/DMS_AppServer]: Servlet Specification Level: 2.2 - WebSphere 3.5 Compatibility Mode
9/18/01 7:25 PM : AUDIT [m23wpk70/DMS_AppServer]: Supported JSP Specification Levels: 1.1, 1.0, 0.91
```

Figure 4-24 Device Manager Server running in WebSphere

Figure 4-25 on page 203 depicts Enable Offering's interface with user directory via Generalized User Directory Access (GUDA). The Generalized User Directory Access (GUDA) was installed by Enable Offering and configured as a WebSphere application server named WebSphere Everyplace Server as shown in Figure 4-24. User EJB is deployed within the WebSphere Everyplace Server's EJB container named WES container.

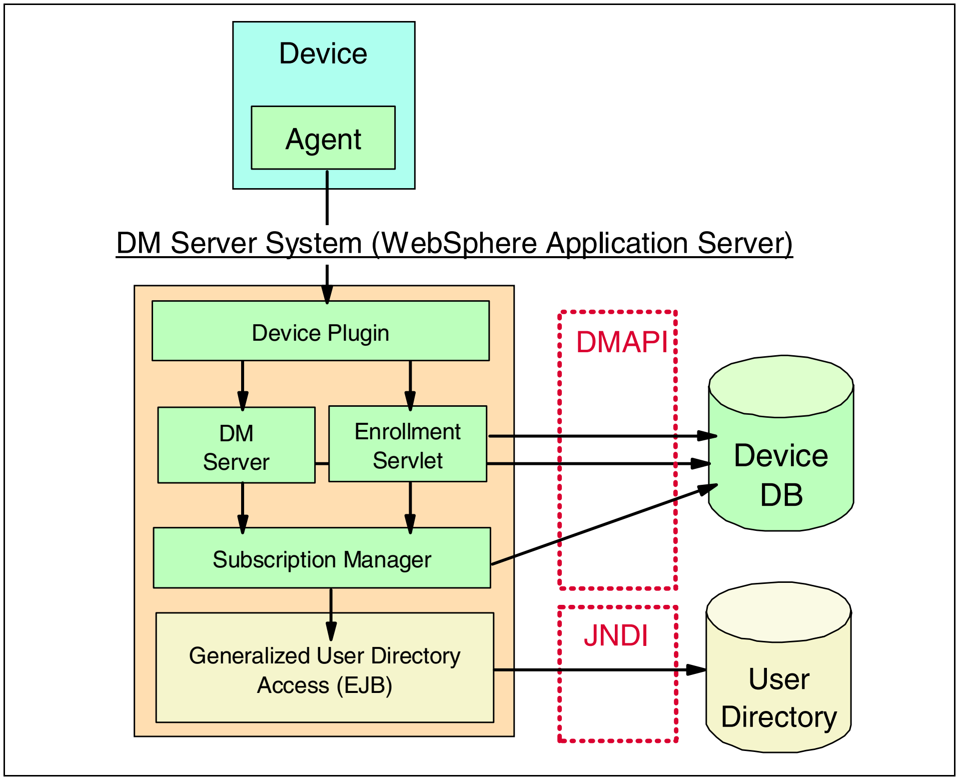


Figure 4-25 Everyplace Server Enable Offering Device Manager Server

Device Manager Server can support various types of devices by using device type-specific plug-ins. It is device plug-ins that actually perform device-specific management functions. For example, Device Manager Server's built-in device plug-ins provide management functions such as software distribution, device configuration, and rest page management.

A device agent to be installed on the device knows how to communicate with the associated Device Manager Server plug-in. Business administrators and systems administrators use the Device Manager Server console to perform management tasks. They can submit jobs, control device parameter settings, and manage software to be distributed. This console accesses and updates the Device Management database through the Device Management API (DMAPI). The Device Manager Server administration console runs on Windows system. The console and device agent can be downloaded from the Device Manager Server. The device console is used to schedule software distribution jobs to devices. Figure 4-26 depicts the interaction between the console and Device Manager Server.

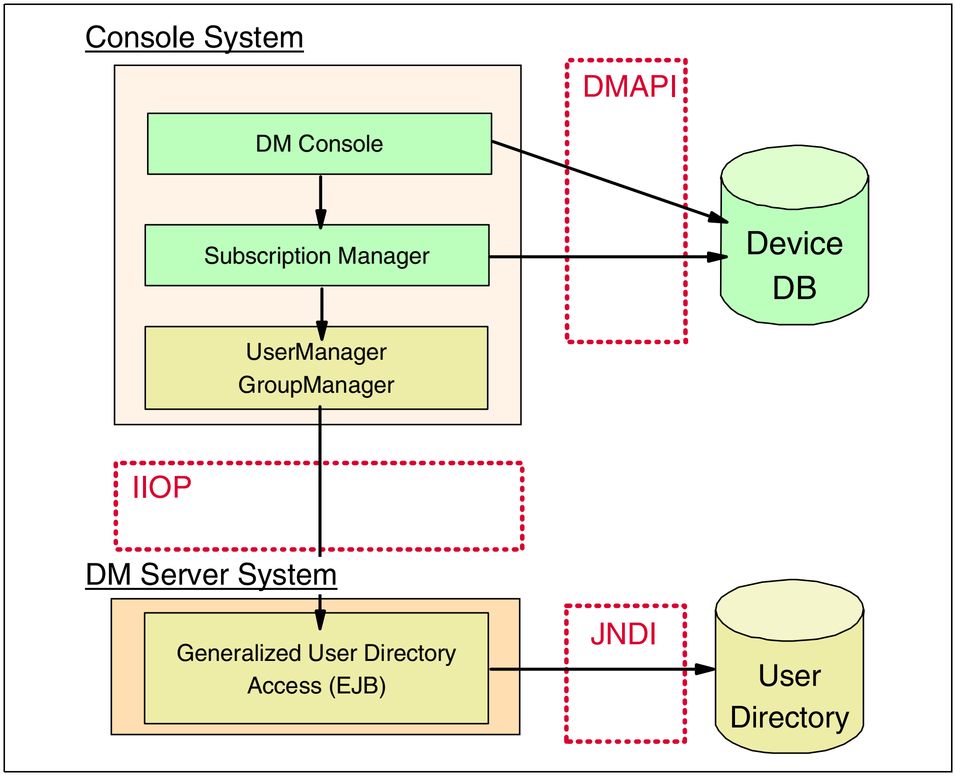


Figure 4-26 Device Manager Server console overview

To execute a job for one or more devices, a job is submitted for the device or devices using the Device Manager Server console or using the job management API. At the time the job is submitted, the administrator or application specifies information such as the job type to be processed, job-specific parameters, target devices, and the activation and expiration time of the job. This job-related information is stored in the Device Manager Server database.

The interfaces between the Device Manager Server and Subscription Manager are well-defined and highly localized. Consequently, the Device Manager Server can be used as a facility through which other applications can perform device management. Essentially, the Device Manager Server is a repository of device data and an engine for processing jobs on devices when they are connected to the network. The Device Manager Server relieves the applications from the need to store detailed device information such as operating system, installed software, and configuration settings, and shields them from the protocols used to communicate with each type of device. The devices supported include:

- Windows CE

- Palm OS
- QNX or Neutrino

When the Device Manager Server is placed behind an authentication proxy software as depicted in Figure 4-23 on page 201, a request from a device arrives at the authentication proxy, then it is routed to the Device Manager Server. Since the Device Manager Server returns its URL (for example, for software distribution) to the client device, it should know the virtual URL of the Device Manager Server that is configured in the authentication proxy. The client device gets this virtual URL and then sends the next request to that virtual URL. The request goes to the Device Manager Server through the authentication proxy.

Device management

Many devices do not have a direct way to retrieve their initial configurations from a Web server, and often require a PC and device accessories for initial configuration. These devices may or may be pre-configured by the issuing companies. If the device is not pre-configured. The device agent program can be made available via diskette or download site, and can be installed onto the device using device facilities such as HotSync for Palm. After the device agent is installed onto the device, the user can start the device agent with the Device Manager Server connection information. When the user presses the Connect button on the device agent panel, the device connects to the Device Management Server and downloads initial configuration data and a set of software that the company has set up for their users.

If the company needs to update or add new services that may require changes to the configuration of their employee's devices, the Device Manager Server can automate the updates. The administrator submits jobs that change the device configuration and distribute the new software. Each employee is notified of this change by e-mail and starts the device agent program to connect to Device Manager Server and complete the update. The following section walks through an example of a device management scenario for a WinCE device.

WinCE device management scenario

This scenario is broken into the following steps:

1. Device agent installation
2. Device enrollment
3. Software distribution
4. User selection for the software distribution
5. User selection for the application package

Device agent installation

First, the user downloads or transfers the WinCE device agent program installation package to the WinCE device. This can be done in one of the following ways:

- If the WinCE device has an Internet connection, the device agent can be downloaded using Pocket PC Internet Explorer.
- If the user has an Internet-connected PC, the user can download the device agent to the PC, then use ActiveSync to transfer the device agent to the WinCE device.
- If the user does not have an Internet connection, the device agent can be received via a CD or diskette. The user copies it to a PC then uses ActiveSync to transfer the program to the WinCE device.

Next, the user double-clicks the installation package to install the device agent on the WinCE device. During the device agent installation, the user enters the following information:

- The user ID
- The machine serial number, which is used to generate the unique device ID
- Other configuration information

If the device agent is put in the WinCE's system startup folder, the device agent runs every time the device starts with the Reset button. Waiting jobs will be processed because the device agent polls the WinCE plug-in running in the Device Manager Server. If the device agent is not in the device's system startup folder, the device agent must be started manually.

Device enrollment

The process of enrolling the device is depicted in Figure 4-27.

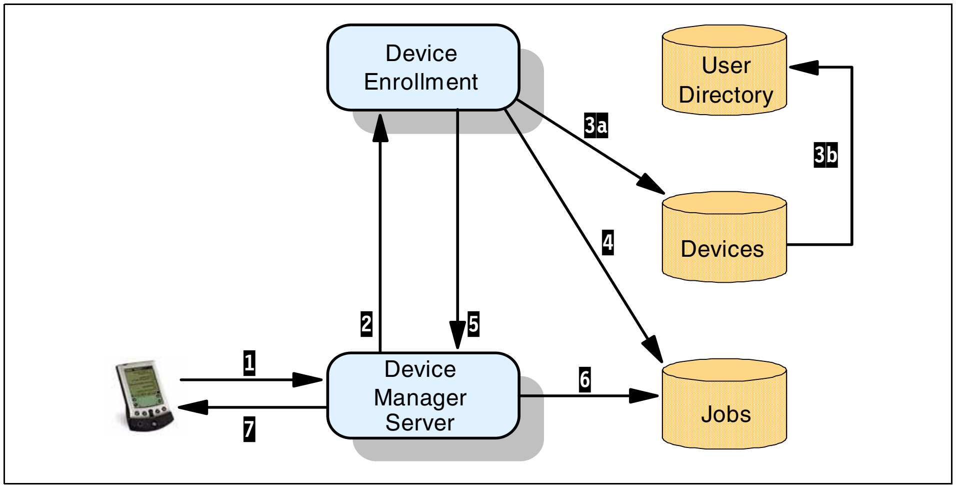


Figure 4-27 Device enrollment process

1. The device agent connects to the Device Manager Server.
2. The device is not known to the Device Manager Server, so the request is redirected to device enrollment. The unique device ID is generated.
3. The device is registered in the Device Manager Server database, and is being associated with a user.
4. A device configuration job is scheduled for the device. The automated device enrollment is completed.
5. The device is redirected back to the Device Manager Server.
6. The newly registered device is now recognized, and because it has jobs scheduled, these jobs will be processed.
7. The actual configuration and/or software is downloaded to the device.

Software distribution

The system administrator uses the Device Manager Server console or an application to create a software distribution job and submit the job for processing. The task to distribute new software is processed by a job created specifically to handle software to WinCE devices. The Windows CE plug-in and the device agent provide all the device-specific communication between the Device Manager Server and the WinCE device.

For a software distribution job, the Device Manager Server retrieves a URL from the Device Manager Server database that points to a Web site where the software can be obtained. The software is distributed to the device with the help of the WinCE device agent.

When distributing software to devices, it may be appropriate to allow the user to choose if the software should be downloaded and installed at this time, to delay the download, or cancel the download. If a software distribution job contains multiple application packages, the job can be set up to present a list of application packages that a user can select which application packages in the job to download.

For WinCE devices, an administrator can define the selections that should be presented to the user. Three selections are available:

1. No software distribution selections allowed for the user
2. User selection for the software distribution job, but not individual application packages
3. User selection for each application package in the job

The administrator prepares a file package definition file for each application package to be distributed, and a metafile package definition file for the software distribution job. The meta file package definition file groups several application packages into a single collection for downloading and it points to file package definition file for each application package in the software distribution job. Normally each application package is packaged as a *.cab file.

No user interaction is required for distributing software. The job is processed when the WinCE device session starts or the device agent polls the Device Manager Server.

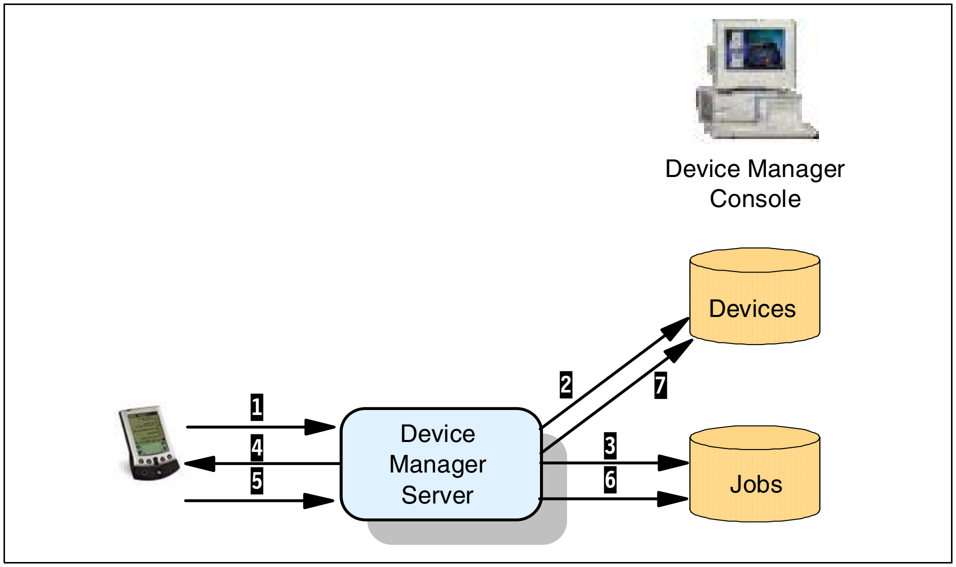


Figure 4-28 Software distribution process

The next time a device agent connects to the Device Manager Server, the following will happen automatically:

1. The device agent connects to the Device Manager Server.
2. The Device Manager Server makes sure that the device is enrolled.
3. The Device Manager Server searches for the job scheduled for the device and its type.
4. The job is processed by sending the software down to the device, where the agent will install it as required.
5. The agent signals that the install has completed successfully.
6. The job database is updated to reflect that the job has run for that device.
7. The device database is updated to reflect that this software is installed on that device.

User selection for the software distribution job

The user can choose to install the software distribution job, delay the distribution, or reject the installation of the software distribution job. In this scenario, there are no choices to install, delay, or reject the distribution for individual application packages.

Before the Device Manager Server downloads any software, the WinCE screen displays a message asking if the user wants to install this software distribution job. The user has these options:

- Yes. Installs the software distribution job now.
- Delay. Installs the software distribution job at a later time.
- Reject. Never installs this software.

User selection for application package in the distribution job

The application package list from the metafile package definition is displayed on the WinCE device screen. The application package list includes the name, version, description, and size for each application package.

4.5.6 WebSphere Transcoding Publisher integration

Refer to WebSphere Transcoding PublisherV3.5 publications for a more detailed discussion of transcoding functions and deployment. WebSphere Transcoding Publisher V3.5 deployment configuration with WebSphere Everyplace Server is detailed in 3.3.6, “IBM WebSphere Transcoding Publisher” on page 103.

WebSphere Transcoding Publisher V3.5 is installed by Enable Offering and configured to store its configuration information in Enable Offering’s product LDAP directory. The Enable Offering’s LDAP directory entry for WebSphere Transcoding Publisher is depicted in Figure 4-29.

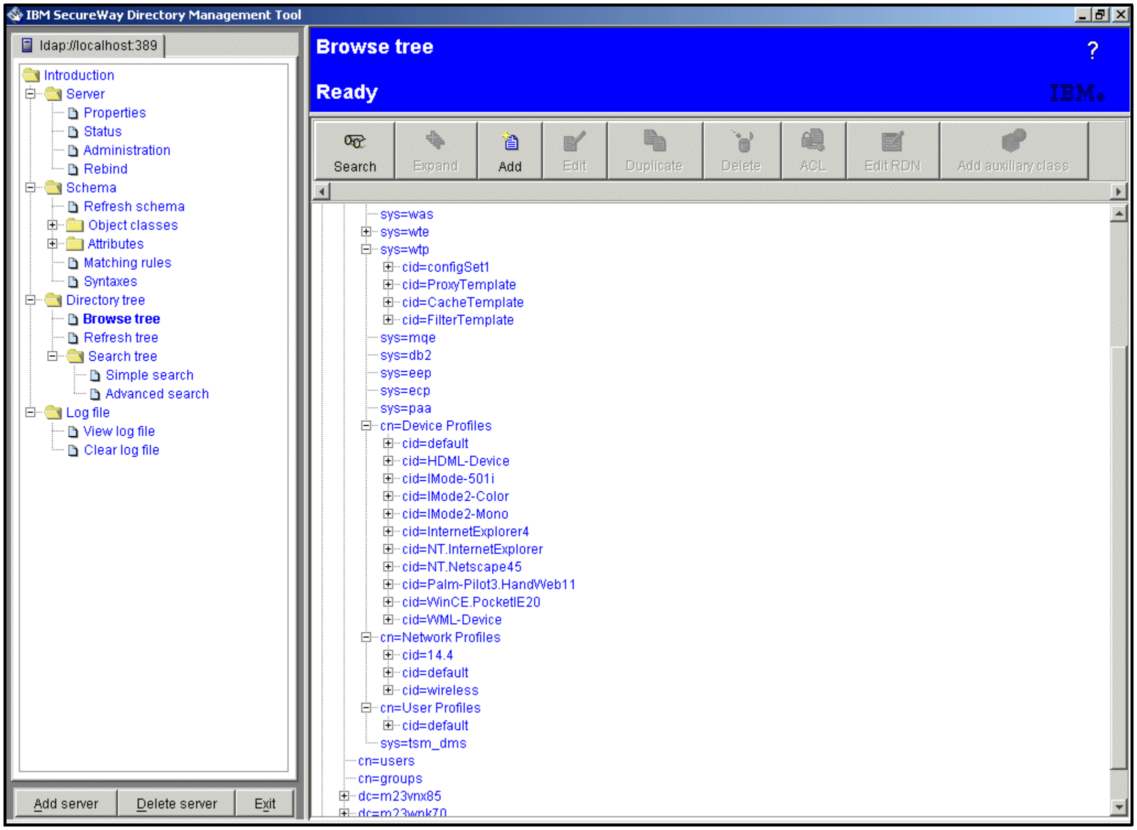
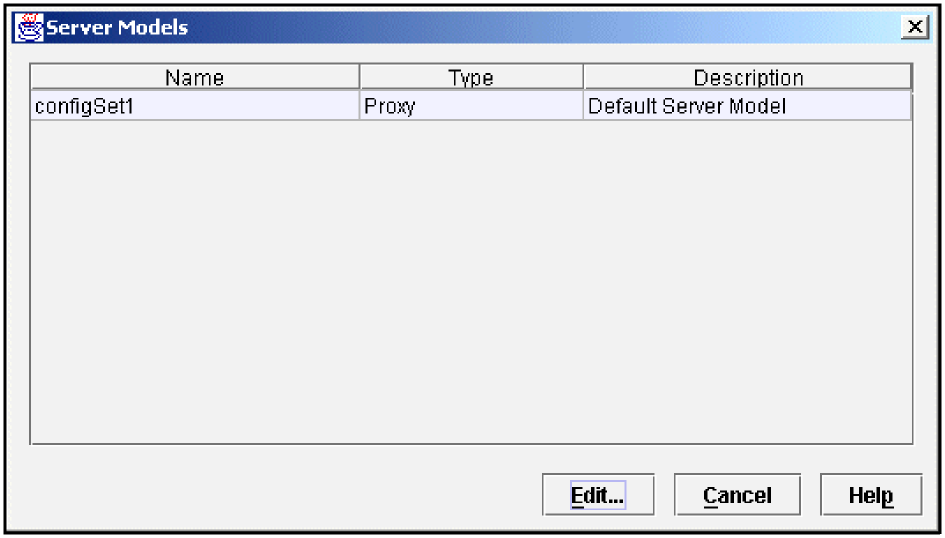


Figure 4-29 Transcoding Publisher integration with Enable Offering LDAP directory

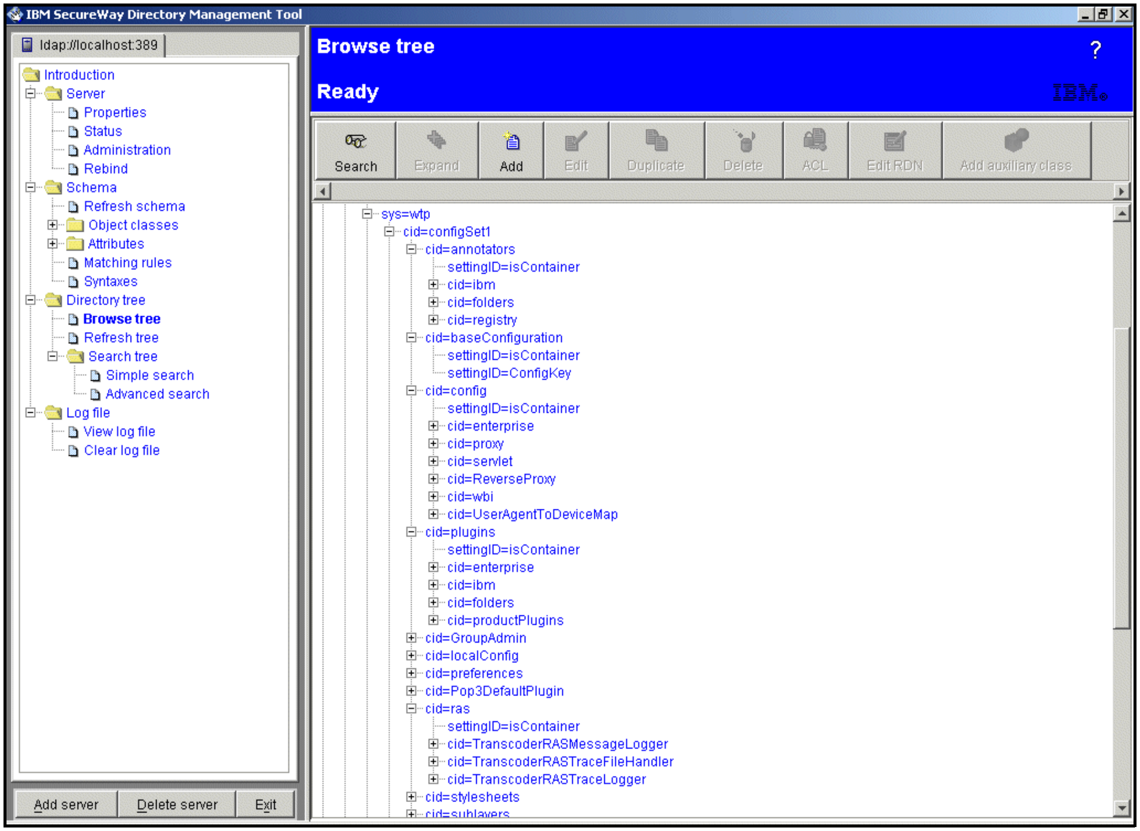
When the configuration is stored in a central LDAP directory, the information is not associated with a particular Transcoding Publisher server. Instead, the directory contains one or more server models, each describing a different set of configuration information, and each Transcoding Publisher identifies which server model it uses. For example, one server model may be used to have high-quality graphical images, and another to be used by the WebSphere Transcoding Publisher server using server model configSet1. The configuration information for configSet1 is depicted in Figure 4-31 on page 212.



Name	Type	Description
configSet1	Proxy	Default Server Model

Figure 4-30 Server model configuration in WebSphere Transcoding Publisher

Figure 4-32 on page 213 shows the relationships between server model and WebSphere Transcoding Publisher servers.



```
sys=wtp
  cid=configSet1
    cid=annotators
      settingID=isContainer
      cid=ibm
      cid=folders
      cid=registry
    cid=baseConfiguration
      settingID=isContainer
      settingID=ConfigKey
    cid=config
      settingID=isContainer
      cid=enterprise
      cid=proxy
      cid=servlet
      cid=ReverseProxy
      cid=wbi
      cid=UserAgentToDeviceMap
    cid=plugins
      settingID=isContainer
      cid=enterprise
      cid=ibm
      cid=folders
      cid=productPlugins
    cid=GroupAdmin
    cid=localConfig
    cid=preferences
    cid=Pop3DefaultPlugin
    cid=ras
      settingID=isContainer
      cid=TranscoderRASMessageLogger
      cid=TranscoderRASTraceFileHandler
      cid=TranscoderRASTraceLogger
    cid=stylesheets
    cid=sublayers
```

Figure 4-31 Server model configuration in Enable Offering LDAP directory

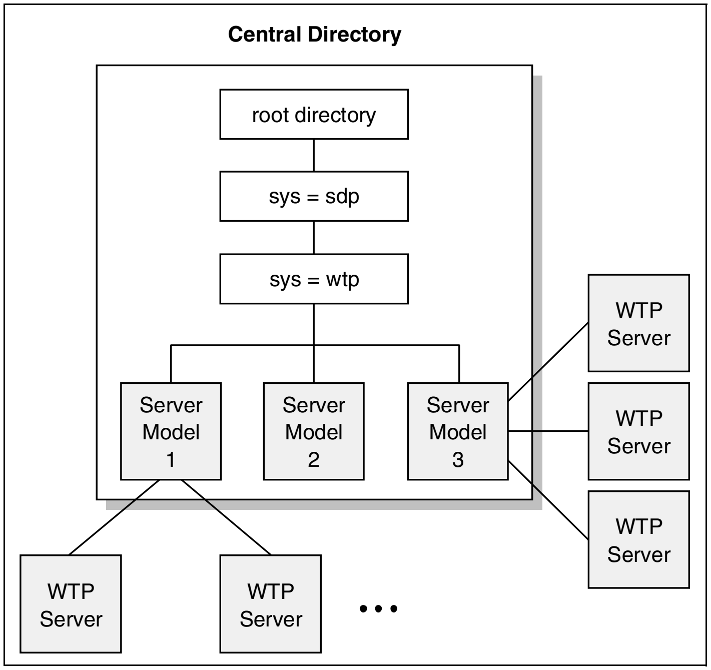


Figure 4-32 WebSphere Transcoding Publisher configured using server model

4.6 Alternative architecture

If QuickClaim's architecture committee agrees to use Tivoli Policy Director as the centralized security management system. This scenario can also be implemented using WebSphere Everyplace Suite Service Provider Offering. WebSphere Everyplace Server Service Provider Offering also provides subscriber management functions.

A detailed discussion of the major architectural differences between the Enable Offering and Service Provider Offering implementations is provided in 4.6.2, "Major differences between implementations" on page 214.

4.6.1 Architecture overview - Service Provider Offering alternative

Figure 4-33 illustrates the alternative solution using IBM WebSphere Everyplace Server Service Provider Offering.

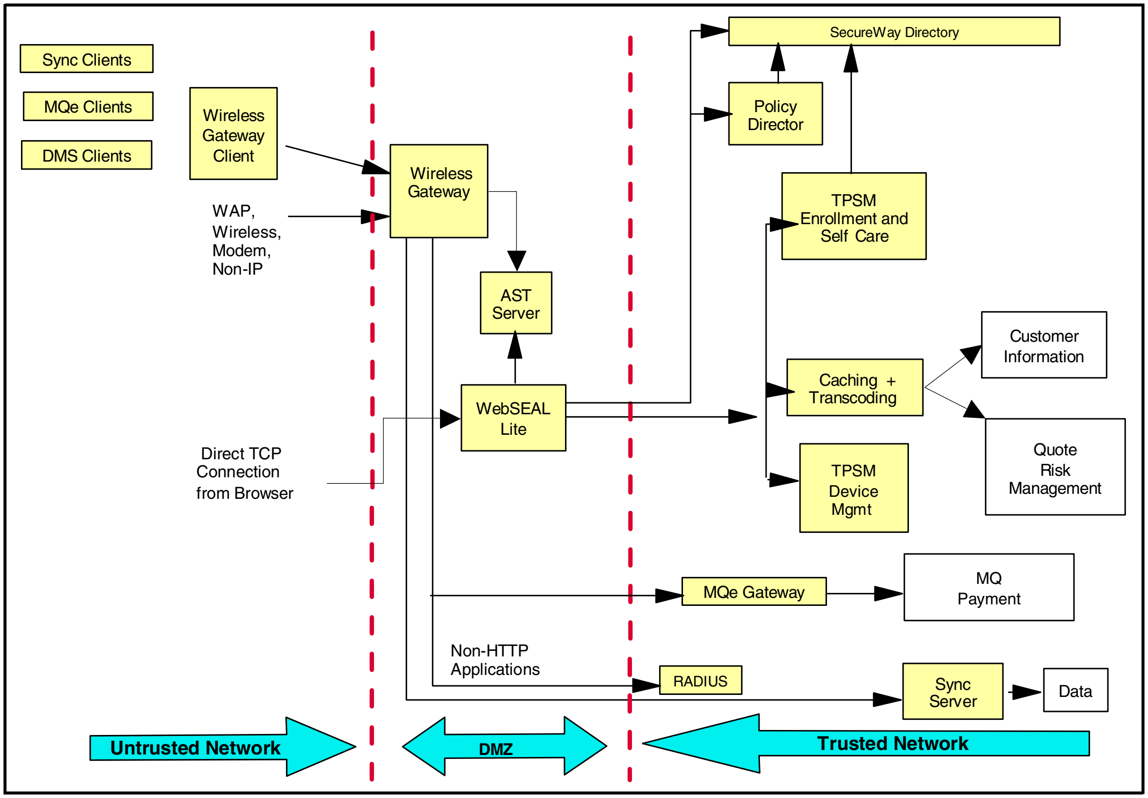


Figure 4-33 B2E implementation using Everyplace Server Service Provider Offering

4.6.2 Major differences between implementations

This section details the major differences between the IBM WebSphere Everyplace Server and the Service Provider Offering.

Authentication

The Service Provider Offering uses WebSEAL-Lite for authentication and supports Tivoli SecureWay Policy Director as the centralized security management system. The Enable Offering uses third-party authentication, such as Netegrity's SiteMinder or Tivoli's WebSEAL for authentication.

Subscriber management

WebSphere Everyplace Server Service Provider Offering uses Tivoli Personalized Services Manager as the subscriber management system, and uses Tivoli SecureWay Policy Director's LDAP directory as described in 3.3.7, “Policy Director” on page 110. The Enable Offering does not provide subscriber management; instead, it relies on the enterprise's existing third-party tool, such as SiteMinder, to provide user enrollment and user directory management. The Enable Offering uses the same user LDAP directory schema as WebSphere Portal Server V1.2's LDAP user directory. Figure 4-13, "Portal Server and Enable Offering user management" on page 174 shows the user LDAP directory relationship between WebSphere Portal Server and the Everyplace Server Enable Offering.

Device management

Service Provider Offering Device Manager Server is based on Tivoli Personalized Services Manager Device Manager Server V1.2, and has the following additional functions:

- Customer care servlet

  Allows management of devices by a customer services representative using a Web browser.

- Self care servlet

  Allows management of devices by the device owners using Web browsers.

Wireless gateway

Everyplace Wireless Gateway is an integral component of Service Provider Offering, but an optional component for the Enable Offering. Refer to "Everyplace Wireless Gateway" on page 12 for an overview of Everyplace Wireless Gateway. "Authentication in Everyplace Wireless Gateway" on page 366 describes the integration between Wireless Gateway and Service Provider Offering in more detail.

With the Enable Offering V 1.1, Everyplace Wireless Gateway can be installed in a stand-alone configuration. The gateway will be configured with no authentication, and authentication will be performed by the third-party authenticator, SiteMinder or Policy Director. With new Everyplace Wireless Gateway fixes soon to be released, one can configure the gateway to accomplish single sign-on from the gateway to WebSphere back-end applications, including device management for WAP devices with Policy Director basic authentication. Refer to Wireless Gateway documentation for more details.

Device type in HTTP header

With WebSphere Everyplace Server Service Provider Offering V2.1, the authentication server adds WebSphere Everyplace Server unique header in the HTTP request that includes network type, device type, user ID, etc.

With WebSphere Everyplace Server Enable Offering, the authentication can be no authentication, WebSphere Application Server V3.5 authentication, or third-party authentication. There is no unique HTTP headers for Enable Offering V1.1.2. All the WebSphere Everyplace Server Enable Offering components - - including WebSphere Transcoding Publisher, and back-end applications can obtain device information from the UserAgent field in the HTTP header, and can obtain a user ID from the TAI.

Synchronization

Everyplace Synchronization Manager was developed on a UNIX platform to provide IBM Mobil Connect functions and a tighter integration with WebSphere Everyplace Suite Service Provider Offering using an LDAP directory server. Synchronization Manager is based on IBM Mobile Connect Version 2.41 level of functions and features. It was ported using MainSoft MainWin libraries that preserve Microsoft API, such as VBScript. The following are Synchronization Manager Core components and relationships with IBM Mobil Connect:

- Synchronization Manager Server

  The Synchronization Manager server runs on a UNIX server (AIX or Solaris), performs data synchronization functions, and manages security.

- Synchronization Manager admin

  The Synchronization Manager admin also runs on a UNIX server and is the user interface to configure and maintain the Synchronization Manager server.

- Exchange Connector

  The Exchange Connector must run on Windows NT 4.0 or Windows 2000 and provides authentication and PIM data transfer between Synchronization Manager server and Microsoft Exchange. The Exchange connector has similar requirements to the IBM Mobil Connect server.

- Notes Connector

  The Notes Connector runs on UNIX and provides authenticating and PIM data transfer between Synchronization Manager server and Notes. The Notes Connector requires Domino Mail Server be installed and configured, but not necessarily running on the same server as the Notes Connector program, to provide the Notes Connector with access to Notes APIs.

- Proxy

  Synchronization Manager proxies are similar to IBM Mobil Connect proxies.

- Clients

  Synchronization Manager clients are similar to IBM Mobil Connect clients.

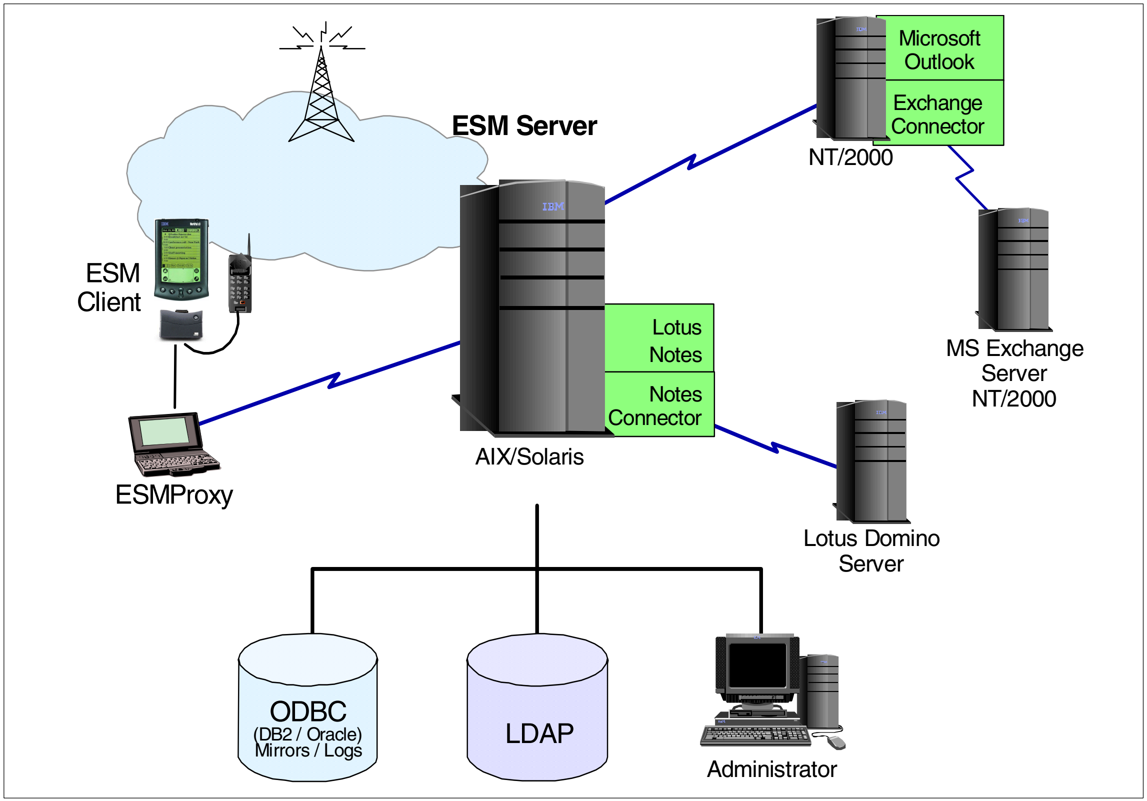


Figure 4-34 Synchronization Manager Overview

Figure 4-34 is an overview of Synchronization Manager components. Microsoft Exchange and Lotus Notes connectors run separately from Synchronization Manager server. The IBM Mobil Connect service directly calls Microsoft Exchange and Lotus Notes APIs. This is done from the connectors in the Synchronization Manager product.

Authentication

Service Provider Offering V2.1 will authenticate with Wireless Gateway first, then authenticate again with either Notes or Exchange. Single sign-on is not supported by Synchronization Manager V2.1 (shipped as a WebSphere Everyplace Server Service Provider Offering 2.1 APAR), which allows for authentication of user ID and password using WebSphere Everyplace Server Service Provider Offering 2.1 LDAP schema for relational database synchronization, while IBM Mobil Connect V2.5 allows user LDAP authentication of user ID and password.

With IBM Mobil Connect, all error and session information was stored in a Microsoft Access database called connlog.mdb and was displayed from the IBM Mobil Connect admin with Crystal Reports. Synchronization Manager stores key information such as mirror files, log files in DB2 or Oracle database, and configuration files in LDAP. With IBM Mobil Connect, configuration and mirror files were stored on the IBM Mobil Connect server in their own proprietary format and not stored in database.

Time-Zone support

Different time zone handling was added by creating a new TimeZone system tag. With IBM Mobil Connect, separate servers are needed for each time zone to support Palm OS and EPOC devices because only one IBM Mobil Connect service can be running at a time on the server. For Synchronization Manager, multiple exchange connectors and the addition of the `$TIMEZONE` tag for Notes provide finer grain control of time zone issues.

Platform support

Service Provider Offering requires UNIX systems, while Enable Offering supports UNIX, Windows NT or Windows 2000.

5

Business-to-consumer

In this chapter, we describe how to integrate a new wireless solution into an existing customer environment. To do this we will use a fictitious company, Star Hotels, that would like to offer their customers the ease of having complete wireless coverage on and off their facilities.

We describe the existing environment at Star Hotels, their requirements and then we will define a solution for our client. We will then demonstrate how we can resolve their requirements using IBM WebSphere Everyplace Server Service Provider Offering.

A technical solution for the customer will be explained, and a detailed implementation of each of the IBM WebSphere Everyplace Server components is included in the solution.

5.1 Business scenario overview

Star Hotels is a large hotel that has hotels all over the world. Currently their IT infrastructure is based on a local intranet within each hotel that is connected through a high-speed communication link to their head office computer system in Paris. All the hotels have their critical applications, such as booking and billing, running locally within the hotel. Changes to these application databases are being replicated to the central computer system on an hourly basis.

All the hotels get a significant percentage of their revenue contribution from their convention and conference offerings. As the competition in this area is hardening, Star Hotels' management wants to be able to offer additional new services to their customers in order to attract more convention business.

These new services should include conference programs available through a centralized portal as well as multimedia services such as viewing selective parallel conference sessions from the customer's laptop PC and being able to open up communication channels (chat) with one or more of the participants in the conference. They also want to be able to offer the convention and conference participants selectable interactive courses to attend before, during and after their conference or convention.

To have a flexible connection to the hotel's intranet, an IEEE 802.11b-based wireless LAN solution is being installed in each hotel enabling hotel employees as well as conference participants to connect with their own laptops. To be able to authenticate on the intranet, each customer must be enrolled into the system and also each customer should easily be able to enroll themselves. This self-enrollment will be the same as being enrolled as a bonus-card member of Star Hotels.

Start Hotels has a bonus-program consisting of bonus program groups such as silver, gold and platinum, depending on how many bonus points the customer has achieved or collected. Once the customer has become a bonus program member, which requires providing a credit-card number, Star Hotels has all the information required to authenticate the user on its intranet and bill them for services provided. This solution also opens up for additional services such as wireless booking and billing.

Each customer should be able to access the hotel services not only when they are connected to the Wireless LAN, but also when they are using pervasive devices such as WAP phones and PDAs. Star Hotels wants to offer their customers a service that allows the hotel to send information on special promotions, such as special weekend offers. Star Hotels also wants to offer location-based services, which would enable each customer with a cell phone to query the system to locate the nearest hotel or other commercial services such as shopping malls, airports, restaurants, and other places of interest.

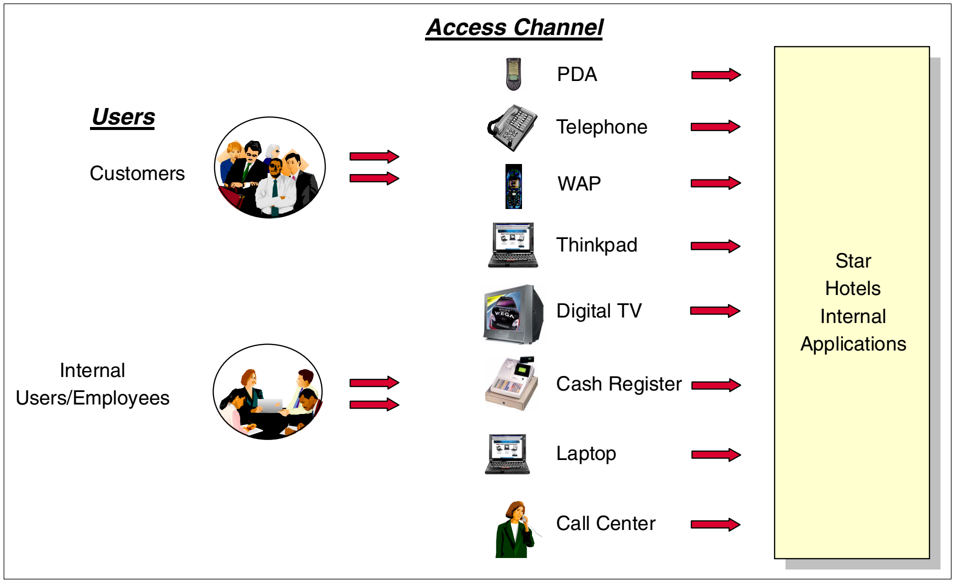


Figure 5-1 Access channels

Access to this system should be through a centralized portal that will contain all the services provided. As different pervasive devices will access the portal, its presentation layer should be in XML, enabling the system to transcode its contents to the appropriate pervasive device. With self-enrollment and self-care functionality, Star Hotels expects fewer loads on their customer care department.

The Star Hotels wants this new system to work with the existing hotel back-end application systems as well as the new centralized conference systems consisting of Lotus QuickPlace, Sametime and Mindspan.

Although Star Hotels has subsidiaries all over the world, the first phase of the new offering will be available in Europe only.

5.1.1 Requirements

Star Hotels wants their system to be working 24 hours a day, 7 days a week. Customers should expect immediate response from the system, which is defined as less than or equal to 5 seconds. The expectation is to have around 100,000 concurrent users online and the capacity to handle five new users per second.

Star Hotels also expects customers to enroll from all over the world. This emphasizes the importance of customers being able to enroll themselves into the system and create their own user IDs and passwords.

As credit-card information may be required to be entered into the system, the system must provide easy and secure transactions. Customers will access the system through a single user name and password, and will be able to access all authorized applications with this single user name and password (single sign-on).

The new system will provide different levels of access for different levels of users, for example different categories of bonus program customers, hotel employees, system administrators, and others.

Since Star Hotels wants to have a special bonus program affiliation to be used by their customers, they will give special gifts and privileges depending on the level of participation of each customer. Depending on their status, customers will have different types of access to the system.

Customers will have the ability to work offline and later connect to the network to submit their work, then later call for new work and messages. For this purpose, the customers will have installed in their devices a special client that will talk with the server of the system.

Each customer will have access to his or her billing account during their stay with the option to deliver the bill by e-mail. Customers will also have the ability to access information via their telephones using the hotel voice facility. Star Hotels expects that the new system will give future support to both GPRS and UMTS.

5.1.2 Current Infrastructure

The current system of the Star Hotels works with Windows 2000 and Solaris servers. The system must be installed through a new centralized high availability multichannel Internet infrastructure, integrated with the existing LAN, WLAN and VPN.

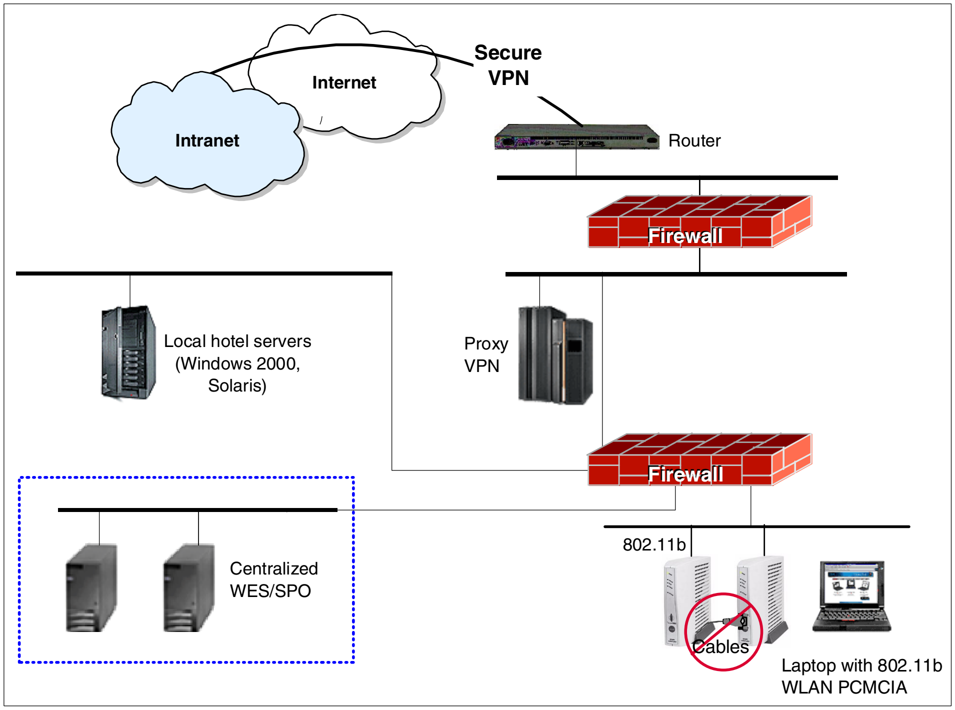


Figure 5-2 Current solution with new centralized Everyplace Server

5.2 Architecture overview

The following solution is based on WebSphere Everyplace Server Service Provider Offering. The components of WebSphere Everyplace Server Service Provider Offering that would be used for this project are as follows (the acronyms are how they are illustrated in Figure 5-3):

- IBM Everyplace Wireless Gateway (EWG)
- Active Session Table Server (AST)
- Edge Server Caching Proxy (ESCP)
- WebSEAL-Lite (WSL)
- Remote Authentication Dial-in User Service (RADIUS)
- Tivoli SecureWay Policy Directorr (PD)
- IBM SecureWay Directory (LDAP)
- Intelligent Notification Services (INS)

- Tivoli Personalized Services Manager (TPSM)
- Tivoli Device Manager Server (DMS)
- WebSphere Transcoding Publisher (WTP)
- Location Based Services (LBS)
- WebSphere Application Server (WAS)

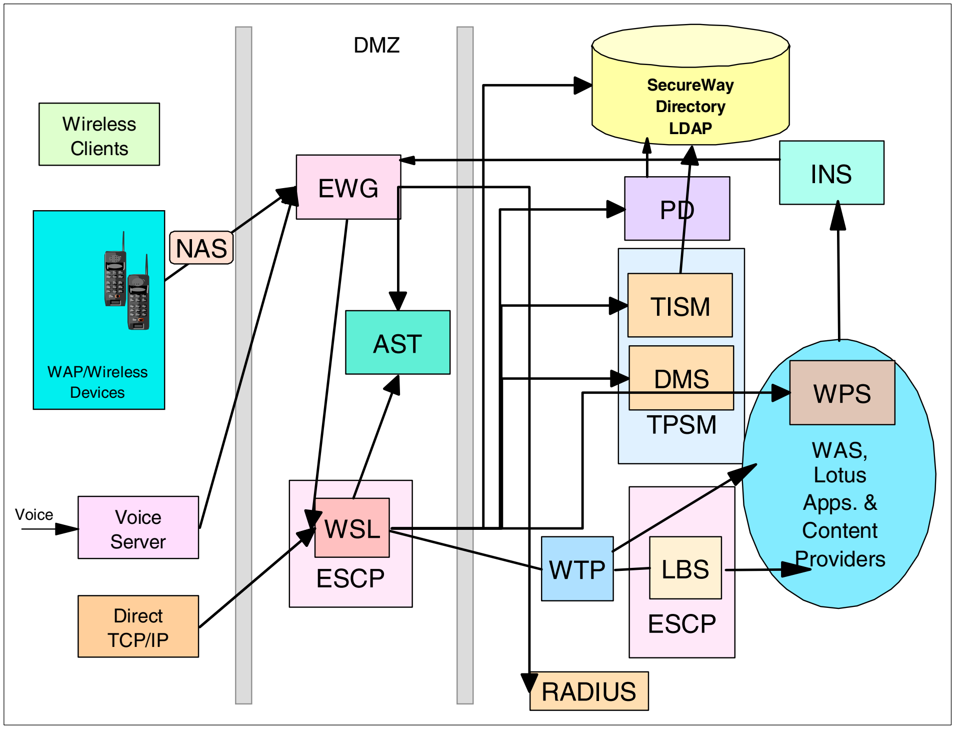


Figure 5-3 Architectural overview of this scenario

As illustrated in Figure 5-3, integration of different components of the IBM WebSphere Everyplace Server Service Provider Offering will require a detailed description of each of the products, its functionalities and its role in the technical solution for Star Hotels.

5.2.1 Architectural decisions

The concerns of high availability and capacity for its users will be addressed by using IBM AIX as the software platform.

Self enrollment is an important requirement for Star Hotels, so to handle the process, WebSphere Everyplace Server Service Provider Offering will handle the issue with Tivoli Personalized Services Manager in coordination with other components.

Granting different levels of access to the system, depending on the level of the users bonus program status, will be handled by Policy Director in conjunction with WebSEAL-Lite, the Edge Server Caching Proxy, and Tivoli Internet Services Manager.

Intelligent Notification Services will provide notification services to Star Hotels' clients and employees alike. The Location Based Services will provide location-based services and the Tivoli Device Manager Server (DMS) will be used for distribution of the wireless client that enables devices to access the system over various wired solutions as well as dial-up solutions.

The Wireless Gateway will be used to enable connections of all wireless clients including WAP phones, PDAs and dial-up users.

The centralized portal will be implemented with WebSphere Portal Server (WPS). This portal server has the ease and flexibility that is required by Star Hotels to handle frequent content changes and personalized presentations.

Transcoding Web content

Star Hotels wants its customers and employees to have access to as many applications as possible from as many different kind of devices as possible, such as PalmPilots, laptops, and telephones. All the different clients should be supported, and the system should work as a translator to format application data into markup-languages used by the specific devices. Also the system must be prepared to easily be adapted to new markup languages used by new or existing devices. To facilitate the location-based services, location-based applications need to be developed to provide content to the subscribers. All these content providers will be deployed as device-independent XML-based applications. Deploying the WebSphere Transcoding Publisher in front of the location-based applications will facilitate transcoding the content request responses into device-specific format such as HTML, WML, and others.

The first application Star Hotels is considering to adapt for use by different types of devices is the reservation application. Customers should be able to access this application to make their reservation from PalmPilots, laptops, telephones, and others.

To perform this requirements we will add into the system WebSphere Transcoding Publisher. WebSphere Transcoding Publisher will perform all the code transformation between client devices and system back-end applications.

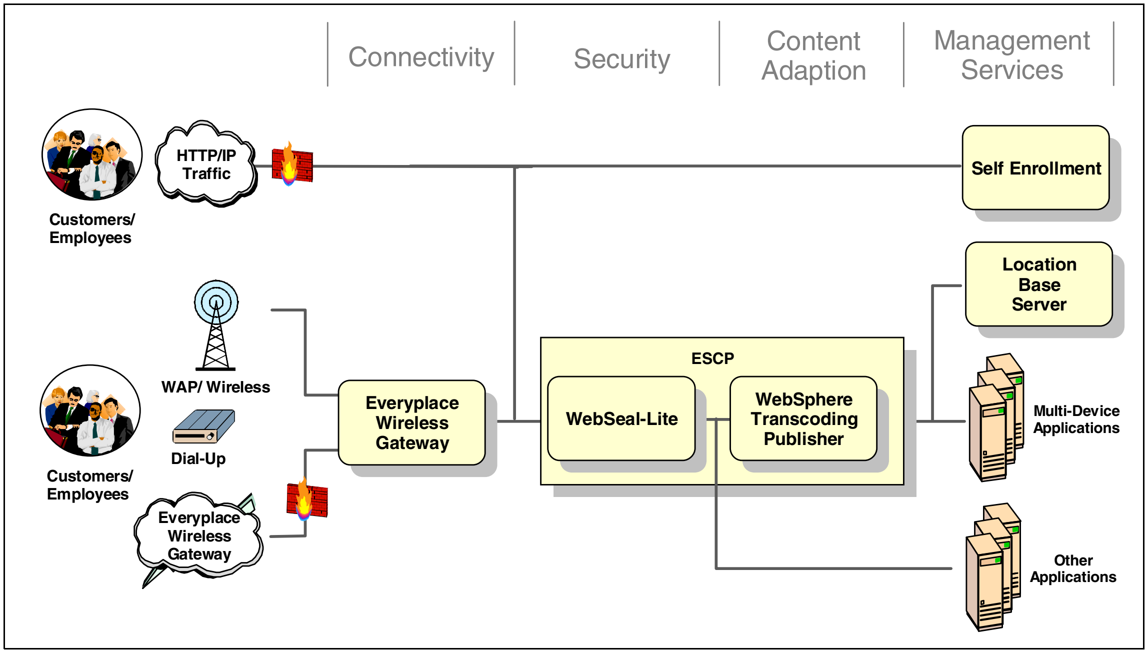


Figure 5-4 WebSphere Transcoding Publisher inside Star Hotels system

WebSphere Transcoding Publisher has three different scenarios in which it can be deployed (see 1.3.7, “WebSphere WebSphere Transcoding Publisher” on page 27). In the Star Hotels system, we will deploy WebSphere Transcoding Publisher as a reverse proxy model.

Figure 5-4 illustrates the Star Hotels system configuration. Included is WebSphere Transcoding Publisher between WebSEAL-Lite and the user applications running on an application server. With this configuration WebSphere Transcoding Publisher sits between the client and any Web server that can be reached by the client.

WebSphere Transcoding Publisher will provide Star Hotels the ability to make Web-based information available to handheld and other devices economically and efficiently. Customers and employees will receive information tailored to the capabilities of the devices they are using. For example, customers with small-screen devices access a scaled version of the information, while employees of a specialized markup language access the same information in a format suitable for their devices. By providing a single dissemination point for multiple renderings of information, WebSphere Transcoding Publisher eliminates the expense of re-authoring or porting data or applications for multiple networks and devices. WebSphere Transcoding Publisher improves the communications and effectiveness of Star Hotels mobile employees.

WebSphere Transcoding Publisher will transform content based on what it knows the requesting device can handle, and on the capacity of the network being used. Web content can be transformed differently for different devices and networks. WebSphere Transcoding Publisher will meet the requirement to support Star Hotels existing applications and data without requiring a redesign of corporate systems to accommodate the different standards found in each device. WebSphere Transcoding Publisher can support all common types of Web data, including HTML pages from a Web server, HTML output from a host application, and Extensible Markup Language (XML) data from a back-end transaction system. WebSphere Transcoding Publisher will also tailor images to adjust screen size, file size, and numbers of colors.

All existing hotel applications should, in the future, work with a standard markup language (XML) as their input and output. This way applications will be multi-device and the responsibility to adapt the information for the device will be taken by WebSphere Transcoding Publisher.

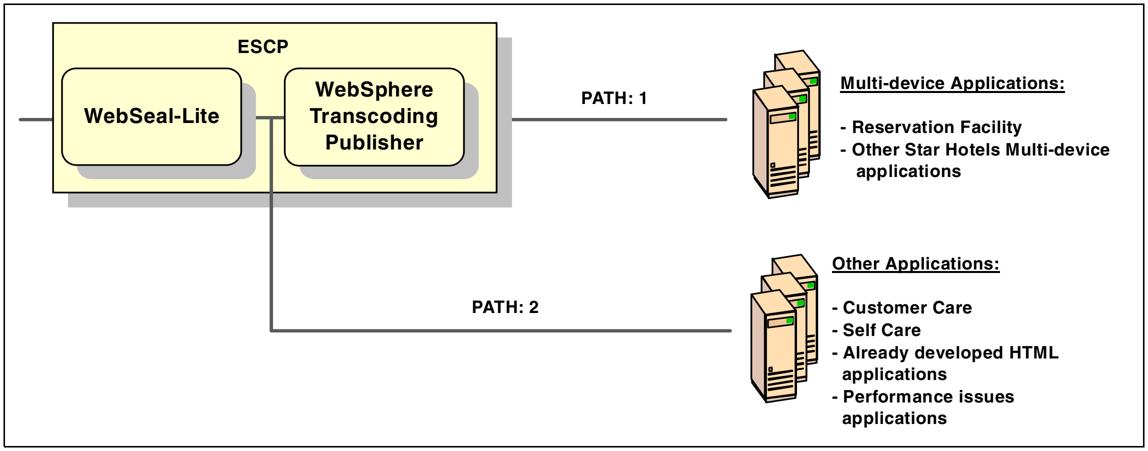

ESCP
WebSeal-Lite
WebSphere Transcoding Publisher
PATH: 1
Multi-device Applications:
- Reservation Facility
- Other Star Hotels Multi-device applications
PATH: 2
Other Applications:
- Customer Care
- Self Care
- Already developed HTML applications
- Performance issues applications

Figure 5-5 WebSphere Transcoding Publisher: Devices accessing applications

In the Star Hotels system there will be two ways to access back-end applications.

1. The normal way will be through WebSphere Transcoding Publisher (see Figure 5-5 Path 1).
2. In cases where performance could be an issue, such as the Check-out application that needs a responds time almost instantly, the access will be directly from the Caching Proxy to applications (see Figure 5-5 Path 2).

Applications already developed in HTML and accessed through browsers can coexist with the new system; they should be deployed as any other application that does not use WebSphere Transcoding Publisher.

To decide how an application should be accessed, either through WebSphere Transcoding Publisher or not, depends on how the application is going to be used. By default all applications should be developed to work with a standard markup language (XML). If the application is going to be accessed from different kinds of devices, the access should definitely go through WebSphere Transcoding Publisher. This allows just one application for all different devices and avoids difficulties with maintenance.

If performance is an issue, or the application is going to be accessed through just one kind of device (such as HTML applications accessed through a browser), as it is for the Star Hotels "Check-out" application, the path should go from the Caching Proxy directly to the application. In this way we avoid any translation done by WebSphere Transcoding Publisher and we improve performance.

Voice server

Although the Star Hotels wants to deploy a voice-based channel to access some of their back-end applications, we will defer this deployment until WebSphere Transcoding Publisher V4.0 is available via IBM WebSphere Everyplace Server. This release is required to transcode VoiceXML into device dependent formats. See 5.2.8, “Voice services” on page 264 for a description of the voice server.

WebSEAL-Lite junctions

As shown in the architectural overview in Figure 5-3 on page 224, the Caching Proxy plug-ins WebSEAL-Lite, Location Based Services and WebSphere Transcoding Publisher ares not going to be configured in sequence, one after one other, since this will create unnecessary overhead. To be able to route requests to location-based applications and portal applications without going through WebSphere Transcoding Publisher, and to route requests to applications that need to be transcoded without going through Location Based Services, we will implement the secure junctions of WebSEAL-Lite.

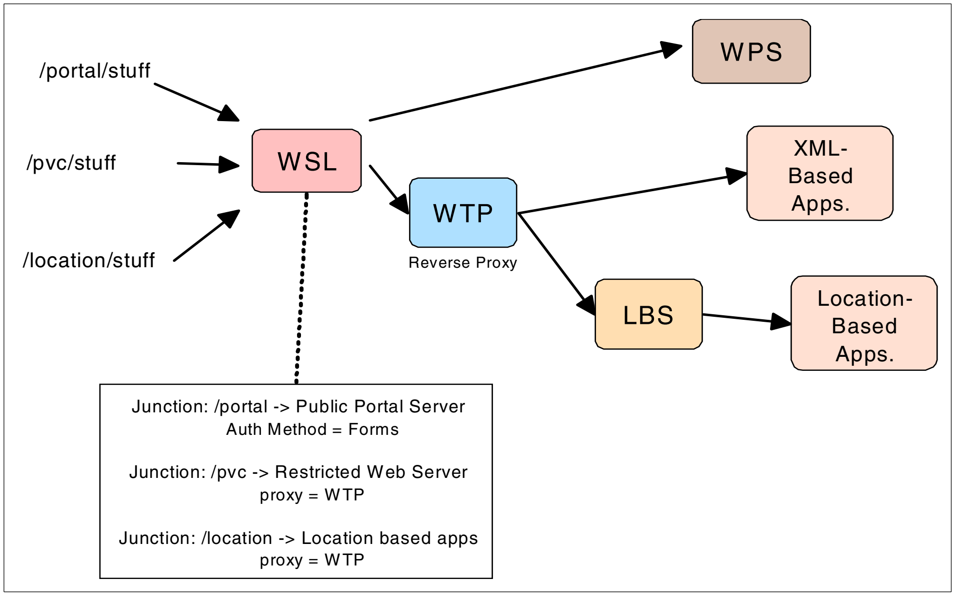


Figure 5-6 Secure junctions in WSL[1]

However, please note that the use of the *junctions* in WebSEAL-Lite will impose another administration task in the operational environment. Each of the junctions are identified by their partial URL, thus, new URLs (new applications) may imply updating the configuration file of WebSEAL-Lite.

Authentication of users connecting over wireless devices

In WebSphere Everyplace Server, all wireless devices (for wireless *clients*, it's different, see "Wireless clients (RC1b)" on page 293) that are connecting to the Everyplace Wireless Gateway (EWG) are assumed to already be authenticated by a Network Access Server (NAS) or another third-party authenticator.

The NAS allocates IP addresses (on a *round-robin* basis) to the WAP-devices. This IP address does not uniquely identify a specific user or device as the NAS is configured to distribute an IP address from a pool (*round-robin*) of addresses to devices as they connect to the network. A WAP device will, in this configuration, typically not have the same address every time it is connecting to the network.

[1] Not all our junctions are included in this figure. Others are needed, such as in "self enrollment" in Figure 5-9 on page 233.

Remote Authentication Dial In User Service (RADIUS) is a protocol for carrying authentication, authorization and configuration information between an NAS and an authentication server. The RADIUS *accounting protocol* provides the means for an NAS to send accounting information to a shared authentication server (in our case the Everyplace Wireless Gateway, not the WebSEAL-Lite authentication server).

The Wireless Gateway has a new gatekeeper resource (Device Resolver) that represents an external NAS. This feature will trust each NAS by its IP address and assume that the user and device authentication occurs at the NAS.

In this scenario the Wireless Gateway will interpret the RADIUS accounting-packets to learn the identifier associated with the device currently connecting to the NAS. This identifier will thus typically be the IP address allocated to the device and the phone number of the device (the MSISDN in our scenario).

Figure 5-7 illustrates the data flow in this environment.

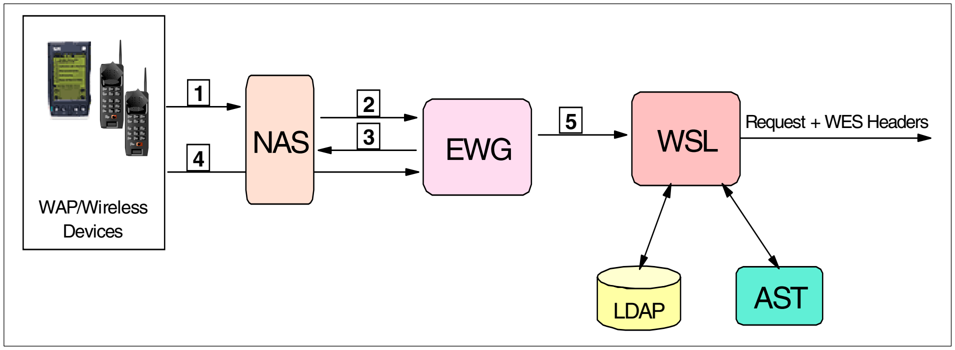


Figure 5-7 Connecting wireless devices

In this scenario the data flows as follows:

1. WAP phone (wireless device) connects to the NAS and authenticates by the NAS.
2. NAS forwards a RADIUS accounting start indication to Everyplace Wireless Gateway. This indication includes the IP address of the device and the phone number (MSISDN).
3. The Everyplace Wireless Gateway acknowledge the RADIUS request.
4. User requests a Web page (URL) using WAP protocols.
5. Everyplace Wireless Gateway inserts the MSISDN into a Client-Id header and forwards the request to WebSEAL-Lite, which will find the matching user ID in the IBM SecureWay Directory (LDAP), insert a session ID in the Active

Session Table (if it's not already there), create a session ID header and pass the request and its headers to the requested Web page.

The reason why WebSEAL-Lite is not authenticating the user, in this scenario, is because the Everyplace Wireless Gateway is configured as a trusted gateway, and when WebSEAL-Lite receives a request with an `X-IBM-PVC-Client-Id` header, it assumes that the user is already authenticated. This scenario will in most cases work well in the United States, whereas in Europe, when the connection is made over GSM, the scenario will be different.

Note: If single sign-on is not your concern, you may also let the back-end application servers take care of the user authentication, that is, deploy the scenario as described and let your back-end HTTP server challenge the user for the user ID/password.

In Europe, most wireless devices such as WAP and PDAs connecting through the mobile network (GSM) will connect through a *specialized* router acting as a Network Access Server (NAS) and having the capability to connect with the mobile network over a number of ISDN-connections (the *specialization* is the V.110[2]).

Because most commonly used GSM WAP phones in Europe have the ability to store both the NAS IP address (which is the WAP phone access point) as well as the user ID and password on that NAS, we would like to use the same user ID and password for all users of the NAS and use the MSISDN as the device identifier (Client-Id) and let WebSEAL-Lite carry out the authentication of the user related to that specific device.

Currently, this scenario will not work with the WebSphere Everyplace Server. During installation the Everyplace Wireless Gateway will be configured in the IBM SecureWay Directory as a trusted gateway to WebSEAL-Lite. When WebSEAL-Lite receives an MSISDN as an `X-IBM-PVC-Client-Id` header from a trusted gateway (in our case, the Everyplace Wireless Gateway), *it will assume the request already has been authenticated* (as previously explained).

As a work-around in this scenario, we will insert another NAS acting as a forward proxy in between the Everyplace Wireless Gateway and WebSEAL-Lite to *conceal* the trusted Everyplace Wireless Gateway.

This configuration is shown in Figure 5-8.

[2] The V.110 recommendation of the ITU-T adapts a low-speed connection to an ISDN-B channel allowing the remote station or terminal adaptor to use the fast call setup times offered by ISDN. This feature enables GSM wireless connectivity.

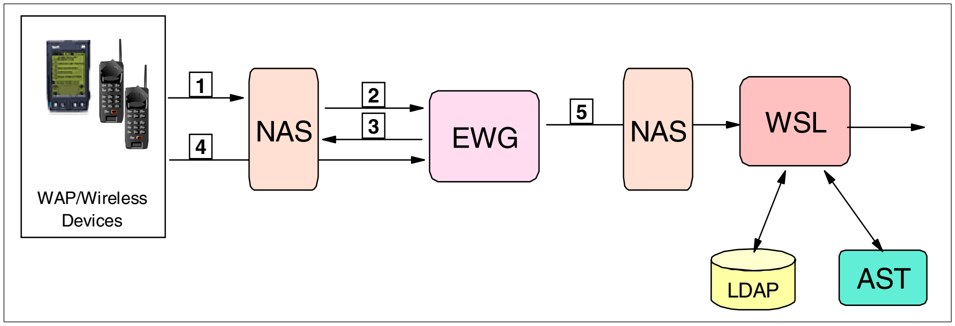


Figure 5-8 Wireless devices to be challenged by WebSEAL-Lite

In this scenario the authentication flows as follows:

1. WAP phone connects to the NAS.
2. NAS forwards a RADIUS accounting start indication to the Everyplace Wireless Gateway. The indication includes the IP address of the device and the phone number (MSISDN).
3. The Everyplace Wireless Gateway acknowledges the RADIUS request.
4. User requests a Web page (URL) using WAP protocols.
5. Everyplace Wireless Gateway converts the request to HTTP and appends the MSISDN (for example, phone number) in a `Client-ID` header. The request is forwarded to the WebSEAL-Lite through another NAS to conceal the IP address of the Everyplace Wireless Gateway. WebSEAL-Lite queries the IBM SecureWay Directory for the user ID corresponding to the MSISDN and checks the Active Session Table for an active session ID, challenges the user (401 error code) for user ID/password if it's not there![3]

The *Device resolver* of the Everyplace Wireless Gateway needs to be configured to device identification attribute that is sent by the first NAS you would like the Everyplace Wireless Gateway to pass on as the `Device-Id` header. In our case we chose the MSISDN as the identification attribute. You also have to configure if you would like to have the Everyplace Wireless Gateway to serve as a RADIUS proxy forwarding the accounting packages to a second RADIUS server. In the case of the Star Hotels, we certainly would like to forward this information, even to a third RADIUS server in the secure zone, for accounting and statistics processing.

[3] See also "WAP requests (RC1a)" on page 292.

5.2.2 Self enrollment

To attract more loyal customers, Star Hotels offers a bonus program. To make it as easy and simple as possible to join the bonus program, Star Hotels wants to offer a self enrollment site over the Internet that enables new customers to sign up for the bonus program as well as to sign up for the various services offered to their customers.

These services will include conference services, check-in and check-out services as well as notification and location-based services. Since many of the services offered will require the customer to enter their credit card number, the connection must be secured via HTTPS.

The bonus program places the users into groups based on how many bonus points they have earned. Different services and/or offerings will be available for the different groups. Tivoli SecureWay Policy Director offers a secure and flexible ACL-based authorization mechanism that will enable us to differentiate the user groups.

The operational implications of this self-enrollment implementation is the 24/7 availability requirement and the complexity of the architecture. However, the advantage is the reduced demand for services from the customer care center, since the user may enroll himself. The availability of the service as such is also an advantage since anyone is enabled to sign up for the bonus program as long as they can provide valid credit card information.

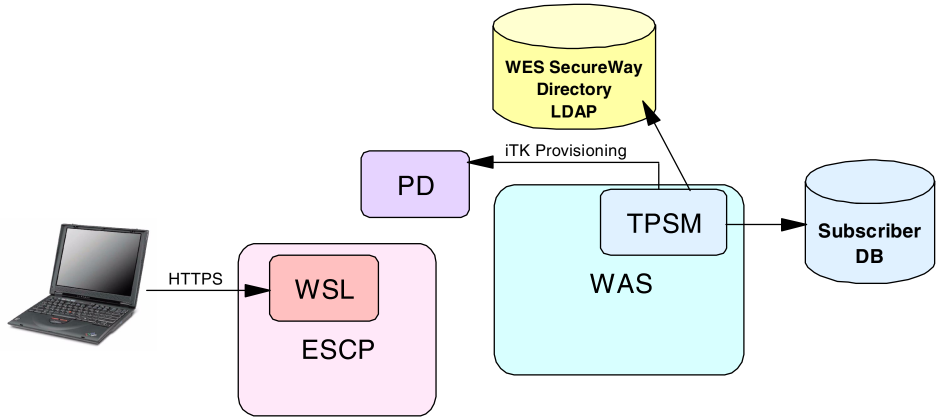


Figure 5-9 Self enrollment

Tivoli Personalized Services Manager provides a self-enrollment servlet and JSP as a base function. The JSP needs to be customized according to customer requirements, such as including the functionality requested by the bonus program and the various customer categories.

In our scenario, the self-enrollment servlet is running in a DMZ with WebSEAL-Lite configured to let any request through to the self-enrollment servlet enabling anyone to enroll. This configuration is accomplished by utilizing the WebSEAL-Lite *junctions* (refer to Figure 5-9).

For authentication and authorization purposes, WebSEAL-Lite will query the Tivoli SecureWay Policy Director which has its own instance of IBM SecureWay Directory for group-affinity purposes. To associate Policy Director group(s) with Tivoli Personalized Services Manager realms and deals, we need to propagate the appropriate user information from the Tivoli Personalized Services Manager subscriber database into the appropriate group in the Tivoli SecureWay Policy Director controlled SecureWay Directory. This is carried out by a Tivoli Personalized Services Manager iTk provisioning client written in Java[4]. The same mechanism is used to propagate the subscriber into the IBM SecureWay Directory.

Tivoli Personalization Services Manager iTk provisioning

The Tivoli Personalized Services Manager iTk provisioning is, as shown in Figure 5-9 and Figure 5-28 on page 256, being used to propagate changes from the Tivoli Personalized Services Manager subscriber database into other databases that need to have the same, or a subset of, data updated in their databases. In general, the iTk provisioning provides a repertoire of elementary functions through business transactions and accessing capabilities in the TPSM database. This is implemented by a set of Java classes that allow applications to interact with the Tivoli Personalized Services Manager database. It provides programmers with tools to extend Tivoli Personalized Services Manager and build custom services utilizing a JDBC back-end allowing it to be database independent.

[4] It is possible to install the Tivoli SecureWay Policy Director LDAP on the same instance as the IBM SecureWay Directory, but they will still have two different directory tree bases.

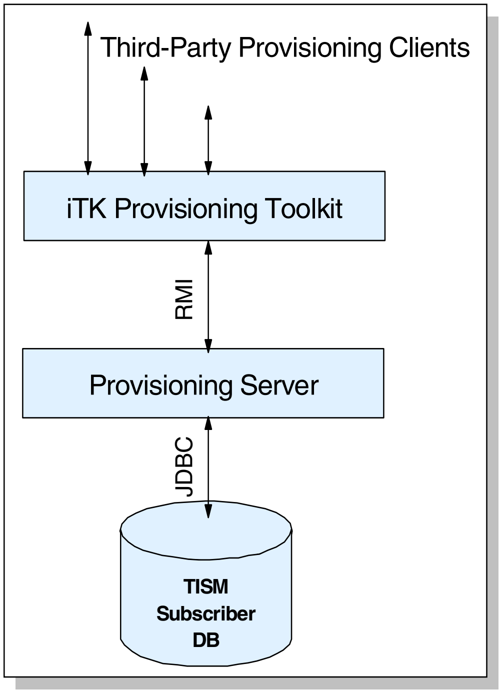


Figure 5-10 Tivoli Internet Services Manager iTk Provisioning

Applications implementing provisioning will typically develop a client application that extends the ProvisioningListener-class provided by iTk Provisioning Toolkit in one of two ways:

1. The client may automatically be invoked and receive transactions over RMI (Remote Method Invocation) from the Tivoli Personalized Services Manager business logic server *when changes occur* to the Tivoli Personalized Services Manager database.
2. Or one can implement an application *that polls* the Tivoli Personalized Services Manager transaction database (utilizing the ProvisioningTransaction-class provided by iTk Provisioning Toolkit) to check if any changes has occurred.

The first alternative is the one requiring less coding and will provide “instant” provisioning, whereas the second alternative will be less of a performance concern if you have many updates/changes over a short period of time.

In our case we will choose the first solution. The client's function will be to capture the wanted transactions, parse the needed information from the ProvisioningTransaction-object and then insert this information into their databases.

This scenario is basically how the provisioning into the Tivoli SecureWay Policy Director directory is implemented. To insert information into the Tivoli SecureWay Policy Director directory, the Tivoli SecureWay Policy Director client will typically use existing Policy Director commands such as **pdadmin**.

> **Note:** Installations with frequent updates/changes to the Tivoli Internet Services Manager database should closely monitor the performance impact of the provisioning and consider alternative approaches such as provisioning on fixed time-intervals (polling) if the performance impact of the implemented solution is significant.

5.2.3 Single sign-on and session management

To provide single sign-on within our new system, Everyplace Server has been equipped with a highly dynamic session file, and a corresponding server for tracking active user sessions. Its purpose is to recognize a session already authenticated, and to share that credibility between all Everyplace Server components.

Active Session Table is a high-speed cache for user authorization, device and network specific information. It is used by Everyplace Wireless Gateway and WebSEAL-Lite components. The Active Session Table Server provides no direct services for users or customer.

When either Everyplace Wireless Gateway or WebSEAL-Lite authenticates a new session, a unique record is created in the Active Session Table. For subsequent requests, the record will provide all necessary information about credibility, the user (customer), the device and the network used.

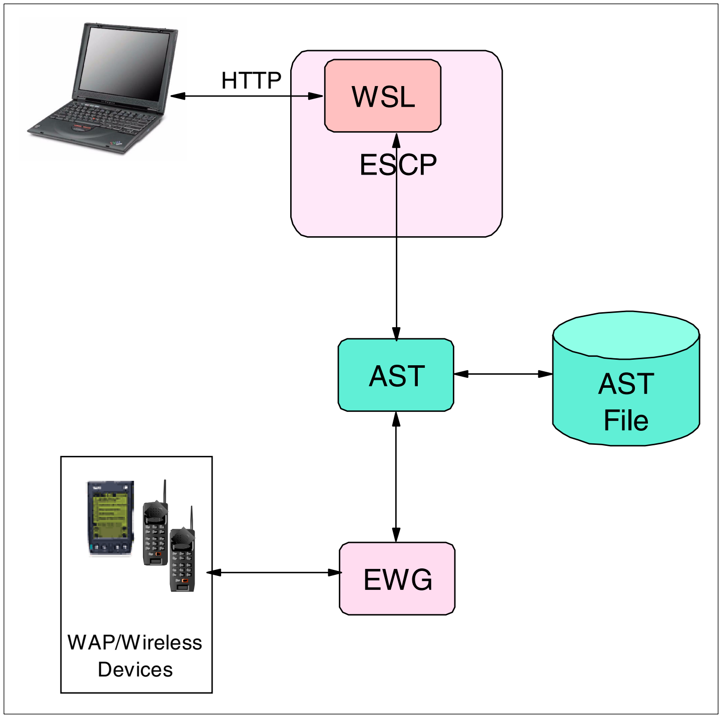


Figure 5-11 Active Session Table Server

Sessions originated from Wireless Gateway are authenticated and allocated a trusted IP address and/or HTTP header identification. When the request passes the WebSEAL-Lite, it will use that as the key for Active Session Table lookups and provide the full set of Everyplace Server headers.

Sessions originating from an HTTP/IP connection via WebSEAL-Lite will be assigned a unique session ID placed in an HTTP cookie. The ID is the key used in the Active Session Table.

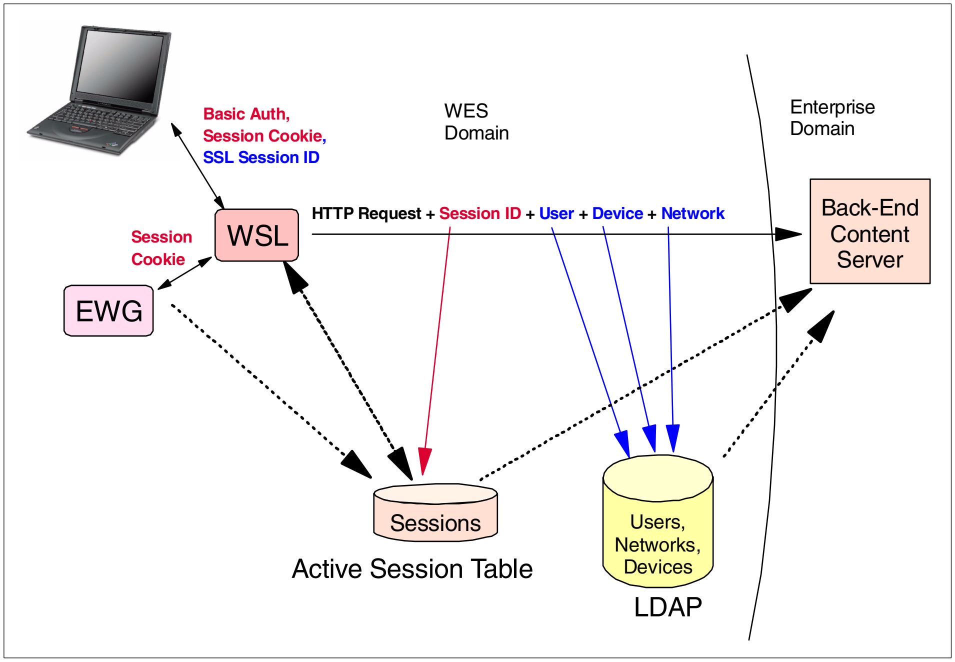


Figure 5-12 Session tracking and single-sign-on

The Active Session Table Server is highly specialized and optimized to have minimal impact on system performance:

- For each session, the Active Session Table Server maintains an index record in virtual memory, and an attribute record on disk. For planning purposes, assume that the index record will consume 100 bytes of virtual memory, and the attribute record will consume 400 bytes of virtual memory.
- Records that are inserted by the Wireless Gateway are only deleted when the client logs off the gateway, however long that takes.
- Records that are inserted byWebSEAL-Lite are deleted after they have been idle for more than `X` minutes, where `X` is defined in the configuration (10 minutes by default).
- When a record has been deleted, the space occupied by its index entry in virtual memory is freed immediately and is available for reuse. The corresponding attribute record on disk is "deallocated". This does not mean that the space is available for reuse.
- This is how the disk space is handled:
  - Each time a new entry is inserted into the Active Session Table, its associated attributes are appended to the "current" Active Session Table cache file.

- After a certain number of records have been appended, a new file is opened and it becomes the "current" Active Session Table cache file. The number of entries in each cache file are computed dynamically based on a number of factors that are too complicated to go into here.
- Each time an existing Active Session Table entry is queried or updated, its associated attribute record is "deallocated" and a new attribute record is written to the current Active Session Table cache file.
- When all of the attribute records in an Active Session Table cache file have been deallocated, it is erased.
- All Active Session Table cache files are erased when the Active Session Table Server is started and shut down.

For more information, see the sequence diagrams in 6.6.1, “Routed connections” on page 289. More information about authentication and single sign-on is available in Chapter 9, “Security” on page 359.

5.2.4 Notification services

Since all customers will have the opportunity to subscribe to various notification services enabling Star Hotels to notify special offerings to the different user-groups, a notification server needs to be deployed.

The Everyplace Server Intelligent Notification Services enables each individual customer to subscribe to the content of his/her preference.

The customer will typically configure his or her preferences over the Internet, whereas the notifications will be submitted to any of the supported devices that the user has signed up for.

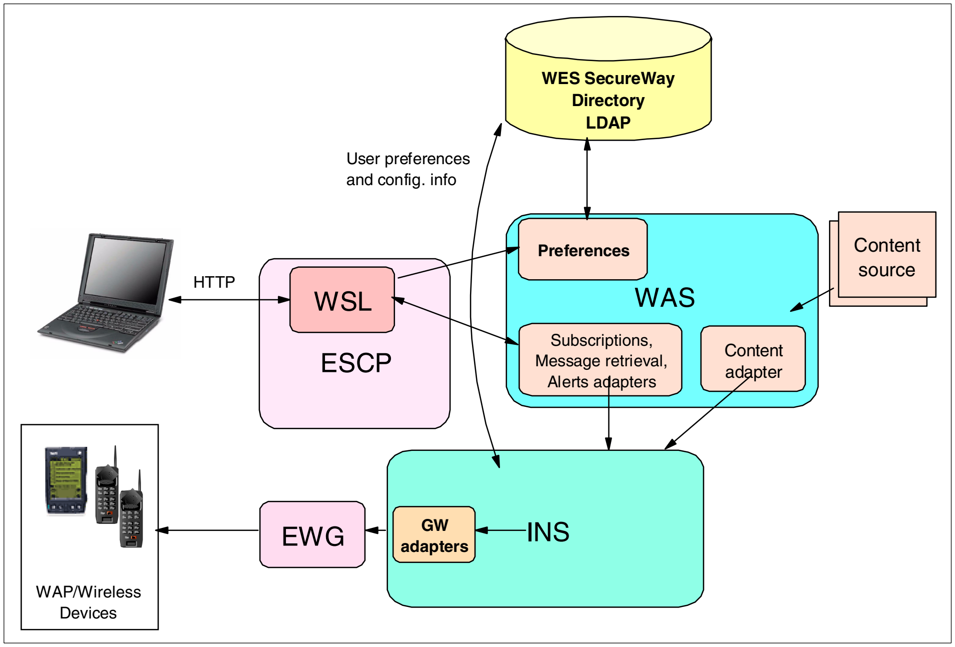
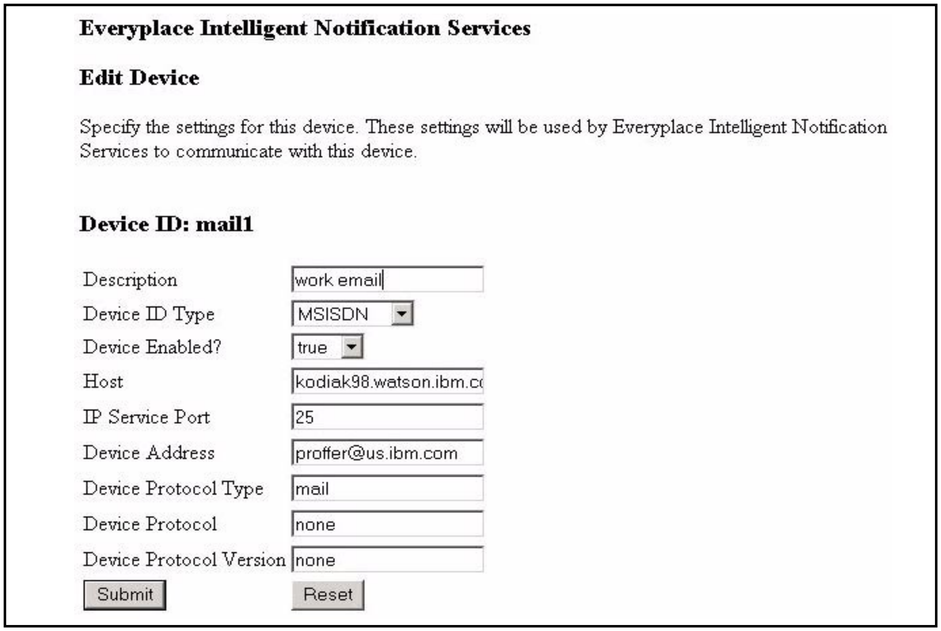


Figure 5-13 Intelligent Notification Services

The *Preferences* is a Tivoli Personalized Services Manager self-care application that supports not only Intelligent Notification Services, but also Location Based Services, Voice Services as well as general WebSphere Everyplace Server preferences. Typically the Intelligent Notification Services user will use the Preferences-application to define which devices are to receive notifications. However, it should be noted that these applications (servlets and JSPs) should be modified and simplified to enable *any* user to be able to fill in the fields and that basically implies to only prompt the user for a mail-address, not all the information that is in the default window:



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.cc
IP Service Port	25
Device Address	proffer@us.ibm.com
Device Protocol Type	mail
Device Protocol	none
Device Protocol Version	none

Figure 5-14 Intelligent Notification Services user preferences - Device example

The selected preferences are stored in the SecureWay Directory. In this solution all the preference settings are carried out over the HTTP, whereas the notifications are being dispatched by Intelligent Notification Services over the Everyplace Wireless Gateway as WAP (push), SMTP, Lotus Sametime or SMS messages.

Intelligent Notification Services provides an API to develop adapters to add subscriptions, publish content, define triggers, retrieve persistent content and send messages. Although sample code is available in installation directories, integrators need to adapt and customize these adapters to facilitate the requirements of the customer.

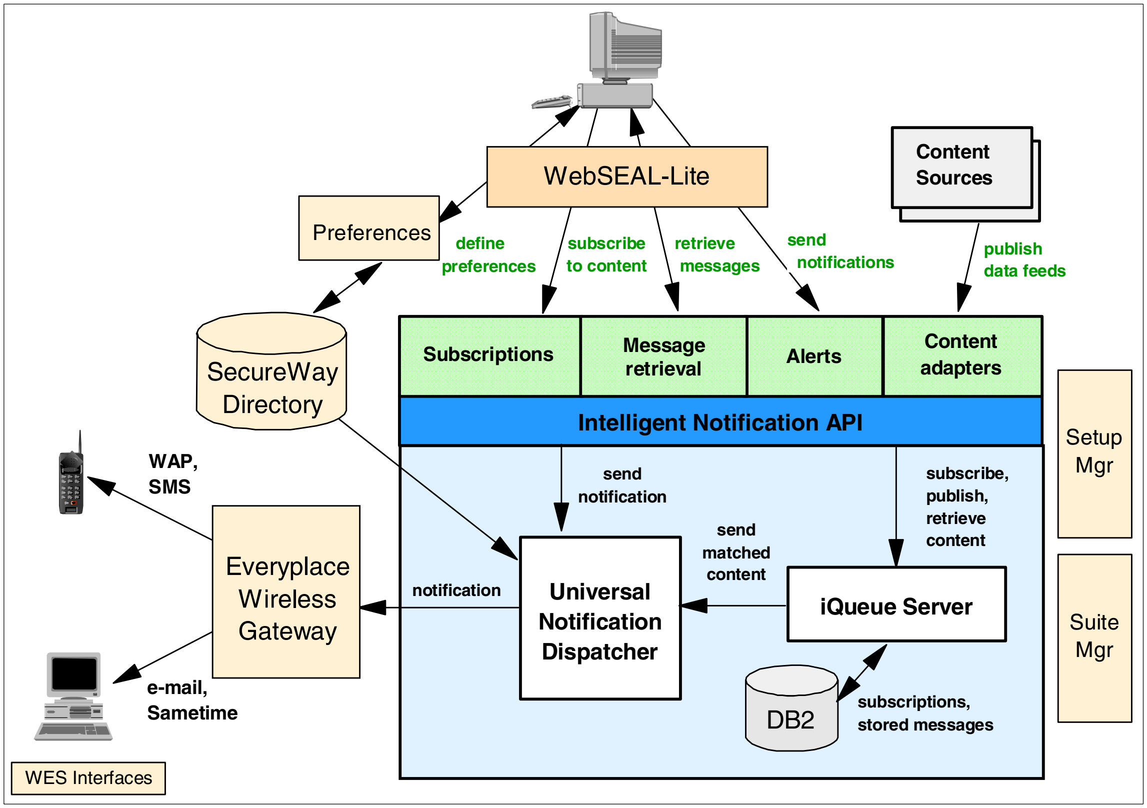
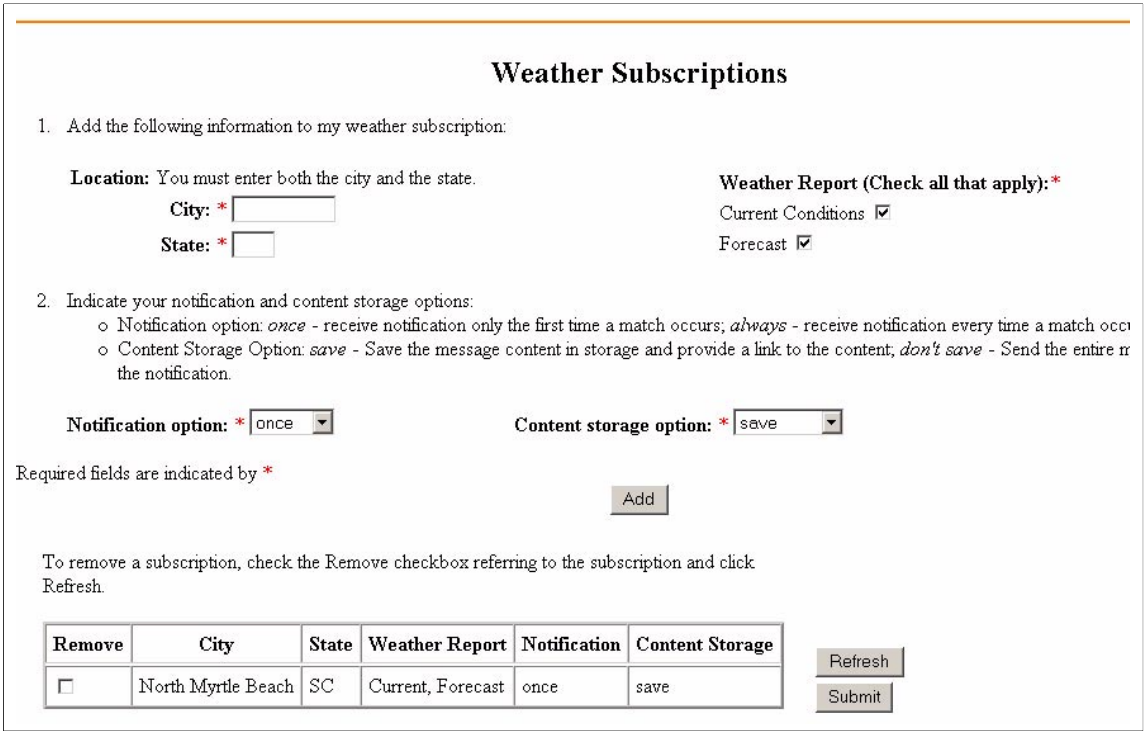


Figure 5-15 Intelligent Notification Services Functional overview

Once preferences are established, users should be provided with a content subscription interface. Sample code in terms of servlets and JSPs provided. One such sample is the *Weather Subscription* content subscription servlet and JSP that produces XML-based data transcoded with a sample XML2HTML stylesheet. Figure 5-16 shows the sample Weather Subscription window:



Weather Subscriptions

1. Add the following information to my weather subscription:

Location: You must enter both the city and the state.

City: *

State: *

Weather Report (Check all that apply): *

Current Conditions ☑

Forecast ☑

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 occ
   - Content Storage Option: *save* - Save the message content in storage and provide a link to the content; *don't save* - Send the entire m the notification.

Notification option: * once

Content storage option: * save

Required fields are indicated by *

To remove a subscription, check the Remove checkbox referring to the subscription and click Refresh.

Remove	City	State	Weather Report	Notification	Content Storage
□	North Myrtle Beach	SC	Current, Forecast	once	save

Figure 5-16 The sample Weather Subscription window

Similarly the integrators will find sample Java code in the Everyplace Server installation directories to implement functions to add a subscription, publish content, define triggers, retrieve persistent content and send messages. These samples (NewsSubscriptionServlet, XMLContentAdapter, NewsHandler) all demonstrate how to use the Intelligent Notification Services API and their Java methods.

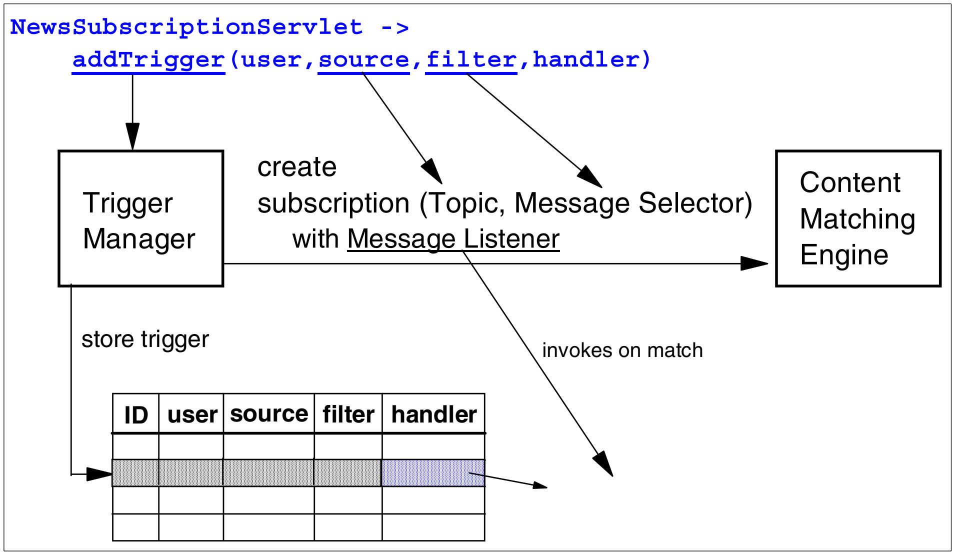
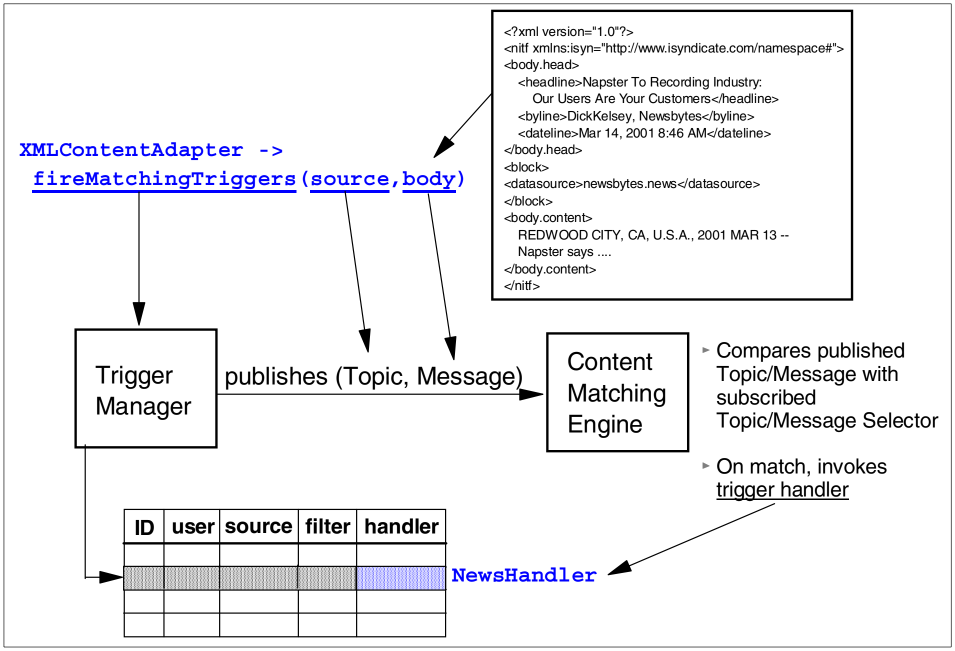

NewsSubscriptionServlet ->
addTrigger(user,source,filter,handler)
Trigger Manager
create subscription (Topic, Message Selector) with Message Listener
Content Matching Engine
store trigger
invokes on match
ID
user
source
filter
handler

Figure 5-17 Add a subscription

The XMLContentAdapter is the sample content handler included in WebSphere Everyplace Server. The Java class will parse the content and invoke the trigger handler whenever a subscription match occurs:



```
<?xml version="1.0"?>
<nitf xmlns:isyn="http://www.isyndicate.com/namespace#">
<body.head>
   <headline>Napster To Recording Industry:
      Our Users Are Your Customers</headline>
   <byline>DickKelsey, Newsbytes</byline>
   <dateline>Mar 14, 2001 8:46 AM</dateline>
</body.head>
<block>
<datasource>newsbytes.news</datasource>
</block>
<body.content>
   REDWOOD CITY, CA, U.S.A., 2001 MAR 13 --
   Napster says ....
</body.content>
</nitf>
```

ID	user	source	filter	handler

XMLContentAdapter ->
fireMatchingTriggers(source,body)
Trigger Manager
publishes (Topic, Message)
Content Matching Engine
Compares published Topic/Message with subscribed Topic/Message Selector
On match, invokes trigger handler
NewsHandler

Figure 5-18 Publish content

When the subscription match occurs, the NewsHandler is invoked to make sure the message is being dispatched to the Universal Notification Dispatcher (UND) and out to the subscriber through the gateway adapters (see Figure 5-22 on page 249).

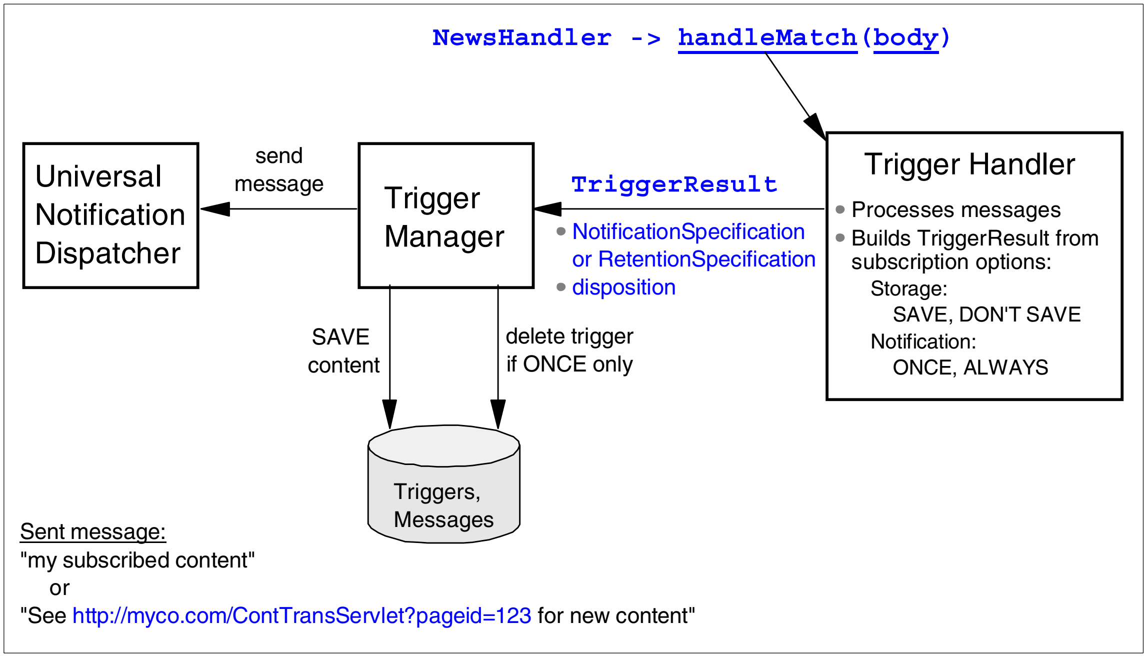

NewsHandler -> handleMatch(body)
Universal Notification Dispatcher
send message
Trigger Manager
TriggerResult
NotificationSpecification or RetentionSpecification
disposition
Trigger Handler
Processes messages
Builds TriggerResult from subscription options:
Storage:
SAVE, DON'T SAVE
Notification:
ONCE, ALWAYS
SAVE content
delete trigger if ONCE only
Triggers, Messages
Sent message:
"my subscribed content"
or
"See http://myco.com/ContTransServlet?pageid=123 for new content"

Figure 5-19 Define trigger handlers

A persistent message (or content) is a message, due to its size or content, that is not sent to the subscribed notification device. Normally the notification device will only receive the URL of the complete message stored in the Intelligent Notification Services message database (DB2). The subscriber may read the message in full by using a browser.

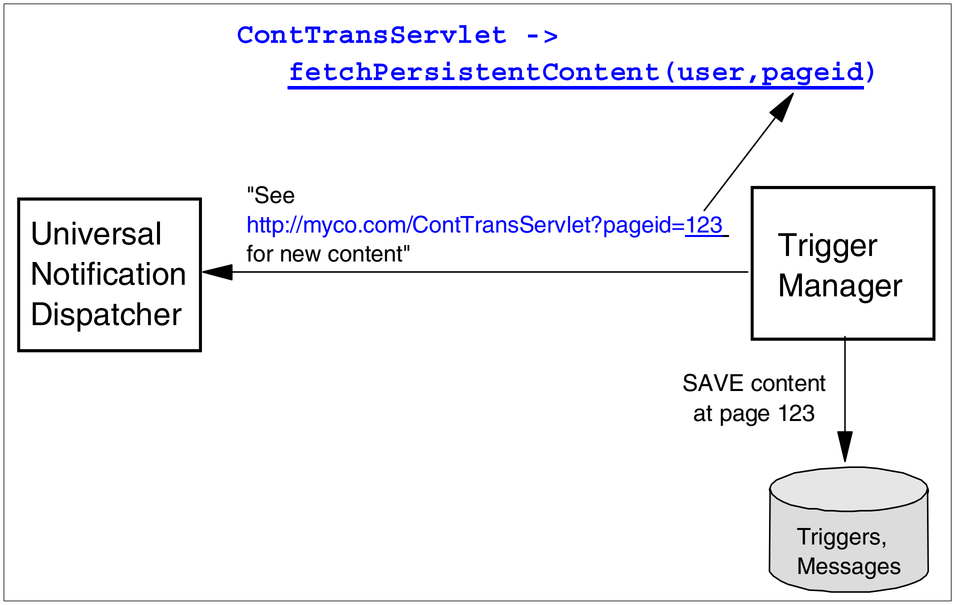
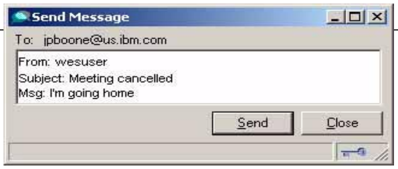

ContTransServlet ->
fetchPersistentContent(user,pageid)
"See http://myco.com/ContTransServlet?pageid=123 for new content"
Universal Notification Dispatcher
Trigger Manager
SAVE content at page 123
Triggers, Messages

Figure 5-20 Retrieval of persistent content

There is no automatic deletion of persistent messages. The integrator should handle this when the message is read or on the basis of the message age (expiration).

All messages created must be in NotificationML format. This format is a standardized XML format that is device independent. Intelligent Notification Services will apply one of its gateway adapters to transcode the file into the device-specific format.

```
// create a client interface to Universal Notification Dispatcher
      NotificationService ns = new NotificationService("und.myloc.myco.com",7005);
// specify delivery options
      DeliveryOptions opts = new DeliveryOptions();
      opts.devices = opts.IM;              // send message to IM device
      opts.priority  = opts.NORMAL;        // set message priority to NORMAL
      opts.multiDevices = opts.ALL;        // send message to all devices of recipient
// create message in NotificationML format
      String message = "<?xml version=\"1.0\" encoding=\"UTF-8\"?><message>"
                          + "<to>jpboone@us.ibm.com</to>"
                          + "<from>wesuser</from>"
                          + "<subject>Meeting cancelled</subject>"
                          + "<text> I'm going home</text></message>";
// send the message with the specified delivery options
       ns.sendMessage(message,opts);
```



To: jpboone@us.ibm.com

From: wesuser
Subject: Meeting cancelled
Msg: I'm going home

Figure 5-21 Send messages

Gateway adapters

Four customizable notification gateway adapters are also installed with Intelligent Notification Services. They are the SMS, WAP, e-mail and IMS-gateway adapters. All these adapters contains a customizable transcoder that uses a transcoder stylesheet (XSL) to transcode the NotificationML XML files into the device-specific format:

- WAPTranscoder_SS.xsl
- SMSTranscoder_SS.xsl
- EMAILTranscoder_SS.xsl
- IMSTTranscoder_SS.xsl

The IMSTTranscoder and its IMSTTranscoder_SS.xsl stylesheet is used to transcode into *Instant Messaging Service* format that is being used by Sametime.

Shown in Figure 5-22 is the WAP Gateway adapter containing a customizable transcoder and its transcoder stylesheet.

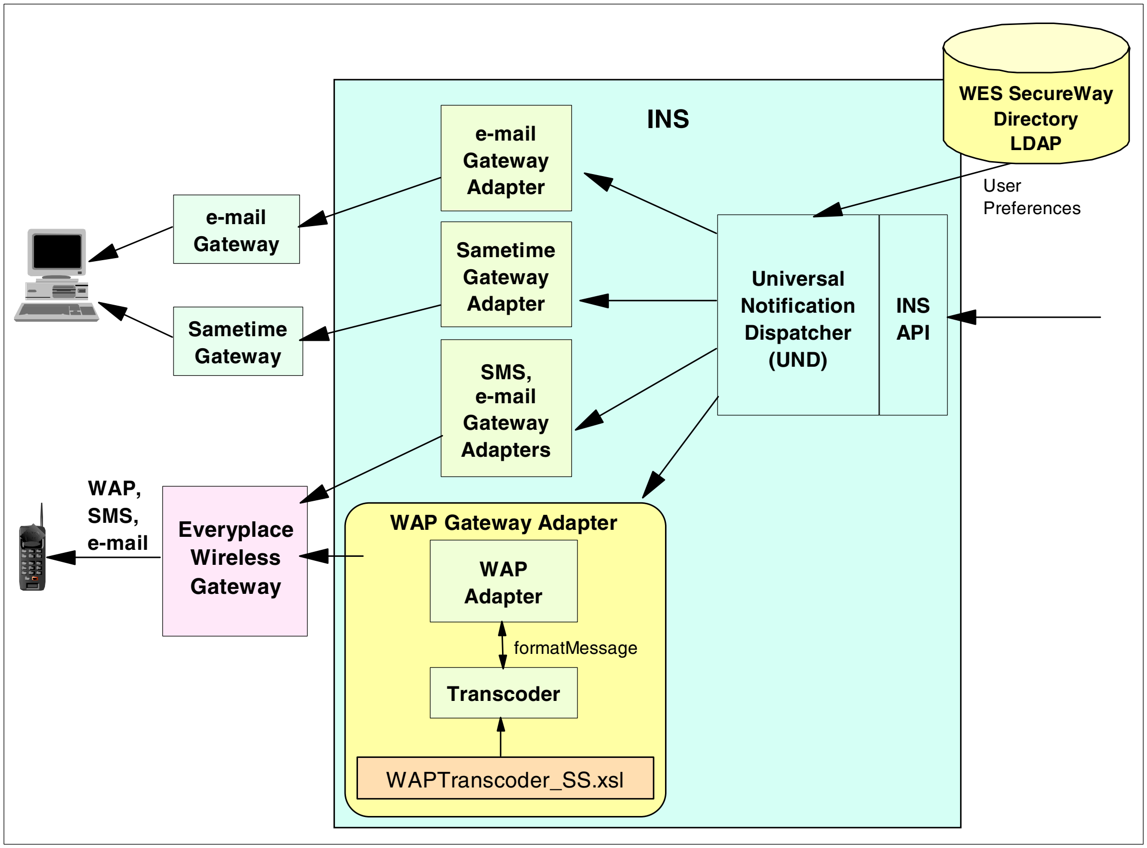
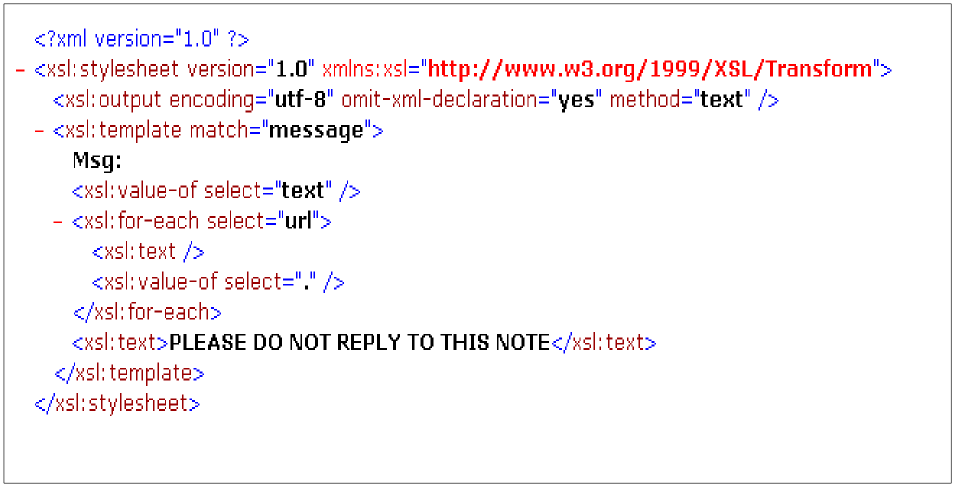


Figure 5-22 Gateway adapters

The “send a message” API expects all messages to be in the XML NotificationML-format. This makes the programming device-independent since the transcoders will apply appropriate device-dependent stylesheets to transcode the messages to the device-specific format.

The customizable stylesheets are fairly simple stylesheets.



```
  <?xml version="1.0" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output encoding="utf-8" omit-xml-declaration="yes" method="text" />
  - <xsl:template match="message">
      Msg:
      <xsl:value-of select="text" />
    - <xsl:for-each select="url">
        <xsl:text />
        <xsl:value-of select="." />
      </xsl:for-each>
      <xsl:text>PLEASE DO NOT REPLY TO THIS NOTE</xsl:text>
    </xsl:template>
  </xsl:stylesheet>
```

Figure 5-23 Sample EMAILTranscoder_SS.xsl stylesheet

When the Wireless Gateway is configured to be a messaging gateway, it enables any application, such as Intelligent Notification Services, to send messages to a client, such as a pager or a phone (SMS) in a wireless network.

In a WAP network, the message gateway is a WAP push proxy gateway:

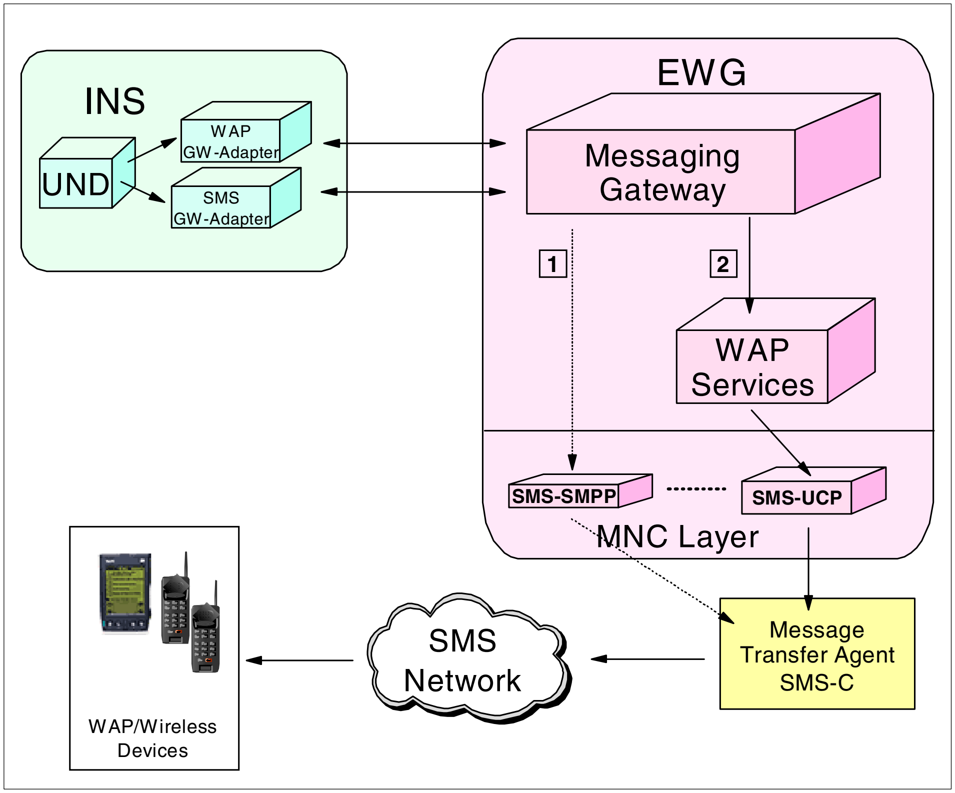


Figure 5-24 Wireless Gateway push proxy gateway

All non-WAP push messages are going through [1], whereas the WAP push messages are going through [2] and the WAP Services module that processes the push messages and sends those messages through the Mobile Network Connection (MNC) layer and its TCP/IP or X.25-based SMS-UCP protocol across a WAP bearer to a WAP-enabled client.

Non-WAP messages will use the SMS peer-to-peer protocol SMS-SMPP that also connects over TCP/IP or X.25.

5.2.5 Location Based Services

Location Based Services are being deployed enabling Star Hotels to offer services such as where to find the nearest hotel with detailed map directions as well as information on the nearest mall, airport, etc.

All enrolled customers may subscribe to the location-based services. By deploying this service Star Hotels adds value to the customer since it will be simple and effective for the travelling customer who needs to find his way around.

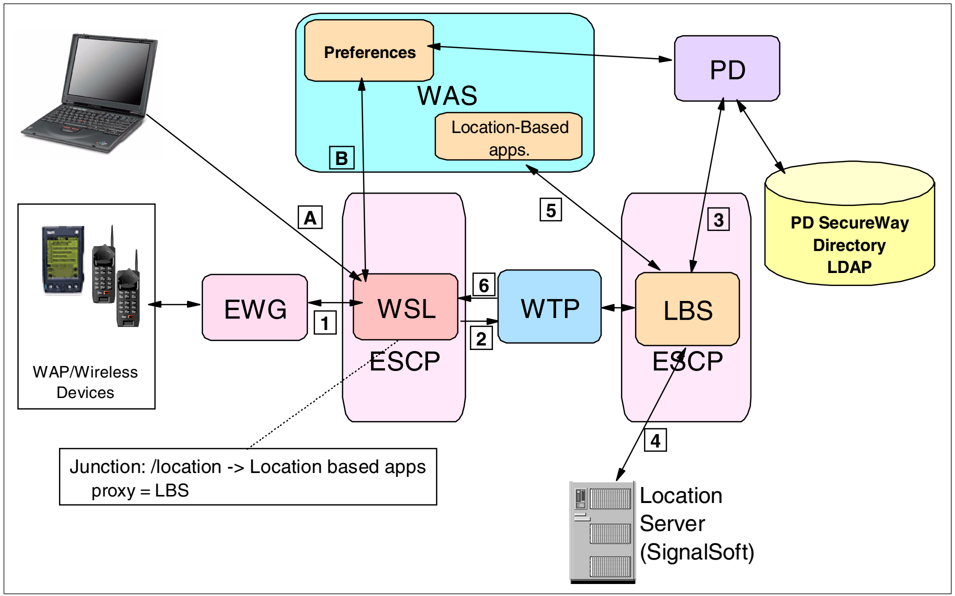

Preferences
WAS
Location-Based apps.
B
PD
PD SecureWay Directory LDAP
A
5
3
WAP/Wireless Devices
EWG
1
WSL
6
2
WTP
LBS
ESCP
ESCP
4
Junction: /location -> Location based apps
proxy = LBS
Location Server (SignalSoft)

Figure 5-25 locatIon-based services

Location Based Services gets its location information from a location server (currently SignalSoft is supported[5]) in an XML-based format. Basically the location server is given a Mobile Identification Number/MSISDN (cell phone number), and returns the current location of the cell phone in an XML file.

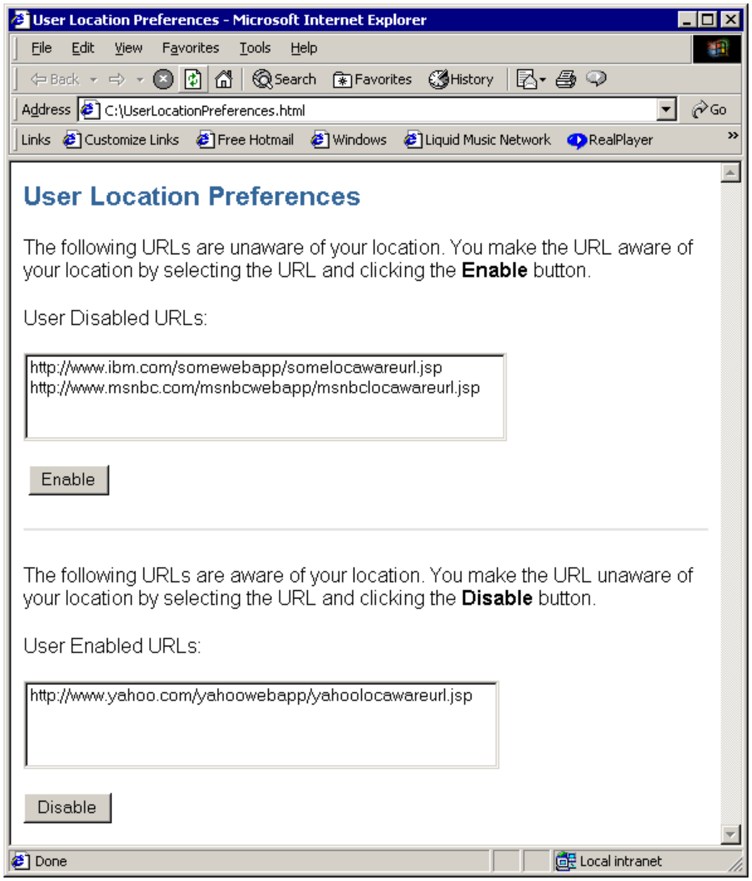
1. In detail, the Wireless Gateway will receive a request for an application, converts this WAP request to an HTTP request, inserts the MSISDN into an HTTP header and forwards it to WebSEAL-Lite.
2. WebSEAL-Lite identifies, by its configured *junction* directive, this request to be a location-based request by its “/location”-portion of the URL and forwards the request to the Location Based Services through the WebSphere Transcoding Publisher.
3. Location Based Services checks with Policy Director to see if the requested application is a location-based application and if the user may access this

[5] Accessing a location server implies involving a carrier business model, since the carriers supply the source of the location data. Various business models, depending on region and country, may use vendors other than SignalSoft, such as Ericsson, Motorola and Nokia.

application as well as if the user has given the application the permission to provide information (privacy).

4. If so, Location Based Services requests the user's location from the location server and receives a TCP XML response with the location information.
5. Location Based Services converts the XML file to Geography Markup Language (GML) and passes it in an HTTP header along with the request to the location-based application. The location-based application needs to be developed by the hotel. The business logic of the application should provide information such as where is the hotel nearest to where the user is currently located (according to the location info in the request header). Possible additional information could be the location of the nearest airport, mall, etc.
6. The view of the application should be a generalized XML file that will be passed back to the requesting client through the WebSphere Transcoding Publisher, this time for transcoding into a device-specific format.

As for Intelligent Notification Services, the user preference for the Location Based Services is set up by accessing the Tivoli Personalized Services Manager Preference application, [A] and [B] in Figure 5-25 but in the case of the Location Based Services preferences, the enabled URLs are stored in the Policy Director directory:


User Location Preferences - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address C:\UserLocationPreferences.html Go
Links Customize Links Free Hotmail Windows Liquid Music Network RealPlayer
Done
Local intranet

User Location Preferences

The following URLs are unaware of your location. You make the URL aware of your location by selecting the URL and clicking the **Enable** button.

User Disabled URLs:

http://www.ibm.com/somewebapp/somelocawareurl.jsp
http://www.msnbc.com/msnbcwebapp/msnbclocawareurl.jsp

Enable

The following URLs are aware of your location. You make the URL unaware of your location by selecting the URL and clicking the **Disable** button.

User Enabled URLs:

http://www.yahoo.com/yahoowebapp/yahoolocawareurl.jsp

Disable

Figure 5-26 Location Based Services Preferences

In this window the user enables those applications for which the user would like to receive location-based information, and disables those applications that he does not want to enable for location-based information.

Location privacy is an important issue for this service to make sure no information that has not been agreed upon by the user is being passed on to any third party. Policy Director is used to store the user privacy information. Policy Director is also managing the application authorization, since a requester may not necessarily be granted access to all applications/services. This administration is carried out using the Policy Director Administration Console.

In summary, the Policy Director is being queried twice:

1. Location Based Services will check to see if the requested URL is a registered location-based service in the Policy Director.

2. Location Based Services will check if the requesting user is authorized to access the requested URL and if the user has given the application the permission to send its info (privacy):

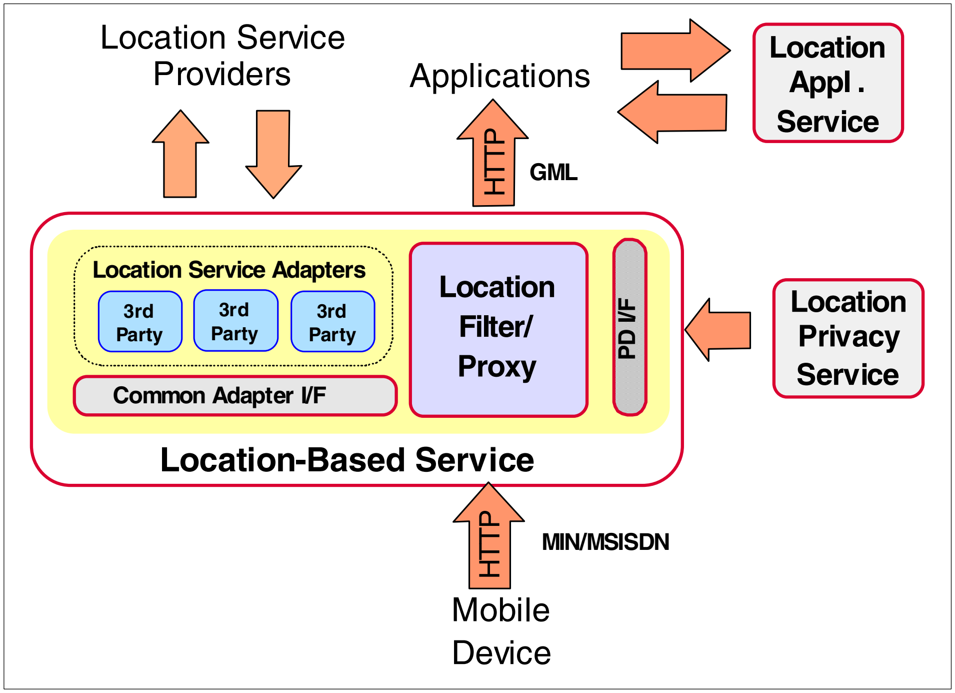


Figure 5-27 Architecture

The Location Based Services is a plug-in to the Caching Proxy and acts as a forward proxy, that is, all traffic coming through the WebSEAL-Lite will also go through the Location Based Services unless a *junction* is configured in WebSEAL-Lite.

Users request location-based services by selecting location-based applications and their URLs. When any of these URLs are requested by the user, the Location Based Services will request the location information from the location server, insert into an HTTP header and act as a proxy passing on the request to the appropriate application (URL).

5.2.6 The portal

The application portal will be accessed by all users, for example customers and employees. The WebSphere Portal Server integrates with the Everyplace Server to enable the single sign-on capability of WebSphere Everyplace Server. It is “just another” application behind the WebSphere Everyplace Server infrastructure.

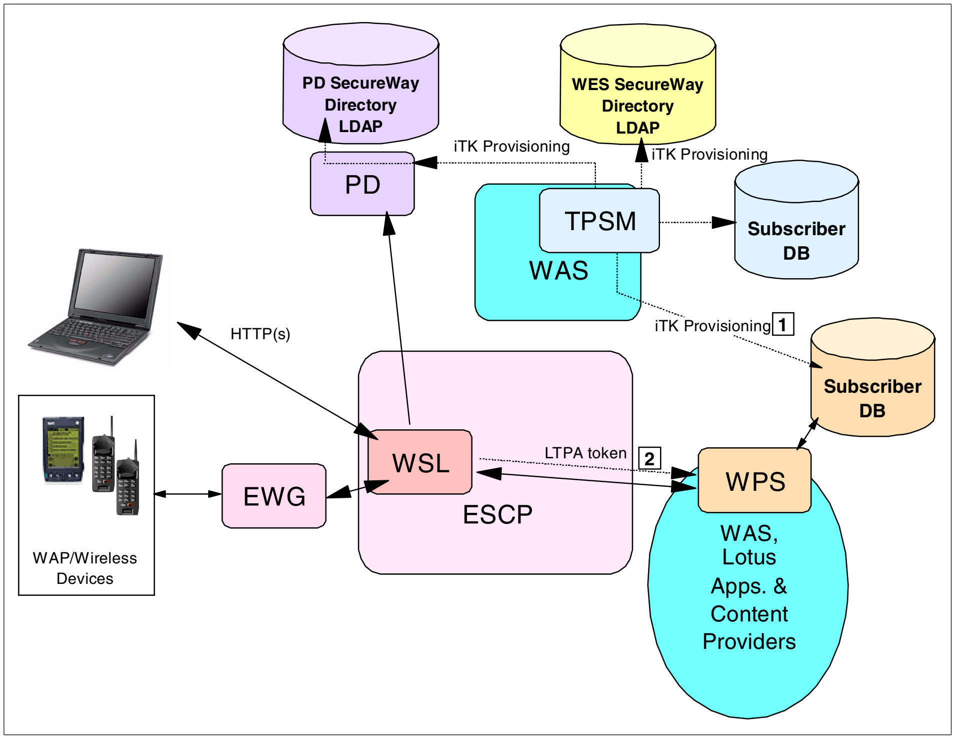


Figure 5-28 Portal Server in a Everyplace Server environment

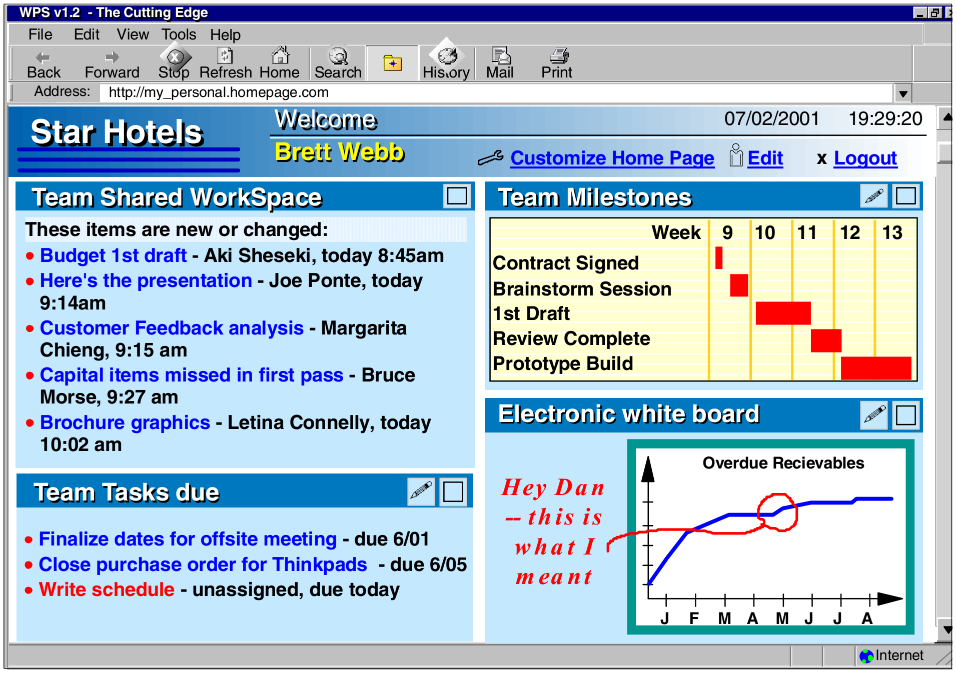
The WebSphere Portal Server has transcoding aggregators to provide translations to language-specific pages (WML or HTML). The WML aggregator does not differentiate between various WAP devices. This can be accomplished with the WebSphere Transcoding Publisher as a proxy in front of WebSphere Portal Server. Such a configuration will improve the flexibility of transcoding into any arbitrary devices to which we can provide a stylesheet (XSL), but it is currently not required by Star Hotels.

1. The WebSphere Portal Server also has its own subscriber database. This database integrates with the centralized Everyplace Server repository controlled by Tivoli Personalized Services Manager by using the iTk

provisioning tool provided with Tivoli Personalized Services Manager (see "Tivoli Personalization Services Manager iTk provisioning" on page 234).

2. The single sign-on integration of Everyplace Server and WebSphere Portal Server is accomplished via WebSEAL-Lite, which can be configured to build a Lightweight Third Party Authentication (LTPA) token that the WebSphere Application Server, running WebSphere Portal Server, understands. LTPA tokens are what WebSphere Application Server normally builds and passes to browsers as cookies in order to maintain a users identity over the course of a session. As WebSphere Application Server uses the LTPA token to build its security context, WebSphere Portal Server can use that security context.

The portal view is highly customizable and should include the new applications as portlets and adopt a look and feel that profiles the hotel:


WPS v1.2 - The Cutting Edge
File Edit View Tools Help
Back Forward Stop Refresh Home Search History Mail Print
Address: http://my_personal.homepage.com
Internet

Star Hotels

Welcome 07/02/2001 19:29:20

Brett Webb

Customize Home Page Edit x Logout

Team Shared WorkSpace

These items are new or changed:

- Budget 1st draft - Aki Sheseki, today 8:45am
- Here's the presentation - Joe Ponte, today 9:14am
- Customer Feedback analysis - Margarita Chieng, 9:15 am
- Capital items missed in first pass - Bruce Morse, 9:27 am
- Brochure graphics - Letina Connelly, today 10:02 am

Team Tasks due

- Finalize dates for offsite meeting - due 6/01
- Close purchase order for Thinkpads - due 6/05
- Write schedule - unassigned, due today

Team Milestones

Electronic white board

Figure 5-29 Sample customized portal-view

This sample customized portal window shows the integration of the Lotus QuickPlace application as one of the many applications Star Hotels will integrate in their portal.

5.2.7 Working offline: Device Manager

As one of the services requested by Star Hotels, one service provided to customers and hotel employees is to be able to connect to the hotel's intranet from their portable devices over the Internet through a secure channel. Also users will have the facility to work with wireless devices as PalmPilot, Pocket PCs and laptops offline and submit their work later on. Their customers will get connected to the server and submit their jobs and download new jobs or messages accumulated while the customer was offline. This service may be facilitated by usage of a wireless client on each of the portable devices. To maintain this client software on each portable device, we will use the Device Manager Server, which enables each user to enroll its device and to subscribe it to client software updates, etc.

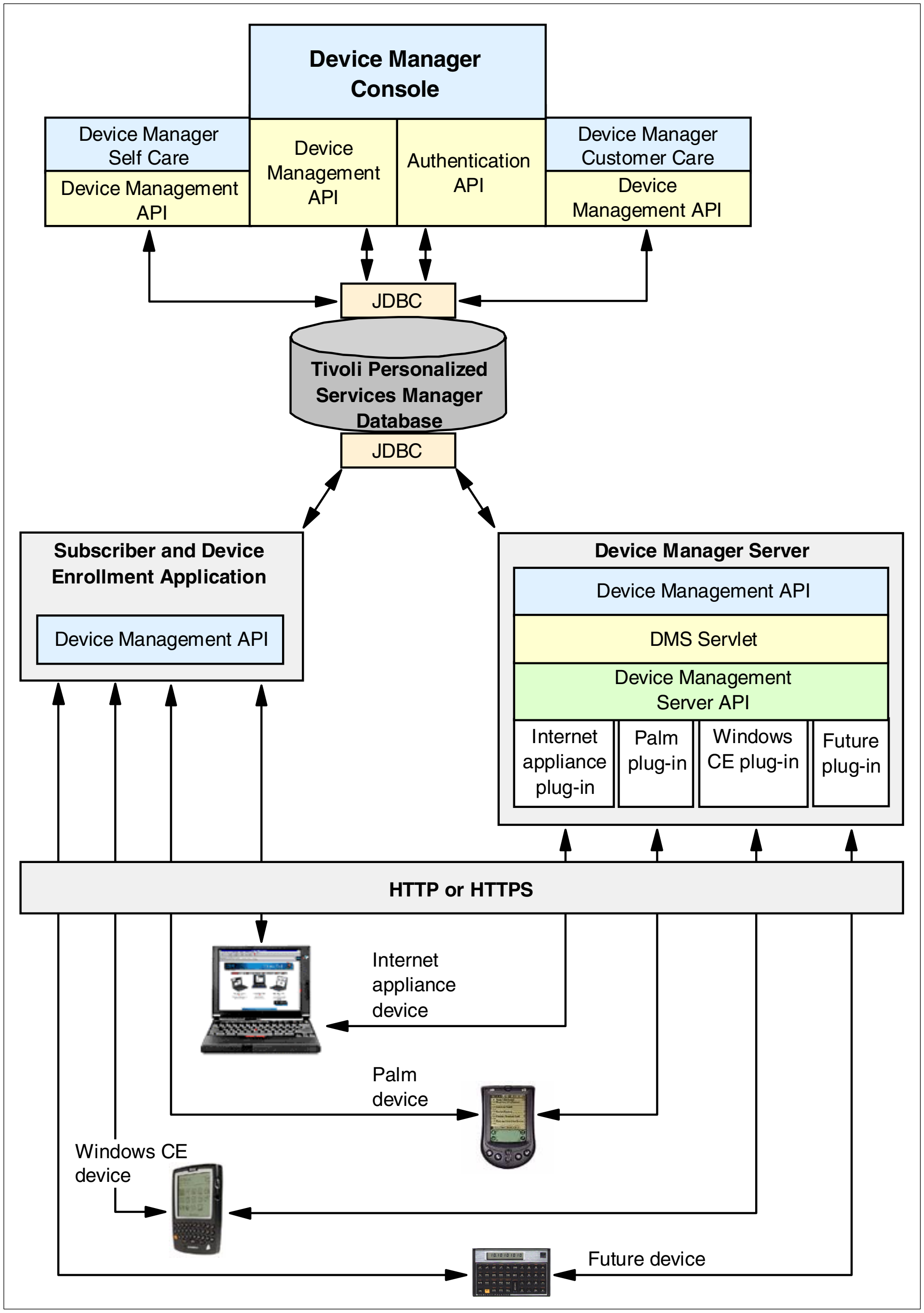


Figure 5-30 Device Manager Server architecture

The Device Manager Server extends the capabilities of Tivoli Personalized Services Manager to include management of devices and their related resource, such as device software. The Device Manager Server provides a flexible framework and a set of services for managing a customer's device or an employee's device. Also Star Hotels can extends the system's framework and use it to enable their devices to be managed by the Tivoli Personalized Services Manager.

The Device Manager Server environment, as illustrated in Figure 5-30 on page 259, shows the Tivoli Personalized Services Manager Self Enrollment Devices, with which users will enroll their own devices.

The Device Manager Server components and their use in our solution will be the following:

- **Device Manager Server:** this is the main component of the Device Manager Server. The Device Manager Server will receive the request from the wireless clients and will process them.
- **Application Programming Interfaces:** This component will define the programming interface between the Device Manager Server, administration clients, or external applications, and the device-related data resource stored in the Device Manager Server database.
- **Tivoli Personalized Services Manager Database:** This is the repository for all device management information in the system. It is accessed using the device management API. The database will contain tables of entries that describe devices and device-related data sources.
- **Device Manager Console:** The Device Manager Console is a graphical user interface for administrating device management operations form a Microsoft Windows client. Administrators use this interface to control the Device Manager Server system.

Device Manager Services inside the scenario

The Device Manager Services includes two components working together to process all jobs submitted into Star Hotels’ system:

- **Devices Manager Server Servlet:** This will ensure, when a device connects to the service provider network, that the device is enrolled, and if it is, will coordinate the processing of all schedule jobs for the device. If is not enrolled, the Device Manager Server will redirect the device to the enrollment application for registration.
- **Device Plug-Ins:** This will provide the logic that handles device identification, communication, job processing, and high-level management task for a particular device class. This logic includes job classes for job types such as device configuration, software distribution, and rest page management.

The main role of the Device Manager Servers will be processing jobs for devices. The unit of work that users will work with is a job. The types of jobs that customers could work with and that are included in the Device Manager Server will be:

- Device configuration: New configurations for software installed in the devices that the system offers to customers so they can receive better services.
- Software distribution: Pick target devices, pick job type of Software Distribution, Specify attributes such as priority or privacy, pick software package, etc.
- Rest page management
- There could be different kinds of internal client applications to be used by Star Hotels' employees to do their work offline.
- Customers could be interested in enrolling themselves into a conference so they have to download special client products such as Lotus Sametime or QuickPlace.

A new job gets submitted for processing by one of the following:

- A hotel administrator on a client computer using the Device Manager Server console.
- A customer services representative (CSR) of Star Hotels using the Customer Care application.
- A customer of Star Hotels using the Self Care application.
- A customer or employee's external application using the device management API. The client application must be prepared to use this API. Customers and hotel employees will work with this client application that later on will submit their new jobs.

The Device Manager Server servlet and device plug-ins work together to process a request. When a device connects to the services provider's network, it is routed to the Device Manager Server for processing. The Device Manager Server servlet ensures that the device is enrolled with the service provider. For a device requiring enrollment (not enrolled), the Device Manager Server servlet redirects the device to the service provider's enrollment server for registration, before any job processing can occur.

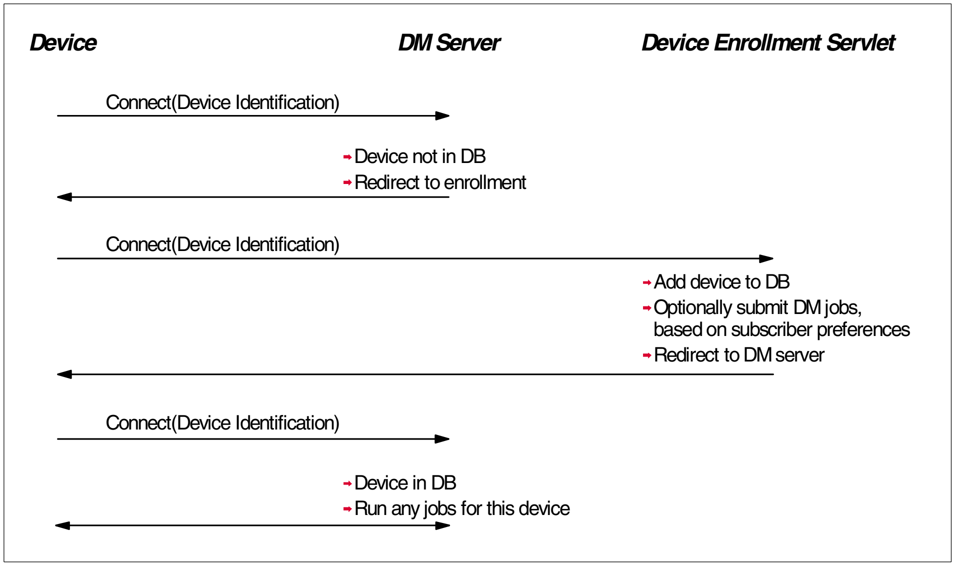

Device
DM Server
Device Enrollment Servlet
Connect(Device Identification)
Device not in DB
Redirect to enrollment
Connect(Device Identification)
Add device to DB
Optionally submit DM jobs, based on subscriber preferences
Redirect to DM server
Connect(Device Identification)
Device in DB
Run any jobs for this device

Figure 5-31 Device Manager Server request process

For an enrolled device, the Device Manager Server servlet search the database for any submitted jobs pertaining to the device and builds a prioritized job list. To turn jobs, the Device Manager Server servlet uses the device plug-in to interact with the device, often through several iterations of request and response. The Device Manager Server servlet determines what jobs are scheduled for a device and obtains information about any scheduled or current job.

Device Manager API

The device management API defines the programming interface between Device Manager Server servers, administration clients, or external applications, and the device-related data resources stored in the Device Manager database.

Authentication API

The authentication API defines the programming interface between the Device Manager console and the authenticating component. The authentication API is used when developing a customized authentication approach to replace the built-in Tivoli Personalized Services Manager authentication for validating Device Manager console administrators.

Using this API, the service provider creates a customized authentication approach and then changes from the built-in authentication to the customized one. The service provider can switch between customized and built-in authentication, but cannot use both together.

Security in Device Manager Server

Because Internet-based services are distributed across so many physical and logical locations, security for an environment such as the one supported by Tivoli Personalized Services Manager must address the full range of security needs.

This security needs to include:

- **Subscriber security:** protecting the ISP environment from the subscriber and securing subscriber data guaranteed though such mechanisms as:
  - The RADIUS security server for remote access devices
  - The HTTPS protocol (secure HTTP) for encryption of subscriber login and personal information
  - The authentication server and authentication checker
- **Single sign-on security:** enabling a single point of entry into the Tivoli Personalized Services Manager servers, The IBM SecureWay Dispatcher, and the Device Manager Server.
- **Customer autonomy security:** securing one customer's data from incursion by another customer through the use of realms.
- **Administrator and CSR login security:** ensuring that the person logged in as a Tivoli Personalized Services Manager administrator or customer services representative is authorized to perform this function. This includes administrators and CSRs who are authenticated to the console.
- **Database security:** preventing unauthorized access to the information stored in the Tivoli Personalized Services Manager subscriber and business catalog database.
- **LDAP security:** for Everyplace Server users, enabled through Tivoli SecureWay Directory.
- **Tivoli SecureWay Policy Director-Based security:** to provide restricted access to critical data.

Tivoli Personalize Service Manager Database

The Tivoli Personalized Services Manager Database is the repository for all device management information. It is implemented in a relational database and accessed using the device management API. The database contain tables of entries that describe devices and device-related data resources.

Self Care

Subscriber Self Care is a suite of applications that allow customers to view and change personal information collected during enrollment. These applications let a subscriber do the following:

- Change password
- Display update personal information such as name, last name, phone, etc.
- Display and update payment information
- Add, delete, enable or disable additional subscribers
- Manage devices that the member has enrolled using Tivoli Personalized Services Manager
- Change an order

The function will be presented to the customer in HTML pages customized by the Star Hotels.

Customer Care

The Customer Care component is used by CSRs to provide management of subscribers' accounts and device information. The purpose of the Customer Care component is to provide simple "easy to use" user interfaces for CSRs to create and update customer data while customers are on the phone. The employees will access customer data through an internal application that will let them manipulate customer information.

The Customer Care component can be used to:

- Add new customers into the system
- Search for a customer's profile and account information
- View and update customer information
- Change or cancel orders for a customer
- Add business organizations and their department structure into the system

The tasks implemented in the Self Care component will be a subset of the Customer Care component.

5.2.8 Voice services

Star Hotels wants their customers to have access to the system by voice. This means that users of the system will have access to applications in which the input and/or output are spoken, rather than a graphical user interface. Users will be able to access the deployed applications anytime, anywhere, from any telephony-capable device, and applications must be restricted to authorized users only.

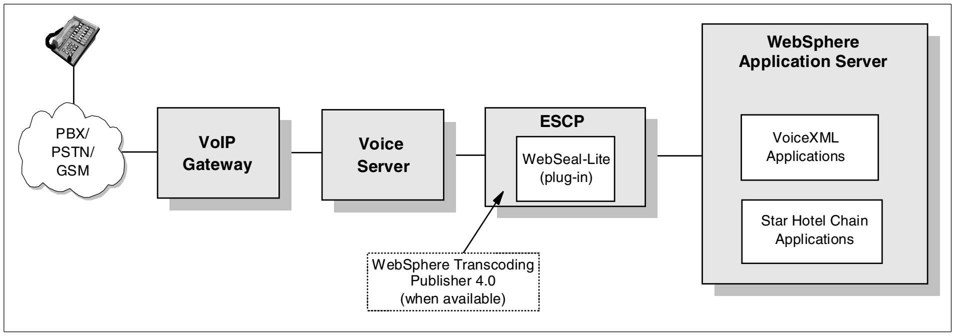
Star Hotels wants a voice system that should provide an easy and novel way for users to surf or shop on the Internet “browsing by voice”, or find places such as the closest hotel building, etc. Users should be able to interact with Web-based data (that is, data available via Web-style architecture such as servlets, ASPs, JSPs, JavaBeans, CGI scripts, and others) using speech rather than a keyboard and a mouse.

Star Hotels wants to develop two different types of categories:

- Queries: where a customer calls into the system to retrieve information from a Web-based infrastructure.
- Transactions: where a customer calls into a system to execute specific transactions with a WAP-based back-end.

This system will be mounted with the Everyplace Server Voice Server. This component will translate and interpret customer voice commands and create the voice responses for customers once his/her request have been processed.

The Voice Server is a WebSphere Application Server product running on a server class hardware system. The Server operates in conjunction with a Voice over IP (VoIP) gateway and existing Web infrastructure to allow delivery of Web-base voice applications written in VoiceXML.



PBX/ PSTN/ GSM
VoIP Gateway
Voice Server
ESCP
WebSeal-Lite (plug-in)
WebSphere Transcoding Publisher 4.0 (when available)
WebSphere Application Server
VoiceXML Applications
Star Hotel Chain Applications

Figure 5-32 Star Hotels’ voice support architecture

The telephony piece of the deployment environment consists of one or more VoIP gateways that provide a connection between the Voice Server in an Internet Protocol (IP) network and the regular telephone lines coming in through a Public Switched Telephone Network (PSTN), Global System for Mobile Communication (GSM), or a Private Branch Exchange (PBX) network.

For further information about the Voice Server refer to “Voice server” on page 401.

Part 3

High availability, scalability and security

In this part we focus on the following issues:

- High availability - Continuous operations, twenty-four hours a day seven days a week, which has become the standard and objective in an online environment.
- Scalability - Systems that can scale to support large numbers of users with a consistent response time.
- Security - A primary concern for users is secure data transmissions, and for providers it is protection of internal systems.

6

Operations model

This chapter provides an operational view of the business-to-consumer scenario described in Chapter 5 such that it can be used as a foundation for understanding the scalability and performance discussions found in Chapter 7 and Chapter 8.

Since the Star Hotel chain scenario is fictitious, so are the requirements and assumptions. However, they are nevertheless based on real installations of IBM WebSphere Everyplace Server. Also, since the scenario is fictitious, the numbers and sizes provided are for illustration use only, and may, or may not, be based on real data.

The business-to-consumer scenario and the example used in this operational model are based upon WebSphere Everyplace Server Service Provider Offering.

6.1 Description

An Operational Model is a representation of a network of computer systems, their associated peripherals and the system software, middleware, and application software that they run. It includes the following:

- One or more diagrams that show the topology and geographic distribution of the system, the definition of the nodes (computer platforms) and network connections, plus where and how users and external systems interact with the system being developed.
- A detailed description of each node, which usually includes a table or box diagram that identifies and classifies the software components that run on the node. For convenience, components (which may not all be software) are often grouped into deployment units for ease of placement. The description includes the node's availability, performance, security and other nonfunctional characteristics.
- A detailed description of the networks that connect the nodes, together with their protocol layers and services.
- A mapping matrix of deployment units to nodes, if a significant number of deployment units appear in more than one node (for example, if there is complex data segmentation and replication). Each deployment unit is a convenient grouping of components from the software architecture.
- A description of the systems management strategy, including decisions about centralized vs. distributed managing stations, backup and recovery strategy, software distribution models and approach, change control, configuration management, and other systems management processes.
- A description of middleware services and products and the key middleware choices, including security.
- Descriptions of walkthroughs, which describe the flow of a business activity from a user all the way through the system and back to the user. These textual descriptions may be augmented by interaction diagrams, which show the flow of messages between nodes.

Nodes and connections may be conceptual, specified, or actual physical computer systems, depending on the stage of analysis and design:

- Conceptual corresponds to an early stage of design. Conceptual nodes ignore many technological limitations and focus on application software components, deferring treatment of middleware and other software.
- Specified refers to a detailed specification of a computer platform or network. Technological limitations are fully taken into account but the detailed choice of technology is not made.

- Physical refers to the specific types of computers, networks, and software that make up the system.

Generally an Operation Model develops from conceptual to specified to physical. Depending on the complexity of the problem and the starting point, it may not be necessary to go through all three stages. For example, an architecture may be heavily constrained by physical platform decisions that have already been made, or by existing specification-level reference architecture. At any one time, different parts of the description may be at different levels.

As the Operational Model is developed, detail is added within and between levels of abstraction. The network topology may also be restructured as the design passes from one level to the next.

6.2 Purpose

Different parts of an Operational Model are used for different purposes at different stages of its development.

Early on, when much of the model is at a conceptual level, an Operational Model is used:

- As an early basis for design reviews and walkthroughs, including confirmation that the business problem is well articulated and that there is a viable IT solution.
- As a way of dividing large problems so that each node can be worked on in relative isolation.
- As the basis for early analysis of nonfunctional requirements such as performance, availability, and capacity, including confirmation of the viability of a solution through specification of the expected nonfunctional characteristics of nodes and components.
- To identify necessary technical, infrastructure, and other middleware components and subsystems.
- To allow application developers to modify and elaborate their designs based on an early view of how the application will be implemented and managed.
- To contribute to early estimates of the cost of the infrastructure to be used both for budgeting and as part of the business case for the solution.

Later, an Operational Model at the specification level is used:

- To document the distribution of application and technical subsystems (deployment units) on preliminary (conceptual or specified) nodes so they can ultimately be installed and run on physical computer systems.

- As the basis for detailed design reviews and walkthroughs, prior to selecting products.
- As a detailed technical specification against which an architect can evaluate alternative products or even against which technology vendors can submit tenders.
- As the basis for detailed predictions of performance, availability, and other service level characteristics. (Predictions are based on the overall architecture and the specifications of deployment units within it. They will have to be revisited, via system tests, when specific products have been chosen.).
- As the basis for a check that all the necessary business and technical functionality has been identified.
- To allow application developers to refine and confirm their architecture and designs based on a detailed view of all the solution's deployment units.

An Operational Model at the physical level is used as a blueprint for the acquisition, installation, and subsequent maintenance of the system.

The design of the network is very important because it can affect application design, middleware selection, component placement, systems management and overall operational system control.

6.3 Topological layers

The purpose of this topological layers section is to provide a methodology that you may use to organize the various users and Everyplace Server components. The logical layers organize the components based upon their functions and/or services provided.

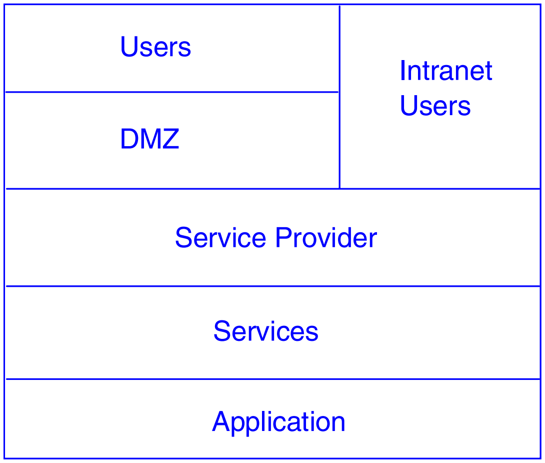


Figure 6-1 Topological layers

In the Users layer you will typically find workstations with browsers directly connected through TCP/IP, or through a dial-up with a wireless clients. You will also find all the supported pervasive devices like the WAP phones, PDAs, and the IBM Voice Server. All the components in the Users layer will access the Service Provider layer through the DMZ, whereas the Intranet Users layer components will not, as they are already in a secure zone.

The Service Provider and Services layers provide all required functions to enable all the users to access the applications found in the Application layer.

The model diagrams and the assumptions made were based on requirements and experience, and are important inputs for architecting scalability and performance, as well as for installing and configuring the surveillance and monitoring systems. Without going through this process, it is unlikely that you will be able to deploy the system into production.

6.4 Deployment units

Deployment units consists of Everyplace Server Service Provider Offering components that would be deployed into the proposed infrastructure. For convenience, we organize the deployment units into the topological layers discussed in “Topological layers” on page 272. Table 6-2 illustrates the placement of the Everyplace Server components into this model.

Once all components are deployed into topological layers, we will start assessing the external access points, the execution nodes and the persistent data nodes.

6.4.1 External access point

Now we start to identify where users enter the system. We now must label the components that are external access points (U), typically accessed by users. The results are shown in Figure 6-2. The lines between the components show the flow between the components.

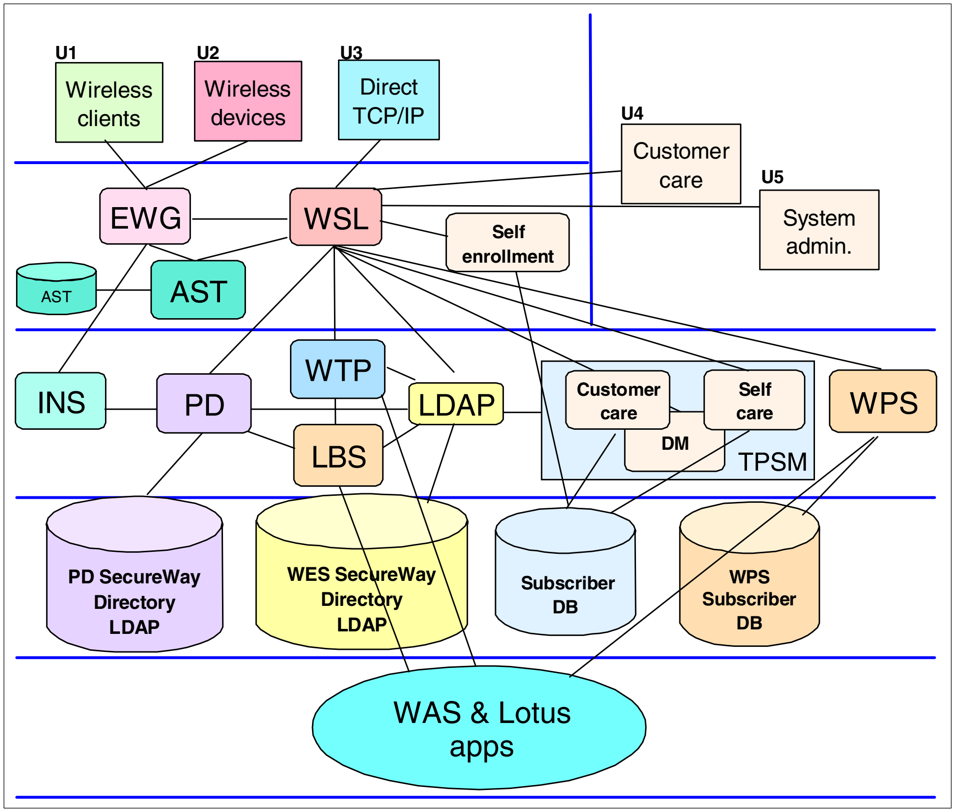


Figure 6-2 Deployment units

Access points U1-U2 are intermittent since they are not always connected. This is basically true for device U3 as well. Although it is a direct TCP/IP-connection, these users will still not always be connected.

Table 6-1 shows for each of the external access points the following information:

- What type of device it is
- Who uses the device
- Where is it located
- When is it likely to be used
- What is the expected level of traffic

Table 6-1 External access points

ID	Device function	Device Type	Who uses the device	Use Location	When used	Concurrent Active
U1	Wireless clients	PC/laptop	Employees & customers	Anywhere	Anytime	5000-10000
U2	Wireless devices	WAP-phone, PDA	Any enrolled user	Anywhere	Anytime	5000-10000
U3	Direct TCP/IP	Any workstation with a browser	Anyone	Anywhere	Anytime	50000-90000
U4	Customer care	Any intranet workstation	Employees	Any Star Hotel	Anytime	50-100
U5	System administration	Any intranet workstation	Employees	System administrators of Star Hotel	Anytime	1-5

6.4.2 Execution elements

Next we must examine where processing occurs within the Everyplace Server infrastructure in order to determine where we may have exposures in capacity and/or availability. In Figure 6-3 we have labeled all components that are “hosting” execution processes (P), or that are performing a decision making function. The lines between the components show the path along which the execution flows.

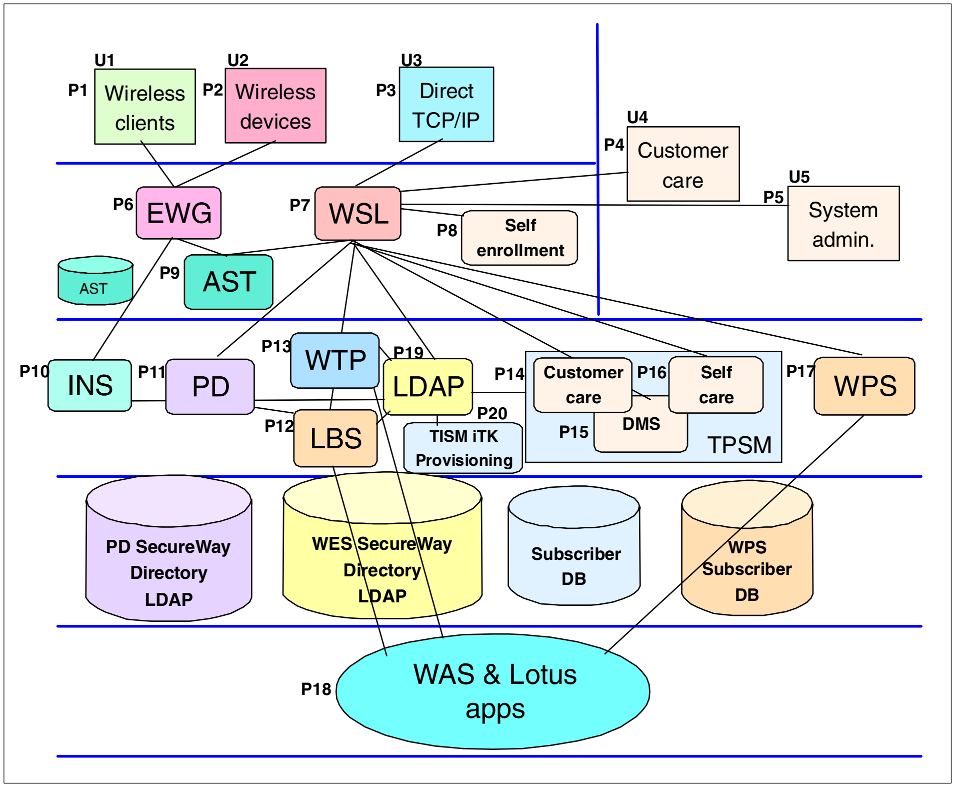


Figure 6-3 Execution elements

Now that we know which components are hosting a decision-making process let us refer to Table 6-2, which adds an assessment of the component availability requirements and expectations. This information will later be used in the availability and scalability design process.

Table 6-2 Executions

ID	Device function	Device Type	Availability	Expectations	Comments
P1	Wireless client	Dial-up	Anytime, but mainly outside office hours	1 client /user	
P2	Wireless devices	WAP-phone and PDA	Anytime	1 client/user	
P3	Direct TCP/IP	online	Anytime	1 client/user	Existing browser
P4	Customer care or system administrators	online	Mainly office hours	1 client/user	Existing browser and system administration tools
P5	System administration	Online	Anytime, but mainly office hours	1 client/user	Accessing system administration tools
P6	Wireless Gateway	online	Continuously	250 request/second	
P7	WebSEAL-Lite	online	Continuously	5700 request/second	Plug-in to Caching Proxy
P8	Self enrollment	online	Anytime	25 request/hour	Servlet
P9	Active Session Table Serverr	online	Continuously	5950 request/second	
P10	Intelligent Notification Server	online	Anytime	200 notifications/minute	Only sending msg out
P11	Policy Director	online	Anytime	4 notification/minute	
P12	Location Based Services	online	Anytime	125 request/second	Plug-in to Caching Proxy
P13	WebSphere Transcoding Publisher	online	Anytime	125 request/second	Stand-alone proxy
P14	Customer care	online	Anytime	100 request/minute	Tivoli Personalized Services Manager Servlet

ID	Device function	Device Type	Availability	Expectations	Comments
P15	Device Manager Server	online	Anytime	200 request/second	Tivoli Personalized Services Manager Servlet
P16	Self care	online	Anytime	300 request/second	Tivoli Personalized Services Manager Servlet
P17	WebSphere Portal Server	online	Anytime	2700 request/second	Providing portal interface to back-end applications
P18	Back-end applications	online	Anytime	5950 request/second	Applications accessed from WebSphere Portal Server portlets, Location Based Services (P12) and WebSphere Transcoding Publisher (P13)
P19	LDAP Directory server	online	Anytime	20 request/second	Accessed by many LDAP-clients
P20	Tivoli Internet Services Manager iTK Provisioning	online	Event or interval driven	Changes to the Tivoli Internet Services Manager DB will trigger the provisioning	Java code executed over RMI

Information in the Expectation column is important, not only for scaling and performance considerations, but also for configuring the specific physical operation model and for the Service Level Agreement (SLA).

Several of the execution components are tightly related or connected, while others are loosely related or connected. Figure 6-3 illustrates that the Wireless Gateway (P6), WebSEAL-Lite (P7) and the Active Session Table Server (P9) are tightly related. All that flows through the Wireless Gateway will affect both WebSEAL-Lite and the Active Session Table. Their relationship is reflected in their performance figures as well as in how their implementation and configuration in the specific physical operational model. These components are critical to the operation of the whole architecture.

6.4.3 Persistent data

In this segment of the model we focus on which functional components contain persistent data (D), and which functional components are related to these data components. Figure 6-4 illustrates where the persistent operational data is stored, and the lines indicate the data flow in the architecture.

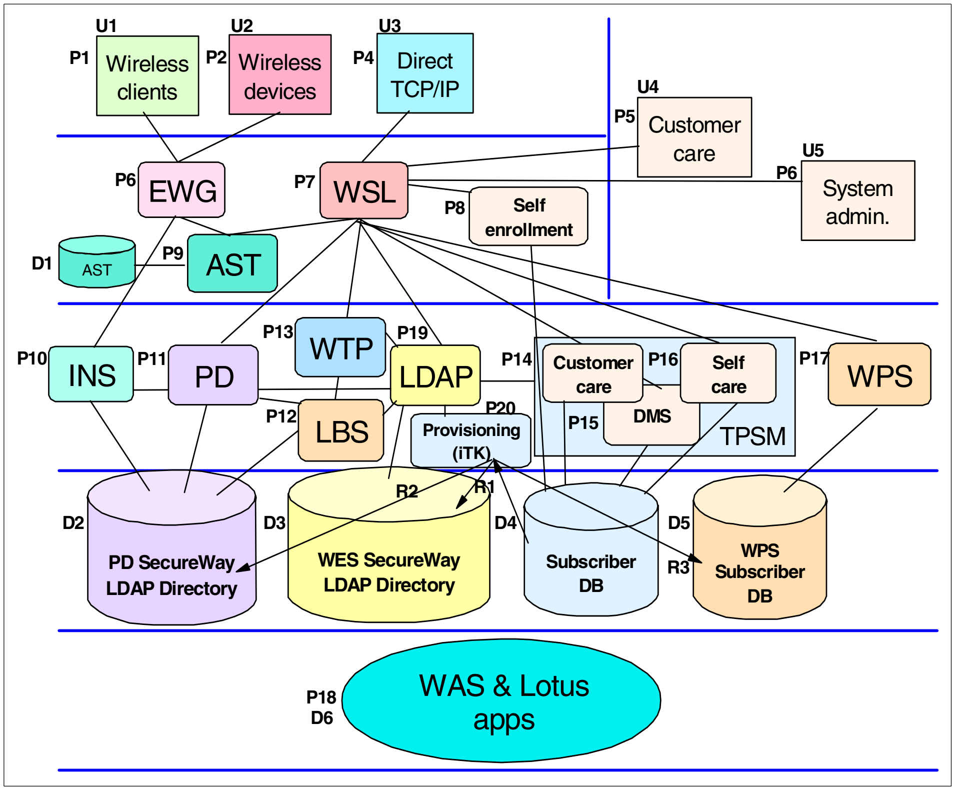


Figure 6-4 Persistent data

The figure also shows the provisioning that Tivoli Internet Services Manager iTK provisioning performs. Original user data, as in self enrollment, are always entered into the subscriber DB and provisioned from there to the Everyplace Server SecureWay Directory (R1), to the Policy Director SecureWay Directory (R2), as well as to the WebSphere Portal Server user directory (R3). See also Figure 5-28 on page 256.

Table 6-3 contains greater details on the persistent data. The table lists the sizes of each entry in the persistence data instances (D), as well as assessment of volumes, volatility and currency.

Table 6-3 Persistent data

ID	Persistence	Type	Volumes	Volatility	Currency
D1	Session IDs	Structured text	1 per session	Volatile	Current, expires after ? seconds
D2	Authentication & authorization data	LDAP	1 ~ 20 per user	Stable	Entries for users, groups, pages and applications
D3	Central config and user preferences.	LDAP	>1 per user and component.	Stable	Current
D4	Subscriber database	LDAP	1 per user	Stable	Current and expired (no auto removal) Populated with about 500.000 users.
D5	Portal user directory	RDBMS	1 per user	Stable	Current and expired (no auto removal)

No estimates have been done on the unit size of each entry in the various databases since it is complicated to give figures and less relevant in infrastructures today with disk space no longer accounting for a significant portion of the investments.

With the exception of the Active Session Table, which is highly volatile, all persistent data will need a backup/recover plan deployed. The Active Session Table Server should be scrutinized for scalability and availability issues to be discussed in Chapter 6 and Chapter 8 respectively.

6.5 Conceptual operational model

The conceptual operational model is a first approximation of where in the topological layers of Figure 6-1 on page 273 we will place business oriented components. Within the conceptual operational model, we refer to the software components as conceptual nodes, and the connections between the conceptual nodes as logical connections (LC).

Conceptual nodes (CN), in general, correspond to an early stage of design. It may ignore some technological limitations and focus on application software components, to some extent, deferring treatment of middleware and other software components.

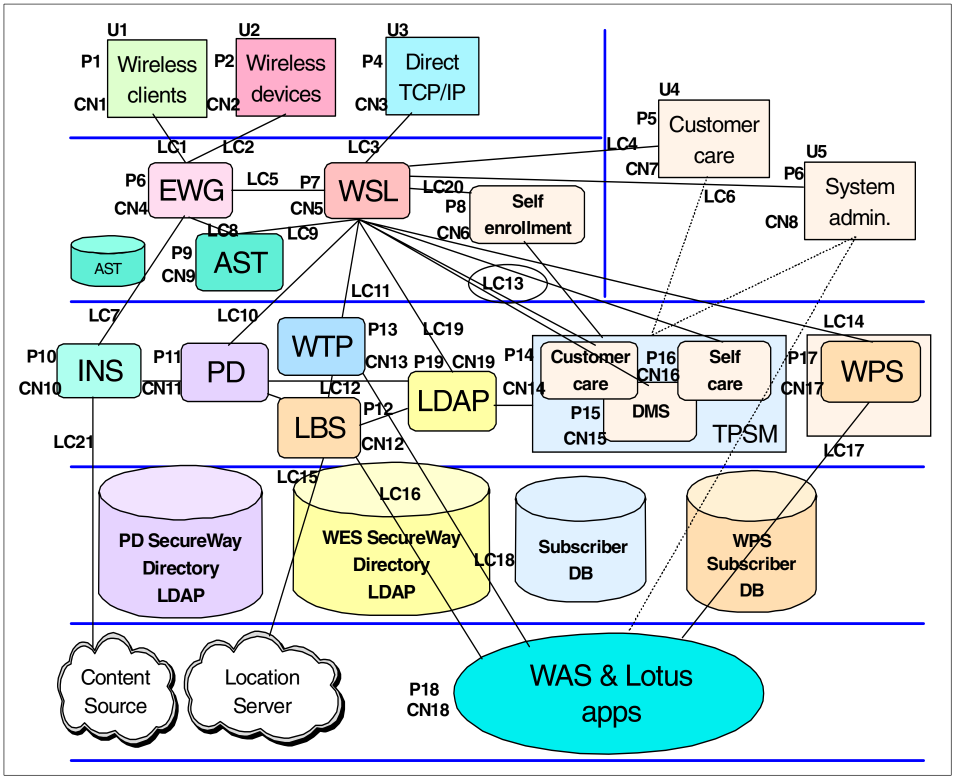


Figure 6-5 Conceptual operational model

Table 6-4 identifies and details all conceptual nodes (CN) shown in Figure 6-5. Table 6-5 on page 285 details all the various connections (LC) between each of the nodes. Together, the conceptual nodes and their detailed connections give important information about availability and performance.

Note: The dotted lines coming out of CN7 and CN8 are to indicate that these two nodes are already in the secure zone and may basically access any nodes that will authenticate them.

Table 6-4 Conceptual nodes (CN)

ID	Conceptual node	Description	Availability	Performance	Comments
CN1	Internet connected by dial-up	The user has installed a wireless client and connects	Anytime	Response time sensitive should be < 5 seconds	
CN2	WAP & PDA devices connecting CN4 over the mobile network	User is authenticated by CN5 and requests forwarded according to WebSEAL-Lite junction configs	Anytime	Response time sensitive should be < 5 seconds	
CN3	Direct connection over TCP/IP	Users connecting over internet	Anytime		
CN4	Wireless Gateway	Converts between WAP and HTTP and handles SMS/WAP-push	Anytime		
CN5	Authentication proxy	Authenticating all users as well as forwarding requests as a junction.	Anytime	Response time critical	
CN6	Self enrollment	Servlet running on WebSphere Application Server to handle first time enrollment.	Anytime	Response time sensitive should be < 5 seconds	
CN7	Customer care	Servlet running on WebSphere Application Server to handle customer care by employees	Mainly office hours.	Response time sensitive should be < 5 seconds	
CN8	System administration	System administrator tools	Anytime, but mainly office hours.	Sensitive, should be < 5 seconds	

ID	Conceptual node	Description	Availability	Performance	Comments
CN9	Session server	Handles session ID and enables single sign-on	Anytime	Critical	Session records expires (by default) after 10 minutes.
CN10	Intelligent Notification Server	Notification services	Anytime	Not response time sensitive as it is a one directional service	
CN11	Policy Director	Handles authorizations	Anytime	Critical	
CN12	Location Based Services	Location-based service	Anytime	Sensitive, should be < 5 seconds	
CN13	WebSphere Transcoding Publisher	Transcoder	Anytime	Sensitive, should be < 5 seconds	
CN14	Customer care	Servlet running on WebSphere Application Server	Mainly office hours.	Sensitive, should be < 5 seconds	
CN15	Device Management	Servlet running on WebSphere Application Server	Anytime, but mainly daytime.	Sensitive, should be < 5 seconds	
CN16	Self care	Servlet running on WebSphere Application Server	Anytime, but mainly daytime	Sensitive, should be < 5 seconds	
CN17	Portal	Complete configurable portal running on a dedicated WebSphere Application Server	Anytime	Sensitive, should be < 5 seconds	
CN18	Back-end applications	Hotel existing back-end applications for booking, check-in/out, etc.	Anytime, but mainly during office hours.	Sensitive, should be < 5 seconds	
CN19	LDAP Directory server	Daemon providing access to the directory data	Anytime	Sensitive and critical	

In addition to giving information about performance and availability, Table 6-5 includes important information on what transport layers and protocols are used over the connections.

Table 6-5 Logical Connections (LCn)

ID	Logical Connection	Description	Availability	Performance	Comments
LC1	Dial-up PPP	Wireless clients connects to the Wireless Gateway	Anytime	Limited by what the carrier provides	
LC2	Dial-up from wireless devices (WAP-protocol)	WAP phones and PDAs connecting over the mobile network	Anytime	Limited by what the network offers.	GSM/GPRS
LC3	TCP/IP connections from the Internet	Workstations with browser connecting through the Internet	Anytime	< 5 seconds	
LC4	Intranet users	Intranet customer care users connecting trough a browser.	Mainly office hours	< 5 seconds	
LC5	HTTP connection	All requests coming in from wireless clients and devices, plus additional HTTP headers inserted by the Wireless Gateway	Anytime	Critical	
LC6	System administration	System administrators accessing any servers for maintenance and surveillance	Anytime, but mainly daytime		
LC7	SMS push requests	TCP/IP-based push requests and responses	Anytime	Not critical	
LC8	Wireless Gateway inserting session-ids	For PPP connections, the Wireless Gateway will insert the session ID into Active Session Table	Anytime	critical	

ID	Logical Connection	Description	Availability	Performance	Comments
LC9	WebSEAL-Lite checks/inserts session-ids	WebSEAL-Lite will check the Active Session Table Server for all requests and insert sessions ids if its not there	Anytime	Highly critical and heavily loaded.	The session an XML will already be in the Active Session Table if the request comes over PPP and is being inserted by the Wireless Gateway
LC10	Policy Director lookup	WebSEAL-Lite checks Policy Director for authorizations. LDAP over SSL	Anytime	Critical	
LC11	Location based applications	WebSEAL-Lite will, based on its *Junction* config, forward all location based requests to Location Based Services through the WebSphere Transcoding Publisher. These request will always originate from U2.	Anytime	< 5 seconds	The Wireless Gateway has inserted the MSISDN in the request header (needed by the location server)
LC12	Location based applications	Request coming through WTP untouched, responses from the location based applications are in XML and will transcoded to device spec. WML by WebSphere Transcoding Publisher.	Anytime		

ID	Logical Connection	Description	Availability	Performance	Comments
LC13	HTTP request/response to Tivoli Internet Services Manager applications	“Cluster” of connections to Customer Care, Self Care and Device Manager Server.	Anytime	< 5 seconds	
LC14	HTTP request/response to the portal	Portal application. Request/response as well as LTPA-tokens with authorization data to WebSphere Application Server	Anytime, but mostly daytime	< 5 seconds	
LC15	Location server	VPN over Internet. The Location server (such as SignalSoft) normally is physically located by the telcos/carriers. The traffic will be TCP XML request and response.	Anytime		
LC16	Location based applications over HTTP	HTTP request/response to location based applications with GML inserted in the request header.	Anytime		
LC17	Portal applications accessed through portlets	HTTP	Anytime, but mostly daytime		
LC18	Back-end applications for transcoding	HTTP	Anytime	< 5 seconds	A selection of existing back-end applications like reservations will be made available to wireless devices
LC19	Directory access	LDAP over SSL	Anytime	critical	

ID	Logical Connection	Description	Availability	Performance	Comments
LC20	Self enrollment	HTTPS	Anytime	< 5 seconds	Requests forwarded through WebSEAL-Lite unauthenticated enabling any user to enroll
LC21	Intelligent Notification Server content source	VPN over internet to various content source providers	Anytime	< 30 seconds	

6.6 Specified operational model

Whereas the conceptual operational model is a higher level, a first approximation, the Specified Operational Model goes into details of the actual workings of the system. The Specified Operational Model determines how:

- "Point-to-point" interconnections between the various application components actually will be realized
- Data placement strategy will be supported
- Operational aspects of the application will be ensured
- Underlying IT services (including systems management functions) will be supplied

6.6.1 Routed connections

This part of the model addresses how connections are routed through the infrastructure. It is important that you know this information in order to appropriately plan for availability and scalability.

Figure 6-6 does not show all possible routed connections, only the three most important from a performance and operational point of view (RC1, R2 and R3).

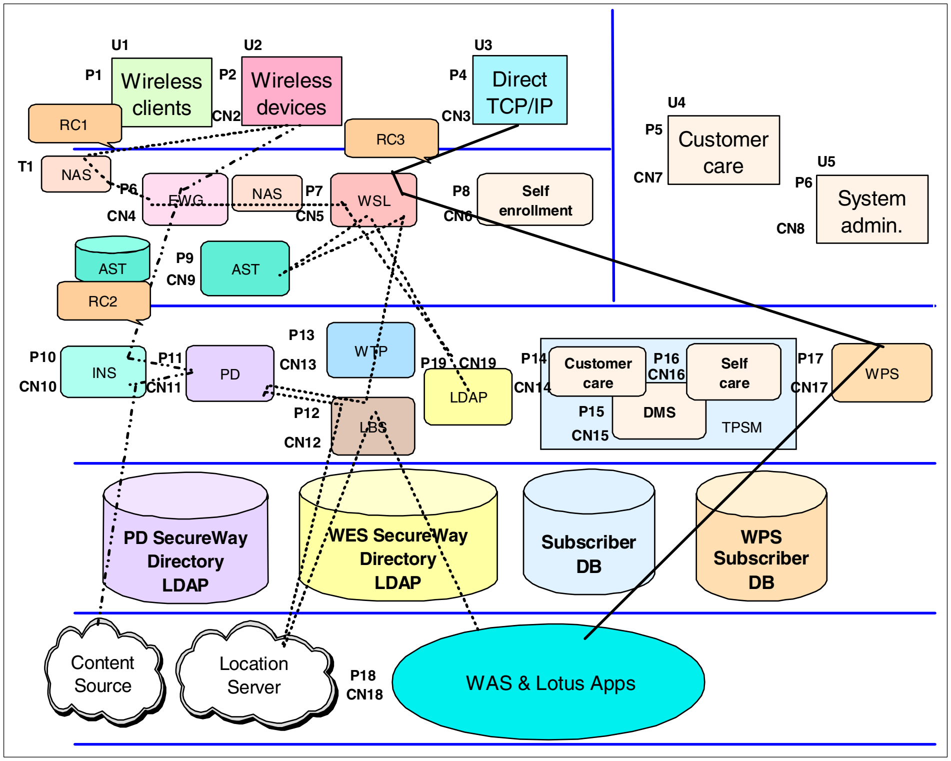


Figure 6-6 Routed connections

Connections from U1 and U2 flow through the system in a similar path but are treated differently by the Wireless Gateway and WebSEAL-Lite, as described in Table 6-6. Note that WebSEAL-Lite is acting as a junction for the requests coming in from all access points. The routed connection from Intelligent Notification Services is distinct for the SMS push, whereas the third connection, RC3 coming from U4, will possibly be the most frequently used connection to the back-end applications.

Table 6-6 Routed connections

ID	Routed connection	Description	Comment
RC1	Wireless clients and devices connecting to portal and back-end applications	a) The wireless devices connects to the Network Access Server (NAS), gets through to Everyplace Wireless Gateway along with a RADIUS-package containing IP address, MSISDN and user ID on the NAS. Everyplace Wireless Gateway inserts the MSISDN into a header and passes the request, through a second NAS, to WebSEAL-Lite for authentication. after being authenticated, the request is passed through to the requested applications, in this case through the WebSphere Transcoding Publisher, Location Based Services, Policy Director, Loc.server and finally the loc.based applications See also “WAP requests (RC1a)” on page 292. b) A second similar scenario is when a wireless client connects. In this scenario, the clients dials up the Everyplace Wireless Gateway, gets authenticated by PPP CHAP/PAP on the Everyplace Wireless Gateway which inserts the session an XML in the Active Session Table and the request header and passes the request to the requested applications through WebSEAL-Lite. See “Wireless clients (RC1b)” on page 293.	The route shown in Figure 6-6 goes through the WebSphere Transcoding Publisher and Location Based Services, but the same flow as shown in Figure 6-7 applies if the route goes to WebSphere Portal Server or other accessible Web applications
RC2	SMS push messages initiated by Intelligent Notification Server	This route starts in Intelligent Notification Server based on content provided from the content source (over VPN) and the subscription and privacy in Policy Director. Intelligent Notification Server will then “push” the message out to the wireless device (if that is the users preferred device) through the Everyplace Wireless Gateway. See “Push messages (RC2)” on page 295 for more details.	This scenario applies for both SMS and WAP push messages.
RC3	Direct TCP/IP connections to applications	This is possibly the most common route initiated from a TCP/IP-connected browser, authenticated by WebSEAL-Lite and forwarded to the back-end application through theWebSphere Portal Server. See “Direct TCP/IP connections” on page 296 for more details.	

Note: In all the following sequence diagrams, firewalls are included just to illustrate to the reader which components are in the DMZ and which are not. The firewalls themselves are not a formal part of the diagrams.

WAP requests (RC1a)

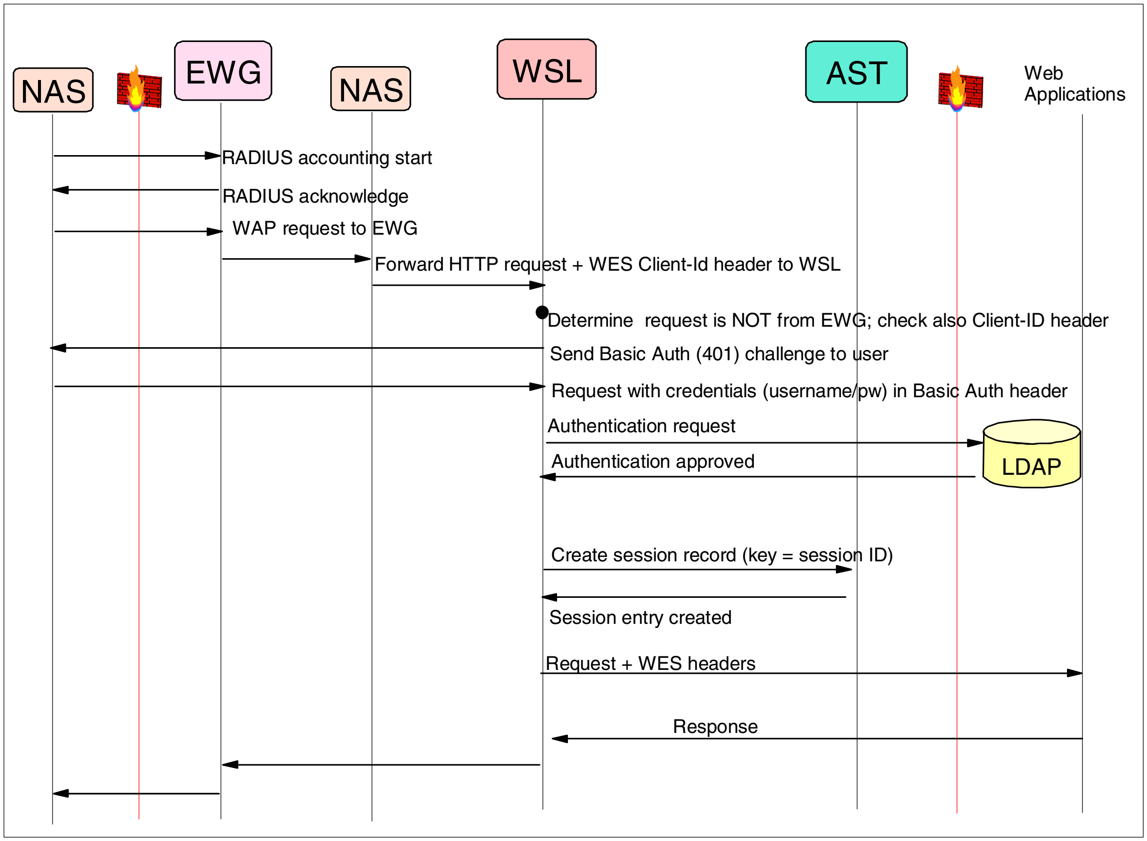
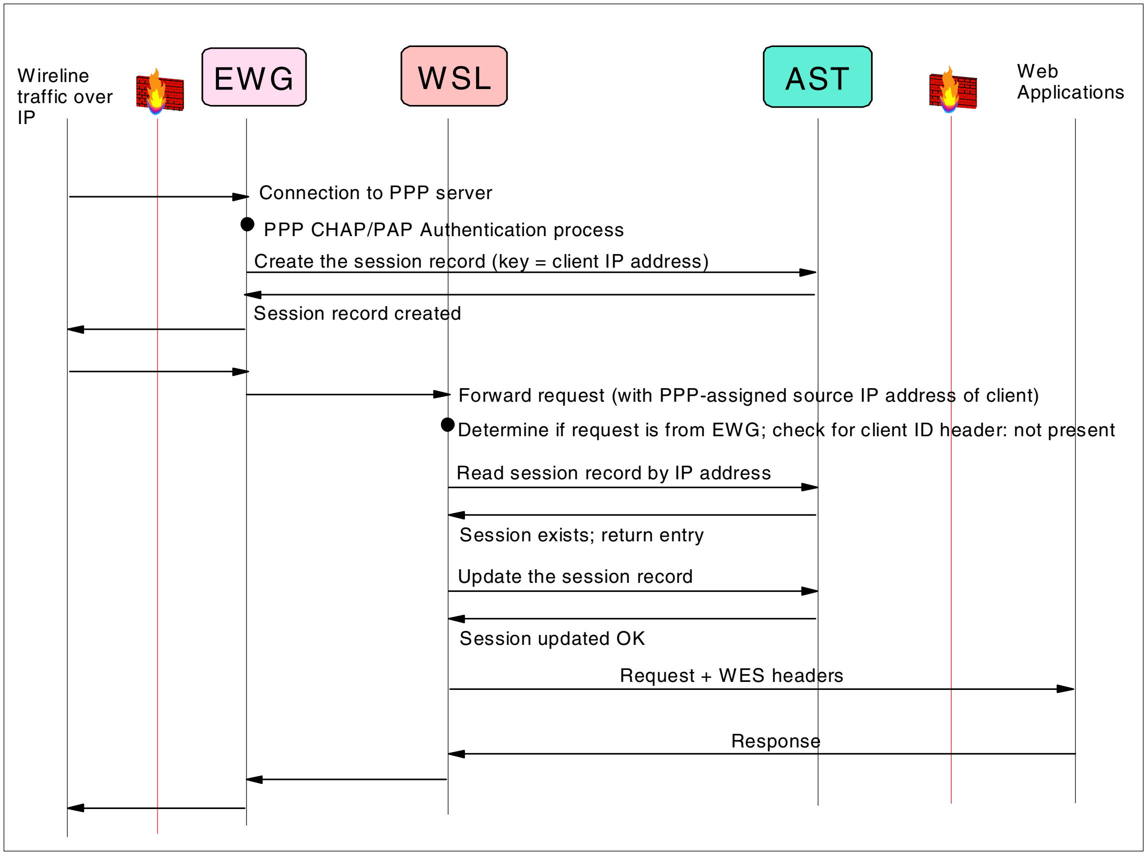

NAS
EWG
NAS
WSL
AST
Web Applications
RADIUS accounting start
RADIUS acknowledge
WAP request to EWG
Forward HTTP request + WES Client-Id header to WSL
Determine request is NOT from EWG; check also Client-ID header
Send Basic Auth (401) challenge to user
Request with credentials (username/pw) in Basic Auth header
Authentication request
Authentication approved
LDAP
Create session record (key = session ID)
Session entry created
Request + WES headers
Response

Figure 6-7 WAP-request through Wireless Gateway with the Active Session Table Server

In Figure 6-7 the Everyplace Wireless Gateway (EWG) is functioning as a WAP Gateway only. As a WAP Gateway, Everyplace Wireless Gateway does not interact with the Active Session Table Server. Rather, it uses HTTP headers to communicate information needed by WebSEAL-Lite (WSL). The WAP request is received from the WAP device through the NAS and the WAP Gateway transforms the request to HTTP and inserts the `x-ibm-pvc-user` and `x-ibm-pvc-client-id` headers. The Everyplace Wireless Gateway then forwards the request to its next proxy hop, which is the second NAS, which is simply forwarding the request to WebSEAL-Lite. WebSEAL-Lite will not recognize the request coming from the Wireless Gateway, so it challenges the user to authenticate. The credentials will be checked against the SecureWay LDAP Directory, when approved, WebSEAL-Lite extracts the header information and creates an Active Session Table record containing information it received from the Everyplace Wireless Gateway and information it generates itself (including the session ID). It then forwards the request to the content server (Web applications).

Wireless clients (RC1b)



Wireline traffic over IP
EWG
WSL
AST
Web Applications
Connection to PPP server
PPP CHAP/PAP Authentication process
Create the session record (key = client IP address)
Session record created
Forward request (with PPP-assigned source IP address of client)
Determine if request is from EWG; check for client ID header: not present
Read session record by IP address
Session exists; return entry
Update the session record
Session updated OK
Request + WES headers
Response

Figure 6-8 PPP authentication through Wireless Gateway and WebSEAL-Lite

The flow in Figure 6-8 illustrates the interactions with the Active Session Table Server for the configuration where Wireless Gateway serves as the PPP server. In this scenario, the Wireless Gateway (EWG) uses the Active Session Table to pass information (such as the user's identity) to WebSEAL-Lite (WSL). After authenticating the user, Everyplace Wireless Gateway generates a session ID and creates an Active Session Table entry for the session containing the user identity and a default network type. An Active Session Table client ID, identifying the calling Wireless Gateway instance, is also included.

The Active Session Table Server name is taken from the common Active Session Table setting in the WebSEAL-Lite part of the LDAP tree. The device type is set to "unknown" since the Everyplace Wireless Gateway has no way of knowing the actual device at the time the PPP protocol logic is driven. The request is forwarded to the destination URL and WebSEAL-Lite detects that the request is from the Wireless Gateway and thus trusts that Wireless Gateway has done authentication.

WebSEAL-Lite also checks for any Wireless Gateway user header, which doesn't exist, so it then looks up the session in the Active Session Table using the source IP address as the key for the lookup (the source IP address is the one allocated to the device/client by the Wireless Gateway during PPP negotiation). Active Session Table Server returns the session record to WebSEAL-Lite, which extracts the user's identity from the record and includes it in the `x-ibm-pvc-user` header. WebSEAL-Lite updates the Active Session Table entry adding the actual device type of the user's device (note that subsequent requests from the same user would not result in an update to the Active Session Table). WebSEAL-Lite then forwards the request to the content server.

Push messages (RC2)

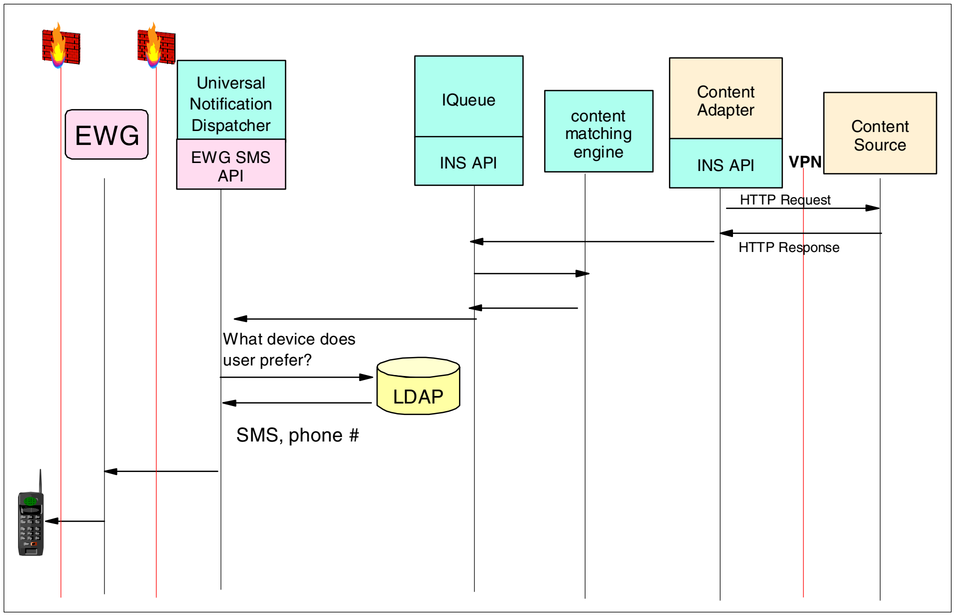


Figure 6-9 Intelligent Notification Services message

The flow in Figure 6-9 demonstrates an Intelligent Notification Services match of content with filters in a user's subscription profile. The various components of the Intelligent Notification Server interact to process the content, determine that there is a match, and check the user's preferences in LDAP to determine which device to use. On determining that the user prefers a Short Message (SMS) to be sent to his wireless phone, the Wireless Gateway Messaging API is invoked to send the message to the user. If the user's preferences indicated a WAP phone was to be used or e-mail was to be sent, this high-level flow would be identical since the Everyplace Wireless Gateway Messaging API provides for SMS and WAP push and SMTP for e-mail.

Direct TCP/IP connections

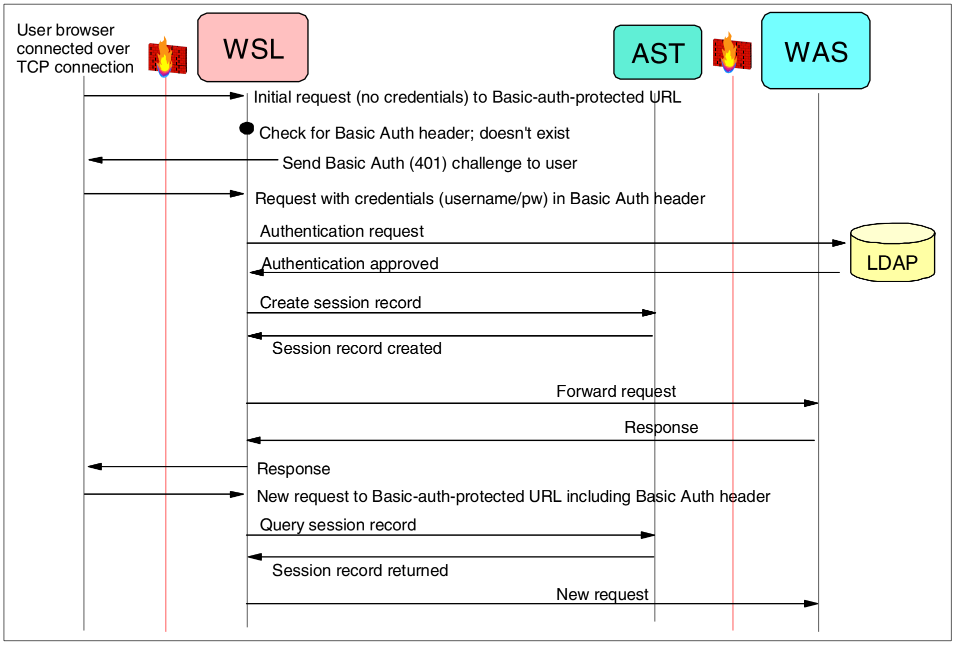

User browser connected over TCP connection
WSL
AST
WAS
LDAP
Initial request (no credentials) to Basic-auth-protected URL
Check for Basic Auth header; doesn't exist
Send Basic Auth (401) challenge to user
Request with credentials (username/pw) in Basic Auth header
Authentication request
Authentication approved
Create session record
Session record created
Forward request
Response
Response
New request to Basic-auth-protected URL including Basic Auth header
Query session record
Session record returned
New request

Figure 6-10 Basic authentication

In the Figure 6-10 flow, the user makes a request from a browser to a URL protected by Basic Authentication. WebSEAL-Lite detects that no Basic Auth header is present in the request, so it challenges the user (browser) to return the header (401 error code). The user enters his user name and password on the browser window and the browser returns the request with the Basic Auth header. WebSEAL-Lite authenticates the user by lookup and password comparison in WebSphere Everyplace Server SecureWay LDAP Directory and then creates a session record for the user in the Active Session Table. The request is forwarded and the response is returned. Subsequent requests to the same domain include the Basic Auth header which WebSEAL-Lite simply validates by lookup in the Active Session Table before forwarding the request.

Technical deployment units

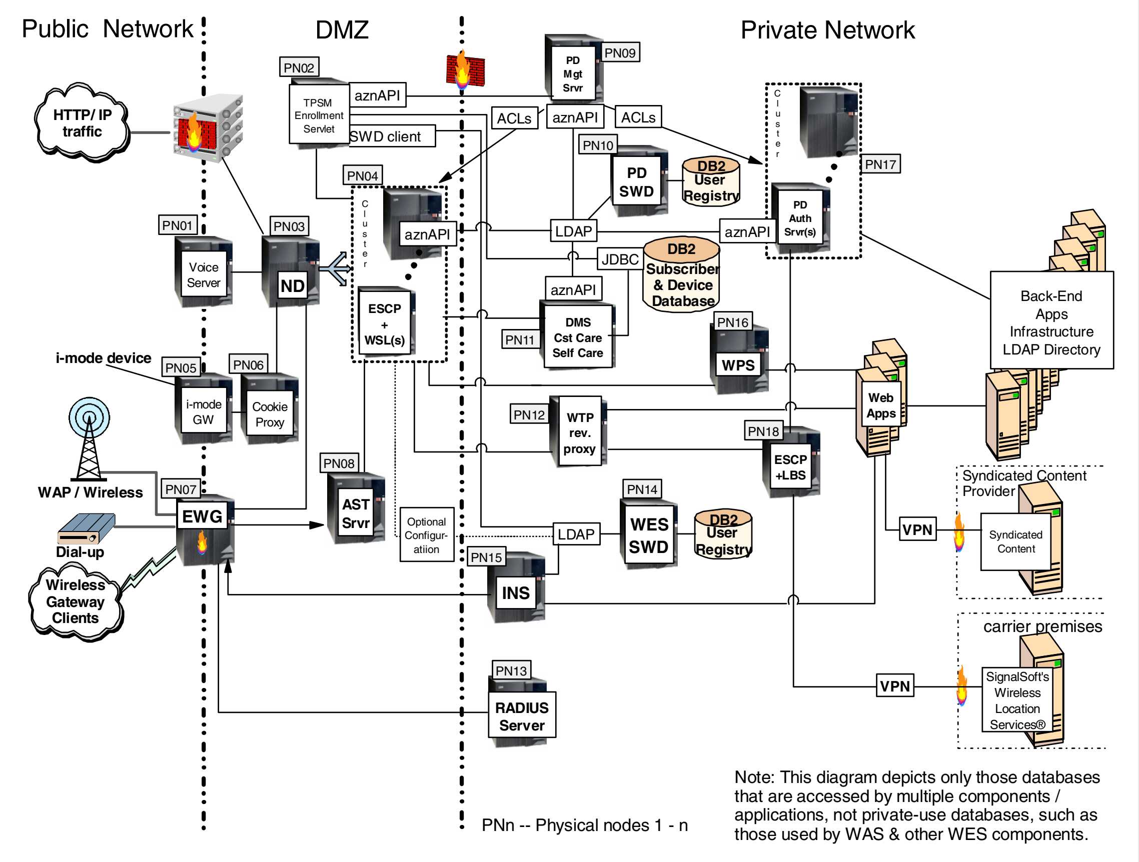
The technical deployment units are units that are not part of the WebSphere Everyplace Server infrastructure, but are nevertheless required units to be deployed to make the entire solution operational.

Table 6-7 Technical deployment units

ID	Purpose	Type	Usage "window"	Number per user	Comments
T1	Connecting ISDN-lines and V110 provisioning for TCP/IP over GSM	CISCO router	Anytime	1 user per ISDN-channel	In Figure 6-7 we have 2 NAS. Only the first needs V110. The second can be "any" forward proxy.
T2	Firewall	Can be any!	Anytime	NA	
T3	VPN	CISCO router	Anytime	NA	In the scenario as a whole, 2 VPNs are needed: one for the Intelligent Notification Server content source and for the Location Based Services Location server.

6.7 Physical operational model

The physical operational model implements the specified model in terms of physical units and technologies and their configured software and relationships:



PNn -- Physical nodes 1 - n

Note: This diagram depicts only those databases that are accessed by multiple components / applications, not private-use databases, such as those used by WAS & other WES components.

Figure 6-11 Physical operational model

The following table describes the products installed on each of the physical nodes in Figure 6-11.

Table 6-8 Physical Structure of the System

Physical Node	Software needed	Installed Products	Purpose of this Node	Comments
PN1	Windows 2000	Voice Server	Voice Support	Connected via WebSphere Transcoding Publisher V4.0
PN2	AIX 4.3.3 or Solaris 2.7	- Tivoli Personalized Services Manager Self Enrollment	Customers Self Enrollment	
PN3	AIX 4.3.3 or Solaris 2.7	Edge Server Load Balancer	Load Balance through Edge Server Caching Proxy	
PN4	AIX 4.3.3 or Solaris 2.7	- Edge Server Caching Proxy - WebSEAL-Lite	Authenticates users	Composed of several servers
PN5	AIX 4.3.3 or Solaris 2.7	i-mode Gateway	For i-mode devices support	
PN6	AIX 4.3.3 or Solaris 2.7	Cookie Proxy	For i-mode support	
PN7	AIX 4.3.3 or Solaris 2.7	Everyplace Wireless Gateway	Wireless client access point	
PN8	AIX 4.3.3 or Solaris 2.7	- Active Session Table Server	Active users information server	
PN9	AIX 4.3.3 or Solaris 2.7	Policy Director Manager Server	Keep accurate ACLs in the system	
PN10	AIX 4.3.3 or Solaris 2.7	- Policy Director - SecureWay Directory	LDAP DB access	
PN11	- AIX 4.3.3 or Solaris 2.7 - WebSphere Application Server 3.5 - DB2	- Device Management Server - Self Care - Customer Care - Tivoli Personalized Services Manager Database	Customizes user profile and devices enrollment	
PN12	AIX 4.3.3 or Solaris 2.7	- WebSphere Transcoding Publisher	Application Transcoding	
PN13	AIX 4.3.3 or Solaris 2.7	- RADIUS Server	- Dial Security	

Physical Node	Software needed	Installed Products	Purpose of this Node	Comments
PN14	AIX 4.3.3 or Solaris 2.7	- IBM WebSphere Everyplace Server SecureWay Directory	LDAP DB access	
PN15	AIX 4.3.3 or Solaris 2.7	- Intelligent Notification Server	Users notification system	
PN16	- AIX 4.3.3 or Solaris 2.7 - WebSphere Application Server 3.5	- WebSphere Portal Server	System access portal	
PN17	AIX 4.3.3 or Solaris 2.7	- Policy Director Authentication Server	Validates user access	Composed of several servers
PN18	AIX 4.3.3 or Solaris 2.7	- Caching Proxy - Location Based Services	Manages location application	

Tivoli Personalized Services Manager products (installed in physical node 11) need WebSphere Application Server 3.5 since they are servlets that run under an application server.

All the products could be installed in either AIX or Solaris except the Voice Server, which needs to be installed in a Windows 2000 server. The current equipment of the Star Hotels chain can be reused (Solaris servers and a Windows 2000 server). If new equipment is needed, it could be either AIX or Solaris.

7

Scalability

This chapter first describes the scalability characteristics of the IBM WebSphere Everyplace Server infrastructure, then describes in greater detail the relevant design features of each individual component.

7.1 Scaling WebSphere Everyplace Server

Scaling an end-to-end solution is a matter of adjusting the capacity and performance of all its components in a balanced, coordinated fashion. Typically as you increase the scalability of one component to remove a bottleneck, it will change the dynamics of the system, often moving the bottleneck, albeit at a higher load, to another component of the solution. To maximally scale an entire solution, many, or perhaps even all, of its components must scale both individually and in concert to be able to cope with the increasing demand.

In this section we summarize the main techniques you can apply to components of an e-business solution to increase their scalability.

- **Faster machines (vertical scaling)**: This is where we upgrade components of a solution to run on faster machines, meaning that they can process tasks more rapidly, allowing them to do more work in a given amount of time. The faster machines achieve this by providing more resources, for example CPU cycles, I/O bandwidth, in a single physical unit.
- **Using replicated machines (horizontal scaling)** is another way of applying more resources to a given workload, typically improving the performance of the affected components. The parallel nature of replicated machines also leads to improved response times. In addition, replicated machines improve availability, since load can be shifted to other machines if one or more replicas become unavailable due to failures or scheduled maintenance. In WebSphere Everyplace Server, this is applied by placing clusters behind Load Balancer, or by using a cluster of WebSphere Application Server servers.
- **Specialized machines** improve the efficiency of a specific component, allowing that component to perform more of the required function in a given amount of time. They tend to be very fast and efficient, but sacrifice function and flexibility to achieve this. Typically they also tend to be more expensive than an equivalent general-purpose solution.
- **Segmenting workload** is a technique where the workload is split into smaller, more manageable pieces. The intention is to provide a more consistent (and so predictable) response time, while also making it easier to manage which server(s) process the workload. In essence, the WebSphere Everyplace Server infrastructure is implemented by segmenting functions across multiple servers.
- **Request batching**: Since the overhead associated with issuing a request from one component to another is virtually identical no matter what the request is, making fewer but larger and more complex requests reduces the total amount of overhead. This reduces the load by eliminating the overhead costs associated with multiple requests, reduces the latency, and possibly

improves end-user response times by reducing time spent on the processing of overhead tasks.

- **Data aggregation** into a more readily accessible form is designed to provide rapid access to user data for a very large number of concurrent users. If the information is readily available in the aggregation, it can typically be supplied to large numbers of users with high performance. If the information has to be generated or accessed each time using an application, then it will not be possible to provide access to many users with high performance. This technique would generally be implemented in the application layer.
- **Connection pooling**: When an application is distributed, a connection between the distributed layers must exist. Connection pooling is designed to minimize the total number of connections needed for an end-to-end system, while eliminating the overhead of connection setup and termination. It is typically controlled by a connection management system.
- **Caching** is a technique to improve the performance and scalability of a solution by reducing the path length a request/response has to travel, and so reducing the resource consumption of components in the solution. Ideally the data should be cached at or very close to the site of the reference, but can be spread along the path between the server and the client, depending on the actual need. Once built, pages stored in and served from a cache are served to clients significantly faster, and consume fewer resources than those pages that are built synchronously by Web servers. In WebSphere Everyplace Server, caching is often implemented by adding the Edge Server Caching Proxy component.

Determining which of the techniques will provide you with the best improvements in scalability and performance is not simple; it will depend to a very large extent on the exact details of your solution. However, the diagram in Figure 7-1 provides an excellent summary of where you may derive benefit from each of the techniques.

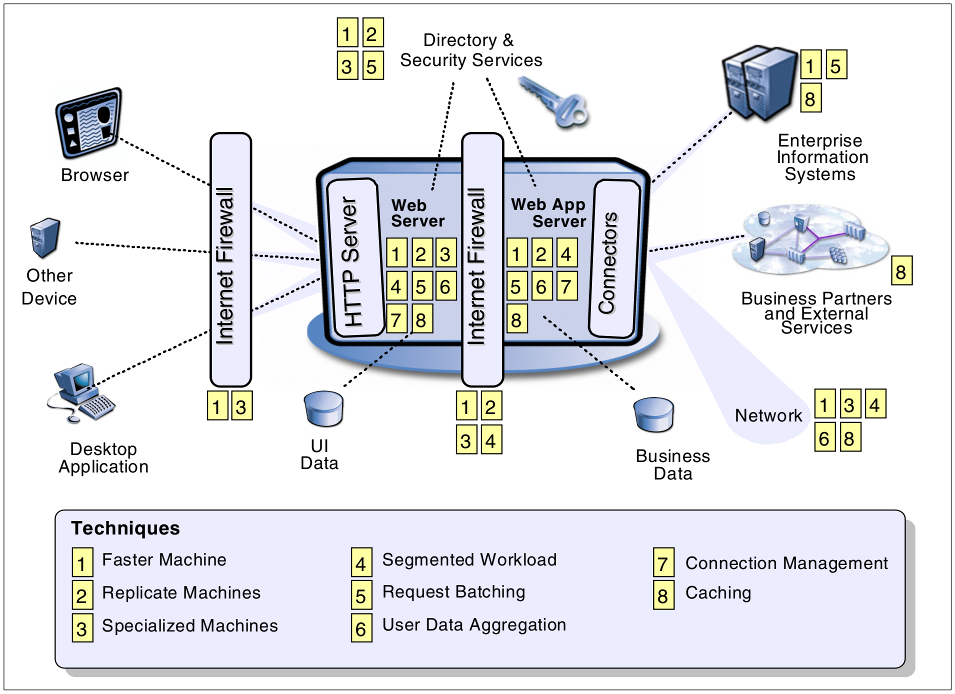


Figure 7-1 Scalability technique applicability

However, as you read them and consider how they can be applied to your environment, remember this truism, attributed to one of IBM's e-business scalability experts:

"All the scaling techniques in the world can easily be rendered useless by the skills of an unwitting programmer or administrator." - Leonard Hand, IBM.

To make your solution perform and scale well, it is essential for you, and all the people who are involved with your solution, to understand the effect that even small, apparently isolated decisions can have on the overall solution.

The next sections describe the application of these techniques in an Everyplace Server environment, and then go on to describe, for individual products, some of these small, apparently isolated decisions.

Vertical and horizontal scaling in Everyplace Server

Figure 7-2 shows how vertical and horizontal scaling techniques may be applied to IBM WebSphere Everyplace Server.

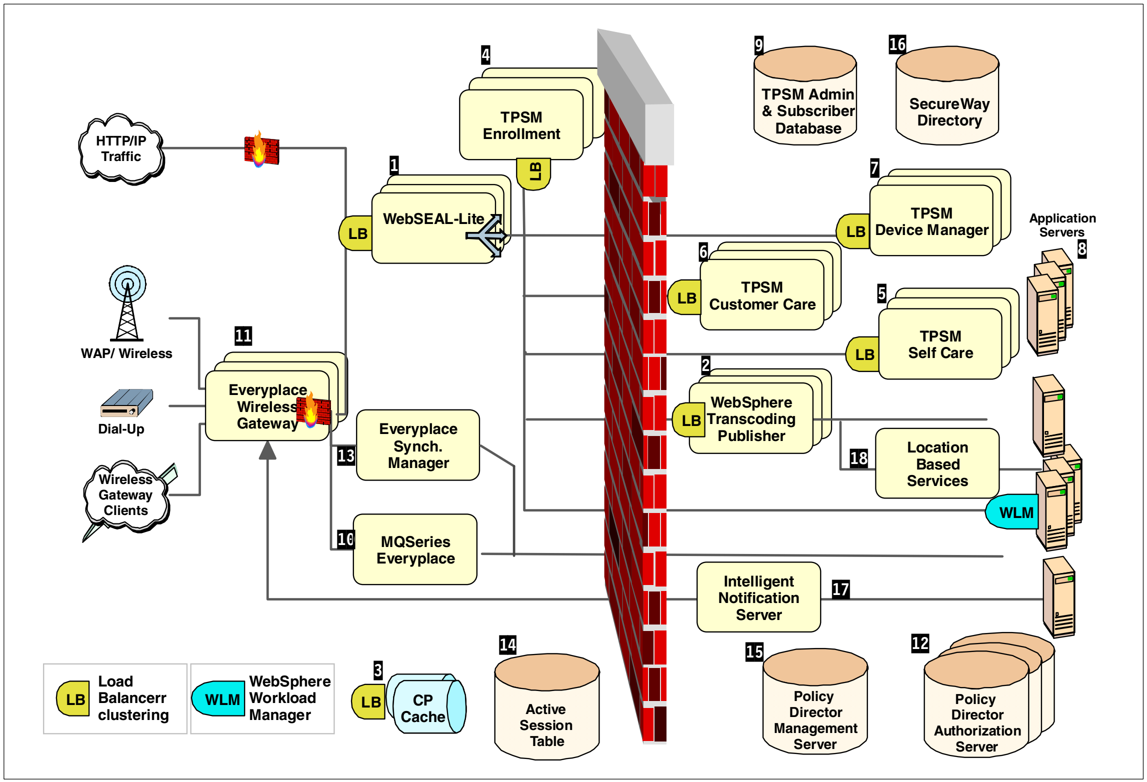


Figure 7-2 Scaling WebSphere Everyplace Server

Note that there are two kinds of clusters shown in the diagram:

- Edge Server Load Balancer clustering, previously known as Network Dispatcher. These clusters are labelled “LB” in the diagram.
- WebSphere Application Server clusters, which can be implemented using WebSphere Application Server Workload Managed clusters, Load Balancer clustering, or, in extreme circumstances, both. These are labelled “WLM” in the diagram.

A key component in the scaling described here is the use of *smart junctions* within WebSEAL-Lite. If the applications are segmented such that the highest-level directory (or, in the case of WebSphere Application Server, the Web application name) is different for each application, then smart junctions within WebSEAL-Lite can be used to direct each request to a different set of servers or different segment of the infrastructure.

The following Everyplace Server components are best suited to scaling via the Load Balancer:

1 WebSEAL-Lite

2 IBM WebSphere Transcoding Publisher

3 Edge Server Caching Proxy (depending on its use and configuration)

4 Tivoli Personalized Services Manager, Enrollment

5 Tivoli Personalized Services Manager, Self Care

6 Tivoli Personalized Services Manager, Customer Care

7 Tivoli Device Manager Server

Since traffic comes to WebSEAL-Lite with the origin’s IP address, Load Balancer’s “sticky bit” affinity can be used for WSL. If affinity to a specific server is required for the other Load Balancer instances and must be provided on a per-session basis, then “cookie affinity” is the most appropriate mechanism.

The following applications are hosted by Web application servers, and should be scaled using the Web application server’s scaling mechanisms. In general, this will include segmenting the workload across the collection of Web application servers, vertical and horizontal scaling, and enabling connection pooling to database servers:

8 Other application servers inside the Everyplace Server domain. For these application servers, techniques such as request batching and data aggregation may also be used, if the applications are appropriately designed.

For other Everyplace Server components it is either not suitable to use the Load Balancer or the individual component may provide its own specialized load balancing, such as workload segmentation. We recommend you use these techniques for the following components in the Everyplace Server:

9 Tivoli Personalized Services Manager Subscriber database - workload segmentation

10 IBM MQSeries Everyplace - workload segmentation

11 IBM Everyplace Wireless Gateway - built-in clustering

12 Policy Director Authorization Server - built-in replication, with applications pointed to individual replicas

13 Everyplace Synchronization Manager - workload segmentation

The following components can only be vertically scaled:

14 Active Session Table Server

15 Tivoli SecureWay Policy Director Management Server

16 IBM SecureWay Directory (in the WebSphere Everyplace Server environment)

17 Intelligent Notification Services

18 Location Based Services

Over-balancing the Everyplace Server

The Load Balancer and other clustering mechanisms can be effectively used with many of the Everyplace Server components to provide availability and scalability by load balancing requests over horizontal servers.

However, introducing a Load Balancer in front of each clustered component can create problems. Avoid over-balancing.

Unnecessary latency is to be prevented at all costs by ensuring that the shortest possible path lengths are used. Careful consideration must be made regarding the extra latency that introducing multiple cluster management systems will create in the overall path. Each extra cluster manager you introduce to your architecture will introduce a further hop that a request must travel through before reaching its final destination. With each hop the client is still waiting for a response and the response rate can be increased. The worst-case scenario of this over-balancing is that performance can be negatively impacted. This is contrary to exactly what the cluster mechanisms are designed to solve. The number of clusters that you deploy in your solution must be weighed against your goals for increasing performance.

Caching in the Everyplace Server

As the number of users entering the Everyplace Server increases, your Everyplace Server architecture must both perform and scale. To design a performing Everyplace Server architecture, response rates must be high and path lengths must be minimized. A key mechanism to assist with this is the addition of caches into the infrastructure.

Note that most applications perform some level of caching at the application level; for example, the LDAP client, installed in each server that requires LDAP access, uses an internal cache to reduce the number of calls required; IBM HTTP Server, used by the Web applications, also provides an internal cache that can be tuned. This section will only describe caches that must be consciously added to the infrastructure.

The Caching Proxy is implemented in the Everyplace Server in two distinct methods:

- The functions that Edge Server Caching Proxy provides can be implemented independently of the WebSphere Everyplace Server components to optimize other component's performance. This is called in-stream caching.
- The Caching Proxy is a prerequisite component for some Everyplace Server components.

In-stream caching

The cache component of the Caching Proxy is beneficial where the *same content* is repeatedly retrieved from a slow source. The Caching Proxy can be positioned in front of a slow source (perhaps a Web server inside the domain), or in front of a whole network (such as the Internet) to generally optimize access to any site.

Figure 7-3 shows where the Caching Proxy can be used for in-stream caching of HTTP traffic flowing within an Everyplace Server domain:

1 Caching proxy as an in-stream cache

Application enhancing

The caching component of the Caching Proxy can be used to enhance the performance of the following components:

2 IBM Everyplace Wireless Gateway. The IBM WAP Gateway installs a special plug-in module on the Caching Proxy to gain improved performance with caching binary WML.

3 WebSEAL-Lite is implemented as a Caching Proxy plug-in.

4 IBM WebSphere Transcoding Publisher. WebSphere Transcoding Publisher installs a special plug-in on Edge Server Caching Proxy to gain improved performance by caching transcoded data. In addition, WebSphere Transcoding Publisher uses a fragmentation cache that is not implemented using the Caching Proxy’s cache.

5 Location Based Services is implemented as a Caching Proxy plug-in.

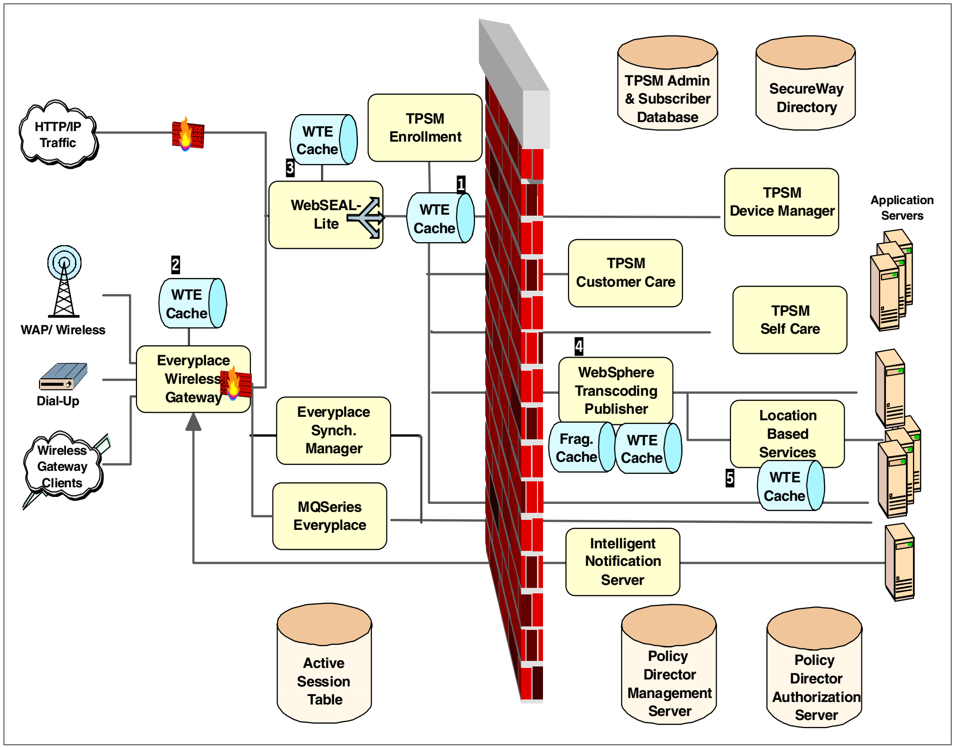


Figure 7-3 Everyplace Server architecture - caching components

7.2 Scalability by component

This section describes the built-in scaling characteristics of each individual component of IBM WebSphere Everyplace Server.

The components described are:

- Wireless Gateway
- WebSEAL-Lite
- Edge Server Caching Proxy
- Edge Server Load Balancer
- WebSphere Transcoding Publisher
- Tivoli Personalized Services Manager
- Intelligent Notification Services
- Location Based Services
- Tivoli SecureWay Policy Director
- IBM SecureWay Directory
- Active Session Table Server

- Everyplace Synchronization Manager
- MQSeries Everyplace

7.2.1 Everyplace Wireless Gateway

Better performance and high availability can be achieved by using Everyplace Wireless Gateway's cluster support. Having multiple Wireless Gateways in one cluster can improve the service response time and minimize the service downtime.

In order to handle high traffic networks, Wireless Gateway incorporates clustering. Clustering, in the context of the Everyplace Wireless Gateway, is the grouping of several machines (each running one instance of the Wireless Gateway), as illustrated in Figure 7-4. Workload from one or more network connections is distributed across them. Each node, in addition to taking advantage of UNIX multiprocessor capabilities, runs its own WAP protocol stack. Multiple clusters may exist on the same physical set of boxes.

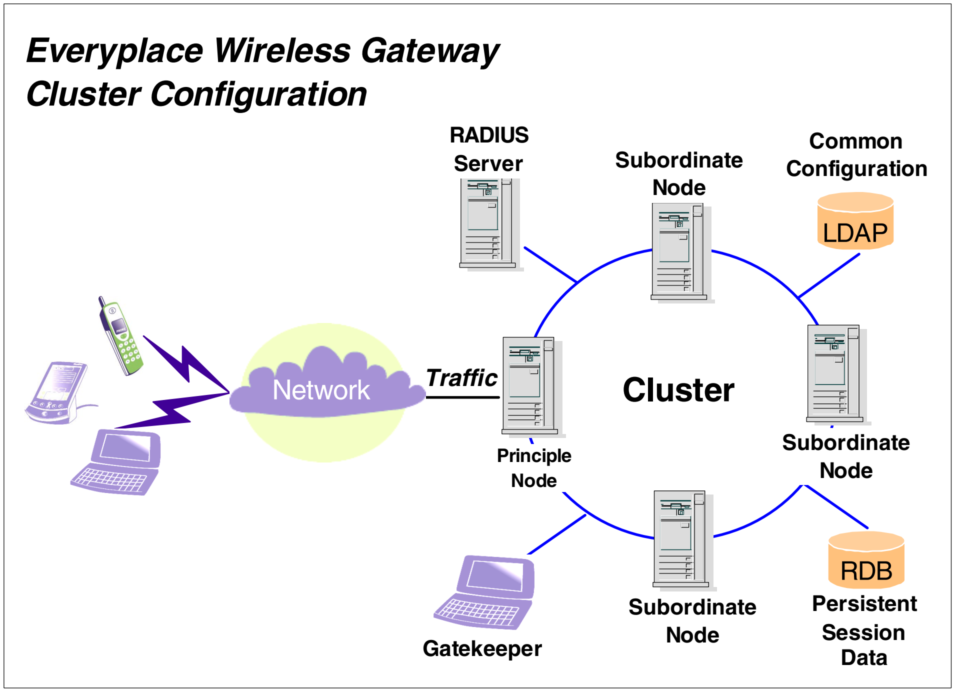


Figure 7-4 Everyplace Wireless Gateway cluster configuration

Everyplace Wireless Gateways share three major resources that make clustering possible:

1. A relational database (such as DB2 or Oracle) that contains persistent session data.
2. An LDAP directory server that contains gateway configuration and user data.
3. A single point of administration: the Wireless Gatekeeper. Wireless Gatekeepers are managed by the Gateway's Access Manager (AM). An Access Manager supports the many-to-many relationship between Gatekeepers and Gateways. The Access Manager insures that one Gatekeeper locks administration of the Gateway topology, in effect preventing overwriting of other configuration modifications by forcing the other Gatekeepers into *read-only* mode. The Access Manager communicates between the Gateway to Gatekeeper using XML messages.

Figure 7-5 shows all the components of a Gateway cluster in context. Tivoli Personalized Services Manager, the application servers and the Web servers are all accessed through the WebSphere Everyplace Server. The Wireless Gateway itself communicates with the Gatekeeper, a relational database, LDAP directory, and with outside networks via Mobile Network Connections (MNCs).

‘

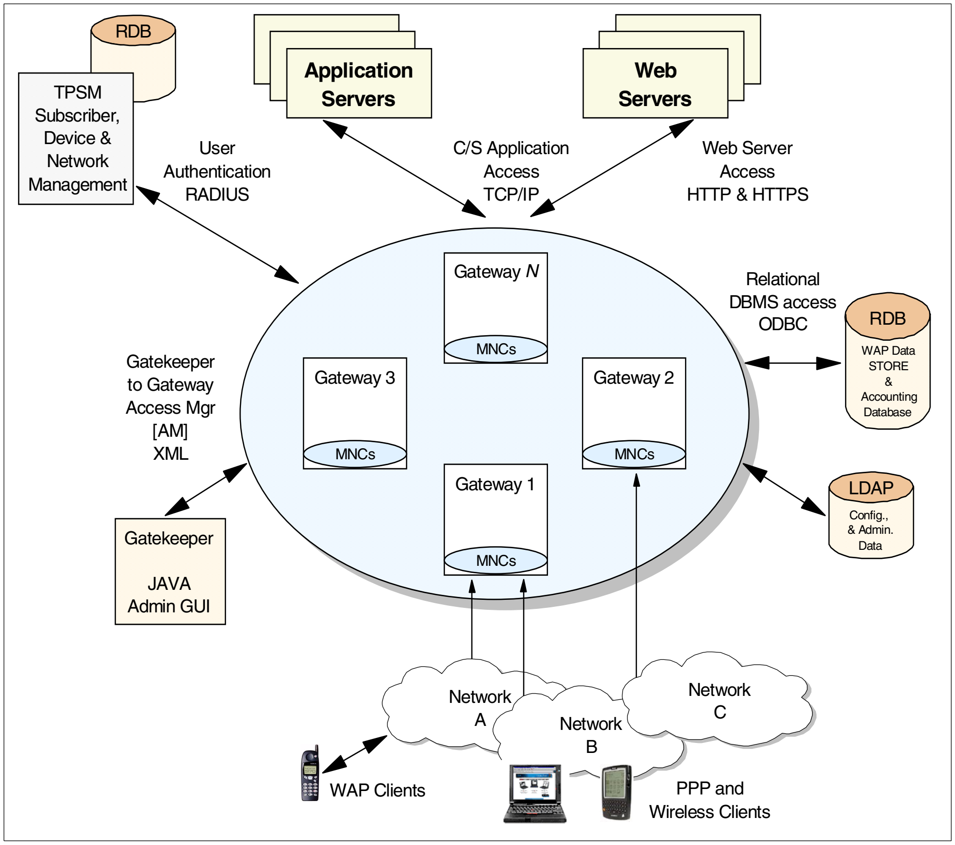

RDB
TPSM Subscriber, Device & Network Management
Application Servers
Web Servers
User Authentication RADIUS
C/S Application Access TCP/IP
Web Server Access HTTP & HTTPS
Gateway N
MNCs
Gateway 3
MNCs
Gateway 2
MNCs
Gateway 1
MNCs
Relational DBMS access ODBC
RDB
WAP Data STORE & Accounting Database
Gatekeeper to Gateway Access Mgr [AM] XML
Gatekeeper JAVA Admin GUI
LDAP
Config., & Admin. Data
Network A
Network B
Network C
WAP Clients
PPP and Wireless Clients

Figure 7-5 Distributed IBM Everyplace Wireless Gateway servers

The Cluster Manager is a function of the Wireless Gateway that allows for Gateway-to-Gateway communication in order to share workload and balance CPU usage in a cluster.

The Cluster Manager allows nodes to be dynamically added and removed from a cluster. To introduce a node to a cluster, configure the Gatekeeper with the node’s IP address and port number. The Cluster Manager will then begin to distribute requests to the new node. To remove a node, delete it from the Gatekeeper. Traffic will then be distributed over the rest of the cluster.

The Wireless Gateway Cluster Manager can be configured in one of three ways:

1. As a *principal* node:
   - Receives network traffic from an MNC and distributes load to itself and *subordinate* nodes
   - Maintains two-way communication to *subordinate* nodes

  - Initiates communication and may send a shutdown notification
  - Receives CPU performance load information from *subordinate* nodes
  - Controls performance-based load balancing
2. As a *subordinate* node:
  - Starts accepting or stops accepting data flow
  - Sends performance load information to *principal* node(s)
3. As both *principal* and *subordinate:*
  - There can be only one *principal* node per cluster. However, there may be multiple clusters, each with one principal and a defined cluster group of *subordinate* nodes.
  - *Subordinate* nodes can belong to more than one cluster group.
  - A Gateway node may participate as a *principal* node for a cluster group while also being a *subordinate* node in one or more cluster groups.

The *principal* node in a cluster group has the potential for being a single point of failure between a network and its associated cluster. If a *subordinate* node fails, its workload is redistributed among the rest of the cluster, simply increasing the load on the other nodes. If the *principal* node fails then the communication between *principal* and *subordinate* nodes ceases as well as the connection to any networks that interface to that machine. The *subordinate* nodes, as a result, will discard current stack processes.

The scenario in Figure 7-6 shows three networks: A, B, and C, and four Wireless Gateways (WG): 1, 2, 3, 4 respectively. WG1 has an MNC to network A while WG2 has MNCs to networks A and B. In Figure 7-7, WG1 is the *principal* node with WG3 and WG4 being its *subordinates*. WG1 receives network traffic from A and distributes it over WG3, 4 and itself. WG2 does not participate in this cluster.

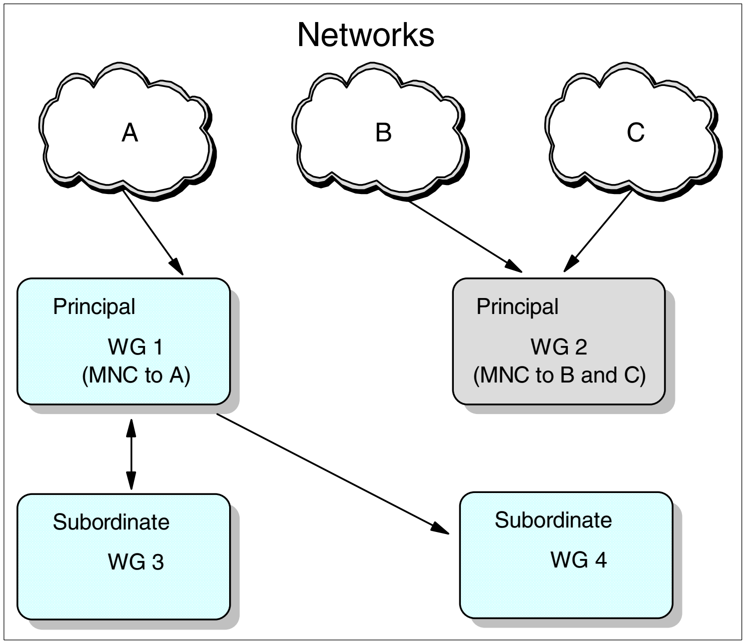


Figure 7-6 Wireless Gateway Clustering Scenario 1

The scenario in Figure 7-7 presents the second case. This time WG2 is the *principal* node. It receives traffic from both B and C and distributes it over WG1, 2, 3 and itself. WG1 is a *subordinate* node in this cluster scenario.

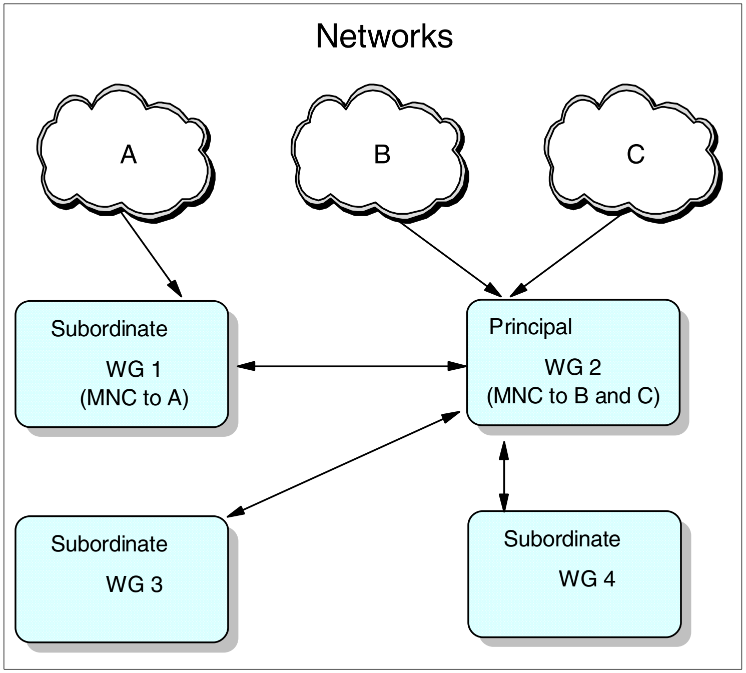


Figure 7-7 Wireless Gateway Clustering Scenario 2

The Everyplace Wireless Gateway allows multiple simultaneous connections to multiple networks. Each physical connection is represented in the Everyplace Wireless Gateway as a separate MNC. Everyplace Wireless Gateway balances the load of messages across similar MNCs.

The Everyplace Wireless Gateways configuration information is stored in an LDAP directory database. IBM SecureWay Directory is an LDAP directory that is shipped with the WebSphere Everyplace Server. The IBM SecureWay Directory server can reside on any host protected in the WebSphere Everyplace Server domain and can store information about more than one gateway. Using the directory, resources can easily be shared among multiple Wireless Gateways. In addition, the Wireless Gateway can be configured to use a second or replica SecureWay Directory if the master is not available.

The session data is stored in a relational database such as DB2. Session information is about the user activities during establishment, maintenance, and release of any connection to the Wireless Gateway. The database can reside on any host within the WebSphere Everyplace Server domain and can store session information for multiple Wireless Gateway servers.

In addition, the Everyplace Wireless Gateway stores the accounting and billing information in either flat files on the host server, or on a relational database such as DB2. Accounting and billing information is about the flow of packets and can be used by other software applications to construct billing information for the subscribers.

Everyplace Wireless Gateway deployment

The Everyplace Wireless Gateway can be installed on a single server along with other products, or it can be installed separately on a dedicated server in a distributed format. IBM recommends using the distributed form, and separating security services, content adaptation, management services, base services, and connection services.

To achieve high performance in providing WAP and wireless services to clients, IBM recommends that you use the Everyplace Wireless Gateway in conjunction with the Edge Server Caching Proxy. The Edge Server Caching Proxy HTTP proxy can enable both Web and WAP content caching for faster services.

The Edge Server Load Balancer is not recommended to be used in front of the Wireless Gateways. This is due to the fact that the Load Balancer is based on TCP/IP while the Wireless Gateway supports many non-IP networks, such as WAP bearer networks. When there are non-IP connections, the Load Balancer's load balancing feature is no longer applicable.

In the general WAP case, it is *not* technically possible for the Edge Server Load Balancer to run in front of WAP gateways. Edge Server Load Balancer is based on TCP/IP while non-IP protocols such as X.25 and WAP protocol include support for non-IP wireless networks, which is not supported by the Edge Server Load Balancer. When there are no TCP connections, the Edge Server Load Balancer's load-balancing feature is lost. In addition, WAP connection-oriented sessions and WTLS secure transactions may not be efficiently dispatched by the Load Balancer. Moreover, for WAP gateways that do not share WAP persistent data among multiple gateways or WAP stacks, there would be further restrictions for using the Load Balancer. The WAP session suspend and resume function is not reliably supported because the client will likely resume with a different IP address.

Everyplace Wireless Gateway's Messaging Gateway

The Messaging Gateway can reside on a server that is part of a cluster, but its performance is not improved by the Everyplace Wireless Gateway clustering architecture. However, Edge Server Load Balancer may be used to scale messaging over multiple wireless gateways.

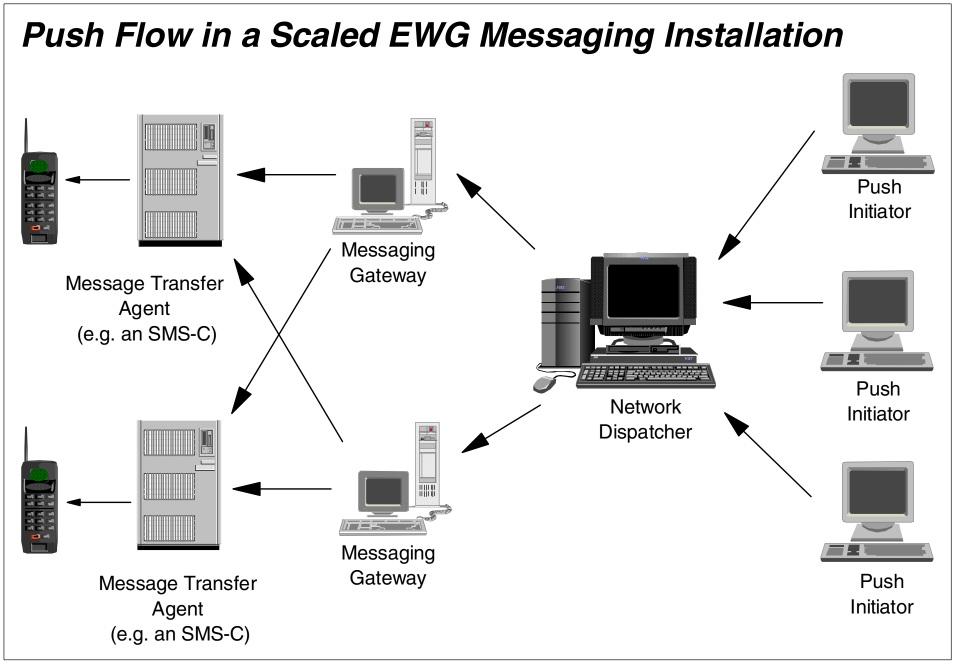

Push Flow in a Scaled EWG Messaging Installation
Message Transfer Agent (e.g. an SMS-C)
Messaging Gateway
Network Dispatcher
Push Initiator
Push Initiator
Push Initiator
Messaging Gateway
Message Transfer Agent (e.g. an SMS-C)

Figure 7-8 Scaling the Wireless Gateway 's Messaging Gateway

There is a known limitation when using Load Balancer and the Messaging Gateway in this fashion. Status queries and cancel requests should be sent to the same gateway that handled the original request. However, when using Load Balancer to balance load among gateways, status queries and cancel requests may be routed to a messaging gateway other than the one that handled the original request. In this case, status queries and cancel requests may incorrectly return failure codes. The Intelligent Notification Services component does not currently create status queries or cancel requests.

For scalability and the provision of virtual private networks (VPN) and interfacing to existing corporate or subscriber access control, a RADIUS server can be used by the gateway to authenticate the users to the WebSphere Everyplace Server. A RADIUS server is shipped with WebSphere Everyplace Server. The Wireless Gateway can communicate with Tivoli management tools in order to send SNMP traps (alerts).

7.2.2 WebSEAL-Lite

WebSEAL-Lite acts as the entry point for all client requests to the Everyplace Server. It is also the base URL of the Everyplace Server domain. In other words, when clients connect to the Everyplace Server, they connect using the WebSEAL-Lite's host name.

For large user-base applications, multiple WebSEAL-Lites will be needed. This can be achieved by using techniques such as the WebSEAL-Lite clustering. Multiple WebSEAL-Lites can be clustered using the Edge Server Load Balancer to provide for upward scalability as the number of concurrent users increases.

If a Load Balancer is positioned in front of a cluster of WebSEAL-Lite servers, the Load Balancer then becomes the site IP address to which your clients send all requests.

Figure 7-9 illustrates an architecture wherein a Load Balancer is deployed to balance requests for a WebSEAL-Lite cluster. In this example, if a client wishes to access an Everyplace Server service, then the client will connect to `http://wes.com`. Client requests are received by the Load Balancer, which will determine which WebSEAL-Lite in the cluster is best placed to handle the client authentication request.

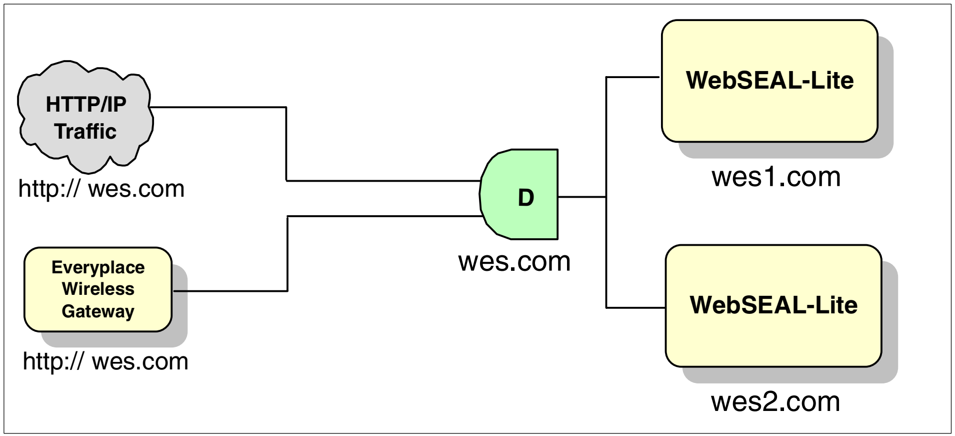


Figure 7-9 Using the Load Balancer with WebSEAL-Lite

Affinity can also be used with the WebSEAL-Lite. Once authenticated, all subsequent user requests can be directed through the same WebSEAL-Lite through the use of the affinity feature that the Load Balancer provides. With the sticky option configured, subsequent connection requests from a client will be dispatched to the original WebSEAL-Lite until a configurable time-out value expires.

If you do not configure the affinity then each subsequent client request will have to go directly to the Active Session Table Server to be validated, if it is not dispatched to the WebSEAL-Lite that still has that client's information in its local cache.

WebSEAL-Lite maintains its copy of the Policy Director authentication information on disk. Therefore, ensuring that this is on high-speed disk will help improve overall WebSEAL-Lite authentication performance.

7.2.3 Edge Server Caching Proxy

This section describes in more detail the caching capabilities available within the Edge Server Caching Proxy.

Caching rules

Technically all Web content retrieved from a content server can be cached. However, caching Web content is ideally suited to static Web content, since it generally does not change that frequently.

The cache component of the Caching Proxy allows you to customize the caching of Web content. Caching rules enable the administrators to determine what should be cached, how long it should remain in the cache, and what should not be cached. A caching agent can also be used to preload into cache specified files.

One of the more important functions that the cache provides is to ensure that the cached content remains current and consistent with the original data on the content server. The Caching Proxy achieves this using passive caching. For each file that is cached, the Caching Proxy computes an expiration time. If the Caching Proxy receives a request for a file and the file has expired, it will issue a request to the content server to check if the expired file has changed. If the file remains consistent with the copy in cache, the Caching Proxy will serve the file from cache. If the content server replies that the file has changed, the Caching Proxy will retrieve the up-to-date file from the content server, cache it, and return the new file to the client.

The cache component of the Caching Proxy also provides a garbage collection feature. The garbage collection maximizes Caching Proxy's use of the cache by deleting expired, old, or irrelevant data to make space available for new cached content.

Refer to the *WebSphere Edge Server for Multiplatforms Administration Guide,* GC09-4567 for full details of the caching and caching rules, and how they can be used and configured.

Caching storage options

The speed of the cache storage devices is critical to performance on the Caching Proxy. The speed at which files are returned from the cache is generally determined by the method of storage. The Caching Proxy can now use three types of cache storage:

- **Memory:** Caching to memory gives the fastest processing, but the size of the cache is limited to the amount of memory (RAM) on the proxy server.
- **Disk:** cached files can be stored on disk. A disk cache can be made up of one or more disks partitions, and is slower than a memory cache but allows larger cache sizes.
- **Files:** cached files can be stored in files. A cache made up of one or more files can be slowest because it requires the proxy server to use the operating system's file system to cache the file.

Caching agents

With time and increased use, a Web server's performance and response rates will improve when the cache is efficiently used. However, the first client request for a file on a content server will not benefit from caching.

The Caching Proxy provides the ability to preload certain files. The Caching Proxy has a *cache agent* that provides this service. The cache agent can automatically retrieve specified URLs, or the most popular URLs, and place them in the cache before they are requested.

There are two ways to specify the files to the cache agent. Both options also specify a limit on the number of URLs that are retrieved.

- **Specific URLs** - allows the administrator to control what files will be delivered faster to the client.
- **Cache Access Log** - used for logging hits on the proxy server. The cache agent loads this file, sorts the URLs by frequency of requests and then retrieves the most frequent requests.

The administrator may determine when the cache agent should be run such as daily and at specific times.

Clusters of Caching Proxies

In large networks, one caching proxy might not be enough to meet the performance and availability requirements. If your Web site is heavily accessed, there can be greater demand for its contents than a single Caching Proxy can handle. This can degrade the Caching Proxy's performance, and therefore your Web site's performance.

Another potential problem with a single Caching Proxy is that it represents a single point of failure - if it fails or becomes inaccessible because of a network failure, users cannot access the Internet or any hosted Web sites until the Caching Proxy is reinstated.

By using multiple Caching Proxies to form a caching cluster, the cluster can serve more users with reliable response time and provide load balancing with fault tolerance. The Caching Proxy does this by evenly distributing cached content among servers in the cluster. If one server in the cluster fails, other servers in the cluster will continue to provide Web content caching. To enable high availability, the Caching Proxy must be used in conjunction with the Edge Server Load Balancer component of the Edge Server.

Figure 7-10 depicts a configuration in which the Load Balancer balances the load across a cluster of two Caching Proxy machines.

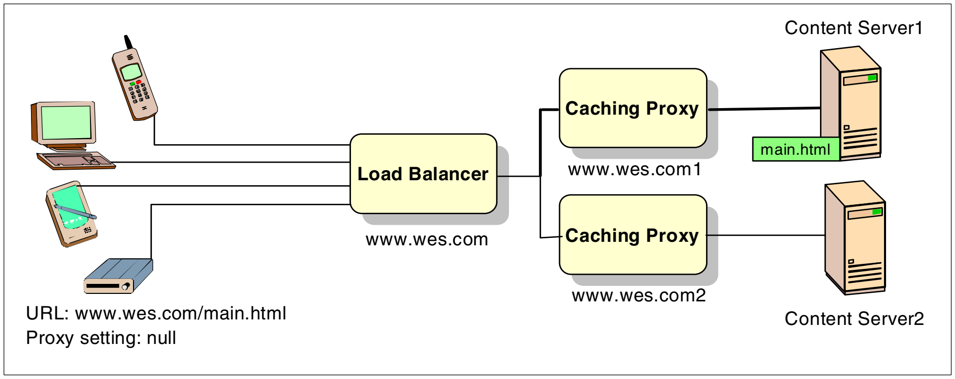


Figure 7-10 High performance with the Caching Proxy and the Load Balancer

The Load Balancer is configured with the cluster's dedicated host name and IP address. Client browsers are configured to direct their requests to the cluster host name. When, for example, a client requests a file main.html that resides on Content Server1, the client directs its request to the cluster host name or address of the Load Balancer, which in turn directs it to the appropriate Caching Proxy. The Caching Proxy will then regenerate the request, pass it to the content server as either explicitly stated in the URL or based on the proxy configuration directives. The Caching Proxy will retrieve the response from the content server and cache it, then return the response directly to the client.

Using multiple Caching Proxies introduces a potential inefficiency, in that more than one Caching Proxy can end up caching the same file if different end users request the file via different Caching Proxies. Each proxy within the proxy cluster will have its own cache. Over time the cache on each node will accumulate the same cache content. Since the cached data cannot be shared, this results in a waste of cache storage as multiple copies of the cached data are stored. This can negate the effectiveness of the Caching Proxy when deploying multiple instances of the Caching Proxy.

Using Remote Cache Access

Remote Cache Access (RCA) addresses this issue by allowing multiple Caching Proxies to share the content of their caches. Using each individual Caching Proxy, RCA allows multiple proxy servers to co-operate to form cache arrays, to create a larger combined logical cache. With RCA cache arrays, each Caching Proxy server knows what files each proxy server has in its cache, and therefore which server in the array is best suited to process the incoming request.

When a client request is received by a Caching Proxy in the cluster, RCA uses the Cache Array Routing Protocol (CARP) to query all the servers' caches in the array determining which server in the array has the requested file in its cache. Using RCA, a proxy server can immediately route the client's request to the proxy server in the array that contains the cached data. If the receiving proxy server does not have the cached data in its cache, it will route the request to the next proxy server in the array. If RCA determines that the file is not contained in the combined, logical cache, the proxy server processing the request will retrieve the file from the content server directly, caching it locally before returning it to the requesting client.

Due to the sorting of requests through these proxy servers that RCA provides, duplication of cache contents is eliminated, and cache space is saved, cache hit rates are improved, and network bandwidth is used more efficiently.

Using proxy chaining

For large-scale implementations, Caching Proxy clusters can be chained together to improve availability and throughput. In a large implementation, there may be multiple points of access to the network. Each point is a place where a Caching Proxy can be placed and configured to point to one parent Caching Proxy.

If a proxy server in the lowest level of a hierarchy, or chain, cannot serve a requested URL from its cache, it forwards the request to the proxy server that has been configured as the next in the chain. The proxy server at the highest level of the chain then determines if it has the requested files stored in its cache. If not, it will retrieve the requested files from the content server. This high-level proxy server will pass the response back down the proxy chain to the client. Each proxy in the chain can then store the response in their local caches.

Proxy chaining offers the following advantages:

- Proxies at lower levels, closer to the client that originated the request, benefit from the caches of the higher-level proxies.
- Proxy chaining reduces the load on the highest-level proxy and ultimately on the content Web server, since lower-level proxies may already have the document cached.
- The larger the number of users, the higher the probability that the proxy server already has the document in its cache.

For more information on proxy chaining refer to *IBM WebSphere Performance Pack: Caching and Filtering with IBM Caching Proxy,* SG24-5859.

7.2.4 Edge Server Load Balancer

A discussion of the functions provided by Load Balancer is limited to the context of this book. Details on the full capabilities and configuration options provided in the Load Balancer can be found in:

- The online InfoCenter product documentation

  http://www-3.ibm.com/pvc/products/wes_provider/infocenter/index.html
- *Network Dispatcher Administration Guide,* GC31-8496
- *WebSphere Edge Server for Multiplatforms Getting Started Guide Version 1.0*, SC09-4566
- "IBM Network Dispatcher Version 3.0 Scalability, Availability and Load Balancing for TCP/IP applications", found at:

  ftp://ftp.software.ibm.com/software/network/dispatcher/whitepapers/nd30whitepaper.pdf
- *WebSphere Edge Server: Working with Web Traffic Express and Network Dispatche*r, SG24-6172
- http://www.ibm.com/software/webservers/edgeserver/library.html

As the number of client requests for Web content increases, so the load that the content server has to process increases. As this acceleration continues, the content server can become unable to handle the load and client requests are either rejected or delayed. In such a case, the ability of your content server to receive client requests and process them effectively becomes a problem.

A solution to this problem is to introduce horizontal scaling. By adding another, or replica, content server, you can spread the load over the multiple content servers. Scaling in such a way can directly improve the performance of high-demand content servers. However, increasing the number of machines that your organization has to deliver Web content is not enough to achieve your availability and performance expectations. The client requests must be intelligently managed and balanced over these replica machines.

The Load Balancer can be implemented in front of a cluster of content servers to efficiently and effectively balance the load of client requests.

The Load Balancer consists of three components that can deployed independently or combined to produce a powerful tool for ensuring that your Web application and Web content server is available and scalable:

- The Dispatcher
- Interactive Session Support (not discussed in this book)
- Content based routing

The Dispatcher

The Dispatcher is an IP packet-level load balancer to balance requests from TCP or UDP protocols. The Dispatcher will distribute requests among HTTP, FTP, SSL Telnet, NNTP, POP3, SMTP or other TCP-based servers or stateless UDP-based servers. Using any of these protocols the Dispatcher provides server and application load balancing. It provides high-performance, low-latency load balancing using weights and measurements that are dynamically set or predefined to ensure that requests are routed to the optimal server.

The Dispatcher balances incoming requests and distributes them to the content servers it services. When the Dispatcher receives a request for content on the content servers it is balancing, it does not process the request but rather, it forwards the request to the content server that is currently best able to fulfill the request. As Figure 7-11 shows, having forwarded the client request to the optimal server, the content server will respond directly to the client without passing back through the Dispatcher.

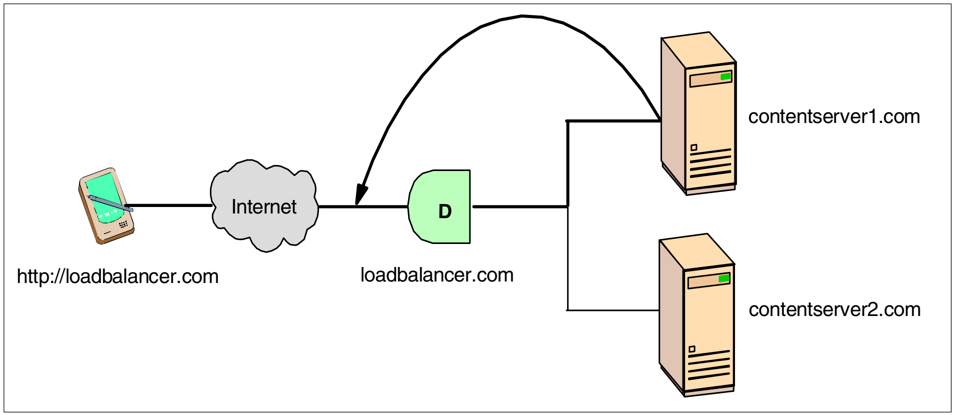


Figure 7-11 How Load Balancer works

The Dispatcher uses three subcomponents to determine the optimal server in a cluster:

- The Executor, which provides the Dispatcher's core functionality. The Executor will examine the header information of each request to decide whether the request belongs to an existing connection or whether it represents a new connection request. If the connection already exists, then the request is forwarded to the same server chosen on the initial connection request. If the packet is a new connection request, the Executor will look at the stored weights for each server in the cluster to determine the best server to forward the connection request to.
- The Manager, which periodically sets the weights that the Executor obeys. The Manager sets weights based on internal counters in the executor and feedback provided by the advisors. The internal counters will, for example, provide information on the number of active and new connections on each server. Alternatively you can define your own weights. The Executor will then use these weights to perform load balancing.
- The Advisor. Advisors can be used to collect and analyze feedback from individual servers regarding the health of the servers and any application running on the servers being load balanced. The Advisor will provide this information to the Manager component. See "Edge Server Load Balancer" on page 351 for more information about Advisors.

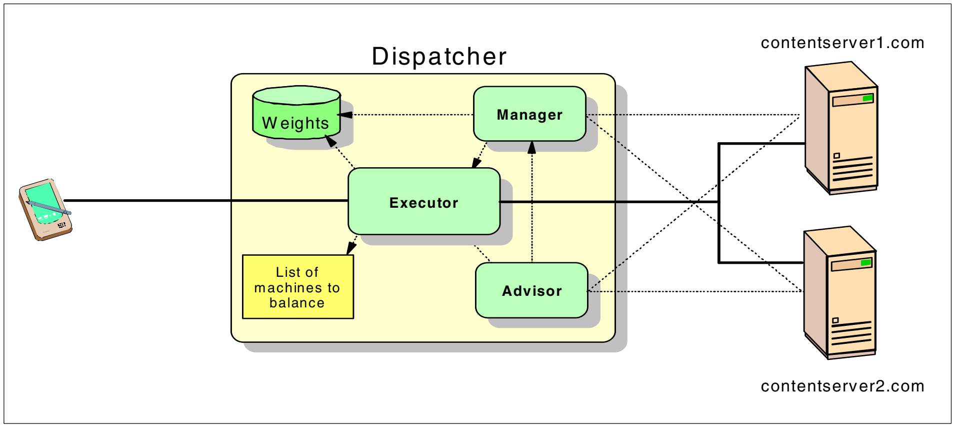


Figure 7-12 How the Dispatcher determines the optimal server in a cluster

For more information on the capabilities and configuration of the Dispatcher refer to the *Network Dispatcher Administration Guide,* GC31-8496.

Affinity

When the Dispatcher receives requests, it will select the optimal server at that time to handle the request. When subsequent requests come from the same client, the Load Balancer treats them as unrelated TCP/IP connections and again selects the optimal server. The overhead in creating the TCP/IP connection is high, and the resources in doing so could be more efficiently used to handle new connection requests.

This server connection independence can be a problem for some client connections that need to hold their client session information on the server, or keep state information in memory or on local disk. A client's initial connection to a server will create a session, stored on the server, which contains session information required by both the client and server. Subsequent connections from the same client may go to another server in the cluster and the client cannot access the session information.

The Load Balancer provides functions that can be used to maintain such a relationship or affinity. Affinity can also be used to ensure that the client continues to be load balanced to the same server for some period of time. Using affinity you can configure a *sticky bit* between client and server. Affinity can be implemented for different Load Balancer components. Affinity can be applied to the Dispatcher, or content based routing components.

However, sticky bit affinity is based on the client’s IP address. If the Load Balancer is placed behind a reverse proxy server (such as WebSEAL-Lite), all requests coming to it will have the same client IP address. In this situation, sticky bit affinity is not appropriate.

Content based routing

Content based routing (CBR) performs application load balancing by distributing a Web site’s load among servers according to the content of browsers requests.

Content based routing combines the load balancing, manager and advisor functions of Dispatcher with the Caching Proxy content filtering functions to permit load balancing based on the content of HTTP, POP3 and IMAP requests. Caching Proxy rules can be written to load-balance client requests over different sets of servers in a cluster based on:

- Client IP address
- Entire URL
- Protocol portion of the URL
- Host portion of the URL
- Path portion of the URL
- Referrer HTTP header
- User-Agent HTTP header

For HTTP requests, content based routing provides the ability to proxy requests to specific servers based on the content requested. Using the rules above, content based routing can direct and balance requests for different content to content-specific servers. For example, if the Load Balancer receives a request for a URL for a page that is made of WML and JSPs, content based routing can determine that the content of the request contains WML and JSP files. The Load Balancer will then direct the WML request to a server dedicated to handling WML, and redirect the request for the JSP to a dedicated application server.

This differentiation based on any criteria provides you the ability to classify a request, for example as a frequent buyer, and direct it to a high-capacity server or give access to specialized Web content.

Rules-based content based routing can be used with HTTP protocol on port 80 and 443.

For POP3 and IMAP requests, content based routing is a proxy that chooses an appropriate server based on user name and password provided by the client.

For more information on the capabilities and configuration of content based routing, refer to the *Network Dispatcher Administration Guide,* GC31-8496*.*

Cookie affinity

Load Balancer's content based routing function supports cookie affinity. The cookie affinity feature applies only to content based routing with the Caching Proxy, which supports load balancing based on rules. This combination of components provides a new way to make clients *sticky* to a particular server.

With cookie affinity enabled, the server that first serviced an end user's request is recorded in a special packet of data (a cookie) included in the server's response. When the end user accesses the same URL again within a period of time that you define, and the request includes the cookie, content based routing routes the request to the original server rather than reapplying its standard rules.

Once a rule has been enabled for cookie affinity, new client requests are load-balanced using standard content based routing algorithms while succeeding requests from the same client are sent to the initially chosen server. The chosen server is stored as a cookie in the response to the client. The cookie is then inserted in the headers that go back to the client, and if the client's browser is configured to accept cookies, it sends back subsequent requests. As long as the client's future requests contain the cookie, and each request arrives within the stickytime interval, the client maintains affinity with the initial server.

The following demonstrates one way the process might work in a system with multiple Transcoding Publishers.

1. A WAP phone requests a URL
2. The Load Balancer, CBR+cookie affinity, dispatches the request to Transcoding Publisher #1
3. Transcoding Publisher #1 serves the request, fragments the resource, and sends the first fragment to the microbrowser
4. The WAP phone requests the second fragment
5. CBR+cookie affinity routes the request back to Transcoding Publisher #1
6. Transcoding Publisher #1 retrieves required fragment if it is still cached

How to enable cookie affinity with the **`rule set`** command:

```
rule set cluster:port:rule stickytime 60
rule set cluster:port:rule affinity cookie
rule set cluster:port:rule stickytime 300
rule set cluster:port:rule affinity clientip
```

Stickiness is set per rule. The default for rule affinity is the client IP, so if you haven't set it to `cookie`, you do not need to set it to `clientip`.

7.2.5 WebSphere Transcoding Publisher

As the number of users accessing the Everyplace Server domain from different client devices grows, the demands for transcoding of application content to suit the different devices will increase. The actual process of transcoding Web content is a highly compute-intensive activity, and as the number of requests for transcoding increases WebSphere Transcoding Publisher's time is spent transcoding rather than accepting requests for Web content on an application server. This delay is most problematic for Web content requests that do not require any form of transcoding.

It is most likely that you will want to install more than one instance of WebSphere Transcoding Publisher to let a heavy workload be spread across several physical units.

You can use transcoding both at the "front" and "behind" your application server as illustrated in Figure 7-13 below.

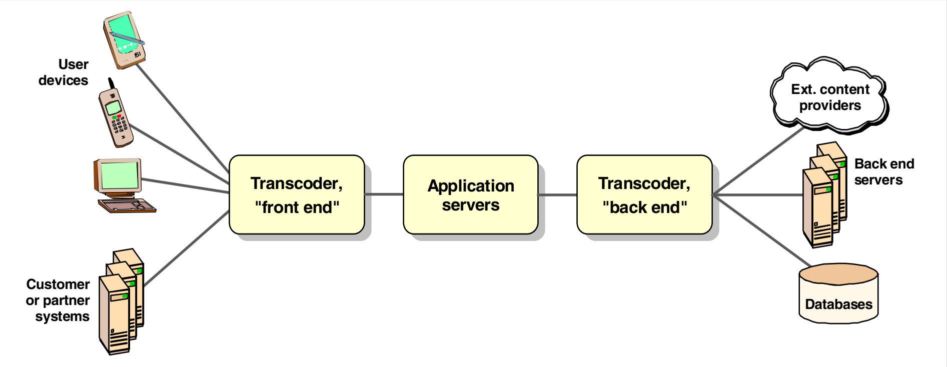


Figure 7-13 Transcoding in front of and behind an application server

In front, transcoding is used to adapt content to diverse clients and systems using your applications and services. This is the most obvious use of WebSphere Transcoding Publisher running as a proxy in the context of IBM WebSphere Everyplace Server.

The back-end transcoder helps adapt content from different sources to a format more easily handled by the application server. Several other options exist depending on application architecture, data formats, database types and other factors. Examples are IBM Enterprise Information Portal, IBM Host Publisher, WebSphere Transcoding Publisher used as JavaBeans (not supported by the Everyplace Server), IBM MQSeries Integrator, or IBM MQSeries Workflow.

The sample scenarios used throughout this book demonstrate use of both front-end and back-end transcoding. Examples in this chapter concentrate on parts of the transcoding required in such a setup.

Optimization with a Caching Proxy

All transcoding takes time and consumes hardware resources. It may be very beneficial to cache transcoded versions of content for reuse.

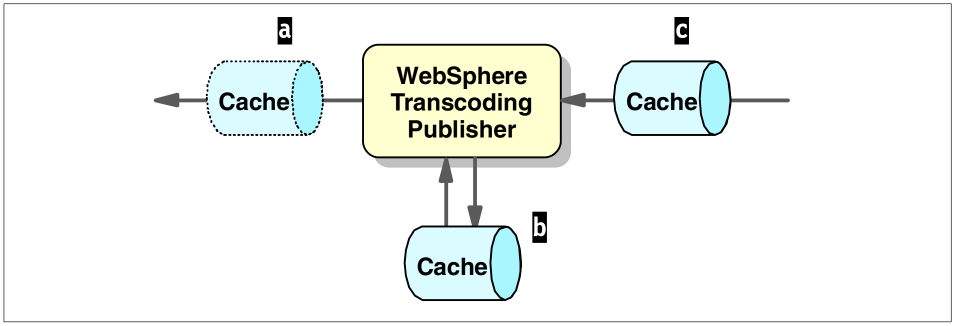


Figure 7-14 Caching with WebSphere Transcoding Publisher

WebSphere Transcoding Publisher can be configured to use an external cache directly (b in Figure 7-6 above). This allows caching of different transcoded representations of the same content element. In the WebSphere Everyplace Server environment, this cache would be Edge Server Caching Proxy. Note that, when configuring WebSphere Transcoding Publisher to use the Caching Proxy, you should specify the IP address of the cache server rather than the host name. If you enter a host name, WebSphere Transcoding Publisher will attempt to validate the host name through your Domain Name System (DNS) server. This may cause a slight delay in processing the address.

Be careful when setting up a Caching Proxy along the HTTP stream in front of WebSphere Transcoding Publisher a, as it will see the same URL for all appearances of an element; an image reduced in size or a page with adapted content will have the same URL. For example the Palm version of `index.html`, the Netscape version of `index.html` and the phone version of `index.html` share the same URL but different content. So a cache set up in front should be set up to cache non-transcoded content only.

Caching applied to a back-end transcoder either controlled by WebSphere Transcoding Publisher (marked b in Figure 7-6) and/or at the source c can provide a major benefit, in particular if dynamic content such as XML is provided from an external or other latent source.

A cache that is serving transcoded content and controlled by WebSphere Transcoding Publisher will obey the same HTTP directives as usual, such as *Expires*, *Last-Modified* and *No-Cache*. That also means transcoded content should be handled properly, considering both timeliness and optimal reuse.

A cache should never be considered just as a black box doing traffic optimization. Parameters such as what domains to cache, cache size, housekeeping settings, caching policies, and even filtering need to be tuned to fit your unique deployment and use.

Load Balancer with WebSphere Transcoding Publisher

The Load Balancer can be positioned in front of a cluster of WTP servers. It will accept requests from authenticated clients and dispatch the request to the optimal server. Figure 7-15 illustrates the hops from the WebSEAL-Lite to WebSphere Transcoding Publisher with the Load Balancer positioned in front to balance. In this scenario, the WebSEAL-Lite will direct user requests to the Load Balancer; this configuration is done in the Caching Proxy's ibmproxy.conf. The Load Balancer will then dispatch the request to WebSphere Transcoding Publisher, which in turn will retrieve the requested content, transcode the content if necessary and delivery the transcoded content directly to the WebSEAL-Lite, which will in turn forward the response back to the client.

Content based routing (CBR) could be used to forward client requests for transcoded content to a particular WebSphere Transcoding Publisher. For example you could forward all requests for WML to a WebSphere Transcoding Publisher server dedicated to transcoding these WML requests.

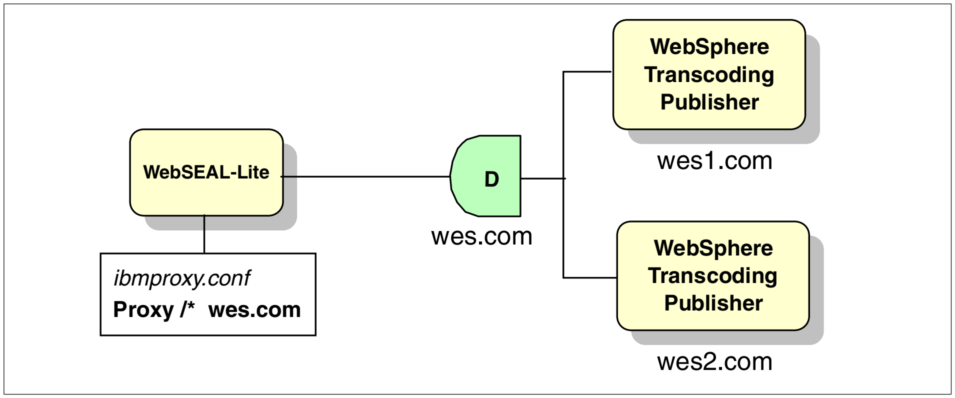


Figure 7-15 Load Balancer with WebSphere Transcoding Publisher

If WebSphere Transcoding Publisher is being used to perform HTML fragmentation on the device's behalf, the non-first fragments are stored in a Transcoding Publisher resource repository. In this case, the Load Balancer should be configured to ensure that subsequent requests are sent to the same WebSphere Transcoding Publisher instance; that is, the Load Balancer should be configured to use cookie affinity.

Note that when running WebSphere Transcoding Publisher behind another reverse proxy server (such as WebSEAL-Lite), both servers must be configured correctly to allow non-first fragments to be found. See "Configuring Load Balancer for WAP Devices" in the IBM WebSphere Everyplace Server InfoCenter for more information. The InfoCenter is located on the first CD of the distribution package or it may be found at the following IBM Internet site:
`http://www-3.ibm.com/pvc/products/wes_provider/infocenter/index.html.`
Note that this information must be adjusted if the WebSphere Transcoding Publisher is running behind a Load Balancer.

7.2.6 Tivoli Personalized Services Manager

Tivoli Personalized Services Manager is, essentially, a collection of Java-based Web applications. These applications run under the control of a Web application server.

For scalability, the individual Tivoli Personalized Services Manager applications may be separated onto multiple WebSphere Application Server instances.

If you need additional scalability beyond this, place WebSphere Application Server clusters behind Load Balancer and use Load Balancer affinity.

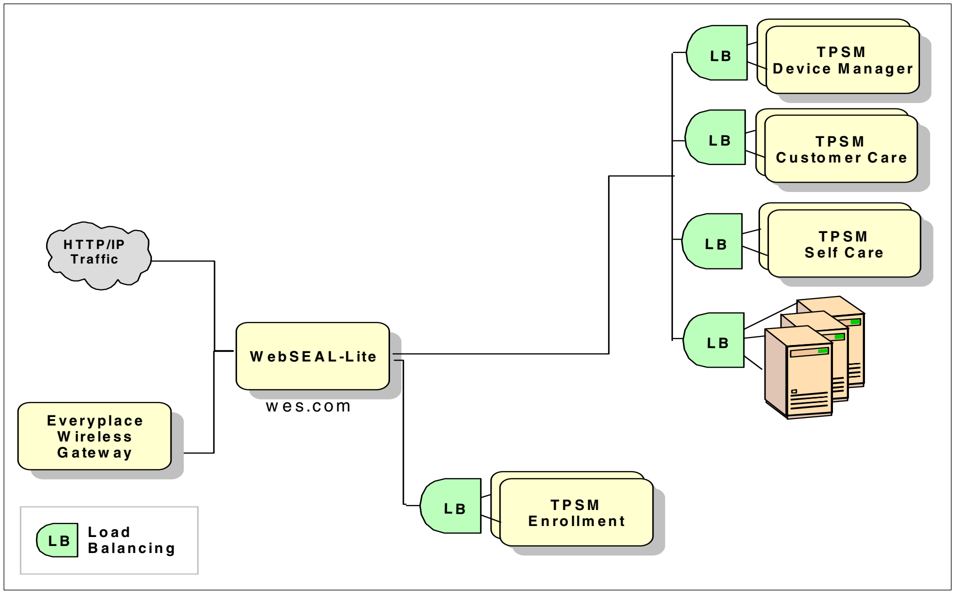


Figure 7-16 Load Balancer with Tivoli Personalized Services Manager Servers

In general, Tivoli Personalized Services Manager Enrollment resides in the DMZ, since users accessing the self-enrollment application will not yet be authenticated. The remaining Tivoli Personalized Services Manager components reside in the protected domain.

Refer to Redbooks such as the following for additional WebSphere Application Server scalability advice:

- *WebSphere V3 Performance Tuning Guide*, SG24-5657
- *Connecting WebSphere to DB2 UDB Server*, SG24-6219

For large subscriber bases, it is theoretically possible to partition the Tivoli Personalized Services Manager subscriber database across multiple Tivoli Personalized Services Manager instances; for example, on a realm basis. WebSEAL-Lite could be used to direct the request for a specific realm to a specific Tivoli Personalized Services Manager instance. These separate instances would all need to provision the same Policy Director Management Server, since only one is allowed per WebSphere Everyplace Server domain.

All components can be deployed on dedicated machines that are optimized to that specific use. For example a separate RADIUS server or a separate database server may be optimized for frequent, but small transactions. However, in most projects, the best solution is to deploy identical sets of software on the front-end and back-end machines.

At install time, the Setup Manager installs a copy of Tomcat, as the Web application server to be used by Tivoli Personalized Services Manager. It also installs a copy of WebSphere Application Server Advanced Edition. Scripts are available that allow you to migrate from using Tomcat to using WebSphere Application Server Advanced Edition as your Web application server. IBM strongly recommends doing this for production use.

Setup Manager installs a DB2 instance for Tivoli Personalized Services Manager to use for its subscriber database. However, Setup Manager installs WebSphere Application Server Advanced Edition using InstantDB for the internal WebSphere Application Server administration database. We strongly recommended that the WAS installation is migrated to DB2 (or Oracle) for the internal WebSphere Application Server administration database prior to using it in production. In addition, this approach allows you to move the WAS administration and Tivoli Personalized Services Manager databases to a separate database server, if there is enough database load to negatively impact performance.

The Tivoli Personalized Services Manager WebSphere Application Server uses IBM HTTP Server. This IBM HTTP Server should also be tuned.

Device Manager deployment

For load balancing and high availability, you may want to deploy multiple Device Manager servers in a Load Balancer cluster as illustrated in Figure 7-17.

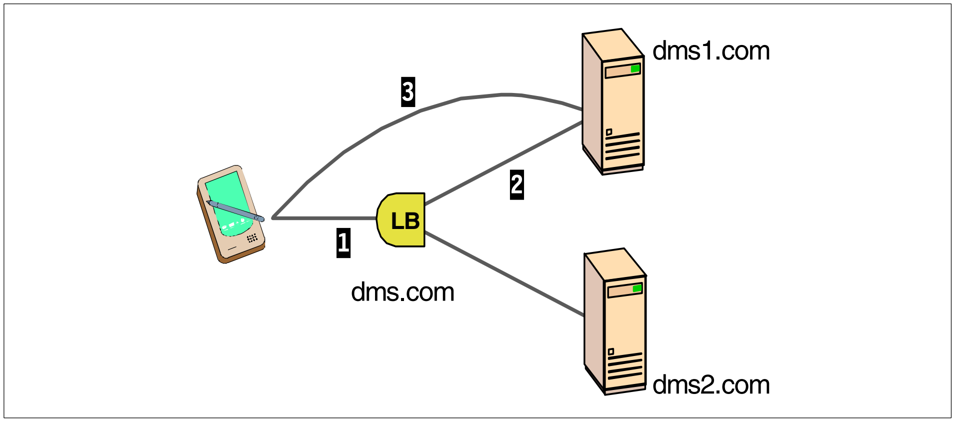


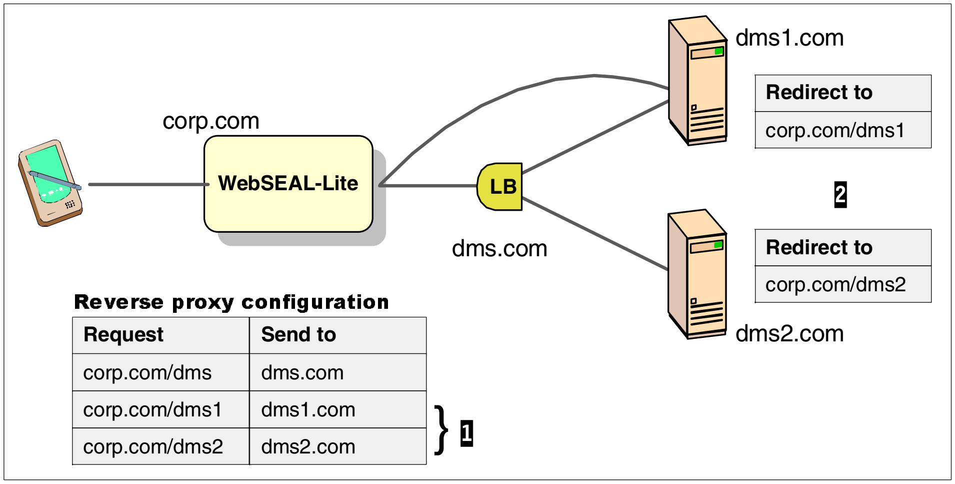
Figure 7-17 Device Manager redirections

With a Device Manager Server only the initial request is sent to the Load Balancer; the Load Balancer with the Caching Proxy is configured using proxy directives to redirect the subsequent requests directly to a Device Manager Server.

This is what will happen:

1. The Load Balancer will receive the initial request directed to the cluster address `http://dms.com`. The client sees that as its destination server.
2. The least busy server will be forwarded the request. In this example that is dms1.
3. To maintain the statefull session, dms1 will *redirect* the client to its own URL. This will make the client ask for `http://dms1.com` directly, bypassing the dispatcher. All subsequent communication will be between the client and dms1.

So, not only does this process maintain the necessary session state, but it also avoids the extra network latency by dispatching every request, which otherwise will be sent directly to the same server. However, in an Everyplace Server implementation you will set up the WebSEAL-Lite as a reverse proxy in front of all application servers. To work properly, both WebSEAL-Lite and Tivoli Personalized Services Manager need a little extra configuration, because the actual server addresses are hidden from users.



Reverse proxy configuration

Request	Send to
corp.com/dms	dms.com
corp.com/dms1	dms1.com
corp.com/dms2	dms2.com

Figure 7-18 Device Manager in Everyplace Server

As illustrated in Figure 7-18, you must configure two things to make this mechanism work in the Everyplace Server environment:

1. The WebSEAL-Lite must be aware of the actual server addresses and their URL defined to users outside.
2. The Device Manager must be “Everyplace Server aware” by letting it redirect to a specified URL, instead of using its host name.

The same consideration applies to any application server using client side HTTP redirections. To make it work in an environment with a central reverse proxy such as the WebSEAL-Lite in the Everyplace Server, configure it to direct those to the URL that is used externally.

7.2.7 Intelligent Notification Services

Since only one instance of each Intelligent Notification Services server is allowed in the WebSphere Everyplace Server domain, Intelligent Notification Services only scales vertically.

7.2.8 Location Based Services

Each Location Based Services server can only communicate with one location server at a time, although multiple Location Based Services servers could be installed to communicate with multiple location servers. Thus, only vertical scalability applies.

7.2.9 Policy Director

Policy Director allows only one Policy Director Management Server per domain. Thus, the Management Server can only be vertically scaled.

The Management Server, however, sustains little load, since its main purpose is to maintain the ACL and provide updates to the Authorization Servers.

To scale Policy Director, you should replicate Authorization Servers. Authorization Servers can be clustered behind a Load Balancer, or applications can be configured to access separate Authorization Servers to spread the load. In the WebSphere Everyplace Server environment, the WebSEAL-Lite and Location Based Services both act as Authorization Servers. You can also dedicate additional Authorization Servers for use by the back-end applications.

In addition, Policy Director improves scalability by allowing load balancing of incoming requests through smart junctions. This is what we recommend for the WebSEAL-Lite configurations.

Thus, to scale Policy Director in the WebSphere Everyplace Server environment, correctly size the WebSEAL-Lite configuration.

7.2.10 IBM SecureWay Directory

The IBM SecureWay Directory contains information about:

- Component configuration, which is generally only read at component start-up time;
- WebSphere Everyplace Server users, which is used for authentication. These requests would be the majority of the calls to the WebSphere Everyplace Server directory.

LDAP directory servers, including SecureWay Directory, scale through two mechanisms:

- Initially, by providing replicas for the master server. Components that only read the directory can be pointed to the replica. Components that update the directory in the normal course of operations, such as Policy Director and Tivoli Personalized Services Manager, should be configured to access the master.
- If desired, the overall directory space can be partitioned. For example, you may choose to store each realm or organizational unit in a different LDAP directory, and use referrals to direct requests for another organizational unit to the LDAP server that contains that information.

In WebSphere Everyplace Server, each component stores the LDAP server name, access user ID and password in its own configuration files. It is therefore possible to point a component to a replica of the master SecureWay Directory. This will work if the component either does not update the LDAP directory, or, is capable of following a referral to the master LDAP directory when it attempts to perform an update. All components that require installation of the SecureWay Directory client or are shipped with a built-in SecureWay Directory client (Tivoli Personalized Services Manager, Tivoli SecureWay Policy Director, WebSphere Transcoding Publisher, WebSEAL-Lite, and Intelligent Notification Services) fall into this category. Other components may also be able to accept referrals, however, we recommend that you test this on a per component basis.

Example 7-1 was taken from WebSEAL-Lite's configuration file, ibmwes.conf, in a non-Policy Director environment.

Example 7-1 non-Policy Director WebSEAL-Lite configuration file

```
# LDAP server will only be consulted if all of the following LDAP
server-related
# information is specified in this file: LDAP_Server, LDAPPort, baseDN,
LDAPAdmin
# and LDAPPasswd.  Either all of them must be specified, or none of them.

# Hostname of LDAP server
LDAP_Server rs61500.itso.ral.ibm.com
# TCP port on which the LDAP server listens
LDAPPort 389

# Base Distinguished Name for the directory tree
```

```
baseDN sys=SDP,dc=rs615001,dc=itso,dc=ral,dc=ibm,dc=com

# LDAP userid to use for authentication with the LDAP server
LDAPAdmin cn=root
# LDAP user's password in encrypted form
LDAPPasswd xxxxxx

# Base Distinguished Name for locating host-specific information in LDAP.
# (Follow the configPtr in the serviceName=svcwep1 entry below wepDN)
wepDN dc=rs615001,dc=itso,dc=ral,dc=ibm,dc=com
```

By modifying the above entries, the WebSEAL-Lite instance can be directed to a replica server.

Figure 7-19 shows a scenario with Tivoli Personalized Services Manager updating a master server, and WebSEAL-Lite reconfigured to access the replica.

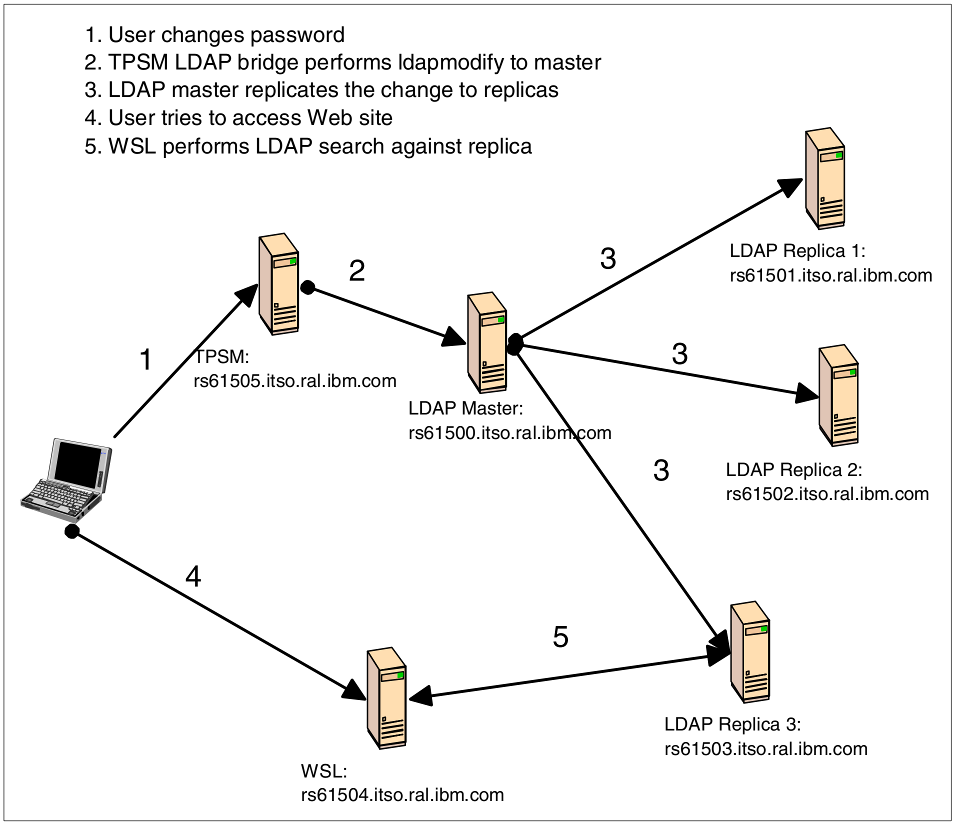


Figure 7-19 LDAP master and replica scenario

Tivoli Personalized Services Manager and Policy Director perform the majority of the updates to the directory. Therefore, they should be pointed to the appropriate master for the organizational unit that they maintain (if you have partitioned your Tivoli Personalized Services Manager subscriber database). Even though all Tivoli Personalized Services Manager updates must be sent to the Policy Director Management Server (since there is one instance per domain), the updates, once sent to the LDAP directory, would be referred to the correct master LDAP server for that sub-domain. This occurs at the LDAP level, and is not visible to Policy Director.

Other components, such as Location Based Services (via Policy Director), Intelligent Notification Services, and WebSphere Transcoding Publisher, primarily read from the directory, and would be good candidates to be configured to access a replica.

For more information on the use of these techniques, refer to *LDAP Implementation Cookbook*, SG24-5110.

For more information on this environment and test results, see the paper “A Highly Available & Scalable LDAP Cluster in an IBM AIX Environment” by Veronika Megler and Jay Bockelman, at IBM Developerworks (`http://www-1.ibm.com/servers/esdd/articles/ldap/index.html`, September 26, 2001).

If the combination of these techniques does not provide sufficient scalability, a group of replicas can be clustered via Load Balancer.

While IBM HTTP Server is also used for SecureWay Directory, it is only used for running the administration GUI. Therefore, no scalability issues are involved for this IBM HTTP Server instance.

7.2.11 Active Session Table Server

The Active Session Table Server manages information about each session that is created and/or authenticated. IBM Everyplace Wireless Gateway and WebSEAL-Lite are the only components that interact with the Active Session Table Server. Therefore, the Active Session Table Server should be sized to sustain the level of traffic required by these two components.

Although a primary and secondary Active Session Table Server instance are supported within a single WebSphere Everyplace Server domain, the secondary Active Session Table Server is only used when the primary fails. Therefore, the secondary instance does not improve Active Session Table Server scalability.

The Active Session Table Server stores its cache on disk, and maintains an index to the cache in memory. Therefore, performance can be improved by ensuring that the server can dedicate enough real memory to the Active Session Table, and provide high-speed disk for the cache. Plan on 100 bytes of real memory and 400 bytes of disk space per active user. The default maximum cache size is 500,000 active users, giving a real memory requirement of 50 Mb. and a disk requirement of 210 Mb.

7.2.12 Everyplace Synchronization Manager

We currently have insufficient information on how to scale this component.

7.2.13 MQSeries Everyplace

MQSeries Everyplace only allows a single queue manager within any given Java Virtual Machine. You may have multiple queue managers on a single system, if they are each running in a separate JVM; or, you may have each queue manager running on a separate server. Only one instance of each queue manager is allowed within the domain. Therefore, clustering technology is not a scaling option for MQe.

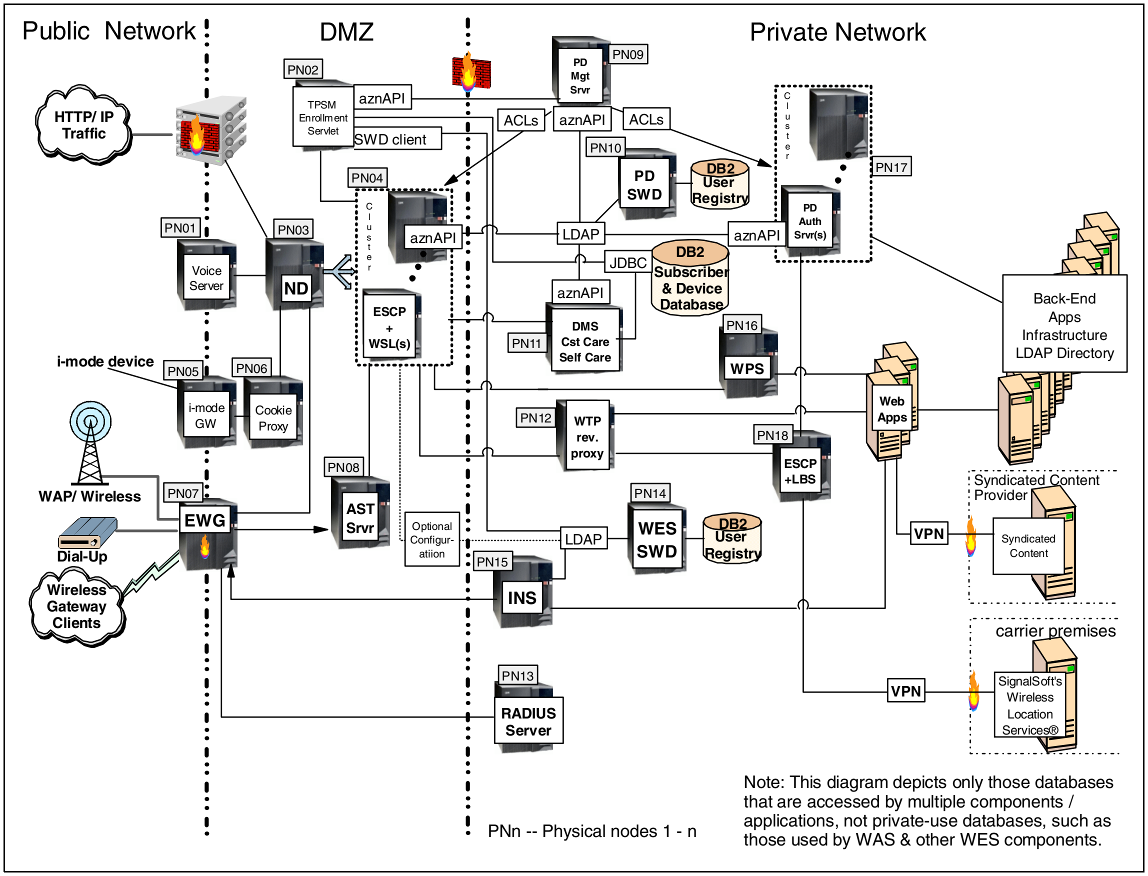
7.2.14 Relational database

A number of Everyplace Server components contain instances of relational databases (SecureWay Directory, Wireless Gateway, Tivoli Personalized Services Manager, Intelligent Notification Services, WebSphere Application Server). These databases should be tuned at the database level to ensure best performance. If database loads are sufficient, a remote, special-purpose database server may be required. For some cluster solutions (WebSphere Application Server, Everyplace Wireless Gateway), a remote database server is required to share information in the cluster.

Note that if the relational database is DB2 and it is installed on AIX, there is a restriction in the number of internal connections that can be opened. This is due to the AIX limitation of 10 shared memory segments per process. If you increase the number of connections that the component may open to the database instance to improve performance, or install several components on a single physical machine that all open connections to the local DB2 instance, you may violate this constraint. To remove this restriction, the databases should be configured as remote databases, using TCP/IP over the loopback interface.

7.3 Scalability in the business-to-customer scenario

In this section we describe how each component can be scaled to improve performance. We use as a foundation for this description the scenario described in Chapter 5, “Business-to-consumer” on page 219, and the physical operational model described in 6.7, “Physical operational model” on page 298. We describe for each component, how it can be scaled to improve performance. Figure 7-20 shows the full physical model that we use as our starting point.



Note: This diagram depicts only those databases that are accessed by multiple components / applications, not private-use databases, such as those used by WAS & other WES components.

Figure 7-20 Physical Operational Model

A physical node (PN) represents a box or a group of boxes that constitute a cluster. Every physical node has an input represented in number of request per second.

Table 7-1 Scenario Physical Nodes Scalability

Physical Node	Installed Products	Scalability	Comments
PN1	- Voice Server	(250 requests/second): - Single box	
PN2	- Tivoli Personalized Services Manager Self Enrollment	(25 requests/hour): - Single box	
PN3	- Edge Server Load Balancer	(5700 requests/second): - Single box	
PN4	- Edge Server Caching Proxy - WebSEAL-Lite	(5700 requests/second): - Cluster system to handle the number of input requests	
PN5	- i-mode Gateway	(10 requests/second): - Single box	
PN6	- Cookie Proxy	(10 requests/second): - Single box	
PN7	- Everyplace Wireless Gateway	(250 requests/second): - Single box	
PN8	- Active Session Table Server	(5950 requests/second): - Single box	- Active Session Table Server does not allow use of more than one instance. - This server should have high speed, fast disks and enough memory to handle all requests.
PN9	- Policy Director Server Manager	(4 notifications./minute): - Single box	
PN10	- Policy Director SecureWay Directory	(175 requests/second): - SecureWay Directory Master and Replica(s)	- The PD system will be replicated across multiple servers to distribute the number of requests

Physical Node	Installed Products	Scalability	Comments
PN11	- Device Manager Server - Self Care - Customer Care - Tivoli Personalized Services Manager Database	(600 requests/second): - WebSphere Application Server Cluster	- This product runs under WebSphere Application Server
PN12	- WebSphere Transcoding Publisher	(125 requests/second): - Cluster system to handle the number of input request	
PN13	- RADIUS server	(100 requests/second): - Single box	
PN14	- Everyplace Server SecureWay Directory	(20 requests/second): - Single box	
PN15	- Intelligent Notification Server	(200 notifications./minute): - Single box	
PN16	- WebSphere Portal Server	(2700 requests/second): - WebSphere Application Server Cluster	- This product runs under WebSphere Application Server
PN17	- Policy Director Authentication Server	(125 requests/second): - Single box	
PN18	- Edge Server Catching Proxy - Location Base Server	(125 requests/second): - Single box	

7.3.1 WebSEAL-Lite cluster

To handle almost 6,000 requests per second we would need to deploy a cluster of Edge Server Caching Proxy and WebSEAL-Lite. The cluster should be behind the Load Balancer so this can distribute the requests among all servers.

7.3.2 WebSphere Transcoding Publisher cluster

WebSphere Transcoding Publisher should be deployed across multiple servers to support the number of requests that this Physical Node is going to receive. For more information about WebSphere Transcoding Publisher refer to *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5 Extending Web Applications to the Pervasive World*, SG24-6233.

7.3.3 Policy Director’s SecureWay Directory

As a first step, you should tune SecureWay Directory and its underlying DB2 database. SecureWay Directory itself can be tuned via its internal cache, and via some additional tuning parameters.

Since SecureWay Directory uses DB2 for its data store, you should tune the DB2 to ensure that it will respond efficiently to this load. Good results are often achieved by moving DB2 into a separate file system or logical volume, adding “containers” and distributing the data across the containers, and tuning the physical disk subsystem. In addition, DB2’s internal parameters, such as buffer pool sizes, can be tuned.

Policy Director’s SecureWay Directory should be set up as a “master” and one or more “replicas”. The overall load should be distributed amongst the replicas. This system should be composed of a number of servers that will have the ACL database system replicated among them. SecureWay Directory has a process that replicates updates from the master to the replicas.

7.3.4 IBM WebSphere Application Server Clusters

Two products, Device Manager Server (part of Tivoli Personalized Services Manager) and WebSphere Portal Server, run under WebSphere Application Server. WebSphere Portal Server should be deployed in a multi-cluster WebSphere Application Server environment. At this time, Device Manager Server has not been tested in a WebSphere Application Server clustered environment.

WebSphere Application Server should use a RDBMS for its administration database, rather than the default install of InstantDB. In addition, you may wish to install a dedicated database server for the Tivoli Personalized Services Manager subscriber database and the administration database.

For more information about WebSphere Application Server scalability refer to *WebSphere Scalability: WLM and Clustering Using WebSphere Application Server Advanced Edition*, SG24-6153.

8

Availability

This chapter describes the availability characteristics of the IBM WebSphere Everyplace Server components, and then applies those characteristics to the scenarios discussed in Chapter 4 and Chapter 5.

8.1 Everyplace Server availability architecture

Availability is an end-to-end issue, affecting application design in addition to infrastructure components. While it is possible to improve availability by judicious placement of the Everyplace Server components, achieving continuous availability is currently still out of reach in most environments.

The following states our subjective conclusion and provides a *general* recommendation for the individual Everyplace Server as illustrated in Figure 8-1. Note that this section should be seen as a guideline, not as providing definitive answers.

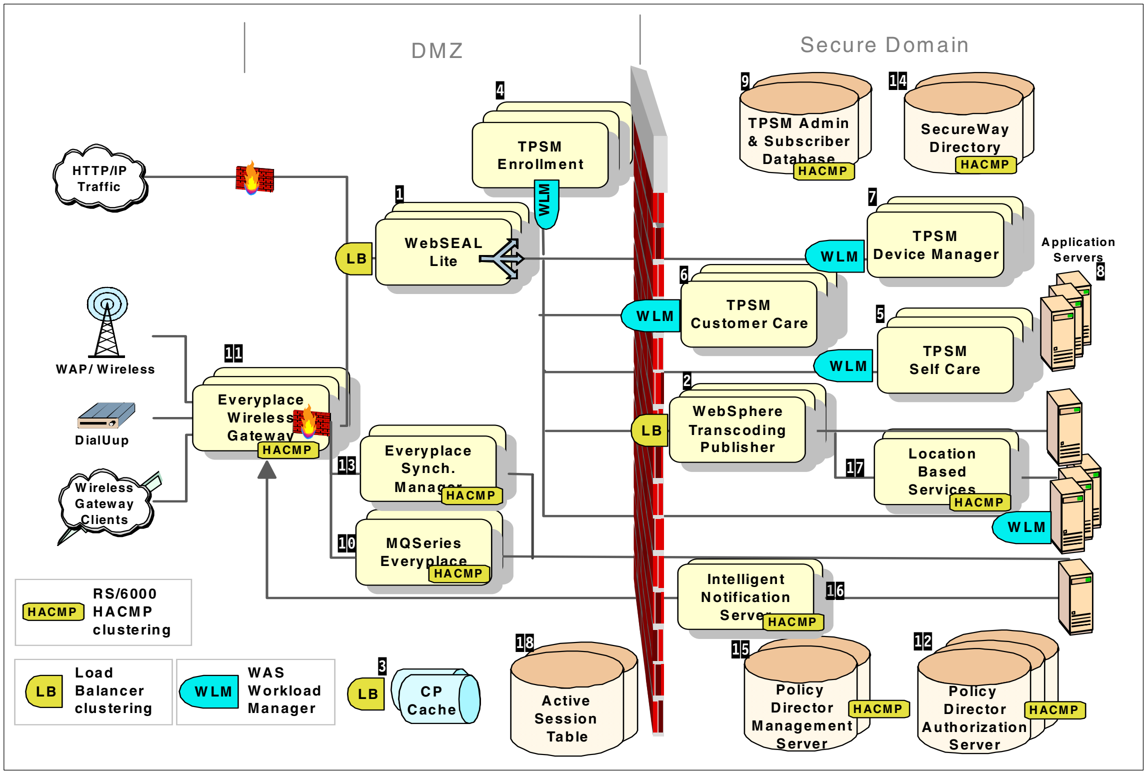


Figure 8-1 Everyplace Server Availability Architecture

Note that there are three kinds of clusters shown in the diagram:

- Edge Server Load Balancer clustering, formerly known as Network Dispatcher. These clusters are labelled “LB” in the diagram.
- WebSphere Application Server clusters, which can be implemented using WebSphere Application Server Workload Managed clusters, Load Balancer

clustering, or, in extreme circumstances, both. These are labelled WLM in the diagram.

- High Availability Clustered Multi-Processing clusters. These are labelled HACMP in the diagram.

IBM WebSphere Everyplace Server includes the Edge Server Load Balancer as the key component to improve availability and scalability.

If the solution is implemented on IBM RS/6000 hardware, High Availability Cluster Multi-Processing (HACMP) can also be used to improve availability. HACMP implements a heartbeat mechanism between two servers running AIX, and runs a set of user-defined failover scripts if the heartbeat is missed for a predefined number of beats. The surviving server takes on the original server's IP address in addition to its own, and so the network configuration does not need to change to direct workload to the backup server. However, the new server will not generally maintain any open connections. Therefore, the server's client application must be able to open a new connection to the failed-over application server, roll back any partially executed transactions, and present a consistent application state across these activities. In addition, failover scripts must be designed so that the application correctly picks up configuration scripts, startup information, and application state as appropriate.

The appropriate failover solution differs depending on the specific software component. The next section describes the appropriate failover solution for each component.

Load Balancer

In general, components that are facing incoming HTTP traffic over TCP/IP are well suited for use with Load Balancer. This is true for:

1 WebSEAL-Lite

2 IBM WebSphere Transcoding Publisher

3 Edge Server Caching Proxy (depending on its use and configuration)

WebSphere Application Server Workload Manager

WebSphere Application Server provides clustering and workload management capabilities. For the applications that run in this environment, this is the recommended first line of defense. An additional Load Balancer cluster can be placed in front of the WebSphere Application Server cluster if required. This is the recommended solution for the following WebSphere Everyplace Server components:

4 Tivoli Personalized Services Manager, Enrollment

5 Tivoli Personalized Services Manager, Self Care

6 Tivoli Personalized Services Manager, Customer Care

7 Tivoli Device Manager Server

8 Other WebSphere Application Servers inside the Everyplace Server domain

HACMP

For other components a Load Balancer cluster is not applicable nor recommended. This can be the case where one or more of these statements are true:

- Horizontal scalability is not required or not supported
- The component cannot have more than one instance active at a time in a single domain
- The component can't be deployed in a cluster (or where this will be too complicated or expensive)
- The network latency introduced by a dispatcher is too significant

A good example is database servers. If scalability requirements allow it, then a deployment of a single database instance on a single (potentially large) machine is a much simpler and less expensive solution. Availability can then be addressed by hardware failover, where the master and slave machines share a single fault-tolerant external disk system such as IBM SSA or Shark.

For these reasons using HACMP is our general recommendation for:

9 Tivoli Personalized Services Manager Subscriber database

10 IBM MQSeries Everyplace. The MQSeries Everyplace protocol runs over TCP/IP. It can be wrapped using HTTP tunneling, which enables it for use with Load Balancer. However, the connection is extremely stateful, so it is nearly impossible to maintain the session if a failover occurs. You will have to set up Load Balancer for strict affinity and by this introduce considerations on performance penalty and fair load balancing. If you are not depending on horizontal scalability, we generally recommend HACMP failover for availability.

11 Everyplace Wireless Gateway's principal node for each cluster

12 Individual Policy Director Authorization Servers that may be required by your back-end applications

13 Everyplace Synchronization Manager

14 IBM SecureWay Directory

15 Policy Director management server

16 Intelligent Notification Services. Only one Intelligent Notification Server instance is allowed in the WebSphere Everyplace Server domain.

17 Location Based Services. Although more than one Location Based Services server may be active in the WebSphere Everyplace Server domain, only one Location Based Services server may connect to a single server providing the location service.

Other Mechanisms

Some components have built-in mechanisms used for availability.

18 The Active Session Table Server has its own availability design. This allows a single, backup Active Session Table Server to be available within the WebSphere Everyplace Server domain.

11 IBM Everyplace Wireless Gateway has built-in clustering and failover functions supporting all protocols, described in “Everyplace Wireless Gateway” on page 310. This specialized support is preferable to Load Balancer, since Location Based Services supports IP only.

8.2 Availability by Product

This section describes in more detail availability characteristics by product. The following products are described in this section:

- IBM Everyplace Wireless Gateway
- WebSEAL-Lite
- Edge Server Caching Proxy
- Edge Server Load Balancer
- WebSphere Transcoding Publisher
- Tivoli Personalized Services Manager
- Intelligent Notification Services
- Location Based Services
- Policy Director
- SecureWay Directory
- Relational Databases

8.2.1 IBM Everyplace Wireless Gateway

Since the secondary nodes back up each other, the Everyplace Wireless Gateway should be configured to delegate the maximum load from the primary node to the secondary nodes.

The principal in an Everyplace Wireless Gateway cluster is a single point of failure. HACMP can be used to provide a failover capability for the primary, as shown in Figure 8-2.

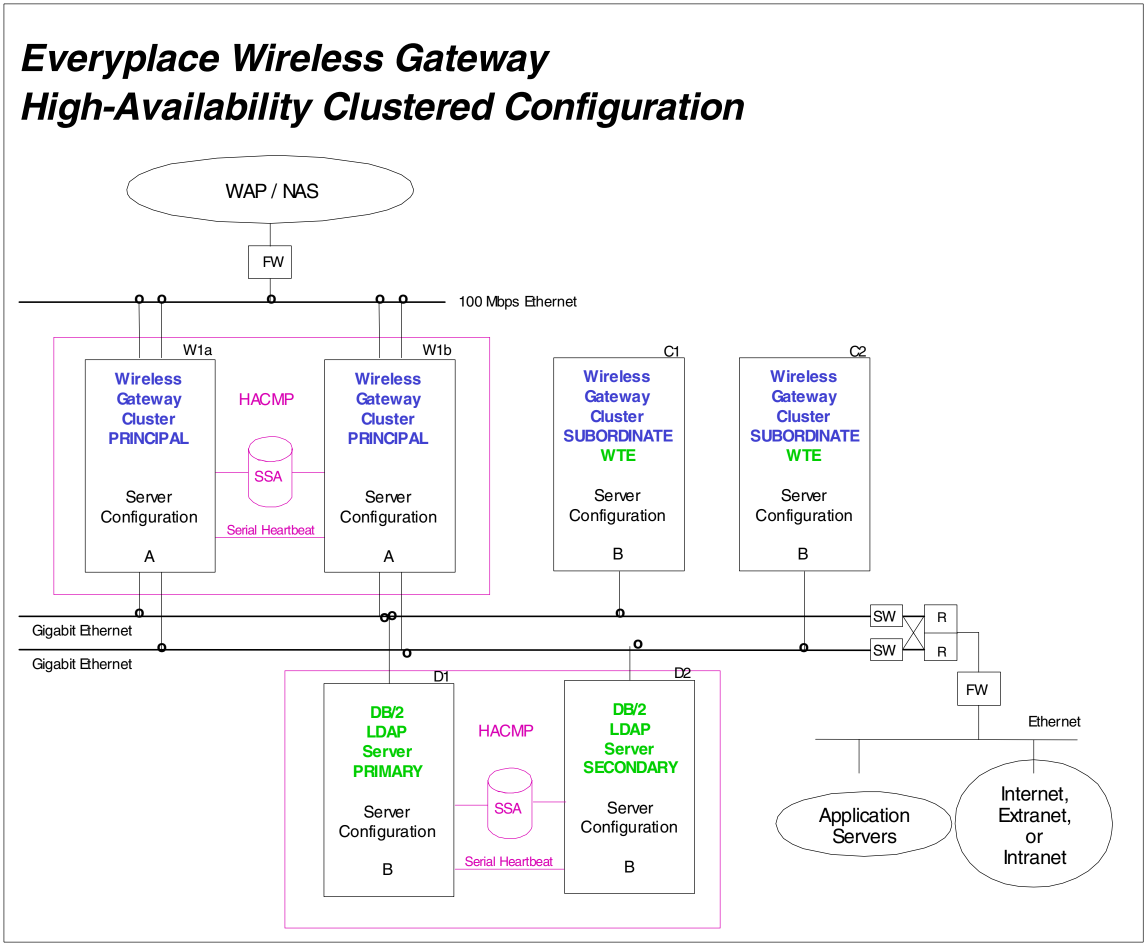


Figure 8-2 HACMP configuration for Everyplace Wireless Gateway

8.2.2 WebSEAL-Lite

Since WebSEAL-Lite runs as an Edge Server Caching Proxy plug-in, its availability characteristics are dependent on the Caching Proxy's availability characteristics.

You will need to restart Edge Server Caching Proxy whenever you change the WebSEAL-Lite configuration files for the changes to become effective.

8.2.3 Edge Server Caching Proxy

The Edge Server Caching Proxy may be used to provide high availability to the Caching Proxy. The Edge Server Load Balancer will detect when one of the Caching Proxies becomes unavailable, whether it has failed or if the Caching Proxy is engaged in garbage collection or cache refresh, and automatically reroutes requests to another proxy.

In a cluster scenario, individual Caching Proxies may be dynamically added or removed. This makes performing maintenance easy.

8.2.4 Edge Server Load Balancer

One of the basic functions of the Load Balancer is to provide availability to a group of servers by not directing client requests to a failed server. Load Balancer does this via the use of advisors.

Overview of advisors

The Load Balancer will perform a TCP/IP level check on the health of the content servers that it is load balancing. This check is generally a very basic ping. If the check is unanswered, the Load Balancer will assume that the server is inactive and not route traffic to that server.

More often, the content server itself may be active but the applications running on it are either inactive or not delivering full functionality. If the basic TPC/IP check is answered, the Load Balancer will still route traffic to these content servers, even though the user will not be able to use the application running on them. This impairs the effectiveness of the Load Balancer for providing an available Web site.

This can be easily solved through the use of advisors. The Load Balancer provides standard and customized advisors to perform checks on the applications that run on the servers it load balances. Using these advisors, if an error or unexpected response is received from the applications, the advisor will provide this information to the manager component of the Dispatcher. The Dispatcher can then use this information to route requests for that particular application to an active application.

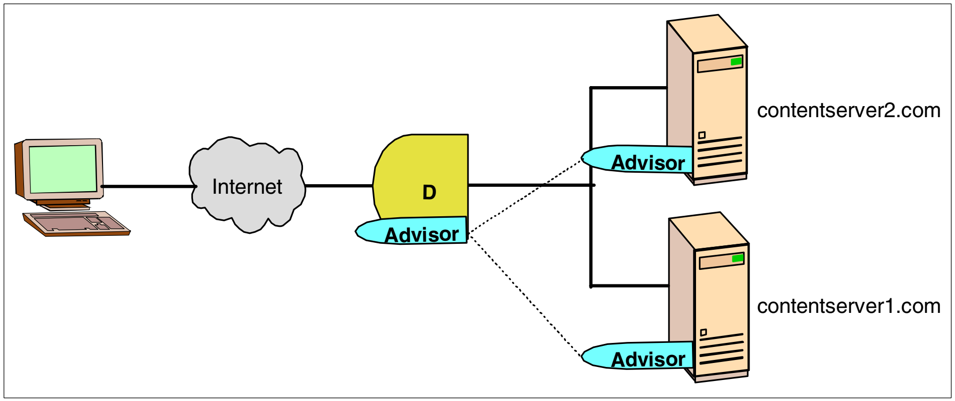


Figure 8-3 Using advisors to monitor the health of your applications

Standard advisors

The Edge Server Load Balancer provides standard advisors to provide an application-level check that each server is up and running. The advisor is a lightweight client that runs as part of the Dispatcher to pass real commands to each application server to simulate application functionality. The Load Balancer provides standard advisors for HTTP, FTP, SSL, SMTP, NNTP, POP3, Telnet, ping, and WLM services.

Customized advisors

Custom advisors are potentially more powerful in resolving the problem of dispatching to dead applications. Custom advisors can be used to provide more specific and tailored load metrics directly from the application. A custom advisor can be created to check other protocols or implement specific extensions to the advisor. Written in Java, it works with counterpart code on the application server to provide a high degree of synergy between the Dispatcher and the application is it balancing. For example, a servlet running on the application server can be coded to extract in-depth performance data from the server, or verify valid connectivity to a back-end server or database and returns the results to the Dispatcher.

The Load Balancer provides a sample custom advisor for use with IBM WebSphere Application Server to check the connectivity of both the WebSphere Application Server and from the WebSphere Application Server to a back-end DB2 database.

Dispatcher High Availability

This same principle can be applied to the Load Balancer itself. Since the Load Balancer is the address to which clients connect, if it fails or is inactive then clients cannot access both the Dispatcher and more importantly the content servers that the Dispatcher is load balancing. The Load Balancer itself should also provide high availability. Both the Dispatcher and Interactive Session Support components can provide high availability.

Interactive Session Support (ISS)

Interactive Session Support (ISS) periodically monitors the level of activity on a group of servers and detects which server is the least heavily loaded. It can also detect a failed server and forward traffic around it. Once every monitoring period, ISS ensures that the information used by the domain name server or the Dispatcher accurately reflects the load on the servers. The load is a measure of how hard the server is working. The system ISS administrator controls both the type of measurement used to measure the load and the length of the load monitoring period. You can configure ISS to suit your environment, taking into account such factors as frequency of access, the total number of users, and types of access (for example, short queries, long-running queries, or CPU-intensive loads). In addition to the functions already mentioned, ISS can perform load balancing and can feed information to Dispatcher as it is performing load balancing.

You can use the Interactive Session Support component with or without a DNS name server:

- If you are using ISS for load balancing, a domain name server is required. This may be either an actual DNS name server or, if you set up a small, separate subdomain for a new name server, a replacement name server provided by ISS. Using this approach, ISS runs on a name server machine. A client submits a request for resolution of the DNS name of an ISS service, which has been set up by an administrator. ISS then resolves the name to the IP address of a server in the cell, and forwards this IP address to the client.
- If you are using ISS just to collect server load information, a domain name server is not needed. The ISS monitor collects server load information from the ISS agents running on the individual servers and forwards it to the Dispatcher. The Dispatcher uses this load information, along with other sources of information, to perform load balancing.

ISS is intrinsically highly available. All nodes in a cluster work together to eliminate any single point of failure. Should the monitor machine fail, the survivors elect a new monitor to take over automatically.

Dispatcher

To eliminate the Dispatcher itself as a single point of failure, another Dispatcher can be implemented. A standby Dispatcher machine could remain ready at all times to take over load balancing should the primary Dispatcher machine fail.

Two options are available to the Dispatcher to increase the availability of your cluster:

- Heartbeat high availability
- Mutual high availability

Heartbeat high availability

To implement high availability using heartbeats, the Dispatcher can be configured with a standby machine on the same cluster that listens for heartbeats from the active machine. These heartbeats are the communication sessions between the two machines. If the standby Dispatcher receives a response then it synchronizes its state with the active machine. If the standby machine detects that the heartbeat from the active machine is no longer being received, it becomes active, and will take over the cluster IP addresses and takes the role of balancing and forwarding requests.

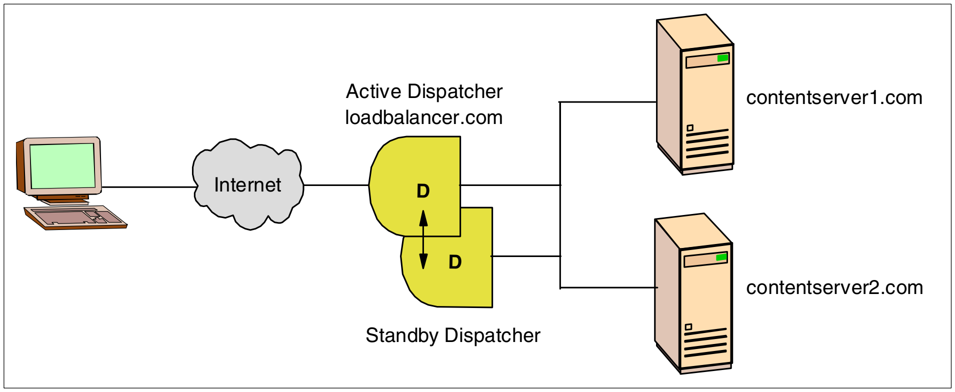


Figure 8-4 Providing high availability to the Dispatcher using heartbeats

Typically failover occurs in five seconds or less, minimizing the number of connections attempts that might fail while the failover is in progress. The newly activated machine still knows where to send all requests that it receives and TCP automatically resends any individual packets that were lost during the actual failover. The synchronized nature of the two machines also means that in the event of failover all existing connections are also failed over to the standby Dispatcher.

Mutual high availability

High availability for the Dispatcher can be achieved through the use of mutual high availability. Mutual high availability allows two Dispatcher machines, an active and standby Dispatcher, to both actively load balance client traffic, while also acting as backups for each other. For mutual high availability, you still have the concept of active and standby dispatchers, but they are linked to the server cluster. Each cluster will be assigned a primary and secondary dispatcher. Each dispatcher is both the primary dispatcher to its own cluster and the secondary dispatcher to the other dispatcher, thereby providing mutual availability. In the event of failure, the other machine performs load balancing for both its own cluster and the failed Dispatcher's cluster.

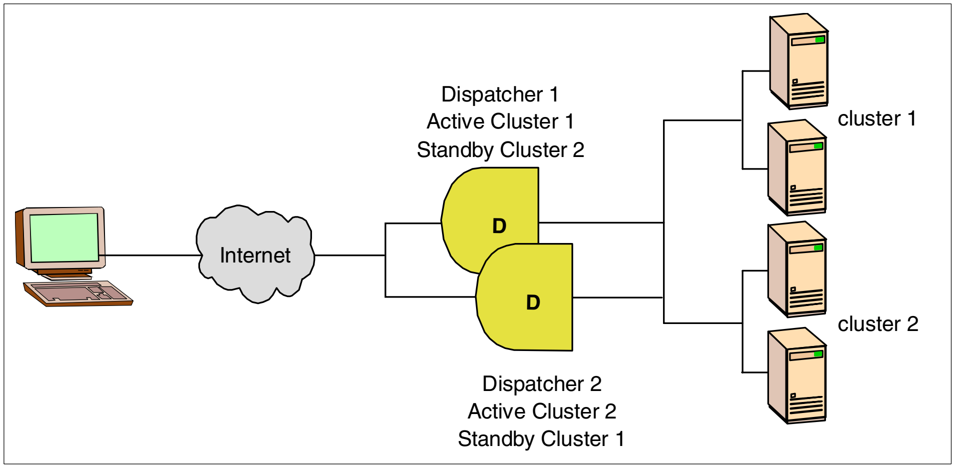


Figure 8-5 High availability to the Dispatcher using mutual high availability

Collocation

When a site's load increases to the point where you need more than one server to handle the traffic, you can add a Dispatcher to your existing network infrastructure with a minimum of hardware investment by using collocation. Collocation enables you to install the Dispatcher on one of your existing machines, which may well be one of the content servers that you wish to load balance.

Alternatively, if you wish to provide high availability to your existing Load Balancer, you may choose to collocate your standby Dispatcher with one of the content servers.

8.2.5 WebSphere Transcoding Publisher

Advisors, whether standard or customized, could be used to provide advanced monitoring of the health of the WebSphere Transcoding Publisher servers in the cluster.

8.2.6 Tivoli Personalized Services Manager

Tivoli Personalized Services Manager is a set of applications that run in a WebSphere Application Server environment. Therefore, solutions to WebSphere Application Server availability should also be applied to Tivoli Personalized Services Manager. For detailed information and possible configurations, refer to resources such as *WebSphere Scalability: WLM and Clustering Using WebSphere Application Server Advanced Edition*, SG24-6153.

The Tivoli Personalized Services Manager subscriber database is the central core of user information for the Everyplace Server environment. Subscriber information from this database provisions Policy Director and the SecureWay Directory directories. You should ensure that this database is properly backed up.

If the Tivoli Personalized Services Manager database and the LDAP directories get out of synchronization due to failures, there are currently no tools available that will allow these databases to be compared and re-synchronized. The directories would need to be rebuilt from the Tivoli Personalized Services Manager subscriber database. For large subscriber environments, this may not be practical.

8.2.7 Intelligent Notification Services

Since only one instance of each Intelligent Notification Services server may exist within the WebSphere Everyplace Server domain, no clustering can be performed. Backup of each Intelligent Notification Services server can be provided using HACMP.

Whenever log or trace properties are modified, the Intelligent Notification Services servers must be restarted to pick up the new settings. These restarts should be scheduled for minimum affect on the user population.

Note that, if logging parameters or product configuration information are changed, the Intelligent Notification Services servers must be restarted to pick up the new information.

If properties is changed, then the application server within which the Intelligent Notification Services applications are running must be restarted.

8.2.8 Location Based Services

Since there may be only one Location Based Services server for each location server at a time, clustering is not an appropriate availability solution. HACMP failover support could be used for each Location Based Services server.

8.2.9 Tivoli SecureWay Policy Director

Since there can be only one Tivoli SecureWay Policy Director Management Server in a domain, high availability for Policy Director can be ensured by placing the Policy Director Management Server in an HACMP cluster with a standby processor. In the Everyplace Server environment, the components that maintain copies of the Policy Director directory are WebSEAL-Lite and Location Based Services.

If the back-end applications use specific Authorization Servers, these may also need to be placed in HACMP clusters.

8.2.10 IBM SecureWay Directory

IBM SecureWay Directory itself can improve availability by using the same mechanisms that are used for scaling; that is, by using a master and one or more replica directory servers.

In addition, the master and one replica can be placed in an HACMP cluster, with the replica restarting as a master in the event of a failure. Fail-back must be managed by the administrator to ensure the directories are synchronized.

For more information on this environment and test results, see the paper “A Highly Available & Scalable LDAP Cluster in an IBM AIX Environment” by Veronika Megler and Jay Bockelman, at IBM Developerworks (`http://www-1.ibm.com/servers/esdd/articles/ldap/index.html`, September 26, 2001).

8.2.11 Relational Databases

A number of Everyplace Server components contain instances of relational databases (SecureWay Directory, Wireless Gateway, Tivoli Personalized Services Manager, Intelligent Notification Services, WebSphere Application Server, Synchronization Manager). Each of these relational databases should have an availability solution.

If the Relational Database Management System (RDBMS) is running on AIX, you can place the server in an HACMP cluster. If the using application is using a database client that supports connection pooling and will reopen severed connections with the database, it is possible to have the database fail over without causing the application as a whole to fail. For example, this has recently been tested with WebSphere Application Server Advanced Edition and a remote DB2 back-end server.

9

Security

Since WebSphere Everyplace Server is specifically designed to extend data and applications beyond the confines of fixed location terminals, it must have security mechanisms and a security architecture to accommodate the various needs of the different ways WebSphere Everyplace Server can be implemented. This chapter addresses the security implementation within the WebSphere Everyplace Server. This chapter deals primarily with security in WebSphere Everyplace Server Service Provider Offering. Enable Offering is intended to be integrated into an existing enterprise security architecture. For more information on security architecture in IBM WebSphere Everyplace Server Enable Offering, refer to 4.4.1, “Security architecture” on page 163.

As an architect or an integrator, you might be functionally familiar with a few or even most of the components in WebSphere Everyplace Server. You might even be familiar with your or your client’s existing security infrastructure. However, recognizing that security might not be your area of expertise, or that we might use different terms to mean the same thing, we will begin with some background information and definitions before going into the security specifics of WebSphere Everyplace Server.

Having established the basic security concepts, we will delve into the security design and components of WebSphere Everyplace Server. First, a review of the WebSphere Everyplace Server components that address security objectives. Second, we will discuss how security is implemented in WebSphere Everyplace Server. Finally, we will discuss firewall considerations for an architecture based on WebSphere Everyplace Server.

9.1 Background

Security for data communications has been well documented in the information processing industry since before the Internet became popular, so it is not our intent to be a compendium of security definitions, standards and technologies. Therefore we will not attempt to review every aspect of security, but focus on the most common and relevant security objectives for extending applications and data to mobile users.

9.1.1 Security objectives

The most common and relevant security objectives for data communications are:

Authentication: The verification of the *identity* of the user attempting to send or receive the data or application. This is to make sure the clients or servers are really who they claim to be.

Authorization: The granting or denying *levels of access* to data and applications. For example, a user (a person or an application) might have authorization to read a certain record, but not update it; or to update, but not to delete nor create; or in Internet environments, to read (browse) but not run an executable, nor go deeper in a directory tree.

Confidentiality: In data communications vernacular, the unlikelihood that data is being intercepted or shared without permission.

Integrity: The assurance that data has not been altered in transit by a third party. This concern relates to forgery, fraud, tampering, and other unauthorized alteration.

Non-repudiation: The preventing of the ability by the parties involved in a transaction to deny that they were involved. In other words, we want to prevent the buyer from being able to say "*I* didn't authorize that purchase," and the seller from being able to say "*I* never said I'd sell it at that price," etc. This is to enforce the accountability for electronic transactions.

These are the five objectives normally desired by online content and application services that involve data communication. Another important objective for secure computing is to create the secure *boundary* for the service domain. This is to reduce the chance of being actively attacked by hackers from the public networks. Such attacks include denial of service, packet spoofing, and impersonation, etc.

9.1.2 Architectural components needing security

An architect is likely to need to know the security needs from one end of the architecture to the other. In a mobile e-business architecture, the affected components would include:

- The mobile device
- The mobile network
- The gateway between the mobile network and the enterprise's network
- The enterprise's network
- The Web server
- The Web application server
- The applications and the data

9.1.3 Security techniques

There are many techniques for achieving the most common security objectives. Such techniques include data encryption, message digest, digital certificates, packet filtering, address concealing, and more. The many implementations of these techniques have led to the popular security solutions or technologies such as the IEFT (Internet Engineering Task Force) and standard Transport Layer Security (TLS, formerly called Secure Socket Layer or SSL).

Transport Layer Security uses data encryption, message digest, digital certificate, and other techniques to achieve multiple security objectives, such as confidentiality, authentication, and data integrity. TLS is primarily used for TCP/IP networks. For a mobile/wireless architecture, TLS alone is not a solution, but is part of a solution. For data communications over wireless networks using the WAP protocol, there is a technique similar to TLS called Wireless Transport Layer Security (WTLS).

Note: SSL (TLS) does not provide non-repudiation automatically but if required it helps you to implement it at the application level.

Other security solutions include proxy servers and firewall devices, both of which are for achieving a secure *boundary* for the service domains. Firewall devices use packet filtering, address concealing, and perhaps other security techniques to protect the service domain on the edge of the network from deliberate attacks that come from the public networks. Some firewalls are also proxy servers. Proxy servers provide delegated authority to access other systems; the other systems trust the proxy server rather than having to authenticate the user.

9.2 Security components

In this section we will discuss the components of WebSphere Everyplace Server that specifically address the security objectives. These components are discussed in more detail throughout this book and a similar overview is given in Chapter 1, “Introduction” on page 3. Here we briefly recap the features as they relate to security in order to provide the background for 9.3, “Security implementation” on page 365. In that section, we will outline how these components interact in WebSphere Everyplace Server to address each of the security objectives.

9.2.1 Everyplace Wireless Gateway

The Everyplace Wireless Gateway is an IP gateway that connects non-IP networks such as CDMA or GSM Cellular Networks to the IP enabled Internet. This allows IP-based applications to run over a number of wireless networks which do not support IP. In addition, Everyplace Wireless Gateway includes a messaging gateway to support Short Message Service (SMS). Everyplace Wireless Gateway also provides support for Wireless Application Protocol (WAP), including support for cookies on behalf of WAP clients.

The Wireless Gateway is a router or bridge and provides some security functionality of a firewall. For example, the Wireless Gateway is capable of packet filtering.

The Wireless Gateway provides a secure and optimized connection to mobile devices. Wireless Devices with client software installed can take advantage of Wireless Gateway’s WLP support.

Wireless Gateway provides Wireless Transport Layer Security (WTLS) between the wireless client and the gateway using the following encryption algorithms:

- Diffie-Hellman
- RSA key exchange (RSA 1024, 768 or 512 bits)

Wireless Gateway supports encryption using:

- RSA RC5 (40, 56, or 128 bit)
- DES and Triple DES algorithms
- SHA 1 (Secure Hash Algorithm) Message Authentication Codes (40-80 bit).

SSL encryption is used to establish a secure connection from the gateway to back-end servers.

9.2.2 WebSEAL-Lite

Currently, maintenance and enforcement of authentication and authorization of users is largely left up to individual Web Servers. This often results in the administration of multiple user databases. Users may also need to manage multiple user IDs and passwords and be required to log in multiple times within a single domain. WebSEAL-Lite solves these problems by providing centralized authentication and authorization (when used with Policy Director) for a secured domain, allowing single sign-on and centralized administration.

WebSEAL-Lite is a plug-in for Edge Server Caching Proxy provided with WebSphere Everyplace Server 2.1. WebSEAL-Lite combines a Web caching proxy of Edge Server Caching Proxy with the authorization engine of Policy Director to deliver protected resources to users, as depicted in Figure 9-1.

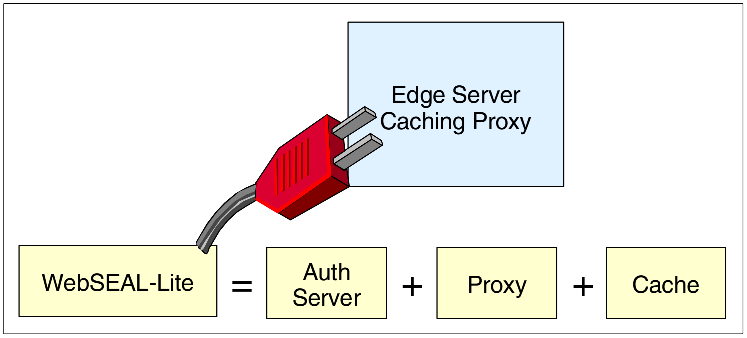


Figure 9-1 WebSEAL-Lite

WebSEAL-Lite is an enhanced version of the AuthServer plug-in that was part WebSphere Everyplace Suite V1.1 and continues to provide all the function AuthServer had. It also provides most of the function of WebSEAL, but in the form of a plug-in. WebSEAL-Lite does not support CDAS or global sign-on.

WebSEAL-Lite allows the internal structure of an enterprise to be hidden from malicious attackers from the Internet by acting as a reverse proxy, thus concealing the IP addresses of the Web and application servers from the Internet, in addition to presenting a single domain name for the set of servers in the enterprise protection space.

9.2.3 IBM SecureWay Policy Director

Policy Director is general access control/authorization software that uses a database of objects, users, groups, and access control lists. It can tell an application what access control list is associated with a specific object, what users and groups are in the list, and what permissions they have. It is up to the application (in our case WebSEAL-Lite) to interpret this information.

Policy Director provides a centralized implementation and administration of access control. When used in combination with WebSEAL-Lite in a WebSphere Everyplace Server environment, access control is enforced at the "Edge of Network", instead of a distributed policy enforcement at the application level. This allows for single sign-on to back-end applications.

9.2.4 IBM SecureWay Directory

IBM SecureWay Directory (SWD) is a server that complies with Lightweight Directory Access Protocol (LDAP) and runs as a stand-alone daemon. It is based on a client/server model that provides client access to an LDAP server. SecureWay Directory provides an easy way to maintain directory information in a central location for storing, updating, retrieving, and exchanging user registration information.

9.2.5 IBM Active Session Table Server

Active Session Table Server provides a high-speed specialized cache that catalogs information about users currently connected to the WebSphere Everyplace Server. Session information is temporary and is configured to expire after a pre-set duration of time.

9.3 Security implementation

The IBM WebSphere Everyplace Server is designed to create a safe environment to support pervasive computing. It is implemented to have centralized user authentication from limited points of entry to the server. It provides single sign-on for user-friendly implementation of credential sharing across the services hosted by the Server. The WebSphere Everyplace Server relies on a set of industry-standard security solutions, such as TLS/SSL and WTLS, to achieve the security objectives for the service domain. The server uses the proxy technology in conjunction with firewalls to define the secure boundary for the service domain.

In this section, we will look at how WebSphere Everyplace Server implements each of the primary security objectives discussed in 9.1.1, “Security objectives” on page 361.

9.3.1 Authentication

WebSphere Everyplace Server employs several industry-standard technologies to perform authentication. WebSphere Everyplace Server uses the HTTP basic authentication process to authenticate the users coming from the Internet and third-party gateways. For this reason, the clients and/or the gateway proxy for the clients must support HTTP. This authentication is handled by the WebSEAL-Lite.

In addition, mobile clients may authenticate themselves at the Everyplace Wireless Gateway when it is part of the WebSphere Everyplace Server environment. Both the Everyplace Wireless Gateway and WebSEAL-Lite use a centralized LDAP directory server to authenticate users. The Everyplace Wireless Gateway may also use a Remote Authentication Dial In User Service (RADIUS) server in certain configurations.

WebSphere Everyplace Server is designed to achieve single sign-on, namely to authenticate the users only once for their access to the services within the WebSphere Everyplace Server domain. This authentication design is achieved by sharing user credentials through a centralized repository. The centralized repository consists of an Active Session Table (AST) database, and a SecureWay LDAP Directory database that may consist of Tivoli Personalized Services Manager user or subscriber information. As previously mentioned, both the Everyplace Wireless Gateway and WebSEAL-Lite can use a centralized LDAP directory server to authenticate users. They both can deposit user active session entries into the Active Session Table database to share the user credential and profiles. Information sharing among the Everyplace Wireless Gateway and WebSEAL-Lite is achieved by their access to the Active Session Table Server and LDAP directory server.

Authentication in Everyplace Wireless Gateway

The Everyplace Wireless Gateway authenticates users entering from three types of different connections. Each type of user connection may use a different authentication process.

- The Wireless Gateway uses the two-party key exchange protocol (2PKDP) with mutual authentication to authenticate wireless clients (non-WAP) at the Wireless Link Protocol (WLP) link layer.
- WAP clients are authenticated by WebSEAL-Lite with a user ID and password using HTTP basic authentication in the Wireless Gateway. WAP clients can also authenticate with the server using Wireless Transport Layer Security (WTLS).

- Everyplace Wireless Gateway can authenticate dial-in clients in conjunction with a RADIUS server.

Wireless client authentication

Devices with Wireless Client application software installed can access Everyplace Server via the Everyplace Gateway. The client and server authentications are simultaneously completed using two-party key exchange protocol included in the Wireless Link Protocol (WLP).

WLP is an optimized link protocol to create a secure connection between the Wireless Gateway and Wireless Clients. WLP was developed by IBM Research and is particularly useful in a wireless environment where transmissions can potentially be readily intercepted and where spoofing is a particular concern. Part of the authenticated logon process includes encrypted distribution of encryption keys, which are dynamically generated, and used only for that user's connection session with the gateway. This logon and two-party authentication process, complete with key distribution, can be efficiently achieved by exchanging four radio network packets. Client authentication occurs during the handshake process. In addition to negotiating a security algorithm and exchanging cipher secrets, the user may also enter a user ID and password.

WLP is used to create a tunnel through which TSL/SSL connections can be made. This allows wireless clients to establish a secure connection all the way to back-end application servers. This eliminates the security gap inherent in WTLS, where the data must be decoded at the gateway and then re-encoded before being passed to the back-end servers via SSL. This is discussed further in 9.4.1, “Securing the WAP gap” on page 385. Figure 9-13 on page 386 illustrates the WTLS and WLP connections through the Wireless Gateway.

Once the user is authenticated, the Everyplace Gateway inserts the trusted IP address into the HTTP request header and enters session information into the Active Session Table. The request is then passed on to WebSEAL-Lite.

WAP client authentication

WAP clients can access the wireless gateway using the Wireless Application Protocol (WAP). One option for client authentication is HTTP basic authentication. More commonly client authentication is done using the handshake protocol of Wireless Transport Layer Security (WTLS which creates a security-rich environment for wireless internet transaction. WTLS is a protocol optimized for the wireless environment and is analogous to TLS (a newer version of SSL) in the wired world.

During the handshake process, in addition to negotiating security algorithms and exchanging cipher secrets, the WAP client sends a user ID and password. WTLS also supports authentication using mini-certificates.

Once the Everyplace Wireless Gateway authenticates the WAP client, Everyplace Wireless Gateway assigns the client its own trusted IP address and inserts it into the HTTP request header, then inserts WebSphere Everyplace Server HTTP headers for user and client ID information. Everyplace Wireless Gateway then passes the request to WebSEAL-Lite.

Dial in client authentication using RADIUS

Authentication for dial-in clients is essentially the same process as wireless client (non-WAP) authentication. Everyplace Wireless Gateway is capable of using a third-party RADIUS server for authentication as well. In this configuration, Everyplace Wireless Gateway is configured as a RADIUS client. WAP or dial-in clients may send user name and password information through a customizable prompt or through a framing protocol such as PPP whose authentication packets carry this information. Everyplace Wireless Gateway can then forward this information to the RADIUS server.

The RADIUS server then authenticates the client based on a list of requirements that must be met. This list of requirements always includes verification of the password, but may optionally contain other criteria such as clients or ports that the user is allowed to access. Once the RADIUS server has authenticated the user, Everyplace Wireless Gateway adds an HTTP header containing client ID and user information to the request before forwarding it to back-end servers.

WebSEAL-Lite with Wireless Gateway authentication

For non-WAP clients, once the Everyplace Wireless Gateway has authenticated the user, Everyplace Wireless Gateway generates a session ID and inserts the session information into the Active Session Table. For WAP clients, WebSEAL-Lite creates the session entry. The Wireless Gateway then adds WebSphere Everyplace Server specific HTTP headers to the initial request. These headers may include a session ID, Active Session Table Server location, user name and realm, and a client ID, such as IP address or phone number. The WebSphere Everyplace Server HTTP headers are discussed in more detail in 3.3.1, “WebSphere Everyplace Server HTTP headers” on page 85.

Everyplace Wireless Gateway then forwards the HTTP request to WebSEAL-Lite. WebSEAL-Lite intercepts all HTTP traffic coming from the gateway to ensure that all the requests have been authenticated.

WebSEAL-Lite first checks each incoming HTTP request to determine if it came from the IBM Everyplace Wireless Gateway based on the sender’s IP address. If the requests intercepted by WebSEAL-Lite are from a trusted IP address that is owned by the Everyplace Wireless Gateway, then the request is assumed to be from the Everyplace Wireless Gateway.

WebSEAL-Lite then ensures that the user has been authenticated. WebSEAL-Lite inspects the HTTP headers for the Session ID, or, if the request is from a WAP client, for user and client ID information that is inserted during the Everyplace Wireless Gateway's authentication process. If the request is from a WAP client device then it is up to WebSEAL-Lite to generate a session ID and insert it into the Active Session Table. The HTTP request coming from the gateway will contain client ID and user information in the HTTP header. WebSEAL-Lite creates a session ID and creates an entry in the Active Session Table.

At this point a session entry should exist for the request, generated either by Everyplace Wireless Gateway or by WebSEAL-Lite. In addition to the headers added by the Everyplace Wireless Gateway, WebSEAL-Lite inserts additional header information for the network and device type. These fields indicate WebSphere Everyplace Server SecureWay Directory device and network profiles to be used by back-end applications. WebSEAL-Lite then forwards the request to the destination server. When the request comes through the IBM Everyplace Wireless Gateway, WebSEAL-Lite skips the authentication process since the user request is considered to be trusted.

Figure 9-2 and Figure 9-3 both illustrate the process of authentication with the IBM Everyplace Wireless Gateway. These figures also illustrate the role of WebSEAL-Lite in the request/response flow. Figure 9-2 shows the request/response process and the role of WebSEAL-Lite when Everyplace Wireless Gateway is used for authentication with a wireless client. Figure 9-3 shows this process when a WAP device is used. The primary difference is the location where the session entry is generated and added to the Active Session Table. With the WAP client, the session is created by WebSEAL-Lite. Everyplace Wireless Gateway provides WebSEAL-Lite with user information by inserting user and client ID WebSphere Everyplace Server HTTP headers into the initial request.

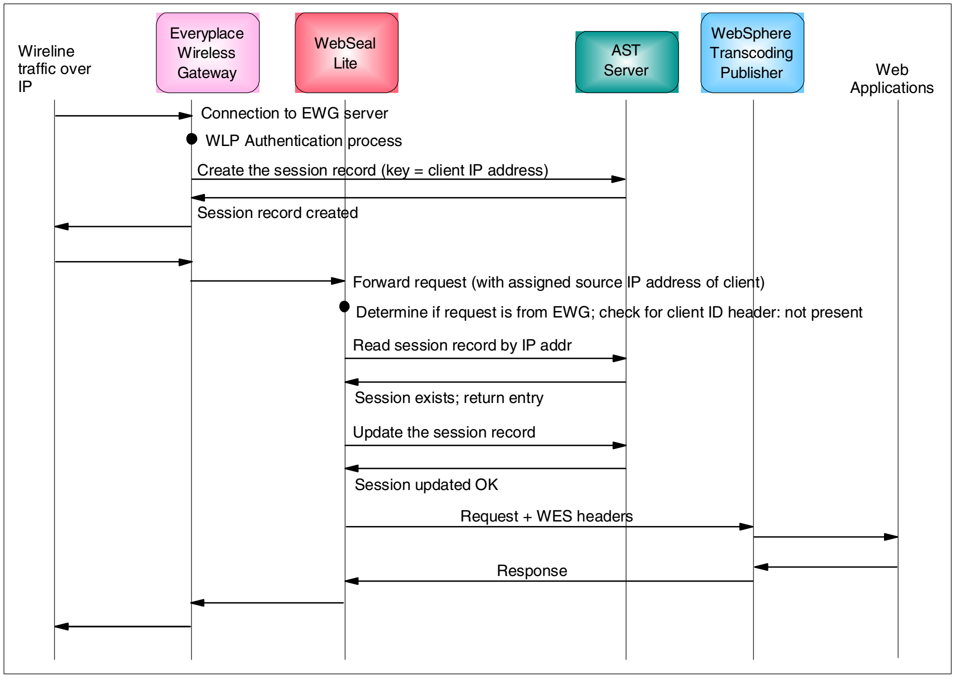

Wireline traffic over IP
Everyplace Wireless Gateway
WebSeal Lite
AST Server
WebSphere Transcoding Publisher
Web Applications
Connection to EWG server
WLP Authentication process
Create the session record (key = client IP address)
Session record created
Forward request (with assigned source IP address of client)
Determine if request is from EWG; check for client ID header: not present
Read session record by IP addr
Session exists; return entry
Update the session record
Session updated OK
Request + WES headers
Response

Figure 9-2 Wireless Gateway authentication process for a wireless client

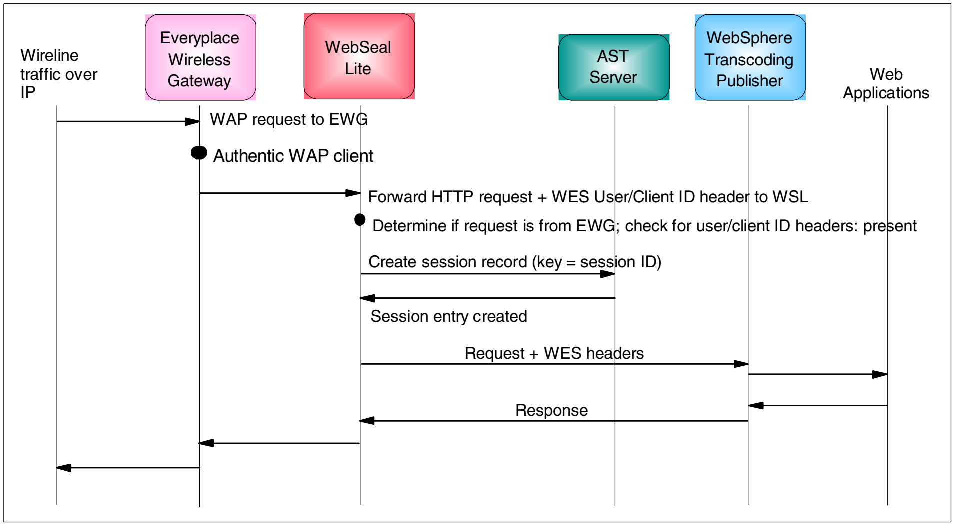
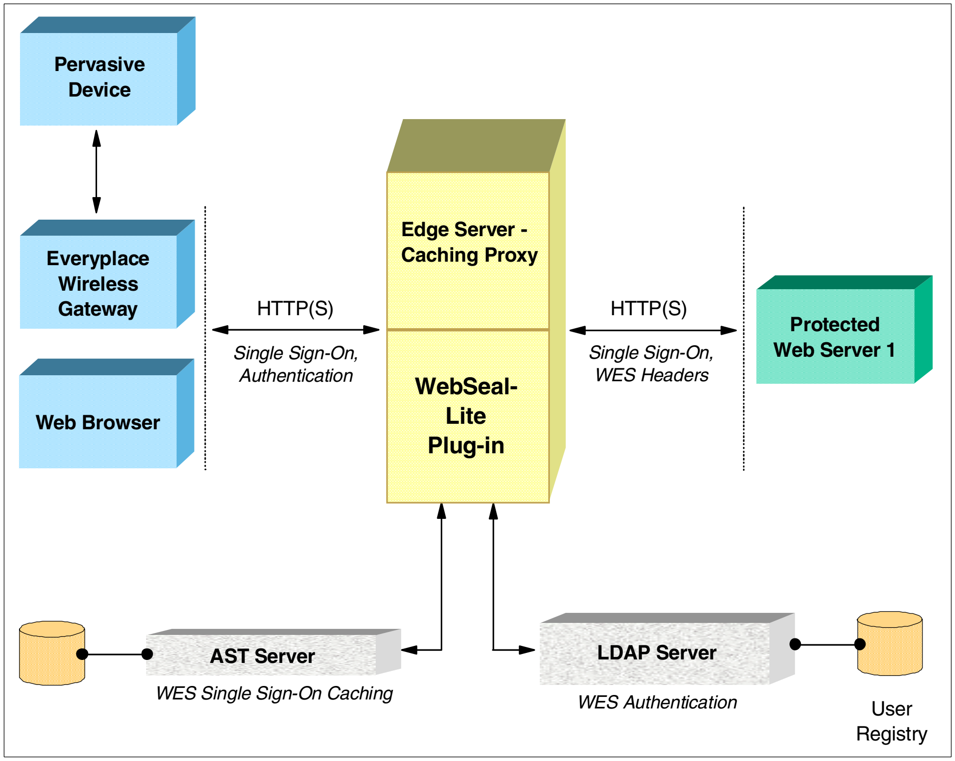

Wireline traffic over IP
Everyplace Wireless Gateway
WebSeal Lite
AST Server
WebSphere Transcoding Publisher
Web Applications
WAP request to EWG
Authentic WAP client
Forward HTTP request + WES User/Client ID header to WSL
Determine if request is from EWG; check for user/client ID headers: present
Create session record (key = session ID)
Session entry created
Request + WES headers
Response

Figure 9-3 Wireless Gateway authentication process for a WAP client

Authentication in WebSEAL-Lite

To enable single sign-on access to services within the WebSphere Everyplace Server domain, WebSEAL-Lite is made the central point of authentication. WebSEAL-Lite is designed to be the first point of entry for all HTTP traffic from the Internet and third-party gateways. The Everyplace Wireless Gateway also routes all HTTP requests to WebSEAL-Lite, which is the next hop after the Everyplace Wireless Gateway. WebSEAL-Lite intercepts all HTTP requests destined for WebSphere Everyplace Server services. It ensures that all the requests have been authenticated and tagged with the active session key.

Figure 9-4 shows WebSEAL-Lite's relationship to other WebSphere Everyplace Server components in the authentication process.



Pervasive Device
Everyplace Wireless Gateway
Web Browser
HTTP(S)
Single Sign-On, Authentication
Edge Server - Caching Proxy
WebSeal-Lite Plug-in
HTTP(S)
Single Sign-On, WES Headers
Protected Web Server 1
AST Server
WES Single Sign-On Caching
LDAP Server
WES Authentication
User Registry

Figure 9-4 Authentication with WebSEAL-Lite

If the requests intercepted by WebSEAL-Lite are from a trusted IP address that is owned by the Everyplace Wireless Gateway, then WebSEAL-Lite skips the authentication process and simply locates the associated active session from the Active Session Table. WebSEAL-Lite inserts the network and devices types of the session into the HTTP request, and forwards the request to the destination server.

If the request does not arrive from a trusted IP address, that is, it arrives from the Internet or a third-party gateway, then it must have authentication credentials included within its WWW-authorization header. If these credentials are not present, WebSEAL-Lite will challenge the user for a user name and password. If the credentials are present, for example in the user's response to the challenge, WebSEAL-Lite queries the SecureWay Directory to verify the user credentials. If the credentials are accepted, WebSEAL-Lite generates a session ID and creates a user session entry in the Active Session Table.

WebSEAL-Lite also adds WebSphere Everyplace Server HTTP headers to the request. These may include the session ID and location, user and realm, and client ID, as well as the network and device types. The content and downstream applications server will use these pointers to retrieve, from the Active Session Table and SecureWay Directory, information necessary to complete the user requests.

To illustrate the WebSEAL-Lite authentication process, we will walk through a series of user requests in this section. Figure 9-5 details the authentication process, using HTTP Basic Authentication, conducted by WebSEAL-Lite for an enrolled user from the Internet or third-party gateways. Even though it only depicts the first two consecutive requests, it does illustrate the process of authentication and active session management done by WebSEAL-Lite. The following six steps describe the process:

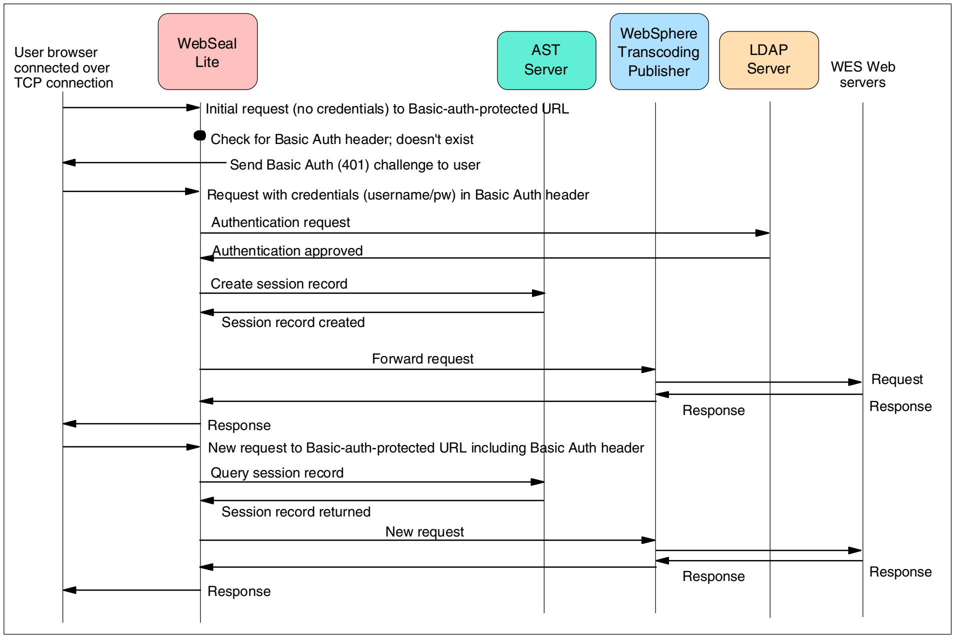

User browser connected over TCP connection
WebSeal Lite
AST Server
WebSphere Transcoding Publisher
LDAP Server
WES Web servers
Initial request (no credentials) to Basic-auth-protected URL
Check for Basic Auth header; doesn't exist
Send Basic Auth (401) challenge to user
Request with credentials (username/pw) in Basic Auth header
Authentication request
Authentication approved
Create session record
Session record created
Forward request
Request
Response
Response
Response
New request to Basic-auth-protected URL including Basic Auth header
Query session record
Session record returned
New request
Response
Response
Response

Figure 9-5 HTTP Basic Authentication for enrolled users through WebSEAL-Lite

1. Request credentials for first request

   Let’s assume that a user who is already enrolled in the WebSphere Everyplace Server domain is trying to access the services hosted by the WebSphere Everyplace Server. The user sends the first request by entering a URL in an HTTP browser from the Internet. Since WebSEAL-Lite acts as a reverse proxy, the URL actually points to WebSEAL-Lite instead of the Web server.

   WebSEAL-Lite receives the request. First, WebSEAL-Lite checks its own cache to see if it already knows the user. Next, it inspects the HTTP request header and cannot find any credentials. Finally, it checks the user IP address against the IP addresses managed by the Everyplace Wireless Gateway, which means this user request has not been authenticated by the Everyplace Wireless Gateway.

   WebSEAL-Lite determines that the user has not yet been authenticated by WebSphere Everyplace Server. It issues a return code 401 to the client HTTP browser, indicating failed authentication. The HTTP browser hence will start the HTTP basic authentication process using the WWW-Authenticate header, prompting the user to enter a user ID and password, then inserts them into the authorization header and forwards it to WebSEAL-Lite.

2. Look-up session

   When the user request with the authorization header is returned to WebSEAL-Lite, WebSEAL-Lite notices the authorization header with a user ID and password. It will first query the Active Session Table to see whether the user already has an active session. Since this is the first request from the user, no active session exists. WebSEAL-Lite thus moves on to step 3.

3. Authenticate the credentials

   Since the user has previously supplied the user ID and password, and there is no active session for the user request, WebSEAL-Lite must start the authentication and create an active session for the user if the authentication is successful.

   WebSEAL-Lite sends an LDAP authentication request to the SecureWay Directory. SecureWay Directory checks the user ID and password against the subscriber database. If the user ID and password match those in the subscriber database, the server returns authentication approved, and WebSEAL-Lite creates an active session entry for the user in the Active Session Table.

4. Forward request and response

   Once WebSEAL-Lite receives a message from the LDAP server indicating the authentication is approved, it directs the user request to the back-end servers for fulfillment. It usually retrieves responses from the Caching Proxy and Web server and sends the response back to the user.

5. Second request and look-up session

   The second request will have the credentials in the HTTP header. WebSEAL-Lite checks against the active session database to see whether the user has an active session or not. Since the user had a session established in step 3, WebSEAL-Lite considers the user to be pre-authenticated and does not check with the LDAP directory server again. Instead, WebSEAL-Lite updates the record in the Active Session Table to reflect the time of the last visit.

6. Forward request and response

   After updating the active session for the user, WebSEAL-Lite again directs the request to the appropriate back-end server. The request can be handled by either the Caching Proxy or the Web application servers. The response flows back through WebSEAL-Lite, which returns the response to the user.

The subsequent user requests will be handled repeatedly using steps 5 and 6 until either the user chooses to log out or the active session expires after a predefined period of time.

Figure 9-6 shows the same authentication process using forms-based login, rather than the HTTP basic authentication. Forms-based login allows content providers to present a more user-friendly interface for user authentication, as well as to add personalization.

In this case, instead of the HTTP 401 message for HTTP Basic Authentication, a custom login form is returned to the user, as well as a pre-login cookie. User credentials are extracted by WebSEAL-Lite from the returned POST request and authenticated. After a session ID is established, it is stored in the WebSphere Everyplace Server cookie and delivered with subsequent requests in order to track user sessions.

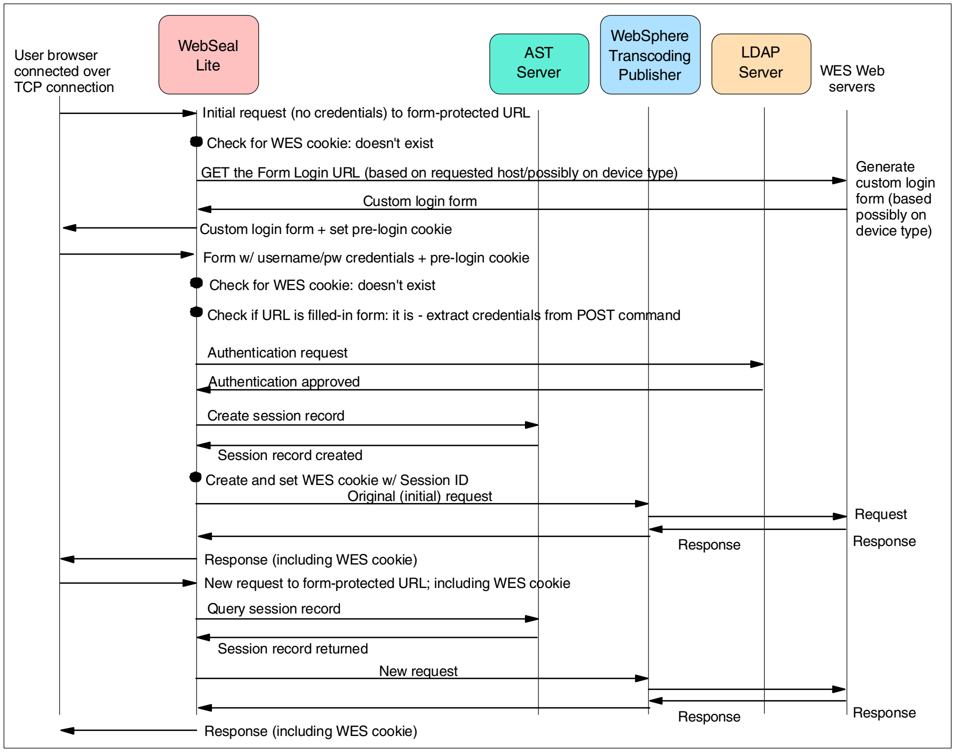

User browser connected over TCP connection
WebSeal Lite
AST Server
WebSphere Transcoding Publisher
LDAP Server
WES Web servers
Initial request (no credentials) to form-protected URL
Check for WES cookie: doesn't exist
GET the Form Login URL (based on requested host/possibly on device type)
Generate custom login form (based possibly on device type)
Custom login form
Custom login form + set pre-login cookie
Form w/ username/pw credentials + pre-login cookie
Check for WES cookie: doesn't exist
Check if URL is filled-in form: it is - extract credentials from POST command
Authentication request
Authentication approved
Create session record
Session record created
Create and set WES cookie w/ Session ID
Original (initial) request
Request
Response
Response
Response (including WES cookie)
New request to form-protected URL; including WES cookie
Query session record
Session record returned
New request
Response
Response
Response (including WES cookie)

Figure 9-6 Authentication process using forms-based login

Figure 9-7 shows the authentication process when digital certificates are used for client authentication.

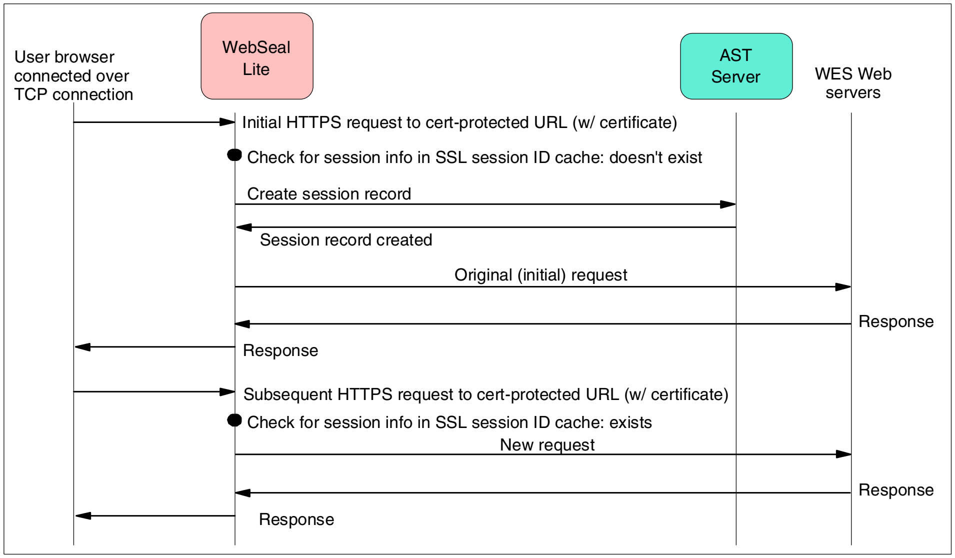

User browser connected over TCP connection
WebSeal Lite
AST Server
WES Web servers
Initial HTTPS request to cert-protected URL (w/ certificate)
Check for session info in SSL session ID cache: doesn't exist
Create session record
Session record created
Original (initial) request
Response
Response
Subsequent HTTPS request to cert-protected URL (w/ certificate)
Check for session info in SSL session ID cache: exists
New request
Response
Response

Figure 9-7 Authentication process using digital certificates

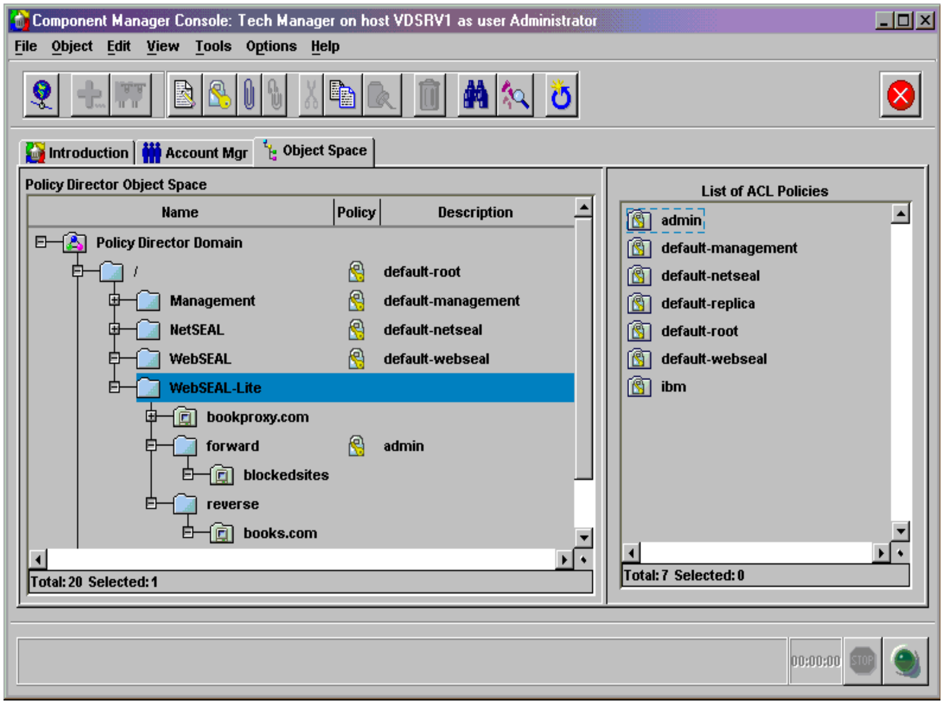
9.3.2 Authorization

WebSphere Everyplace Server achieves basic access control (all or nothing) by using HTTP proxy technology. Individual components might grant finer levels of access (read-only, read/write, update, create, delete, execute, traverse) to the user, but basic HTTP proxy has no way of mediating this. To address the need for centralized control with fine-grained access, WebSphere Everyplace Server Service Provider Offering includes Tivoli SecureWay Policy Director. If you choose to implement Policy Director, WebSEAL-Lite, Tivoli Personalized Services Manager, and Location Based Services integrate with it.

Policy Director is a standard component of WebSphere Everyplace Server Service Provider Offering (meaning that it is included in the CD-ROM package). However, in Service Provider Offering V2.1, integration with Policy Director is limited to WebSEAL-Lite, Location Based Services, and Tivoli Personalized Services Manager. Setup Manager does not install it and Suite Manager does not administer it. You must manually install Policy Director, and administer it through the Policy Director administration console.

Policy Director lets you centrally implement, manage and maintain access control within one domain or across multiple domains. Policy Director is the only way WebSphere Everyplace Server Service Provider Offering achieves access control and privacy (required by Location Based Services), and it is essential for single sign-on. Policy Director uses a database of object spaces, users, groups, and access control lists (ACLs). It can tell WebSEAL-Lite what ACL is associated with a specific object, what users and groups are in the ACL, and what permissions they have.

Object spaces, consisting of directories and data objects, have a hierarchical (tree) structure. Users and groups in lower branches of the tree inherit the properties of groups that appear higher in the hierarchy tree. Therefore, a group with permissions to access objects within one directory will, by default, also be able to access all objects in that directory's subdirectories, unless explicitly prevented from doing so by another ACL further down the directory tree. Figure 9-8 shows the hierarchy tree of the Management Console window.


Component Manager Console: Tech Manager on host VDSRV1 as user Administrator
File
Object
Edit
View
Tools
Options
Help
Introduction
Account Mgr
Object Space
Policy Director Object Space
List of ACL Policies
00:00:00
STOP

Name	Policy	Description
Policy Director Domain		
/		default-root
Management		default-management
NetSEAL		default-netseal
WebSEAL		default-webseal
WebSEAL-Lite		
bookproxy.com		
forward		admin
blockedsites		
reverse		
books.com		

Total: 20 Selected: 1

- admin
- default-management
- default-netseal
- default-replica
- default-root
- default-webseal
- ibm

Total: 7 Selected: 0

Figure 9-8 Policy Director's Component Manager Console

Policy Director's Component Manager Console provides a user-friendly interface for setting up ACLs and assigning them to specific object spaces. The Tivoli Personalized Services Manager enrollment process assigns users to their appropriate groups, and enters them into the Policy Director object space through the aznAPI. You can manage enrollees through either Tivoli Personalized Services Manager or Policy Director.

When using Policy Director for access control, WebSEAL-Lite checks with Policy Director to see if the user requesting an object has the appropriate permissions before sending the request to the back-end servers. These permissions are defined in ACLs associated with each object. If the user does not have the appropriate credentials, WebSEAL-Lite issues a challenge to the user. If the user cannot provide the appropriate credentials, then a `forbidden access` message or error page may be returned.

Figure 9-9 shows how WebSEAL-Lite uses Policy Director in the authentication process.

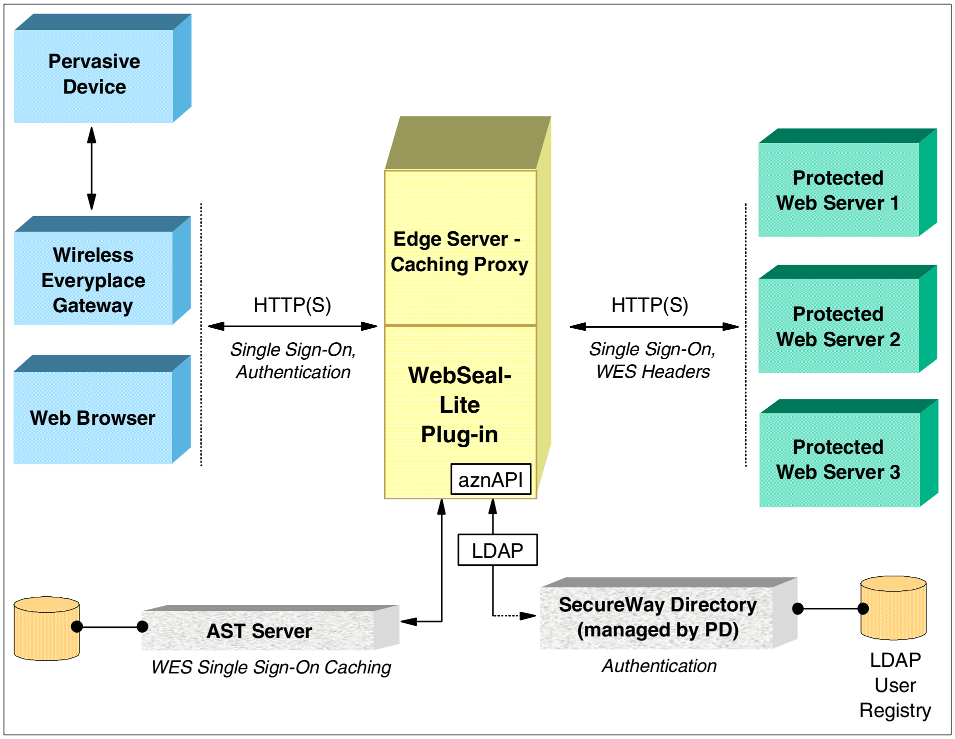


Figure 9-9 WebSEAL-Lite using Policy Director for authorization

Policy Director extends the basic LDAP schema for the SecureWay Directory instance that it manages. It then stores the user credential information in its copy of the SecureWay Directory using these schema extensions. Because the information is stored using this schema extension, any access to this credential data must occur through the authorization API.

By default, WebSEAL-Lite authenticates against Policy Director's LDAP directory when Policy Director is installed. This means that WebSEAL-Lite authenticates against the LDAP directory through Policy Director's authorization API, aznAPI, rather than through the SecureWay Directory client. It is possible to change the WebSEAL-Lite configuration to authenticate against the WebSphere Everyplace Server SecureWay Directory instead of Policy Director's copy.

Important: Authentication is a separate step from authorization. WebSEAL-Lite authenticates against Policy Director's LDAP directory using the aznAPI.

For authorization, Policy Director has a private database in which it stores the ACLs that associate each object space with the user groups having permissions to access that object, as well as the groups to which users belong. The Policy Director Management Server manages the master ACL and publishes replicas to the Authorization Servers, of which WebSEAL-Lite is one. WebSEAL-Lite checks against its copy of the ACL through the aznAPI. When there are changes to the ACL, the Policy Director Management Server updates WebSEAL-Lite's copy of the ACL and any entries in WebSEAL-Lite's cache.

Figure 9-10 outlines the authorization process for content contained in the WebSphere Everyplace Server domain and controlled by Policy Director. The following six steps discuss the process in more depth.

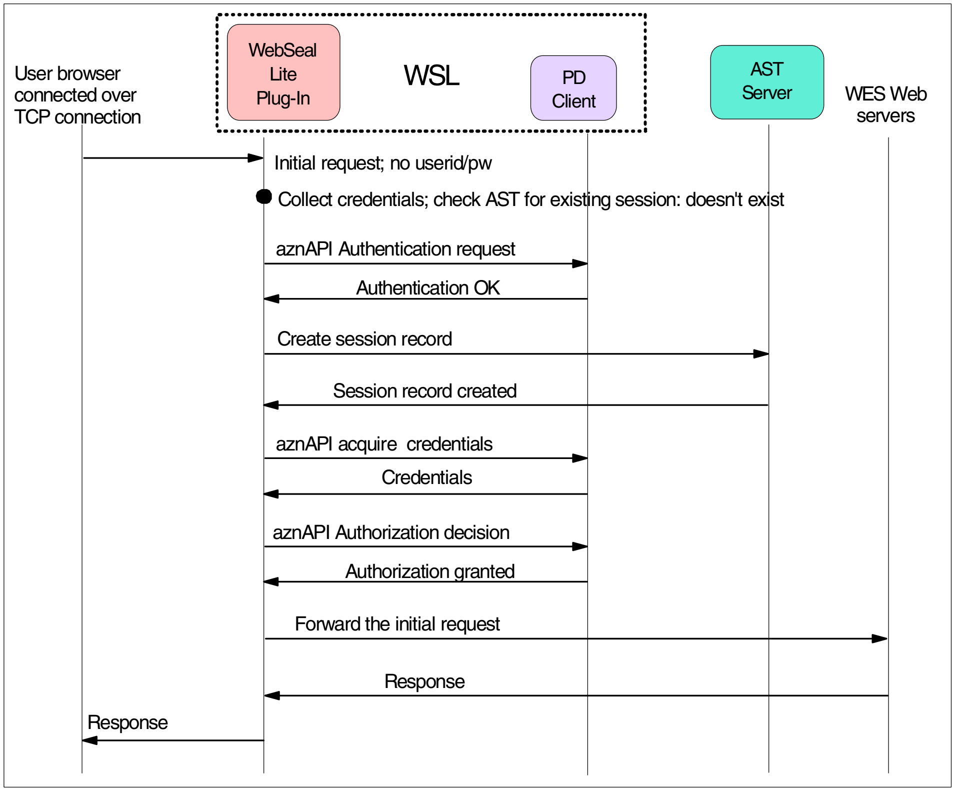

WebSeal
Lite
Plug-In
WSL
PD
Client
AST
Server
User browser
connected over
TCP connection
WES Web
servers
Initial request; no userid/pw
Collect credentials; check AST for existing session: doesn't exist
aznAPI Authentication request
Authentication OK
Create session record
Session record created
aznAPI acquire credentials
Credentials
aznAPI Authorization decision
Authorization granted
Forward the initial request
Response
Response

Figure 9-10 Successful authorization using Policy Director

1. Request credentials for first request

   Let's assume that a user is already enrolled in the WebSphere Everyplace Server domain, and is trying for the first time today to access the services hosted by the WebSphere Everyplace Server. Using an HTTP browser on the Internet, the user enters a URL or clicks on a link that resolves to a Web object within WebSEAL-Lite's domain.

   WebSEAL-Lite, acting as a reverse proxy, intercepts the request. It inspects the HTTP request header and, in this example, cannot find any credentials. To get the user credentials, WebSEAL-Lite uses either basic authentication, form-based login, or digital certificates as we discussed in the previous section.

   Once WebSEAL-Lite collects the user credentials, it checks its own cache to see if it can authenticate against a cached copy. Failing that, WebSEAL-Lite checks the Active Session Table for an existing session. In this case, none exists, so it must authenticate the credentials.

2. Authenticate the credentials

   WebSEAL-Lite authenticates the user credentials and, if the authentication is successful, creates an active session entry in the Active Session Table for the user. In this configuration, WebSEAL-Lite uses Policy Director for authentication (although you can manually change the WebSEAL-Lite configuration to use the WebSphere Everyplace Server LDAP directory server instead).

   In the Policy Director vernacular, WebSEAL-Lite is both an application and an Authorization Server, meaning it both asks for and fulfills requests for credential authentication/authorization. The Policy Director Management Server replicates its ACL database and any dynamic changes to WebSEAL-Lite for local storage However, WebSEAL-Lite does not bring individual user entries into memory until a session is established.

Policy Director dynamically updates WebSEAL-Lite's cache with any AC changes.

   To continue with our example, WebSEAL-Lite sends the authentication request through the aznAPI to the local Authorization Server replica. If the user ID and password match those in the subscriber database, the server will return authentication as successful. WebSEAL-Lite will then create an active session for the user.

3. Authorize the user

   Once WebSEAL-Lite authenticates the user, it checks authorization to determine whether the user has the appropriate permissions for the action requested. WebSEAL-Lite first requests the user's credentials or permissions from Policy Director and stores this information in a local cache to improve performance of subsequent requests to protected materials. WebSEAL-Lite then requests an authorization decision from Policy Director based on the user's credentials, the type of request, and object requested. In our example, Policy Director responds that the user has the appropriate permissions.

4. Once authorization for the requested object has been granted, WebSEAL-Lite forwards the user request to the back-end WebSphere Everyplace Server servers.

Figure 9-11 shows the authorization process when the user does not have the appropriate permissions. In this case, step 4 returns an `HTTP 403 Forbidden` message. The user may be prompted for additional user credentials in order to access the protected materials. For subsequent requests, WebSEAL-Lite uses the cached Policy Director credentials to determine the user's authorization to access Web objects.

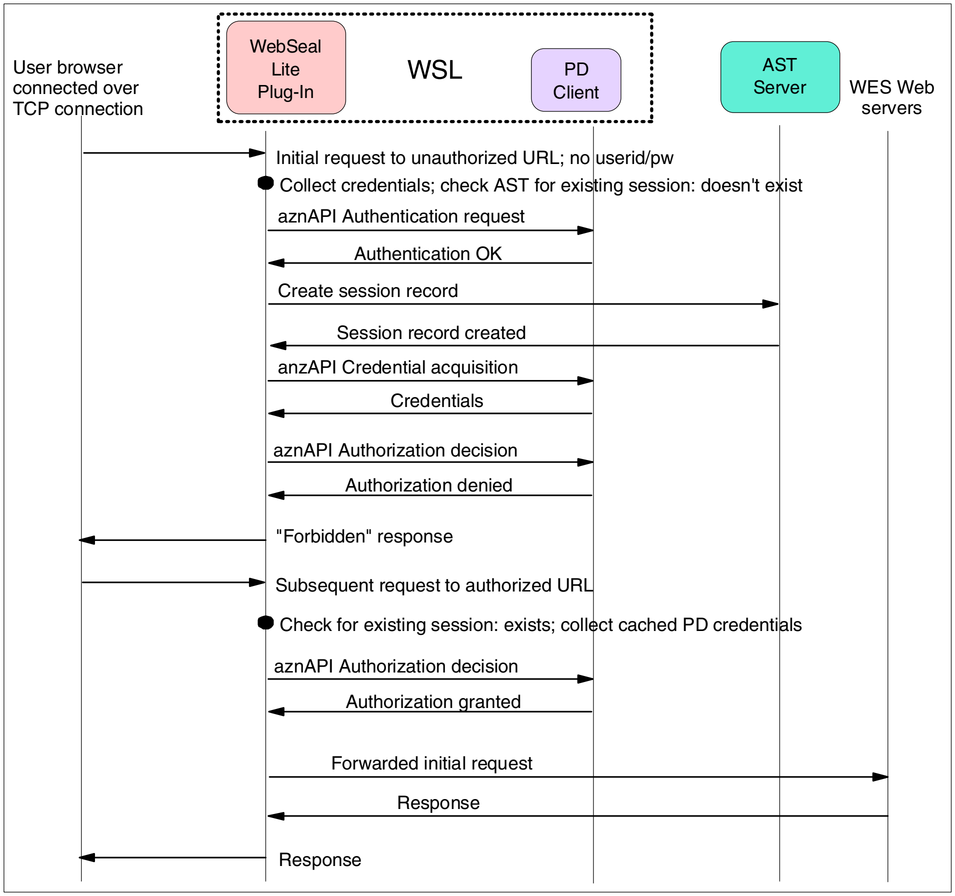

WebSeal Lite Plug-In
WSL
PD Client
AST Server
User browser connected over TCP connection
WES Web servers
Initial request to unauthorized URL; no userid/pw
Collect credentials; check AST for existing session: doesn't exist
aznAPI Authentication request
Authentication OK
Create session record
Session record created
anzAPI Credential acquisition
Credentials
aznAPI Authorization decision
Authorization denied
"Forbidden" response
Subsequent request to authorized URL
Check for existing session: exists; collect cached PD credentials
aznAPI Authorization decision
Authorization granted
Forwarded initial request
Response
Response

Figure 9-11 Authorization failure

9.3.3 Confidentiality

WebSphere Everyplace Server achieves confidentiality by providing encrypted connections between clients and WebSphere Everyplace Server, as well as between WebSphere Everyplace Server components. The WebSphere Everyplace Server uses a set of industry-standard security technologies to achieve the goal of confidentiality. Such technologies include SSL, TLS, and wireless transport layer security (WTLS) for WAP clients. In addition, the Everyplace Wireless Gateway creates secure tunnels to achieve confidentiality for Everyplace Wireless Gateway wireless clients (non-WAP) by using a modified version of the Point-to-Point Protocol (PPP) called the Wireless Link Protocol (WLP), formerly called the ARTour Link Protocol (ALP).

Connections within a common deployment of WebSphere Everyplace Server can be configured to implement confidentiality by enabling appropriate technology, as displayed in Figure 9-12.

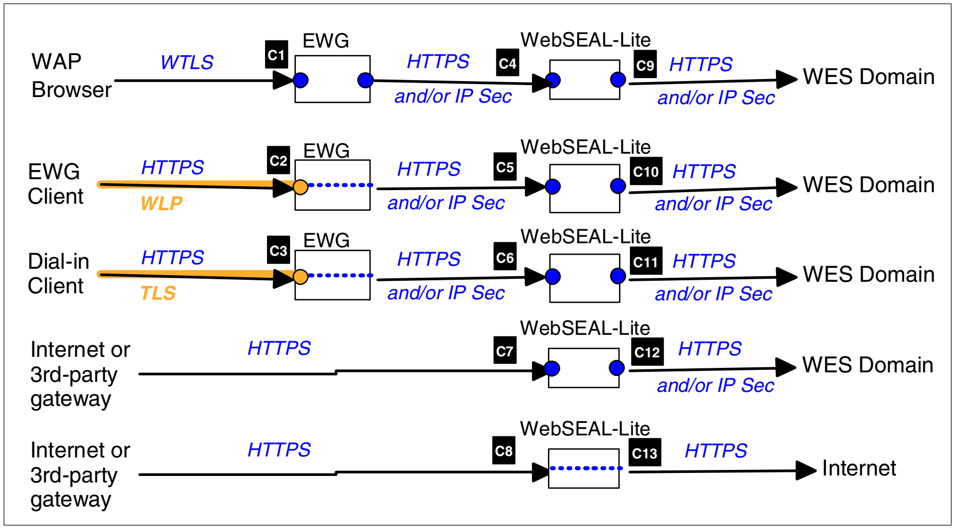


Figure 9-12 Secure connections

- **C1**: Connection between WAP clients and Wireless Gateway: the Everyplace Gateway provides Wireless Transport Layer Security (WTLS) support for WAP devices. The WTLS connection is between the WAP device and the wireless gateway.
- **C2**: Connection between Everyplace Wireless Gateway clients and Wireless Gateway: the Everyplace Gateway provides end-to-end SSL (HTTPS) for the wireless clients as well as for dial up and LAN connected clients. The feature is optional and when enabled, the HTTPS traffic is encapsulated in the WLP protocol with the wireless client. The Everyplace Gateway provides end-to-end SSL for these clients. The connection will not be broken in the wireless gateway in this case. After the client and server authenticate each other, an encrypted tunnel is established between the gateway and the client using, DES, RSA's RC 5 or triple DES which are strong symmetric key encryption algorithms using the keys created as part of the authentication process, double encrypting the traffic between the wireless client and the Everyplace Wireless Gateway.
- **C3**: Connection between dial-up clients and Wireless Gateway: the Everyplace Wireless Gateway can use a TLS connection and create an encrypted tunnel for the clients.
- **C4**, **C5**, **C6**: Connection between IBM Everyplace Wireless Gateway and WebSEAL-Lite: the Everyplace Wireless Gateway provides the capability to

enable TLS between itself and WebSEAL-Lite running on the Edge Server Caching Proxy, or directly to back-end Internet/intranet Web servers.

- C7, C8: Connection from Internet to WebSEAL-Lite: for the HTTP clients from the Internet, SSL can be enabled between browsers and WebSEAL-Lite running with Edge Server Caching Proxy.
- C9, C10, C11, C12: Connection between WebSEAL-Lite and secure application servers within the WebSphere Everyplace Server domain: even though it is considered to be trusted, users have the option to enable a TLS or IPSec connection between WebSEAL-Lite and application servers.

Note: A connection between the WebSEAL-Lite and WebSphere Transcoding Publisher cannot be encrypted; otherwise, WebSphere Transcoding Publisher would not be able to transcode.

- C13: Connection between WebSEAL-Lite and Internet/Intranet: this connection generally is considered prone to security compromise. It is recommended that SSL be enabled if possible.

9.3.4 Data integrity

WebSphere Everyplace Server achieves data integrity using features such as message digests and certificates included in the security technologies TLS and WTLS. These protocols contain features that prevent data from being corrupted or manipulated during transit. For example, WTLS uses SHA 1 Message Authentication Codes (MAC) to ensure that data is not corrupted or modified during transit. This is desirable to prevent “spoofing”, where a malicious user attempts to assume the identify. It also invalidates messages that may have become corrupted in transit due to outside interference or network problems.

9.3.5 Non-repudiation

Since WebSphere Everyplace Server uses TLS and WTLS with digital certificates, the objectives of non-repudiation can be achieved at the transaction level. When a user has signed a document with a digital certificate, it is assumed that only that user has access to the private key. In this context, the meaning of non-repudiation does not take into account the possibility of private key theft or certificate misuse, essentially a form of identity theft.

9.3.6 Summary of security implementation

Table 9-1 summarizes the security capabilities of IBM WebSphere Everyplace Server.

Table 9-1 WebSphere Everyplace Server security implementation

Technology used in Everyplace Server	SSL	WTLS	PPP/WLP	Proxy	Firewall
Components using the technology	WebSEAL-Lite Wireless Gateway	Wireless Gateway	Wireless Gateway	WebSEAL-Lite	not bundled
Authentication	Yes	Yes	Yes	Yes	No
Confidentiality	Yes	Yes	Yes	No	No
Authorization	No	No	No	Yes	No
Integrity	Yes	Yes	Yes	No	No
Non-repudiation	Possible	Possible	No	No	No

9.4 Security considerations

Many security decisions depend on the application deployed within WebSphere Everyplace Server. The locally available carriers will determine whether the Everyplace Wireless Gateway should be used or a third-party gateway provided by the carrier. Whether a third-party gateway can be trusted will also affect where authentication should take place.

The type of authentication used will also largely depend on the capabilities of the mobile devices that will be used, for example, whether wireless client software can be installed on the device to facilitate WLP authentication at the gateway. The user interface of the device should be considered as well. Devices with limited input capabilities, such as cell phones, provide poor usability when long or complicated user names and passwords are required.

9.4.1 Securing the WAP gap

WTLS was designed for wireless devices and networks. It is not compatible with SSL (TLS) used by today’s Web and application servers. Thus, a wireless gateway must convert WTLS to and from SSL. This conversion creates a security risk known as the “WAP gap”. The gap occurs during the conversion, after the massage has been decrypted from WTLS but before it has been encrypted with TLS, and vice versa; see Figure 9-13. At this point, brief as it may be, the message lies exposed in the memory of the gateway and vulnerable to malicious use. Therefore, it is important that the Wireless Gateway run in a secure domain, protected from intruders on both sides.

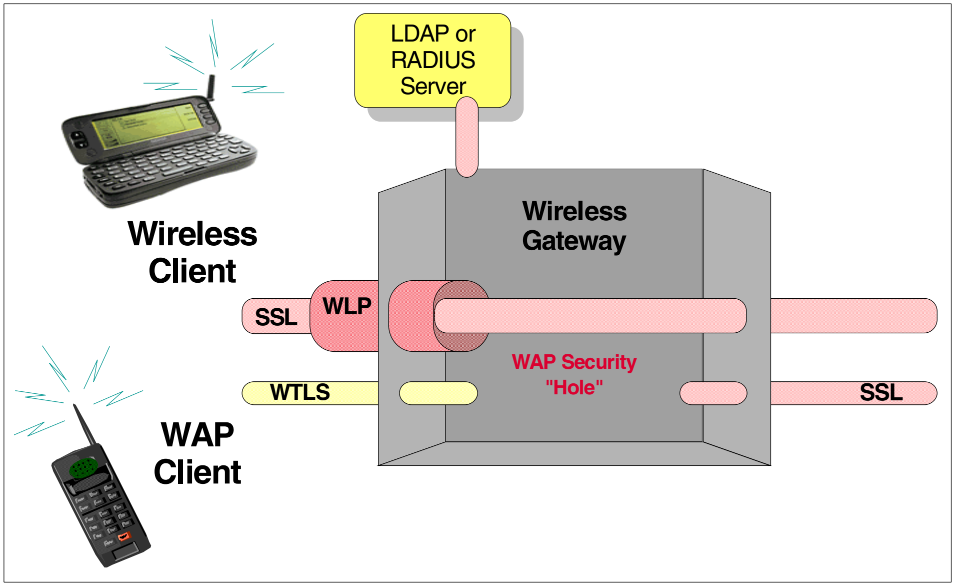


Figure 9-13 WAP gap

An option to secure the WAP gap is to deploy primary and secondary Everyplace Wireless Gateway instances in a cluster, the primary being in the WebSphere Everyplace Server domain and the secondary existing in the secure enterprise domain. Thus the WTLS session will be maintained all the way to the enterprise domain before the WTLS session converted to SSL (or clear text).

The Wireless Gateway must be a trusted server in WebSphere Everyplace Server. WebSphere Everyplace Server Setup manager installs an entry in the LDAP directory as a trusted Wireless Gateway.

9.4.2 Firewall considerations

To complete the security picture, use firewalls with Everyplace Server.

The general objectives for firewalls are to:

- Only allow traffic flow that is determined safe and in our interests.
- Give away a minimum of information about our private network.

- Keep track of access attempts to identify suspicious behavior.

Figure 9-14 depicts the most generic deployment model of Everyplace Server that suits the needs of many content providers, network operators, service providers, and enterprises. The Everyplace Server domain can be protected by two or three firewalls. For clarity, the firewalls are labelled **a**, **b**, **c**, and **d** in the figure. Firewall **a** controls all access to Everyplace Server from third-party gateways or the Internet. Firewall **b** is placed between the Everyplace Gateway and devices connecting from IP networks. Firewall **c** is optional and can be the same as firewall a. Additionally, WebSEAL-Lite (**d**) can use the underlying Edge Server Caching Proxy with multiple network adaptors to achieve certain firewall functions. Alternatively, a firewall can be installed between the wireless gateway and WebSEAL-Lite.

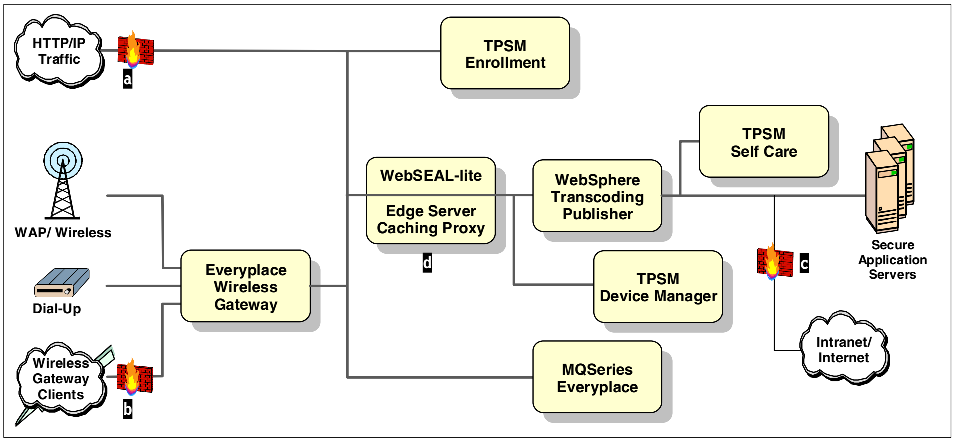


Figure 9-14 Firewall protection for the Everyplace Server domain

The firewall configuration guidelines are given as the following:

Firewall **a** is configured to:

- Allow HTTP requests destined from IP addresses that are not owned by Everyplace Wireless Gateway for WebSphere Everyplace Server services and route such requests to WebSEAL-Lite.
- Allow HTTP requests destined for the public Web servers and/or the enrollment server, originating from IP addresses that are not owned by Everyplace Wireless Gateway, and route such requests to Internet and/or enrollment server.
- Reject all other packets.

Firewall **b** is configured to:

- Allow IP requests destined for the pre-defined Everyplace Wireless Gateway IP ports but from any IP address.
- Allow HTTP requests for Everyplace Server services.
- Reject all other packets.

Firewall **c** is configured to:

- Allow outbound HTTP requests destined for the public Web servers.
- Allow inbound HTTP response for the active user session from the public Web servers.
- Reject all other packets.

Firewall **d** is configured to:

- Allow HTTP requests destined for the Edge Server services.
- Use separate network interface for the internal network.
- Reject all other packets.

Part 4

Integration

There are other components that do not come integrated into the IBM WebSphere Everyplace Server bundle. Some are from IBM such as the IBM WebSphere Voice Server and some are from third-party vendors such as third-party wireless gateways. In this part we discuss how to integrate the IBM WebSphere Voice Server and third-party gateways into the IBM WebSphere Everyplace Server environment.

10

Third-party gateways

The components of IBM WebSphere Everyplace Server are integrated in that they understand how to communicate with one another. The IBM Everyplace Wireless Gateway and WebSEAL-Lite know how to communicate. WebSphere Everyplace Server allows the use of a WAP gateway other than the IBM Everyplace Wireless Gateway. Since third-party gateways do not necessarily communicate in the same manner as IBM Everyplace Wireless Gateway, additional integration software is required. This chapter discusses the method by which third-party gateways are integrated with IBM WebSphere Everyplace Server.

The gateway-specific plug-in is written by IBM, the gateway providers, or customers. Currently IBM had developed and tested the following third-party gateway plug-ins:

- Nokia
- Ericsson
- Motorola

> **Important:** Sample plug-ins are available from IBM on an as-is basis for the above named gateways, and they are only available to licensed users of IBM WebSphere Everyplace Server. To access these samples requires your license key and registration with IBM. For access refer to the following URL: `http://www-3.ibm.com/pvc/products/wes_enable/code_fixes.shtml`.

10.1 Third-party gateway support

Figure 10-1 illustrates the basics of how a third-party gateway integrates into the IBM WebSphere Everyplace Server environment. Support for third-party gateways requires the use of a Edge Server Caching Proxy plug-in. This plug-in is gateway specific and is written in C or C++ for the Edge Server Caching Proxy.

When a request comes from a WAP device through the third-party gateway it is routed to WebSEAL-Lite. WebSEAL-Lite searches SecureWay Directory for third-party gateway definitions and invokes the appropriate plug-in. Using the Edge Server Caching Proxy APIs the plug-in must obtain the HTTP header, the URL, the query string, and other information about the current request, then parse this information and return to WebSEAL-Lite a client ID. This client ID is the identification, such as the MSISDN, that identifies the user who initiated the HTTP request.

WebSEAL-Lite then uses the client ID to search the `cn=users` portion of IBM SecureWay Directory for the ePerson entry whose client ID attribute matches the client ID returned by the plug-in. If a matching ePerson entry is found, its user ID and organizational unit attributes are used to construct the ID placed on the X-IBM-PVC-User HTTP header, and the client ID is placed on an X-IBM-PVC-Client-id HTTP header. Both of these headers are then available for use by downstream components.

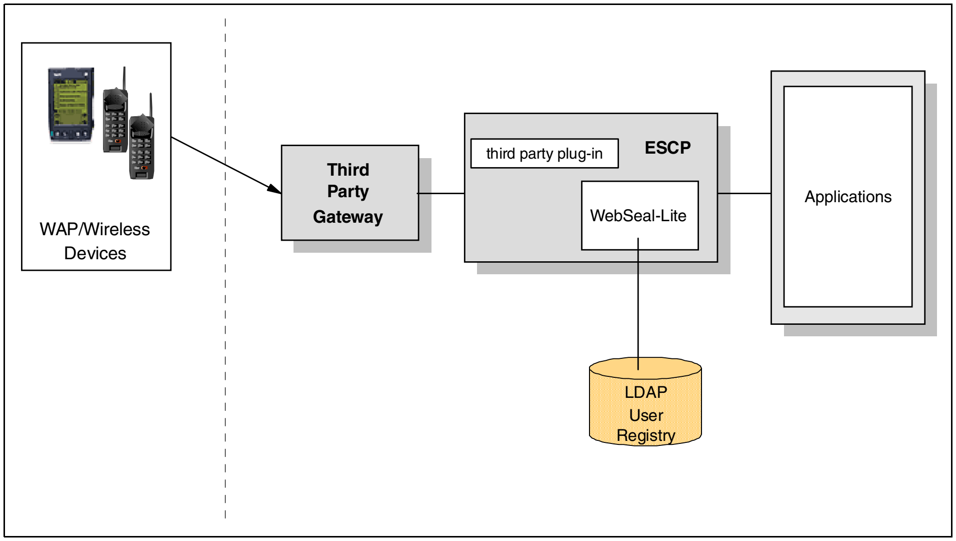


Figure 10-1 Third-party gateway architecture

For up-to-date information on third-party gateway support refer to the third-party gateway support documentation located on the IBM WebSphere Everyplace Server InfoCenter CD. You may access it from the InfoCenter main page by selecting **WebSphere Everyplace Server -> Configure -> Gateways -> Third party gateway support.** The InfoCenter is also available on the Internet at `http://www-3.ibm.com/pvc/products/wes_provider/infocenter/index.html`

10.2 Installing third-party gateway support

General installation instructions for the plug-in are provided in the third-party gateway support document from the InfoCenter. Before installing the IBM sample third-party gateway plug-in refer to the specific installation instructions provided in the package that is available for authorized users.

General third-party gateway installation instructions are as follows:

1. Code the plug-in routine for each third-party gateway you plan to use.
2. Create a shared library containing the plug-in and place it in an appropriate directory with appropriate file ownership and permissions. Shared libraries typically have a `.so` file extension and reside in /usr/lib/. Detailed plug-in information is shown in 10.5, “Customer supplied gateway plug-in” on page 398.
3. Shut down Edge Server Caching Proxy so that WebSEAL-Lite is no longer running.
4. Make a backup copy of your existing WebSEAL-Lite shared library. The library is shipped with Everyplace Server as wesauth.so and is typically installed in /usr/bin/.
5. If you want to capture additional debugging information in the Caching Proxy log files, replace the library with wesauth.so.debug. The file ownership and permissions must match those of the original wesauth.so library.
6. Choose a location in IBM SecureWay Directory for each type of third-party gateway you plan to use.

> **Note:** Multiple instances of a particular type of third-party gateway may share the same SecureWay Directory location.

This location must be under the baseDN (the distinguished name) specified in your WebSEAL-Lite configuration file, but *not* under the `cn=root, sys=ewg` entry, because that section is reserved for use by Everyplace Wireless Gateway.

The configuration file is usually called ibmwesas.conf, and is installed in the following locations:

- For AIX:

  /usr/lpp/IBMEPS.Auth/

- For Solaris:

  /opt/IBMEPSAu/

7. For each gateway, add a gatewayDN statement to your WebSEAL-Lite configuration file. This statement specifies the gateway's location in the SecureWay Directory tree, relative to the baseDN. The syntax for this statement is:

   gatewayDN DN_relative_to_baseDN

   (for example: `gatewayDN cn=mygateway`).

8. If they are not already located within your SecureWay Directory schema, add the `wlGateway` and `wlMni` object classes by performing the following steps.

   a. Use the SecureWay Directory administration Web pages to make a backup copy of the SecureWay Directory tree and schema (LDIF).

   b. Run the `wg_schema` program to add these object classes.

   c. To access the SecureWay Directory administration Web pages, point your browser to `/ldap` on the machine running the SecureWay Directory server (for example `http://dirserver2.company.com/ldap`). You must have a Web server running on the same machine as the SecureWay Directory server.

      The syntax is:

   wg_schema -h <ldap_server_hostname> -D <admin_DN> -w <admin_pw>

   As an alternative to the method detailed above, you can create the wlGateway and wlMni object classes automatically by performing the following steps:

   a. Install Everyplace Wireless Gateway on one machine in the WebSphere Everyplace Server domain.

   b. Use the Everyplace Wireless Gateway Gatekeeper to configure one WAP Gateway.

   c. Everyplace Wireless Gateway creates the object classes automatically.

9. Use the Directory Management Tool (see Web site `http://www-4.ibm.com/software/network/directory/library/publications/31/dmt/dparent.htm` for instructions on using the Directory Management Tool). Add entries to the SecureWay Directory as follows:

   a. Add a container entry at the spot in the tree you identified on the gatewayDN statement in the configuration file. The RDN (Relative distinguished name) for this entry must match that on the gatewayDN statement in the configuration file. The RDN (Relative Distinguished Name) for this entry must match that on the gatewayDN statement. Continuing with the example in step 7, you would use cn=mygateway as the RDN.

b. Under the container entry add an ePropertySet. A typical RDN for this entry might be cid=Common, but the RDN is not important.

c. Under the ePropertySet add the following eProperty entries:

   - An entry with an RDN of settingID=pluginName. Set the entry's cesProperty attribute to the name of your plug-in for this gateway. The plug-in name must be of the form: library_name:function_name where library_name is the name of the shared library that contains function_name. An example plug-in name is myLibrary.so:getClientID.
   - An entry with an RDN of settingID=clientIDAttributeName. Set the entry's cisProperty attribute to the name of the attribute in the ePerson object that you use to locate, in SecureWay Directory, the user associated with the client ID returned from your plug-in. For example, if your plug-in returns an MSISDN number, you could set this eProperty to internationalISDNNumber.
   - (optional) An entry with an RDN of settingID=pluginToken. Set the entry's cesProperty to a string token that you want passed to your plug-in. Any use of the string token is defined solely by your plug-in. For example, it could be used to pass in the name of an HTTP header, a gateway type, or a plug-in version number.

d. Under the container entry, add wlGateway or wlMni entries to identify specific instances of the type of third-party gateway that you have installed. The RDN is not important (for example, cn=mygateway1). When adding a wlGateway, specify a structural object class of wlGateway. When adding a wlMni, specify a structural object class of wlMni.

   The wlGateway entry identifies a single gateway machine, while the wlMni entry can identify a range of IP addresses running such gateways. If you add a wlGateway entry, you must specify an IP address in the host attribute. If you add a wlMni entry, you must specify an IP address in the netaddr attribute and a mask in the netmask attribute; the mask is applied to the IP address to define a range of IP addresses.

10. Restart Edge Server Caching Proxy. As with all SecureWay Directory information used by WebSEAL-Lite, if you change any of the SecureWay Directory information used in WebSEAL-Lite's third-party gateway, you must either issue an authsrv refresh command or stop and restart Caching Proxy.

If you don't, the changes might not have any effect on WebSEAL-Lite operation.

11. If you installed the debug version of `wesauth.so`, you can set the debug message level via the `debugLevel` statement in the WebSEAL-Lite configuration file to provide debugging/progress information in one of the Caching Proxy log files.

12. You must now integrate ClientID support into the Tivoli Personalized Services Manager applications to assign unique IDs to Everyplace Server users. For example, you could configure Customer Care to use ClientIDs as follows:

    a. Update the `/usr/TivTSM/custcare/content/JPanSearch.cfg` file on AIX, or the `/usr/TivTSM/custcare/content/JPanSearch.cfg` file on Solaris to modify the following two lines:

    ```
    SP_Defined_1=MSISDN
    SP_Defined_2=
    ```

    b. Save the file.

    c. Restart the customer care servlet using the WebSphere adminclient tool.

    d. When a customer service representative logs into Customer Care, they have the option to add or modify the value of the MSISDN field of any user.

    Note: MSISDN is the name of the field where the client ID is specified. You may give this field another name (for example, ClientID).

Important: If you installed Policy Director in the WebSphere Everyplace Server domain, ensure that you do not put the Policy Director user data under the same LDAP tree of Tivoli Personalized Services Manager. For example, you should not put user data under "`ou=IBM, dc=tw, dc=ibm, dc=com`". Doing so may cause the Authentication Server to retrieve the wrong user information from the Policy Director tree.

10.3 Unique Client ID support

If using client ID as the third-party authentication mechanism in the Everyplace Server environment, you must add further validation to the Tivoli Personalized Services Manager sub-components to ensure that client IDs are unique. By default, there is no data validation of end user input for client ID, which allows duplicate or incorrect client IDs to be created. For additional information about validating profile extensions, please contact your IBM service representative.

WebSEAL-Lite creates an X-IBM-PVC-Client-ID header containing the value returned from the plug-in if that header does not already exist in the HTTP stream. Some Everyplace Server functions, such as Location Proxy, depend on this header to work correctly. Therefore, the plug-in must return the correct value format. The format is as follows:

```
X-IBM-PVC-Client-Id
```

This provides additional information to uniquely identify the user's session and/or device. The value is a token that represents a single device and depends on the connection type and/or connecting gateway. The token consists of a variable-length string that is the device identifier in the format dictated by the type indicator and a two-digit type indicator which can be:

- `08` - indicates the client ID is an IP address
- `1F` - indicates the client ID is a phone number (caller number or MSISDN)
- Examples:
  - `x-ibm-pvc-client-id: 089.37.58.55`
  - `x-ibm-pvc-client-id: 1F0119195551212`

10.4 Performance considerations

To improve the performance of the SecureWay Directory search for ePerson entries with a client ID attribute that matches that returned from your plug-in, you can create a DB2 index for that relationship. WebSEAL-Lite caches the result of these lookups in memory so that not every request from your third-party gateways involves an SecureWay Directory search, but the DB2 index is still beneficial to performance.

This support adds one or more caches for SecureWay Directory information mapping client IDs to user information. The size and cleanup interval of these caches is the same as for the Active Session Table cache and is controlled by the `MaxSessionCache` and `ASTCleanupInterval` configuration values, respectively. These new caches are flushed when an `authsrv flush` or `authsrv refresh` command is issued.

Since the plug-in is called for every request coming through a third-party gateway, it is important that the plug-in be efficient and not perform lengthy operations unless absolutely necessary.

10.5 Customer supplied gateway plug-in

The customer-supplied gateway specific plug-in is responsible for returning a client ID for the current request. This routine runs as part of the same process and threads as WebSEAL-Lite. Because Caching Proxy plug-ins run in a multithreaded environment, the gateway-specific plug-in must be thread-safe. The plug-in has access to the HTTP headers, URL and query string, and other information about the current request through the same Caching Proxy plug-in APIs as WebSEAL-Lite. API `httpd_getvar()` is used to obtain the request information.

The plug-in can use `HTTPD_log_error()` to write information to the Edge Server Caching Proxy error log, which is identified on the ErrorLog directive in the Caching Proxy configuration file. Caching Proxy API information is available in the *Web Traffic Express Programming Guide* available from:
http://www.ibm.com/software/webservers/edgeserver/library.html

You can have multiple plug-ins in the same library and coded in the same source code file. However, they must have different function names.

10.5.1 Interface

The plug-in receives a buffer into which it must place the client ID. The plug-in must also set a return code indicating its success or failure.

The C++ prototype for this function is:

```
extern "C" int function_name(const unsigned char *logHandle, char *buffer,
int *bufferSize, const char *token);
```

The C prototype for this function is:

```
int function_name(const unsigned char *logHandle, char *buffer, int
*bufferSize, const char *token);
```

The variables you should enter are defined as follows:

- logHandle: handle to the Edge Server Caching Proxy log. Used on Edge Server Caching Proxy APIs only.
- buffer: pointer to a buffer provided by the caller to hold the client ID upon return from this function
- bufferSize: pointer to a variable containing the size, in bytes, of the buffer provided by the caller
- token: pointer to a string token that was fetched from the `pluginToken` eProperty in this gateway's configuration section in SecureWay Directory. This token's content, format and use are under complete control of the

plug-in. If the pluginToken eProperty is not defined in SecureWay Directory or has no value, the token argument is NULL when this plug-in is called.

Output is shown in the form of the following return codes indicating the function's success or failure:

- 0: The client ID was obtained and copied to the caller-supplied buffer as a null-terminated string.
- 1: The caller-supplied buffer is too small to hold the entire client ID. When returning this code, this function must first update the size variable pointed to by the `bufferSize` property to contain the size of the buffer required to hold the result. In that case, the caller acquires a buffer of the requested size and calls this function again.
- 2: This function was unable to return the client ID. No specific reason is implied by this return code.
- 100-199: These return codes are reserved for plug-in use. They may be used to indicate a specific error instead of returning return code 2. They are treated the same as return code 2 by the caller.

10.5.2 Creating the shared library

The plug-in must reside in a shared library accessible to WebSEAL-Lite during request processing. A shared library containing a C++ plug-in can be created using compile and link commands similar to the following:

- C++ plug-in using IBM CSet++ on AIX:

```
xlC_r -c myPlugin.cxx -I/usr/samples/internet_server/API
makeC++SharedLib -p0 -o myPluginLibrary.so
-bI:/usr/samples/internet_server/API/libhttpdapi.exp myPlugin.
```

- C++ plug-in using Sun Workshop on Solaris:

```
CC -c -mt myPlugin.cxx -I/usr/samples/internet_server/API
CC -G -mt -o myPluginLibrary.so myPlugin.o
-L/usr/samples/internet_server/API -lhttpdapi
```

- C plug-in using IBM CSet++(R) on AIX:

```
xlc_r -c myPlugin.c -qcpluscmt -I/usr/samples/internet_server/API
makeC++SharedLib -p0 -o myPluginLibrary.so
-bI:/usr/samples/internet_server/API/libhttpdapi.exp myPlugin.o
```

- C plug-in using Sun Workshop on Solaris:

```
cc -c -mt myPlugin.c -I/usr/samples/internet_server/API
cc -G -mt -o myPluginLibrary.so myPlugin.o
-L/usr/samples/internet_server/API -lhttpdapi
```

These commands create a shared library called `myPluginLibrary.so` from a single source file (myPlugin.cxx) using Edge Server Caching Proxy API files in the `/usr/samples/internet_server/API` directory.

11

Voice server

This chapter provides information on using the IBM WebSphere Voice Server with ViaVoice Technology Version 1.5 to deploy VoiceXML applications in a telephony environment. With these applications, the Voice Server provides voice access to Web-based application data using standard telephony interfaces.

This chapter introduces the system architecture needed to support Web-based voice applications in a deployment environment.

The Voice Server product is not included in WebSphere Everyplace Server 2.1; t must be obtained separately.

11.1 Voice Server overview

Until recently, the World Wide Web has relied exclusively on visual interfaces to deliver information and services to users via computers equipped with a monitor, keyboard, and pointing device. In doing so, a huge potential customer base has been ignored: people who (due to time, location, and/or cost constraints) do not have access to a computer.

Many of these people do, however, have access to a telephone. Providing conversational access (that is, spoken input and audio output over a telephone) to Web-based data will permit companies to reach this untapped market. Users benefit from the convenience of using the mobile Internet for self-service transactions, while companies enjoy the Web's relatively low transaction costs. And, unlike applications that rely on DTMF (telephone keypress) input, voice applications can be used in a hands-free or eyes-free environment, as well as by customers with rotary pulse telephone service or telephones in which the keypad is on the handset.

11.1.1 Typical application scenarios

Voice applications will typically fall into one of the following categories:

- Queries
- Transactions

Queries

In this scenario, a customer calls into a system to retrieve information from a Web-based infrastructure.

The system guides the customer through a series of menus and forms by playing instructions, prompts, and menu choices using prerecorded audio files or synthesized speech.

The customer uses spoken commands and/or DTMF (telephone keypress) input to make menu selections and fill in form fields.

Based on the customer's input, the system locates the appropriate records in a back-end enterprise database. The system presents the desired information to the customer, either by playing back prerecorded audio files or by synthesizing speech based on the data retrieved from the database.

Examples of this type of self-service interaction include applications or voice portals providing weather reports, movie listings, stock quotes, health care provider listings, and customer service information (Web call centers).

Transactions

In this scenario, a customer calls into a system to execute specific transactions with a Web-based back-end.

The system guides the customer to provide the data required for the transaction by playing instructions, prompts, and menu choices using prerecorded audio files or synthesized speech. The customer responds using spoken commands and/or DTMF (telephone keypress) input.

Based on the customer's input, the system conducts the transaction and updates the appropriate records in a back-end enterprise database. Typically the system also reports back to the customer, either by playing back prerecorded audio files or by synthesizing speech based on the information in the database records.

Examples of this type of self-service interaction include applications or voice portals for employee benefits, employee timecard submission, financial transactions, travel reservations, calendar appointments, electronic relationship management, sales automation, and order management.

11.1.2 VoiceXML

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 (`http://www.voicexml.org`), of which IBM is a founding member. It has been accepted for submission by the World Wide Web Consortium (W3C) as a standard for voice markup on the Web.

The VoiceXML language enables Web developers to use a familiar markup style and Web server-side logic to deliver voice content over the Internet. The resulting VoiceXML applications can interact with your existing back-end business data and logic.

Using VoiceXML, application developers can create Web-based voice applications that users can access by telephone.

VoiceXML supports dialogs that feature:

- Recognition of spoken input
- DTMF (telephone key) input
- Recording of spoken input
- Synthesized speech output ("text-to-speech")

- Pre-recorded digitized audio output
- dialog flow control
- Scoping of input
- Automatic Number Identification (ANI)
- Dialed Number Identification Service (DNIS)
- Call Transfer

11.1.3 Components of the Voice Server

The Voice Server includes:

- A VoiceXML browser that interprets VoiceXML markup.
- A connection environment that uses VoIP (H.323 V2.0) protocols to manage telephone calls and process telephony audio.
- IBM ViaVoice Speech Recognition and Text-To-Speech engines to accept voice input and generate synthesized speech output.
- Telephony-specific acoustic models for accurate speech recognition over telephone lines.
- A System Management component that performs basic system management functions.
- A sample VoiceXML application to test your installation and configuration.
- User documentation.

11.2 System architecture

The Voice Server deployment environment supports Web-based voice applications written in VoiceXML. This chapter describes a typical deployment configuration that may include the following:

- Telephony Infrastructure
- Voice Server Components
- Web Application Server
- Enterprise Server

Figure 11-1 on page 405 shows the overall system. The VoiceXML browsers are installed on the Voice Server. Your configuration may include one or more Voice over IP (VoIP) gateways communicating with one or more Voice Servers.

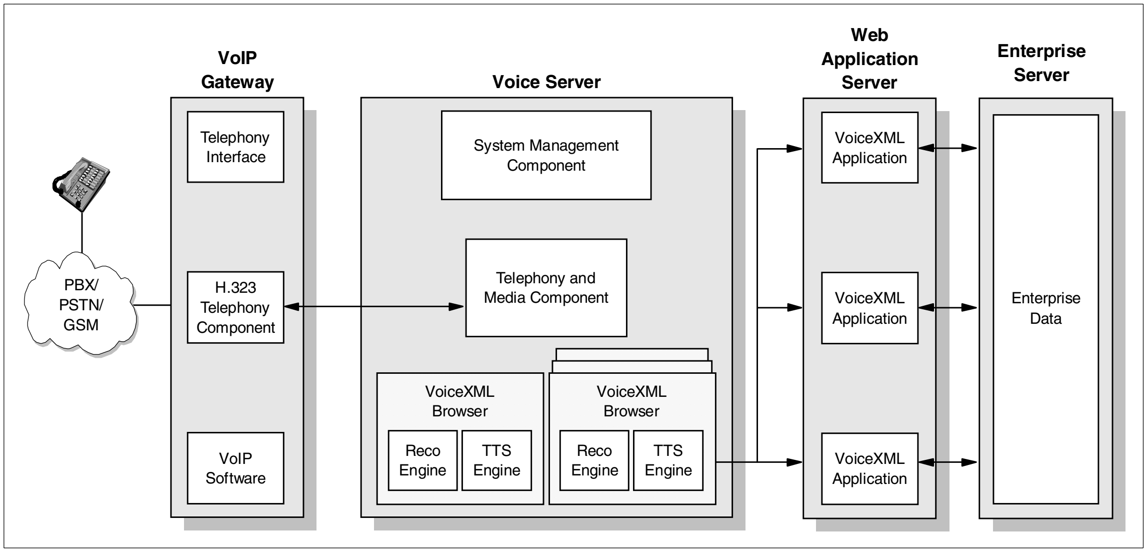


Figure 11-1 Overall System Architecture

11.2.1 Telephony infrastructure

The telephony piece of the deployment environment consists of one or more VoIP gateways that provide a connection between Voice Servers in an Internet Protocol (IP) network and the regular telephone lines coming in through a Public Switched Telephone Network (PSTN), Global System for Mobile Communication (GSM), or a Private Branch Exchange (PBX) network.

Voice over IP (VoIP) gateway

The VoIP gateway provides a connection to set up calls and manage audio data flow between a telephone and VoiceXML browsers running on a Voice Server that resides on an IP network. The gateway includes an H.323 telephony component that communicates with the Voice Server to:

- Provide the appropriate conversion from data over Integrated Services Digital Network Primary Rate Interface (ISDN PRI) T1 or E1, or Channel Associated Signalling (CAS) T1 lines into data packets for the IP network.
- Route a call to an available VoiceXML browser in the IP network, as specified by the call distribution scheme configured on the gateway.
- Pass any ANI (Automatic Number Identification) or DNIS (Dialed Number Identification Service) information received from the central office through to the Voice Server.
- Detect and deliver DTMF digits and voice audio to the VoiceXML browser.

- Provide load balancing among multiple Voice Servers.

The Voice Server requires an H.323 Version 2.0 compliant VoIP gateway. The VoIP gateway must support:

- G.711 uncompressed audio standard
- DTMF detection and extraction (in-band and out-of-band)

Telephony features

The Voice browser supports the following telephony features for use in a deployment environment.

Automatic Number Identification

The ANI function requires an ISDN PRI trunk with Q931 call signalling protocol. If available from your central office, you may substitute a CAS T1 line that supports ANI.

You can use ANI information to personalize the caller's experience by coding your VoiceXML application to:

- Determine who the caller is and whether they are using the system for the first time.
- Greet the caller by name.
- Log when various customers are most likely to call.

Dialed Number Identification Service (DNIS)

The DNIS function requires an ISDN PRI trunk with the Q931 call signalling protocol. If available from your central office, you may substitute a CAS T1 line that supports DNIS.

You can use the DNIS information to:

- Distribute incoming calls based on the telephone number that the user dialed (that is, assign a different telephone number to each application). This is useful for voice portals.
- Provide one telephone number and distribute incoming calls among multiple copies of your VoiceXML application running on one or more Voice Servers.
- Direct all incoming telephone calls to a top-level VoiceXML application, which routes the calls to the appropriate sub-application based on the DNIS number, either by transferring the call to another VoiceXML browser or to a human, or by requesting a specific VoiceXML page.
- Eliminate the first application decision point by assigning a separate telephone number to each choice on what had been your top-level menu.

- Track responses to advertising. You can list different telephone numbers in different ads, and then track the response rates based on which number the user dialed.

Call transfer

Typically, you would use the call transfer feature to transfer a call to:

- A human operator
- Another VoiceXML application

This release of the Voice Server supports only “blind transfer without dialing rules and without status.” The VoiceXML browser initiates the call transfer request and then disconnects from the call; the system cannot verify whether the telephone number for the call transfer has been correctly configured in the gateway. In addition, no information is returned on the status of the transfer, so there is no way to know whether the transfer was successful, the line was busy, the telephone number was invalid, or the call was dropped.

The call transfer function occupies two lines (ports) on the gateway while a call is connected. This is due to the implementation of the transfer as a call forward Q931 message which uses two ports on the gateway. The lines are freed up when the caller disconnects.

The configuration for the transfer on the gateway must correspond to the telephone number in the application. For example, you may append a prefix (such as 0) to a telephone number in order to match an exact configuration for a call leg on the gateway. You cannot transfer a call to an arbitrary number; you must match a number that is configured on the gateway.

11.2.2 WebSphere Voice Server components

The IBM WebSphere Voice Server Version 1.5 consists of the following components:

- H.323 Telephony Component
- System Management Component
- VoiceXML browser
- IBM ViaVoice Text-To-Speech engine
- IBM ViaVoice Speech Recognition engine

H.323 telephony component

The telephony environment includes an H.323 telephony component on the Voice Server that interacts with the H.323 telephony component on the VoIP gateway. The H.323 telephony component routes call information between the telephony connection and a VoiceXML browser running on the Voice Server.

Call processing

Each call is processed as follows:

1. The end user dials the telephone number of the Voice Server. The call travels through the PBX, PSTN, or GSM environment.
2. The Voice Server communicates with the H.323 telephony component to accept or terminate the telephone call.
3. The VoIP gateway converts the actual voice data stream into packets, translates the dialed number (DNIS) to a specific Voice Server IP address, and routes the call appropriately.
4. Telephone audio is received from the H.323 telephony component and streamed to the IBM ViaVoice Speech Recognition engine for processing. The results are used within the VoiceXML application to fill in forms or select menu items.
5. DTMF input is received from the H.323 telephony component.
6. The call is received by a Voice Server, which routes it to a registered VoiceXML browser that is available and has been configured to handle calls from that DNIS number.
7. When the VoiceXML browser encounters an element that requires speech output, it sends telephony audio to the H.323 telephony component. The audio may be generated from the IBM ViaVoice Text-To-Speech engine, recorded audio stored in an external file, or audio that was recorded within the VoiceXML application itself via the VoiceXML <record> element.
8. The VoiceXML browser converts the audio into packets, which it sends to the VoIP gateway. The gateway, in turn, converts the audio data to voice data and delivers it to the end user.
9. If the VoiceXML browser encounters a <transfer> element, it initiates a call transfer and then exits. However, the line is not freed up until the caller hangs up from the transferred connection.
10. The call ends when the user hangs up or the VoiceXML application is terminated.

System management component

The System Management component is responsible for starting, stopping, and monitoring VoiceXML browsers. There is one System Management component per Voice Server, and it controls all VoiceXML browser sessions running on that server. The System Management component also provides logging and error data in the SM log and trace files.

After the system reboots and the IBM WebSphere Voice Server NT service is running, you can use the System Management component to start the VoiceXML browsers.

VoiceXML browser

The VoiceXML browser is the implementation of the interpreter context. The VoiceXML browser is responsible for handling telephone calls and managing the dialog with the caller. It processes the calls, fetches VoiceXML documents from a location specified by the application (that is, a URI on a Web application server or local disk), and parses and interprets the documents. The VoiceXML browser uses the VoIP protocol (H.323 Version 2.0) connection environment for managing telephone calls and processing telephony audio.

Interaction with the system management component

The System Management component is responsible for starting and stopping the VoiceXML browsers. You can configure a Voice Server to run one or more VoiceXML browser instances simultaneously; each VoiceXML browser processes a single telephone call at a time.

When a VoiceXML browser starts, it requests an initial VoiceXML document that specifies an interaction (or “dialog”) between the application and the end user. The VoiceXML browser interprets and renders the document, managing the dialog with the user by playing audio prompts (using text-to-speech or recorded audio), accepting user voice and DTMF inputs, and acting on those inputs. Each of these activities changes the state of the dialog.

When a dialog does not specify a transition to another dialog, or when the caller hangs up, the VoiceXML browser exits and the session terminates. The System Management component then restarts the VoiceXML browser, which waits for another telephone call.

Interaction with the Text-to-Speech and Speech Recognition engines

During initialization, the VoiceXML browser starts the text-to-speech (TTS) and Speech Recognition engines.

As the VoiceXML browser processes a VoiceXML document, it plays audio prompts using text-to-speech or recorded audio; for text-to-speech output, it interacts with the TTS engine to convert the text into audio.

Based on the current dialog state, the VoiceXML browser enables and disables speech recognition grammars. When the VoiceXML browser receives user audio input, the Speech Recognition engine decodes the input stream, checks for valid user utterances as defined by the currently active speech recognition grammar(s), and returns the results to the VoiceXML browser. The VoiceXML browser uses the recognition results to fill in form items or select menu options in the VoiceXML application. If the input is associated with a <record> element in the VoiceXML document, the VoiceXML browser stores the recorded 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.

Interaction with the Web server and enterprise data server

You can publish your VoiceXML applications to any Web application server running on any platform; however, we recommend the IBM WebSphere Application Server Version 3.5 or later, which has been enhanced to receive VoiceXML pages published from IBM WebSphere Studio Version 3.5 or later.

When the VoiceXML browser starts, it sends an HTTP request over the LAN or Internet to request 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 an enterprise database using the same type of server-side logic (CGI scripts, JavaBeans, ASPs, JSPs, Java servlets, etc.) that you use to generate dynamic HTML documents.

The VoiceXML browser uses the local file system to cache documents fetched from the server. The cache is shared between all VoiceXML browsers running on the same Voice Server.

The VoiceXML browser interprets and renders the document. Based on the user’s input, the VoiceXML browser may request a new VoiceXML document from the Web server, or may send data back to the Web server to update information in the back-end database. The important thing is that the mechanism for accessing your back-end enterprise data does not need to change; your VoiceXML applications can access the same information from your enterprise servers that your HTML applications do.

IBM ViaVoice text-to-speech engine

The IBM ViaVoice Text-To-Speech engine allows ASCII text to be converted to synthesized speech. The text source may come from prompts within a VoiceXML application or from data that is retrieved from a database.

IBM ViaVoice Speech Recognition engine

The VoiceXML browser uses the IBM ViaVoice Speech Recognition engine to recognize voice input from a user responding to prompts generated by a VoiceXML application. The recognition results are used within the VoiceXML application to fill in form items or select a menu item.

The IBM ViaVoice Speech Recognition engine uses telephony acoustic models for accurate speech recognition over telephone lines.

The high-level process looks like this:

1. During application development, the developer creates a series of speech recognition grammars defining the words and phrases that can be spoken by the user, and specifying where each grammar should be active within the application.
2. When you start the Voice Server, each VoiceXML browser starts an instance of the IBM ViaVoice Speech Recognition and Text-To-Speech engines.
3. 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.
4. The Speech Recognition engine compares the sounds to the list of words and phrases in the currently active grammar(s). Only words and phrases in the active grammars are considered as possible speech recognition candidates.
5. The Speech Recognition engine returns the results to the VoiceXML browser.
6. The VoiceXML browser uses the results within the VoiceXML application to fill in form fields or select menu items, thereby managing the dialog with the user.

11.2.3 Web application server

Typically, your VoiceXML application files will reside on a Web application server.

The Web application server can use server-side programs to locate or update records in a back-end enterprise database and return the information to the VoiceXML browser.

11.2.4 Enterprise server

VoiceXML applications can access existing enterprise data from your enterprise servers in the same manner as HTML applications. A requested document can be generated dynamically from data stored in the enterprise database using the same types of server-side logic (CGI scripts, Java Beans, JSPs, Java Servlets, etc.) that you use to generate HTML applications.

11.3 Integration with WebSphere Everyplace Server

Voice Server may be integrated with WebSphere Everyplace Server through the Edge Server Caching Proxy. The scenario of Voice Server integrated with Everyplace Server is shown in Figure 11-2.

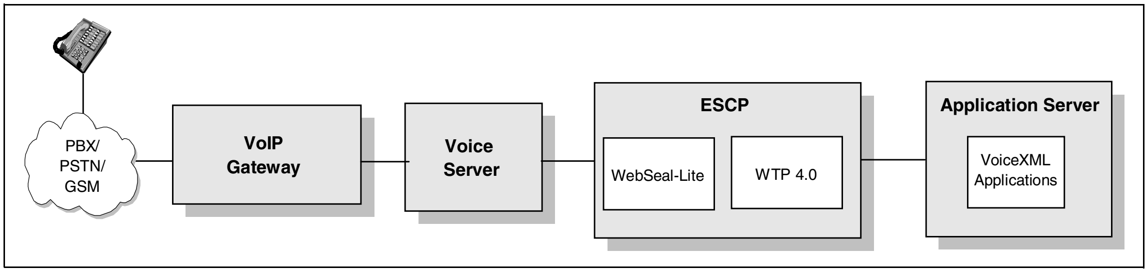


Figure 11-2 Voice Server and WebSphere Everyplace integration

In this scenario Voice Server is connected to Edge Server Caching Proxy and uses HTTP as the communication protocol. WebSEAL-Lite and WebSphere Transcoding Proxy are deployed as plug-ins of Edge Server Caching Proxy.

The current WebSphere Transcoding Publisher Version 3.5 included in WebSphere Everyplace Server does not support Voice Server; however, WebSphere Transcoding Publisher V4.0 has been enabled to support Voice Server. The test of the integration between Voice Server and WebSphere Transcoding Publisher V4.0 is already done.

11.3.1 Execution example

This is an example of integration between Voice Server and WebSEAL-Lite running as a Edge Server Caching Proxy plug-in.

The Voice browser for an application will send an initial request to download required resources. This request cannot be challenged even though it is going to the same URL that must be protected when an incoming call arrives. The foundation for Voice support is ensuring that users of voice applications are integrated with WebSphere Everyplace Server SecureWay Directory and authentication.

Everyplace Server should supply a sample VoiceXML application with a login form and a corresponding form interpretation plug-in. The sample Edge Server Caching Proxy configuration files should contain example mapping statements for the sample Voice application.

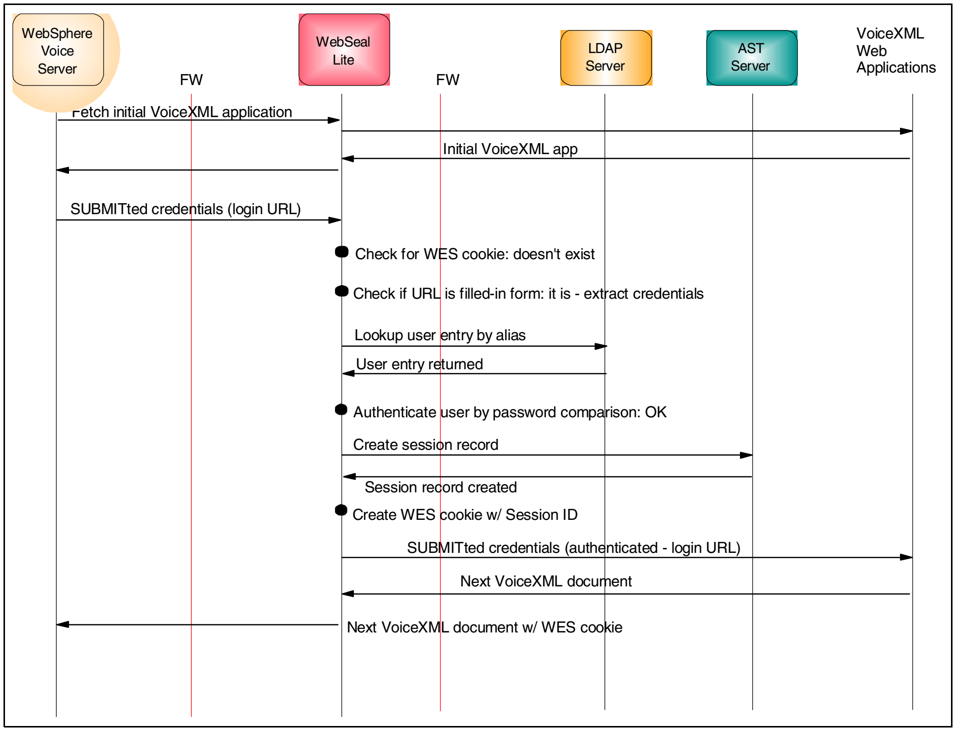

WebSphere Voice Server
FW
WebSeal Lite
FW
LDAP Server
AST Server
VoiceXML Web Applications
Fetch initial VoiceXML application
Initial VoiceXML app
SUBMITted credentials (login URL)
Check for WES cookie: doesn't exist
Check if URL is filled-in form: it is - extract credentials
Lookup user entry by alias
User entry returned
Authenticate user by password comparison: OK
Create session record
Session record created
Create WES cookie w/ Session ID
SUBMITted credentials (authenticated - login URL)
Next VoiceXML document
Next VoiceXML document w/ WES cookie

Figure 11-3 Voice Server application execution example

This flow shown in Figure 11-3 assumes a single initial VoiceXML application that contains the form for logging in callers. First, the Voice Server loads the initial VoiceXML application from the Web server. The application's URL is configured as public on the WebSEAL-Lite so that no authentication challenge occurs on this flow. At some later point, a caller calls in to the Voice Server and it maps the call to the VoiceXML application. The application collects and passes the caller's credentials on a form SUBMIT which includes a hidden field in the submitted form identifying it as a WebSEAL-Lite login form. The same URL is also flagged as a form that contains a user alias that must be looked up in the SecureWay Directory to get the Everyplace Server user entry. WebSEAL-Lite extracts the credentials from the form and then calls SecureWay Directory to get the Everyplace Server user entry corresponding to the alias. WebSEAL-Lite compares the password with the user password from SecureWay Directory.

Part 5

Appendixes

In these Appendixes you will find information regarding installation issues that affect architectural decisions, and tables containing devices/platforms/protocols supported by WebSphere Everyplace Server.

A

Installation notes

The Setup Manager for IBM WebSphere Everyplace Server Service Provider Offering V2.1 documents the silent install well; however, we have provided a setup sequence for you to follow, so you will have a better understanding of how the installation takes place.

Install SecureWay Directory Server

Always install the IBM SecureWay Directory, because IBM WebSphere Everyplace Server requires a SecureWay Directory for all of its components.

Install Tivoli Personalized Services Manager

A database management system is required. If you decide to use Oracle, you must have the database installed *before* the installation of Tivoli Personalized Services Manager, or else IBM DB2 will automatically be installed.

Installation with Setup Manager

During the silent install, the following products will be installed:

- IBM HTTP server
- DB2 Universal Database (if not installed)
- MQseries Everyplace
- WebSphere Application Server, Advanced Edition
- WebSphere Edge Server (Both caching proxy and Load Balancer)
- Everyplace Wireless Gateway
- WebSphere Transcoding Publisher
- Tivoli Personalized Services Manager (The default Web application server is TomCat, but a migration script to WebSphere Application Server will be available through the IBM Web site)
- Everyplace Active Session Table
- Everyplace Cookie Proxy (only on Japanese locales)
- Everyplace Intelligent Notification services
- Everyplace Location Based Services
- Everyplace Synchronization Manager Voice Services
- WebSEAL-Lite

Setup manager tips

- Setup manager will not install Tivoli Secureway Policy Director. This product needs to be installed manually.

- The installation panel will disappear on Solaris systems if you click the background area outside of the install console. Press ALT+TAB to bring the panel back to the foreground.
- Setup manager will prompt users to enter configuration data with dialog boxes. If the setup manager seems to stop responding or installing, check if a dialog box has opened behind another window.
- The Insert Next CD dialog might not appear on slow X-Window servers, or on a slow connection to an X-Windows server.

DB2 Installation errors

The installation of the NetQuestion SBCS search engine 1.2.3.1 produces an error because IBM WebSphere Everyplace Server does not use this package; however, the error message can be ignored.

B

Devices, platforms, and protocols

The following tables detail the devices, platforms, and protocols supported by Everyplace Server.

Supported networks and platforms

Table 11-1 lists by Everyplace Server component the platforms and operating systems supported.

Table 11-1 Platforms supported by Everyplace Server

Everyplace Server component	Platforms supported
Everyplace Synchronization Manager	► Palm OS ► Windows CE ► Pocket PC ► EPOC (in the future)
Everyplace Wireless Gateway	► Windows CE 2.0 & 2.1 ► Windows CE (PPC) ► Windows CE V3.0 ► Windows 95 & 98 ► Windows 2000 ► Palm OS ► Pre- EPOC WAP ► QNX/Neutrino
MQSeries Everyplace	► Windows CE ► Windows 95 & 98 ► Windows NT & 2000 ► Palm OS ► EPOC ► Devices running a Java JVM 1.1 or later
Tivoli Personalized Services Manager (Device Manager)	► Windows CE ► Palm OS ► QNX/Neutrino ► NetVista Internet Appliance
WebSphere Transcoding Publisher	► Windows CE ► Windows 95 & 98 ► Windows NT & 2000 ► Palm OS ► EPOC

Mobile network connections

Table 11-2 lists the MNC types per wireless link protocol or PPP connection supported by Everyplace Server.

Table 11-2 Mobile Network Connection (MNC) for WLP or PPP connections

MNC types	Description	Install this network support
ardis-tcp	ARDIS Standard context routing (SCR) using TCP-Motient owns and operates the ARDIS network	Ardis
ardis-x25	ARDIS SCR using X.25 -Motient owns and operates the ARDIS network	Ardis
dataradio-bdic	Dataradio base station data link controller	Dataradio
dataradio-msc	Dataradio multi-site controller (MSC)	Dataradio
datatac 5000	DataTAC 5000 using X.25	DataTAC
datatac 6000	DataTAC 6000 using TCP	DataTAC
dial-isdn	Integrated services digital network, including native PPP connections	Dial
dial-pstn	Public switched telephone network, including native PPP connection	Dial
dial-tcp	Dial connection through an IP-attached modem server, including native-PPP connections	Dial
ip-lan	IP-based network, such as CDPD, frame relay, connection with internet service provider (ISP), or LAN	IP LAN

MNC types	Description	Install this network support
mobitex	Mobitex international standard connection using X.25	Mobitex
mobitex-tcp	Mobitex using TCP, such as Mobitex Internet application server (IAS)	Mobitex
modacom-scr	Modacom SCR using X.25	Modacom
mc-3000	Radio network controller 3000	RNC-3000

Table 11-3 lists the available MNC types for supporting wireless application protocol (WAP) connections.

Table 11-3 Available MNCs for WAP connections

MNC type	Description	Install this network support
ardis-tcp	ARDIS standard content routing (SCR) using TCP -owned and operated by Motient	Ardis
ardis-x25	ARDIS SCR using X.25 -Motient owns and operates the ARDIS network	Ardis
dial-isdn	Integrated services digital network, only using native -ppp connections	Dial
dial-pstn	Public switched telephone network, only using native PPP connections	Dial
dial-tcp	Dial connection through an IP-attached modem server, including native-PPP connections	Dial

MNC type	Description	Install this network support
ip-wdp	IP/UDP bearer adapter using wireless datagram protocol	Installed automatically with the Wireless Gateway
mobitex	Mobitex international standard connection using X.25	Mobitex
mobitex-tcp	Mobitex using TCP, such as Mobitex Internet application server (IAS)	Mobitex
sms-smpp	Short message service using universal computer protocol/external machine	SMS
sms-ucp	Short message service using short message peer to peer protocol/external machine interface (UCP/EMI)	SMS
sms-smpp-x.25	Short message service using SMPP Version 3.4 over X.25	SMS
sms-ucp-x.25	Short message service using UCP/EMI over X.25	SMS
smtp	Simple mail transport protocol, as specified in RFC 821. Note that extensions to the RFC, such as MIME support or mail server authentication, are not supported.	SNPP
snpp	Simple network paging protocol, as specified in RFC 1861. Note that all function of level one and required elements from level two are supported. SNPP level 3 is not supported	SNPP

Table 11-4 lists the available MNC type for a variety of messaging types.

Table 11-4 Available MNCs for Messaging Connections

MNC type	Description	Install this network support
ip-wdp	IP/UDP bearer adapter using wireless datagram protocol	Installed automatically with the Wireless Gateway
mobitex	Mobitex International standard connection using X.25	Mobitex
mobitex-tcp	Mobitex using TCP, such as Mobitex Internet Application Server (IAS)	Mobitex
sms-smpp	Short message service using short message peer to peer protocol (SMPP) V 3.4	SMS
sms-ucp	Short message service using universal computer protocol/external machine interface (UCP/EMI)	SMS
sms-smpp x.25	Short message service using SMPP version 3.4	SMS
sms-ucp-x25	Short message service using UCP/EMI over X.25	SMS
smtp	Simple mail transport protocol, as specified in RFC 821. Note that extensions to the RFC, such as MIME support or mail server authentication are not supported	SMTP
snpp	Simple Network paging protocol, as specified in RFC 1861. Note that all function of level one and required elements from level two are supported. SNPP level 3 is not supported.	SNPP

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 429.

- *An Introduction to IBM WebSphere Everyplace Suite Version 1.1 Accessing Web and Enterprise Applications,* SG24-5995
- *Extending e-business to Pervasive Computing Devices Using IBM WebSphere Everyplace Suite Version 1.1.2*, SG24-5996
- *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5 Extending Web Applications to the Pervasive World*, SG24-6233
- *Enterprise Wireless Applications using WebSphere Everyplace Server Offerings*, SG24-6519
- *Tivoli SecureWay Policy Director Centrally Managing e-business Security*, SG24-6008
- *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
- *Capabilities in IBM WebSphere Transcoding Publisher Version 3.5 Extending Web Applications to the Pervasive World*, SG24-6233
- *Understanding LDAP*, SG24-4986
- *IBM WebSphere Performance Pack: Caching and Filtering with IBM Caching Proxy,* SG24-5859
- *The Semi-Walled Garden: Japan's "i-mode Phenomenon",* REDP0166

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

- *Network Dispatcher Administration Guide,* GC31-8496
- *WebSphere Edge Server Caching Proxy (Web Traffic Express) User's Guide* - shipped with the product
- *WebSphere Transcoding Publisher Developer's Guide*

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 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 DoCoMo 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/`

- Netegrity, 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 & Scalable LDAP Cluster in an IBM AIX Environment”

  `http://www-1.ibm.com/servers/esdd/articles/ldap/index.html`

- IBM Network Dispatcher Version 3.0 Scalability, Availability and Load Balancing for TCP/IP applications

  `ftp://ftp.software.ibm.com/software/network/dispatcher/whitepapers/nd30whitepaper.pdf`

- 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.

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

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

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation

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 codec 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. (1) 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. Mobile 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 life cycle 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 Notification Services
IP	Internet Protocol
ISDN	Integrates 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: A Guide for Architects and Integrators

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

IBM WebSphere Everyplace Server:

A Guide for Architects and Integrators

Business-to-consumer and business-to-employee examples

How to plan for scalable and high-availability solutions

Integrate voice and third-party wireless gateways

This book is essential reading for the information technology (IT) architect or system integrator who is responsible for designing a successful mobile e-business solution based on IBM WebSphere Everyplace Server V2.1.1.

An introduction to IBM WebSphere Everyplace Server and a product differentiation between the Service Provider Offering and Enable Offering is provided for those less familiar with the IBM WebSphere Everyplace Server offerings.

The reader will learn how information flows through the IBM WebSphere Everyplace Server solution, including information on how data is passed from one component to the other. This information is then illustrated using two business case scenarios, a business-to-employee scenario highlighting WebSphere Everyplace Server Enable Offering, and a business-to-consumer scenario highlighting WebSphere Everyplace Server Service Provider Offering.

Readers will also find planning considerations for scalable and high-availability solutions. Security issues are covered as well as information necessary for integrators interested in deploying or to provide voice access to Web-based applications using standard telephony interfaces and IBM WebSphere Voice Server.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6189-00 ISBN 0738424234